Gustavo Ribeiro da Costa Alves

Projecto para o Teste e Depuração com base nas Arquitecturas 1149.1 e P1149.4

Dissertação submetida para a obtenção do grau de Doutor
em Engenharia Electrotécnica e de Computadores

Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia Electrotécnica e de Computadores

Abril de 1999

Tese realizada sob supervisão do

Prof. Dr. José Manuel Martins Ferreira

Professor Auxiliar do

Departamento de Engenharia Electrotécnica e de Computadores da

Faculdade de Engenharia da Universidade do Porto

Aos meus Pais.

Labor parit laborem.

Par est fortuna laboris.

Resumo

Palavras chave: *projecto para o teste e depuração, IEEE 1149.1, IEEE P1149.4, instruções opcionais, infraestrutura para o teste e depuração, controlador residente, metodologia e geração automática do programa de teste e depuração.*

Nesta dissertação descreve-se uma solução de projecto para o teste e depuração, baseada na reutilização de infraestruturas de teste compatíveis com a norma IEEE 1149.1 e com a proposta de norma P1149.4. Os modos de operação básicos e opcionais destas infraestruturas permitem apenas implementar um subconjunto das operações de depuração identificadas no segundo capítulo. Esta conclusão decorre da análise da utilização destas infraestruturas para a implementação de um conjunto de operações de depuração, que inclui: operações de controlo, observação e verificação de estados; operações passo-a-passo; operações de pontos de paragem por condição; e operações de análise em tempo real. As principais lacunas situam-se na detecção em tempo real de condições de paragem, na aquisição em tempo real e na detecção de tempos de atraso. Para colmatar estas lacunas, propõe-se um conjunto de novas instruções opcionais (IEEE 1149.1, P1149.4), que implicam um conjunto reduzido de alterações na respectiva infraestrutura de teste mínima. O suporte deste conjunto de instruções opcionais justifica a expressão *infraestrutura para o teste e depuração*.

As operações de depuração que não dependem unicamente da infraestrutura proposta são implementadas através de um controlador residente. A arquitectura deste controlador assenta em dois processadores, um para controlar as acções efectuadas através da infraestrutura para o teste e depuração e o outro para controlar as acções da lógica funcional do sistema sob depuração, através do controlo do fornecimento de impulsos de relógio. Esta solução garante o sincronismo entre a lógica de teste e a lógica funcional. Ambos os processadores possuem um interface com memórias de dados externas, para armazenamento de informação referente à execução do programa de teste e depuração. O controlador possui ainda um conjunto de canais de sincronismo, que lhe permitem interagir com um equipamento de teste exterior.

Define-se ainda uma metodologia de utilização da infraestrutura proposta e do controlador, de acordo com o tipo e nível hierárquico do sistema sob depuração, sendo considerados os níveis do circuito integrado e da carta de circuito impresso. Em relação ao tipo, consideram-se sistemas baseados em lógica dedicada, sistemas baseados em microprocessadores e sistemas híbridos. A metodologia definida serve de suporte à criação do programa executado pelo controlador, que é gerado automaticamente por uma ferramenta computacional a partir de um conjunto de informação de entrada, que vai sendo disponibilizada à medida que avança o projecto do sistema: descrição da infraestrutura de teste e depuração; resultado da simulação do sistema; descrição das cadeias de varrimento internas dos circuitos integrados; descrição das ligações entre os componentes da carta de circuito impresso; vectores para o teste estrutural; e opções introduzidas pelo utilizador. A disponibilização do modelo do controlador permite a simulação da implementação das operações de depuração, mesmo quando não existe ainda qualquer tipo de protótipo físico do sistema em desenvolvimento.

Abstract

Keywords: *design for debug and test, IEEE 1149.1, IEEE P1149.4, optional instructions, debug and test infrastructure, built-in controller, automatic debug and test program generation, debug and test methodology.*

This dissertation describes a design for debug and test solution, based on the re-use of IEEE 1149.1-compatible, and P1149.4-compatible test infrastructures. The basic and optional operating modes of these infrastructures allow the implementation of a subset of the debug operations identified in chapter 2. This conclusion arises from the analysis on the use of such test infrastructures for implementing a set of debug operations, that includes: state control, observation and verification operations; single-step operations; breakpoint operations; and operations on real-time analysis. The main gaps lay on breakpoint real-time detection, real-time acquisition, and delay fault detection. A new set of optional instructions (for IEEE 1149.1, P1149.4) is proposed, to fill these gaps, implying a reduced number of alterations in the minimum test infrastructure. This set of optional instructions justifies the expression *debug and test infrastructure*.

A built-in controller implements the debug operations that do not depend entirely on the proposed infrastructure. The controller architecture is based on two processors, one for controlling the actions performed through the debug and test infrastructure, and the other for controlling the functional logic of the system under debugging, by controlling the supply of clock pulses. This type of solution guarantees the synchronism between the test logic and the functional logic. Both processors have an interface with external data memories, for storing information related to the debug and test program execution. A set of synchronism channels allows the controller to interact with external test equipment.

We also define a methodology for using the proposed debug and test infrastructure and the built-in controller, according to the type and hierarchical level of the system under debugging. We consider the integrated circuit level and the printed circuit board level. As for the system type, we consider three possible types: ASIC-based systems, microprocessor-based systems and hybrid systems. The defined methodology also supports the generation of the program executed by the built-in controller. An automatic generation tool, which uses information that is progressively released as the system design evolves, performs this task. The input information includes: the debug and test infrastructure description file; the system simulation results file; the internal scan-chains description files; the netlist; the vectors for the structural test (imported from an automatic test pattern generation tool); and the user options. The implementation of the debug operations may be simulated, using a model of the built-in controller, even before the first physical prototypes are released.

Résumé

Mots clé: *conception en vue du test de conception et de fabrication, IEEE 1149.1, IEEE P1149.4, instructions optionelles, infrastructure pour le test de conception et fabrication, contrôleur intégré, méthodologie et génération automatique de programmes de test de conception et fabrication.*

Dans cette thèse on decrit une solution pour la conception en vue du test de conception et de fabrication, basée sur la reutilisation d'infrastructures de test compatibles avec la norme IEEE 1149.1 et avec la proposition de norme P1149.4. Les modes d'opération de base de ces infrastructures et ceux optionels ne permettent que d'implémenter un sous-ensemble d'opérations de test de conception identifiées dans le deuxième chapître. Cette conclusion parvient de l'analyse de l'utilisation de ces infrastructures pour l'implémentation d'un ensemble d'opérations de test de conception, inclus: opérations de contrôle, observation et vérification d'états; opérations pas-à-pas; opérations de points d'arrêt conditionnels; et opérations d'analyse en temps réel. Les problèmes les plus importants se situent dans la détection en temps réel de conditions d'arrêt, dans l'aquisition en temps réel et dans la détection de temps de retard. Afin de surmonter ces problèmes, on propose un ensemble de nouvelles instructions optionelles (IEEE 1149.1, P1149.4), qui impliquent dans un ensemble réduit de changements dans l'infrastructure minimum de test associée. Le support de cet ensemble d'instructions optionelles justifie l'expression *infrastructure pour le test de conception et de fabrication*.

Les opérations de test de conception qui ne dépendent pas uniquement de l'infrastructure proposée sont implementées par un contrôleur intégré. L'architecture de ce contrôleur est basée sur deux processeurs, l'un pour contrôler les actions efectuées par l'infrastructure de test de conception et fabrication et l'autre pour contrôler les actions de la logique fonctionnelle du système sous test, atravers le contrôle des pulses d'horlôge. Cette solution garanti le synchronisme entre la logique de test et la logique fonctionnelle. Les deux processeurs possèdent une interface avec mémoires de données externes, pour le stockage d'information associée a l'exécution du programme de test de conception et fabrication. Le contrôleur possède un ensemble de chaînes de synchronisme, qui permètent d'interagir avec l'équipment de test extérieur.

On définit aussi une méthodologie d'utilisation de l'infrastructure proposée et du contrôleur, conformement le type et niveau hiérarchique du système sous test, tout en considérant les niveaux du circuit intégré et de la carte. Par rapport au type, on considère les systèmes basées sur logique dédiée, les systèmes basées sur microprocesseurs et les systèmes hybrides. La méthodologie définie sert de support à la création du programme à exécuter par le contrôleur, ceci étant géneré automatiquement par un outil CAO à partir d'un ensemble d'information d'entrée qui est disponibilisé au fur et à mesure que la conception du système avance: la description de l'infrastructure de test de conception et fabrication; le résultat de la simulation du système; la description des chaînes de décalage internes de circuits intégrés; la description des interconnections entre les composants de la carte; les vecteurs pour le test structurel; et les choix introduit par l'utilisateur. La disponibilisation du modèle de contrôleur permet la simulation de l'implémentation des opérations de test, même avant qu'il n'y ait pas de prototype physique du système en dévelopment.

Agradecimentos

A realização deste trabalho não teria sido possível sem a colaboração de várias pessoas que ao longo dos últimos anos me têm ajudado, apoiado e incentivado com dedicação e amizade. Embora correndo o risco de inadvertidamente omitir alguém, gostaria de aproveitar esta ocasião para reconhecer as seguintes pessoas e instituições.

Começo por agradecer ao ISEP a oportunidade concedida para efectuar este trabalho e em especial ao Engº Mesquita Guimarães pela confiança depositada, sem a qual dificilmente teria iniciado este percurso. Aos colegas do grupo de CAD e Microelectrónica do INESC, pela paciência e pela compreensão perante as minhas constantes dúvidas no início deste trabalho. Em especial, gostaria de agradecer ao Professor José Machado da Silva pelo apoio e sugestões dadas. Aos colegas do grupo PTSE da FEUP, pela camaradagem e pelo excelente ambiente de trabalho. Deste grupo quero, de modo particular, destacar o Engº Manuel Gericota, pela revisão atenta deste texto, a Drª Inês Cambeiro, pela ajuda prestada em várias ocasiões, nomeadamente na elaboração do relatório sobre o estado da arte, o Engº Miguel Santiago, um verdadeiro *guru* da informática, e o Telmo Amaral pela ajuda preciosa nas várias simulações, traduzidas em imensas horas passadas em frente ao computador.

Ao Doutor Marcelo Lubaszewski pela ajuda na tradução do resumo para francês. *I am also grateful to Doutor Oystein Ra for the facilities provided at HiBU, and for arranging the interviews in Norway and Sweden.* Aos Monges Beneditinos do Mosteiro de Singeverga pelo ambiente ideal para a revisão final desta dissertação.

Quase a terminar, quero agradecer ao Professor José Manuel Martins Ferreira as inúmeras conversas e discussões que muito me ajudaram a trilhar o percurso certo e a dele não me afastar. Não posso, pois, deixar de expressar a minha profunda admiração e estima, que ao longo destes últimos anos se consolidou numa sincera e sã amizade.

Por último, e principalmente, gostaria de agradecer à minha família e aos meus amigos, que de uma forma despercebida me ajudaram. Em particular, à Ró, minha companheira de jornada, e à Diana e ao Tiago, nossos pequenos rebentos, a quem furtei muitas das devidas atenções.

Nota ao leitor

Ao longo da escrita de um documento técnico desta natureza, é frequente o conflito entre o emprego de termos estrangeiros e a clareza de exposição. Embora seja possível optar por uma estratégia global de tudo ou nada, ou seja, empregarem-se todos os termos originalmente apresentados numa língua estrangeira, ou traduzirem-se todos para português, não é claro que o resultado final favoreça aquilo que se pretende com este documento: a apresentação clara dos conhecimentos adquiridos e do trabalho efectuado. Opta-se, pois, por uma estratégia mista que privilegie a clareza de exposição. O conjunto de termos estrangeiros com utilização habitual na linguagem verbal é deste modo mantido, embora se favoreça a utilização de termos portugueses sempre que o seu emprego não suscitar dúvidas. Termos estrangeiros são sempre referidos em *itálico*, utilizando-se este mesmo estilo sempre que se pretenda realçar uma determinada palavra ou expressão em português. Os acrónimos, sejam em português, sejam em estrangeiro, são sempre apresentados em maiúsculas, em estilo normal. Uma lista de acrónimos é incluída para facilitar a sua identificação.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

A/D	Analógico / Digital
ABM	*Analog Boundary Module*
AHDL	*Altera Hardware Description Language*
ALU	*Arithmetic and Logic Unit*
ARM	*Advanced* RISC *Machines*
ASIC	*Application Specific Integrated Circuit*
ASP	*Addressable Scan Port*
ATAP	*Analog Test Access Port*
ATE	*Automatic Test Equipment*
ATPG	*Automatic Test Pattern Generation*
BCR	*Boundary Control Register*
BDM	*Background Debug Mode*
BF	*Bus-functional*
BGA	*Ball Grid Array*
BIST	*Built-in Self-Test*
BP	*Breakpoint*
BS	*Boundary Scan*
BSDL	*Boundary Scan Description Language*
BST	*Boundary Scan Test*
CAD	*Computer-Aided Design*
CCI	Carta de Circuito Impresso
CD-ROM	*Compact Disc – Read-Only Memory*
CI	Circuito Integrado
CLF	Controlo da Lógica Funcional
CLT	Controlo da Lógica de Teste
COTS	*Components Off-The-Shelf*
COV	Controlo, Observação e Verificação
CPLD	*Complex Programmable Logic Device*
CTLR	*Control Register*
D/A	Digital / Analógico
DfDT	*Design for Debug and Test*
DfT	*Design for Testability*
E/S	Entrada / Saída
ECD	Entrada de Condição Detectada
EQI	*Event Qualification Input*
EQM	*Event Qualification Module*
EQO	*Event Qualification Output*
EQR	*Event Qualification Register*
FCT	Fundação para a Ciência e Tecnologia
FEUP	Faculdade de Engenharia da Universidade do Porto

FF	*Flip-Flop*
FIB	*Focused-Ion-Beam*
FIFO	*First-In First-Out*
FIPB	*Firm* IP *Block*
FPGA	*Field-Programmable Gate-Array*
FPIC	*Field-Programmable Interconnect Chip*
GAPTD	Geração Automática do Programa de Teste e Depuração
HDL	*Hardware Description Language*
HIPB	*Hard* IP *Block*
HP	Hewlett-Packard
ICE	*In-Circuit Emulator*
IEEE	*Institute of Electrical and Electronic Engineers*
INESC	INstituto de Engenharia de Sistemas e Computadores
INSIGHT	*INtegrated Small Industries Goal-directed High technology Training*
IP	*Intellectual Property*
ISS	*Instruction-Set Simulation*
JNICT	Junta Nacional de Investigação Científica
JTAG	*Joint Test Action Group*
LFSR	*Linear-Feedback Shift Register*
LPE	*Layout Parameter Extractor*
LPM	*Library of Parameterized Modules*
LSI	*Large Scale Integration*
MCM	Multi-Chip Module
MEF	Máquina de Estados Finita
MSI	*Medium Scale Integration*
MTM	*Module Test and Maintenance* (*-bus*)
PBIC	Programa Base de Investigação Científica
PCI	*Programmable Clock Interface*
PMCT	Programa Mobilizador de Ciência e Tecnologia
PPA	*Program Performance Analysis*
PRPG	*Pseudo-Random Pattern Generation*
PSA	*Parallel Signature Analysis*
PTSE	Projecto e Teste de Sistemas Electrónicos
RAM	*Random Access Memory*
RAMR	RAM *Register*
RISC	*Reduced Instruction Set Computer*
ROM	*Read-Only Memory*
RTL	*Register Transfer Level*
SCD	Saída de Condição Detectada
SI	*Scan Input*
SIPB	*Soft* IP *Block*
SO	*Scan Output*
SP	Selecção de Pino
SS	*Single-Step*

STC	Selecção do Tipo de Condição
STIL	*Standard Test Interface Language*
SVF	*Serial Vector Format*
TAP	*Test Access Port*
TBC	*Test Bus Controller*
TBIC	*Test Bus Interface Circuit*
TCK	*Test Clock*
TDI	*Test Data In*
TDO	*Test Data Out*
TI	*Texas Instruments*
TIT	Tecnologias da Informação e Telecomunicações
TMS	*Test Mode Select*
TRST	*Test Reset*
TTL	*Transistor Transistor Logic*
VHDL	VHSIC *Hardware Description Language*
VHSIC	*Very High Speed Integrated Circuit*
VLC	Valor Lógico da Condição
VLSI	*Very Large Scale Integration*
VSI	*Virtual Sockets Interface*
ΣΔ	Sigma-Delta

Capítulo 1

Introdução

Este capítulo apresenta o tema e o enquadramento do trabalho realizado, incluindo a identificação dos aspectos de inovação e originalidade, a que se segue a descrição da forma como está organizada esta dissertação.

1.1 Tema e enquadramento

Entre os antecedentes deste trabalho, destacam-se o final de um projecto suportado pelo programa Leonardo, denominado *INtegrated Small electronic Industries Goal-directed High technology Training* (INSIGHT) e o final de um outro projecto suportado pelo Programa Mobilizador de Ciência e Tecnologia[1] (PMCT), denominado "Sistema de Validação e Teste de Cartas de Cartas de Circuito Impresso com BST". O primeiro destinava-se ao desenvolvimentos de cursos avançados na área do projecto de sistemas electrónicos, incluindo a matéria de projecto para a testabilidade. O segundo destinava-se ao desenvolvimento de um testador de baixo custo para Cartas de Circuito Impresso (CCI) que incluíssem componentes com uma infraestrutura de teste IEEE 1149.1 (*Boundary-Scan Test,* BST). Acresce-se aos factores anteriores a realização da tese de mestrado do autor neste mesmo domínio [Alv95], surgindo naturalmente este trabalho como uma continuação em face de novos desafios, nomeadamente no que respeita à reutilização das infraestruturas de teste normalizadas para a implementação de operações de depuração de protótipos (e eventualmente ainda em outras etapas do ciclo de vida de um produto, e.g. na sua manutenção).

[1] Da ex- Junta Nacional de Investigação Científica (JNICT), actual Fundação para a Ciência e Tecnologia (FCT).

1.1.1 Enquadramento proporcionado pelos projectos Leonardo INSIGHT II e JNICT PBIC/C/TIT/2474/95

O objectivo principal do projecto JNICT PBIC/C/TIT/2474/95 consistiu na criação de ferramentas de apoio ao desenvolvimento e depuração de sistemas que incluam componentes com BST. A solução adoptada divide-se em duas componentes principais: inserção no sistema de componentes cuja funcionalidade principal consiste em apoiar as operações de teste e depuração, e o desenvolvimento de aplicações para ambiente *Windows*® que permitem tirar partido dos recursos assim disponibilizados. Para demonstrar a viabilidade desta solução e a aplicabilidade das ferramentas desenvolvidas, construiu-se um sistema de demonstração baseado num microcontrolador da família 80C51. Dado que este microcontrolador não possui uma infraestrutura de teste, foi necessário adicionar ao sistema um conjunto de componentes comerciais com BST. Apesar de se terem atingido os objectivos inicialmente propostos, a quantidade de componentes adicionais, a área de CCI ocupada e o tempo necessário para implementar algumas das tarefas de depuração, levou a concluir pela necessidade de se explorarem outras formas de utilização da infraestrutura de teste, para os mesmos efeitos de depuração.

O objectivo principal do projecto Leonardo INSIGHT II consiste no desenvolvimento de um conjunto de cursos na área do projecto de sistemas electrónicos [Leo99]. Estes cursos cobrem três assuntos principais, nomeadamente:

- Modelação e síntese.
- Desenvolvimento e teste de sistemas digitais e mistos.
- Tecnologias de módulos multi-pastilha e micro-sistemas.

Cada assunto constitui um núcleo de desenvolvimento, sendo a Faculdade de Engenharia da Universidade do Porto (FEUP) responsável pela coordenação do segundo núcleo acima referido, formado por três módulos:

- Projecto para o teste e depuração.
- Projecto de sistemas confiáveis.
- Projecto de sistemas de prototipação rápida.

A FEUP tem a seu cargo a produção dos quatro cursos individuais que constituem o primeiro módulo, três dos quais foram desenvolvidos sob responsabilidade do autor desta dissertação:

- Técnicas actuais de projecto de sistemas [Alv98b].
- Técnicas de depuração e validação de protótipos [Alv98c].
- Técnicas de projecto para o teste e depuração [Alv98d].

Para a criação do primeiro curso elaborou-se um questionário que aborda os vários aspectos do ciclo de vida de um sistema electrónico, com ênfase na etapa de projecto e desenvolvimento. Este questionário propõe e utiliza uma divisão dos sistemas electrónicos em três tipos: sistemas baseados em lógica dedicada, sistemas baseados em microprocessadores[2] e sistemas híbridos[3]. Esta divisão e o nível hierárquico do sistema influenciam de uma forma geral o tipo e o número de etapas do ciclo de vida, e em particular a etapa de projecto e desenvolvimento[4]. O ciclo de vida de um sistema inclui tipicamente as seguintes etapas:

- Especificação e análise de requisitos.
- Projecto e desenvolvimento.
- Produção e teste.
- Integração, teste e depuração.
- Manutenção no ambiente de utilização final.

A etapa de projecto e desenvolvimento é tipicamente formada pelas seguintes fases:

- Descrição do sistema (inclui a criação de modelos e desenvolvimento de código).
- Simulação funcional.
- Síntese (ao nível do Circuito Integrado, CI).
- Simulação após síntese (ao nível do CI).
- Colocação e ligação.
- Simulação após colocação e ligação.

No questionário referido reúnem-se várias perguntas acerca da validade desta divisão, das acções efectuadas em cada etapa, particularmente em cada fase da etapa de projecto e desenvolvimento, focando-se em concreto a utilização de infraestruturas de teste normalizadas. O questionário foi utilizado numa série de entrevistas a projectistas de diversas empresas de desenvolvimento de produtos electrónicos, nomeadamente na Noruega e na Suécia, tendo os resultados obtidos sido compilados e descritos em [Alv98b]. Destes resultados podem-se retirar duas conclusões principais: a utilização generalizada de regras de projecto para a testabilidade e uma crescente preocupação com a depuração e verificação dos produtos. No final deste curso elaborou-se um relatório sobre o estado da arte que reúne uma quantidade substancial de informação na área do projecto para o teste e depuração. Esta tarefa estendeu-se por três meses,

---

[2] O termo microprocessdor é usado aqui de forma genérica, ou seja, podendo referir-se a um microprocessador ou a um microcontrolador.

[3] No sentido de incluírem uma componente de lógica dedicada e um ou mais microprocessadores.

[4] Por exemplo, para o caso dos sistemas baseados em microprocessadores, existe a fase de desenvolvimento da parte física do sistema e a fase de desenvolvimento do código da aplicação executada pelo microprocessador.

terminando num relatório que na sua versão impressa ocupa cerca de 350 páginas. No segundo curso descrevem-se os aspectos relacionados com a depuração e validação de protótipos de cada tipo de sistema: modelos para simulação; ferramentas; operações e técnicas de depuração. A conclusão deste curso aponta para as vantagens da utilização de ferramentas de depuração baseadas no acesso electrónico através de infraestruturas de teste normalizadas, e na necessidade de considerar a fase de depuração e validação na etapa inicial da especificação. Esta conclusão serviu de mote para os dois restantes cursos: o primeiro descreve as actuais normas e propostas de normas de infraestruturas de teste e o segundo descreve a utilização da infraestrutura BST para a implementação das operações de depuração apresentadas no segundo curso.

Dada a quantidade de informação existente nos cursos individuais, no questionário e no relatório sobre o estado da arte, e a sua importância para clarificarem o enquadramento que proporcionaram a este trabalho, optou-se por integrá-los no CD-ROM que acompanha esta tese e que constitui um anexo em formato electrónico.

1.1.2 Objectivos e assuntos tratados

A infraestrutura BST tem sido tradicionalmente utilizada para o teste estrutural de CCI na fase de produção [Ble93, Mau90, Par92]. O seu aparecimento deveu-se, entre outras razões, à crescente dificuldade das tradicionais tecnologias de teste de CCI (o teste *in-circuit* e o teste funcional) em lidar com os novos tipos de encapsulamento de CI e com a sua crescente complexidade. A utilização de CI de montagem superficial veio reduzir o distanciamento entre os pinos e permitir a montagem de componentes em ambos os lados da CCI, dificultando assim o acesso físico requerido pelo teste *in-circuit* [Bat85]. A crescente complexidade veio por sua vez dificultar a propagação de valores no interior da CCI, diminuindo assim a qualidade do teste funcional.

A crescente preocupação dos fabricantes e dos utilizadores com a qualidade em geral e com a redução dos tempos de projecto e dos ciclos de vida, têm vindo a colocar diversas dificuldades à verificação e validação dos sistemas electrónicos. A utilização de ferramentas de depuração baseadas no acesso físico enfrenta porém as mesmas restrições sentidas pelas tradicionais tecnologias de teste. Esta conjugação de factores tem levado à utilização do 1149.1 na depuração de protótipos e na manutenção dos sistemas electrónicos, estendendo assim o seu uso para além do teste de produção, conforme se ilustra na Figura 1-1 [Mau92b, Sed94, Whi96]. Os modos de operação definidos na norma não contemplam porém directamente os requisitos levantados por estas novas áreas de utilização, pelo que as lacunas do 1149.1, em face destes novos requisitos, constituem o assunto principal deste trabalho de doutoramento.

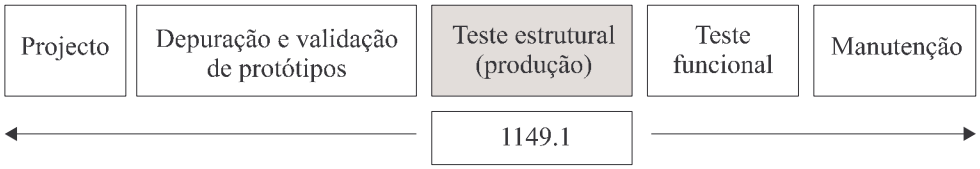


Figura 1-1: Utilização do 1149.1 nas várias etapas do ciclo de vida de um sistema electrónico.

Apesar de já existirem exemplos de modos de operação opcionais do 1149.1, para implementarem operações de depuração, as soluções actualmente disponíveis dificilmente se podem considerar de carácter geral. A dependência entre as soluções propostas e o tipo de sistema sob depuração dificulta uma utilização generalizada, visível no aumento das interligações existentes entre a lógica de teste e a lógica funcional [Hol94, Mit97, Sie95, Zor89]. No presente trabalho propõem-se novos modos de operação opcionais, que requerem apenas um conjunto de pequenas alterações na infraestrutura mínima definida na norma IEEE 1149.1, favorecendo a sua utilização generalizada em qualquer tipo de sistema sob depuração.

Em paralelo define-se a arquitectura de um controlador de teste e depuração para controlar os novos modos de operação. Os requisitos deste controlador incluem a garantia do sincronismo entre a actividade da lógica de teste e a actividade da lógica funcional, necessária para a implementação das principais operações de depuração. A metodologia e a geração automática do programa de teste e depuração constitui o último dos assuntos tratados.

1.1.3 Contributo inovador e originalidade do trabalho

Um dos contributos originais do trabalho efectuado consiste na extensa análise da utilização dos modos de operação básicos e opcionais do 1149.1 para a implementação de operações de depuração. A identificação e caracterização das principais lacunas que esta infraestrutura apresenta para este fim, constituem um guia de desenvolvimento de novos modos de operação opcionais, necessários para as ultrapassar. Os modos propostos nesta dissertação incluem-se entre os contributos originais do trabalho efectuado. A sua inclusão numa infraestrutura de teste normalizada permite dotá-la de capacidades acrescidas de apoio à implementação de operações de depuração, justificando assim a expressão *infraestrutura para o teste e depuração*.

Também a definição da arquitectura de um controlador residente (a partir de uma solução parcial anteriormente desenvolvida em [Fer92a]), bem como da metodologia para a geração automática do respectivo programa de teste e depuração, merecem referência entre os contributos originais aqui descritos. O controlador residente tem como característica inovadora o facto de garantir o sincronismo entre a lógica de teste e a lógica funcional do sistema, através

do controlo simultâneo dos relógios que alimentam estes tipos de lógica. É também inovadora a forma como se conjugam as acções que dizem respeito ao controlo da lógica de teste e ao controlo do relógio do sistema.

1.2 Organização da dissertação

Após o enquadramento geral aqui apresentado, efectua-se no capítulo 2 uma descrição dos aspectos relacionados com a depuração de sistemas electrónicos. Seguindo uma divisão em três tipos de sistemas (referida anteriormente), descrevem-se os vários modelos utilizados, as ferramentas de depuração, e as operações e técnicas de depuração mais comuns.

No capítulo 3 descrevem-se os modos de operação básicos e opcionais das infraestruturas de teste definidas na norma IEEE 1149.1 e na proposta de norma IEEE P1149.4 (para CI mistos, ou seja, que incluem uma componente digital e uma componente analógica). A utilização destes modos de operação básicos e opcionais, para a implementação das operações de depuração apresentadas no capítulo anterior, é analisada em pormenor, identificando-se diferentes opções e as principais lacunas.

No capítulo 4 faz-se uma apresentação global do trabalho que é descrito ao longo dos três capítulos seguintes. Apresentam-se os requisitos do projecto para o teste e depuração, e explica-se a sua influência na definição dos modos de operação opcionais propostos.

O capítulo 5 descreve em detalhe o conjunto de instruções que suportam os modos de operação opcionais propostos para a infraestrutura para o teste e depuração, baseada na norma IEEE 1149.1, ou na proposta P1149.4. Para cada instrução opcional descreve-se o respectivo objectivo, o procedimento de utilização e as modificações necessárias, em termos da infraestrutura mínima (1149.1 ou P1149.4). No final apresenta-se uma análise qualitativa das vantagens e desvantagens dos modos de operação opcionais propostos.

O capítulo 6 descreve a arquitectura do controlador residente, que é apresentada de forma modular, iniciando-se com a descrição do nível de topo e a definição dos pinos de Entrada / Saída, seguidas por uma visão geral da sua constituição interna, dos blocos principais, do conjunto de instruções, e finalmente de cada uma das instruções. No final deste capítulo refere-se em maior detalhe a questão da garantia do sincronismo entre lógica de teste e lógica funcional.

O capítulo 7 descreve a metodologia para a geração automática do programa de teste e depuração, para cada tipo de sistema: lógica dedicada, microprocessadores, híbridos. Apresenta-se o fluxo de informação em função do tipo e nível hierárquico do sistema sob depuração e descreve-se em detalhe cada elemento de entrada e de saída. Ao longo do capítulo são ilustrados vários exemplos do código gerado para o controlador residente.

Os resultados obtidos e as perspectivas de evolução são apresentadas no capítulo 8. O anexo em formato impresso, inserido após as referências bibliográficas, contém vários ficheiros referentes à implementação da infraestrutura para o teste e depuração, e ainda outros referentes ao controlador residente. O anexo em formato electrónico, constituído por um CD-ROM, encontra-se numa bolsa de plástico fixada no verso da contracapa, e contém uma aplicação auto-executável que apresenta, de imediato, os vários elementos armazenados.

Capítulo 2

A depuração de sistemas electrónicos

No capítulo anterior foram introduzidos alguns conceitos relacionados com o desenvolvimento de sistemas electrónicos, nomeadamente uma apresentação geral do ciclo-de-vida e da etapa de projecto e desenvolvimento. Nessa apresentação, referiu-se uma divisão dos sistemas electrónicos em três tipos:

- Sistemas baseados em lógica dedicada.
- Sistemas baseados em microprocessadores.
- Sistemas híbridos, no sentido de incluírem uma componente de lógica dedicada e um ou mais microprocessadores.

Este capítulo segue a mesma divisão ao longo das várias secções que o compõem. A primeira secção descreve os vários tipos de modelos utilizados na simulação, incluindo, para o caso dos sistemas baseados em lógica dedicada, os modelos comportamentais, funcionais, temporais e estruturais. Para o caso específico dos sistemas que incluem microprocessadores, referem-se modelos baseados na Simulação do Conjunto de Instruções e baseados no Funcionamento da Interface com os Barramentos. Após a simulação, o sistema evolui para a fase da fabricação dos primeiros protótipos, pelo que a segunda secção apresenta as ferramentas de depuração actualmente disponíveis para cada tipo de protótipo. Para o caso dos sistemas baseados em lógica dedicada referem-se os sistemas de emulação com capacidades de depuração e as ferramentas baseadas em infraestruturas de varrimento, comparando-se ambos em termos de precisão, aproximação ao sistema final, reutilização e custo. Para o caso dos sistemas baseados em microprocessadores referem-se os monitores residentes, os emuladores por acesso físico (de memória ou do próprio microprocessador), os analisadores lógicos, os módulos de pré-processamento e as ferramentas de depuração baseadas em lógica de emulação embutida no próprio microprocessador. A apresentação das ferramentas de depuração para o caso dos sistemas híbridos encerra a segunda secção. A terceira secção introduz as operações e técnicas de

depuração, implementadas através das ferramentas anteriormente apresentadas. Referem-se quatro tipos básicos de operações de depuração: controlo, observação e verificação de estado; operações passo-a-passo; operações de pontos de paragem por condição; e operações de análise em tempo real. No seu conjunto, estes quatro tipos de operações formam um modelo simplificado de depuração. Ao longo do resto da terceira secção descrevem-se em maior pormenor as técnicas de depuração aplicadas a cada um dos três tipos de sistemas. A quarta secção conclui o capítulo, com um breve resumo dos conceitos mais relevantes, aqui apresentados.

2.1 Modelo e simulação do sistema

A etapa da simulação constitui uma das mais importantes de todo o processo de depuração do sistema ou circuito[5] electrónico, dados os elevados níveis de controlabilidade e observabilidade, garantidos pelo ambiente computacional em que decorre. A simulação é sempre efectuada com base num *modelo* do sistema, pelo que importa caracterizar convenientemente quais os tipos de modelos existentes, nomeadamente quanto à sua complexidade, nível de abstracção, dimensão, aproximação ao funcionamento real do circuito, etc. Deve-se realçar neste ponto que um modelo constitui uma representação computacional do sistema, pelo que existem sempre diferenças (por mais pequenas que sejam) entre o resultado obtido pela simulação do modelo e o resultado obtido através do funcionamento do sistema em si [Pet95, Woo95].

Nesta secção apresenta-se o vocabulário, taxinomia e conceitos básicos referentes ao modelo de um sistema, seguindo duas obras de referência [Abr90, RTW98]. O tipo de simulação efectuada recebe geralmente o mesmo nome do tipo de modelo utilizado ou seja, por exemplo, se se usar um *modelo temporal*, utiliza-se a expressão *simulação temporal*.

2.1.1 Sistemas baseados em lógica dedicada

Independentemente do nível de abstracção, um sistema digital pode ser caracterizado como uma caixa-preta, processando informação a partir das suas entradas, para produzir valores nas suas saídas. Dependendo do nível de abstracção, pode-se caracterizar o seu *comportamento* como um relacionamento entre as suas entradas e saídas, em termos de valores lógicos (ao nível individual de cada bit), palavras digitais (ao nível de conjuntos de bits), etc. Esta transformação dos valores à entrada para os valores à saída ocorre ao longo do tempo, pelo que se torna conveniente, quando se descreve o comportamento de um sistema, separar o universo dos valores do universo temporal. Um *modelo funcional* de um sistema responde a esta necessida-

[5] Os termos "sistema" e "circuito" são frequentemente utilizados de forma indistinta, não existindo uma regra específica que indique claramente as situações em que se deve empregar um outro termo.

de, uma vez que consiste numa representação da sua função lógica, i.e. no relacionamento entre valores à entrada e valores à saída, ignorando qualquer tipo de atraso entre ambos[6]. Um *modelo comportamental* consiste na junção, ao modelo funcional, da informação relacionada com os atrasos existentes ou especificados.

Um *modelo estrutural* descreve um circuito como uma colecção de circuitos mais pequenos interligados entre si, geralmente designados por componentes ou elementos. Um modelo estrutural contém sempre uma hierarquia, de modo que um componente pode ser visto como uma interligação de componentes de nível hierárquico inferior. Os elementos pertencentes ao nível hierárquico mais baixo são designados por elementos primitivos, sendo o correspondente modelo funcional (ou comportamental) conhecido por natureza. Um modelo estrutural comporta sempre (implicita ou explicitamente) informação referente à função dos seus componentes internos. Um *modelo da interface* contém informação relativa aos detalhes da interface, ou Entradas / Saídas (E/S) primárias, de um dado circuito, sem no entanto descriminar qualquer tipo de informação acerca da implementação ou conteúdo interno. Este tipo de modelo permite obter alguma informação acerca da resposta funcional e temporal da interface de um circuito.

Modelo comportamental

Um modelo comportamental pode ser de tipo *abstracto* ou *detalhado*. Um *modelo comportamental abstracto* descreve a interface de um circuito de modo abstracto, i.e. sem pormenorizar a sua interface com o exterior ao nível dos pinos individuais. Os principais objectivos da utilização de um modelo comportamental abstracto são os seguintes:

- Estabelecer e verificar os requisitos funcionais e temporais das interfaces de cada componente de forma a verificar que, colectivamente, estes cumprem os requisitos impostos ou existentes ao nível do sistema.
- Facilitar a reutilização do projecto ao nível do sistema, independentemente de alterações efectuadas na implementação dos componentes internos.
- Ajudar a visualizar a operação de sistemas complexos, de modo a permitir uma melhor compreensão das suas características para efeitos de optimização, especialmente ao nível da CCI - numa filosofia de projecto do tipo *cima-para-baixo* (*top-down*) - antes de assentar na natureza exacta das interface de cada componente.
- Documentar a operação pretendida na implementação do sistema.

[6] Deve-se entender o relacionamento entre entradas e saídas num sentido lato, ou seja, podendo referir-se tanto a um circuito combinatório como a um circuito sequencial. No primeiro, o valor presente numa saída depende exclusivamente do valor actual de uma ou mais entradas, enquanto que no segundo depende também da história passada, ou seja, dos valores anteriormente aplicados numa ou mais entradas.

Um *modelo comportamental detalhado* descreve em pormenor a interface de um componente ao nível de cada pino individual, possuindo em si toda a informação relativa à funcionalidade e à resposta temporal do componente modelado, sem no entanto especificar a estrutura de implementação. O principal objectivo de um modelo comportamental detalhado consiste em desenvolver e verificar a estrutura, função e resposta temporal da interface de um componente, especialmente ao nível da CCI. Um modelo comportamental pode ser criado ou desenvolvido com base numa linguagem de descrição de *hardware*, por exemplo do tipo VHDL[7] ou Verilog.

Modelo funcional

Um *modelo funcional* descreve a função de um sistema ou componente sem qualquer referência ao tipo de implementação efectuada ou a efectuar. Este tipo de modelo pode existir a qualquer nível de abstracção, dependendo do nível de resolução pretendido. Por exemplo, um modelo funcional pode descrever de uma forma abstracta a função de um algoritmo de processamento de sinal, ou corresponder a um modelo com um nível de abstracção inferior, que descreva a função de uma Unidade Lógica e Aritmética (*Arithmetic and Logic Unit*, ALU) que executa esse mesmo algoritmo. A resolução dos valores, internos e externos, depende do nível de abstracção do modelo. Este tipo de modelo é habitualmente utilizado na verificação de um sistema, pela vantagem de separar o universo da função do universo das relações temporais, permitindo que o projectista se concentre no primeiro, acelerando assim o processo de verificação.

Modelo temporal

Em contraste com um modelo funcional, que apenas modela a função de um sistema sem considerar as suas relações temporais intrínsecas, um *modelo temporal* inclui a descrição dos atrasos internos que estão presentes num sistema real, físico. A precisão da informação temporal encontra-se intimamente ligada ao nível de abstracção do modelo e depende da tecnologia de implementação. Outro aspecto importante consiste na inclusão, no modelo temporal, de informação relativa aos atrasos existentes nas interligações, que depende do processo de colocação e ligação dos elementos. Ao nível do CI é possível utilizar uma ferramenta denominada Extractor de Parâmetros da Topologia (*Layout Parameter Extractor*, LPE) que calcula os vários atrasos em função do percurso efectuado por cada ligação (seguindo um processo de análise a duas, ou, mais recentemente, a três dimensões). Alguns autores utilizam o termo *modelo temporal* (*timing model*) para referir os atrasos entre elementos lógicos, e o termo *modelo de atrasos* (*delay model*) para referir os atrasos no interior dos próprios elementos. Estes termos não se

[7] VHSIC *(Very High Speed Integrated Circuit) Hardware Description Language* [IEEE87].

encontram no entanto normalizados, sendo frequentemente utilizados indistintamente. Existem ainda outros termos que designam vários tipos de atrasos:

- Um atraso de pino-a-pino consiste num atraso entre um pino de entrada e um pino de saída de um elemento lógico, excluindo qualquer tipo de contribuição temporal da interligação.
- Um atraso de malha, ou de linha, consiste num atraso exterior ao elemento lógico. Este tipo de atraso representa sempre informação temporal relativa a uma interligação.

Um largo conjunto de ferramentas de projecto permite apenas modelar o atraso referente às portas (ou células) lógicas. A modelação dos atrasos nas interligações ao nível lógico encontra-se ainda pouco desenvolvida, em virtude da escassa informação existente a este nível, muito embora seja possível impor alguns limites, quando se utilizam directivas para a planificação do processo de colocação e ligação (*floorplanning*), ao nível dos CI. De uma forma generalizada, utilizam-se os seguintes modelos, no que se refere aos atrasos ao nível das portas lógicas:

- *Modelo de atraso unitário* (*unit delay model*).
- *Modelo de atraso unitário por saída* (*unit fanout model*).
- *Modelo de atraso específico* (*library model*).

No modelo de atraso unitário, cada expressão Booleana é factorizada em termos de portas lógicas, no máximo, com duas entradas. Assume-se que cada porta lógica possui um atraso unitário, sendo calculado o atraso total em função da soma do número de portas lógicas existentes no percurso mais longo entre uma entrada e uma saída. O modelo de atraso unitário por saída permite refinar o nível de informação existente no modelo anterior, através da soma de um factor constante, por cada entrada ligada à saída de uma porta lógica. No modelo de atraso específico, cada atraso é calculado através do seu mapeamento numa base de dados que contém o atraso específico de cada tipo de porta lógica, em alguns casos com informação relativa ao número de entradas alimentadas. A Figura 2-1 ilustra a aplicação destes três tipos de modelos para uma expressão lógica com duas entradas e duas saídas.

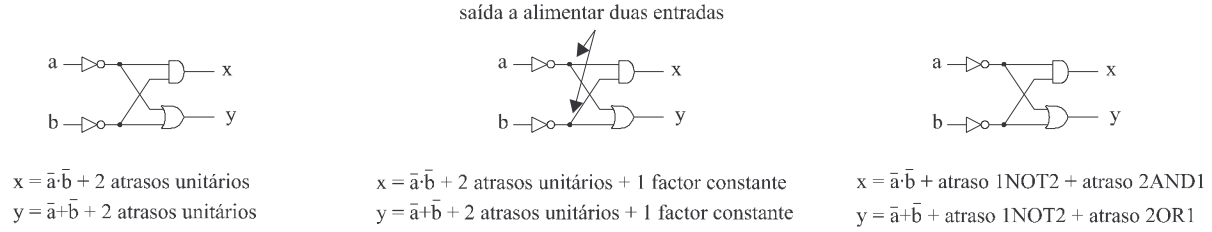


Figura 2-1: Exemplo de aplicação de três modelos de atraso a um circuito combinatório simples.

Modelo estrutural

Um *modelo estrutural* descreve um componente ou sistema em termos das ligações entre os seus elementos constituintes. Um modelo estrutural segue a hierarquia física do sistema modelado, reflectindo a organização de uma dada implementação, através da especificação dos vários componentes e da topologia das ligações existentes entre estes. Os componentes podem ser descritos de uma forma estrutural, funcional ou comportamental. A simulação com base num modelo estrutural implica que todos os modelos dos componentes, pertencentes ao nível hierárquico mais baixo, sejam do tipo funcional ou comportamental. O comportamento deste modelo composto é fornecido pelo comportamento dos blocos de nível inferior, e não apenas pela descrição de como são combinados os blocos individuais. Repare-se que se pode obter um comportamento distinto no caso de se mudar o comportamento de um dos blocos, mantendo no entanto a mesma estrutura. A resolução temporal, funcional e comportamental, depende assim dos blocos de nível hierárquico inferior. No caso de não existir nenhum bloco de tipo funcional ou comportamental no nível hierárquico mais baixo, atribui-se a designação de *modelo puramente estrutural*, não podendo inferir-se qualquer tipo de comportamento em relação ao modelo em questão.

Um modelo estrutural pode existir em qualquer nível de abstracção. A resolução estrutural depende da granulosidade dos vários blocos constituintes e o nível de abstracção depende do nível de detalhe da implementação interna. A estrutura pode ser descrita de uma forma abstracta, como a interligação de blocos de alto-nível, ou mais concretamente como a interligação de portas lógicas.

Níveis de modelação

O nível de modelação refere o tipo de elementos primitivos utilizados num dado modelo. Os elementos primitivos de nível hierárquico mais baixo, num dado sistema modelado, são aqueles que não podem ser decompostos, de uma forma estrutural, em elementos mais simples. De uma forma geral, no nível hierárquico mais baixo consideram-se portas lógicas ou transístores. Qualquer outro tipo de elemento primitivo é designado por elemento primitivo funcional (ou de nível hierárquico superior). A seguinte lista contém alguns dos níveis hierárquicos frequentemente utilizados ou referidos:

- *Nível de transferência entre registos* (*Register Transfer Level*, RTL).
- *Nível lógico* (*logic-level*)
- *Nível da porta lógica* (*gate-level*)
- *Nível do transístor* (*switch-level*)

Um *modelo RTL* descreve um sistema em termos de registos, circuitos combinatórios, barramentos de interligação e circuitos de controlo, geralmente implementados na forma de Máquinas de Estados Finitas (MEF). Das transformações efectuadas pelos registos pode-se inferir alguma informação acerca da estrutura de implementação, apesar desta informação não se encontrar formalmente expressa no modelo. Um modelo RTL serve principalmente para desenvolver e testar uma arquitectura para um dado CI, de forma a verificar as suas especificações funcionais e temporais. Este tipo de modelo é igualmente utilizado para descrever um circuito num formato neutro, passível de ser implementado numa dada tecnologia, através de síntese automática. Um *modelo ao nível lógico* descreve um componente em termos de funções lógicas de tipo Booleano e simples elementos de memória, como por exemplo elementos bi-estáveis (*flip-flops*, FF). Este tipo de modelo não descreve uma implementação exacta da função em termos de portas lógicas. Um *modelo ao nível da porta lógica* descreve a função, atraso e estrutura de um componente, em termos de interligações entre elementos simples de lógica Booleana, como por exemplo: NAND, NOR, NOT, AND, OR, X-OR e X-NOR. São objectivos principais de um modelo ao nível da porta lógica:

- Documentar uma implementação particular de um dado CI, em termos de interligação de elementos de uma família lógica específica.
- Determinar uma resposta temporal precisa do circuito aos estímulos de entrada aplicados, em contraste com os modelos de nível abstracto superior, que descrevem o mesmo circuito.
- Verificar que o projecto de um dado circuito cumpre todos os seus requisitos funcionais e temporais.
- Permitir uma optimização da implementação ao nível da porta lógica de uma descrição lógica do circuito modelado.

Durante o projecto de um determinado sistema é necessário utilizar diferentes níveis de modelação. Numa metodologia de projecto do tipo cima-para-baixo, começa-se geralmente por utilizar um modelo de alto-nível que descreve a especificação inicial de uma forma abstracta, passando progressivamente por diferentes níveis de refinamento, até atingir uma implementação final. A transição da especificação para uma implementação corresponde à transição de um nível superior para um nível inferior de modelação. Um modelo de alto-nível proporciona uma visão mais abstracta de um sistema, em comparação com um modelo de baixo-nível, com a desvantagem de possuir um menor índice de detalhe e precisão, especialmente no que se refere à descrição dos atrasos existentes. Em paralelo, o grau e a quantidade de detalhe de um modelo de baixo-nível requer maiores capacidades de processamento para a ferramenta de simulação que utiliza esse modelo, especialmente no caso dos Circuitos Integrados de Aplicação Específica (*Application Specific Integrated Circuit,* ASIC), mais complexos e de maiores dimensões.

O nível de modelação permite controlar o ponto de opção entre precisão e complexidade de um modelo. À utilização de diferentes níveis de modelação, para descrever os vários componentes internos de um sistema, atribui-se geralmente a designação de modelação hierárquica mista. Este tipo de modelação encontra-se frequentemente associada à modelação de tipo multi-nível, que consiste na utilização de modelos de diferentes níveis hierárquicos para um mesmo componente interno do sistema, podendo o projectista comutar de um modelo para outro entre sessões de simulação [Pic95]. Este tipo de solução combinada permite que se foque a atenção nos detalhes ou pormenores de baixo-nível numa determinada área (que poderá corresponder a um componente isolado ou a um grupo de componentes internos), mantendo em simultâneo modelos de alto-nível para o resto do sistema, de forma a manter, ou a não afectar demasiado, a eficiência e a rapidez de simulação.

2.1.2 Sistemas baseados em microprocessadores

Os vários conceitos e termos apresentados na subsecção anterior mantêm-se igualmente válidos para os sistemas baseados em microprocessadores. O facto, porém, da funcionalidade deste tipo de sistemas se basear num programa, constituído por instruções e operandos, armazenado em memória, sugere a criação de modelos específicos, como o *modelo de Simulação do Conjunto de Instruções* (*Instruction Set Simulation*, ISS) do microprocessador e o *modelo da Interface com os Barramentos* (*Bus-functional*, BF).

Modelo de Simulação do Conjunto de Instruções

Esta abordagem modela o sistema-alvo ao nível da arquitectura do seu conjunto de instruções, tipicamente interpretando as instruções passo-a-passo e simulando o seu efeito ao nível dos pinos de entrada e saída do microprocessador e dos seus registos internos. Este tipo de modelo permite aos programadores observar em detalhe (e controlar) a execução de um dado programa, que poderá corresponder, por exemplo, a um sistema operativo. Simuladores que utilizam este tipo de modelo proporcionam acesso ao estado interno do microprocessador, em alguns casos difícil de obter quando se utiliza o próprio componente em si, para além de permitirem avaliar o comportamento do sistema em face de condições difíceis de reproduzir com o sistema real, físico [Mag97].

Modelo da Interface com os Barramentos

O modelo da interface com os barramentos (controlo, endereços e dados) permite simular o comportamento do microprocessador, como se fosse observado a partir do exterior, ou seja,

sem necessidade de conhecer o resultado originado pela execução do conjunto de instruções. O objectivo principal consiste em gerar os ciclos de acesso do microprocessador aos barramentos externos, por forma a verificar a sua interacção com os restantes componentes, que em conjunto formam o sistema. Este tipo de modelo é mais simples, rápido e eficiente, que o correspondente modelo do próprio microprocessador, permitindo no entanto simular qualquer tipo de comportamento externo.

2.1.3 Sistemas híbridos

O termo co-projecto é frequentemente utilizado para descrever o projecto de sistemas que contêm um (ou mais) microprocessador(es), executando um dado programa, e um (ou mais) componente(s) de lógica dedicada, que em conjunto implementam uma dada função. Este termo é igualmente utilizado no projecto de novos microprocessadores, em que se desenvolve em simultâneo o sistema operativo. O ponto de decisão entre implementação em lógica dedicada ou implementação em código, é frequentemente obtido na fase de especificação, através da utilização de ferramentas computacionais de análise de alternativas de projecto (*design state exploration tools*). Este tipo de ferramentas permitem analisar e avaliar diferentes alternativas para a arquitectura do projecto, sendo geralmente baseadas numa linguagem de alto-nível (como por exemplo o VHDL comportamental, ou o C++).

Após assentar numa determinada arquitectura, é frequente aparecerem alterações nas especificações iniciais do sistema, que resultam em modificações no projecto, pondo assim em causa o ponto de divisão inicialmente estabelecido entre lógica e código. Desta forma, torna-se importante manter todas as opções em aberto antes de atingir o ponto de não retorno, normalmente associado ao fabrico do protótipo inicial. Outro aspecto importante reside nas diferenças que existem entre tempo de desenvolvimento de código e tempo de desenvolvimento de um protótipo físico, sendo o primeiro várias vezes menor que o segundo. A utilização de um modelo da parte implementada em lógica, ligado a um modelo do código a executar, permite à equipa de desenvolvimento do programa (ou sistema operativo) proceder à simulação da interacção entre estas duas partes, ainda antes do fabrico de qualquer protótipo [Olc95]. Este tipo de simulação pode ainda efectuar-se a diferentes níveis de abstracção, dependendo do tipo de modelos existentes. Por exemplo, a Figura 2-2 ilustra o caso de um co-projecto, referente ao desenvolvimento de um novo microcontrolador, onde são utilizados três níveis de simulação, que correspondem a outros tantos níveis de abstracção dos modelos utilizados (comportamental, RTL, e ao nível da porta lógica) [Pau96].

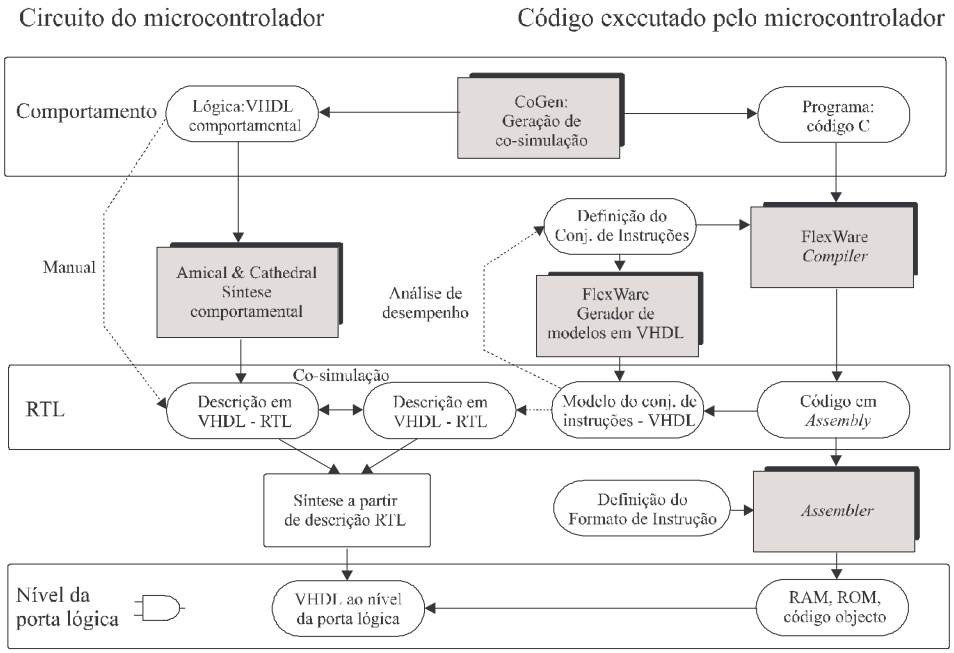


Figura 2-2: Exemplo de um processo de co-projecto com três níveis hierárquicos de simulação.

À medida que o projecto do sistema evolui, são disponibilizados modelos de menor nível de abstracção para a parte implementada em lógica dedicada, até se atingir o ponto em que se obtém um modelo ao nível da porta lógica, com anotação dos tempos de atraso derivados do processo de colocação e ligação, que na prática não tem correspondência directa no lado do código executado pelo microprocessador, onde geralmente se utilizam modelos de simulação do conjunto de instruções ou modelos de interface com os barramentos. Estes dois tipos de modelo não possuem o mesmo nível de detalhe de informação temporal, que existe no modelo da lógica dedicada, pelo que se pode afirmar que, apesar dos ambientes de co-simulação providenciarem um ambiente de depuração com boas potencialidades, apenas se pode aferir acerca do correcto funcionamento em tempo real, quando se interligam as duas partes físicas. Refira-se ainda que o nível hierárquico do sistema sob desenvolvimento constitui um dos factores importantes para a selecção da metodologia de projecto e das ferramentas de geração de modelos mais apropriadas. São exemplos de níveis hierárquicos de sistemas híbridos:

- CCI com um ou mais microprocessadores e um ou mais componentes de lógica dedicada.
- Um CI que inclui no seu interior um macrobloco correspondente ao núcleo de um microprocessador, interagindo com outros macroblocos de lógica definida pelo utilizador.
- Um novo microprocessador, em que código da aplicação / sistema operativo é desenvolvido em simultâneo (ver exemplo ilustrado na Figura 2-2).

Os recursos computacionais necessários para efectuar a co-simulação com modelos de baixo-nível constituem actualmente um dos maiores obstáculos para a sua realização, especialmente para os dois primeiros níveis hierárquicos referidos na lista anterior. Apesar da utilização de placas de aceleração para computadores, que permitem aumentar a rapidez do processo de simulação, é frequente as equipas de projecto optarem por outras alternativas para a verificação do sistema, como por exemplo a emulação.

2.2 Ferramentas de depuração

Esta secção descreve e compara várias das ferramentas de depuração existentes para cada tipo de sistema. À semelhança da organização adoptada na secção anterior, inicia-se a análise pelos sistemas baseados em lógica dedicada, a que se seguem os que são baseados em microprocessadores e por fim os de tipo híbrido.

2.2.1 Protótipos de sistemas baseados em lógica dedicada

Os sistemas de emulação e as ferramentas de depuração baseadas em infraestruturas de varrimento constituem actualmente as duas alternativas mais utilizadas na depuração de protótipos baseados em lógica dedicada. Estas alternativas são descritas em seguida em maior detalhe, efectuando-se no final a comparação entre ambas, em termos de custo, reutilização e nível de precisão / aproximação ao sistema prototipado.

Sistemas de emulação com capacidades de depuração

Um sistema de emulação representa uma alternativa eficiente para a depuração, tanto de CI individuais, como também de protótipos de sistemas baseados em CI, que se encontrem ainda em desenvolvimento. Esta dupla função constitui uma das maiores vantagens associadas aos sistemas de emulação, ou de prototipação rápida, expressão igualmente utilizada com frequência. É comum ainda encontrar-se sistemas de emulação associados a analisadores lógicos, que permitem a amostragem em tempo real de sinais externos, e internos, do circuito emulado. Este tipo de associação é visível na Figura 2-3, que ilustra um sistema de emulação comercializado pela VirtuaLogic Inc. [Vir98], onde é possível distinguir-se a plataforma que executa a aplicação de depuração, o sistema-alvo, o analisador lógico e o emulador, que emula o funcionamento da lógica dedicada, parte integrante do sistema sob desenvolvimento.

Figura 2-3: Sistema de emulação VirtuaLogic-8 [Vir98].

A filosofia base de um sistema de emulação consiste em implementar o modelo que descreve o circuito de lógica dedicada, em componentes complexos de lógica programável[8], que constituem o núcleo central do emulador. Alguns emuladores integram outros componentes do tipo digital e analógico, incluindo memórias, com a possibilidade de efectuar qualquer tipo de interligações entre os seus recursos, através de componentes com matrizes de ligação programáveis (*Field-Programmable Interconnect Chip*, FPIC) [Apt98]. Estes dois tipos de componentes programáveis permitem criar um ambiente de depuração poderoso, onde qualquer modificação no circuito sob desenvolvimento é rapidamente convertido num novo ficheiro de programação. Refira-se ainda que alguns destes componentes possuem pinos de observação programáveis, que permitem trazer para o exterior um número limitado de quaisquer sinais internos, cujos valores podem ser visualizados em tempo real através de um analisador lógico. A aplicação de depuração permite, em alguns sistemas, actualizar o visor do analisador lógico com os nomes dos sinais capturados, de acordo com o nome atribuído no próprio modelo do circuito. A informação capturada pelo analisador lógico pode ainda ser transferida para a plataforma de depuração, para posterior análise numa ferramenta de visualização de formas de onda. A aplicação de depuração pode também fornecer estímulos de entrada para o emulador, através da transferência para um equipamento apropriado dos estímulos de entrada utilizados na simulação do modelo.

O principal benefício de um sistema de emulação assenta na sua velocidade. Pode-se exercitar um maior número de funções num determinado período de tempo, no sistema de emulação, do que numa sessão de simulação. Os sistemas de prototipação rápida permitem emular

[8] Do tipo *Field-Programmable Gate-Array* (FPGA), ou *Complex Programmable Logic Device* (CPLD).

determinadas funções a velocidades próximas (ou idênticas) da velocidade de funcionamento normal do sistema em desenvolvimento. Este factor é crucial quando se pretende verificar se a interface, ou uma dada porção do circuito em desenvolvimento, interage de forma correcta, em tempo real, com o resto do sistema, numa fase do projecto em que não existam ainda protótipos [Mei95, Pet95, Sid95, Woo95]. A possibilidade de avaliar o comportamento do circuito durante as fases iniciais do projecto, através da sua emulação, constitui uma vantagem crítica no desenvolvimento de sistemas que envolvam a manipulação de informação de tipo subjectivo, como por exemplo sinais de audio ou vídeo. Só através da avaliação do funcionamento em tempo real deste tipo de sistemas, é que se pode aferir se este cumpre ou não a função pretendida. Repare-se que é extremamente difícil avaliar, através de formas de onda produzidas com base numa simulação do sistema, se uma determinada imagem de vídeo possui ou não a qualidade desejada [Mad95].

Ferramentas de depuração baseadas no acesso por varrimento

O actual número de componentes comerciais e ASICs que dispõem de uma infraestrutura de teste por varrimento periférico, compatível com a norma IEEE 1149.1 [IEEE93], permite suportar a ideia de que um sistema baseado em lógica dedicada possuirá, muito provavelmente, um ou mais componentes BST. Apesar desta infraestrutura ser tradicionalmente utilizada para o teste estrutural de CCI, a partir de inícios / meados dos anos noventa, a sua utilização tem evoluído para outros objectivos, como por exemplo suportar a emulação de lógica ou a depuração de projectos aos níveis hierárquicos do CI e da CCI [Kat94, Sed94]. Refira-se ainda que a sua utilização alternativa, para efeitos de depuração de protótipos de CCI, foi inicialmente sugerida e experimentada numa fase em que ainda era uma proposta de norma do IEEE [Hal89].

Projecto ao nível da CCI

A existência de componentes BST numa CCI em desenvolvimento, facilita a tarefa da sua depuração. Esta vantagem é porém mais evidente quando se utilizam ferramentas de depuração apropriadas, que recorrem precisamente às capacidades disponibilizadas pela existência de cadeias de varrimento no interior da CCI. Um exemplo deste tipo de ferramentas é dado pelo sistema de diagnóstico (*Diagnostic System*) da Asset-Intertech, Inc. [Ass98], que proporciona ao utilizador, através de dois módulos denominados Analisador de Varrimento (*Scan Analyser*) e Depurador (*Debugger*), a possibilidade de controlar, observar e verificar, de uma forma interactiva, sinais da CCI acessíveis através de células de varrimento (ao nível de registos internos ou periféricos). O utilizador pode aplicar um conjunto de vectores de entrada e capturar os resultados obtidos a partir das cadeias de varrimento, para posterior visualização em formas de onda, tal como se ilustra na Figura 2-4, ou em formato tabular (com valores hexadecimais, de-

cimais, binários ou simbólicos), à feição de um analisador lógico. Os valores capturados correspondem à resposta, num formato vector-a-vector, da actividade do circuito, permitindo assim analisar a sua funcionalidade. A comparação entre os valores esperados e os valores capturados pode ainda ser efectuada automaticamente, de forma a aumentar a velocidade de verificação do funcionamento do circuito. Um outro módulo, denominado Visualizador de Pinos (*Pin View*), permite aceder à mesma informação, sob a orientação do nome / número de pino associado à célula de varrimento que aplicou / capturou esses mesmos valores. Uma vez que a informação se refere a pinos individuais, todos os valores são exibidos em formato binário, conforme se ilustra na Figura 2-5.

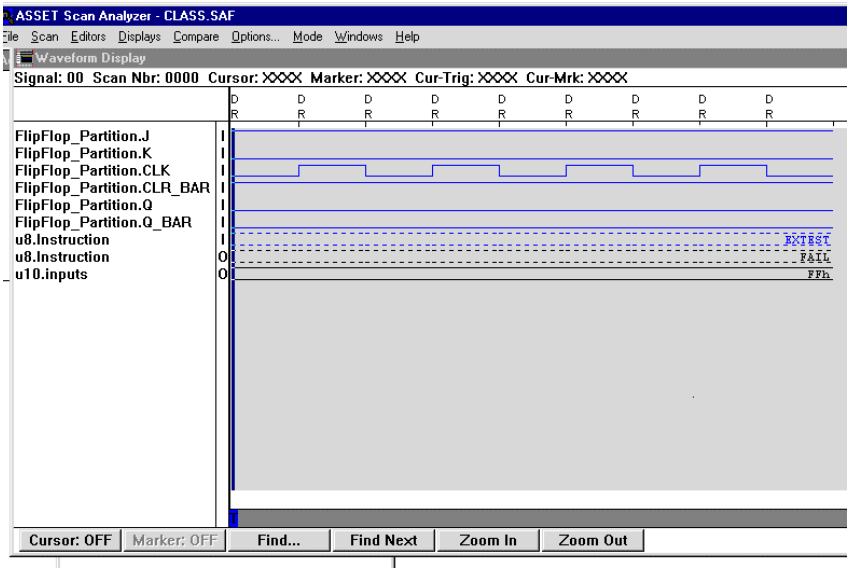


Figura 2-4: Janela do módulo *Scan Analyser* exibindo formas de onda referentes aos vectores capturados através das cadeias de varrimento existentes na CCI [Ass98].

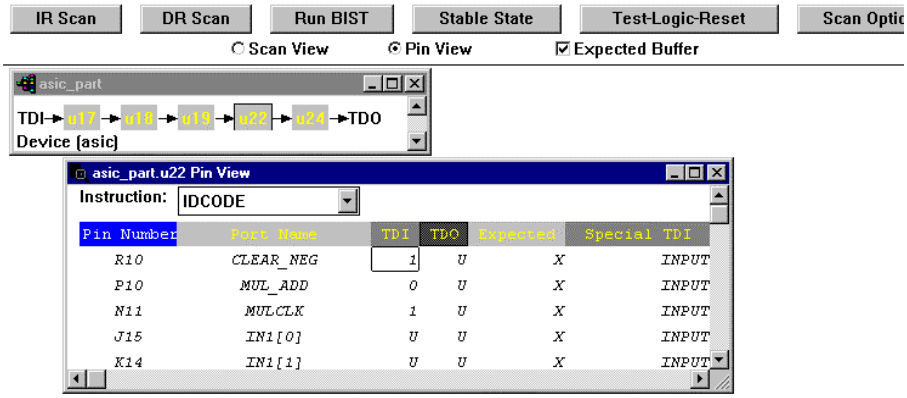


Figura 2-5: Janela do módulo *Pin View* exibindo valores binários referentes aos valores capturados nas células de varrimento associadas aos pinos de componentes BST [Ass98].

Projecto ao nível do CI

Ao nível do projecto de novos CI, é frequente encontrarem-se exemplos de ferramentas de depuração desenvolvidas internamente, i.e. pela própria equipa de desenvolvimento. Na realidade, diversas equipas implementam lógica dedicada de emulação (para efeitos de depuração) no interior do próprio CI, utilizando em seguida o sistema ou produto final, que incorpora esse CI, como o ambiente de emulação. Várias destas equipas têm publicado os resultados obtidos na utilização de lógica de depuração, baseada em cadeias de varrimento tanto do tipo interno como do tipo periférico [Hao95, Hol94, Lev95, Sie95]. Refira-se ainda que dos vários resultados obtidos se evidencia uma redução no custo da depuração dos protótipos iniciais, uma minimização do esforço de desenvolvimento e uma maior aproximação entre o sistema de emulação e o sistema ou produto final. As características da lógica de depuração inserida nos CI podem ser resumidas nos seguintes termos:

- Um controlador de porto de acesso compatível com a norma IEEE 1149.1.
- Cadeias de varrimento internas completas, ou seja, atravessando todos os elementos sequenciais do CI (à excepção daqueles que pertencem a blocos de memória internos).
- Acesso por varrimento à periferia dos blocos de memória internos (o custo de garantir um acesso por varrimento a todos os elementos de memória é ainda proibitivo).
- Um controlador interno de relógio, que permite implementar operações de arranque e paragem, comandadas através do porto de acesso por varrimento.

Refira-se por último duas das maiores vantagens das ferramentas de depuração baseadas no acesso por varrimento: a facilidade de transporte e a facilidade de reutilização em diferentes etapas do ciclo de vida do sistema. Ambas as vantagens decorrem do facto da lógica de depuração ser parte integrante do próprio CI, acompanhando-o sempre, desde o momento em que os primeiros protótipos são fabricados, até ao momento em que o CI se encontra já implementado no sistema e a trabalhar no seu ambiente de utilização final [Sta96, Whi96].

Comparação entre as ferramentas existentes

As ferramentas de depuração para CI, baseadas no acesso por varrimento, são ainda praticamente inexistentes como solução disponível comercialmente em larga escala. As várias ferramentas anteriormente referidas para este nível hierárquico correspondem a soluções proprietárias, desenvolvidas internamente, para casos específicos. Ao nível hierárquico da CCI, o panorama é radicalmente oposto, uma vez que este tipo de ferramenta assenta na existência de uma infraestrutura de teste normalizada (compatível com a norma IEEE 1149.1), nos componentes inseridos no sistema sob desenvolvimento. A razão entre o número de componentes BST e não

BST, constitui um dos critérios principais para o sucesso da utilização destas ferramentas. De uma forma geral, as ferramentas disponíveis no mercado suportam as instruções obrigatórias e opcionais definidas na norma. As instruções opcionais, que permitem aceder a cadeias de varrimento internas de CI, não se encontram definidas na norma. Porém, é possível identificar este tipo de instrução, nomeadamente o seu código e o comprimento do registo de varrimento interno seleccionado, através do ficheiro que descreve a infraestrutura de teste, escrito numa linguagem normalizada denominada *Boundary Scan Description Language* (BSDL) [Par92]. O suporte de outro tipo de instruções de controlo ou acesso a funções de apoio a tarefas de teste ou depuração, requer geralmente o desenvolvimento de rotinas próprias, numa linguagem aceite pela ferramenta utilizada. No caso da ferramenta da Asset-Intertech Inc., referida anteriormente, estes módulos são desenvolvidos em linguagem C.

As ferramentas de depuração baseadas no acesso por varrimento e os sistemas de emulação com capacidades de depuração, podem de facto representar soluções complementares e não concorrentes, apesar de servirem o mesmo objectivo. A possibilidade de utilizar um sistema de emulação nas fases iniciais de desenvolvimento, ainda sem qualquer tipo de protótipo disponível, e o facto de as cadeias de varrimento serem parte integrante do produto final, são argumentos que favorecem este tipo de solução mista. A principal desvantagem desta solução reside nos custos associados. A Tabela 2-1 sumaria a análise comparada das ferramentas de depuração apresentadas.

Tabela 2-1: Comparação das ferramentas de depuração para sistemas baseados em lógica dedicada.

	Precisão / aproximação ao sistema final.	Facilidade de transporte / reutilização	Custo
Sistema de emulação com capacidades de depuração.	Média. O sistema de emulação pode operar a uma velocidade próxima ou idêntica à velocidade de funcionamento normal do circuito emulado. Tempos de atraso, níveis de consumo e outras características, diferem geralmente do circuito emulado para o circuito final.	Média / baixa. Um sistema de emulação é um equipamento pesado, geralmente utilizado em laboratório, numa fase em que ainda não existem protótipos. Pode ser reutilizado em vários projectos embora a complexidade crescente dos novos circuitos limite o seu tempo de vida útil.	Investimento inicial elevado. Custo repartível se o sistema de emulação for utilizado em vários projectos. É possível expandir as capacidades do sistema adquirido, em função do aumento da complexidade do circuito a emular, através da aquisição de módulos de expansão. O custo do analisador lógico, geralmente associado a este tipo de sistema, pode ou não ser considerado, consoante já se encontre ou não adquirido.
Ferramentas de depuração baseadas no acesso por varrimento.	Elevada. Apesar da lógica associada à implementação das cadeias de varrimento representar uma área acrescida e um aumento nos tempo de atraso, estes estão presentes tanto no protótipo como no circuito final.	Elevada. As cadeias de varrimento podem ser utilizadas em todas as etapas do ciclo de vida do produto, a partir do fabrico do primeiro protótipo, sendo parte integrante do próprio circuito.	Principalmente relacionada com a área de silício (ou da carta) adicional. Os pinos do porto de acesso à infraestrutura de teste podem no entanto fazer já parte da especificação inicial do sistema, em virtude de requisitos associados ao teste estrutural.

2.2.2 Protótipos de sistemas baseados em microprocessadores

Existem diversos tipos de ferramentas que podem ser utilizadas na depuração de sistemas baseados em microprocessadores[9]. A seguinte lista refere algumas das ferramentas de depuração mais comuns, de acordo com [Eri95]:

- Programas monitor.
- Emuladores por acesso físico.
- Analisadores lógicos e osciloscópios.
- Módulos de pré-processamento.
- Emuladores embutidos.

Descreve-se em seguida, com maior detalhe, cada um destes tipos de ferramentas.

Programas monitor

Um programa monitor consiste num pequeno programa que permite controlar a execução da aplicação principal, proporcionando acesso a determinados recursos, como registos internos do microprocessador e memórias, ou ainda transferir da plataforma de depuração, para a memória de programa (inserida no sistema), o código da aplicação a executar. O nível de impacto é elevado, dado que um programa monitor requer não só espaço de memória, como ainda tempo de execução. Tipicamente, o sistema sob depuração fornece, para suporte do programa monitor, espaço em memória (volátil e não não-volátil), um porto de acesso para a comunicação com a plataforma de depuração e ainda, pelo menos, um pino de interrupção do microprocessador, instalado no sistema em desenvolvimento. Este impacto é especialmente elevado em termos de código, sendo mínimo em termos de modificação de tempos de atraso / cargas nas linhas do microprocessador.

Das ferramentas de depuração referidas na lista apresentada, pode-se considerar esta como a menos robusta ou fiável, uma vez que todo o sistema necessita de estar operacional, para que o programa monitor cumpra a sua função. No entanto, apesar de ser considerada a ferramenta menos robusta, continua a ser utilizada por duas razões principais: baixo custo e reduzida complexidade [Eri95, CAD98]. Diversos sistemas comerciais de desenvolvimento para microprocessadores incluem geralmente um programa monitor residente, para apoiar os programadores no desenvolvimento de aplicações. Com efeito, existem exemplos de fabricantes deste tipo de

[9] Alguns autores referem-se a este tipo de ferramentas como ferramentas de depuração da aplicação executada pelo microprocessador.

sistemas que disponibilizam, através da Internet e sem custos para o utilizador, versões actualizadas de programas monitores [CED98]. A plataforma de depuração permite estabelecer a ligação entre as ferramentas de desenvolvimento da aplicação e o programa monitor, possibilitando ao utilizador desenvolver o código, transferi-lo rapidamente para o sistema de desenvolvimento, controlar a sua execução e posteriormente visualizar o conteúdo dos registos internos e da memória, em determinados pontos de execução.

Emuladores por acesso físico

Inclui-se neste tipo de ferramentas de depuração os emuladores de memória (ROM *emulators*) e os emuladores do próprio microprocessador (*In-Circuit Emulators*, ICE), dado que ambos necessitam de aceder fisicamente ao sistema sob depuração, através do suporte utilizado pela memória / microprocessador, ou por ligação ao seu topo.

Emuladores de memória

Um emulador de memória permite substituir a memória acedida pelo microprocessador, por uma "outra" que possibilita a sobreposição de um programa monitor ao código da aplicação principal. Este tipo de ferramenta possibilita ao utilizador a transferência do código da aplicação das ferramentas de desenvolvimento para o sistema, e o controlo da sua execução através de uma técnica que consiste na substituição das instruções da aplicação original por outras instruções apropriadas, que implementam as operações de controlo seleccionadas. A principal desvantagem deste tipo de ferramenta reside na necessidade de acesso físico ao sistema em desenvolvimento, podendo considerar-se esta desvantagem como uma restrição importante à sua utilização. A Figura 2-6 ilustra a ligação física de um emulador de memória a um sistema, onde se evidencia o aspecto / dimensão da ponta de emulação.

Figura 2-6: Exemplo de um emulador de memória ligado a um sistema baseado num microprocessador [Int98].

Alguns emuladores de memória possuem ainda circuitos especiais que possibilitam a comunicação bidireccional de dados, possibilitando o envio de resultados de certas operações, para a plataforma de depuração. As vantagens principais deste tipo de emuladores consistem na sua velocidade de operação, que facilmente pode igualar a velocidade de funcionamento do sistema sob depuração, e no facto de, normalmente, um único tipo de arquitectura do emulador cobrir quase todos os tipos de sistemas baseados em microprocessadores.

*Emuladores de microprocessadores (*ICE*)*

Esta é por muitos considerada a mais eficaz de todas as ferramentas de depuração para sistemas baseados em microprocessadores. Esta afirmação é suportada pelas suas diversas vantagens, nomeadamente:

- Pode ser utilizada sem o sistema em desenvolvimento estar totalmente operacional, incluindo as memórias de programa e / ou de dados.
- Inclui a sua própria memória, que pode ser sobreposta sobre a memória do sistema.
- Permite controlar a execução da aplicação, aceder aos registos internos do microprocessador, transferir o código da aplicação da plataforma de depuração para o sistema, ou ainda visualizar o conteúdos das memórias.
- Alguns emuladores incorporam também a capacidade de amostrar em tempo real os valores presentes nos pinos do microprocessador (em sincronismo com o seu funcionamento – *análise de estados* – ou em sincronismo com um relógio interno do próprio emulador – *análise temporal*[10]).
- Funciona à velocidade normal / máxima de operação do microprocessador emulado, possuindo ainda algumas capacidades de teste estrutural e funcional do sistema em si [Fen96].

O impacto sentido no sistema pela utilização desta ferramenta é nulo ou praticamente desprezável. Do lado da aplicação, o impacto é nulo, dado que o emulador possui o seu próprio espaço de endereçamento, não acessível a partir do sistema, e o seu próprio programa de controlo, que lhe permite responder às tarefas solicitadas, nomeadamente o acesso aos registos internos e à memória do sistema. Refira-se neste ponto que o microprocessador inserido no sistema é desactivado pelo emulador (por exemplo, através do pino de inicialização), permitindo-lhe assim aceder e controlar os vários barramentos (controlo, dados e endereços). Do lado físico do sistema existe um ligeiro aumento nos tempos de atraso e nas cargas das linhas do microprocessador, que resultam da ligação física do emulador ao sistema.

[10] Estas duas expressões resultam da tradução da expressão "*state and timing analysis"*, frequentemente associada aos analisadores lógicos, que se descrevem mais adiante nesta mesma secção.

A maior parte dos emuladores são construídos com base no próprio microprocessador que emulam[11] e alguns circuitos adicionais que lhes permite comutar da, e para a, plataforma de depuração, que se encontra normalmente instalada num computador. Esta característica é evidenciada na Figura 2-7 e na Figura 2-8, que ilustram respectivamente a arquitectura de um emulador sem e com capacidade de amostragem em tempo real, onde se destaca a existência, no seu interior, de um microprocessador.

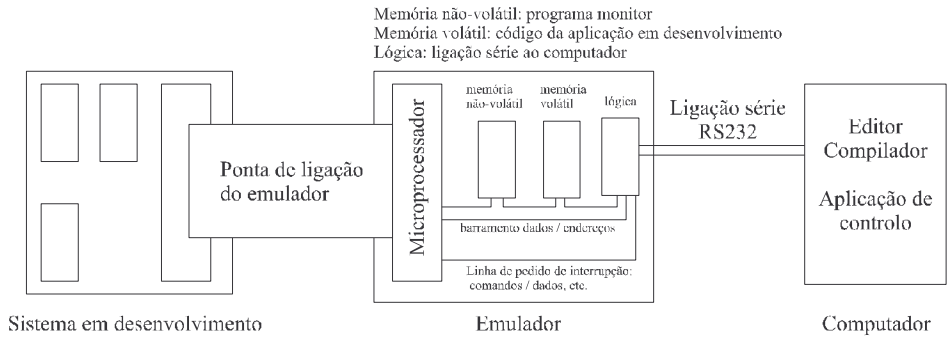


Figura 2-7: Arquitectura interna de um emulador sem capacidade de amostragem em tempo real.

A arquitectura ilustrada na Figura 2-7 consiste basicamente num sistema de avaliação / demonstração disponibilizado pelo fabricante do microprocessador, com uma ligação de tipo série ao computador onde se encontra instalado o sistema de desenvolvimento e a aplicação de depuração. O emulador possui internamente uma memória de tipo volátil para armazenar o código da aplicação em desenvolvimento, e uma memória de tipo não volátil, com um programa monitor responsável pelo interface com o computador. O programa monitor inclui rotinas que permitem ao utilizador visualizar / modificar o conteúdo dos registos internos ou da memória do microprocessador, e ainda executar o código da aplicação num modo de operação passo-a-passo, ou até ser satisfeito um determinado ponto de paragem por condição. Sempre que a aplicação de depuração necessita de implementar um comando, gera através do canal de comunicação série um pedido de interrupção para o microprocessador interno do emulador. Neste tipo de arquitectura, todas as comunicações entre o emulador e o computador são efectuadas

[11] Alguns emuladores são construídos com base numa versão especial do microprocessador que emulam, designada por versão *bond-out* [Lam96]. Estas versões possuem alguns fios de ligação extra, normalmente desligados nas versões comerciais (encapsuladas) do microprocessador. Do lado da pastilha de silício, estes fios acedem a sinais internos do microprocessador, permitindo ao emulador monitorar e controlar determinados estados internos, que de outra forma seriam inacessíveis a partir do exterior. Alternativamente, alguns emuladores são construídos com base em ASIC, o que lhes permite maior grau de acessibilidade e controlo, uma vez que o projecto é efectuado de raiz, com base num conjunto de especificações que normalmente requerem maior acessibilidade e controlo dos estados internos.

através da porta de comunicação série. A Figura 2-8 ilustra uma arquitectura típica de emuladores de topo de gama, onde se distingue a existência de uma memória com portas de acesso diferenciadas, permitindo que o microprocessador continue a sua operação normal, enquanto a aplicação de depuração baseada no computador acede à memória do emulador. Dado que o microprocessador não actua na transferência de informação entre o computador e a memória do emulador, não existe necessidade de implementar um programa monitor em memória não-volátil (a aplicação de depuração simplesmente transfere o programa monitor para a memória volátil do emulador, após o seu arranque). Sempre que a aplicação de depuração necessita que o microprocessador efectue um qualquer tipo de operação, como por exemplo visualizar / modificar o conteúdo de um registo interno, gera directamente um pedido de interrupção que leva o microprocessador a executar o programa monitor. Note-se ainda a existência de uma placa de aquisição de dados, instalada no computador, que permite a amostragem em tempo real dos valores presentes nos pinos do microprocessador, suportando assim a análise em tempo real da actividade nos seus barramentos, em sincronismo com o seu funcionamento.

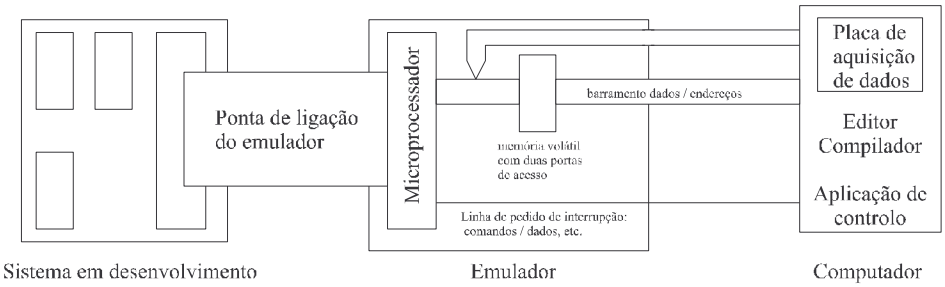


Figura 2-8: Arquitectura interna de um emulador com capacidade de amostragem em tempo real.

A qualidade, e por conseguinte o custo, de um emulador, depende da sua aproximação ao microprocessador emulado. Idealmente, deveria ser nula a diferença entre a situação de se encontrar inserido no sistema o emulador ou o microprocessador em si. De uma maneira geral, quanto maior / melhor a aproximação, maior será o custo. Este factor é em especial crítico no desenvolvimento de emuladores de elevada qualidade, para alguns dos mais recentes microprocessadores, originando assim a adopção de outro tipo de soluções, como por exemplo a inclusão, no próprio microprocessador, de lógica dedicada para o suporte a operações de depuração.

Analisadores lógicos e osciloscópios

Os osciloscópios permitem visualizar as características eléctricas e temporais dos sinais do microprocessador. Na situação de nada funcionar após o arranque do sistema, muito provavelmente, esta será uma das primeiras de ferramentas a utilizar para diagnosticar o problema. Os

analisadores lógicos são instrumentos que permitem visualizar os valores lógicos existentes nos pinos do microprocessador. Existem equipamentos que incorporam estas duas funções, conforme se ilustra na Figura 2-9, que representa um exemplo de um osciloscópio / analisador lógico, fabricado pela Hewlett-Packard [HP99].

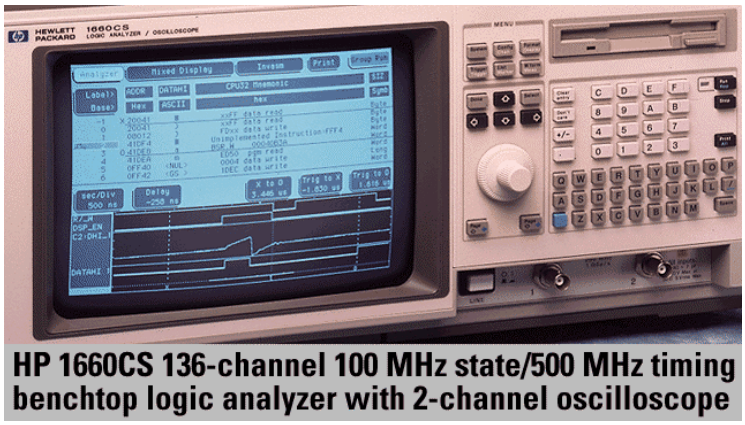


Figura 2-9: Exemplo de um equipamento misto osciloscópio / analisador lógico [HP99].

Existem dois tipos básicos de analisadores lógicos: os de estado (*state*) e os temporais (*timing*). A diferença entres estes tipos está patente na Figura 2-10, que ilustra o esquema típico de ligação de um analisador lógico ao sistema sob depuração. Na qualidade de *analisador lógico de estados*, é o relógio do sistema que regula o momento da amostragem dos valores presentes nos pontos acedidos pelas pontas de prova. Na qualidade de *analisador lógico temporal*, o relógio de amostragem é gerado internamente no analisador, não existindo sincronismo entre o funcionamento do circuito e o momento da amostragem. Este facto é geralmente irrelevante, dado que o relógio interno do analisador atinge frequências superiores à frequência do relógio do sistema amostrado. Os analisadores lógicos de estado são geralmente utilizados para monitorar a actividade nos barramentos de dados e endereços do microprocessador, sendo os analisadores lógicos temporais geralmente utilizados para monitorar a actividade em outras linhas, como por exemplo, linhas de pedido de interrupção, ou ainda linhas de controlo de protocolos de comunicação, em que os valores lógicos não dependem directamente do microprocessador.

Um analisador lógico captura, e armazena na sua memória interna, os valores amostrados como uma série de valores lógicos, '0' ou '1', dependendo de no momento da amostragem o valor em tensão presente no ponto acedido pela ponta de prova, ser inferior ou superior a uma determinada referência, normalmente denominada por tensão de *limiar* (*threshold*). Estes valores podem posteriormente ser visualizados no mostrador do analisador como uma lista de valores binários, octais, decimais, hexadecimais, ou ainda como formas de onda, com possibilidade de identificação do nome dos sinais a que correspondem na descrição do sistema.

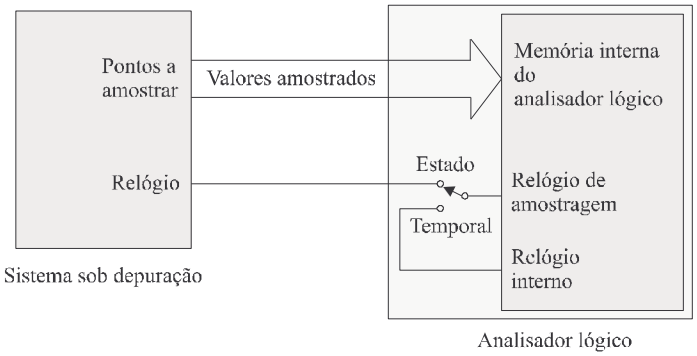


Figura 2-10: Ligação de um analisador lógico de estados / temporal ao sistema sob depuração.

Módulos de pré-processamento

Um pré-processador consiste basicamente numa ponta de ligação física e num conjunto de módulos de processamento, que geralmente incluem os seguintes elementos:

- Módulo de conversão de valores amostrados em mnemónicas de linguagem *assembly* (normalmente designado numa forma abreviada por *disassembler*).
- Processador de símbolos.
- Módulo de análise combinada de código-fonte[12] / valores capturados.
- Módulo de análise de desempenho de programa (*Program Performance Analysis*, PPA).

O módulo de conversão divide os valores capturados pelo analisador lógico, em endereços, códigos de instrução, dados e sinais de controlo, convertendo posteriormente os códigos de instrução em mnemónicas *assembly* do microprocessador. O processador de símbolos atribui nomes simbólicos aos endereços de instruções e dados, que resultam do módulo de conversão, relacionando desta forma a actividade dos barramentos com o correspondente código-fonte. O módulo de análise combinada exibe conjuntamente o código-fonte e os valores capturados pelo analisador lógico, num modo misto, em que cada linha do código-fonte é relacionada com o resultado do processamento efectuado pelos dois módulos anteriores. O módulo de análise de desempenho de programa extrai do ficheiro com o código-objecto informação sobre os endereços, de forma a determinar quanto tempo de execução foi gasto com cada uma das rotinas do programa do microprocessador. Os resultados da análise são exibidos num formato tabular, ou de gráfico, conforme se ilustra na Figura 2-11. Estes módulos efectuam geralmente a sua análise sobre o conjunto de valores armazenados na memória interna do analisador lógico, garantin-

[12] O código-fonte corresponde geralmente a uma linguagem de nível mais elevado que a linguagem *assembly*, como por exemplo a linguagem C.

do-se que apenas é capturada a actividade correspondente às rotinas seleccionadas, através da capacidade existente neste equipamento para especificar condições de início / fim de amostragem [Dea98].

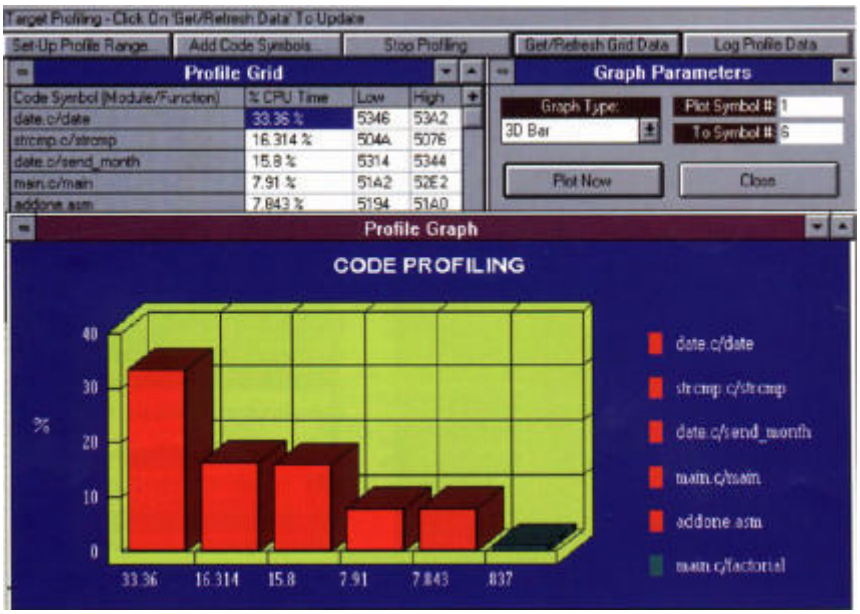


Figura 2-11: Exemplo da janela de um módulo de análise de desempenho de programa, com visualização em forma de gráfico do tempo gasto na execução de várias rotinas [EST98].

A ponta de ligação (também designada de ponta do pré-processador) estabelece a conexão física entre o analisador lógico e o conector onde se instala o microprocessador pertencente ao sistema sob depuração. Tal como sucede com a ponta de ligação de um emulador de microprocessador, a utilização deste tipo de ferramentas de depuração requer algum cuidado no desenvolvimento da CCI, sendo necessário reservar algum espaço livre à volta do conector ou local onde se encontra instalado o microprocessador, devido à dimensão e formato destas pontas de ligação. A Figura 2-12 ilustra dois exemplos de pontas de ligação de pré-processadores, de forma a evidenciar precisamente este aspecto.

Refira-se, por último, que os módulos de pré-processamento não possuem qualquer tipo de controlo sobre a execução do programa do microprocessador, nem qualquer tipo de acesso a registos internos ou memórias. O impacto ao nível do sistema é praticamente nulo, existindo apenas um ligeiro aumento nos tempos de atraso e nas cargas das linhas do microprocessador, que resultam da ligação da ponta do pré-processador. O impacto ao nível da aplicação do sistema é nulo, dado que não são utilizados quaisquer recursos do microprocessador.

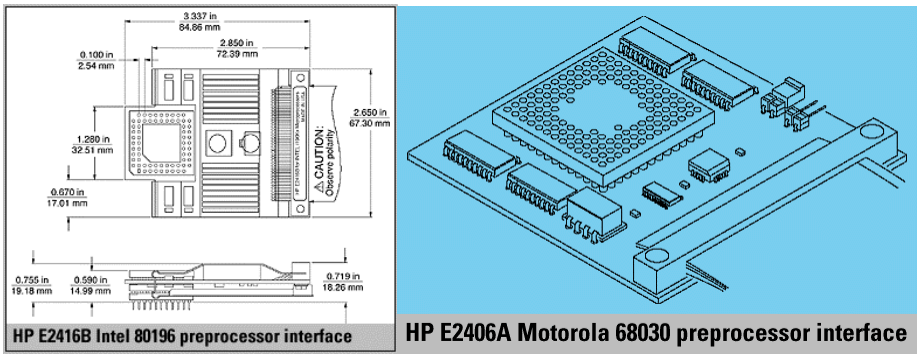


Figura 2-12: Exemplos de pontas de ligação para módulos de pré-processamento [HP99].

Emuladores embutidos

Um emulador embutido consiste num circuito dedicado, de apoio a tarefas de depuração, implementado no interior do próprio microprocessador. Esta ferramenta possui uma funcionalidade idêntica aos emuladores de microprocessadores, anteriormente apresentados, garantindo o controlo e acesso a registos internos, capacidade de visualizar / modificar posições de memória, transferir o código da aplicação para a memória do programa e executá-lo num modo de operação passo-a-passo, ou até um determinado ponto de paragem por condição. O acesso a um emulador embutido é efectuado através de um conjunto de pinos do microprocessador, que poderá ter um número reduzido, no caso de corresponder, por exemplo, ao porto de acesso definido na norma IEEE 1149.1, ou um número mais elevado, no caso de corresponder, por exemplo, ao porto de acesso utilizado em alguns microcontroladores da Motorola, que possuem um emulador embutido designado por *Background Debug Mode* (BDM).

Este tipo de ferramenta de depuração pode ser utilizado na situação em que o sistema não se encontra totalmente operacional, sendo o seu impacto quase nulo. O impacto ao nível físico do sistema é mínimo, sendo apenas necessário incluir um pequeno conector de acesso para ligação à plataforma de depuração. O impacto ao nível da aplicação do sistema é nulo. Refira-se no entanto que o impacto ao nível da área de silício inicial (i.e. sem considerar o circuito dedicado de apoio à depuração) pode ser significativo, embora existam poucos dados acerca de valores concretos. Apesar do seu impacto ao nível do próprio microprocessador, a implementação deste tipo de ferramenta de depuração tem vindo a crescer significativamente, devido às vantagens que lhes estão associadas, especialmente no que se refere à sua utilização em sistemas embutidos baseados em microprocessadores[13] [Lau96, Lea95, Nov95]. A sua utilização tem vindo a crescer pelo facto de a velocidade de funcionamento e a complexidade dos novos

[13] Expressão que resulta da tradução de "*embedded microprocessor-based systems*", que designa um tipo de sistema em que o acesso físico ao microprocessador é muitas vezes difícil ou impossível.

microprocessadores, terem vindo a aumentar em paralelo com uma diminuição acentuada das dimensões dos encapsulamentos[14], que dificultam cada vez mais a ligação física de outros tipos de emuladores. Outra das vantagens importantes deste tipo de ferramenta consiste no seu tempo de desenvolvimento, inferior ao tempo necessário para desenvolver e colocar no mercado um emulador equivalente, por acesso físico, que requer geralmente um período de seis a nove meses, para um novo microprocessador [Lau96].

Refira-se por último que um emulador embutido é parte integrante do microprocessador, e por conseguinte do sistema final, sendo por isso reutilizável ao longo de todo o ciclo de vida do sistema. Este aspecto é especialmente importante no caso de se pretender efectuar a depuração de problemas surgidos quando o sistema se encontra instalado no seu ambiente de utilização normal. A ligação da plataforma de depuração, instalada num computador portátil, pode ser efectuada através de um simples cabo ligado a um conector do sistema. Desta forma pode-se montar rapidamente um ambiente de depuração nos locais em que não seja possível deslocar ou montar outro tipo de equipamento mais pesado, como por exemplo um emulador por acesso físico, ou um analisador lógico. Este último equipamento é no entanto necessário, no caso de se pretender amostrar em tempo real os valores presentes nos pinos do microprocessador, excepto nos casos em que se implementa no sistema uma memória dedicada para esse efeito, controlada / acedida através do mesmo porto de acesso ao emulador embutido.

Comparação entre as ferramentas existentes

A Tabela 2-2 sumaria a análise comparada (em termos de recursos utilizados / impacto no sistema / reutilização / custo) das várias ferramentas de depuração apresentadas para o caso dos sistemas baseados em microprocessadores.

Tabela 2-2: Comparação das ferramentas de depuração para sistemas baseados em microprocessadores.

	Recursos utilizados / impacto	Reutilização	Custo
Programa monitor	Elevado. Utiliza espaço em memória, uma ou mais portas de entrada / saída e em alguns casos uma linha de interrupção do microprocessador.	Baixo / nulo. Geralmente utilizado na fase de desenvolvimento. Não fazendo parte do sistema final, não é utilizado na depuração de problemas surgidos no ambiente de trabalho normal. Depende do microprocessador.	Baixo. Consiste em código a executar pelo microprocessador do sistema. O tempo de desenvolvimento é geralmente mínimo.

[14] Especialmente no caso de microprocessadores de montagem superficial. Refira-se ainda que alguns tipos de encapsulamento (e.g. *Ball Grid Array*, BGA), em que o processo de fixação do componente à carta encobre totalmente os pinos ou pontos de ligação, impossibilitam por completo o acesso físico.

Emulador de memória	Baixo. O emulador possui os recursos necessários para implementar as funções de depuração. Não são utilizados recursos do microprocessador ou do sistema. Podem ocorrer alguns efeitos (aumento das cargas nas linhas) que resultam da ligação física do emulador ao sistema.	Elevado / médio. Pode ser utilizado na depuração de problemas surgidos no ambiente de utilização normal, embora a ligação física possa não ser trivial. O emulador é independente do tipo de microprocessador, podendo ser utilizado em sistemas onde a aplicação a executar se encontra armazenada numa memória externa.	Médio / baixo. Dependendo da largura do barramento de dados, pode ser necessário utilizar uma ou mais pontas de ligação adicionais.
Emulador do microprocessador	Baixo. Não são utilizados recursos do microprocessador. Podem ocorrer alguns efeitos (aumento dos tempos de atraso e cargas nas linhas) que resultam da ligação física do emulador ao sistema. A ligação física da ponta do emulador pode requerer alguns cuidados na colocação dos componentes na CCI.	Elevado / médio. Pode ser utilizado na depuração de problemas surgidos no ambiente de utilização normal, embora a ligação física possa não ser trivial. A arquitectura do emulador depende do tipo do microprocessador instalado no sistema, podendo ser comum a uma mesma família. Apesar de a aplicação de depuração poder ser reutilizada, a ponta de ligação (de custo considerável) é específica.	Elevado. A parte física do emulador é geralmente a mais cara. Se o emulador suportar a amostragem de valores em tempo real, o custo será ainda maior.
Pré-processador ligado a um analisador lógico	Baixo. Podem ocorrer alguns efeitos (aumento dos tempos de atraso e cargas nas linhas) que resultam da ligação física ao microprocessador instalado no sistema. Não existe impacto ao nível do código da aplicação a executar.	Elevado / médio. Pode ser utilizado na depuração de problemas surgidos no ambiente de utilização normal, embora um analisador lógico seja um equipamento pesado, de dimensões apreciáveis e a ligação física possa não ser trivial. O nível de reutilização de um analisador lógico é elevado.	Médio. Um analisador lógico de "boa" qualidade pode ser caro. O custo da ponta de ligação e dos módulos de pré-processamento não é geralmente elevado.
Lógica embutida de apoio à depuração	Baixo. Utiliza recursos internos dedicados para as funções de depuração. Requer ainda um pequeno conector de ligação no sistema. Não existe impacto ao nível do código da aplicação a executar pelo microprocessador instalado no sistema.	Elevado. A infraestrutura de depuração é parte integrante do sistema, podendo ser utilizada ao longo de todo o ciclo de vida do produto. Um simples cabo permite ligar a aplicação de depuração, instalada num computador, à infraestrutura de depuração.	Médio / baixo. Principalmente associado à área de silício (ou da CCI) adicional, requerida pelo circuito de apoio à depuração. O custo da aplicação de depuração é geralmente baixo.

2.2.3 Protótipos de sistemas híbridos

Na análise seguinte consideram-se os níveis hierárquicos do CI e da CCI, para o protótipo de sistema híbrido sob depuração.

O nível hierárquico do CI, correspondente a um protótipo onde coexistem um ou mais núcleos de lógica dedicada, definida pelo utilizador, e um ou mais núcleos de microprocessador, apresenta dois desafios importantes para as ferramentas de depuração até agora apresentadas: a complexidade do sistema e o acesso físico. A inexistência de acesso físico ao núcleo de um microprocessador embutido no interior de um CI exclui por completo a possibilidade de utilização de ferramentas do tipo emulador de microprocessador, ou ainda de pré-processadores. A utilização de um emulador de memória obriga a que o microprocessador execute o seu programa a partir de uma memória externa, o que pode não ser possível para todos os casos, conso-

ante os requisitos inicialmente definidos na fase de especificação do sistema. As duas restantes ferramentas de depuração (programa monitor e emulador embutido) podem ser integradas no próprio CI, emergindo por isso como soluções possíveis a este nível hierárquico. Os recursos utilizados por um programa monitor e o seu limitado nível de reutilização, podem no entanto favorecer a opção pela inclusão no CI de lógica dedicada de apoio à depuração, conforme opinião partilhada por diversos projectistas [Mit97, Tuc97]. Em relação à depuração da totalidade do sistema, pode-se optar por um sistema de emulação com capacidades de depuração, ou alternativamente incluir no CI mais alguma lógica de apoio à depuração. A complexidade do sistema dificulta naturalmente a primeira solução[15]. Outro factor importante de entrave consiste no tipo de núcleo utilizado. Se este corresponder a um bloco com direitos de Propriedade Intelectual (*Intellectual Property*, IP), dificilmente se poderá obter um modelo ao nível da porta lógica, necessário para que o sistema de emulação cumpra a sua função. Uma solução para este caso consiste em colocar somente a parte do CI, relativa à lógica definida pelo utilizador, no sistema de emulação e ligá-lo posteriormente a um microprocessador encapsulado. A utilização de lógica dedicada de apoio à depuração, tanto para o núcleo do microprocessador como para o bloco de lógica definida pelo utilizador, representa no entanto a solução com maiores vantagens, especialmente no caso de se utilizar um único porto de acesso para a depuração, por exemplo compatível com a norma IEEE 1149.1. A Figura 2-13 ilustra precisamente esta opção, representando um CI que inclui no seu interior um núcleo-processador e um macrobloco de lógica de apoio à depuração, ambos comercializados pela *Advanced RISC Machines* (ARM) Inc., com ligação à infraestrutura de teste do CI [ARM99].

No nível hierárquico da CCI, apesar de poderem igualmente existir restrições ao acesso físico, consoante o tipo de encapsulamentos utilizados, é a complexidade do sistema que requer especial atenção, no que se refere à escolha das ferramentas de depuração mais adequadas. Esta complexidade é evidente no caso da CCI incluir vários componentes de lógica dedicada e um ou mais microprocessadores. A solução de repartir a complexidade do circuito implementado na CCI, por vários sistemas de emulação, colide com os enormes custos associados. Esta conclusão aplica-se ainda à solução de se utilizar um emulador ou um pré-processador por cada um dos microprocessadores existentes na CCI. No entanto, se os vários microprocessadores partilharem uma única memória, a utilização de um emulador de memória poderá representar uma solução aceitável. A utilização de lógica interna de apoio à depuração, em cada um dos microprocessadores e componentes de lógica dedicada, apresenta (mais uma vez) diversas vantagens:

---

[15] Emular um CI que no seu interior possui um ou mais núcleos de microprocessador pode ser considerado impossível em alguns casos, dependendo das capacidades de integração (e expansão) do sistema de emulação utilizado.

- Divide a complexidade. Cada microprocessador / componente de lógica dedicada possui a sua lógica de emulação interna. Se o porto de acesso a este recurso for compatível em todos os componentes com a norma IEEE 1149.1, é possível utilizar somente um pequeno conector de acesso na CCI, para efeitos de ligação à plataforma de depuração.
- Existem já casos relatados da utilização de lógica de emulação interna, acessível através de um porto de acesso compatível com a norma IEEE 1149.1, na depuração de sistemas com vários microprocessadores, que podem assim servir de guia de exemplo [Coo98, Mar96, Mon95, Pater98].
- À medida que a tecnologia evolui, as CCI de hoje vão-se tornando nos CI de amanhã, pelo que qualquer solução, para a depuração, baseada no acesso electrónico (ao invés do acesso físico), é facilmente integrável e por conseguinte utilizável após esse mesmo processo de integração. O número crescente de níveis hierárquicos pode no entanto levantar alguns problemas, em termos de utilização de portos de acesso normalizados, não existindo actualmente soluções eficazes para este problema emergente.

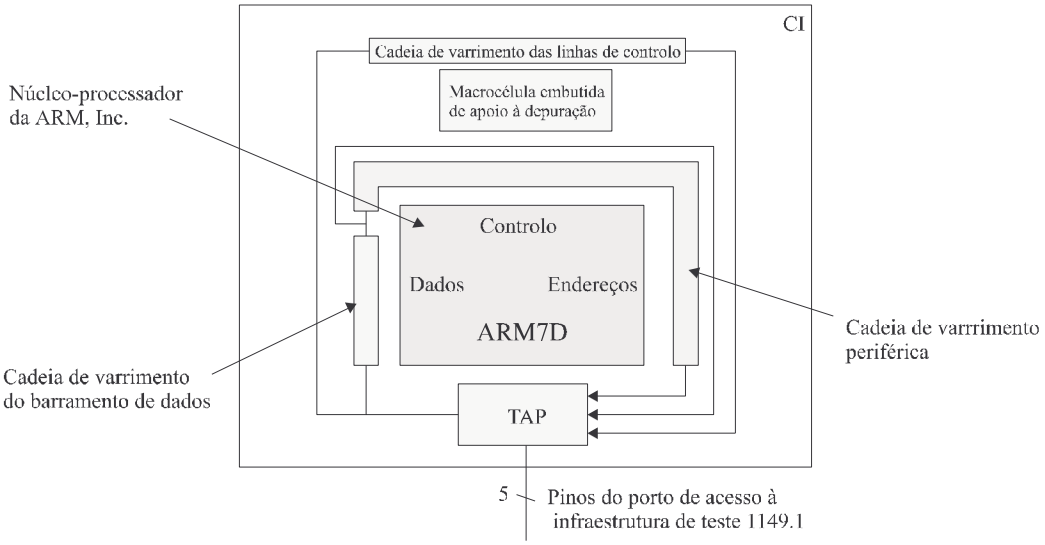


Figura 2-13: Macrobloco interno de apoio à depuração de aplicações baseadas num núcleo-processador, com ligação à infraestrutura de teste do CI [ARM99].

2.3 Operações e técnicas de depuração

Esta secção descreve um conjunto de técnicas de depuração para os três tipos de sistemas citados. Estas técnicas baseiam-se em quatro tipos básicos de operações, que se apresentam em primeiro lugar, geralmente associados à depuração de sistemas digitais. A depuração de sistemas mistos envolve um outro conjunto de operações, principalmente para a parte analógica, que excede largamente a complexidade do primeiro conjunto. Este facto deve-se à passagem

dos valores e do tempo de um universo discreto[16] para um universo contínuo. A análise das operações de depuração neste segundo universo extravasa por completo a dimensão deste capítulo, pelo que não serão referidas. A secção termina com uma breve abordagem ao problema da validação do protótipo, onde se refere a necessidade de efectuar operações de depuração quando o sistema se encontra a trabalhar no seu ambiente de utilização final.

2.3.1 Operações básicas e definição de um modelo de depuração

Apesar de todo o processo de depuração ser considerado complexo, os tipos de operações nele utilizadas reduzem-se geralmente a quatro categorias básicas, que se passam a apresentar:

- Controlo, Observação e Verificação (COV) de estados do sistema.
- Implementação do modo de funcionamento passo-a-passo (*Single-Step*, SS).
- Implementação de pontos de paragem por condição (*Breakpoint*, BP).
- Análise em tempo real.

Estes quatro tipos básicos de operações, ou modos básicos, fazem parte do modelo simplificado de depuração, ilustrado na Figura 2-14, onde as setas que unem cada um dos modos, representam sequências possíveis de utilização para execução de uma dada tarefa[17]. A complexidade do processo de depuração resulta do número de possíveis sequências de utilização destes tipos de operações e do número de condições associadas com cada tipo. No modo de funcionamento passo-a-passo existem algumas variantes para aquilo que se pode considerar como sendo *um passo*. No modo de funcionamento de pontos de paragem por condição existe uma quase infinidade de condições possíveis. No modo de análise em tempo real, baseado na amostragem de valores no sistema, existem iguais possibilidades para a definição de condições de início / fim de amostragem. Se o equipamento de aquisição de valores permitir ainda a especificação de filtros de amostragem, a complexidade aumentará ainda mais.

Descreve-se em seguida cada um dos modos básicos definidos, por forma a explicar com maior detalhe os vários termos e conceitos a eles associados.

---

[16] Os valores lógicos ‘0’ e ‘1’ passam a valores contínuos limitados por dois extremos. No que respeita ao tempo, deixam de se considerar apenas os momentos referentes às transições activas do relógio (para sistemas digitais síncronos).

[17] Por exemplo, especificar uma dada condição de paragem, deixar o sistema executar a sua função até que a condição se torne verdadeira, e a partir desse ponto executar a função passo-a-passo, observando após cada passo o estado de um determinado registo, posição de memória, ou de outro qualquer recurso do sistema.

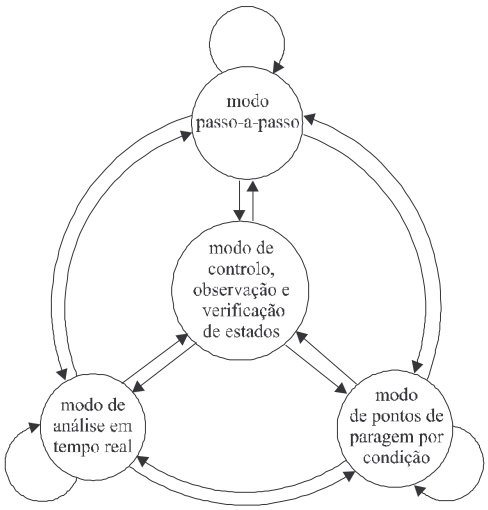


Figura 2-14: Modelo com os quatro tipos básicos de operações de depuração.

Operações de controlo, observação e verificação de estado

Este modo básico agrupa o conjunto de acções efectuadas para impor / observar / verificar o estado de um dado ponto do sistema, que dependendo do seu nível de abstracção, poderá ser entendido de diferentes formas. Por exemplo, pode-se referir ao estado de todo o sistema, de um bloco interno de um componente, de uma variável do programa executado por um microprocessador (que pode corresponder ao conteúdo de uma posição de memória ou de um seu registo interno) ou simplesmente ao estado da saída de uma simples porta lógica. A granulosidade deste tipo de operação depende assim não só do nível de abstracção considerado, como também dos mecanismos de acesso que existem no sistema sob depuração.

Enquanto que o objectivo da observação / verificação de estados é quase imediato (observar qual é o estado actual ou verificar se este é idêntico a um dado estado esperado), o objectivo do controlo de estados poderá não ser tão óbvio. Este é geralmente utilizado para acelerar[18] o processo de depuração ou para implementar acções do tipo "*e se?*", onde tipicamente se procura avaliar o comportamento do sistema após entrar em determinados estados de erro. Este modo básico é ainda extensivamente utilizado na implementação de testes estruturais do sistema. As operações individuais agrupadas sob a designação de operações de COV de estados podem ser obtidas através da aplicação dos seguintes critérios de divisão:

[18] Atingir um determinado estado do sistema pode implicar períodos de espera evitáveis, no caso de se utilizar uma operação de controlo de estado. Por exemplo, suponha-se que se pretende verificar a saída da função que indica que um contador atingiu o seu valor máximo. Dependendo do valor actual do contador, poderão ser necessários mais ou menos ciclos de relógio até que este atinja o valor máximo. No caso do contador ser implementado com elementos sequenciais acessíveis por varrimento, pode-se impor directamente esse estado, através do desbcamento do vector apropriado para o interior da cadeia de varrimento.

- Tipo de operação aplicada: controlo, observação ou verificação.
- Tipo de elemento ao qual a operação é aplicada: pino, elemento sequencial ou porta lógica.
- Tipo de acesso que existe ao elemento: directo, por varrimento (periférico) ou propagação.

No final, considerando cada combinação possível dos critérios apresentados, obtém-se 3^3, ou 27, operações individuais. Algumas destas combinações não fazem porém sentido, pelo que se fala apenas num subconjunto de operações. Cada uma das operações individuais é descrita nas Tabela 2-3, Tabela 2-4 e Tabela 2-5, referentes respectivamente a pinos, elementos sequenciais e portas lógicas.

Tabela 2-3: Operações individuais de COV do estado de pinos.

	Controlo	Observação	Verificação
Directo	Pinos correspondentes a entradas primárias da CCI são controláveis através de um Equipamento de Teste Automático (*Automatic Test Equipment*, ATE), ou outro equipamento similar. Outros pinos de componentes inseridos na CCI podem ser mais difíceis de controlar, seja porque durante o funcionamento normal se encontram num determinado estado lógico activo (obrigando o ATE a forçar o valor lógico – *backdriving*), ou porque não existe acesso físico (caso de encapsulamentos tipo BGA).	Pinos correspondentes a saídas primárias da CCI são observáveis através de um ATE, ou outro equipamento similar. Os valores presentes em outros pinos de componentes inseridos na CCI podem ser directamente observados através do mesmo equipamento, ou de um analisador lógico, desde que seja possível ligar fisicamente uma ponta de prova.	Requer um valor observado, um valor esperado e uma máscara de comparação. O resultado da comparação é verdadeiro se o valor observado for igual ao valor esperado, com a máscara activa. Máscaras inactivas tornam o resultado da comparação sempre verdadeiro.
Varrimento periférico	O valor lógico de um pino de saída de um componente BST é sempre controlável, de modo intrusivo, através da célula de varrimento associada. O valor lógico de um pino de entrada é apenas controlável, caso se encontre ligado a um pino de saída de um componente BST, ou caso o próprio componente suporte uma instrução opcional que permita este tipo de operação.	O valor lógico de um pino de um componente BST é sempre observável, de modo não intrusivo, através da célula de varrimento associada. Exceptuam-se os casos dos pinos de alimentação e do porto de acesso à infra-estrutura de teste, que não possuem células de varrimento associadas.	Idêntico ao anterior.
Propagação	Um pino de um componente não BST pode ser controlado por via da propagação do valor desejado através do seu cone de influência. Entende-se como cone de influência de um elemento todos os pinos, elementos sequenciais, ou portas lógicas, cujo valor influencia a saída do elemento considerado.	Um pino de um componente não BST pode ser observado por via da propagação do seu valor até um pino directamente acessível, ou por varrimento periférico, ou até um elemento sequencial acessível por varrimento.	Idêntico ao anterior (todas as operações de verificação seguintes repetem o especificado nesta tabela).

Tabela 2-4: Operações individuais de COV do estado de elementos sequenciais.

	Controlo	Observação
Varrimento	O elemento deve encontrar-se inserido numa cadeia de varrimento interna, acessível através da infraestrutura BST, ou através de pinos dedicados. O relógio que controla o deslocamento pode corresponder a um pino dedicado, ao pino de relógio do componente, ou ser derivado do relógio de teste. As memórias não são normalmente acessíveis por varrimento, dada a lógica adicional necessária.	O valor armazenado num elemento sequencial é observável na saída da cadeia de varrimento, após a aplicação de um determinado número de impulsos de relógio. A observação do conteúdo da cadeia, através de operações de varrimento, sem modificação do estado actual, requer que o valor deslocado do elemento mais próximo da saída seja novamente aplicado ao elemento mais próximo da entrada, durante um número de impulsos igual ao comprimento total da cadeia.
Propagação	Elementos sequenciais não acessíveis por varrimento podem ser controlados através da aplicação de sequências de Homing ou de sincronização [Lee96]. Estas sequências podem ser calculadas por computador, no caso dos elementos pertencerem a MEF. O cálculo de sequências de Homing requer que todas as saídas da MEF sejam observáveis. As sequências de sincronização não obrigam a este requisito, embora sejam mais difíceis de determinar, não existindo às vezes uma sequência possível para colocar a MEF num dado estado. Ambas as sequências são calculadas em relação às entradas da MEF. Se estas não forem acessíveis directamente, ou por varrimento, poderá ainda ser necessário propagar as sequências até esses mesmos pontos.	Elementos sequenciais não acessíveis por varrimento podem ser observáveis através da aplicação de sequências de distinção [Lee96]. Estas sequências podem ser calculadas por computador, no caso dos elementos pertencerem a MEF. O cálculo de sequências de distinção requer que todas as saídas da MEF sejam observáveis. A MEF pode ser recolocada no seu estado original, ou seja, antes da aplicação da sequência de distinção, através da aplicação de uma sequência de Homing ou de sincronização.

Tabela 2-5: Operações individuais de COV do estado de portas lógicas.

	Controlo	Observação
Propagação	O valor da saída de uma porta lógica pode ser controlado, através da aplicação de um determinado vector no seu cone de influência. O vector pode ser calculado com relação aos pinos ou elementos sequenciais mais próximos, acessíveis directamente ou por varrimento.	O valor da saída de uma porta lógica pode ser observado, através da sensibilização de todo o percurso lógico entre essa saída e um pino acessível directamente ou por varrimento, ou entre ela e um elemento sequencial acessível por varrimento. O vector de sensibilização pode ser calculado utilizando o algoritmo-D [Rot66] (ou outro similar), e posteriormente aplicado recorrendo a operações de controlo de pinos ou elementos sequenciais, acessíveis directamente ou por varrimento.

Operações passo-a-passo

Dependendo do nível de abstracção do sistema, este tipo de operação pode corresponder à execução de uma linha do programa (por exemplo, em linguagem C) de um microprocessador, de uma única instrução em *assembly*, ou ainda à aplicação de um único impulso de relógio. A re-

solução de uma operação passo-a-passo depende igualmente do tipo de ferramenta de depuração utilizada. Por exemplo, um programa monitor é capaz, no mínimo, de executar um única instrução, enquanto que um emulador de microprocessador, com circuito interno de geração de relógio, é capaz de diferenciar passos com uma resolução de um único ciclo de relógio. Independentemente do nível de abstracção, é possível decompor uma operação passo-a-passo na aplicação de um determinado número de impulsos de relógio, no caso de sistemas síncronos. Este modo básico é suportado por quase todas as ferramentas de depuração, com excepção dos analisadores lógicos e dos pré-processadores, que não possuem capacidades de controlo. A seguinte lista refere algumas das alternativas mais comuns para a implementação desta operação, que dependem do tipo de ferramenta utilizada:

- Aplicar um determinado número de ciclos de relógio, dependendo do tipo de sistema.
- Utilizar uma linha de interrupção. A linha encontra-se normalmente activa, forçando o microprocessador a executar a rotina de atendimento. Quando a linha é momentaneamente desactivada, o microprocessador executa uma única instrução do programa.
- Através de uma rotina especial de controlo, executar uma única instrução do programa do microprocessador, retornando de seguida a essa mesma rotina.

As operações passo-a-passo servem fundamentalmente para observar o funcionamento do sistema em velocidade lenta, controlada. Após a execução de um passo, a informação acerca do estado dos vários recursos do sistema é actualizada pela ferramenta de depuração, de forma a que o utilizador possa verificar se o funcionamento do sistema é ou não correcto. O tempo necessário para executar um passo e actualizar / verificar manualmente toda a informação acerca do estado do sistema, implica que o número de passos executados utilizando este tipo de operação é geralmente reduzido.

Operações de pontos de paragem por condição

Os pontos de paragem são expressos em termos de condições que, quando verdadeiras, levam a ferramenta de depuração a suspender o funcionamento do sistema, mantendo o seu estado actual. A implementação deste tipo de operação divide-se em três fases:

- Entrada ou especificação da condição do ponto de paragem.
- Avaliação do valor lógico da condição.
- Paragem do sistema, no momento em que a condição assume o valor lógico verdadeiro.

Especificação da condição

De forma idêntica ao modo de operação passo-a-passo, a especificação de uma condição de paragem depende do nível de abstracção do sistema. Por exemplo, pode corresponder a uma linha de um programa em linguagem C, a uma dada instrução em *assembly*, ou a um determinado vector, presente num conjunto de pinos do componente de lógica dedicada, ou num registo interno do microprocessador. As condições de paragem dependem ainda dos mecanismos existentes para a sua avaliação, podendo abordar-se este aspecto do ponto de vista estrutural, tomando como referência os seguintes exemplos de condições:

- A ocorrência de um determinado valor (vector) / sequência[19] num dado pino ou conjunto de pinos.
- A ocorrência de um determinado valor (vector) / sequência num dado elemento sequencial.
- A ocorrência de um determinado valor (vector) / sequência numa dada porta lógica ou conjunto de portas lógicas.

Em termos de mecanismos de avaliação, pode-se considerar a observabilidade como a regra principal a seguir, ou seja, o suporte de operações de observação (analisado anteriormente) influencia o conjunto de condições que se podem especificar num dado sistema, existindo assim uma relação intrínseca entre este dois aspectos.

Avaliação da condição

Uma condição é avaliada através da aplicação de depuração, ou através de circuitos dedicados implementados no interior do próprio sistema sob depuração. No primeiro caso, a condição é avaliada *enquanto* ou *antes* do sistema se encontrar a funcionar. Na primeira opção a ferramenta de depuração impõe o modo passo-a-passo, observando o estado do sistema após a execução de cada passo, para avaliar a condição. Se o valor lógico da condição for falso, executa um novo passo, caso contrário a ferramenta suspende o funcionamento do sistema, impedindo-o de sair do estado actual. Um programa monitor fornece um exemplo da forma como uma condição é avaliada antes do sistema entrar em funcionamento. Tipicamente, o monitor analisa o código do programa a executar pelo microprocessador e sempre que encontra uma instrução que torna verdadeira a condição especificada, troca essa instrução por uma outra apropriada, que retorna o controlo da execução ao programa monitor. A avaliação através de circuitos de-

[19] Uma dada condição (complexa) pode na realidade corresponder à associação, através de operadores lógicos, matemáticos ou temporais, de condições mais simples. Por exemplo, uma condição A pode corresponder à ocorrência do vector X OU vector Y (operador lógico) num dado conjunto de pinos. Outros exemplos incluem a ocorrência de um vector situado entre os limites X e Y, ou seja, vector X < vector actual < vector Y (operador matemático), ou ainda a ocorrência de um vector Y APÓS um vector X (operador temporal).

dicados de apoio à depuração é geralmente efectuada em tempo real, embora o número de condições que se podem especificar seja limitado pelos recursos disponibilizados para este fim. A Tabela 2-6 resume as formas alternativas da avaliação do valor lógico da condição de paragem.

Tabela 2-6: Formas alternativas de avaliação da condição referente a um ponto de paragem.

Avaliação da condição de paragem	Antes da execução	Durante a execução
Através da aplicação de depuração	O código do programa é analisado e todas as instruções que tornam verdadeira a condição especificada são substituídas por uma outra que retorna o controlo de execução, voltando a entregá-lo à aplicação de depuração. Condições que dependem de entradas do sistema não podem no entanto ser avaliadas antes da execução do programa.	Efectuado num modo passo-a-passo. A condição é avaliada após o sistema avançar um passo. Se a condição é falsa, o sistema avança um novo passo, se é verdadeira o sistema pára no seu estado actual. O número de possíveis condições depende dos mecanismos de observação existentes, embora seja possível construir associações complexas de diferentes condições.
Através dos circuitos dedicados de apoio à depuração	Inexistente.	Efectuado em tempo real. As condições restringem-se a sinais monitorados pelo circuito dedicado. Tipicamente, são suportadas condições simples, com possibilidade de mascarar alguns dos sinais.

Paragem do sistema

A última fase da implementação de um ponto de paragem depende igualmente do tipo de sistema e de ferramenta utilizada. Uma maneira simples de parar o funcionamento do sistema consiste em parar o fornecimento de impulsos de relógio. Esta abordagem pode no entanto causar problemas, no caso de existirem memórias dinâmicas que necessitem de refrescamento. Uma outra alternativa consiste em colocar o sistema em modo "inactivo", permitindo à ferramenta de depuração memorizar o estado que tornou verdadeira a condição de paragem e efectuar de seguida outras operações, antes de repor o sistema no estado memorizado.

Operações de análise em tempo real

A depuração de problemas que surgem quando o sistema se encontra a trabalhar, à sua velocidade de funcionamento normal, requer a utilização de operações de análise em tempo real. Note-se que a aplicação de um ponto de paragem por condição implica a paragem do sistema, podendo assim mascarar a origem do problema. As operações de análise em tempo real podem ser implementadas na forma de:

- Aquisição de valores em tempo real.
- Aplicação do algoritmo de alargamento de um impulso de relógio.
- Detecção / observação de tempos de atraso.

Aquisição de valores em tempo real

Observabilidade e memória são dois requisitos básicos da aquisição em tempo real. Por exemplo, um analisador lógico possui pontas de prova ligadas fisicamente aos sinais do sistema que se pretendem amostrar, capturando e armazenando os valores numa memória. Os valores podem ser capturados sincronamente com a operação do sistema (analisador lógico de estados), ou com o relógio interno do analisador (analisador lógico temporal). Uma outra alternativa consiste em implementar no interior do sistema uma memória dedicada (e um circuito de controlo), com as entradas de dados ligadas aos sinais que se pretendem amostrar em tempo real. Um segundo nível de requisitos, para este tipo de operação, inclui a possibilidade de se poder especificar condições de início / fim de amostragem (que necessitam igualmente de ser avaliadas em tempo real). Os analisadores lógicos possuem geralmente mecanismos poderosos de especificação deste tipo de condições, que incluem a possibilidade de o utilizador definir sequências de condições (numa forma análoga a uma máquina de estados). A Tabela 2-7 sumaria a análise comparada das duas alternativas referidas (analisadores lógicos e memória dedicada, implementada no sistema), através de parâmetros que incluem os requisitos de acesso físico, o tipo e dimensão da memória, a velocidade de amostragem e a especificação de condições de amostragem. Refira-se por último que os valores capturados podem ainda ser visualizados num estado isento de tratamento, ou após terem passado por um filtro, que inclui geralmente alguma forma de especificar o critério de selecção da informação que é relevante.

Tabela 2-7: Análise comparada de equipamentos / circuitos de aquisição de valores em tempo real.

Aquisição em tempo real	Características da memória	Tipo de acesso	Velocidade de amostragem	Condições de amostragem
Analisador lógico	Dimensão considerável, tanto em largura (número de entradas de dados) como em profundidade (número de posições). Tipicamente, dezenas de Kbits para cada sinal de entrada.	Requer acesso físico para ligação das pontas de prova aos sinais a monitorar.	Depende. É geralmente mais elevada no caso de um analisador lógico temporal, podendo atingir as várias centenas de MHz.	Possibilidade de especificar condições e associações de condições complexas, limitadas aos sinais amostrados.
Memória dedicada de armazenamento, implementada no sistema	Dimensões limitadas pela área e custo adicional aceitável.	Não requer acesso físico externo. As entradas de dados da memória são ligadas aos sinais a monitorar.	Idêntica à velocidade de funcionamento do sistema, desde que a memória possa trabalhar a essa velocidade.	Requer circuitos de controlo complexos, aumentando assim a área e o custo. Flexibilidade limitada.

Algoritmo de alargamento de um impulso de relógio

A controlabilidade do relógio e a observabilidade do estado interno do sistema são dois requisitos básicos para a aplicação do algoritmo de alargamento de um impulso de relógio, que consiste em alargar, selectivamente, um impulso isolado de relógio, num conjunto de ciclos anteriores à detecção externa de um erro. A detecção do erro tem lugar pela comparação em tempo real, entre os valores presentes e os valores esperados nas saídas do sistema. O racional deste algoritmo consiste em admitir que, ao alargar-se o impulso de relógio no qual o erro ocorre (devido a um tempo de atraso superior ao normal), se está a disponibilizar mais tempo para que o circuito estabilize, permitindo assim que os valores correctos sejam agora capturados. Este processo tem que ser executado iterativamente, com o alargamento de impulsos sucessivos, até que externamente deixe de ser detectado o erro. Quando isso acontece, o actual impulso alargado é aquele que exercita o percurso lógico onde ocorre o tempo de atraso superior ao previsto. Após a identificação do número de ordem do impulso alargado, aplica-se uma sequência de impulsos (não alargados) de relógio ao sistema, que termina precisamente no número de ordem identificado, após o qual se observa todo o estado interno do sistema, para localização do elemento que capturou o valor errado. Esta operação de análise em tempo real tem sido utilizada, com bons resultados, na depuração de vários microprocessadores [Hao95, Hol94, Kat94].

Detecção / observação de tempos de atraso

Determinados erros associados ao funcionamento em tempo real são causados pela existência de tempos de atraso, em interligações ou elementos activos, superiores aos normais ou inicialmente previstos. Estes tempos de atraso acrescidos podem ter origem em erros de projecto (um percurso lógico superior ao pretendido) ou faltas estruturais de tipo não catastrófico [20], como sejam ligações quase em aberto (aumento da impedância da ligação), ou quase em curto-circuito (capacidades parasitas). Os atrasos podem ser classificados com base nos pontos de origem e destino do percurso considerado, conforme é expresso na seguinte lista:

- Atraso entre pinos (interno ou externo).
- Atraso entre um pino e um elemento sequencial, ou vice-versa (interno).
- Atraso entre elementos sequenciais (interno).

Atrasos entre pinos podem corresponder a um único CI (atraso interno) ou a diferentes componentes (atraso externo, também designado por atraso de interligação). Os atrasos internos entre pinos são normalmente analisados e caracterizados durante o projecto do CI, sendo os

[20] Em contraposição com faltas do tipo catastrófico em que a detecção não depende da distância temporal entre a aplicação dos estímulos de teste e a captura das respectivas respostas.

atrasos externos analisados e caracterizados durante o projecto da CCI. Os atrasos entre pinos e elementos sequenciais (ou vice-versa), e os atrasos entre elementos sequenciais, referem-se sempre ao interior de um CI.

2.3.2 Sistemas baseados em lógica dedicada

Uma boa estratégia de depuração, para um sistema baseado em lógica dedicada, depende do fluxo de projecto. Frequentemente, o desenvolvimento de um sistema compreende o projecto simultâneo de um CI de lógica dedicada e da CCI onde este será implementado, juntamente com os restantes componentes que com ele interagem. Nesta situação, a estratégia de depuração necessita de acompanhar ambos os fluxos de projecto, influenciando assim a escolha das técnicas de depuração a utilizar.

O modelo de um CI atravessa vários níveis de abstracção durante a sua etapa inicial de projecto. A primeira técnica de depuração a seguir consiste em simular extensivamente o comportamento do CI em todos os seus níveis de abstracção, por forma a garantir que: (1) cumpre a sua função correctamente, (2) nos tempos previstos. Após a disponibilização do primeiro protótipo, seja através da utilização de um sistema de emulação, seja do próprio CI em si, a segunda técnica de depuração consiste em efectuar um teste estrutural exaustivo (ou completo), de forma a eliminar qualquer tipo de erro de fabrico ou de montagem, como por exemplo más ligações ou inserção incorrecta de componentes. Por outras palavras, deve-se validar a estrutura do circuito (tanto ao nível do CI, como ao nível do sistema), utilizando, na medida do possível, ferramentas de Geração Automática de Vectores de Teste (*Automatic Test Pattern Generation*, ATPG). A terceira técnica de depuração consiste em reutilizar os estímulos de entrada aplicados na simulação funcional, para efectuar um teste funcional, tanto do CI, como do sistema. As respostas obtidas na simulação funcional servem de termo de comparação para as respostas produzidas pelos circuitos reais. Após se ter garantido um funcionamento correcto, a última técnica de depuração consiste em verificar o comportamento em tempo real do CI e do sistema, que deverão funcionar à sua velocidade de operação normal. Estas quatro técnicas de depuração serão em seguida analisadas em maior detalhe, considerando a seguinte ordem:

- Simulação funcional / temporal extensiva.
- Teste estrutural exaustivo.
- Depuração funcional.
- Depuração temporal.

Simulação funcional / temporal extensiva

A disponibilização de um modelo funcional do CI (com a lógica dedicada), para integração no modelo utilizado para a simulação funcional do sistema onde será integrado, permite a recolha de dados que poderão ajudar no refinamento das especificações iniciais deste componente. O diagrama de projecto ilustrado na Figura 2-15 evidencia esta possibilidade, destacando ainda a ligação existente entre a simulação do CI e a simulação do sistema, utilizando modelos de nível abstracto elevado. No final deste processo de simulação funcional extensiva, obtém-se um ficheiro que se designa por *modelo dourado de estímulos / respostas*, que serve de referência para todas as etapas de simulação seguintes, efectuadas com modelos de nível de abstracção inferior. Este ficheiro serve ainda de termo de comparação para a depuração funcional do CI, quando este se encontra integrado no sistema.

Após se obter uma maior confiança na funcionalidade conjunta do CI e do sistema, nas etapas iniciais de simulação, o projecto do CI avança para níveis de abstracção inferiores, tipicamente para o nível RTL ou da porta lógica. Durante o processo de síntese, é possível utilizar ferramentas de inserção automática de infraestruturas de teste, cadeias de varrimento internas, mecanismos de auto-teste, etc[21]. A simulação deve, na medida do possível, incluir os circuitos de teste, implicando assim a geração de um novo conjunto de estímulos de entrada, para verificar não só o funcionamento dessa lógica, como ainda da sua interacção com a restante lógica do CI. No caso da lógica de teste ser alterada manualmente, após o processo de síntese, devem ser efectuadas novamente todas as etapas de simulação, para assegurar que nenhum erro foi introduzido em resultado dessa ou outra qualquer alteração. Efectuar a simulação em todos os níveis de abstracção constitui um dos passos mais importantes para a garantia de que o CI efectua correctamente a sua função, especialmente quando contém lógica de teste. Note-se que determinados tipos de erro podem não ser detectados / removidos ao nível RTL, uma vez que a lógica de teste não se encontra geralmente incluída a este nível de abstracção, sendo na maior parte dos casos incluída após o processo de síntese. A utilização de um sistema de emulação não contribui igualmente para a sua detecção / remoção, dado que tipicamente a lógica de teste não é incluída no processo de emulação.

A simulação temporal constitui, em paralelo com a utilização de ferramentas de análise temporal estática ou dinâmica, uma das formas de verificar que o CI cumpre a sua função, dentro dos parâmetros temporais especificados ou aceites pelo sistema. Esta tarefa requer tipicamente bastante memória, e capacidade / tempo de processamento, dependendo no entanto do nível de abstracção do modelo utilizado. Deve-se salientar neste ponto que a emulação dificilmente substitui a simulação temporal, apesar do sistema de emulação implementar o mesmo

[21] Para uma lista mais completa, consultar [Man96], ou o capítulo 4 do relatório sobre o estado da arte, incluído no CD-ROM que acompanha esta tese.

circuito ao nível da porta lógica, uma vez que os tempo de atraso são diferentes, quer para cada tipo de porta lógica, quer para as interligações existentes. Mesmo que o sistema de emulação possa trabalhar à mesma velocidade do CI, nada garante que o CI exibirá o mesmo comportamento, uma vez que os processos de colocação e ligação são diferentes para um e para outro tipo de circuito. A utilização de uma ferramenta de análise temporal estática é mais simples, embora esta necessite de um modelo ao nível da porta lógica. A identificação do percurso lógico mais extenso não é no entanto uma tarefa trivial, para os circuitos mais complexos. Para além disso, a natureza estática da análise pode negligenciar alguns tipos de atraso (nomeadamente ao nível das interligações), que são especialmente críticos em circuitos baseados em tecnologias sub-micrométricas. A utilização de uma ferramenta de análise temporal dinâmica, ou de uma ferramenta de extracção de parâmetros da topologia constituem duas possíveis soluções para este problema.

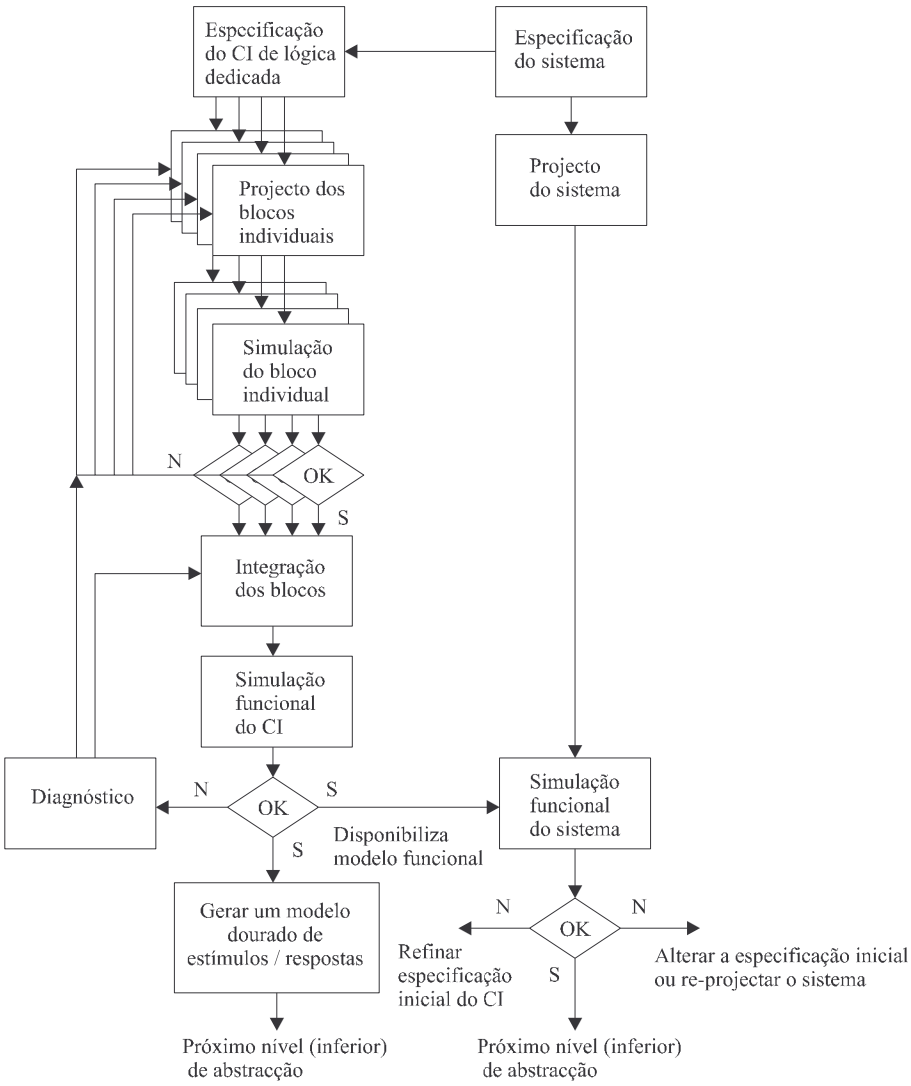


Figura 2-15: Etapas iniciais de simulação para um sistema baseado em lógica dedicada.

A verificação temporal requer geralmente bastante tempo e esforço, embora seja uma etapa crucial no processo de verificação do CI, especialmente nos casos em que existem requisitos deste tipo nas suas especificações iniciais. A verificação temporal da lógica de teste do CI é muitas vezes negligenciada, particularmente no que se refere à verificação da velocidade máxima a que podem funcionar as suas cadeias de varrimento, apesar de se conhecerem casos relatados de problemas surgidos por este motivo [Hol94].

Teste estrutural

A primeira acção de depuração física consiste em efectuar um teste estrutural exaustivo[22] do sistema. Assumindo que o sistema se situa ao nível da CCI, a sequência de teste a seguir depende da existência de CI com infraestruturas de teste compatíveis com a norma IEEE 1149.1. As seguintes situações são geralmente consideradas:

- Nenhum CI possui BST, o que corresponde à pior situação. O teste de interligações entre os componentes requer a utilização de um equipamento de teste por matriz-de-agulhas, ou de um equipamento de teste funcional. O primeiro necessita de aceder fisicamente aos nós do circuito, enquanto que a viabilidade do segundo depende largamente da complexidade do circuito a testar [Bat85, Cor87].
- Alguns CI possuem BST, o que corresponde a uma situação mista. O teste das interligações que envolvam unicamente pinos com células de varrimento periféricas associadas é geralmente simples. O teste de interligações que envolvam pinos sem células de varrimento periféricas associadas é um pouco mais complexo. A sequência de teste da CCI é faseada de acordo com uma metodologia que inclui o teste da infraestrutura em si, o teste das interligações e o teste dos componentes.
- Todos os CI possuem BST. Este é o caso mais simples, e portanto preferível, que favorece a utilização de ferramentas de ATPG, capazes de gerarem vectores de teste para as interligações, com uma cobertura de faltas na ordem dos 100% para o modelo de faltas individuais do tipo *Sempre-a* (*Single Stuck-at*, SS@), e / ou outros modelos de faltas. Se os CI suportarem funções de auto-teste, accionadas por uma instrução BST opcional, a complexidade / morosidade do teste decresce ainda mais, atingindo-se um ponto máximo de simplificação de todo o processo de teste estrutural.

[22] Entende-se por teste estrutural exaustivo aquele que utiliza conjuntos de vectores de teste gerados com base em diversos modelos de faltas (com um índice máximo de *cobertura de faltas* para cada um), de forma a aumentar o índice de *cobertura de defeitos* do sistema [Fra96, Hay85, Hug84, Max93, Sav91, Sto77].

Sistemas pertencentes a outros níveis hierárquicos, como por exemplo sistemas formados pela ligação num barramento comum de CCI, ou ainda sistemas integrados, requerem outras metodologias, que dependem igualmente da testabilidade do sistema. Desta forma, a utilização de uma metodologia de projecto para a testabilidade (*Design for Testability*, DfT) constitui o primeiro, e o melhor, passo no sentido de facilitar a tarefa da geração e execução do teste estrutural do sistema.

A melhor forma de precaver alterações ao teste estrutural, originadas pelos ciclos de re-projecto, consiste na utilização de ferramentas computadorizadas que garantam um processo simples e automático de geração do respectivo programa. Efectuar um teste estrutural exaustivo ao sistema, após a observação de um comportamento errático, ou após qualquer alteração física, constitui uma das técnicas de depuração mais eficazes, em virtude de frequentemente a causa do erro se relacionar com más ou deficientes interligações, ou devido à danificação dos circuitos que alimentam os pinos de saída dos CI, por exemplo, por efeito de excesso de carga, derivado de conflitos nas linhas ou barramentos.

Depuração funcional

Enquanto que o estado de um bloco de lógica combinatória é completamente definido pelos valores presentes nas suas entradas e saídas[23], o estado de um bloco de lógica sequencial é definido não só pelos valores presentes nas entradas e nas saídas, como ainda pelos valores presentes nos elementos de memória internos. Estes dois tipos de blocos estabelecem diferentes requisitos para a implementação de operações de COV. Num bloco de lógica combinatória é apenas necessário aceder às suas fronteiras (ou periferia) para completa observação do estado actual. Os requisitos de observabilidade para um bloco sequencial, por sua vez, incluem o acesso aos elementos de memória internos, por exemplo através da sua ligação numa cadeia interna. Apesar das técnicas de varrimento serem largamente usadas para controlar / observar o valor de registos internos, a lógica adicional associada limita a sua utilização em blocos de memória, onde é necessário recorrer a outros esquemas para observação do estado interno. Se existir um acesso directo às linhas externas do bloco de memória (endereços, dados e controlo), é possível ler o conteúdo de cada uma das posições, através da aplicação da combinação correcta nas entradas de endereços e de controlo, e observando nas linhas de dados o resultado obtido. No caso do bloco de memória se encontrar embutido no interior de um bloco mais

[23] Deve-se entender o termo "estado" como "estado correcto". As saídas de um bloco de lógica combinatória dependem dos valores presentes nas entradas, de acordo com a função definida pelo tipo de portas lógicas e com as ligações existentes entre elas. No caso de existir um erro que implique, por exemplo, a troca de um tipo de porta lógica por outro, os valores presentes nas saídas poderão diferir de um caso para o outro. Desta forma, apenas se podem tirar conclusões se se observarem em simultâneo os valores presentes nas entradas *e* nas saídas.

complexo, pode-se rodear as linhas de acesso à memória por uma cadeia de varrimento, unicamente para efeitos de COV do seu conteúdo.

A metodologia de depuração funcional assenta na capacidade de observar por completo o estado interno do sistema, qualquer que seja o seu nível hierárquico. Para o caso de um CI, refere-se em [Hol94] que se pode observar o seu estado interno através do deslocamento para o exterior dos valores presentes nas cadeias de varrimento interna e periférica. Inicialmente, efectua-se este processo de deslocamento no ambiente de simulação (utilizando o modelo do CI), de modo a criar um ficheiro que contém o estado correcto para todos os blocos internos do CI. Posteriormente, liga-se o CI a um equipamento de teste, que aplica os estímulos de entrada utilizados na simulação e efectua novamente o deslocamento dos valores capturados nas cadeias de varrimento, comparando-os com os existentes no referido ficheiro. O conteúdo deste ficheiro serve ainda como termo de comparação com os valores capturados / deslocados, quando o CI se encontra inserido no sistema. A Figura 2-16 ilustra o esquema subjacente a esta metodologia de depuração funcional, onde se pode verificar a dupla comparação existente entre os valores obtidos por simulação e: i) os valores obtidos quando o CI se encontra inserido no sistema; ii) os valores obtidos quando o CI se encontra ligado a um ATE. Quando um erro é detectado num dos domínios físicos, os vectores obtidos por varrimento permitem formar uma visão completa do estado interno do CI. A comparação desse estado com o estado esperado, obtido por simulação, serve de guia de auxílio para isolar a causa do erro.

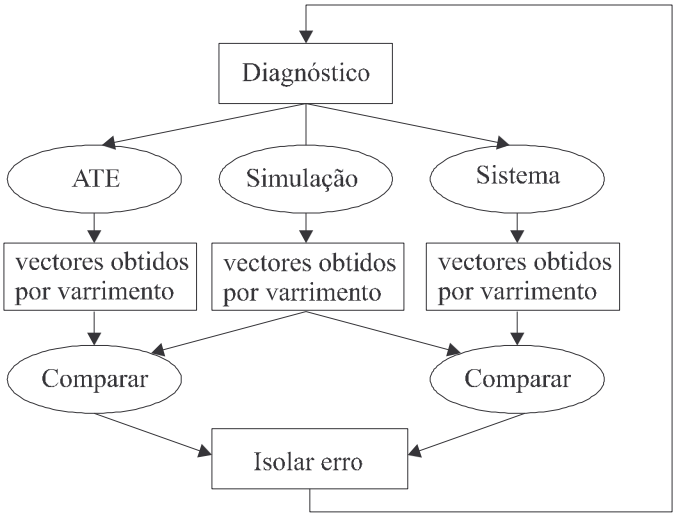


Figura 2-16: Metodologia de depuração funcional baseada em técnicas de varrimento [Hol94].

A metodologia de depuração funcional sofre uma ligeira adaptação, nos casos em que se utiliza um sistema de emulação em substituição de um protótipo do CI. Este tipo de ferramenta dispensa o acesso por varrimento a elementos sequenciais que se encontrem atribuídos a célu-

las lógicas ligadas a pinos de saída. O acesso simultâneo a elementos sequenciais referentes a células lógicas internas requer no entanto a utilização de técnicas de varrimento.

Após se ter isolado o erro detectado, e se ter identificado uma solução, importa confirmar esta conclusão, antes de se proceder em definitivo à alteração do projecto. No caso em que se utiliza um protótipo do CI, as potenciais soluções são implementadas e verificadas no próprio circuito através da utilização de um equipamento de Feixes-de-Electrões-Convergentes (*Focused-Ion-Beam*, FIB), para modificar as ligações ao nível das camadas de ligação de metal – uma técnica de custos elevados, que permite no entanto reduzir o número de vezes em que é necessário fabricar protótipos em silício [Hol94]. Contrastando com esta situação, emerge uma das principais vantagens do sistema de emulação, baseada na sua capacidade de implementar e verificar soluções possíveis, antes do fabrico dos primeiros protótipos, sem necessidade de equipamento adicional de elevados custos.

Depuração temporal

Uma das principais causas dos erros relacionados com a operação do sistema em tempo real consiste na existência de tempos de atraso nas portas lógicas, ou nas interligações (nos vários níveis hierárquicos), superiores aos inicialmente estipulados ou calculados. Um erro pode assim localizar-se numa porta lógica, numa interligação ao nível do CI ou numa interligação ao nível da CCI, levando ao aparecimento daquilo que se designa por *percurso lógico alongado.* Na maioria dos casos, ao nível do CI, um percurso lógico alongado termina num elemento sequencial interno. Quando um destes percursos é activado, o elemento sequencial situado no extremo final captura o valor errado, uma vez que o conjunto formado pela lógica e interligações que alimenta a sua entrada de dados, não teve tempo para estabilizar. Assim que o valor errado é capturado no ciclo de relógio n (por hipótese), vai influenciar a lógica colocada a jusante do elemento sequencial, propagando-se o erro em cada ciclo seguinte, até ser externamente detectado num pino de saída do CI, m ciclos após aquele (n) que activou inicialmente o percurso lógico alongado. Nesta altura, a maior parte do esforço da depuração temporal concentra-se na tarefa de isolar o ciclo exacto em que foi activado o percurso lógico alongado e qual o seu ponto de destino. Este último objectivo é atingido através da comparação entre os valores obtidos por simulação e os valores capturados no circuito, que são deslocados para o exterior através das cadeias de varrimento. Para isolar o ciclo de relógio no qual é activado o percurso lógico alongado, aplica-se o algoritmo de alargamento de um impulso de relógio, tal como foi já anteriormente descrito.

Na situação em que se utiliza um sistema de emulação é possível detectar o mesmo erro, no próprio ciclo de relógio em que o percurso lógico alongado é activado, desde que o elemento sequencial de destino esteja atribuído a uma célula lógica de um componente programável do

sistema, ligada a um pino de saída monitorado em tempo real por um analisador lógico. Este pino de saída pode ser monitorado directamente, ou através de um dos canais de observação existentes nos componentes do tipo FPIC. Apesar do emulador dispensar a aplicação do algoritmo de alargamento de um impulso de relógio, o número total de elementos sequenciais que podem ser monitorados em tempo real representa geralmente uma pequena fracção do número total existente no CI emulado.

2.3.3 Sistemas baseados em microprocessadores

O esforço principal na depuração de sistemas baseados em microprocessadores concentra-se geralmente na parte do programa executado pelo microprocessador, sendo menor para a parte da estrutura física do sistema. As ferramentas de simulação estabelecem uma primeira etapa de depuração do programa, permitindo controlar o seu fluxo de execução e examinar / modificar registos internos, portos de entrada e saída, e posições de memória interna do microprocessador. Quando o programa é executado no sistema físico, utilizam-se outras ferramentas de depuração, que incluem emuladores de memória, emuladores de microprocessador, analisadores lógicos, pré-processadores, ou lógica dedicada (interna do microprocessador) de apoio à depuração, que providenciam diferentes níveis de apoio à depuração do programa. As técnicas tradicionais de depuração de programas incluem a utilização extensiva de operações do tipo passo-a-passo, ou de pontos de paragem por condição. Contudo, para programas destinados a sistemas que necessitam de operar em tempo real, é necessário aplicar um outro tipo de técnicas de depuração, que incluem a amostragem em tempo real de todos os valores presentes nos pinos do microprocessador, e a captura / amostragem de informação do sistema antes / durante / após a ocorrência de determinadas condições [Bla95].

A tarefa de depuração de um sistema baseado em microprocessador carece ainda de uma metodologia modular / estruturada semelhante à apresentada para o caso de um sistema baseado em lógica dedicada. Esta circunstância pode explicar-se pelo facto de não existirem actualmente soluções estruturadas para o teste de programas (*software testing*), semelhantes às existentes para o teste de lógica / interligações (*hardware testing*). A depuração de um programa em execução no sistema final segue frequentemente um processo dedutivo, no qual o projectista tipicamente procura um sintoma claro, específico e repetitivo do erro, que o ajude a localizar e a diagnosticar a causa subjacente. No caso de o erro ser do tipo intermitente, este processo dedutivo resulta num verdadeiro desafio, mesmo para os projectistas mais experientes. A paragem da execução do programa mascara frequentemente a causa que provoca os erros deste tipo, sendo por isso necessário empregar técnicas de depuração em tempo real, que envolvem em alguns casos a especificação de condições de início / fim, ou de filtros de amostragem, de

forma a reduzir, ou filtrar, a quantidade de informação capturada. Deve-se realçar neste ponto que é impossível efectuar a depuração deste tipo de erros durante a simulação do programa, uma vez que estes se relacionam com violações temporais, que normalmente extravasam as capacidades dos simuladores utilizados.

Benefícios e limitações da simulação extensiva do programa

A simulação extensiva do programa estabelece uma primeira etapa de depuração, permitindo a eliminação de erros grosseiros e a detecção de potenciais causas de conflito, ou de ciclos infinitos. Esta etapa não utiliza qualquer tipo de ferramenta de depuração física, estando por isso sujeita às seguintes limitações:

- Equipamentos como teclados, monitores, impressoras, ou outros periféricos, não se encontram normalmente disponíveis num ambiente de simulação.
- Existem interacções complexas com o ambiente que rodeia o sistema, que não são possíveis de reproduzir num ambiente de simulação.
- Alguns programas para sistemas críticos em tempo real (por exemplo, uma aplicação de controlo para um equipamento nuclear) não podem ser depurados através da utilização de operações passo-a-passo, ou de pontos de paragem por condição.
- A velocidade é por vezes um factor crítico do sistema.
- Erros relacionados com violações temporais só ocorrem quando o sistema se encontra a trabalhar em tempo real.

Estas limitações obrigam habitualmente o projectista a utilizar ferramentas físicas de depuração, para verificar o programa, quando este é executado no sistema.

Operações passo-a-passo e de pontos de paragem por condição

A aplicação de operações passo-a-passo e de pontos de paragem por condição, constitui uma forma tradicional de verificar o programa executado no sistema físico, que permite ultrapassar as duas limitações anteriormente referidas para os ambientes de simulação. Contudo, nos casos em que o programa é executado a partir de uma memória não-volátil, não se pode utilizar uma ferramenta do tipo programa monitor, uma vez que esta requer que o programa seja armazenado numa memória do tipo volátil, de forma a poder implementar estes dois tipos básicos de operações de depuração. Um emulador de memória não apresenta este problema, dado que substitui a memória do programa. No entanto, esta ferramenta de depuração apenas suporta um conjunto limitado de condições de paragem, ou seja, apenas suporta condições em que os valores especificados dependem de acções que incluem acessos à memória do programa. Uma sim-

ples condição de paragem, referente ao aparecimento de um determinado valor nos pinos de entrada / saída do microprocessador, não directamente ligados à memória do programa[24], não é suportada por este tipo de ferramenta de depuração.

Outro aspecto importante consiste na capacidade de parar a actividade de todo o sistema após a execução de um passo, ou da ocorrência de uma condição de paragem verdadeira. Tanto o programa monitor como o emulador de memória não possuem as capacidades necessárias para parar o relógio do microprocessador. Estas ferramentas apenas são capazes de colocar o microprocessador num estado em que não executa mais nenhuma instrução do programa principal, "parando" assim a sua execução. Um emulador de microprocessador (e em certos casos a lógica dedicada de apoio à depuração), que possui o seu próprio circuito de geração de relógio, é capaz de parar o fornecimento de impulsos de relógio ao sistema, nos casos em que este é comum ao (ou derivado do) relógio do microproce ssador.

Apesar das técnicas de depuração baseadas na utilização de operações passo-a-passo e de pontos de paragem por condição, serem amplamente utilizadas, alguns programas não podem ser verificados unicamente através destas técnicas, principalmente nos casos em que o sistema não possa ser parado, como por exemplo, em sistemas de controlo de motores ou de processos químicos. A natureza de funcionamento em tempo real destes sistemas requer a utilização de outras técnicas que servem igualmente o propósito da depuração de erros relacionados com violações temporais, em sistemas que não partilham do requisito de funcionarem ininterruptamente em tempo real.

Aquisição de valores em tempo real antes / durante / após condição e amostragem por evento

A aquisição de valores em tempo real permite registar a actividade do microprocessador, enquanto o sistema trabalha ininterruptamente à sua velocidade máxima de operação. A informação capturada é normalmente armazenada numa memória, que funciona como uma pilha de retenção de dados, em que o primeiro valor armazenado corresponde ao primeiro a ser lido (*First-In First-Out*, FIFO). Dado que estaremos tipicamente a considerar a execução de milhões de instruções por segundo, torna-se necessário utilizar condições para especificar o início e o fim da amostragem. Refira-se, no entanto, que esta técnica de depuração é não intrusiva, servindo apenas para capturar e visualizar informação relativa à execução do programa enquanto este continua, sem interrupção, com as instruções a serem executadas à cadência normal. Desta forma, o utilizador pode efectuar a depuração de erros intermitentes, ou de outros

[24] Ou em que o valor não depende directamente do acesso à memória do programa (por exemplo, note-se a diferença entre a execução de uma instrução que coloca num porto de saída do microprocessador um valor armazenado na memória do programa, ou um valor armazenado num registo interno).

que apenas se manifestem quando o sistema se encontra a trabalhar em tempo real, sem parar a execução do programa com a aplicação de pontos de paragem por condição.

Outra técnica de depuração, habitualmente referida como *amostragem por evento*, consiste em capturar o conteúdo de determinados registos internos do microprocessador, após a ocorrência de um determinado evento, definido como uma condição interna, ou em alguns casos como uma condição externa. Os recursos necessários para implementar esta técnica incluem registos paralelos, que capturam o conteúdo dos registos a que estão associados, após a condição especificada ter assumido o valor lógico verdadeiro. A implementação de registos paralelos implica uma área de silício adicional, o que restringe a sua aplicação a um reduzido número de registos internos do microprocessador.

2.3.4 Sistemas híbridos

Uma das dificuldades principais associadas à depuração de sistemas híbridos consiste em verificar a interacção entre o núcleo embutido do microprocessador e a parte correspondente à lógica dedicada, ou lógica definida pelo utilizador. O processo de verificação da parte definida pelo utilizador segue a metodologia definida para o caso dos sistemas baseados em lógica dedicada (subsecção 2.3.2): simulação extensiva nos vários níveis de abstracção, teste estrutural, depuração funcional e temporal. A verificação do programa executado pelo microprocessador depende do tipo de núcleo utilizado, nomeadamente quanto à sua classificação em termos de IP. Actualmente, propõe-se uma classificação básica em três níveis [Lew96]:

- HIPB (*Hard IP Block*).
- FIPB (*Firm IP Block*)
- SIPB *(Soft IP Block*)

Cada um destes tipos de blocos colocam diferentes desafios ao processo de verificação do CI. Os HIPB correspondem a blocos em silício, com as ligações já definidas (entre as portas lógicas que os compõem), já testados e verificados para um dado processo de fabrico. Alguns autores classificam-nos ainda como macrocélulas específicas de uma dada tecnologia de implementação. Os modelos funcionais e temporais encontram-se normalmente disponíveis, permitindo ao utilizador a correspondente simulação de todo o CI. O fabricante assegura igualmente a exactidão estrutural do núcleo do microprocessador. Os FIPB e os SIPB são vendidos ou licenciados sob a forma de uma descrição funcional ou comportamental (geralmente encriptada), utilizável nos ambientes de simulação e síntese. As características temporais dos modelos comportamentais não são definitivas, dado que o núcleo será ainda sujeito ao processo de colocação e ligação, pelo qual passa igualmente toda a restante lógica do CI. Apesar de se poderem utilizar directivas para a planificação do processo de colocação e ligação, as caracterís-

cas temporais finais podem, ainda assim, ser ligeiramente diferentes. O problema da atribuição da responsabilidade do teste estrutural do núcleo do microprocessador, no caso deste ser fornecido como um FIPB ou SIPB, não se encontra ainda resolvido. As companhias que fabricam os CI, e as que comercializam as respectivas ferramentas de desenvolvimento, trabalham actualmente juntas num consórcio denominado *Virtual Sockets Interface* (VSI) *Alliance*, na resolução deste e de outros problemas relacionados, esperando-se resultados para meados do corrente ano. A verificação do programa executado pelo núcleo do microprocessador, qualquer que seja a sua classificação, é no entanto facilitada no caso de existir lógica dedicada de apoio à depuração, quer seja implementada no interior do próprio núcleo, quer seja no exterior[25] [Gou98]. Outras alternativas passam pela ligação das E/S do núcleo embutido do microprocessador a pinos do CI ou a colocação de um colar de células de varrimento à volta dessas E/S. Estas duas alternativas possuem no entanto algumas desvantagens. A primeira implica a necessidade de um maior número de pinos no CI (que poderá ultrapassar o número máximo disponível, ou mesmo admissível, para o tipo de encapsulamento escolhido). A segunda implica mais lógica adicional e um aumento no tempo de acesso às E/S do núcleo, afectando assim o desempenho do CI na sua totalidade.

A depuração de sistemas híbridos ao nível da CCI não enfrenta os mesmos problemas de inexistência de acesso físico ao microprocessador, embora possam subsistir algumas dificuldades ou restrições, mais ou menos importantes. A simulação de todo o sistema depara provavelmente com o problema da sua enorme complexidade, embora em contrapartida o teste estrutural seja mais fácil, especialmente no caso em que os vários componentes inseridos na CCI suportam a infraestrutura BST. Enquanto a depuração da lógica do sistema segue a metodologia estruturada definida anteriormente (subsecção 2.3.2), a depuração do programa segue uma vez mais um processo manifestamente dedutivo, cujo grau de sucesso depende das capacidades das ferramentas utilizadas.

No final, o problema da verificação dita que se siga uma metodologia de projecto para o teste e depuração (*Design for Debug and Test*, DfDT), remetendo-se para segundo lugar questões como a quantidade de lógica ou o tempo de projecto adicional, já que em caso contrário se excederá muito provavelmente o período de tempo inicialmente estipulado para esta fase do desenvolvimento, afectando assim negativamente a colocação do sistema no mercado.

2.4 Conclusão

A primeira técnica de depuração consiste na simulação extensiva do sistema, utilizando os vários modelos que acompanham, com um nível de abstracção decrescente, a sua descrição, des-

[25] Exterior ao núcleo, mas interior ao CI.

de a etapa inicial da especificação até à etapa da implementação. Ao nível da simulação funcional, é geralmente detectada e eliminada a maior parte dos erros triviais. Após esta fase, é normal que os projectistas comecem a sentir alguma confiança na funcionalidade do sistema, avançando o projecto para níveis de abstracção inferiores, tipicamente para o nível da porta lógica, dando assim lugar à simulação temporal. A exactidão do modelo temporal depende do tipo de atrasos considerados para as portas lógicas e da inclusão, ou exclusão, dos atrasos inerentes às ligações entre elas. A complexidade do modelo cresce, provocando um impacto negativo no tempo de simulação. Os recursos computacionais e o tempo necessário para simular o comportamento do sistema impelem os projectistas a utilizarem apenas um subconjunto dos estímulos de entrada usados na simulação funcional.

Após a disponibilização dos primeiros protótipos físicos, efectuam-se os testes iniciais com base nos estímulos de entrada utilizados durante a simulação funcional. Se não existir qualquer tipo de diferença nas respostas obtidas, pode-se afirmar que o protótipo inicial está validado, perante a simulação efectuada. Se existirem diferenças, procede-se à sua depuração, até que as causas sejam localizadas e corrigidas. Na maioria dos casos os protótipos são verificados em laboratório, onde se encontram instalados os vários equipamentos utilizados para o teste e a depuração. As condições ambientais e de utilização existentes num laboratório diferem geralmente das condições em que o sistema opera no seu ambiente de utilização final. Mesmo que se tentem reproduzir essas condições, dificilmente se consegue garantir uma igualdade perfeita. A verificação do funcionamento do protótipo do sistema, no seu ambiente de utilização final, tende a ser descurada por alguns projectistas, originando situações em que só mais tardiamente, já com a etapa de desenvolvimento concluída, se detectam alguns tipos de erros. Se o sistema incluir lógica dedicada de apoio à depuração, pode-se reutilizar este recurso para localizar e diagnosticar os erros detectados no ambiente de utilização final [Whi96].

Após a detecção de um erro, a equipa responsável pela verificação do sistema tenta localizar e diagnosticar a sua causa, por forma a que a equipa de projecto possa idealizar e implementar uma solução. O tipo de sistema sob desenvolvimento, as ferramentas de projecto e depuração utilizadas e a metodologia seguida, influenciam largamente este processo de detecção / localização / diagnóstico / idealização e implementação de uma solução, que se repete ciclicamente, até ao ponto em que mais nenhum erro é detectado. Para um sistema baseado em lógica dedicada, o processo de localização inclui um deslocamento do conteúdo de todos os elementos que definem o estado interno do sistema, ou, na situação em que se utiliza um sistema de emulação, a observação do estado de pontos suspeitos do circuito. Os erros associados a violações temporais, ou que só ocorrem quando o sistema se encontra a trabalhar à sua velocidade normal de operação, requerem técnicas de depuração baseadas na aplicação do algoritmo de alargamento de um impulso de relógio, para sistemas que incluem cadeias de varrimento internas, ou quando se utilizam sistemas de emulação, na aquisição de valores em tempo real através de um

analisador lógico. Embora seja necessário aceder a estes sinais a partir do exterior, é relativamente simples passar de células lógicas internas para células lógicas associadas a pinos de saída dos componentes programáveis, ou alterar a configuração das linhas de monitorização dos componentes com matrizes de ligação programáveis. Repare-se ainda que as soluções idealizadas podem ser avaliadas mais rapidamente num sistema de emulação, do que através da sua implementação no próprio sistema, especialmente no caso de um CI, que requer a utilização de equipamentos de elevado custo. Se o custo de desenvolvimento for um factor menor, então pode-se afirmar que a melhor solução consiste numa estratégia mista, que inclua lógica dedicada de apoio à depuração e que utilize um sistema de emulação para avaliar o comportamento do sistema, antes de se produzirem os primeiros protótipos físicos. A Figura 2-17 ilustra precisamente estas alternativas, para o fluxo do processo de verificação de sistemas baseados em lógica dedicada.

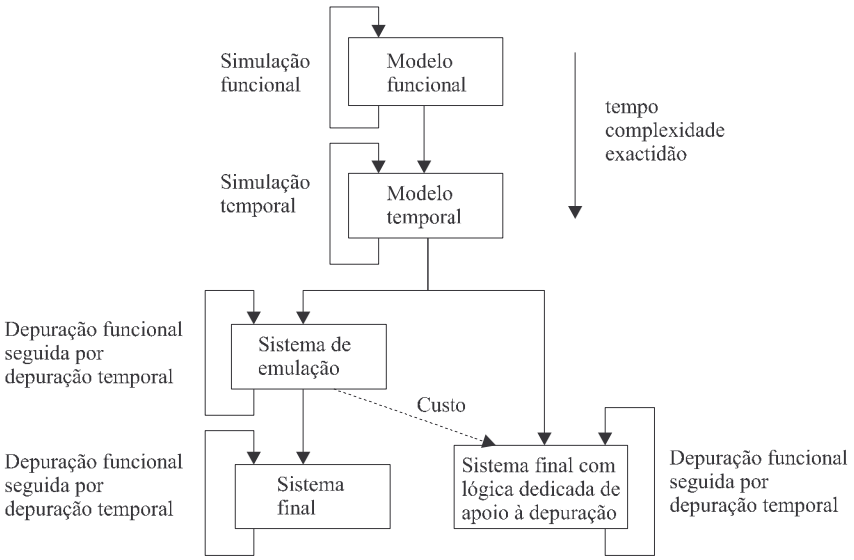


Figura 2-17: Alternativas para o fluxo de depuração de sistemas baseados em lógica dedicada.

A depuração de um sistema baseado em microprocessador incide com maior relevância no programa executado, que determina a funcionalidade do sistema. A interacção entre o microprocessador e o resto do sistema é simulada, recorrendo-se a um modelo da interface com os barramentos, e a execução do programa é simulada utilizando-se um modelo de simulação do conjunto de instruções do microprocessador. A depuração do programa, quando este é executado no sistema físico, requer a utilização de ferramentas do tipo programa monitor, emulador de memória, emulador de microprocessador, pré-processador ligado a um analisador lógico, ou lógica dedicada de apoio à depuração. Cada uma destas ferramentas possui vantagens e desvantagens, que devem ser avaliadas por forma a seleccionar-se aquela ou aquelas que melhor

servem os requisitos de depuração. A depuração funcional baseia-se, na maior parte dos casos, em operações passo-a-passo e na aplicação de pontos de paragem por condição. O número máximo de condições suportadas constitui claramente, para esta fase, um factor de distinção entre as várias ferramentas referidas. Para a depuração temporal, apenas algumas ferramentas possuem as capacidades necessárias para ajudar na localização e diagnóstico de erros intermitentes, relacionados com violações temporais, ou que só se manifestam quando o sistema se encontra a trabalhar em tempo real.

A depuração dos sistemas híbridos requer uma metodologia mista, que integra uma metodologia estruturada para o caso da componente baseada em lógica dedicada, e um processo de índole francamente dedutiva, para a verificação do programa, que depende nomeadamente da utilização das ferramentas mais apropriadas e do conhecimento específico acerca da funcionalidade do sistema. A utilização de metodologias de projecto para o teste e depuração, que resultem na implementação de lógica de apoio à depuração, tanto ao nível do microprocessador como ao nível dos componentes ou blocos de lógica dedicada, permite criar um ambiente de depuração poderoso, utilizável tanto para a depuração da parte física como para a depuração do programa, nas várias etapas do ciclo de vida.

Capítulo 3

As infraestruturas IEEE 1149.1 e P1149.4 como veículo de depuração

Este capítulo descreve de forma concisa a arquitectura e os modos de funcionamento das infraestruturas de teste definidas pela: a) norma IEEE 1149.1; b) proposta de norma IEEE P1149.4. Com base na descrição efectuada, analisam-se as capacidades ou formas de utilização de cada infraestrutura no apoio às operações de depuração identificadas no capítulo anterior. A estrutura deste capítulo assenta em três eixos principais:

- Tipo de infraestrutura / modo de funcionamento.
- Tipo / hierarquia do sistema sob depuração.
- Tipo de operação depuração a efectuar.

Dada a dimensão da análise a efectuar, divide-se o capítulo em dois blocos, um referente ao 1149.1 (quatro secções) e outro referente ao P1149.4 (uma única secção, que congrega tudo aquilo que se considera mais relevante para o 1149.1). A primeira secção descreve detalhadamente a arquitectura da infraestrutura BST. A segunda secção descreve os modos de funcionamento básicos desta infraestrutura, baseados nas instruções obrigatórias definidas na norma, a que se seguem os que são baseados em instruções opcionais. A terceira secção descreve a implementação das operações de depuração, através dos modos de funcionamento básicos, para os vários tipos de sistema (baseados em lógica dedicada, em microprocessadores, ou híbridos). A quarta secção descreve a implementação das mesmas operações de depuração, através dos modos de funcionamento opcionais. A quinta secção resume todos os aspectos anteriores para a infraestrutura P1149.4: apresentação da arquitectura básica e blocos principais, modos de funcionamento básicos e opcionais, e sua utilização na implementação das operações de depuração, neste caso em simultâneo para os vários tipos de sistemas.

3.1 A infraestrutura 1149.1

A infraestrutura de teste mínima representada na Figura 3-1 é constituída pelo controlador do Porto de Acesso à lógica de Teste (*Test Access Port*, TAP), registo de Varrimento Periférico (*Boundary Scan*, BS), registo série de atalho (*bypass)*, registo de instrução e bloco de descodificação. Identificam-se ainda os quatro sinais que constituem o interface para o exterior da infraestrutura de teste. Um quinto sinal opcional (*Test Reset*, /TRST) permite fazer a inicialização assíncrona da lógica de teste. Dois destes sinais, o de Selecção do Modo de Teste (*Test Mode Select*, TMS) e o Relógio de Teste (*Test Clock*, TCK), servem para comandar e sincronizar a lógica de teste. O sinal de Entrada de Dados de Teste (*Test Data In*, TDI) permite deslocar dados (em formato série) para o interior do CI e o sinal de Saída de Dados de Teste (*Test Data Out*, TDO) permite deslocar dados para o seu exterior. A constituição e funcionalidade de cada um dos blocos representados na Figura 3-1 é desenvolvida a seguir.

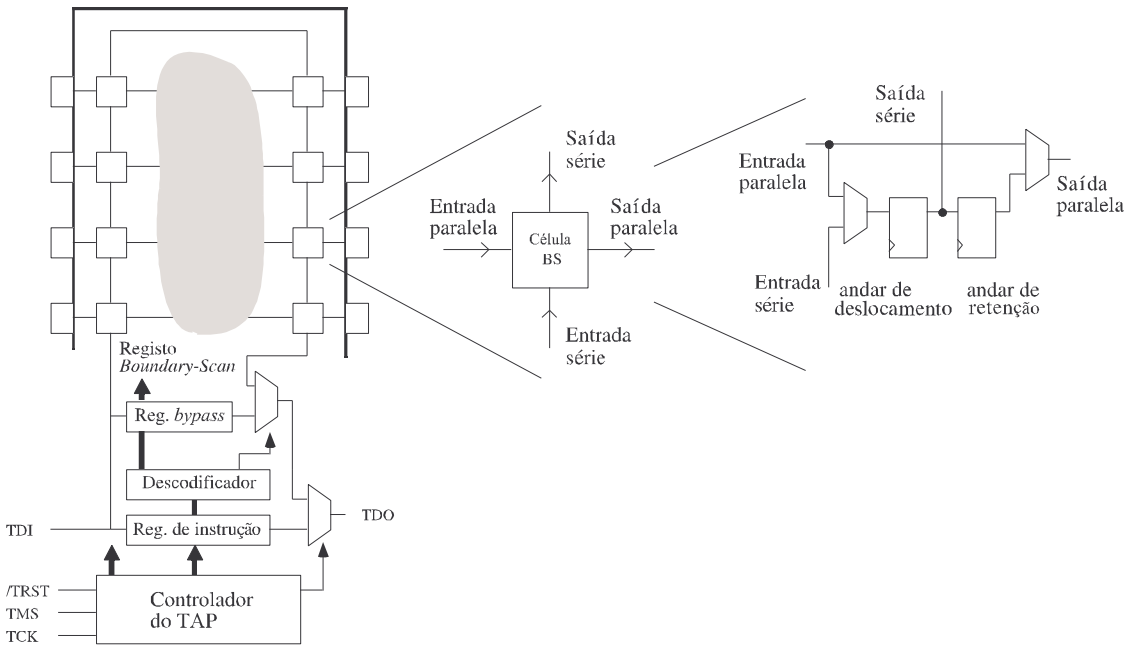


Figura 3-1: Infraestrutura de teste mínima definida na norma IEEE 1149.1.

3.1.1 O controlador do TAP

O controlador do TAP é uma máquina de estados, cujo diagrama se representa na Figura 3-2, com as transições entre os dezasseis estados comandadas pela entrada TMS e cadenciadas pela transição ascendente de TCK. A máquina de estados pode ser dividida em três blocos: um anel superior de selecção que engloba os estados *Test-Logic-Reset*, *Run-Test/Idle*, *Select-DR* e *Select-IR*; o ramo dos registos de dados, que engloba os estados *Capture-DR*, *Shift-DR*, *Exit1-DR*,

Pause-DR, *Exit2-DR* e *Update-DR*; e o ramo do registo de instrução, que engloba os estados *Capture-IR*, *Shift-IR*, *Exit1-IR*, *Pause-IR*, *Exit2-IR* e *Update-IR*. Uma descrição breve do significado e propósito de cada estado dá uma ideia mais clara do respectivo protocolo de controlo:

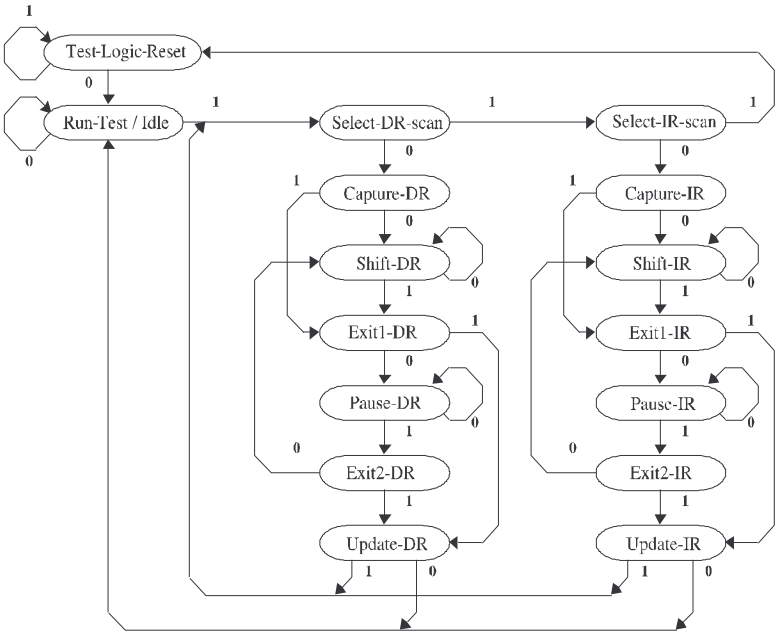


Figura 3-2: Diagrama de transições do controlador do TAP.

- *Test-Logic-Reset* - A lógica de teste encontra-se inactiva, não interferindo com o funcionamento normal do CI. Este estado pode ser alcançado através de um impulso ao nível baixo na linha TRST, ou após cinco transições ascendentes em TCK, com a linha TMS no estado alto.
- *Run-Test/Idle* - O estado da lógica de teste depende da instrução actual, que poderá ou não activar funções de teste.
- *Select-DR* - Este estado temporário permite, no caso de a linha TMS estar no estado baixo, seleccionar o ramo dos registos de dados de teste. A lógica de teste encontra-se inactiva.
- *Capture-DR* - A instrução actual determina qual o registo de dados que irá ser afectado. O Andar de Deslocamento (A_D) das células do registo seleccionado captura o valor presente nas suas entradas paralelas.
- *Shift-DR* - O conteúdo do A_D do registo seleccionado pela instrução actual é deslocado um bit por cada transição ascendente do relógio de teste, no sentido de TDI para TDO.
- *Update-DR* - Neste estado, o Andar de Retenção (A_R) das células do registo de dados seleccionado captura o valor presente no A_D.
- *Select-IR*, *Capture-IR*, *Shift-IR* e *Update-IR* - Estes estados são análogos aos estados *Select-DR*, *Capture-DR*, *Shift-DR* e *Update-DR*, com a única diferença de dizerem respeito ao re-

gisto de instrução. A nova instrução deslocada no estado *Shift-IR* fica activa à passagem pelo estado *Update-IR*.

- Nos estados *Exit1-DR*, *Exit1-IR*, *Pause-DR*, *Pause-IR*, *Exit2-DR* e *Exit2-IR*, a lógica de teste encontra-se inactiva, servindo estes estados para efeitos de pausa ou como pontos de decisão.

3.1.2 Os registos de dados

A norma define os vários registos de dados de teste: o registo *bypass* (obrigatório), o registo BS (obrigatório), o registo de identificação do CI (opcional) e uma classe de registos de dados de teste definidos pelo fabricante (opcionais).

O registo BS é constituído pelas células BS associadas aos pinos funcionais do CI. Os únicos pinos que não possuem estas células são os pinos de alimentação e os que constituem o TAP. As células BS podem ser representadas na forma de um bloco, como se ilustra na Figura 3-1. Consoante o tipo de pino (bidireccional, entrada, saída com ou sem controlo de estado), este pode ter uma ou mais células associadas. Embora existam algumas variações, as células BS são geralmente constituídas pelo circuito representado igualmente na Figura 3-1. São possíveis quatro modos de funcionamento, definidos pela instrução activa e pelo estado em que se encontra o controlador do TAP, tal como se ilustra na Figura 3-3:

- Controlabilidade - O valor aplicado num pino de saída ou numa entrada da lógica funcional do CI é determinado pela célula BS.
- Observabilidade - A célula BS captura o valor presente num pino de entrada ou numa saída da lógica funcional do CI.
- Deslocamento - O conteúdo do registo BS, formado por todas as células BS, é deslocado no sentido de TDI para TDO.
- Transparência - A célula BS não afecta o funcionamento normal do CI (a entrada paralela liga à saída paralela).

O registo *bypass* é constituído por um único elemento sequencial, que forma um percurso de comprimento mínimo entre os pinos TDI - TDO do CI. O registo de identificação permite identificar electronicamente um determinado componente. É possível assim, ao seleccionar este registo em todos os CI presentes numa determinada cadeia BS, verificar se não existem trocas de componentes e identificar qual o fabricante e versão de cada CI. O fabricante do CI pode ainda definir um conjunto adicional de registos de dados de teste, acessíveis através da infraestrutura BST. A norma não obriga o fabricante a publicar a maneira de aceder a estes registos,

dando-lhe assim a possibilidade de os utilizar para o teste de produção. Estes registos podem pertencer a uma ou várias das categorias a seguir enumeradas:

- Registos com o resultado de operações de auto-teste.
- Registos de cadeias de varrimento internas.
- Registos de identificação do utilizador (ex.: componentes lógicos programáveis).
- Registos de geração de estímulos pseudo-aleatórios, que necessitem de uma semente.

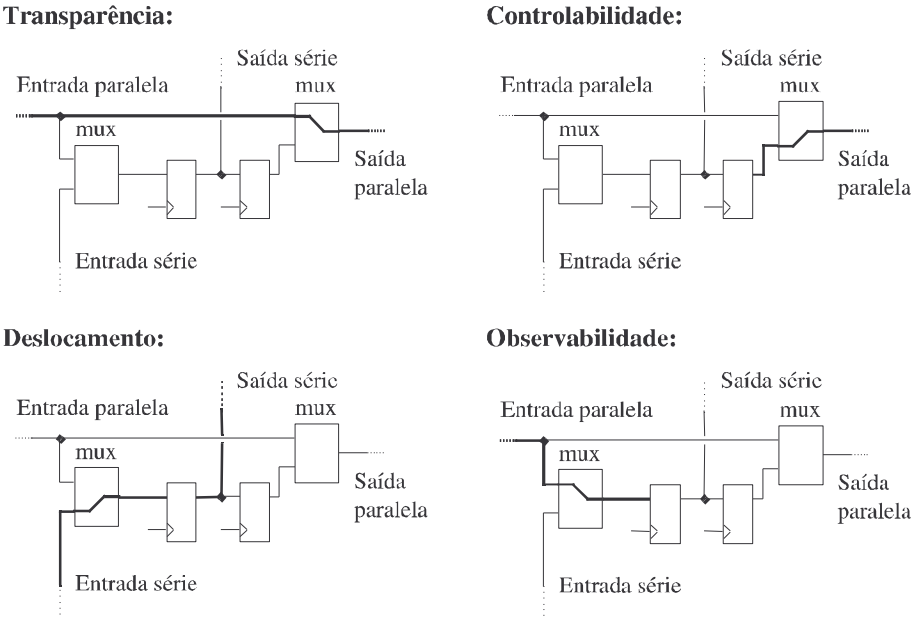


Figura 3-3: Modos de funcionamento de uma célula BS com andar de retenção.

3.1.3 O registo de instrução

Este registo, representado na Figura 3-1, é um dos blocos obrigatórios, sendo constituído por dois andares. O primeiro destes andares constitui um A_D com carga paralela, por onde passam os valores que entram por TDI, sendo o outro andar constituído por um A_R que armazena o conteúdo do A_D. Quando o controlador do TAP passa pelo estado *Capture-IR*, o A_D é carregado com o vector '01' nos dois bits menos significativos (i.e. mais próximos de TDO). No estado *Shift-IR* o A_D desloca, por cada transição do relógio de teste, um bit no sentido de TDI para TDO. Quando o controlador do TAP passa pelo estado *Update-IR*, o A_R é carregado com o conteúdo do A_D.

3.1.4 O conjunto de instruções

O conjunto de instruções descritas na norma inclui três categorias:

- Instruções obrigatórias.
- Instruções opcionais.
- Instruções definidas pelo fabricante / utilizador.

Estes três tipos de instruções permitem colocar a lógica de teste em vários modos de operação, que podem, ou não, interferir com o funcionamento normal do CI. Ao invés de apresentar neste ponto cada uma das instruções, pertencentes a cada uma destas categorias, opta-se por criar uma secção em que se descrevem os modos de operação básicos / opcionais da infraestrutura BST, baseados, respectivamente, nas instruções obrigatórias e opcionais que são definidas na norma. As instruções definidas pelo fabricante permitem modos de operação mais complexos, especialmente dedicados a uma dada tarefa de teste ou verificação. Apresentam-se ainda na secção seguinte algumas instruções opcionais, que se encaixam na terceira e última categoria mencionada, existentes em componentes comerciais de catálogo.

3.2 Modos de operação básicos / opcionais do 1149.1

Esta secção divide-se em duas partes principais. A primeira parte apresenta os modos de operação baseados nas instruções obrigatórias e opcionais definidas na norma IEEE 1149.1. A segunda parte apresenta alguns CI comerciais de catálogo (*Components Off-The-Shelf*, COTS) com uma infraestrutura BST, que para além de suportarem alguns dos modos de operação opcionais apresentados na parte anterior, suportam ainda outros modos para apoio à realização de tarefas adicionais de teste e depuração. Esta funcionalidade acrescida da infraestrutura BST, corresponde à definição e implementação de instruções públicas adicionais, permitida pela norma [IEEE93, pag. 7-2[26]].

3.2.1 Instruções obrigatórias

A norma IEEE 1149.1 define três instruções obrigatórias (*EXTEST*, *SAMPLE / PRELOAD*, e *BYPASS*), que devem ser suportadas por todos os CI que possuam uma infraestrutura de teste compatível com esta especificação. Descreve-se em seguida cada uma destas instruções.

---

[26] *Permission 7.2.1.(g) – "A design may offer public instructions in addition to those defined in this standard to allow the device purchaser access to design-specific features."*

A instrução *EXTEST* coloca o registo BS entre os pinos TDI - TDO do CI. Sempre que esta instrução estiver activa, o estado de todos os pinos de saída é definido pelos valores presentes no A_R do registo BS, mudando apenas na transição descendente de TCK, com o controlador do TAP no estado *Update-DR*. Os valores capturados nas células do registo BS, na transição ascendente de TCK, durante o estado *Capture-DR*, não dependem da operação da lógica funcional do CI. Opcionalmente, as células BS associadas aos pinos de entrada podem controlar o valor aplicado às entradas da lógica funcional do CI, seja pela aplicação de um valor constante, ou através de um valor previamente deslocado para o registo BS. Neste último caso, compete ao fabricante garantir que os valores potencialmente geradores de conflito, ao nível da lógica funcional, não provocam danos no CI, quando a instrução *EXTEST* se encontra activa. A norma define que o código desta instrução corresponde a carregar todas as células do registo de instrução com o valor lógico '0'.

A instrução *EXTEST* permite o teste do circuito externo ao encapsulamento do CI – tipicamente as interligações existentes na CCI. As células BS associadas aos pinos de saída são utilizadas para aplicar os estímulos, enquanto as células BS associadas aos pinos de entrada capturam as respectivas respostas. Os valores presentes nos pinos de entrada são inicialmente capturados no A_D, e posteriormente deslocados, com o controlador do TAP no estado *Shift-DR*, para posterior observação, ou comparação com valores esperados.

A instrução *SAMPLE / PRELOAD* coloca o registo BS entre os pinos TDI - TDO do CI. Sempre que esta instrução estiver activa, a operação da lógica funcional não é afectada pela operação da lógica de teste. O estado de todos os sinais que fluem através dos pinos (de entrada ou saída) é capturado no registo BS à transição ascendente de TCK, com o controlador do TAP em *Capture-DR*. Os A_R das células BS carregam os valores presentes nos respectivos A_D, à transição descendente de TCK, com o controlador do TAP em *Update-DR*. Esta instrução permite efectuar dois tipos de acção: a) *SAMPLE* efectua uma amostragem dos valores que fluem dos pinos para a lógica funcional do CI, e vice-versa, sem interferir na operação normal do componente; b) *PRELOAD* permite carregar um dado vector inicial no A_R do registo BS, antes da selecção de outro tipo de operação da lógica de teste, por exemplo antes de se carregar a instrução *EXTEST*.

A instrução *BYPASS* coloca o registo *bypass,* de comprimento igual a um bit, entre os pinos TDI - TDO do CI, proporcionando um caminho mais curto entre estes dois pinos e reduzindo assim o comprimento total da cadeia onde se encontra inserido o CI. Sempre que esta instrução estiver activa, a operação da lógica funcional não é afectada pela operação da lógica de teste. A

norma define que o código desta instrução corresponde a carregar todas as células do registo de instrução com o valor lógico '1'.

3.2.2 Instruções opcionais definidas na norma

A norma IEEE 1149.1 define seis instruções públicas opcionais: *INTEST*, *RUNBIST*, *CLAMP*, *IDCODE*, *USERCODE* e *HIGH-Z*. Cada uma destas instruções é descrita em seguida em maior pormenor.

A instrução *INTEST* coloca o registo BS no percurso TDI - TDO do CI. Todos os pinos de saída são colocados num estado inactivo (por exemplo, em alta-impedância) *ou* passam a ter o seu estado definido pelos valores presentes no A_R do registo BS, mudando apenas na transição descendente de TCK, com o controlador do TAP em *Update-DR*. O estado de todos os sinais de entrada (excluindo em geral os de relógio) da lógica funcional é definido pelos valores presentes no A_R das células BS associadas aos respectivos pinos. O estado de todas as saídas da lógica funcional é capturado no A_D das células BS associadas aos respectivos pinos, na transição ascendente de TCK, no estado *Capture-DR*. A norma define ainda que o valor capturado pelas células BS associadas a pinos de entrada, ou a pinos bidireccionais (quando configurados como entradas), deverá ser independente da operação de circuitos externos ao CI ou das interligações existentes na CCI. Esta instrução permite o teste estático (a baixa velocidade) da lógica funcional do CI, sendo cada estímulo de entrada e a respectiva resposta deslocados através do registo BS. O CI deverá funcionar num modo passo-a-passo, de forma a que a lógica funcional avance um passo, na sua operação, sempre que se proceda a um novo deslocamento de vectores. Tipicamente, a lógica funcional recebe uma sequência de impulsos de relógio, entre a aplicação do estímulo e a captura da respectiva resposta. O tipo de células BS definidas para pinos de entrada de relógio permite obter diferentes modos de aplicação / geração deste sinal na lógica funcional do CI, enquanto a instrução *INTEST* se encontra activa. A própria norma oferece a título de exemplo alguns modos de aplicação / geração do sinal de relógio, para obter um modo de funcionamento passo-a-passo:

- Os sinais recebidos nos pinos de entrada de relógio são fornecidos directamente à lógica funcional do CI, sendo esta projectada de forma a garantir que se avança apenas um passo na sua operação normal, quando se aplica (no mínimo) um determinado número de impulsos em TCK, com o controlador do TAP no estado *Run-Test/Idle*.
- A lógica funcional é alimentada com impulsos de relógio derivados de TCK, no estado *Run-Test/Idle*. Nos restantes estados, os sinais de relógio mantêm-se estáveis.
- Um circuito interno dedicado permite que a lógica funcional avance um passo, sempre que o controlador do TAP entre no estado *Run-Test/Idle*. Por exemplo, no caso do CI corres-

ponder a um microprocessador, executa-se um ciclo de processamento completo, por exemplo através da geração de um impulso no sinal de retenção do acesso ao barramento (*hold*). Nesta situação, o relógio de entrada do CI pode encontrar-se em livre-oscilação.

- Os sinais de relógio são gerados através do deslocamento, via registo BS, dos valores apropriados, de forma idêntica aos restantes sinais de entrada. Este modo requer uma operação de deslocamento por cada estado lógico (alto e baixo) do relógio, ou seja, dois deslocamentos para o caso de um circuito síncrono com um único sinal de relógio.

A instrução *RUNBIST* coloca o registo de dados que contém o resultado das operações de auto-teste no percurso TDI - TDO do CI. Estas operações são apenas executadas quando o controlador do TAP se encontra no estado *Run-Test/Idle*, sendo a sua duração expressa em termos do número de impulsos que se devem aplicar em TCK, neste estado. Os resultados das operações de auto-teste, correspondentes aos valores deslocados, devem ser independentes do que é exterior ao CI, nomeadamente dos valores presentes nas interligações da CCI. O estado dos pinos de saída do CI é definido de uma das seguintes maneiras: a) através dos valores existentes no A_R das respectivas células BS, ou; b) a activação da instrução *RUNBIST* força um estado inactivo, ou seja, coloca os pinos de saída num estado de alta-impedância. Esta instrução permite que o utilizador efectue o auto-teste do CI, que poderá ocorrer em paralelo com outras operações de auto-teste, no caso de uma CCI com diversos CI que suportem esta capacidade, fornecendo assim uma rápida indicação do estado da carta. O fabricante pode ainda implementar outras instruções (públicas ou privadas) para aceder a funções de auto-teste de blocos internos, em simultâneo ou um de cada vez, que não sejam invocadas pela instrução *RUNBIST*. Dado que no final de um auto-teste a lógica funcional poderá estar num estado indeterminado, que poderá manter-se até que esta seja reinicializada, deverá colocar-se o CI num modo em que se possa aplicar um sinal de reinicialização.

A instrução *CLAMP* coloca o registo *bypass* no percurso TDI - TDO do CI, sendo no entanto o registo BS a definir os valores existentes nos pinos de saída. Desta forma, não é possível mudar o estado destes pinos, enquanto esta instrução se encontrar activa. Durante o teste de um determinado CI, ou grupo de CI inseridos numa CCI, pode ser necessário colocar *valores de guarda*[27] em sinais que controlem a operação da lógica de circuitos não directamente envolvidos nesse teste, por exemplo, para garantir que não reajam aos sinais recebidos da lógica sob teste. Este tipo de situação ocorre com frequência nomeadamente quando se comuta de um

[27] Um valor de guarda pode ser definido como um valor aplicado a um sinal de controlo (ou a um grupo de sinais deste tipo), com o objectivo de impedir que o respectivo circuito mude de estado, em função dos valores aplicados nos restantes sinais de entrada.

teste baseado no acesso por matriz-de-agulhas, para um teste baseado na utilização da infraestrutura BST. Apesar de se poder utilizar a instrução *EXTEST* para este fim, o comprimento total do vector a deslocar para o interior da cadeia, teria que levar em conta a inclusão dos registos BS dos CI envolvidos na aplicação de valores de guarda. A utilização da instrução *CLAMP* permite deste modo aumentar a velocidade de aplicação do teste. Dado que após a utilização desta instrução a lógica funcional poderá estar num estado indeterminado, que poderá manter-se até que esta seja reinicializada, deverá colocar-se o CI num modo em que se possa aplicar um sinal de reinicialização. Deve-se ainda garantir que não são provocados danos no circuito interno em função dos valores aplicados nas entradas (normais ou de relógio) do CI, enquanto esta instrução está activa. Esta garantia pode ser obtida através da colocação da lógica funcional do CI num estado de paragem ou no próprio estado de reinicialização, quando se activa esta instrução.

A instrução *IDCODE* coloca o registo de identificação do CI no percurso TDI - TDO, permitindo a leitura de um código que identifica: a) o fabricante; b) o tipo de componente; c) a versão do componente. Este código é carregado no referido registo, à transição ascendente de TCK, no estado *Capture-DR*, quando esta instrução se encontra activa. Quando um CI possui um registo de identificação, e após a inicialização da lógica de teste, seja pela passagem ao estado *Test-Logic-Reset,* seja por acção da entrada /TRST, deverá ser automaticamente carregado o código desta instrução no A_R do registo de instrução. Este requisito permite verificar quais os CI instalados numa CCI, quando estes não são conhecidos. Esta instrução tem um carácter não intrusivo, dado que a operação da lógica funcional não é afectada pela operação da lógica de teste.

No caso de o CI possuir simultaneamente um registo de identificação e ser do tipo programável, a norma especifica que o utilizador pode definir um código de identificação da versão programada, que será carregado no referido registo, à transição ascendente de TCK, no estado *Capture-DR*, quando a instrução *USERCODE* se encontrar activa. Esta instrução coloca assim o registo de identificação do CI no seu percurso interno TDI - TDO, sendo apenas necessária para CI programáveis, cuja programação ou configuração não possa ser determinada através de outro tipo de utilização da lógica de teste. Esta instrução tem um carácter não intrusivo.

A instrução *HIGH-Z* coloca o registo *bypass* no percurso TDI - TDO do CI, colocando todos os seus pinos de saída num estado inactivo (por exemplo, em alta-impedância). Em CCI onde coexistem CI com e sem BST, continua a ser necessária em alguns casos a utilização de equipamentos de teste baseados no acesso físico por matriz-de-agulhas. De modo a facilitar a

execução do teste por acesso físico, sem que daí resultem danos para os CI que normalmente controlam a aplicação dos estímulos de teste, torna-se vantajoso colocar os seus pinos de saída num estado inactivo, por exemplo através da instrução *HIGH-Z*. Dado que no final da utilização desta instrução a lógica funcional poderá estar num estado indeterminado, deverá prever-se a aplicação de um sinal de reinicialização. Deve ainda garantir-se que enquanto esta instrução está activa, não são provocados danos no circuito interno em função dos valores aplicados nas entradas do CI.

3.2.3 Outras instruções opcionais

Para além das instruções opcionais definidas na norma, é possível existirem outras instruções destinadas à realização de tarefas específicas, como por exemplo aceder a cadeias de varrimento internas via TAP, colocar os pinos de saída do CI num modo de geração de estímulos de teste pseudo-aleatórios, ou ainda construir uma assinatura com os valores capturados nos pinos de entrada.

Os componentes da família SCOPE™ (da Texas Instruments, TI) [SCO94], possuem diversas instruções públicas opcionais, que lhes permitem realizar operações adicionais orientadas ao teste ou à depuração. Esta família pode ser dividida em dois grandes grupos: um primeiro grupo que inclui componentes especificamente vocacionados para controlar ou apoiar tarefas de teste ou depuração (entre os quais se inclui o exemplo que será apresentado já a seguir), e um segundo grupo cuja função lógica corresponde à de diversos componentes da família TTL.

Os componentes da família SCAN™ (da National Semiconductors) [SCA94], apesar de suportarem as instruções opcionais *HIGH-Z*, *CLAMP* e *IDCODE* (definidas na norma), não suportam outras instruções públicas opcionais. Note-se no entanto que a família SCAN™ possui CI orientados a operações de teste, como o SCANPSC100F (controlador de teste) e o SCANPSC110F (porto de endereçamento hierárquico para configurações do tipo *multidrop*).

Digital Bus Monitor

O componente SN74ACT8994, ou *Digital Bus Monitor* (DBM), da família SCOPE™, foi inicialmente apresentado em [Whe91], tendo sido desenvolvido com base em dois objectivos principais: ser compatível com a norma IEEE 1149.1 e permitir a monitorização / armazenamento (ou compressão) dos valores presentes num barramento digital, com uma largura máxima de dezasseis linhas, para suporte a operações de depuração ou teste funcional. O DBM é ligado em paralelo com o barramento que se pretende monitorar, sendo os valores capturados nas suas entradas de dados posteriormente armazenados numa memória interna com 1024 palavras de dezasseis bits. O utilizador pode gerar uma assinatura paralela dos valores capturados

nas entradas, ou em alternativa dos valores armazenados na memória interna, com base na utilização de um Registo de Deslocamento com Realimentação Linear (*Linear-Feedback Shift Register*, LFSR), que efectua a compressão dos dados. O utilizador pode também especificar máscaras para impedir a contribuição de uma ou mais entradas de dados para o processo de formação da assinatura, e ainda especificar qual a malha de realimentação para o LFSR. Todas as instruções enviadas para o DBM são recebidas através do TAP, podendo este operar em dois modos, conforme esteja ou não em sincronismo com o sistema a que se encontra ligado. Neste último caso, as operações efectuadas pelo DBM são independentes das condições do sistema, como sejam por exemplo a geração da assinatura do conteúdo da sua memória interna, ou o teste estrutural das ligações externas. No primeiro caso o DBM pode ser configurado para efectuar operações de monitorização, que são iniciadas com base em condições do sistema sob depuração, sincronizadas através de uma combinação lógica de uma, duas, ou três entradas de relógio externas (CLK1, CLK2 e CLK3). Estas operações podem ser efectuadas com base num de oito protocolos, cuja descrição sumária se apresenta na Tabela 3-1.

Tabela 3-1: Protocolos do DBM.

Protocolo 1	**Protocolo 2**	**Protocolo 3**	**Protocolo 4**
Durante *m* vezes faça: Após *n* vezes *condição* armazene valores na memória[28].	Durante *m* vezes faça: Após *n* vezes *condição* armazene valores enquanto *condição* for verdadeira.	Durante *m* vezes faça: Após *n1* vezes *condição início* armazene valores até *n2* vezes *condição fim.*	Durante *m* vezes faça: Após *n1* vezes *condição início* armazene valores até *n2* vezes *condição fim*, mais *n3* impulsos de relógio.
Protocolo 5	**Protocolo 6**	**Protocolo 7**	**Protocolo 8**
Durante *m* vezes faça: Após *n1* vezes *condição* armazene valores durante *n2* impulsos de relógio.	Durante *m* vezes faça: Após *n1* vezes *condição,* espere *n2* impulsos de relógio, armazenando depois valores durante *n3* impulsos de relógio.	Após *n1* vezes *condição* armazene valores durante *n2* impulsos de relógio. Durante *m* vezes faça: Espere *n3* impulsos de relógio, armazenando depois valores durante *n4* impulsos de relógio.	Após *n1* vezes *condição.* Durante *m* vezes faça: Espere *n2* impulsos de relógio, armazenando depois valores durante *n3* impulsos de relógio.

A arquitectura geral do DBM pode ser representada através do diagrama de blocos ilustrado na Figura 3-4. O bloco de memória interna pode ser acedido através de um registo, denominado RAMR (RAM *Register*), que é colocado no percurso TDI - TDO, quando se activa uma de duas instruções opcionais. O bloco de qualificação de eventos (*Event Qualification Module*, EQM) contém dois registos de dados de teste, denominados EQR1 e EQR2, que armazenam

[28] Os valores *n* e *m* são especificados pelo utilizador. A *condição* refere-se a um qualquer vector com dezasseis bits (com a possibilidade de se mascarar alguns desses bits) presente nas entradas de dados do DBM.

informação relativa à configuração do DBM, o vector da condição esperada, e a respectiva máscara de comparação, para implementar as operações especificadas pelo protocolo seleccionado. No interior deste bloco encontram-se igualmente as MEF que implementam os oito protocolos anteriormente referidos, e que operam em sincronismo com o sinal gerado pelo bloco denominado Interface Programável de Relógio (*Programmable Clock Interface*, PCI). Este bloco permite implementar uma de 32 combinações lógicas entre os sinais dos pinos de entrada CLK1, CLK2, CLK3 e TCK. A combinação é seleccionada através do vector deslocado para o registo de controlo (*Control Register*, CTLR) do DBM. O bloco de geração de assinatura é na realidade um registo, denominado TCR (*Test Cell Register*), acessível através de TDI - TDO. Durante a geração da assinatura, este registo é configurado como um LFSR. O registo de instrução, o CTLR, o EQR1, o EQR2 e o RAMR são descritos em seguida em maior pormenor.

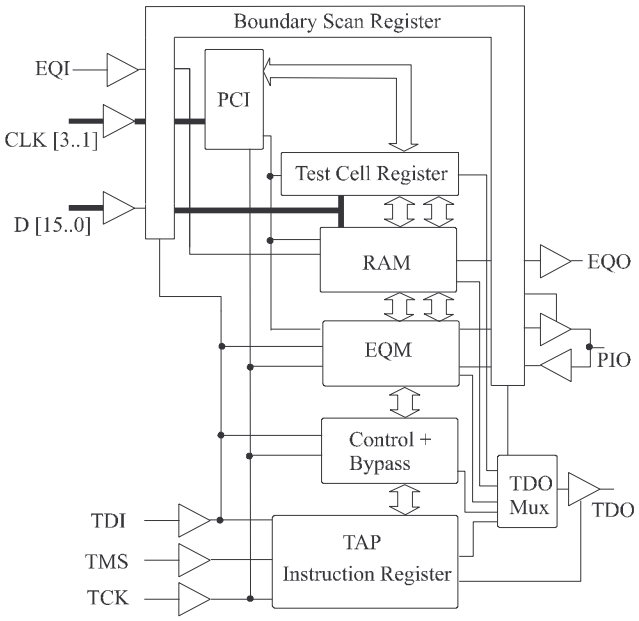


Figura 3-4: Diagrama de blocos da arquitectura interna do SN74ACT8994 (DBM).

Registo de instrução - oito bits - para além de conter a instrução que controla todas as operações do DBM, captura na passagem por *Capture-IR* uma palavra com quatro bits, que indicam a ocorrência de condições internas. O bit de estado denominado IERRR é activado sempre que é carregado um código de instrução que não exiba paridade par. O bit de estado denominado OVF é activado quando o apontador da memória interna do DBM é incrementado, passando do seu valor máximo para o valor zero. Os bits de estado denominados RUN e EOT indicam que um protocolo se encontra em execução, ou que terminou, respectivamente.

CTLR – 45 bits - gera vários sinais de controlo de recursos internos, contém a máscara e a definição da malha de realimentação do registo de geração de assinatura, selecciona qual a operação a efectuar pelo DBM e configura o sinal de saída do PCI.

EQR1 – 32 bits - configura o DBM na situação em que este executa operações de monitorização. O vector deslocado para este registo permite seleccionar um dos oito protocolos, o sinal de indicação de evento para as MEF (que implementam os protocolos) e o sinal que alimenta o pino de saída de indicação de evento (*Event Qualification Output*, EQO). Este registo contém ainda o contador que controla o número de vezes que um protocolo é executado (referido como *m* na Tabela 3-1).

EQR2 - utilizado para carregar o contador de eventos, o vector esperado (ou condição) e a máscara de comparação. Na realidade, existe um banco com dezasseis (2^4) registos deste tipo. Dependendo da instrução actual, pode ter um comprimento de 48 bits, ou de 56 bits, onde os oito bits a mais correspondem a dois grupos de quatro bits que identificam, respectivamente, qual o contador que irá ser carregado (com o número que na Tabela 3-1 é referido como *n*, *n1*, *n2*, *n3*, ou *n4*) e qual o registo que irá conter o vector esperado e a máscara de comparação. O comprimento de 48 bits é seleccionado por instruções que permitem ler directamente todos os dezasseis registos, da seguinte forma: sempre que são deslocados 48 bits para o exterior, é apontado o registo seguinte, pelo que são necessários 48 * 16 impulsos de TCK para ler todo o banco de registos.

RAMR – permite aceder à memória interna de duas maneiras, seleccionadas através de duas instruções opcionais: posição a posição ou como um todo. Na primeira o registo tem um comprimento de 26 bits, sendo dez dos bits para indicar qual a posição actual e os restantes dezasseis para armazenar o conteúdo da posição anterior. Na segunda o registo tem um comprimento de dezasseis bits, que são carregados com o conteúdo da posição de memória, apontada por um contador, que é incrementado sempre que são deslocados dezasseis bits para o exterior.

Outros componentes da família SCOPE™

Para além do DBM, os CI da família SCOPE™ dividem-se em dois grandes grupos, como foi já anteriormente referido:

- Componentes de uso genérico: *buffers* (inversores e não-inversores), *transceivers* (registados e não-registados), registos multi-bit do tipo D, etc.
- Componentes dedicados ao teste: *Test Bus Controllers* (TBC, ou SN74ACT8990), *Scan-Path Linkers* (SN74ACT8997), *Scan-Path Selectors* (SN74ACT8999) e *Addressable Scan Ports* (ASP, ou SN74ABT8996).

Os componentes do segundo grupo possuem funções especialmente orientadas para o teste estrutural de CCI. O TBC permite efectuar o interface entre um microprocessador e o TAP da cadeia BST da CCI, por forma a executar o seu teste estrutural, aplicando vectores e recolhendo as respostas para posterior análise. O ASP permite estabelecer um interface entre a cadeia BST de uma CCI e um barramento de teste ao nível hierárquico superior (caracterizado pela ligação em barramento comum de várias CCI), baseado igualmente nas cinco linhas que formam o TAP. Em relação aos componentes de uso genérico verifica-se que a maioria suporta as instruções opcionais *HIGH-Z*, *CLAMP* e *IDCODE*, definidas na norma, e ainda um conjunto de outras instruções opcionais disponibilizadas pelo fabricante, que permitem efectuar diversas funções de teste e depuração. Estas instruções incluem:

- O acesso a um registo que permite controlar o modo de funcionamento do registo BS do CI[29]. Esta instrução, denominada *Boundary-control-register scan*, coloca este registo no percurso TDI - TDO do CI. O valor presente no registo não é alterado pela passagem do controlador do TAP pelo estado *Capture-DR*. Antes de se iniciar uma operação de teste activada pela instrução *Boundary run test* (a descrever em seguida), deve-se especificar qual a operação a efectuar, através do deslocamento da combinação correcta para o registo de controlo. As cinco operações de teste descodificadas pelo registo são:
  - Inverter Saídas / Amostrar Entradas (*Toggle Outputs* / *Sample Inputs*, TOPSIP) - os valores existentes nos pinos de entrada do CI são capturados no A_D das respectivas células BS, à transição ascendente de TCK, e os valores existentes nas células BS associadas aos pinos de saída do CI, são invertidos na transição ascendente de TCK, e aplicados a esses pinos à transição descendente de TCK, através do A_R.
  - Geração de Estímulos Pseuso-Aleatórios (*Pseudo-Random Pattern Generation*, PRPG) As células BS são configuradas de forma a produzirem, nas suas saídas paralelas, uma sequência de estímulos pseudo-aletórios. Isto significa que tanto os pinos de saída como as entradas da lógica funcional recebem estes estímulos. Antes de activar a operação de teste, o utilizador deverá deslocar para o interior do registo BS uma semente inicial (um vector 'tudo-0' não permite gerar qualquer tipo de sequência).
  - Análise de Assinatura Paralela (*Parallel-Signature Analysis*, PSA). Os valores capturados no A_D são comprimidos numa assinatura paralela, com um número de bits idêntico ao comprimento do registo BS, sendo os valores existente no A_R aplicados às saídas paralelas das respectivas células. Antes de activar a operação de teste, o utilizador deverá deslocar para o interior do registo BS uma semente, de forma a definir um ponto de partida conhecido.
  - Execução simultânea das duas operações anteriores (PRPG / PSA).

[29] Este registo é designado por *Boundary Control Register* (BCR).

- Análise de Assinatura Paralela e Contagem binária (PSA / COUNT), em simultâneo. Esta operação difere da anterior pelo facto de, ao invés de se gerar uma sequência pseuso-aletória, se proceder agora ao incremento do valor existente no A_R do registo BS, em cada transição descendente de TCK. O utilizador deve deslocar para o interior do registo BS a semente inicial que condiciona a formação da assinatura paralela.

- A activação da operação de teste seleccionada através do registo de controlo (anteriormente mencionado). Esta instrução, denominada *Boundary run test*, coloca o registo *bypass* no percurso TDI – TDO do CI. A operação de teste seleccionada é executada enquanto o controlador do TAP se mantiver no estado *Run-Test/Idle*.
- A leitura do conteúdo do registo BS. Esta instrução, denominada *Boundary read*, permite deslocar para o exterior o conteúdo do registo BS do CI, com a particularidade de não serem modificados os valores nele existentes, à passagem pelo estado *Capture-DR*. Esta instrução é útil quando se pretende verificar a assinatura formada no interior do registo, após a realização de uma operação do tipo PSA.
- A realização de um auto-teste do registo BS. Esta instrução, denominada *Boundary Self Test*, coloca o registo BS no percurso TDI – TDO do CI. Todas as células BS capturam, no estado *Capture-DR*, o valor oposto ao existente no A_R. Desta forma, pode-se ler o conteúdo do A_R, verificando-se assim a integridade de ambos os andares do registo BS.

3.3 Depuração através dos modos de operação básicos do 1149.1

Esta secção descreve as formas de implementação das operações de depuração que constam do modelo ilustrado na Figura 2.14, utilizando os modos de operação básicos da infraestrutura BST. A distinção entre os três tipos principais de sistemas (lógica dedicada, microprocessadores ou híbridos) é referida implicitamente ao longo da descrição efectuada.

3.3.1 Operações de COV

A implementação das operações de COV serão descritas pela seguinte ordem:

- COV de pinos directamente acessíveis.
- COV de pinos acessíveis por varrimento periférico.
- COV de pinos acessíveis por propagação.
- COV de registos acessíveis por varrimento periférico.
- COV de registos acessíveis por propagação.
- COV de portas lógicas acessíveis por propagação.

COV de pinos directamente acessíveis

Pinos directamente acessíveis de CCI (e.g. os pinos de entrada / saída da CCI) podem ser controlados, observados e verificados, utilizando a instrução *EXTEST*, na condição de se encontrarem ligados a pinos de componentes com uma infraestrutura BST (no caso das entradas / saídas primárias da CCI, requer-se um colar exterior de células BS a elas ligadas). A Figura 3-5 ilustra esta solução, que apesar de simples e directa, é desvantajosa em termos de velocidade, em face das operações de deslocamento necessárias para colocar / ler os valores nos registos BS dos componentes externos. As operações de COV de pinos directamente acessíveis de uma CCI, transformam-se assim em operações de COV de pinos de componentes externos, acessíveis por varrimento (descritas a seguir).

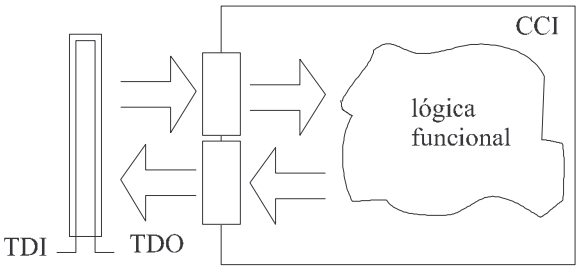


Figura 3-5: COV do estado de pinos directamente acessíveis, através da sua ligação a pinos acessíveis por varrimento periférico.

COV de pinos acessíveis por varrimento periférico

Pinos de componentes com uma infraestrutura BST podem ser controlados, observados e verificados (a comparação deverá ser efectuada através de uma ferramenta externa) utilizando as instruções *EXTEST* e *SAMPLE / PRELOAD*, de acordo com as possibilidades expressas na Figura 3-6. A principal diferença entre estas duas instruções reside na sua intrusão, ou seja, enquanto a instrução *EXTEST* coloca o componente num modo de teste, onde os valores presentes nos pinos de saída são determinados pelo registo BS, no caso da instrução *SAMPLE / PRELOAD* é a lógica funcional que determina os valores neles presentes. Os valores presentes nos pinos de entrada do componente são observáveis através de ambas instruções, com a diferença de a instrução *EXTEST* colocar o componente num modo de teste, que impede a operação normal da lógica funcional. Em termos de verificação, assume-se que a observação dos valores é efectuada através da infraestrutura BST, sendo os valores capturados posteriormente comparados com os valores esperados (por uma ferramenta externa ou por um controlador dedicado), após o seu deslocamento para o exterior do componente.

COV de pinos acessíveis por propagação

A propagação do valor pretendido, através do cone de influência do pino que se pretende controlar, pode ser efectuada utilizando as duas operações anteriores de controlo, na condição de os elementos que formam o cone de influência corresponderem a pinos acessíveis directamente, ou por varrimento periférico. O mesmo se aplica às operações de observação e verificação, na situação em que se pretende observar o valor presente num dado pino, através da sua propagação até um pino acessível directamente, ou por varrimento periférico. Repare-se que em ambas as situações é necessário utilizar a instrução *EXTEST* (para efeitos de controlo), pelo que a implementação deste tipo de operação impede o funcionamento normal do sistema.

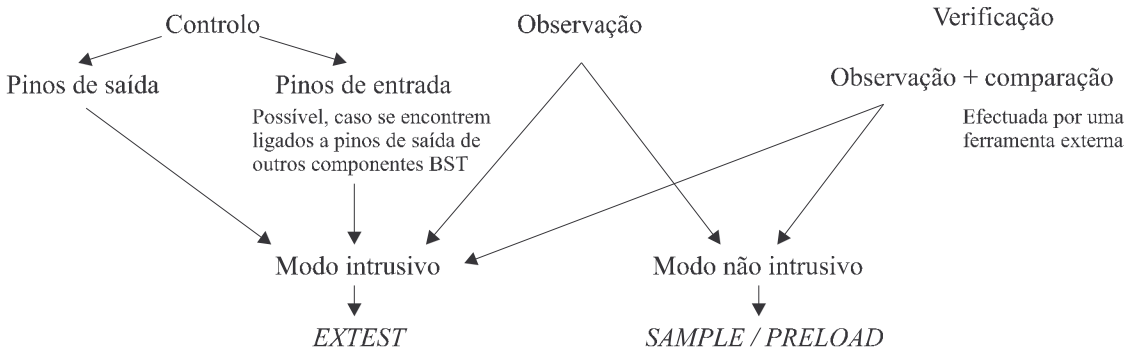


Figura 3-6: Implementação de operações de COV de pinos acessíveis por varrimento periférico, utilizando as instruções *EXTEST* e *SAMPLE / PRELOAD*.

COV de registos acessíveis por varrimento (periférico)

O acesso à cadeia de varrimento interna de um componente pode ser efectuado através de pinos dedicados de Entrada de Varrimento (*Scan Input*, SI) e Saída de Varrimento (*Scan Output*, SO), ou através dos seus pinos TDI – TDO. Esta última opção implica a existência de uma instrução opcional, situação que apenas se considera na secção seguinte (nesta secção consideram-se apenas as três instruções obrigatórias). Considerando a primeira opção, pode-se controlar, observar e verificar o conteúdo da cadeia de varrimento interna, na condição de:

- Os pinos de selecção de modo de funcionamento (normal / varrimento), de relógio de varrimento[30] e SI, corresponderem a pinos acessíveis directamente, ou se encontrarem ligados a pinos de saída de componentes BST.
- O pino SO corresponder a um pino acessível directamente, ou se encontrar ligado a um pino de entrada de um componente BST.

[30] Em determinados componentes, o relógio de varrimento corresponde ao próprio relógio que alimenta a lógica funcional.

Verificando-se esta condição, utiliza-se a instrução *EXTEST* para efectuar as operações de COV dos registos inseridos na cadeia de varrimento interna.

COV de registos / portas lógicas acessíveis por propagação

As operações de controlo são consideradas exequíveis, no caso de ser possível propagar os valores pretendidos através do cone de influência das entradas do registo, ou da porta lógica, tendo em atenção o exposto na implementação de operações de controlo de pinos acessíveis directamente, ou por varrimento periférico. As operações de observação (e verificação) são consideradas exequíveis, no caso de ser possível propagar o valor presente na saída do registo, ou da porta lógica, até um pino acessível directamente, ou por varrimento periférico.

3.3.2 Operações passo-a-passo

Esta operação de depuração pode ser implementada utilizando a instrução *EXTEST*. Se não existir um limite inferior para a frequência de relógio do sistema, pode-se avançar um passo através do fornecimento do número de impulsos de relógio necessários, utilizando uma célula BS. Esta célula deverá estar associada ao pino que corresponde à saída do circuito de geração de relógio do sistema, ou alternativamente corresponder à célula associada ao pino de saída de um componente BST, inserido no final do circuito anterior, como se ilustra na Figura 3-7, para efeitos de implementação desta operação via infraestrutura BST. A aplicação de um impulso de relógio corresponde à execução das seguintes acções (considerando que a instrução *EXTEST* já se encontra activa, que o A_R da célula BS associado ao pino de relógio foi previamente carregada com o valor lógico '0' e que o controlador do TAP se encontra no estado *Shift-DR*):

- Deslocar o vector que coloca o valor lógico '1' na célula BS associada ao sinal de relógio.
- Mover o controlador do TAP para o estado *Shift-DR*, através dos estados *Exit1-DR*, *Update-DR*, *Select-DR* e *Capture-DR*. Ao passar no estado *Update-DR,* o valor lógico '1' será aplicado no pino de saída de relógio.
- Deslocar o vector que coloca o valor lógico '0' na célula BS.
- Mover novamente o controlador do TAP para o estado *Shift-DR*, através dos estados *Exit1-DR*, *Update-DR*, *Select-DR* e *Capture-DR*. Ao passar no estado *Update-DR,* o valor lógico '0' será aplicado no pino de saída de relógio.
- Repetir as quatro acções anteriores o número de vezes necessário.

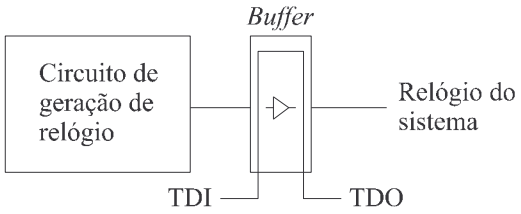


Figura 3-7: Controlo do relógio do sistema através da inclusão de um componente BST à saída do seu circuito de geração, para implementar operações passo-a-passo.

3.3.3 Operações de pontos de paragem por condição

Esta operação de depuração pode ser implementada de duas formas distintas, utilizando as instruções *EXTEST* e / ou *SAMPLE / PRELOAD*. Se não existir um limite inferior para a frequência do relógio do sistema, pode utilizar-se a instrução *SAMPLE / PRELOAD* para executar sucessivos ciclos de captura e deslocamento (entre a aplicação de dois impulsos contíguos do relógio do sistema), de maneira a que a avaliação de uma dada condição de paragem seja efectuada em cada ciclo de relógio. No caso da condição ser falsa, aplica-se um novo impulso no relógio de sistema, utilizando o procedimento descrito na subsecção anterior (3.3.2).

A segunda forma aplica-se a sistemas baseados em microprocessadores. A avaliação da condição de paragem é efectuada através da simulação da execução do programa do microprocessador, identificando-se todas as posições de memória que tornam verdadeira a condição especificada. As posições imediatamente seguintes às identificadas deverão ser preenchidas com uma instrução que efectue um salto para uma rotina especial de controlo, ou que em alternativa suspenda a actividade do microprocessador (por exemplo, efectuando um salto para a mesma posição, de maneira a criar um ciclo infinito). Saliente-se porém que o número de condições possíveis de detectar é mais reduzido, quando comparado com a forma anterior, uma vez que condições que dependam de determinados eventos externos não ocorrem normalmente durante a simulação do programa. As posições da memória de programa (que deverá corresponder a uma memória do tipo volátil) podem ser alteradas utilizando o procedimento baseado no seguinte conjunto de acções[31]:

- Colocar o controlador do TAP no estado *Shift-IR*.
- Deslocar o código da instrução *SAMPLE / PRELOAD* para os componentes BST em questão e o código da instrução *BYPASS* para os restantes.

[31] Considerando que o microprocessador dispõe de uma infraestrutura BST, ou então que todas as linhas dos barramentos de endereços, dados e controlo da memória de programa, provêm de componentes BST (por exemplo do tipo SN74BCT8244A ou SN74BCT8245A).

- Mover o controlador do TAP para *Shift-DR*, através de *Exit1-IR*, *Update-IR*, *Select-DR* e *Capture-DR*. Ao passar em *Update-IR,* as instruções carregadas ficarão activas.
- Deslocar o vector que coloca o endereço e os dados pretendidos (linha de selecção de memória activa e linha de escrita inactiva) nas células BS associadas aos vários barramentos (endereços, dados e controlo).
- Mover o controlador do TAP para o estado *Shift-IR*, através dos estados *Exit1-DR*, *Update-DR*, *Select-DR*, *Select-IR* e *Capture-IR*.
- Deslocar o código da instrução *EXTEST* para os componentes BST em questão e o código da instrução *BYPASS* para os restantes.
- Mover o controlador do TAP para *Shift-DR*, através de *Exit1-IR*, *Update-IR*, *Select-DR* e *Capture-DR*. Ao passar em *Update-IR*, as instruções carregadas ficarão activas e os valores pretendidos serão aplicados nas entradas da memória de programa.
- Deslocar o vector que activa a linha de escrita (mantendo os valores anteriores para as linhas de endereços, dados e selecção de memória).
- Mover novamente o controlador do TAP para o estado *Shift-DR*, através dos estados *Exit1-DR*, *Update-DR*, *Select-DR*, e *Capture-DR*.
- Deslocar o vector que desactiva a linha de escrita (mantendo os valores anteriores para as linhas de endereços, dados e selecção de memória).
- Repetir as acções anteriores, no caso de ser necessário modificar o conteúdo de mais posições de memória (considerando no entanto que a instrução *EXTEST* já se encontra activa).

Ambas as formas anteriores requerem a utilização de uma ferramenta computacional. Na primeira, a ferramenta é responsável pela comparação, utilizando possivelmente máscaras, entre os valores deslocados do interior da cadeia de varrimento periférica (do CI ou da CCI) e os valores correspondentes à condição de paragem. Esta última poderá abranger valores pertencentes a qualquer pino ou conjunto de pinos, onde exista pelo menos uma célula BS associada, uma vez que a instrução *SAMPLE / PRELOAD* garante a observabilidade em modo não intrusivo. A principal limitação desta forma de implementação reside nos longos períodos de espera associados às operações de deslocamento, supondo que a avaliação da condição de paragem se efectua numa fracção deste período. Casos reais, em que uma condição de paragem ocorre após a aplicação de milhões de impulsos de relógio ao sistema, resultam num processo extremamente moroso, que desaconselha esta forma de implementação. As principais desvantagens associadas à segunda forma de implementação residem na: a) necessidade de utilizar uma ferramenta de simulação; b) necessidade de armazenar o programa numa memória de programa do tipo volátil, acessível a partir da infraestrutura BST; c) diminuição do número de condições possíveis de detectar. A principal vantagem reside no facto de o programa ser executado à velocidade normal de funcionamento do microprocessador, embora seja necessário considerar o

tempo que demora a fase inicial de simulação, identificação e substituição do conteúdo das posições de memória imediatamente contíguas àquelas que tornam verdadeira a condição de paragem especificada.

3.3.4 Operações de análise em tempo real

A implementação das operações de análise em tempo real serão descritas pela seguinte ordem:

- Aquisição de valores em tempo real.
- Implementação do algoritmo de alargamento de um impulso de relógio.
- Detecção de tempos de atraso.

Aquisição de valores em tempo real

A natureza série da cadeia de varrimento periférica impede a aquisição em tempo real dos valores existentes nas células BS, em virtude do tempo necessário para deslocar o conteúdo de toda a cadeia e mover o controlador do TAP entre os estados *Shift-DR* e *Capture-DR* (que idealmente deveria ter lugar entre dois ciclos consecutivos do relógio do sistema). No entanto, é possível aplicar uma técnica de amostragem baseada na execução intercalada de sucessivos ciclos de captura e deslocamento, de forma a criar um conjunto de janelas de amostragem do funcionamento de um sistema em tempo real [Lef90, Wag91]. A Figura 3-8 ilustra uma forma de implementação desta técnica, originalmente descrita em [Lef90], que utiliza a instrução *SAMPLE / PRELOAD*, e onde o relógio do sistema é alimentado a partir do relógio de teste, de forma a garantir o sincronismo entre ambos. A implementação desta técnica pode ser descrita da seguinte forma:

- Após um atraso inicial (correspondente a um determinado número de impulsos de TCK) efectua-se a primeira amostragem dos valores presentes no sistema, à transição ascendente de TCK, com o controlador do TAP no estado *Capture-DR*.
- A próxima amostragem é efectuada após o deslocamento para o exterior de todo o conteúdo da cadeia de varrimento periférica. A distância entre estas duas amostragens corresponde a um número fixo de impulsos de TCK, ou seja, à soma do número de impulsos de TCK necessários para: a) colocar o controlador do TAP no estado *Shift-DR*; b) deslocar o conteúdo de toda a cadeia e; c) mover o controlador do TAP para o estado *Capture-DR*.
- A acção anterior é repetida até se obter uma sequência de amostragem. Esta corresponde a (considerando um atraso inicial de p ciclos de TCK e uma distância fixa entre amostragens consecutivas de q ciclos de TCK) amostrar os valores presentes nos seguintes ciclos de funcionamento do sistema: ciclo p, ciclo $p + q$, ciclo $p + 2q$, …, ciclo $p + n*q$.

- A próxima sequência de amostragem, efectuada após se inicializar o sistema, começa com um atraso inicial de $p + 1$ ciclos de TCK, correspondendo assim a amostrar os valores presentes nos ciclos de funcionamento $p + 1$, $p + 1 + q$, …, ciclo $p + 1 + n*q$.
- Nas sequências de amostragem seguintes adiciona-se sucessivamente um ciclo de TCK ao atraso inicial, de forma a que, quando no final se juntarem todas as sequências de amostragem, se obtenha uma base de dados correspondente à aquisição em tempo real do funcionamento do sistema, tal como se ilustra no canto inferior direito da Figura 3-8.

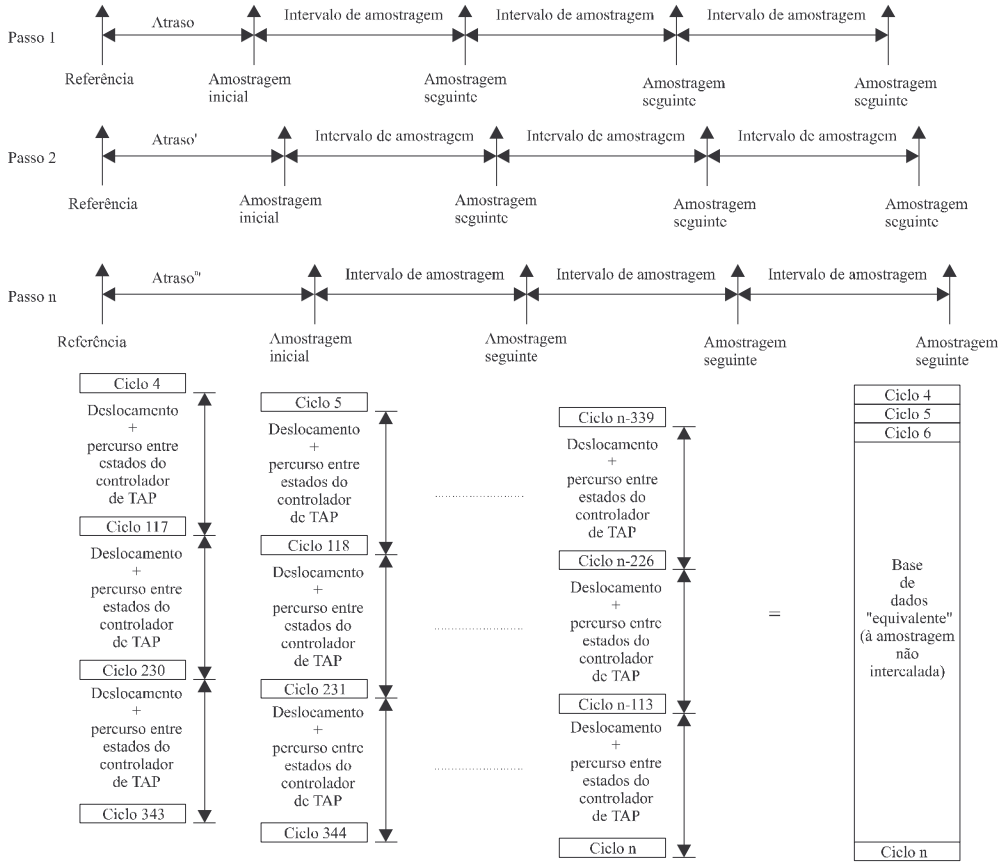


Figura 3-8: "Aquisição" em tempo real utilizando a instrução *SAMPLE / PRELOAD*.

Implementação do algoritmo de alargamento de um impulso de relógio

O algoritmo de alargamento de um impulso de relógio implica um esquema de controlo do relógio do sistema, que não é possível de implementar através das instruções *EXTEST* ou *SAMPLE / PRELOAD*. Repare-se que mesmo no caso do relógio do sistema corresponder a um pino de saída com uma célula BS associada, o tempo necessário para comutar o seu valor lógico, utilizando o procedimento baseado no deslocamento de um vector, facilmente excede o período mínimo aceitável para o funcionamento do sistema em tempo real.

Detecção de tempos de atraso

O intervalo entre o momento em que um vector é aplicado nas células BS associadas aos pinos de saída e o momento em que a resposta a esse vector é capturado nas células BS associadas aos pinos de entrada, depende do percurso efectuado pelo controlador do TAP, entre o estado *Update-DR* (transição descendente) e o estado *Capture-DR* (transição ascendente). Através do respectivo diagrama de estados, verifica-se que existem duas hipóteses: a primeira equivale a um intervalo de 2,5 ciclos de TCK - percurso *Update-DR -> Select-DR -> Capture-DR*, sendo a segunda equivalente a um intervalo de 3,5 ciclos de TCK - percurso *Update-DR -> Run-Test/Idle -> Select-DR -> Capture-DR*. No caso da frequência máxima do relógio de sistema ser 2.5 vezes inferior à frequência máxima de TCK, é possível detectar atrasos (externos) entre pinos[32], superiores ao período mínimo do relógio do sistema, utilizando para isso a instrução *EXTEST*. No entanto, dado que a frequência de TCK não ultrapassa geralmente as poucas dezenas de MHz, e dado este intervalo mínimo de 2,5 ciclos, o número de sistemas a que se pode aplicar esta solução reduz-se àqueles que trabalham a uma frequência máxima de aproximadamente 10 MHz (considerando uma frequência de 25 MHz para TCK). Atrasos internos entre pinos e atrasos entre pinos e registos, ou entre registos, são considerados não detectáveis utilizando as instruções obrigatórias da norma IEEE 1149.1.

3.4 Depuração através dos modos de operação opcionais do 1149.1

Esta secção descreve a implementação das operações de depuração que constam do modelo ilustrado na Figura 2.14, utilizando os modos de operação opcionais definidos na norma IEEE 1149.1, ou ainda os existentes nos componentes comerciais da família SCOPE™. A distinção entre os três tipos de sistemas considerados (lógica dedicada, microprocessadores ou híbridos), é referida implicitamente ao longo da descrição efectuada.

3.4.1 Operações de COV

A implementação das operações de COV de estado será descrita pela seguinte ordem:

- COV de pinos acessíveis directamente / por varrimento periférico.
- COV de pinos acessíveis por propagação.
- COV de registos acessíveis por varrimento periférico.
- COV de registos / portas lógicas acessíveis por propagação.

[32] Ou seja, entre um pino de saída e um pino de entrada de componentes BST.

COV de pinos acessíveis directamente / por varrimento periférico

Para além da utilização para este fim das instruções *EXTEST* e *SAMPLE / PRELOAD*, analisada na secção anterior, existe um conjunto de instruções opcionais, disponíveis nos componentes SCOPE^{TM}, que permitem (em modo de teste):

- A geração de sequências pseudo-aleatórias / de contagem binária nos pinos de saída.
- A comutação do valor lógico dos pinos de saída.
- A geração de uma assinatura dos valores capturados nos pinos de entrada.

Outros componentes compatíveis com a norma IEEE 1149.1 permitem a geração de sequências pseudo-aleatórias num subconjunto dos seus pinos de saída e a manutenção de valores constantes nos restantes, com a possibilidade de controlar individualmente o estado de cada um (activo ou em alta-impedância) [Alv93, Alv95, Fer93]. Estas capacidades tornam-se úteis, por exemplo, na situação em que existem grupos de componentes não BST numa CCI. Quando se aplicam sequências pseudo-aleatórias a pinos de selecção, ou do tipo habilita, a probabilidade de estes ficarem activos (através da aplicação do valor lógico '0', ou '1') é normalmente de cinquenta por cento (devido à natureza das sequências pseudo-aleatórias). Uma vez que os vectores aplicados nas restantes entradas só provocam alterações nas saídas quando estes pinos de controlo se encontram num dado estado lógico, metade dos vectores aplicados serão redundantes. A aplicação de valores constantes permite assim reduzir a redundância das sequências pseudo-aleatórias geradas.

COV de pinos acessíveis por propagação

As instruções opcionais consideradas nesta secção não acrescentam nenhuma forma expedita de implementar estas operações, em face do exposto na secção anterior.

COV de registos acessíveis por varrimento periférico

Apesar do acesso a cadeias de varrimento internas através de uma instrução opcional se ter vindo a generalizar, como se demonstra pela quantidade de CI que suportam este modo de acesso [Bru91, Hun94, Nee96, Patel93, Pyr95, Sie95, Vid95], a questão de esta instrução pertencer ao domínio público ou privado continua sem uma resposta clara. O acesso a cadeias de varrimento internas constitui um poderoso mecanismo de inspecção do interior de um CI, sendo por isso potencialmente perigoso (em termos de protecção dos pormenores de implementação) para o seu fabricante. Em contrapartida, é extremamente útil para o utilizador, uma vez

que lhe permite detectar situações de erro, pela análise dos valores capturados em registos internos. Independentemente desta questão, considera-se ao longo deste documento (excepto quando assinalado em contrário) que se uma cadeia de varrimento interna é acessível através da infraestrutura de teste, então a respectiva instrução opcional é do domínio público.

COV de registos / portas lógicas acessíveis por propagação

Em relação ao exposto na secção anterior, acrescenta-se agora a possibilidade de propagar os valores desejados a partir / até elementos sequenciais inseridos em cadeias de varrimento internas, acessíveis através da infraestrutura BST.

3.4.2 Operações passo-a-passo

A possibilidade de fornecer um certo número de impulsos, a partir da célula BS associada ao pino de relógio do sistema, utilizando a instrução obrigatória *EXTEST,* foi já explorada e analisada na secção anterior. A deslocação de dois vectores por cada impulso de relógio de sistema é morosa e pode ser evitada se o referido pino pertencer a um componente da família SCOPE^{TM}, que suporte a instrução opcional *TOGGLE*. Uma vez que esta instrução permite inverter o valor lógico presente em cada pino de saída, em cada transição descendente de TCK, é agora apenas necessário aplicar dois impulsos em TCK para se obter um impulso no relógio do sistema, o que permite atingir (para este último sinal) metade da frequência de TCK. O único senão consiste em que os valores presentes em todos os pinos de saída do componente são igualmente invertidos – competindo ao utilizador ter em atenção este facto.

Uma segunda alternativa corresponde à utilização da instrução opcional *INTEST*. Tal como foi referido na apresentação dos modos de operação opcionais definidos na norma IEEE 1149.1, esta instrução requer que a lógica funcional do CI possa funcionar em modo de passo-a-passo. Se todos os componentes de uma dada CCI suportarem esta instrução opcional, a implementação de um passo corresponde a deslocar o respectivo código de instrução para todos os componentes, e a aplicar um determinado número de impulsos em TCK (no mínimo igual ao maior número de impulsos requeridos por cada um dos componentes para avançar um passo). Deve-se no entanto ter em atenção que os estímulos, aplicados a cada um dos componentes, são fornecidos a partir do respectivo registo BS. Este facto implica a necessidade de actualizar o conteúdo de toda a cadeia BS, de forma a se avançar um passo no funcionamento da CCI.

3.4.3 Operações de pontos de paragem por condição

A implementação deste tipo de operação de depuração será analisada considerando os três aspectos referidos na sua apresentação inicial: a) especificação do tipo de condição; b) avaliação da condição e; c) paragem do funcionamento do sistema, após a detecção de uma condição verdadeira.

Em relação ao exposto na secção anterior, pode-se alargar o domínio da especificação da condição a valores pertencentes a elementos sequenciais inseridos em cadeias de varrimento internas, acessíveis através da infraestrutura BST. Refira-se porém que o processo de leitura deve ser não destrutivo, ou seja, os valores deslocados para o exterior devem ser simultaneamente deslocados para o seu interior, para que no final se mantenha o conteúdo inicial.

Em relação à avaliação da condição de paragem, devem ser destacadas as capacidades internas do DBM, que lhe permitem efectuar a avaliação em tempo real de condições de paragem relacionadas com valores presentes nos seus dezasseis pinos de entrada de dados. Por exemplo, considerando o sistema[33] ilustrado na Figura 3-9, com dois DBM a monitorar, em tempo real, os barramentos de endereços e de dados de um microcontrolador, pode-se implementar um ponto de paragem (por exemplo, no endereço de programa X5AAh - hexadecimal), através do seguinte conjunto de acções:

- Deslocar para o interior do DBM, ligado ao barramento de endereços, a instrução que selecciona o seu registo de controlo.
- Deslocar o vector que selecciona as seguintes opções no registo de controlo:
  - Operação de amostragem dos valores presentes nos pinos de entrada; comparação directa dos valores presentes nos pinos de entrada com os valores esperados; relógio que regula o momento da amostragem = pino de entrada de relógio ligado à linha do microcontrolador que habilita a leitura da memória de programa; os restantes bits do registo de controlo não influenciam o modo de operação pretendido.
- Deslocar a instrução que selecciona o EQR1.
- Deslocar o vector que selecciona as seguintes opções no EQR1:
  - Pino de saída EQO = sinal que indica o resultado da comparação entre os valores presentes nas entradas de dados e os valores esperados, armazenados no EQR2. Os restantes bits não influenciam o modo de operação pretendido.
- Deslocar a instrução que selecciona o EQR2 em modo de acesso posição a posição.

[33] Correspondente a um circuito real descrito em [Alv97a] e ilustrado na face do CD-ROM, que acompanha esta tese.

- Deslocar o vector que coloca os seguintes valores no primeiro registo (0) do EQR2:

Endereço do registo	Contador de eventos	Valores esperados	Máscara de comparação
0000 (bin)	sem significado	05AA (hex)	1111000000000000 (bin)

- Deslocar a instrução que activa o modo de teste no DBM.
- O pino EQO passa ao estado lógico '1' no momento em que o resultado da comparação assume o valor verdadeiro, dando indicação a um circuito externo para parar o fornecimento de impulsos de relógio ao sistema. No exemplo citado, existe uma linha (não representada na Figura 3-9) que liga o pino EQO ao circuito de geração de relógio do sistema, servindo assim de sinal de controlo para o fornecimento de impulsos de relógio.

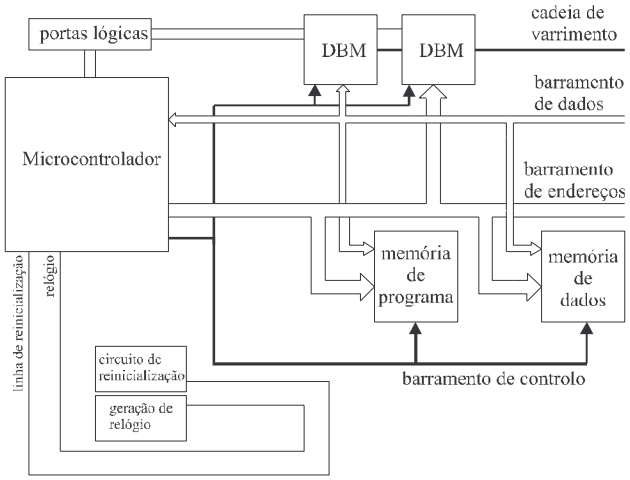


Figura 3-9: Exemplo de um sistema baseado num microcontrolador, com dois DBM ligados aos barramentos de endereços, dados e controlo [Alv97a].

A vantagem da avaliação da condição em tempo real, utilizando o DBM, é no entanto diminuída pelo facto de ser necessário efectuar uma escolha inicial, rígida, de quais as linhas do circuito que devem ser ligadas às entradas de dados deste componente.

3.4.4 Operações de análise em tempo real

A implementação das operações de análise em tempo real serão descritas pela seguinte ordem:

- Aquisição de valores em tempo real.
- Implementação do algoritmo de alargamento de um impulso de relógio.
- Detecção de tempos de atraso.

Aquisição de valores em tempo real

Nenhuma das instruções opcionais definidas na norma endereça a questão da utilização do registo BS para aquisição de valores em tempo real. Dos modos de operação opcionais apresentados na secção 2.2, apenas se pode utilizar para este fim a capacidade existente nos componentes da família SCOPE^{TM}, que permite a geração de uma assinatura dos valores capturados nos pinos de entrada, em sincronismo com TCK. Este tipo de aproximação enfrenta porém algumas limitações:

- O componente deverá estar em modo de teste, não podendo tomar parte na implementação da função do sistema.
- Representa apenas uma assinatura, não os valores individuais.
- TCK deve estar sincronizado com o relógio do sistema.
- A frequência do relógio do sistema é geralmente mais elevada do que a frequência máxima admissível para TCK.

Uma outra aproximação consiste em ligar os pinos de entrada de dados de um DBM às linhas da CCI que se pretendem amostrar em tempo real. As três entradas de relógio do DBM permitem sincronizar a aquisição com a actividade do sistema. Repare-se que é possível, através das capacidades existentes no PCI do DBM, obter um esquema de sincronização flexível, de forma a diminuir o número de amostragens efectuadas, através da selecção de uma combinação lógica entres os sinais presentes nessas três entradas. Os valores amostrados são armazenados na memória interna do DBM, de acordo com um dos oito protocolos de que dispõe, e que foram já sumariamente apresentados na Tabela 3-1. O conteúdo da memória pode ser posteriormente lido através da cadeia BS da CCI, correspondendo aos valores amostrados em tempo real. Deve-se salientar neste ponto que este tipo de solução corresponde à implementação de uma memória dedicada de amostragem no interior do sistema (referido no capítulo anterior), em que o circuito de controlo é, por sua vez, controlado através da infraestrutura BST. As principais desvantagens desta solução baseiam-se no custo dos DBM, na área de CCI adicional, e no facto de existir uma escolha inicial rígida, devido às ligações físicas que é necessário estabelecer entre os pinos de entrada do DBM e os sinais que se podem amostrar em tempo real.

Implementação do algoritmo de alargamento de um impulso de relógio

O algoritmo de alargamento de um impulso de relógio pode ser implementado através da utilização da instrução opcional, existente em diversos componentes da família SCOPE^{TM}, que permite comutar o valor lógico presente nos pinos de saída (em cada ciclo de TCK, quando o controlador do TAP se encontra em *Run-Test/Idle*). Refira-se porém que o relógio do sistema deverá provir de um pino de saída de um componente pertencente a esta família lógica. Para

alargar um impulso de relógio, move-se temporariamente o controlador do TAP do estado *Run-Test/Idle*, através de *Select-DR*, *Capture-DR*, *Exit1-DR* e *Update-DR*, para o estado *Run-Test/Idle*, novamente. A frequência máxima que se pode obter para o relógio de sistema corresponde no entanto a metade da frequência máxima de TCK, geralmente limitada a poucas dezenas de MHz.

Detecção de tempos de atraso

A detecção de tempos de atraso será analisada na seguinte ordem:

- Atraso externo entre pinos.
- Atraso interno entre pinos.
- Atraso entre um pino e um elemento sequencial, ou vice-versa (interno).
- Atraso interno entre elementos sequenciais.

As capacidades de PRPG e de PSA, existentes nos componentes da família SCOPE™, permitem detectar atrasos externos entre pinos, superiores a metade do período mínimo admissível para TCK. Esta capacidade de detecção resulta do facto de os valores aplicados nos pinos de saída serem actualizados à transição descendente TCK, e os valores existentes nos pinos de entrada serem amostrados à sua transição ascendente, do que resulta uma distância temporal de meio ciclo de relógio entre estas duas transições. Supondo que o ciclo de trabalho (*duty cycle*) de TCK é de cinquenta por cento, meio ciclo de relógio equivale a metade do período mínimo, ou metade do inverso da frequência máxima, admissível para este sinal de relógio. Este processo de detecção implica no entanto que se proceda inicialmente à detecção de faltas do tipo catastrófico – do tipo circuitos abertos e curto-circuitos [Was84] – nas linhas da CCI, para evitar a sobreposição destes dois tipos de faltas (catastróficas e do tipo atraso). Refira-se ainda que a capacidade de diagnóstico é adversamente afectada pelo facto de os valores amostrados serem comprimidos numa assinatura, obrigando a que se proceda posteriormente a uma análise computacional, de forma a identificar qual a ligação que possui um atraso superior ao normal.

A utilização da instrução opcional *INTEST* permite detectar atrasos entre pinos de um mesmo componente (atraso interno entre um pino de entrada e um pino de saída – ambos com células BS associadas), superiores a 2,5 vezes o período mínimo admissível para TCK. Este período de tempo corresponde a mover o controlador do TAP do estado *Update-DR* para o estado *Capture-DR*, via *Select-DR*. Esta capacidade de detecção reduz-se no entanto a percursos puramente combinatórios, podendo afirmar-se que um largo conjunto de faltas do tipo atraso escapa facilmente a esta janela de detecção (de 2,5 vezes o período mínimo admissível para TCK), pelo que se tornam não detectáveis através deste processo.

Faltas do tipo atraso entre um pino e um elemento sequencial, ou vice-versa (interno), não podem ser detectadas através da infraestrutura BST, uma vez que o registo BS (que contém as células que permitem controlar / observar o valor dos pinos) e o registo de varrimento interno (que contém os elementos sequenciais internos), são seleccionados através de diferentes instruções. O tempo necessário para comutar entre uma e outra instrução exclui qualquer possibilidade de detectar este tipo de atrasos internos.

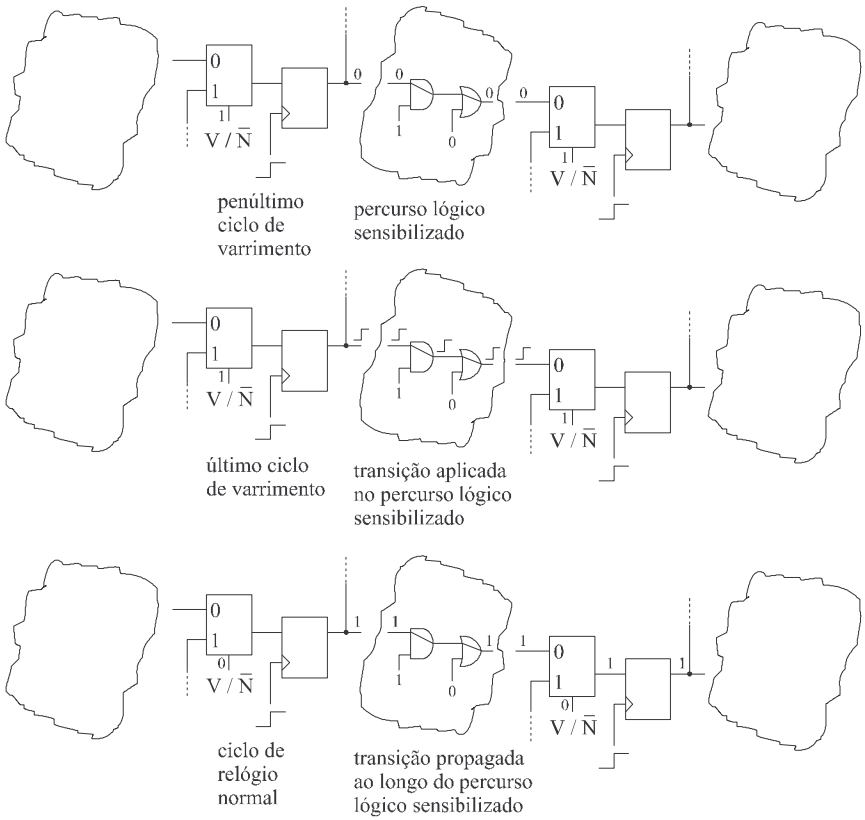


Figura 3-10: Sequência temporal da detecção de faltas do tipo atraso entre elementos sequenciais inseridos numa cadeia de varrimento interna, utilizando a técnica de justificação por varrimento.

Um pequeno conjunto de faltas do tipo atraso, entre elementos sequenciais pertencentes a uma cadeia de varrimento interna de um componente, podem ser detectadas utilizando uma técnica denominada por *justificação por varrimento* [Der91, Fuc91, Hsi77, Var94]. Esta técnica consiste em sensibilizar, no penúltimo ciclo de varrimento, o percurso lógico entre dois elementos sequenciais, e aplicar uma transição no percurso sensibilizado no último ciclo de varrimento, imediatamente antes de se comutar o sinal que passa de modo de varrimento para modo normal. Todas estas acções são executadas com o relógio de sistema a funcionar à sua

velocidade normal, de modo a que o primeiro ciclo de relógio que sucede à comutação do valor do sinal de controlo, permite capturar a resposta à transição aplicada. Para aceder ao valor capturado comuta-se novamente para modo de varrimento, de maneira a deslocar para o exterior o conteúdo do elemento sequencial em questão. A Figura 3-10 ilustra esta sequência temporal, que no entanto não pode ser implementada quando o acesso à cadeia de varrimento interna se efectua através da infraestrutura BST. A impossibilidade resulta de o tempo necessário para comutar entre modo de varrimento e modo normal, que deverá ter lugar imediatamente após o último ciclo de varrimento. A comutação entre os dois modos é efectuada através da colocação, no registo de instrução, de uma instrução diferente daquela que selecciona a cadeia de varrimento interna, enquanto o último ciclo de varrimento é aplicado com o controlador do TAP no estado *Shift-DR*, pelo que a distância entre estes dois eventos impede a aplicação desta técnica em componentes com cadeias de varrimento internas acessíveis através da infraestrutura BST. Uma outra forma de detectar atrasos entre registos consiste em utilizar-se elementos de varrimento que incluem um segundo registo, normalmente designado por registo "sombra" [Che91]. Esta solução implica no entanto uma quantidade considerável de lógica adicional, que normalmente exclui a sua utilização em circuitos com um elevado número de registos.

3.5 Modos de operação da infraestrutura P1149.4 e capacidades de apoio à depuração

Esta secção descreve a arquitectura básica e blocos principais da infraestrutura de teste definida na proposta de norma IEEE P1149.4 [IEEE98]. Com base nos seus modos de funcionamento básicos e opcionais, analisam-se as formas de implementação das operações de depuração consideradas nas duas secções anteriores, neste caso em simultâneo para vários tipos de sistemas, uma vez que se assume partilharem o mesmo tipo de componente analógica. Embora estas operações tenham sido definidas com base nas necessidades de depuração para circuitos digitais, a sua implementação mantém-se válida em circuitos mistos, em virtude da coexistência de um domínio analógico e um domínio digital.

A proposta P1149.4 encontra-se neste momento em fase final de aprovação como norma do IEEE, esperando-se que em meados do corrente ano seja formalmente aceite e publicada. O P1149.4 define uma infraestrutura (e um protocolo) de teste para componentes mistos, cujo objectivo principal consiste um proporcionar aproximações normalizadas para:

- Teste de interligações, nomeadamente teste de faltas do tipo circuitos abertos e curto-circuitos em pistas de CCI.

- Teste paramétrico, nomeadamente para efectuar medições de caracterização analógica, e testar a presença e o valor de componentes discretos (do tipo resistências, condensadores ou bobinas) inseridos na CCI.
- Teste interno, com os objectivos tornados óbvios por esta designação.

3.5.1 Arquitectura básica do P1149.4

Nesta subsecção apresenta-se a infraestrutura P1149.4, destacando-se os aspectos comuns com a infraestrutura 1149.1 e evidenciando-se as principais diferenças (e suas implicações) entre ambas. A lista seguinte contém as estruturas adicionais definidas na proposta de norma IEEE P1149.4, em comparação com aquelas definidas na norma IEEE 1149.1:

- Um Porto de Acesso ao circuito de Teste Analógico (*Analog Test Access Port*, ATAP) com dois pinos (AT1 e AT2). O pino AT1 é normalmente utilizado para aplicar valores analógicos às entradas do circuito funcional analógico ou aos pinos analógicos, e o pino AT2 é utilizado para monitorar as respostas aos estímulos analógicos (embora a solução descrita em [IEEE98] permita que estes pinos troquem de funções entre si).
- Um barramento interno de teste analógico com pelo menos duas linhas (AB1 e AB2), utilizado para aplicar e monitorar sinais analógicos, no interior e exterior do CI misto (ao nível do seu circuito funcional ou pinos analógicos).
- Um Circuito de Interface com o Barramento interno de Teste analógico (*Test Bus Interface Circuit*, TBIC), inserido entre os pinos do ATAP e as linhas do barramento interno de teste analógico, que define o tipo de ligações e sinais presentes.
- Módulos Analógicos de varrimento Periférico (*Analog Boundary Modules*, ABM), inseridos entre os pinos analógicos e o circuito funcional. Os ABM constituem os equivalentes no domínio analógico das células BS do 1149.1, sendo estas últimas designadas por Módulos Digitais de varrimento Periférico (*Digital Boundary Modules*, DBM[34]) na proposta P1149.4. Note-se que, não fosse a área de circuito adicional, poder-se-iam inserir ABM entre os pinos digitais e a lógica funcional de um CI misto.

A arquitectura básica do P1149.4 pode ser representada da forma ilustrada na Figura 3-11, onde se identificam claramente as quatro estruturas adicionais referidas na lista anterior. Cada pino analógico possui um ABM associado, que providencia capacidade de controlo e observação, permitindo o teste de interligações da CCI e certos tipos de operações de teste paramétrico. Os vários ABM e DBM encontram-se ligados em série, num registo BS estendido, que in-

[34] Repare-se que este acrónimo é idêntico ao utilizado para designar o componente *Digital Bus Monitor* (DBM) da Texas Instruments, embora este facto não seja aqui susceptível de criar ambiguidades.

clui ainda as células pertencentes à estrutura de controlo do TBIC (descrita mais adiante). Para além das estruturas adicionais referidas, a generalidade da arquitectura da infraestrutura de teste é claramente idêntica à definida no 1149.1, como aliás foi inicialmente referido. A operação do P1149.4 pode ser sumariada da seguinte forma: a) um estímulo analógico é externamente aplicado ao pino AT1 e a resposta analógica é monitorada no pino AT2; b) os pinos AT1 e AT2 são ligados às duas linhas (AB1 e AB2) do barramento interno de teste analógico; c) através da linha AB1 pode-se aplicar o estímulo externo a uma entrada analógica do circuito funcional ou a um pino de saída analógico; d) o sinal de resposta, obtido numa saída analógica do circuito funcional ou num pino de entrada analógico, é enviado através da linha AB2.

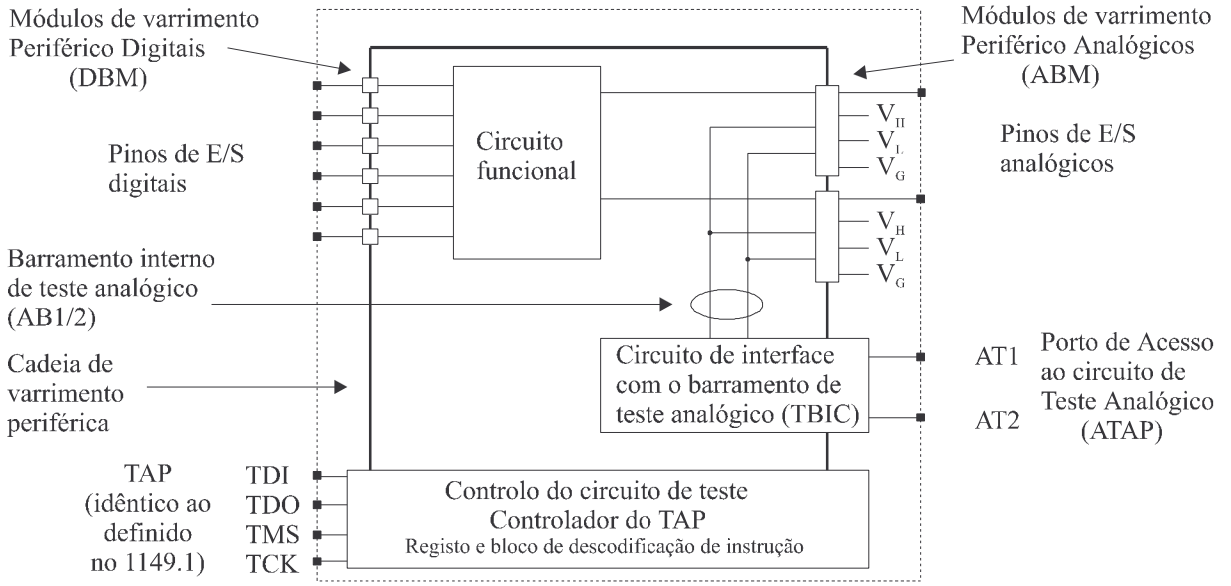


Figura 3-11: Arquitectura básica da infraestrutura definida na proposta de norma IEEE P1149.4.

Estrutura de registos

A arquitectura do P1149.4 pode também ser representada como se ilustra na Figura 3-12, onde se destaca a estrutura de registos de teste. A arquitectura de registos do P1149.4 é inteiramente digital, sendo praticamente igual à definida no 1149.1. O registo BS inclui os registos de controlo do TBIC e dos ABM, que são utilizados para definir os respectivos modos de operação.

Conjunto de instruções

Para além das três instruções obrigatórias definidas no 1149.1, o P1149.4 inclui uma quarta instrução obrigatória, denominada *PROBE*, cuja funcionalidade pode ser sumariada da seguinte forma: a) o registo BS é colocado no percurso TDI – TDO do CI; b) cada ABM liga o respe c-

tivo pino ao circuito funcional; c) os pinos AT1 e AT2, são ligados às linhas AB1 e AB2, respectivamente; d) a ligação entres os pinos analógicos e as linhas AB1 e AB2, é definida de acordo com a combinação presente no registo de controlo de cada ABM; e) os DBM são colocados no modo de funcionamento transparente. Em relação às três instruções obrigatórias comuns ao 1149.1, aplicam-se as seguintes regras específicas do P1149.4:

- Para as instruções *BYPASS* e *SAMPLE / PRELOAD*: a) os pinos AT1 e AT2 devem ficar isolados das linhas AB1 e AB2, e de qualquer fonte de tensão para o teste; b) todos os pinos analógicos devem estar ligados ao circuito funcional; c) todos os pinos analógicos devem estar isolados das linhas AB1 e AB2, e de qualquer fonte de tensão para o teste.
- Para a instrução *EXTEST*: os ABM devem isolar os pinos do circuito funcional.

As instruções opcionais do P1149.4 são iguais às definidas no 1149.1 (*INTEST*, *IDCODE / USERCODE*, *RUNBIST*, *CLAMP* e *HIGH-Z*), sendo descritas de uma forma similar e incluindo apenas algumas regras específicas aos modos de operação do TBIC e dos ABM.

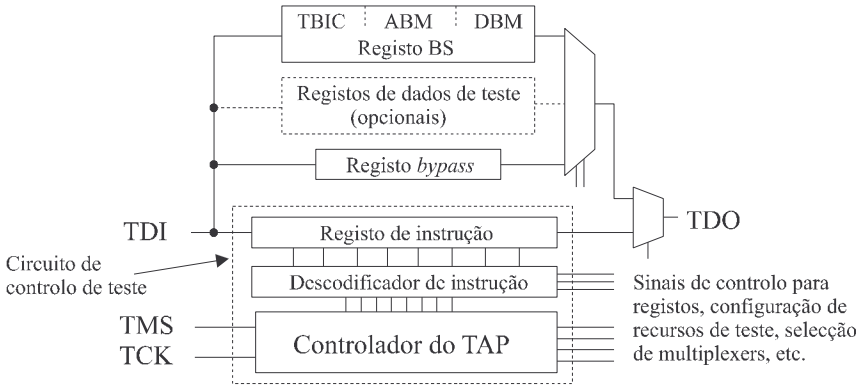


Figura 3-12: Estrutura de registos de teste do P1149.4.

3.5.2 Blocos principais do P1149.4

A arquitectura básica do P1149.4, ilustrada na Figura 3-11, revela a adição de dois blocos principais à infraestrutura original do 1149.1, ilustrada na Figura 3-1, designados por TBIC e ABM. Estes blocos asseguram as principais características do P1149.4, que consistem na aplicação e monitorização de sinais de teste analógicos, possuindo ambos uma estrutura de comutação (que define as interligações entre os seus sinais de E/S) e uma estrutura de controlo (que define o modo de operação da estrutura de comutação). Cada uma destas estruturas (para o TBIC e o ABM) será agora analisada.

Circuito de interface com o barramento de teste analógico (TBIC)

O TBIC controla as interligações entre o ATAP e as linhas do barramento interno de teste analógico. O documento da proposta P1149.4 especifica que devem existir pelo menos duas linhas neste barramento, AB1 e AB2, conforme ilustrado na Figura 3-11. Nesta situação, pode-se representar a estrutura de comutação do TBIC da forma ilustrada na Figura 3-13. Os dez interruptores analógicos representados nesta figura (S1 a S10) definem quais os sinais presentes nos pinos do ATAP e nas linhas AB1 e AB2. A estrutura de comutação do TBIC permite assim, entre outras combinações: a) ligar o pino AT1 ou AT2 a V_H ou V_L; b) ligar o pino AT1 ou AT2 a AB1 ou AB2; c) ligar o pino AT1 ou AT2 à fonte de tensão interna V_{CLAMP}; d) obter uma representação, em um bit, da tensão em AT1 ou AT2 (em comparação com a tensão de limiar V_{TH}). A condição (aberto / fechado) dos interruptores representados na Figura 3-13 é definida pela estrutura de controlo do TBIC. Uma vez que nem todas as combinações possíveis são úteis (ou legais), quatro bits apenas são suficientes para seleccionar o modo de operação pretendido, tal como se ilustra na Figura 3-14 (a actual especificação do P1149.4 descreve apenas dez combinações, que satisfazem os requisitos básicos definidos para o TBIC). A estrutura de controlo do TBIC é parte integrante do registo BS, podendo ser descrita resumidamente da seguinte forma:

- Possui quatro células (bits), cada uma constituída por um A_D e um A_R. Estas células não diferem significativamente da configuração típica sugerida no 1149.1, residindo a principal diferença na inexistência de um multiplexador de saída, que neste caso não é necessário por não existir uma entrada (externa) paralela para o A_R.
- Os quatro bits do A_R são designados por CALIBRATE, CONTROL, DATA1 e DATA2. Estes dois últimos capturam uma representação, em um bit, das tensões presentes em AT1 e AT2, em comparação com V_{TH}, enquanto que os dois primeiros se encontram disponíveis.
- Os quatro bits no A_R, em conjunto com os sinais de saída, MODE1 e MODE2, do descodificador de instruções, definem o modo de operação para a estrutura de comutação do TBIC.

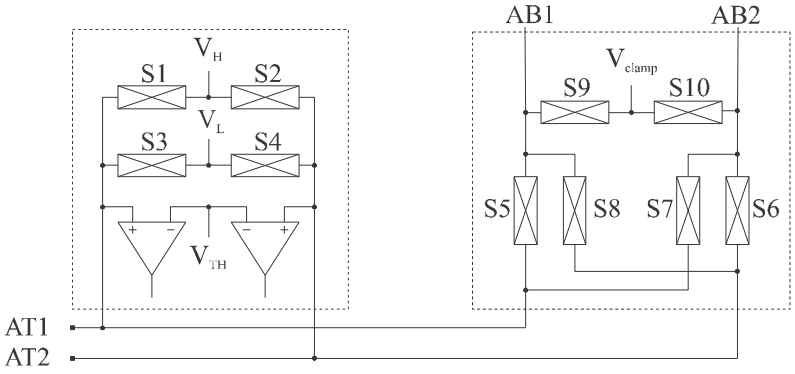


Figura 3-13: Estrutura de comutação do TBIC.

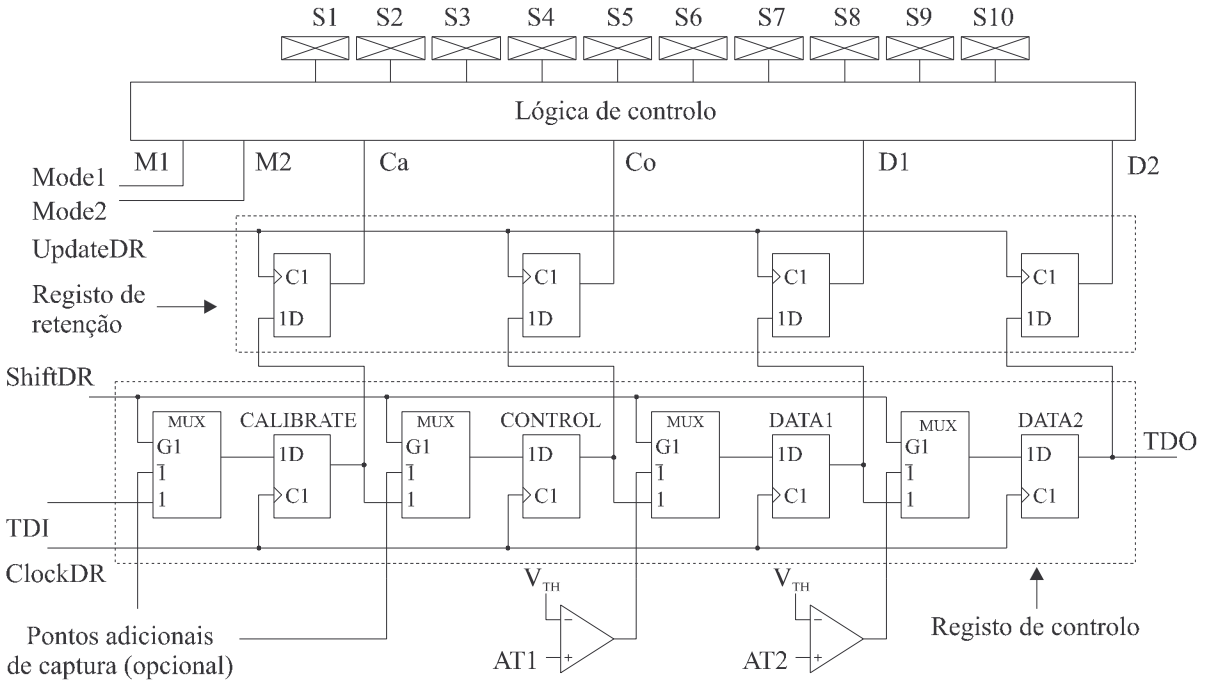


Figura 3-14: Estrutura de controlo do TBIC.

Módulos de varrimento periférico analógico (ABM)

Os ABM constituem o núcleo central da arquitectura do P1149.4. A aplicação de sinais de teste e a captura das respectivas respostas é efectuada através destes módulos, combinando o acesso série via registo BS e o acesso analógico via ATAP. De um modo semelhante ao TBIC, os ABM compreendem uma estrutura de comutação e uma estrutura de controlo, para definir o fluxo dos sinais analógicos em ambos os sentidos, nos pinos funcionais analógicos. A estrutura de comutação ilustrada na Figura 3-15 compreende seis interruptores, que proporcionam as seguintes funções: a) desligar um pino analógico do circuito funcional; b) ligar o pino à linha AB1 (controlabilidade); c) ligar a linha AB2 ao pino (observabilidade); d) obter uma representação, em um bit, da tensão no pino (em comparação com V_{TH}); e) ligar o pino a V_H ou V_L (para teste de interligações); f) ligar uma tensão de referência de elevada precisão (V_G) ao pino (útil para medições paramétricas). A estrutura de controlo destes módulos pode ser representada como se ilustra na Figura 3-16. Esta estrutura é similar à utilizada no TBIC, podendo ser descrita da seguinte forma:

- Possui quatro células (bits), cada uma constituída por um A_D e um A_R.
- Os quatro bits no A_R são designados por DATA, CONTROL, BUS1 e BUS2. O primeiro captura uma representação, em um bit, da tensão presente no pino funcional, em comparação com V_{TH}, enquanto que os restantes bits se encontram disponíveis.

- Os quatro bits no A_R, em conjunto com os sinais de saída, MODE1 e MODE2, do descodificador de instruções, definem o modo de operação para a estrutura de comutação do ABM.

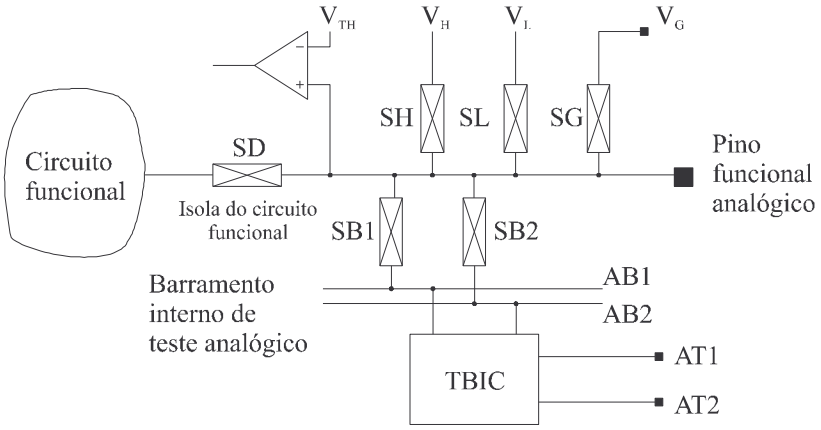


Figura 3-15: Estrutura de comutação dos ABM.

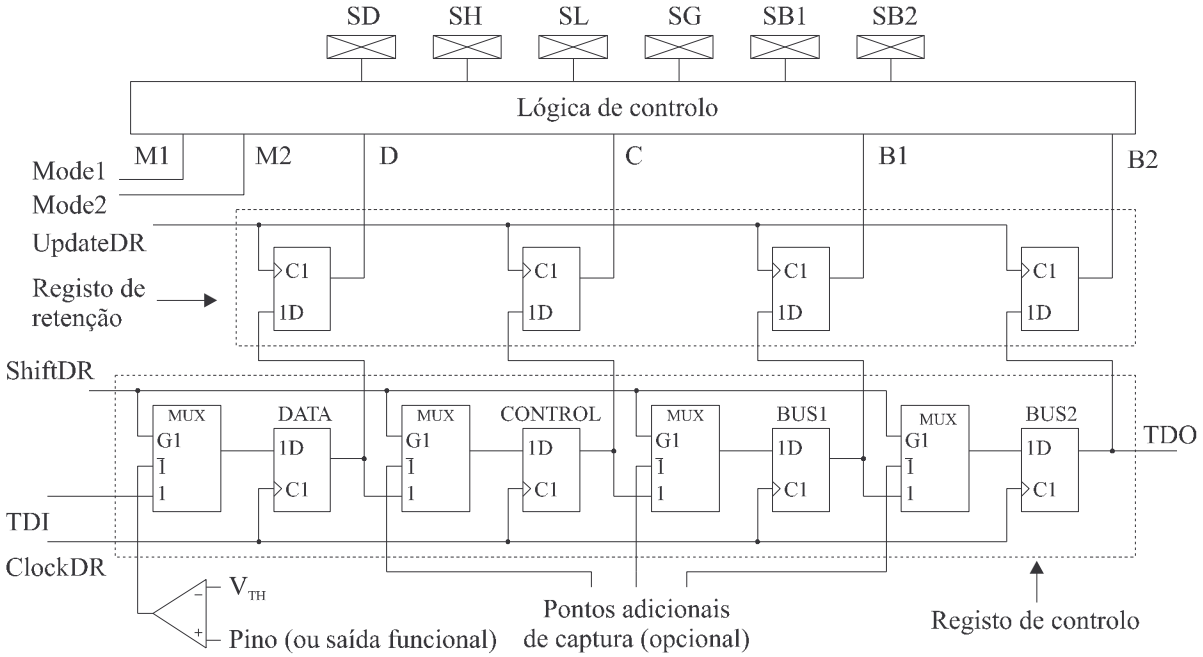


Figura 3-16: Estrutura de controlo dos ABM.

3.5.3 Implementação das operações de depuração via P1149.4

Com base nos modos de operação básicos e opcionais do P1149.4, apresentados detalhadamente na subsecção anterior, procede-se em seguida à análise das formas de implementação das operações de depuração que constam do modelo ilustrado na Figura 2.14.

Operações de COV

Em relação ao domínio analógico, exclui-se a implementação de operações de COV do estado de registos e portas lógicas, uma vez que estes elementos pertencem ao domínio digital. As operações de COV do estado de pinos analógicos podem ser implementadas através das capacidades disponibilizadas pelo P1149.4, que permitem um acesso directo através dos pinos do ATAP, a qualquer pino analógico de um CI misto, que seja compatível com esta proposta de norma. Para efeitos de controlo, em modo intrusivo, do valor analógico presente num determinado pino[35], é necessário efectuar o seguinte conjunto de acções:

- Deslocar a instrução *SAMPLE / PRELOAD* para o componente misto a que pertence o pino pretendido, e a instrução *BYPASS* para os restantes componentes. Nesta situação as linhas dos barramentos internos analógicos dos vários componentes mistos são ligadas a uma fonte de tensão interna (V_{CLAMP}) para supressão de ruído. Os pinos AT1 e AT2 ficam desligados de qualquer linha ou fonte de tensão interna.
- Deslocar: a) o vector[36] para as células da estrutura de controlo do TBIC do componente seleccionado ('CALIBRATE, CONTROL, DATA1, DATA2' = '0010') de forma a ligar AT1 à linha AB1 e a linha AB2 à fonte de tensão interna V_{CLAMP}; b) o vector para as células da estrutura de controlo do ABM associado ao pino seleccionado ('DATA, CONTROL, BUS1, BUS2' = '0010'), que permite simultaneamente ligá-lo à linha AB1 e desligá-lo do circuito funcional analógico; c) o vector que liga os restantes pinos analógicos do componente a uma das três fontes de tensão (V_L, V_H ou V_G) existentes na estrutura de comutação do ABM (por exemplo, para ligar a V_L deve-se deslocar o vector '1000' para cada conjunto de quatro células dos restantes ABM).
- Aplicar o sinal pretendido à linha AT1 da CCI, que poderá ser um determinado valor fixo em tensão, uma sinusóide, ou outro qualquer estímulo analógico.
- Deslocar a instrução *EXTEST* para o componente misto a que pertence o pino pretendido, e a instrução *BYPASS* para os restantes componentes. Quando o controlador do TAP passar no estado *Update-IR*, a instrução ficará activa e o valor presente na linha AT1 será aplicado ao pino analógico seleccionado (aos restantes pinos analógicos será aplicada a tensão V_L). Deve-se salientar neste ponto que o pino analógico fica permanentemente ligado à linha

[35] Que se supõe pertencer a um componente misto inserido numa cadeia que contém outros componentes (mistos e digitais), todos ligados em série via TDI – TDO, *e* ligados em barramento via AT1 - AT2 (componentes mistos). Pode-se controlar simultaneamente um pino de cada componente misto, no caso de se ligar os pinos do ATAP numa configuração em paralelo, ou seja, o pino AT1 de cada componente ligado a uma entrada dedicada da CCI.

[36] Estes vectores são determinados com base nas tabelas existentes em [IEEE98, pág. 47-48 e pág. 66-68], que contêm os padrões de comutação para o TBIC e os ABM, respectivamente.

AT1 (enquanto se mantiverem os vectores actuais nas estruturas de controlo do TBIC e de cada um dos ABM), pelo que qualquer variação no sinal presente nesta linha se reflectirá no pino analógico, após um determinado atraso de propagação.

Para efeitos de observação, em modo não intrusivo, do valor analógico presente num determinado pino[37], é necessário efectuar o seguinte conjunto de acções:

- Deslocar a instrução *SAMPLE / PRELOAD* para o componente misto a que pertence o pino pretendido, e a instrução *BYPASS* para os restantes componentes. Nesta situação, as linhas dos barramentos internos analógicos dos vários componentes mistos são ligadas a uma fonte de tensão interna (V_{CLAMP}), para supressão de ruído. Os pinos AT1 e AT2 ficam desligados de qualquer linha ou fonte de tensão interna.
- Deslocar: a) o vector para as células da estrutura de controlo do TBIC do componente seleccionado ('CALIBRATE, CONTROL, DATA1, DATA2' = '0001') de forma a ligar AT2 à linha AB2 e a linha AB1 à fonte de tensão interna V_{CLAMP}; b) o vector para as células da estrutura de controlo do ABM associado ao pino seleccionado ('DATA, CONTROL, BUS1, BUS2' = '0001'), que permite ligá-lo à linha AB2; c) o vector que liga os restantes pinos analógicos ao circuito funcional (via SD).
- Deslocar a instrução *PROBE* para o componente misto a que pertence o pino pretendido, e a instrução *BYPASS* para os restantes componentes. Quando o controlador do TAP passar no estado *Update-IR*, a instrução ficará activa e o valor presente no pino pretendido poderá ser observado em AT2 (via linha interna AB2). Deve-se salientar neste ponto que o pino analógico fica permanentemente ligado à linha AT2 (enquanto se mantiverem os vectores actuais nas estruturas de controlo do TBIC e de cada um dos ABM), pelo que qualquer variação do sinal presente no pino analógico se reflectirá na linha AT2, após um determinado atraso de propagação.

Operações passo-a-passo

As operações passo-a-passo definidas para o domínio digital fazem pouco sentido quando transpostas para o domínio analógico, em virtude da natureza contínua deste tipo de circuitos. Na situação em que se avança um passo no estado da parte digital de um sistema misto, e se pára o fornecimento de impulsos de relógio, dificilmente se pode aferir se a parte analógica continua, ou não, a efectuar a sua função, uma vez que essa continuidade depende da relação

[37] Pode-se observar simultaneamente o valor analógico presente num pino de cada componente misto, no caso de se ligar os pinos do ATAP numa configuração em paralelo, ou seja, o pino AT2 de cada componente ligado a uma saída dedicada da CCI.

ou interface existente entre as duas partes. Neste sentido, não se explora em maior profundidade a utilização das capacidades do P1149.4 na implementação deste tipo de operação.

Operações de pontos de paragem por condição

O tipo de condições de paragem que se podem especificar em circuitos mistos engloba valores de natureza contínua referentes a grandezas em tensão, corrente ou frequência. As capacidades disponibilizadas pelo P1149.4 permitem a avaliação deste tipo de condições, em tempo real, através dos pinos do ATAP. Na situação em que se ligam estes pinos em barramento, i.e. todos os pinos AT1 e AT2 de componentes mistos ligados em paralelo, através de duas linhas, respectivamente, é possível especificar (e avaliar) condições que englobem valores presentes em dois pinos analógicos pertencentes a um mesmo componente, ou a dois componentes (um pino em cada componente). De facto, a estrutura de comutação do TBIC e dos ABM permite a monitorização em simultâneo dos valores presentes em dois pinos analógicos, num caso através da linha AB1 ligada a AT1 e no outro através da linha AB2 ligada a AT2. Esta possibilidade de reversão ou duplicação dos papéis normalmente associados a cada uma destas linhas do barramento interno de teste analógico, e dos pinos do ATAP, é referida no P1149.4. Na situação em que se pretenda estender a condição de paragem a um maior número de pinos analógicos de diferentes componentes (com um limite máximo de dois pinos por componente), devem-se ligar os pinos dos vários ATAP a saídas dedicadas independentes da CCI, o que implica no entanto um custo adicional, em virtude do maior número de ligações e saídas da CCI.

Refira-se ainda que é possível detectar condições de paragem que correspondam ao vector definido pela comparação, com a tensão de limiar V_{TH}, do valor presente em cada um dos pinos analógicos de um ou vários componentes mistos. Esta possibilidade decorre da existência de um comparador em cada ABM, conforme se pode verificar através da Figura 3-15. O resultado da comparação é capturado na célula denominada DATA, da estrutura de controlo de cada ABM. Utilizando o procedimento descrito na secção 3.3, baseado na utilização cíclica da instrução *SAMPLE / PRELOAD* para determinar o valor lógico da condição entre duas transições sucessivas do relógio do sistema (que neste caso poderá ou não existir, em face da função executada pelo sistema em causa), pode-se estender a condição de paragem a todos os pinos analógicos dos componentes que disponham de uma infraestrutura P1149.4. Esta condição, embora mais abrangente em termos de número de pinos, não pode ser avaliada em tempo real, em virtude da necessidade de deslocamento para o exterior dos valores capturados, que dão apenas uma imagem da tensão existente em cada pino, em relação à tensão de limiar V_{TH}. Outra restrição importante decorre do facto de não existir qualquer tipo de sincronismo entre o momento da captura (determinado por TCK) e a operação da parte analógica do componente.

Operações de análise em tempo real

Dado que o valor presente em qualquer pino analógico pode ser observado em tempo real através da linha AB2 e do pino AT2, utilizando a instrução *PROBE*, considera-se exequível a implementação de operações de visualização em tempo real em componentes mistos.

A detecção de faltas do tipo atraso entre pinos analógicos, pertencentes a componentes que disponham de uma infraestrutura P1149.4, é considerada possível e implementável através das capacidades de controlo / observação em tempo real dos valores presentes nos pinos analógicos. Repare-se que estas capacidades permitem medir o tempo de propagação em linhas que unam dois quaisquer pinos analógicos, um controlado através de AT1 e o outro observado através de AT2.

3.6 Conclusão

Algumas das formas de implementação descritas neste capítulo foram inicialmente estudadas e desenvolvidas no âmbito do projecto JNICT PBIC/C/TIT/2474/95, referido no capítulo introdutório. No primeiro relatório deste projecto [Alv96] foram analisadas em detalhe as formas de implementação de operações de depuração, utilizando os modos de operação básicos e opcionais do 1149.1 (incluindo os existentes nos componentes da família SCOPE^{TM}), tendo esta análise resultado na especificação de um sistema de demonstração, baseado num microcontrolador, com capacidades de depuração via 1149.1. O desenvolvimento e a posterior utilização deste sistema, descrito em [Alv98a], permitiu identificar algumas das lacunas associadas aos modos de operação básicos e opcionais do 1149.1, nomeadamente na sincronização entre a lógica funcional e a lógica de teste, na implementação de pontos de paragem por condição (com a avaliação em tempo real da condição de paragem) e de operações de análise em tempo real. Refira-se que nele foram utilizados componentes da família SCOPE^{TM}, nomeadamente dois DBM, conforme se pode verificar pelo diagrama da Figura 3-9, que ilustra a sua arquitectura de uma forma simplificada.

A utilização do P1149.4 não foi inicialmente considerada no referido projecto em face do estado desta proposta de norma no momento em que ele teve início. Porém, em face do desenvolvimento desta especificação, tornou-se evidente a necessidade de analisar, ainda que apenas ao nível básico, a utilização das suas capacidades para a implementação das operações de depuração consideradas.

Capítulo 4

Projecto para o teste e depuração: análise de requisitos

Este capítulo analisa os requisitos a que deve obedecer o projecto para o teste e depuração, de forma a facilitar a verificação estrutural e funcional de sistemas electrónicos nas etapas de desenvolvimento e produção, e ainda em ambiente de utilização final. A reutilização das infraestruturas de teste 1149.1 e P1149.4, para a implementação das operações de depuração apresentadas no capítulo 2, foi neste sentido amplamente analisada no capítulo 3, com o intuito de identificar as capacidades disponíveis para esse fim e as suas principais lacunas. No seguimento desse exercício definem-se neste capítulo as linhas directoras para o desenvolvimento de uma infraestrutura para o teste e depuração, que constitui o primeiro dos três pilares em que assenta o trabalho efectuado no âmbito desta dissertação. Com base nos requisitos de projecto para o teste e depuração, definem-se em seguida as linhas directoras para o desenvolvimento de um controlador residente, que enderece simultaneamente o controlo da infraestrutura proposta e a sua sincronização com a lógica funcional do sistema, para implementar as operações básicas de depuração. O controlador residente constitui o segundo pilar, correspondendo o terceiro ao desenvolvimento de uma metodologia de utilização da infraestrutura proposta e de um módulo de geração automática do programa de teste e depuração (executado pelo controlador).

Este capítulo permite ao mesmo tempo introduzir a apresentação global do trabalho realizado, descrito mais detalhadamente ao longo dos três próximos capítulos:

- A segunda secção apresenta a proposta de uma infraestrutura para o teste e depuração, descrita no capítulo 5.
- A terceira secção apresenta a arquitectura do controlador residente, descrita no capítulo 6.
- A quarta secção apresenta a metodologia e geração automática do programa de teste e depuração, descrita no capítulo 7.

4.1 Requisitos de projecto para o teste e depuração

Os requisitos de projecto para o teste e depuração decorrem da análise do fluxo de verificação identificado para os sistemas electrónicos: simulação, teste estrutural, depuração funcional e depuração temporal.

O modelo empregue na simulação funcional deve incluir a infraestrutura de teste de forma a permitir a verificação, nas fases iniciais de projecto, da interacção entre a lógica funcional e a lógica de teste, e da utilização desta última na implementação de algumas das operações de depuração. Este requisito tem diferentes implicações em face do nível hierárquico considerado. Ao nível do CI, é usual o projecto começar por níveis abstractos elevados, onde não é possível utilizar ferramentas de inserção automática de infraestruturas de teste [Man96]. A partir do momento em que se define a estrutura da interface do circuito em desenvolvimento, ao nível dos seus pinos individuais de entrada e saída, pode-se acoplar ao modelo da lógica funcional, um modelo que descreva a infraestrutura de teste mínima, formando o conjunto um modelo completo do CI, conforme se ilustra na Figura 4-1. Esta possibilidade resulta do facto da interacção entre a lógica funcional e a lógica de teste se situar apenas ao nível das entradas e saídas da lógica funcional, via registo BS. Recorde-se que a infraestrutura mínima definida na norma IEEE 1149.1 corresponde à implementação das três instruções obrigatórias, que não contemplam a existência de outros registos, para além do registo BS e do registo *bypass*. A implementação das instruções opcionais *INTEST*, *RUNBIST*, ou de acesso a cadeias de varrimento internas, implica um tipo de interacção que vai para além do registo BS. Se os modos de operação propostos para a infraestrutura para o teste e depuração, implicarem apenas pequenas modificações ao nível da infraestrutura mínima, e não criarem mais interacções entre a lógica de teste e a lógica funcional, torna-se possível a sua inclusão no referido modelo, o que permite a sua utilização nas fases iniciais de simulação funcional. Do cumprimento deste requisito resultam vantagens adicionais para a verificação ao nível hierárquico da CCI[38].

A existência de modelos de todos os CI inseridos na CCI, que incluam igualmente a infraestrutura para o teste e depuração, permite que se simule a execução das operações de depuração a este nível hierárquico. Refira-se ainda que é possível incluir componentes destinados a apoiar a realização de testes estruturais ou de operações de depuração [Bal89, Ben84, Mat93, Mau92a, Rag91, Wen96, Whe91]. No caso da CCI se inserir num sistema de nível hierárquico superior, baseado por exemplo na ligação de várias CCI através de um barramento comum, pode-se incluir ainda um componente dedicado à interligação entre os barramentos de teste existentes num e noutro nível hierárquico [And94, Bhav91, Cru94, Hab94, Lun92, McH94, Whe92, Whe93]. A inserção de um controlador residente que permita efectuar o teste estrutural

[38] Supondo que o projecto da CCI e de um (ou mais) CI nela inserido(s) decorrem em paralelo.

da CCI, sob a forma de um auto-teste, constitui outra opção [Fer92a, Fer92b, Jar91, Kon96, Mat92], que beneficia particularmente dois aspectos:

- A possibilidade de incluir um modelo do controlador para efeitos de simulação do funcionamento da infraestrutura de teste da CCI, tal como se ilustra na Figura 4-2.
- O aumento da rapidez do teste estrutural de um sistema formado por várias CCI, supondo que todas elas efectuam o auto-teste em paralelo.

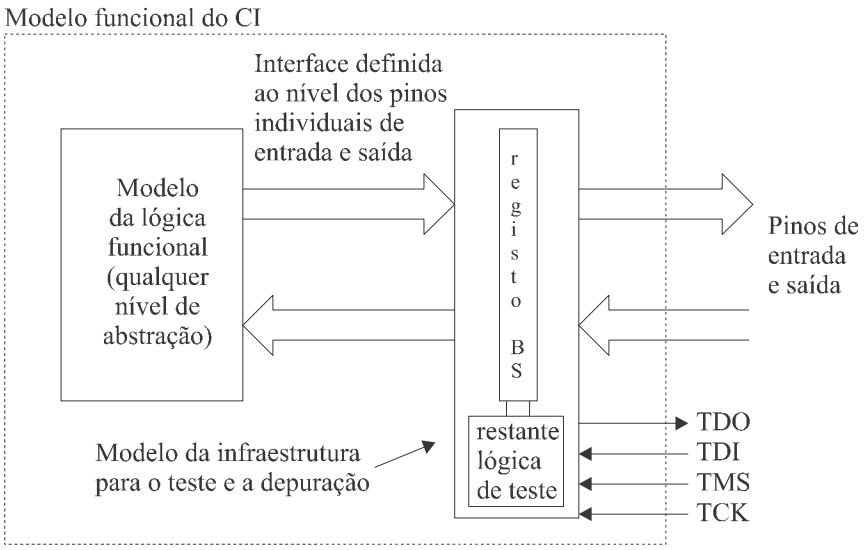


Figura 4-1: Criação de um modelo funcional do CI, através da junção de um modelo da lógica funcional (a qualquer nível de abstracção) e de um modelo da lógica de teste, com a restrição de apenas existir interacção ao nível do registo BS.

Note-se porém que os controladores actualmente disponíveis no mercado [SCA94, SCO94] suportam apenas o controlo de uma infraestrutura compatível com a norma IEEE 1149.1, para efeitos de teste estrutural. É pois necessário definir novos requisitos para este componente, que incluam a capacidade de controlar os modos de operação propostos para a infraestrutura para o teste e depuração, e de garantir o sincronismo entre a lógica funcional e a lógica de teste, através nomeadamente da sincronização entre os respectivos relógios. Este último requisito podia igualmente ser colocado ao nível do CI, porém com a desvantagem de se aumentar a interacção entre a lógica funcional e a lógica de teste, que tal como se referiu anteriormente impede que se antecipe a disponibilização de um modelo conjunto que inclua ambas as partes. O controlador deverá ainda dar apoio à implementação das operações de depuração que não dependam directa ou exclusivamente da infraestrutura para o teste e depuração, como por exemplo operações passo-a-passo ou a aplicação do algoritmo de alargamento de um impulso de relógio.

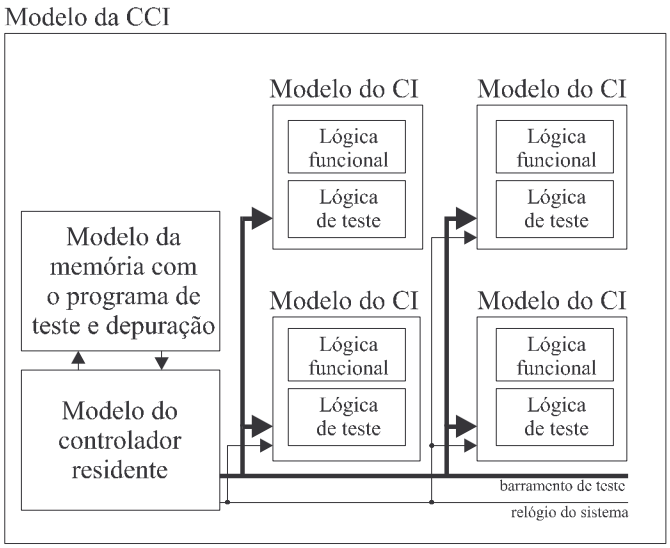


Figura 4-2: Inclusão do modelo do controlador residente no modelo da CCI.

O último requisito relaciona-se com a capacidade de gerar automaticamente o programa executado pelo controlador residente, a partir da informação que vai sendo disponibilizada ao longo da evolução do projecto do CI, ou da CCI onde será posteriormente inserido. Este requisito é particularmente importante, dado que qualquer tipo de sistema sofre diversas transformações ao longo do seu projecto, o que implica tempos de atraso consideráveis no caso da geração do programa se basear num processo com um elevado índice de intervenção humana, e por conseguinte lento e passível de introdução de um maior número de erros. A Figura 4-3 ilustra uma visão geral do processo de geração automática do programa de teste e depuração, onde se identifica a informação de entrada necessária.

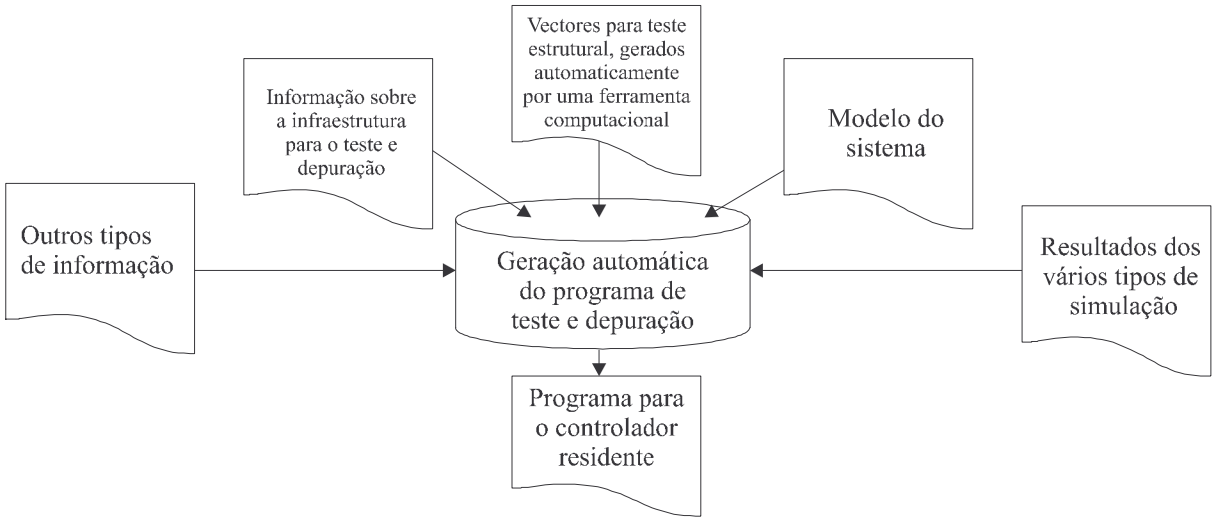


Figura 4-3: Informação de entrada para a geração automática do programa de teste e depuração.

Os vários requisitos identificados nesta secção serão agora expandidos nas três secções seguintes, dedicadas: 1) à infraestrutura para o teste e depuração; 2) ao controlador residente; 3) à metodologia e geração automática do programa de teste e depuração.

4.2 Infraestrutura para o teste e depuração

Os requisitos identificados na primeira secção, para uma infraestrutura para o teste e depuração, incluem:

- O preenchimento das principais lacunas identificadas na implementação das operações de depuração, através dos modos de operação básicos e opcionais das infraestruturas 1149.1 e P1149.4. O preenchimento destas lacunas deverá ser obtido através da especificação de novos modos de operação, que resultem num conjunto de alterações a introduzir nas infraestruturas de teste. As principais lacunas identificadas no capítulo anterior situam-se:

i. Na implementação de pontos de paragem por condição, mais especificamente na detecção em tempo real da condição de paragem.

ii. Na capacidade de efectuar a aquisição de valores em tempo real.

iii. Na capacidade de detectar tempos de atraso entre pinos, em interligações externas.

- A redução e simplificação do número e tipo de alterações a introduzir, ao nível da infraestrutura básica definida no 1149.1, ou no P1149.4, de forma a não criar mais interligações / interacções entre a lógica funcional e lógica de teste de um CI.
- A possibilidade de incluir os novos modos de operação num modelo inicial da infraestrutura para o teste e depuração, que possa ser acoplado ao modelo da lógica funcional (qualquer que seja o seu nível de abstracção, desde que exista uma definição dos pinos individuais de entrada e saída).

Em relação à primeira (i) das três lacunas identificadas pondera-se a possibilidade de utilizar o modo de observação não intrusivo dos valores presentes nos pinos de entrada e saída do CI, para permitir detectar condições de paragem referentes a estes elementos. Dado que quando se implementa esta operação de depuração o sistema se encontra em funcionamento normal e a lógica de teste se encontra tipicamente inactiva, armazena-se no registo BS a condição a detectar[39]. A avaliação da condição é efectuada através de um circuito posteriormente englobado

[39] Supondo que todas as células do registo BS possuem um A_D e um A_R, existirão 2**n* elementos sequenciais para armazenar a condição a detectar (*n* representa o número de pinos do CI, excluindo os do TAP e de alimentação).

na lógica de teste, de acordo com o último dos três requisitos da lista anterior. A sinalização para o exterior do valor lógico da condição de paragem obriga à inclusão de um pino adicional, dado que não se pode utilizar para este efeito qualquer pino do TAP. A Figura 4-4 ilustra o esquema conceptual deste modo de operação opcional.

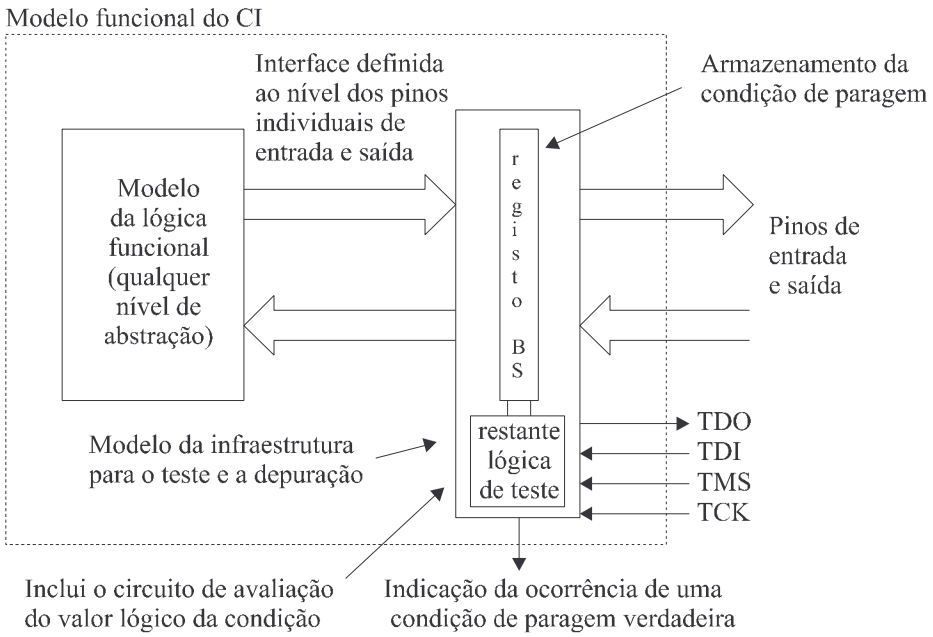


Figura 4-4: Detecção em tempo real de condições de paragem, referentes a valores presentes nos pinos de entrada e saída de um CI, através de um modo de operação opcional da infraestrutura para o teste e depuração.

Em relação à segunda lacuna (ii) identificada, pondera-se a possibilidade de armazenar no registo BS, em tempo real, os valores presentes nos pinos de entrada e saída do CI. A amostragem é efectuada utilizando o modo de observação não intrusivo existente nas células BS. A sincronização entre o momento da amostragem e a actividade da lógica funcional é assegurada através da sincronização dos respectivos sinais de relógio. Dado que a capacidade de armazenamento do registo BS é limitada pelo número de elementos sequenciais originalmente existentes (dois por cada célula), a profundidade de amostragem está limitada a dois vectores. Nestas circunstâncias, os A_D e A_R de cada célula são utilizados para armazenar os valores capturados no respectivo pino e cada novo valor capturado substitui o mais antigo dos dois valores armazenados. A Figura 4-5 ilustra o esquema conceptual deste modo de operação opcional a ser suportado pela infraestrutura para o teste e depuração.

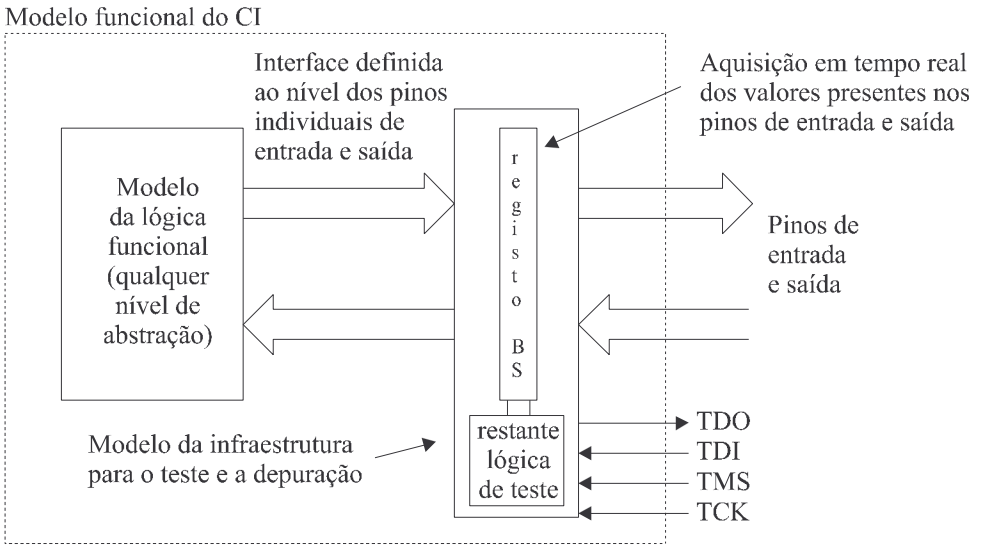


Figura 4-5: Aquisição em tempo real dos valores presentes nos pinos de entrada e saída de um CI, através de um modo de operação opcional da infraestrutura para o teste e depuração.

Em relação à terceira e última (iii) lacuna identificada, pondera-se a possibilidade de reduzir a distância temporal que separa o momento em que são actualizados os valores nas saídas do CI (no estado *Update-DR*), do momento em que são capturados os valores presentes nos pinos de entrada (no estado *Capture-DR*). Apesar de já existir uma proposta de alteração da infraestrutura BST para este fim [Lof96], a necessidade de incluir um elemento sequencial extra em cada célula BS viola o segundo requisito estabelecido para a infraestrutura para o teste e depuração. A partir do modo de geração de estímulos pseudo-aleatórios e de formação de uma assinatura, em simultâneo, existente em alguns componentes da família SCOPE^{TM}, verificou-se a possibilidade de utilizar o estado *Run-Test/Idle* para aplicar e capturar valores, de que resulta uma distância temporal de apenas meio ciclo de relógio de teste entre estes dois momentos (aplicação e captura). Em face das desvantagens da geração pseudo-aleatória e da compressão dos valores capturados, considera-se em alternativa a aplicação nos pinos de saída de valores armazenados no A_R, a que se segue a captura no A_D dos valores presentes nos pinos de entrada, sem qualquer tipo de processamento. De forma a reduzir ainda mais a distância temporal entre estes dois momentos, altera-se o ciclo de trabalho (*duty cycle*) do relógio de teste, sem alterar a sua frequência – por forma a respeitar os limites admissíveis para este sinal, quando o controlador do TAP se encontra no estado *Run-Test/Idle* e este modo de operação opcional se encontra activo. A observação destas duas condições anteriores permite restringir a análise do impacto da alteração proposta do ciclo de trabalho.

4.3 Controlador residente de teste e depuração

Os requisitos identificados na primeira secção, para um controlador residente de teste e depuração, incluem:

- A capacidade de controlar os modos de operação básicos e opcionais da infraestrutura de teste, definidos na norma IEEE 1149.1. Esta capacidade destina-se a permitir a execução do teste estrutural da CCI, na forma de um auto-teste, e a implementação das operações de depuração que incluam apenas a utilização dos referidos modos de operação. Para este efeito, considera-se a possibilidade de reutilizar o núcleo de um controlador desenvolvido no âmbito de anteriores trabalhos de investigação na área do teste estrutural de CCI [Fer92a, Fer92b, Fer92c, Fer93]. Este núcleo permite controlar duas cadeias BS, possuindo ainda canais de sincronismo com o exterior, que podem ser utilizados, por exemplo, quando existem equipamentos de teste externos para as E/S primárias da CCI.
- A capacidade de controlar os modos de operação opcionais propostos para a infraestrutura para o teste e depuração. Esta capacidade deverá ser definida após a determinação das acções que ocorrem ao nível do TAP, ou de outros pinos dedicados, durante a implementação das operações que utilizam os modos opcionais propostos.
- A sincronização entre a lógica funcional e a lógica de teste, através do controlo simultâneo e em sincronismo de ambos os sinais de relógio[40]. Estes sinais deverão ser derivados do relógio de entrada, que regula o funcionamento do controlador.
- O apoio à implementação de operações de depuração que não dependam directa ou exclusivamente da infraestrutura para o teste e depuração, como por exemplo operações passo-a-passo ou aplicação do algoritmo de alargamento de um impulso de relógio. Inclui-se neste ponto a capacidade de implementar operações de COV de pinos, seja para verificar a ocorrência de uma condição de paragem verdadeira, seja para controlar a compatibilidade[41] com a norma IEEE 1149.1 [Alv97b, Jar94].

Com base nestes requisitos, é possível delinear um primeiro esboço da interface, ou seja, dos pinos de entrada e saída do controlador, conforme se ilustra na Figura 4-6. Em face do desenvolvimento da infraestrutura proposta, poderá resultar a necessidade de incluir mais pinos de

---

[40] Refira-se neste ponto que foi recentemente disponibilizado no mercado um controlador, para instalação em computadores pessoais, que permite o controlo simultâneo e em sincronismo do relógio de teste e do relógio do sistema [Tem98].

[41] A norma IEEE 1149.1 define a possibilidade da existência de um ou mais pinos de entrada dedicados, que permitem comutar o modo de funcionamento (compatível ou não com o 1149.1) da infraestrutura de teste existente num componente. Esta possibilidade encontra-se descrita em [IEEE93, pág. 3-7 a 3-9].

entrada e saída. Dado que o capítulo que apresenta esta infraestrutura precede aquele em que se apresenta o controlador, esta informação prossegue sequencialmente de um para outro. A re-utilização do núcleo de um controlador para o auto-teste de CCI pode ainda implicar a inclusão de mais pinos, não identificados nesta secção, dado que não se referem aqui todos os pormenores da sua arquitectura e interface com o exterior.

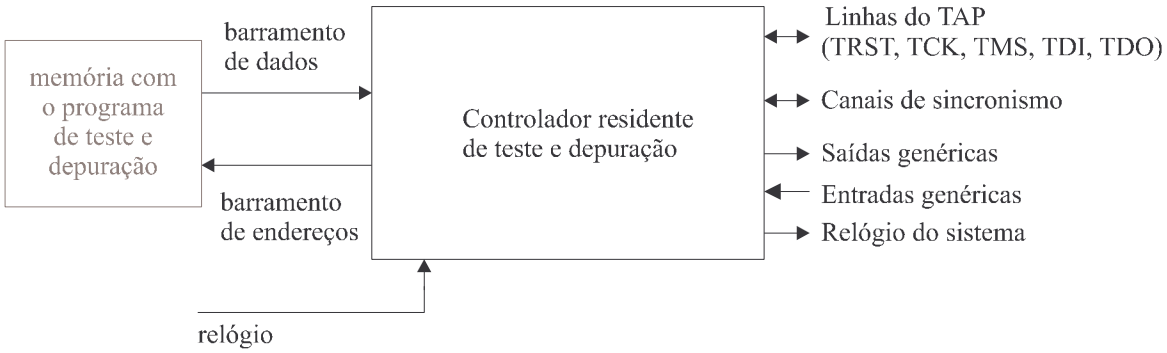


Figura 4-6: Definição inicial dos pinos de entrada e saída do controlador residente de teste e depuração.

4.4 Geração automática do programa de teste e depuração

Os requisitos identificados na primeira secção, para o processo de geração automática do programa de teste e depuração, incluem:

- A definição de uma metodologia para a utilização dos modos de operação opcionais da infraestrutura proposta. Esta metodologia influencia directamente a geração do programa de teste e depuração, dado que define uma sequência de execução de acções (que poderão ocorrer em simultâneo) referentes à lógica funcional e à lógica de teste. A metodologia depende ainda do tipo e nível hierárquico do sistema sob depuração.
- A utilização imediata e automática da informação que vai sendo disponibilizada à medida que o projecto avança ao longo das suas etapas. Esta informação depende igualmente do tipo e nível hierárquico do sistema sob depuração. Em termos do desenvolvimento da ferramenta computacional de geração automática do programa de teste e depuração, consideram-se os níveis do CI e da CCI. O modelo do CI constitui o primeiro item do conjunto de informação de entrada requerido pela ferramenta computacional e deverá incluir a funcionalidade da infraestrutura de teste, de acordo com o ilustrado na Figura 4-1. No caso de uma CCI são ainda necessários os modelos de todos os CI e uma descrição das interligações existentes. O modelo do controlador residente é acoplado ao modelo do sistema, permitindo assim controlar a infraestrutura de teste presente. São ainda necessários outros elementos de informação sobre a infraestrutura de teste, nomeadamente os que descrevem o

registo BS, os códigos das instruções, a forma de aceder a cadeias de varrimento internas (via TDI – TDO), etc. O ficheiro que contém esta informação baseia-se numa linguagem normalizada de descrição da infraestrutura 1149.1, que será referida em maior detalhe no capítulo 7. Cada CI com uma infraestrutura de teste possui um destes ficheiros, pelo que no caso de uma CCI existirá uma base de dados que inclui o conjunto destes ficheiros. Os estímulos de entrada utilizados nas várias sessões de simulação e o conjunto das respostas obtidas, fazem igualmente parte da informação de entrada da ferramenta a desenvolver. Os vectores para o teste estrutural, do CI ou da CCI, são importados a partir de outras ferramentas que os geram automaticamente, dependendo este processo da testabilidade do circuito [Agr88, Che89, Fuc91, Goe81, Goo91, Han89, Has89, Jar89, Jon92, Lie91, Mel92, Muri93, Rob90]. Finalmente, considera-se que poderá existir alguma informação de entrada, não identificada em concreto neste ponto, resultante do desenvolvimento específico da infraestrutura para o teste e depuração, do controlador residente, ou ainda definida pelo utilizador.

4.5 Conclusão

Neste capítulo introduziu-se a apresentação global do trabalho realizado, e particularmente dos requisitos a que obedeceu, que vai agora ser descrito em detalhe ao longo dos três próximos capítulos: a proposta de uma infraestrutura para o teste e depuração (capítulo 5), a arquitectura de um controlador residente (capítulo 6), e a metodologia e geração automática do programa de teste e depuração (capítulo 7).

Capítulo 5

Proposta de uma infraestrutura para o teste e depuração

No capítulo anterior foi estabelecido um conjunto de requisitos a observar no projecto para o teste e depuração, no qual se incluiu a necessidade de dotar as infraestruturas de teste normalizadas de capacidades acrescidas de apoio à depuração. Este assunto é agora abordado, sendo proposta uma infraestrutura para o teste e depuração, com base nas normas IEEE 1149.1 e P1149.4, para circuitos digitais e mistos, respectivamente. A proposta apresentada visa suprir as principais lacunas identificadas na análise efectuada ao longo do capítulo 3, nomeadamente no que se refere a:

- Implementação em tempo real de pontos de paragem por condição.
- Operações de amostragem em tempo real.
- Detecção de atrasos em ligações.

Nas próximas secções descreve-se o conjunto de instruções opcionais que são propostas para colmatar estas lacunas. Para cada uma delas apresenta-se uma descrição sumária que inclui o respectivo objectivo, o procedimento / sequência de utilização, as modificações necessárias em termos de estrutura de registos / células de teste e as implicações ao nível das equações lógicas dos seus sinais de controlo, apontando-se no final um conjunto de questões inerentes ao modo de implementação. O capítulo termina com uma análise qualitativa das vantagens e desvantagens associadas aos modos de funcionamento opcionais propostos.

5.1 Detecção em tempo real de pontos de paragem por condição

A implementação de pontos de paragem por condição em tempo real engloba três fases distintas: a especificação da condição; a sua avaliação / detecção, em sincronismo com o funciona-

mento normal do circuito sob depuração e; a paragem deste último após a entrada num estado que torne verdadeira a condição especificada.

Num circuito digital, as condições a detectar traduzem-se em combinações de valores discretos ('0' ou '1') presentes nos pinos funcionais (i.e. não utilizados para funções de teste ou de alimentação) dos CI que disponham de uma infraestrutura de teste compatível com a proposta a apresentar. A implementação de condições de paragem é parcialmente assegurada por duas novas instruções opcionais: uma que possibilita a especificação do tipo de condição a detectar (a primeira das três fases referida no parágrafo anterior) e uma outra que coloca o registo BS num modo de detecção da condição especificada (segunda fase). O resultado da avaliação é disponibilizado para o exterior através de um pino adicional que se designa a partir daqui por *Saída de Condição Detectada* (SCD). Esta saída é ligada ao controlador residente de teste e depuração, que garante o sincronismo entre o momento da sua amostragem e o funcionamento do circuito sob depuração. O controlador residente é ainda responsável pela paragem do circuito, após a detecção de uma condição verdadeira (terceira fase).

O tipo de comparação a efectuar é especificado através do registo de dados *Selecção de Tipo de Condição* (STC) seleccionado pela instrução opcional *Selecciona Condição* (*SELCOND*, na forma abreviada). O número de condições possíveis é virtualmente ilimitado (considerando sequências de condições ou uma única condição), pelo que em última análise são os recursos necessários para a implementação dos mecanismos de detecção que impõem o limite para o número máximo de condições. No presente caso optou-se por implementar um conjunto de condições limitado à especificação de dois vectores, correspondentes a um vector esperado e a uma máscara de comparação, ou a dois limites, que garantissem a satisfação dos requisitos mais frequentes deste tipo, nomeadamente para o caso de sistemas baseados em microprocessadores, onde a utilização de técnicas de mapeamento em espaço de memória (ou de E/S) origina várias vezes a necessidade de se determinar qual o valor presente no barramento de endereços (ou dados), aquando de determinado evento, ou vice-versa. Repare-se ainda que estes dois vectores correspondem à capacidade máxima de armazenamento do registo BS, um vector no A_D e outro no A_R[42]. Uma vez seleccionado o tipo de comparação a efectuar, a instrução opcional *Activa Comparação* (*COMP*, na forma abreviada) permitirá iniciar a avaliação / detecção pretendida. A lista seguinte apresenta as oito condições seleccionadas, que requerem um registo adicional de dados com três ($\log_2 8$) bits.

[42] Considerando que todas as células BS são idênticas à exemplificada em [IEEE93, pág. 1-4], onde existem dois elementos de memória, o A_D e o A_R.

- Igual, com comparação através de uma máscara (vector actual[43] = vector esperado).
- Diferente, com comparação através de uma máscara (vector actual ≠ vector esperado).
- Maior que limite A (vector actual > limite A).
- Menor que limite A (vector actual < limite A).
- Maior ou igual que limite A (vector actual ≥ limite A).
- Menor ou igual que limite A (vector actual ≤ limite A).
- Situado no intervalo]A, B[(limite A < vector actual < limite B).
- Situado fora do intervalo [A, B] (vector actual < limite A **ou** vector actual > limite B).

O vector esperado / limite A é armazenado no A_R das células BS e a máscara / limite B é armazenado no A_D das células BS. Dado que as operações de comparação com limites só fazem sentido com números inteiros e ordenados, o conjunto de células que entram na determinação do resultado da comparação devem estar agrupadas por uma determinada ordem. Por exemplo, para um conjunto de oito saídas de um CI com BST o vector poderá ser definido como o conteúdo ordenado da saída 7 até à saída 0 (S[7..0], em que S7 representa o bit mais significativo e S0 representa o bit menos significativo). Nas operações de comparação com limites não são utilizadas máscaras de comparação.

Procedimento / sequência de utilização

O processo que decorre desde a especificação de uma condição até à posterior sinalização para o exterior da sua detecção engloba um conjunto de acções que são agora apresentadas, por ordem cronológica de execução:

- Deslocar a instrução *SELCOND* para o registo de instrução, por forma a colocar no percurso TDI-TDO o registo STC.
- Deslocar para o registo STC a combinação que selecciona o tipo de comparação pretendida. Atribui-se o código ‘000’ à primeira condição apresentada, ‘001’ à segunda e assim por diante, até se atingir a oitava e última condição apresentada, à qual se atribui o código ‘111’.
- Carregar a instrução *SAMPLE / PRELOAD* e deslocar para o registo BS o vector esperado / limite A. Ao passar no estado *Update-DR,* o vector ficará armazenado no A_R do registo BS.
- Deslocar a instrução opcional *COMP*, que activa a comparação com o vector presente nos pinos de entrada (ou nas saídas da lógica funcional) do CI. O registo BS é colocado no percurso TDI - TDO, por forma a permitir o deslocamento da máscara de comparação / limite B.

[43] Corresponde ao conjunto de valores presentes, num dado momento, nas entradas paralelas das células BS associadas aos pinos funcionais do CI.

- Colocar o controlador do TAP no estado *Run-Test/Idle*. Na passagem pelo estado *Update-DR*, o A_R não captura o valor presente no A_D (por especificação da instrução opcional *COMP*).
- O pino SCD, activo ao nível alto ('1'), passa a exibir o resultado da comparação, enquanto o controlador do TAP se encontrar no estado *Run-Test/Idle* (nos outros estados o valor presente neste pino será '0').
- O pino SCD encontra-se ligado ao controlador de teste e depuração, que deverá interromper o fornecimento de impulsos de relógio funcional, imediatamente após a detecção de uma condição de paragem, por forma a reter o sistema no seu estado actual.

A aplicação de condições que envolvam mais do que um CI implica a existência de mais um pino adicional em cada CI, denominado *Entrada de Condição Detectada* (ECD), para permitir a sua concatenação seguindo o esquema ilustrado na Figura 5-1. Apesar de versátil, este esquema é penalizado pelo somatório dos tempos de propagação no interior de cada um dos CI neste conjunto, que não poderá ultrapassar o período do relógio do sistema, sob pena de se aplicar um impulso de relógio extra, após a ocorrência do estado que torna verdadeira a condição especificada.

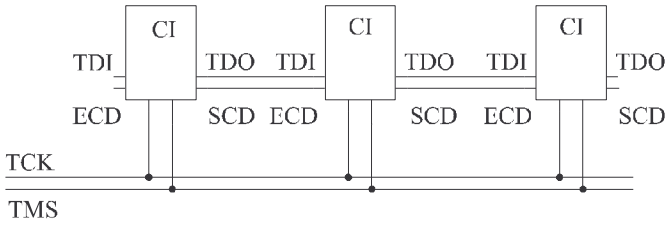


Figura 5-1: Expansão da condição a vários CI, através da concatenação dos pinos ECD e SCD.

Modificações na estrutura de registos e das células BS

A infraestrutura de teste do CI passa a incluir o registo STC, de acordo com o esquema ilustrado na Figura 5-2. O conjunto de acções descritas anteriormente determina as modificações a introduzir nas células BS, por forma a suportarem a funcionalidade pretendida para a instrução *COMP*, conduzindo à estrutura que se apresenta na Figura 5-3.

Uma primeira hipótese para avaliar a condição através do registo BS seria congregar a função de geração do sinal SCD num único bloco de lógica combinatória, alimentado por todas as entradas paralelas do registo BS e pelas saídas do A_D (máscara / limite B), A_R (vector esperado / limite A), registo STC (tipo de condição), controlador do TAP (estado *Run-Test/Idle*) e registo de instrução (instrução *COMP* activa). O elevado número de sinais (que facilmente pode

atingir as várias dezenas ou mesmo centenas) constitui uma importante limitação desta hipótese inicial de implementação. O esquema apresentado na Figura 5-3, sugere uma estratégia de "dividir para simplificar", sendo em cada célula BS efectuada a avaliação parcial da condição, através do bloco de lógica combinatória designado por F_n, que possui um número mais reduzido de entradas (conteúdo do registo STC, entrada paralela da célula BS, saída do A_D, saída do A_R e resultado de F_{n-1}). Esta forma de implementação possui como principais vantagens a independência em relação ao comprimento do registo BS e a redução do número de entradas de cada bloco F_n (menor complexidade), embora implique que o tempo de avaliação da condição passe a depender do número de blocos existentes (menor velocidade).

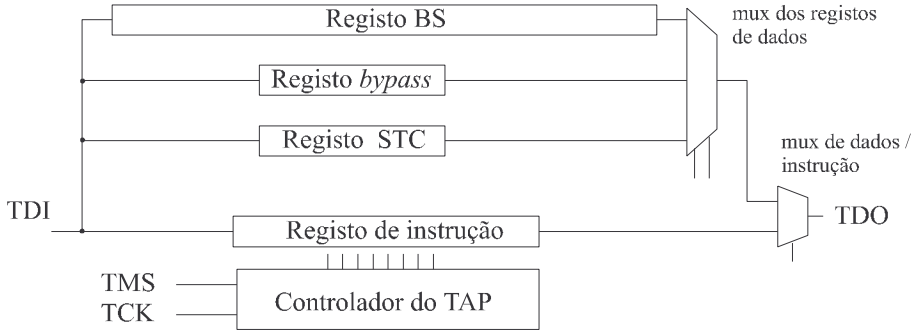


Figura 5-2: Estrutura de registos da infraestrutura de teste, após a implementação da instrução *SELCOND*.

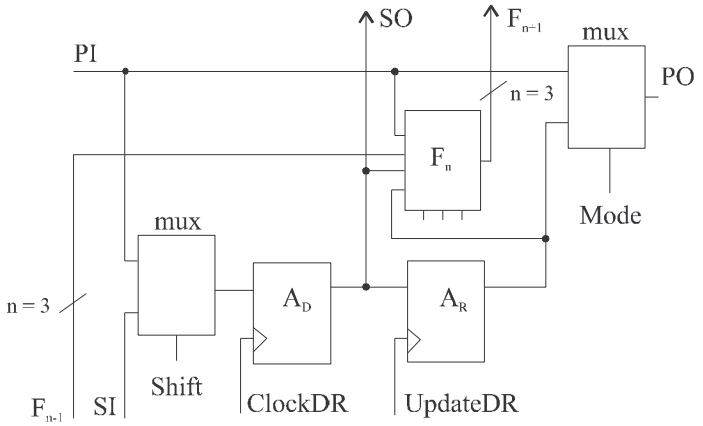


Figura 5-3: Estrutura da célula BS modificada para suportar a instrução opcional *COMP*.

A solução proposta não invalida porém outras opções de implementação, que poderão ir desde o extremo oposto (geração do sinal SCD através de um único bloco de lógica combinatória), com o aumento inerente da velocidade à custa de uma maior complexidade, até soluções intermédias, com a implementação de blocos semelhantes a F_n de *m* em *m* células BS.

Alterações nos sinais de controlo das células BS

Tomando como ponto de referência as equações lógicas sugeridas na norma IEEE 1149.1 para o caso de um CI que suporte as três instruções obrigatórias [IEEE93, capítulo 5], demonstra-se que apenas o sinal UpdateDR[44] sofre uma ligeira alteração, mantendo-se a geração dos sinais Mode, Shift e ClockDR. A equação lógica de UpdateDR passa a incluir um novo sinal de entrada, que impede que este sinal seja activado quando a instrução *COMP* se encontra carregada no registo de instrução, conforme se descreve na Tabela 5-1[45].

Tabela 5-1: Modificações nas equações lógicas dos sinais de controlo das células BS, para implementar as instruções opcionais *SELCOND* e *COMP*.

Sinal	Infraestrutura obrigatória	Suporte das instruções *SELCOND* e *COMP*
Shift	*Shift-DR* (registado)	Mantém-se
ClockDR	TCK·(*Shift-DR* + *Capture-DR*)	Mantém-se
UpdateDR	!TCK·*Update-DR*	!TCK·*Update-DR*·!*COMP*[46]
Mode	*EXTEST*	Mantém-se

Geração do sinal SCD

Em relação ao bloco F_n, a função lógica implementada é definida pelas seguintes tabelas, seleccionadas através do conteúdo do registo STC. A Tabela 5-2a) refere-se à detecção da igualdade entre o vector presente nas entradas e um vector esperado, com comparação através de uma máscara. A Tabela 5-2b) refere-se à detecção da desigualdade (*bit* a *bit*) entre o vector presente nas entradas e um vector esperado, com comparação através de uma máscara. A Tabela 5-2c) e a Tabela 5-2d) referem-se à detecção do posicionamento do valor determinado pelo vector presente nas entradas, em relação a um limite A. Em c) detectam-se as situações "vector > limite A" e "vector ≥ limite A" (a distinção entre estes dois casos é efectuada posteriormente) e em d) detectam-se as situações "vector < limite A" e "vector ≤ limite A" (a distinção entre estes dois casos é efectuada posteriormente). A Tabela 5-2e) e a Tabela 5-2f) refe-

---

[44] Opta-se por manter o estilo seguido em [IEEE93] onde estados do controlador do TAP são referidos em *itálico*, nomes de instruções são em maiúsculas e *itálico* e os nomes de sinais são apenas iniciados por uma maiúscula.

[45] Nas equações lógicas utilizam-se os seguintes símbolos para os operadores lógicos NOT, AND e OR:

!	NOT	.	AND	+	OR

[46] !*COMP* representa um sinal que estará *activo* quando a instrução actual *não for COMP*.

rem-se igualmente à detecção do posicionamento do valor determinado pelo vector presente nas entradas, em relação a dois limites A e B. Em e) detecta-se a situação "limite A < vector < limite B" e em d) detecta-se a situação "vector < limite A **ou** limite B < vector".

As Tabelas 5-2a) a 5-2f) codificam a condição propagada através do bloco F_n (o que corresponde à célula BS de ordem *n*) através de cinco símbolos, com o seguinte significado:

- "V" (verdadeiro): significa que a condição que se pretende detectar, qualquer que ela seja, não foi ainda provada falsa, no que diz respeito ao conjunto de células que precedem a célula considerada (de *n-1* até 0[47]). Propagar-se-á como "V" para a saída do bloco F_n, desde que não passe a falsa nesta célula.
- "F" (falso): significa que a condição que se pretende detectar, qualquer que ela seja, já foi provada falsa (e será falsa quaisquer que sejam os valores presentes nas células que sucedem a esta, pelo que se deve propagar como "F" independentemente daqueles valores).
- "=" (igualdade): significa que o vector presente nas entradas das células e os valores esperados (armazenados no A_R) são iguais em todos as células que precedem a célula considerada (de 0 até *n-1*). Esta igualdade propagar-se-á para a saída do bloco F_n, desde que se mantenha nesta célula.
- ">A" (maior que A): significa que o valor representado pelo vector presente nas entradas das células é maior que o valor representado pelo limite A (definido pelos valores lógicos armazenados no A_R), no que diz respeito ao conjunto de células que precedem a actual. Este valor propagar-se-á para a saída do bloco F_n desde que a condição que se pretende detectar (neste caso poderá ser "vector $\in$]A, B[" ou "vector $\notin$ [A, B]") não seja provada falsa ou verdadeira nesta célula.
- "<B" (menor que B): significa que o valor representado pelo vector presente nas entradas das células é menor que o valor representado pelo limite B (definido pelos valores lógicos armazenados no A_D), no que diz respeito ao conjunto de células que precedem a actual. Este valor propagar-se-á para a saída do bloco F_n desde que a condição que se pretende detectar (neste caso será uma vez mais "vector $\in$]A, B[" ou "vector $\notin$ [A, B]") não seja provada falsa ou verdadeira nesta célula.

Atendendo a que é preciso codificar estas cinco situações cada bloco F_n requer três[48] linhas de entrada e três linhas de saída, apenas para este efeito. A situação a codificar nas entradas do bloco F_n (o da célula mais próxima de TDI), de acordo com as Tabelas 5-2a) a 5-2f), deverá ser "V" para as condições "vector actual = vector esperado" e "vector actual $\neq$ vector esperado", e

[47] A célula 0 é aquela que, no grupo considerado, está mais próxima de TDO.

[48] [Menor inteiro superior a Log_2 (5)].

"=" para todas as restantes. A situação codificada à saída do bloco F_n (associado à última célula BS, a mais próxima de TDO), em conjunto com o tipo de condição a detectar, determinam o *Valor Lógico da Condição* (VLC), tal com se apresenta na Tabela 5-3. Este sinal VLC propagar-se-á para o pino SCD desde que a instrução actual seja a instrução *COMP* e que o controlador do TAP esteja no estado *Run-Test/Idle*, de acordo com o bloco ilustrado na Figura 5-4.

Tabela 5-2: Tabelas de verdade para F_n.

F_{n-1}	A_D	A_R	PI	F_n
F	X	X	X	F
V	0	X	X	V
V	1	0	0	V
V	1	0	1	F
V	1	1	0	F
V	1	1	1	V

a)

F_{n-1}	A_D	A_R	PI	F_n
F	X	X	X	F
V	0	X	X	V
V	1	0	0	F
V	1	0	1	V
V	1	1	0	V
V	1	1	1	F

b)

F_{n-1}	A_D	A_R	PI	F_n
=	X	0	0	=
=	X	0	1	V
=	X	1	0	F
=	X	1	1	=
F	X	X	X	F
V	X	X	X	V

c)

F_{n-1}	A_D	A_R	PI	F_n
=	X	0	0	=
=	X	0	1	F
=	X	1	0	V
=	X	1	1	=
F	X	X	X	F
V	X	X	X	V

d)

F_{n-1}	A_D	A_R	PI	F_n
=	0	0	0	=
=	0	X	1	F
=	X	1	0	F
=	1	0	0	<B
=	1	0	1	>A
=	1	1	1	=
<B	X	0	0	<B
<B	X	0	1	V
<B	X	1	0	F
<B	X	1	1	<B
>A	0	X	0	>A
>A	0	X	1	F
>A	1	X	0	V
>A	1	X	1	>A
F	X	X	X	F
V	X	X	X	V

e)

F_{n-1}	A_D	A_R	PI	F_n
=	0	0	0	=
=	0	X	1	V
=	X	1	0	V
=	1	0	0	<B
=	1	0	1	>A
=	1	1	1	=
<B	X	0	0	<B
<B	X	0	1	F
<B	X	1	0	V
<B	X	1	1	<B
>A	0	X	0	>A
>A	0	X	1	V
>A	1	X	0	F
>A	1	X	1	>A
F	X	X	X	F
V	X	X	X	V

f)

Tabela 5-3: Geração do valor lógico da condição, a partir da situação codificada na saída do bloco F_n associado à última célula BS (a mais próxima de TDO).

F_n	Tipo de condição	VLC
F, V	Todas	0 se F_n = F, 1 se F_n = V
=	vector > A ou vector < A	0
=	vector ≥ A ou vector ≤ A	1
=, >A, <B	vector ∈]A, B[ou vector ∉ [A, B]	0

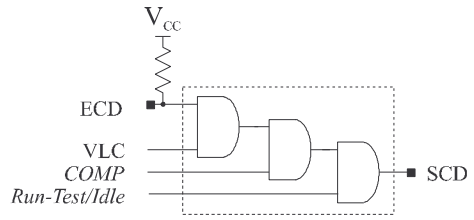


Figura 5-4: Bloco de geração do sinal presente no pino SCD.

Exemplo de aplicação

Apresenta-se de seguida um pequeno exemplo de aplicação do modo de funcionamento opcional proposto, com base num CI que se assume possuir oito pinos de entrada, oito pinos de saída e duas entradas de controlo com a seguinte funcionalidade:

- Quando as entradas de controlo se encontram activas ao nível lógico baixo (‘0’), as saídas exibem os valores presentes nas entradas.
- Se assim não for, as saídas exibem um estado de alta-impedância.

A Figura 5-5 ilustra os pinos e células do registo BS deste CI, representando ainda o TAP e os pinos adicionais ECD e SCD. Pretende-se detectar a situação em que o vector presente nas entradas do CI se situa entre dois limites, um inferior de valor igual a 64 (limite A) e um superior igual a 128 (limite B). Esta situação corresponde a uma condição do tipo “limite A < vector < limite B”, pelo que é necessário deslocar, após se ter carregado previamente a instrução *SELCOND*, o vector ‘110’ para o registo STC. É então necessário deslocar o limite A, após se ter carregado previamente a instrução *SAMPLE / PRELOAD*, que ficará guardado no A_R do registo BS no final do estado *Update-DR*, tal como se ilustra na Figura 5-6a). A acção seguinte consiste em deslocar o limite B, após se ter carregado previamente a instrução *COMP*, que ficará guardado no A_D do registo BS (que não sofre qualquer alteração na passagem pelo estado *Update-DR*), tal como se ilustra na Figura 5-6b). Finalmente, coloca-se o controlador do TAP no estado *Run-Test/Idle*, passando o pino SCD a exibir o resultado da comparação. Na Figura 5-6c), Figura 5-6d) e Figura 5-6e), apresentam-se os valores gerados no bloco F_n de cada célula BS quando o vector presente nas entradas do CI corresponde ao valor dez, duzentos e cem, respectivamente.

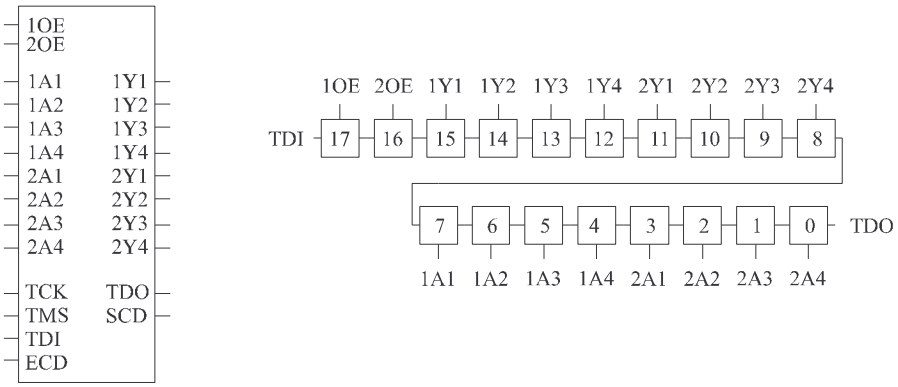


Figura 5-5: Descrição dos pinos e estrutura do registo BS do CI utilizado no exemplo de aplicação.

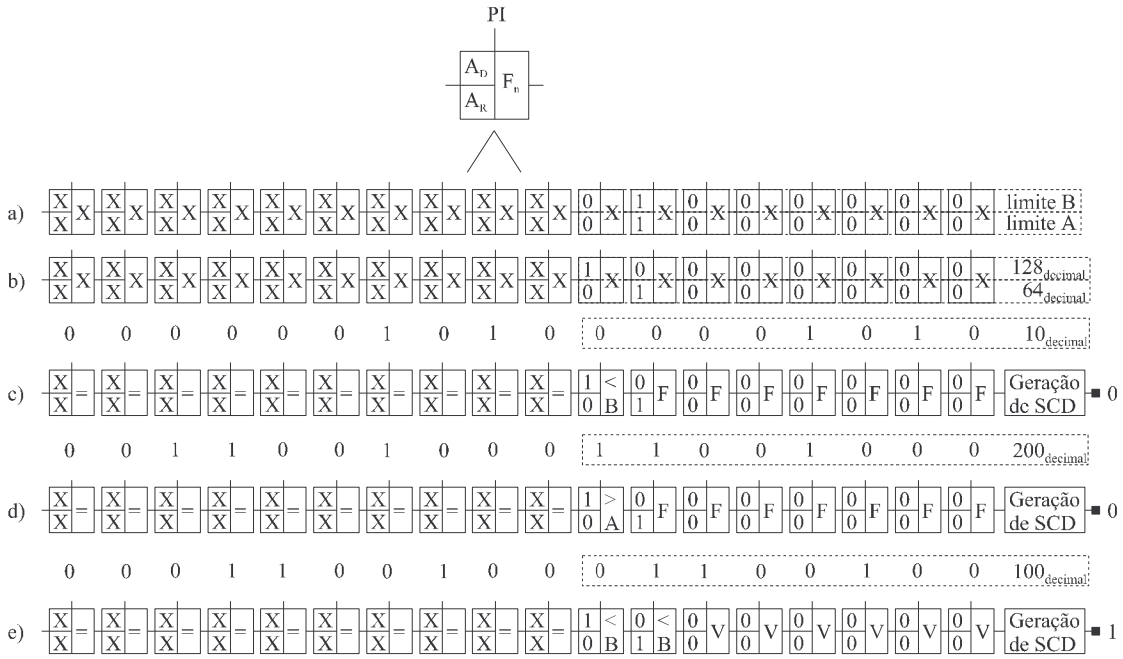


Figura 5-6: Valores presentes ao longo do tempo, no registo BS e no pino SCD, para o exemplo de aplicação de um ponto de paragem pela condição 64 < vector actual < 128.

Questões a observar na implementação

Do anteriormente exposto decorrem naturalmente algumas questões de pormenor, referentes ao modo de implementação, nomeadamente:

- O registo STC não necessita de possuir um andar de retenção, dado que enquanto se efectua a operação de deslocamento, os valores intermédios não são susceptíveis de produzir qualquer efeito adverso. O sinal de relógio deste simples registo de deslocamento (sem entradas paralelas) é gerado pela saída de um AND entre o sinal ClockDR e um sinal activo quando a instrução *SELCOND* estiver carregada no registo de instrução.
- O multiplexador dos registos de dados, na infraestrutura BST (ver Figura 5-2), deverá ter uma nova entrada de dados, ligada à saída da última célula de deslocamento do registo STC, e uma nova combinação de controlo válida, activa quando a instrução *SELCOND* estiver carregada no registo de instrução.
- Manter o valor presente no A_R, na passagem por *Update-DR,* com a instrução *COMP* activa, não constitui uma violação da norma [IEEE93, pág. 5-5][49], dado não ser obrigatório que o A_R capture o valor presente no A_D, para todas as instruções.

[49] *"... Some test data registers may be provided with a latched parallel output to prevent changes at the parallel output while data is shifted in the associated shift-register path in response to certain instructions (e.g.,* EXTEST*,* INTEST *and* RUNBIST*)."*

5.2 Amostragem de valores em tempo real

A amostragem de valores em tempo real constitui uma das técnicas mais utilizadas na depuração de problemas que se manifestam unicamente quando o circuito se encontra a funcionar ininterruptamente e à frequência normal de operação. Das questões que se colocam à implementação desta técnica, tendo presente os vários conceitos referidos aquando da apresentação dos analisadores lógicos (a ferramenta tradicionalmente utilizada para esta técnica de depuração), destacam-se a capacidade de amostragem, quer em quantidade (número de canais e número de amostras por canal) quer em velocidade, o sincronismo entre o momento de amostragem e o relógio do circuito sob depuração, e a especificação dos momentos de início / fim de amostragem, com a possibilidade de filtragem dos dados capturados.

As instruções opcionais a seguir apresentadas têm como denominador comum a utilização do registo BS para guardar as amostras efectuadas. A capacidade de armazenamento de uma célula limita a capacidade do registo BS a duas amostras contíguas dos valores presentes em todos os pinos funcionais do CI (uma em A_D e outra em A_R). A extensão do número de amostras é possível à custa de mais andares de retenção, pelo que a captura e armazenamento de uma sequência com n ($n > 2$) amostras implica a adição a cada célula BS de $n-2$ A_R. Uma outra alternativa, explorada na proposta apresentada, consiste em aumentar a capacidade de armazenamento de amostras à custa da redução do número de canais a amostrar. A velocidade de amostragem é limitada, em princípio, à velocidade máxima de funcionamento da lógica de teste, geralmente especificada pelo fabricante do CI como a frequência máxima admissível para TCK. Com base no esquema de amostragem / retenção proposto, é teoricamente possível, como se demonstra mais adiante, aumentar ligeiramente a frequência do TCK durante esta operação, aumentando-se assim a velocidade de amostragem. O sincronismo entre o momento de amostragem, regulado por TCK, e o relógio do circuito sob depuração é da responsabilidade do controlador residente de teste e depuração. A especificação dos momentos de início / fim de amostragem constitui o ponto de distinção das três primeiras instruções opcionais a seguir apresentadas, sendo que a primeira permite capturar / armazenar sequências de dois vectores contíguos sem qualquer condição, a segunda até uma condição detectada no pino adicional ECD e a terceira após uma condição detectada no próprio registo BS ou no pino adicional ECD. A quarta instrução opcional permite a captura / armazenamento de n^{50} amostras contínuas do valor presente num único pino funcional.

Para CI mistos que disponham de uma infraestrutura P1149.4, coloca-se a questão da amostragem de valores nos pinos analógicos, sujeita a parâmetros distintos da amostragem de

[50] n representa o número de células do registo BS.

valores em pinos digitais. A capacidade de armazenamento requerida é geralmente maior, dependendo do número de amostras significativas, quer em extensão (número de bits necessários para codificar o valor analógico) quer em comprimento (intervalo de tempo amostrado). Por outro lado, considera-se ainda que a velocidade de amostragem depende da frequência do sinal, variando de pino para pino, que a questão do sincronismo é radicalmente diferente em comparação com os circuitos digitais e que as condições de início / fim de amostragem passam a depender tanto do domínio analógico como do domínio digital. Sendo difícil endereçar todas estas questões neste momento, remete-se a sua análise para a subsecção onde se apresenta uma instrução opcional que permite a conversão *sigma-delta* (ΣΔ) do valor presente num dado pino analógico e o armazenamento de *n* amostras contíguas no registo BS.

5.2.1 Capturar sequências de dois vectores contíguos

O objectivo deste modo de funcionamento opcional consiste em capturar e armazenar em tempo real uma sequência de dois vectores contíguos no registo BS. Este modo de funcionamento é activado através de uma instrução opcional, designada por *Memoriza Sequência* (*MSEQ*, na forma abreviada).

Procedimento / sequência de utilização

O processo de memorização de sequências de dois vectores contíguos engloba as acções que se passam a descrever:

- Deslocar a instrução opcional *MSEQ* para o registo de instrução.
- Colocar o controlador do TAP no estado *Run-Test/Idle*:
  - Por cada ciclo do TCK, o A_R captura o valor presente no A_D, que por sua vez captura o valor presente na entrada paralela, conforme se ilustra na Figura 5-7. Este procedimento mantém-se enquanto o controlador do TAP continuar no presente estado.
- Para deslocar a sequência dos dois últimos vectores capturados / armazenados no registo BS efectuam-se os seguintes passos:
  - Colocar o controlador do TAP no estado *Shift-DR*.
  - Aplicar um número de impulsos do TCK que permita deslocar o conteúdo do registo BS, que corresponde ao último vector capturado / armazenado.
  - Colocar o controlador do TAP no estado *Exit2-DR,* via *Pause-DR*. Ao passar no estado *Exit2-DR* o A_D captura o valor presente no A_R.
  - Colocar novamente o controlador do TAP no estado *Shift-DR*.
  - Aplicar um número de impulsos do TCK que permita deslocar o conteúdo do registo BS, que corresponde agora ao penúltimo vector capturado / armazenado.

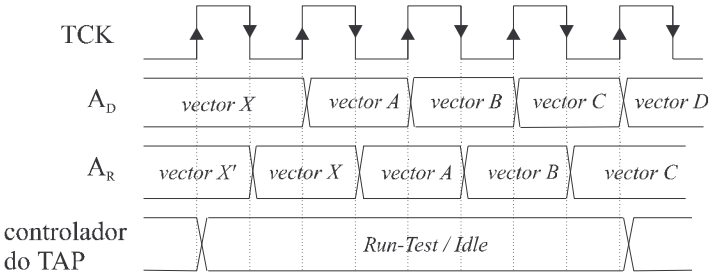


Figura 5-7: Diagrama temporal da memorização de dois vectores contíguos no registo BS.

Modificações na estrutura das células BS

O conjunto de acções descritas anteriormente determina as modificações a introduzir na estrutura das células BS por forma a suportarem a funcionalidade pretendida para a instrução *MSEQ*. Opta-se aqui por apresentar de imediato, na Figura 5-8, a estrutura modificada de uma célula BS e em seguida descrever o porquê das modificações introduzidas.

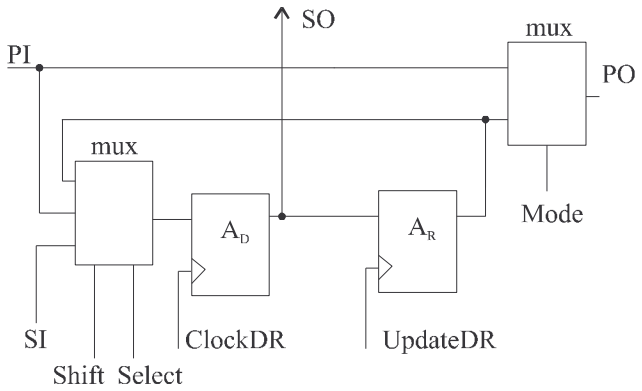


Figura 5-8: Estrutura da célula BS modificada para suportar a instrução opcional *MSEQ*.

A funcionalidade associada à instrução *MSEQ* implica que o A_R seja capaz de capturar o valor presente no A_D e vice-versa, i.e. o A_D seja capaz de capturar o valor presente no A_R. O primeiro requisito corresponde à funcionalidade já existente na célula BS definida pela norma IEEE 1149.1. A alteração a efectuar situa-se apenas ao nível da equação lógica do sinal UpdateDR, uma vez que se pretende que essa acção passe a existir quando o controlador do TAP esteja no estado *Run-Test/Idle*, com a instrução *MSEQ* activa. O segundo requisito resulta da necessidade se deslocar para o exterior, através do A_D, o vector armazenado no A_R. Neste caso é necessário acrescentar mais uma entrada de dados para o multiplexador que alimenta o A_D (que por sua vez implica mais uma entrada de controlo devido a existirem agora três entradas

de dados – entrada paralela, saída do A_D da célula BS anterior e saída do A_R). Em termos de estrutura da célula BS, não é necessário mais qualquer alteração.

Alterações nos sinais de controlo das células BS

Tomando novamente como ponto de referência as equações lógicas sugeridas na norma IEEE 1149.1 para o caso de um CI que suporte as três instruções obrigatórias, apresentam-se na Tabela 5-4 as novas equações lógicas para os sinais ClockDR e UpdateDR, e a equação lógica do novo sinal Select. As equações lógicas dos sinais Mode e Shift permanecem as mesmas. As novas equações lógicas implementam o modo de funcionamento pretendido para a instrução opcional *MSEQ*, mantendo a funcionalidade anterior das três instruções obrigatórias.

Tabela 5-4: Modificações nas equações lógicas dos sinais de controlo das células BS, para implementar a instrução opcional *MSEQ*.

Sinal	Infraestrutura obrigatória	Suporte da instrução *MSEQ*
UpdateDR	!TCK·*Update-DR*	!TCK·(*Update-DR* + *Run-Test/Idle*·*MSEQ*)
ClockDR	TCK·(*Shift-DR* + *Capture-DR*)	TCK·[*Shift-DR* + *Capture-DR*·!*MSEQ* + (*Exit2-DR* + *Run-Test/Idle*)·*MSEQ*][51]
Shift	*Shift-DR* (registado)	Mantém-se
Select	Inexistente	*Exit2-DR*·*MSEQ* Nota: Quando activo selecciona a linha de dados ligada à saída do A_R. Nos restantes casos a selecção da linha de dados é determinada pela linha de controlo Shift.
Mode	*EXTEST*	Mantém-se

Questões a observar na implementação

Do anteriormente exposto decorrem naturalmente algumas questões de pormenor, referentes ao modo de implementação, nomeadamente:

- Modificar os valores presentes no A_D e no A_R, durante *Run-Test/Idle,* com a instrução *MSEQ* activa, é compatível com a norma IEEE 1149.1 [IEEE93, pág. 5-3 [52]].
- Manter o valor presente no A_D, durante *Capture-DR*, com a instrução *MSEQ* activa, é compatível com a norma IEEE 1149.1 [IEEE93, pág. 5-4 [53]].

---

[51] A especificação apresentada impede a operação de captura à passagem por *Capture-DR*, quando *MSEQ* está activa, para não se perder o último vector capturado (em *Run-Test/Idle*).

[52] *"... In the* Run-Test/Idle *controller state, activity in selected test logic occurs only when certain instructions are present."*

- Modificar o valor presente no A_D, durante *Exit2-DR*, com a instrução *MSEQ* activa, viola a norma IEEE 1149.1 [IEEE93, pág. 5-5 [54]]. No entanto, e no que respeita às instruções obrigatórias, e a eventuais outras instruções opcionais que assim se pretendam definir, esta imposição da norma continua a ser verificada (a modificação de A_D, em *Exit2-DR*, tem *apenas* lugar quando *MSEQ* é a instrução actual).
- Para sequências de *n* vectores ($n > 2$) deve existir em cada célula BS um registo adicional com *n-2* FF, com a seguinte funcionalidade: no estado *Run-Test/Idle*, por cada ciclo de TCK, o *n-ésimo* FF captura o valor presente no *n-ésimo - 1* FF, que por sua vez captura o valor presente no *n-ésimo – 2* FF, e assim por diante até que o primeiro FF do registo adicional captura o valor presente no A_R da célula BS. Para ler a sequência capturada / armazenada procede-se da seguinte forma:
  - Colocar o controlador do TAP no estado *Shift-DR*.
  - Aplicar impulsos em TCK até deslocar todo o conteúdo do registo BS.
  - Colocar o controlador do TAP no estado *Exit2-DR*, via *Pause-DR*. Ao passar em *Exit2-DR* o A_D captura o valor presente no A_R e um contador interno de $\log_2(n)$ bits é incrementado (a partir valor inicial zero). Colocar o controlador do TAP no estado *Shift-DR* e deslocar novamente o conteúdo do registo BS.
  - Colocar o controlador do TAP no estado *Exit2-DR*, via *Pause-DR*. Ao passar em *Exit2-DR* o A_D captura o valor presente no FF do registo adicional apontado pelo valor actual do contador interno. Colocar o controlador do TAP no estado *Shift-DR* e deslocar novamente o conteúdo do registo BS. O contador é incrementado uma unidade.
  - Repetir a operação anterior *n-2* vezes.

5.2.2 Capturar sequências de dois vectores contíguos até condição

O objectivo deste modo de funcionamento opcional consiste em capturar e armazenar em tempo real uma sequência de dois vectores contíguos no registo BS, até à detecção de uma condição verdadeira no pino adicional ECD. Este modo de funcionamento é activado através de uma instrução opcional, designada por *Memoriza Sequência Até Condição* (*MSATC*, na forma abreviada). A captura / armazenamento de vectores no registo BS corresponde à funcionalidade especificada para a instrução opcional *MSEQ*, sendo necessário acrescentar uma simples MEF com dois estados, denominados *monitoriza_condição* e *fim_de_sequência*, para distinguir entre os momentos anterior e posterior à ocorrência de uma condição verdadeira.

---

[53] *"... if capturing is not required for the selected test, then the register retains its previous state unchanged"*

[54] *" ... All test data registers selected by the current instruction retain their previous state unchanged."*

Procedimento / sequência de utilização

O processo de memorização de sequências de dois vectores contíguos até à ocorrência de uma condição engloba as acções que se passam a descrever:

- Deslocar a instrução opcional *MSATC* para o registo de instrução.
- Colocar o controlador do TAP no estado *Run-Test/Idle*:
  - O valor lógico da condição, monitorizado no pino ECD, determina a transição da MEF, conforme se ilustra na Figura 5-9.
  - Com a MEF no estado *monitoriza_condição*, por cada ciclo de TCK, o A_R captura o valor presente no A_D, que por sua vez captura o valor presente na entrada paralela, conforme se ilustra na Figura 5-10. Este procedimento mantém-se enquanto o controlador do TAP e a MEF se mantiverem nos referidos estados.
- Quando a condição for verdadeira (ECD = '1'), na próxima transição descendente de TCK, a MEF passa para o estado *fim_de_sequência*, deixando o A_R de capturar o valor presente no A_D, que por sua vez deixa também de capturar o valor presente na entrada paralela. Enquanto o controlador do TAP e a MEF se mantiverem nos referidos estados, os A_D e A_R não alteram os seus valores actuais.
- Para deslocar para o exterior a sequência armazenada procede-se de forma idêntica à descrita na instrução opcional *MSEQ*.

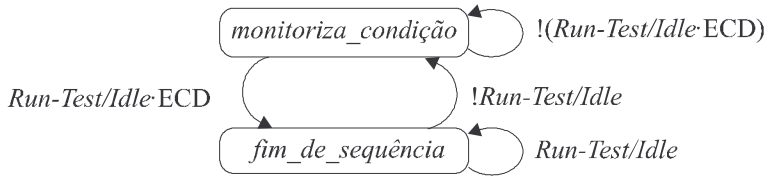


Figura 5-9: Diagrama de estados da MEF para suporte da instrução opcional *MSATC*.

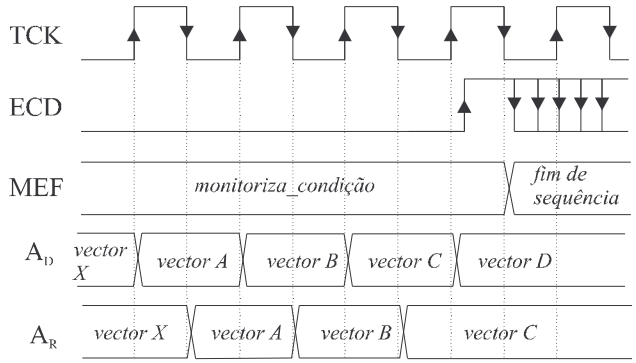


Figura 5-10: Diagrama temporal da memorização de dois vectores contíguos no registo BS, até se verificar uma dada condição no pino ECD.

Modificações na estrutura das células BS

A estrutura da célula BS que suporta a instrução opcional *MSEQ*, apresentada na Figura 5-8, possui todos os recursos necessários para suportar a instrução opcional proposta *MSATC*. As únicas alterações adicionais situam-se ao nível das equações lógicas dos sinais de controlo das células BS, que passam, em certos casos, a depender do estado actual da MEF.

Alterações nos sinais de controlo das células BS

Tomando novamente como ponto de referência as equações lógicas sugeridas na norma IEEE 1149.1, para o caso de um CI que suporte as três instruções obrigatórias, apresentam-se na Tabela 5-5 as novas equações lógicas para os sinais ClockDR e UpdateDR, e a equação lógica do novo sinal Select. As equações lógicas dos sinais Mode e Shift permanecem as mesmas. As novas equações lógicas implementam o modo de funcionamento pretendido para a instrução opcional *MSATC*, mantendo a funcionalidade das três instruções obrigatórias.

Tabela 5-5: Modificações nas equações lógicas dos sinais de controlo das células BS, para implementar a instrução opcional *MSATC*.

Sinal	Infraestrutura obrigatória	Suporte da instrução *MSATC*
UpdateDR	!TCK·*Update-DR*	!TCK·(*Update-DR* + *monitoriza_condição*·*MSATC*)
ClockDR	TCK·(*Shift-DR* + *Capture-DR*)	TCK·[*Shift-DR* + *Capture-DR*·!*MSATC* + (*Exit2-DR* + *monitoriza_condição*)·*MSATC*][55]
Shift	*Shift-DR* (registado)	Mantém-se
Select	Inexistente	*Exit2-DR*·*MSATC* Nota: Quando activo selecciona a linha de dados ligada à saída do A_R. Nos restantes casos a selecção da linha de dados é determinada pela linha de controlo Shift.
Mode	*EXTEST*	Mantém-se

Questões a observar na implementação

Do anteriormente exposto decorrem naturalmente algumas questões de pormenor, referentes ao modo de implementação, nomeadamente:

[55] A especificação apresentada impede a operação de captura à passagem por *Capture-DR*, quando *MSATC* está activa, para não se perder o último vector capturado (em *monitoriza_condição*).

- A MEF é inicializada no estado *monitoriza_condição*. As transições de estado ocorrem na transição descendente de TCK.
- As questões apontadas para a instrução opcional *MSEQ* são igualmente válidas, dado que o modo de funcionamento proposto para a actual instrução opcional *MSATC* constitui essencialmente uma extensão daquele.

5.2.3 Capturar sequências de dois vectores contíguos após condição

O objectivo deste modo de funcionamento opcional consiste em capturar e armazenar em tempo real uma sequência de dois vectores contíguos no registo BS, após a detecção de uma condição verdadeira. Este modo de funcionamento é activado através de uma instrução opcional, designada por *Memoriza Sequência Após Condição* (*MSAPC*, na forma abreviada). A detecção de condições corresponde à funcionalidade especificada na instrução opcional *COMP*[56] e a captura / armazenamento de vectores corresponde à funcionalidade especificada para *MSEQ*. Para implementar este modo de funcionamento é necessário também uma MEF, que poderá basear-se na anteriormente apresentada, com quatro estados denominados *monitoriza_condição*, *captura_sequência_I*, *captura_sequência_II* e *fim_de_sequência*. Repare-se no entanto que existe uma diferença considerável entre a instrução opcional agora proposta e a anterior. *MSATC* captura vectores consecutivos enquanto o pino ECD se mantiver no estado inactivo, e armazena-os no A_D e A_R, que por esta razão não estão disponíveis para conter máscara e valores esperados (trata-se portanto, no caso de *MSATC*, de uma *condição no pino ECD*). Ao contrário, a instrução agora proposta só efectua a captura e armazenamento após a detecção da condição pretendida, pelo que não existem impedimentos relativamente a usar A_D e A_R para conter máscara e valores esperados (trata-se portanto, no caso de *MSAPC*, de uma *condição no conjunto de pinos funcionais* do componente).

Procedimento / sequência de utilização

O processo de memorização de sequências de dois vectores contíguos após a detecção de uma condição engloba as acções que se passam a descrever:

- Deslocar a instrução *SAMPLE / PRELOAD*.
- Deslocar o vector esperado. Ao passar no estado *Update-DR,* o vector ficará armazenado no A_R do registo BS.

---

[56] Por questões de simplificação, optou-se por restringir o tipo de condições à igualdade entre o vector actual e o vector esperado, com comparação através de uma máscara. Desta forma evita-se a necessidade de referir o registo STC e consequentemente a instrução opcional *SELCOND*, na descrição da actual instrução opcional proposta.

- Deslocar a instrução *MSAPC* (por especificação, e à semelhança de *COMP*, esta instrução não permite a actualização dos A_R, aquando da passagem por *Update-DR*).
- Colocar o controlador do TAP no estado *Shift-DR* e deslocar a máscara de comparação, que ficará armazenada no A_D do registo BS.
- Colocar o controlador do TAP no estado *Run-Test/Idle*, após o que o pino SCD (opcional) passa a exibir o resultado da comparação (nos outros estados o valor presente neste pino será '0'). Quando o vector actual for idêntico ao vector esperado, a MEF, cujo diagrama de transição de estados se ilustra na Figura 5-11, passa para o estado *captura_sequência_I* na transição descendente de TCK.
- No estado *captura_sequência_I*, e à transição ascendente de TCK, o A_D captura o vector presente nas entradas paralelas das células BS. Na transição descendente seguinte de TCK, o A_R captura o valor presente no A_D e a MEF passa para o estado *captura_sequência_II*.
- No estado *captura_sequência_II*, o A_D captura o valor presente na entrada paralela da célula BS e o A_R mantém o valor anterior. Na transição descendente seguinte de TCK a MEF passa para o estado *fim_de_sequência.*
- No estado *fim_de_sequência*, um dos bits (à excepção dos dois mais próximos de TDO) do registo de instrução captura o valor lógico '1' quando o controlador do TAP passar pelo estado *Capture-IR*. Esta funcionalidade destina-se a permitir a sinalização para o exterior da ocorrência de uma condição verdadeira, quando não existe o pino adicional (opcional) SCD. No final do processo ilustrado na Figura 5-12, o vector esperado e a máscara terão sido substituídos pelos dois vectores capturados / armazenados.
- Para deslocar a sequência armazenada procede-se de forma idêntica à descrita para a instrução opcional *MSEQ*.

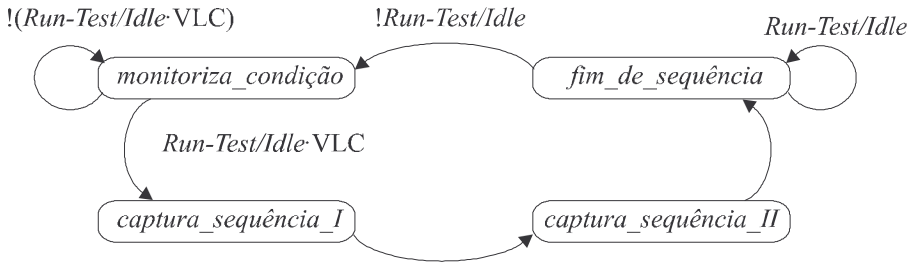


Figura 5-11: Diagrama da MEF para suporte da instrução opcional *MSAPC*.

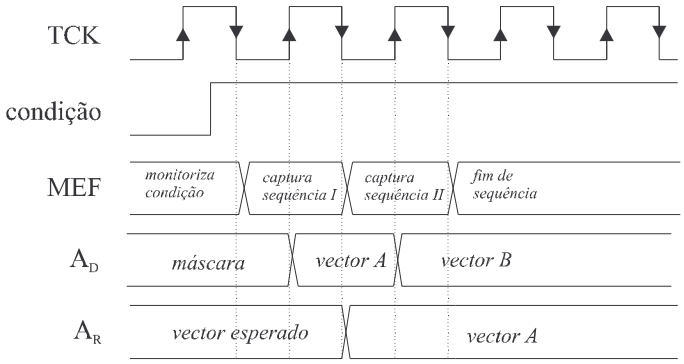


Figura 5-12: Diagrama temporal da memorização de dois vectores contíguos no registo BS, após se verificar uma dada condição no conjunto de pinos funcionais.

Modificações na estrutura das células BS

Dado que a actual instrução opcional proposta se baseia parcialmente na funcionalidade das instruções opcionais *MSEQ* e *COMP*, é com base na associação da estrutura modificada das células BS que suportam estas instruções, apresentadas respectivamente nas Figura 5-3 e Figura 5-8, que se obtém a estrutura da célula BS que suporta a instrução *MSAPC*, ilustrada na Figura 5-13.

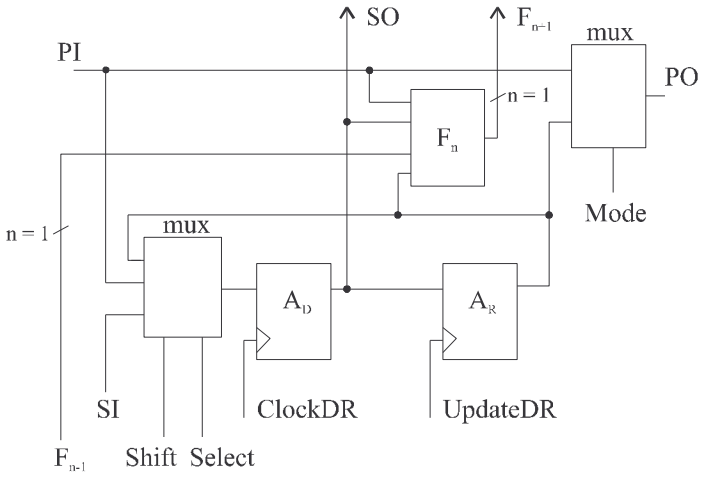


Figura 5-13: Estrutura da célula BS modificada para suportar a instrução opcional *MSAPC*.

Atendendo a que os blocos adicionais apresentados na Figura 5-13, face à estrutura típica de uma célula BS, foram já justificados quando se apresentaram as instruções *COMP* e *MSEQ*, evita-se aqui a sua repetição.

Alterações nos sinais de controlo das células BS

Tomando mais uma vez como ponto de referência as equações lógicas sugeridas na norma IEEE 1149.1, para o caso de um CI que suporte as três instruções obrigatórias, apresentam-se na Tabela 5-6 as novas equações lógicas para os sinais ClockDR e UpdateDR, e a equação lógica do novo sinal Select. As equações lógicas dos sinais Mode e Shift permanecem as mesmas. As novas equações lógicas implementam o modo de funcionamento pretendido para a instrução opcional *MSAPC*, mantendo a funcionalidade anterior das três instruções obrigatórias.

Tabela 5-6: Modificações nas equações lógicas dos sinais de controlo das células BS, para implementar a instrução opcional *MSAPC*.

Sinal	Infraestrutura obrigatória	Suporte da instrução *MSAPC*
UpdateDR	!TCK·*Update-DR*	!TCK·(*Update-DR*·!*MSAPC* + *captura_sequência_I*·*MSAPC*)
ClockDR	TCK·(*Shift-DR* + *Capture-DR*)	TCK·[*Shift-DR* + *Capture-DR*·!*MSAPC* + (*Exit2-DR* + *captura_sequência_I* + *captura_sequência_II*)·*MSAPC*]
Shift	*Shift-DR* (registado)	Mantém-se
Select	Inexistente	*Exit2-DR*·*MSAPC* Nota: Quando activo selecciona a linha de dados ligada à saída do A_R. Nos restantes casos a selecção da linha de dados é determinada pela linha de controlo Shift.
Mode	*EXTEST*	Mantém-se

Questões a observar

Do anteriormente exposto decorrem naturalmente algumas questões de pormenor, referentes ao modo de implementação, nomeadamente:

- A MEF é inicializada no estado *monitoriza_condição*. As transições de estado ocorrem na transição descendente de TCK.
- As questões apontadas para a instrução opcional *MSEQ* são igualmente válidas, dado que o modo de funcionamento proposto para a actual instrução opcional *MSAPC* reutiliza a funcionalidade proposta para aquela instrução.
- Algumas das questões apontadas para a instrução opcional *COMP* são igualmente válidas, dado que o modo de funcionamento proposto para a actual instrução opcional *MSAPC* reutiliza também a funcionalidade proposta para aquela instrução.

5.2.4 Capturar sequências de *n* bits contíguos num só pino

O objectivo deste modo de funcionamento opcional consiste em capturar e armazenar no registo BS, em tempo real, uma sequência de *n* amostras contíguas capturadas num único pino funcional do CI. Este modo de funcionamento é activado através de uma instrução opcional, designada por *Memoriza Sequência num único Pino* (*MSEQ1P*, na forma abreviada). A selecção do pino a amostrar é efectuada através de uma outra instrução opcional, designada por *Selecciona Pino* (*SELPIN*, na forma abreviada), que coloca no percurso TDI - TDO um registo adicional de dados, de comprimento igual ao logaritmo, na base dois, do número de células do registo BS do CI. O vector que identifica a posição da célula BS associada ao pino pretendido é deslocado para este registo adicional, a que é dada a designação de registo de *Selecção de Pino* (SP).

Procedimento / sequência de utilização

O processo de memorização de sequências de *n* bits contíguos, amostrados num único pino, engloba as acções que se passam a descrever:

- Deslocar a instrução *SELPIN* para o registo de instrução, por forma a colocar no percurso TDI - TDO o registo SP.
- Deslocar a combinação que selecciona a célula BS associada ao pino a amostrar. O mapeamento entre o nome / número do pino e a respectiva célula BS encontra-se disponível no ficheiro BSDL do CI, ou na sua folha de características, associando-se geralmente o valor zero à posição da célula BS mais próxima de TDO. O deslocamento através deste registo, para armazenamento dos valores amostrados, é efectuado de forma circular, de modo a permitir o mesmo comprimento de amostra, independentemente da posição (na cadeia BS) do pino seleccionado.
- Deslocar a instrução opcional *MSEQ1P*, que activará a amostragem de valores no pino seleccionado através do registo SP, a partir do momento em que o controlador do TAP seja colocado no estado *Run-Test/Idle*. O registo BS é colocado no percurso TDI - TDO, dado que constitui o recurso utilizado para armazenar as amostras efectuadas, de acordo com o diagrama temporal ilustrado na Figura 5-14.
- Para se ter acesso ao resultado da amostragem, coloca-se o controlador do TAP no estado *Shift-DR* e aplicam-se impulsos em TCK, até se deslocar para o exterior todo o conteúdo do registo BS. A última amostra efectuada (ou seja, a mais recente) encontra-se armazenada na célula BS associada ao pino seleccionado, encontrando-se a amostra mais antiga armazenada na célula que precede a seleccionada (na direcção TDI - TDO). A passagem pelo estado *Capture-DR* não afecta o conteúdo do registo BS, com a instrução *MSEQ1P* activa (por especificação da própria instrução).

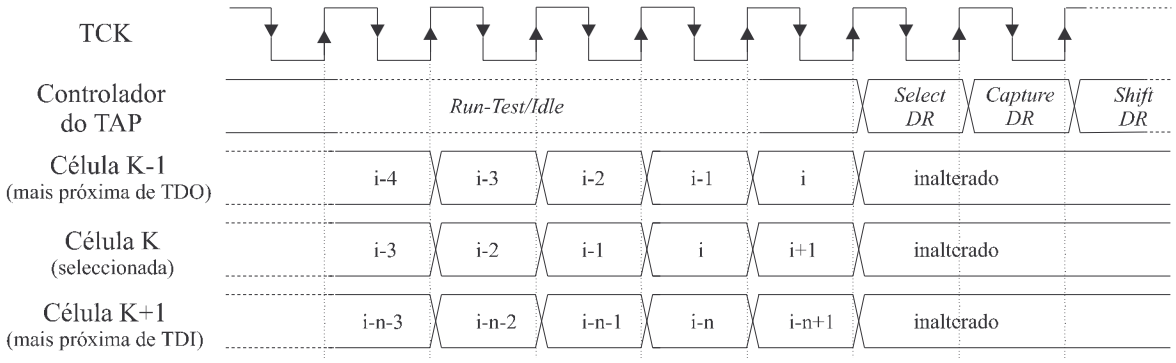


Figura 5-14: Diagrama temporal da memorização de *n* bits contíguos no registo BS.

Modificações na estrutura de registos e das células BS

A estrutura de registos da infraestrutura de teste do CI passa a incluir o registo SP, de acordo com o esquema ilustrado na Figura 5-15.

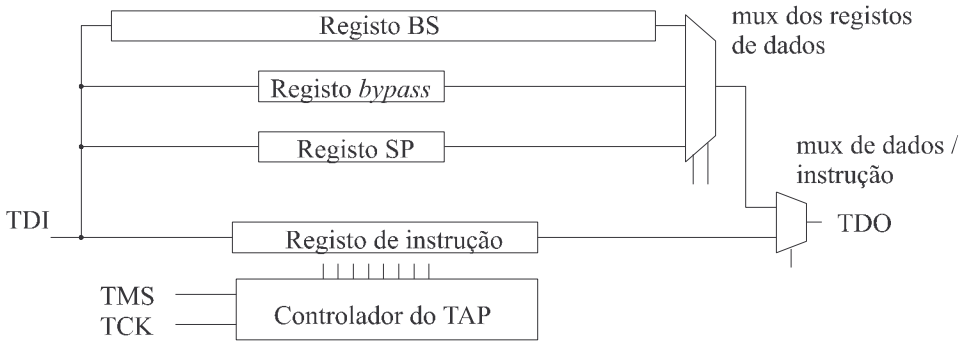


Figura 5-15: Estrutura de registos da infraestrutura de teste, após a implementação da instrução *SELPIN*.

O conjunto de acções descritas anteriormente determina as modificações a introduzir na estrutura das células BS, por forma a suportarem a funcionalidade pretendida para a instrução *MSEQ1P*. Opta-se por apresentar de imediato na Figura 5-16 a estrutura modificada de uma célula BS e em seguida justificar as modificações introduzidas.

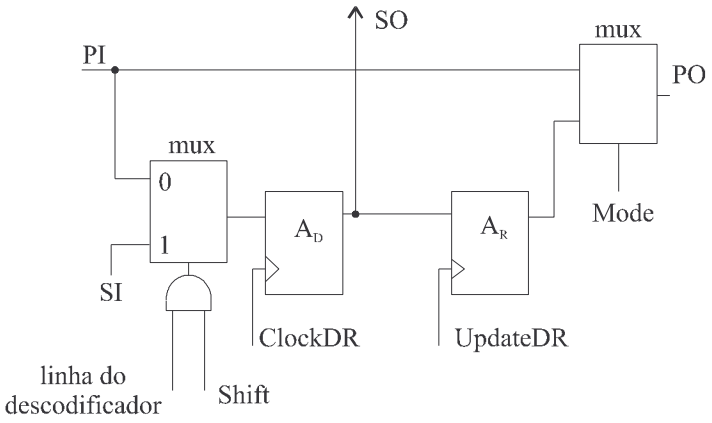


Figura 5-16: Estrutura da célula BS modificada para suportar a instrução opcional *MSEQ1P*.

A única alteração na estrutura das células BS situa-se na linha de controlo do multiplexador que alimenta o A_D. Este passa a ser gerado por um AND entre o sinal Shift e o sinal vindo de um descodificador de m para 2^m cujas linhas de entrada são alimentadas pelo conteúdo do registo SP, de acordo com o ilustrado na Figura 5-17. Quando a instrução *MSEQ1* se encontra activa e o controlador do TAP se encontra no estado *Run-Test/Idle*, a célula BS seleccionada pelo valor contido no registo SP, passa a capturar o valor presente na sua entrada paralela, uma vez que a linha do descodificador é activa ao nível lógico baixo '0' (seleccionando desta forma o andar superior de entrada de dados do referido multiplexador, tal como sucede no estado *Capture-DR*). Quando a instrução *MSEQ1P* não se encontra activa, todas as saídas do descodificador estão inactivas, ou seja, no nível lógico alto (valor neutro para a operação AND), não afectando dessa forma o valor da linha de controlo do multiplexador, que passa a ser determinada exclusivamente pelo sinal Shift.

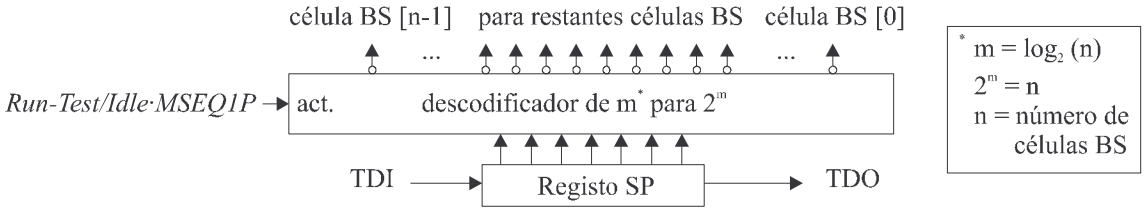


Figura 5-17: Descodificador da célula BS a amostrar.

A tabela de verdade do descodificador, apresentada na Tabela 5-7, permite uma compreensão mais rápida de quais os valores presentes nas saídas do descodificador, que ligam ao multiplexador de entrada do A_D de cada célula BS, de acordo com o conteúdo do registo SP.

Tabela 5-7: Tabela de verdade do descodificador do pino seleccionado.

Act.	Reg. SP	Cél. BS[n–1]	Cél. BS[n–2]	Outras cél. BS	Cél. BS[1]	Cél. BS[0]
0	X	1	1	1	1	1
1	0	1	1	1	1	0
1	1	1	1	1	0	1
1	…	1	1	…	1	1
1	n-2	1	0	1	1	1
1	n-1	0	1	1	1	1

A estrutura da célula BS mais próxima de TDI diverge ligeiramente da que foi apresentada na Figura 5-16, por forma a implementar um registo de armazenamento do tipo circular. A entrada do multiplexador que alimenta o A_D, normalmente ligada a TDI, passa a receber o sinal de saída de um novo multiplexador que selecciona entre TDI e o sinal presente no A_D da célula BS mais próxima de TDO (para a memorização de *n* bits), conforme se ilustra na Figura 5-18. Com esta estrutura é possível aproveitar o comprimento do registo BS na sua totalidade para armazenar as amostras efectuadas na célula BS_i, com $0 \leq i \leq n\text{-}2$. Recorde-se que a célula BS seleccionada pelo registo SP contém a amostra mais recente e a célula BS imediatamente anterior contém a amostra mais antiga. Dado que no modo de funcionamento opcional proposto, o registo BS é convertido num registo de armazenamento circular, quando a célula BS apontada pelo registo SP for a mais perto de TDI, então a imediatamente anterior será a mais próxima de TDO. Quando a instrução *MSEQ1P* não se encontrar activa, a linha do descodificador estará ao nível lógico alto, sendo a entrada de dados do multiplexador que alimenta o A_D, seleccionada de acordo com o valor lógico do sinal Shift. A linha de controlo do multiplexador mais à esquerda da Figura 5-18, estará no nível lógico baixo, sendo neste caso seleccionada a linha de entrada ligada a TDI (andar superior). O novo multiplexador insere porém um atraso adicional, que deverá ser considerado para efeitos de cálculo da frequência máxima admissível para TCK, dado que é geralmente o processo de deslocamento série que impõe este valor.

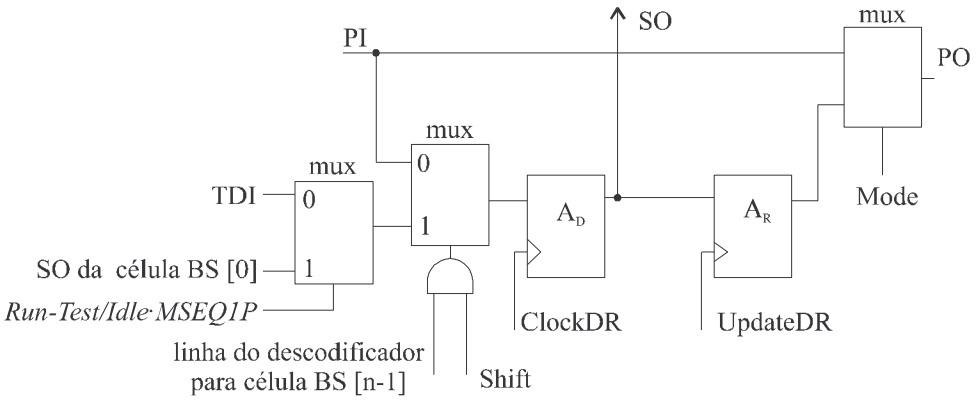


Figura 5-18: Estrutura da célula BS mais próxima de TDI, modificada para suportar a instrução opcional *MSEQ1P*.

Alterações nos sinais de controlo das células BS

Tomando mais uma vez como ponto de referência as equações lógicas sugeridas na norma IEEE 1149.1 para o caso de um CI que suporte as três instruções obrigatórias [IEEE93, capítulo 5], demonstra-se que apenas os sinais ClockDR e Shift sofrem ligeiras alterações, mantendo-se inalterados os sinais Mode e UpdateDR. A equação lógica do sinal ClockDR passa a incluir um novo sinal de entrada, activo quando a instrução *MSEQ1P* for a actual, e o controlador do TAP se encontrar no estado *Run-Test/Idle*. O sinal Shift passa a estar também activo no estado *Run-Test/Idle*, conforme se descreve na Tabela 5-8.

Tabela 5-8: Modificações nas equações lógicas dos sinais de controlo das células BS, para implementar a instrução opcional *MSEQ1P*.

Sinal	Infraestrutura obrigatória	Suporte da instrução *MSEQ1P*
UpdateDR	!TCK·*Update-DR*	Mantém-se
ClockDR	TCK·(*Shift-DR* + *Capture-DR*)	TCK·(*Shift-DR* + *Capture-DR*·!*MSEQ1P* + *Run-Test/Idle*·*MSEQ1P)*
Shift	*Shift-DR* (registado)	*Shift-DR* + *Run-Test/Idle* (registado)
Mode	*EXTEST*	Mantém-se

Questões a observar na implementação

Do anteriormente exposto decorrem naturalmente algumas questões de pormenor, referentes ao modo de implementação. Sendo difícil de determinar uma ordem de apresentação, opta-se por seguir os pontos que constam da descrição do procedimento / sequência de utilização:

- O registo SP não necessita de um andar de retenção, dado que enquanto se efectua a operação de deslocamento, os valores intermédios não são susceptíveis de produzir qualquer efeito adverso. O sinal de relógio deste simples registo de deslocamento (sem entradas paralelas) é gerado por um AND entre o sinal ClockDR e um sinal activo quando a instrução *SELPIN* estiver carregada no registo de instrução.
- O multiplexador dos registos de dados (ver Figura 5-15) deverá ter uma nova entrada de dados, ligada à saída do último andar de deslocamento do registo SP, e uma nova entrada de controlo, activa quando a instrução *SELPIN* estiver carregada no registo de instrução.

5.2.5 Conversão ΣΔ em pinos analógicos e armazenamento de *n* bits contíguos

A técnica de captura / armazenamento apresentada na subsecção anterior pode ser empregue em circuitos mistos que disponham de uma infraestrutura P1149.4. Dado que as estruturas de controlo dos ABM e do TBIC requerem cada uma quatro células, o número total de células do registo BS de um CI misto será maior do que o de um CI digital com o mesmo número de pinos, supondo que o número de pinos bidireccionais existentes em ambos os CI não difere largamente. Esta maior capacidade de armazenamento facilita o modo de funcionamento opcional a seguir proposto, para esta infraestrutura de teste, que consiste na implementação de um conversor do tipo ΣΔ no TBIC, de forma a permitir, através das capacidades disponibilizadas conjuntamente pelo barramento de teste analógico interno e pelos ABM, a conversão A/D do valor presente em qualquer pino analógico e o armazenamento da sequência de bits no registo BS. A velocidade de amostragem pode igualar a velocidade máxima admissível para TCK (em relação ao CI em causa), optando-se por colocar o filtro decimador digital no exterior do CI, dado os recursos necessários para a sua implementação ultrapassarem em larga escala os normalmente requeridos para o modulador de 1ª ordem. Com efeito, o filtro decimador pode ser responsável, num conversor ΣΔ, por mais de 75% da área de silício, e por consumir a maior parte da potência absorvida pelo conversor [Leu91].

A escolha de um conversor ΣΔ de um bit, idêntico ao ilustrado na Figura 5-19, facilita a aplicação quase imediata do modo de funcionamento opcional proposto na subsecção anterior, com a vantagem da implementação externa do filtro decimador, que diminui drasticamente os recursos necessários para a solução proposta, à custa do alargamento do tempo que medeia entre a recolha de amostras e a sua conversão numa palavra digital, devido ao processo intermédio de deslocamento para o exterior do CI dos valores amostrados.

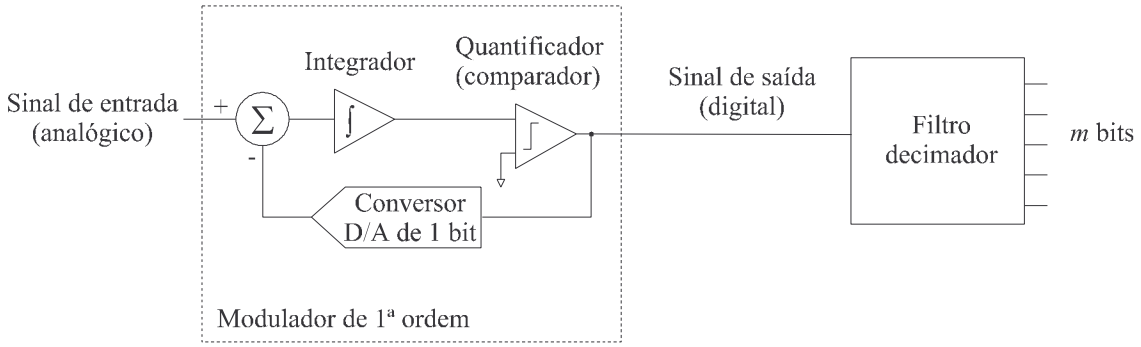


Figura 5-19: Conversor sigma-delta (ΣΔ) de 1ª ordem.

A implementação deste modo de funcionamento é simplificada se a estrutura de controlo do TBIC, mais concretamente a célula CALIBRATE, for a mais próxima de TDI. Desta forma, evita-se a introdução de um descodificador[57] para identificação / endereçamento desta célula, aproveitando-se a totalidade da extensão do registo BS, sem necessidade de multiplexadores adicionais, para armazenar as várias amostras efectuadas.

Procedimento / sequência de utilização

O processo de memorização de sequências de *n* bits contínuos, gerados na saída digital do modulador de 1ª ordem, engloba as acções que se passam a descrever:

- Deslocar a instrução opcional designada por *Conversão Sigma-Delta num único Pino* (*CΣΔ1P*, na forma abreviada).
- Colocar o controlador do TAP no estado *Shift-DR* (a passagem pelo estado *Capture-DR* não altera o conteúdo do registo BS, por especificação de *CΣΔ1P*).
- Deslocar o vector que selecciona as seguintes opções:
  - Ligar o pino analógico seleccionado, para efeitos de conversão ΣΔ, à linha AB2 do barramento de teste interno, através do interruptor SB2. Manter o interruptor SD na posição de fechado e os restantes em aberto, tal como se ilustra na Figura 5-20. Repare-se que a configuração seleccionada é não intrusiva, o que permite manter o funcionamento normal do circuito.
  - Relativamente aos ABM associados aos restantes pinos analógicos, manter o interruptor SD na posição de fechado e os restantes interruptores na posição de aberto.
  - Para todos os DBM – irrelevante.
- Colocar o controlador do TAP no estado *Run-Test/Idle*.
- Enquanto o controlador do TAP se mantiver neste estado e a instrução actual for *CΣΔ1P*, a célula CALIBRATE (inserida na estrutura de controlo do TBIC), captura na transição ascendente de TCK o resultado do modulador de 1ª ordem inserido no TBIC, de acordo com o esquema ilustrado nas Figura 5-21 e Figura 5-22. Todas as restantes células (as restantes deste ABM e as de todos os outros ABM e DBM) encontram-se em modo de deslocamento.
- Para deslocar para o exterior o resultado da amostragem, colocar o controlador do TAP no estado *Shift-DR* (a passagem pelo estado *Capture-DR* não modifica o conteúdo do registo BS) e aplicar o necessário número de impulsos em TCK. O resultado da amostragem deverá passar pelo filtro decimador externo, de forma a obter-se o resultado final no formato pretendido, por exemplo, numa ou várias palavras digitais com vinte bits. O número máxi-

[57] Que, a existir, seria idêntico ao descrito na subsecção anterior e que teria como vantagem principal não condicionar o posicionamento das células do registo BS, conforme sugerido na norma IEEE P1149.4 [IEE98, pág. 59].

mo de palavras depende do comprimento do registo BS, sendo igual a $\lfloor(n/20)\rfloor$[58] para o exemplo referido.

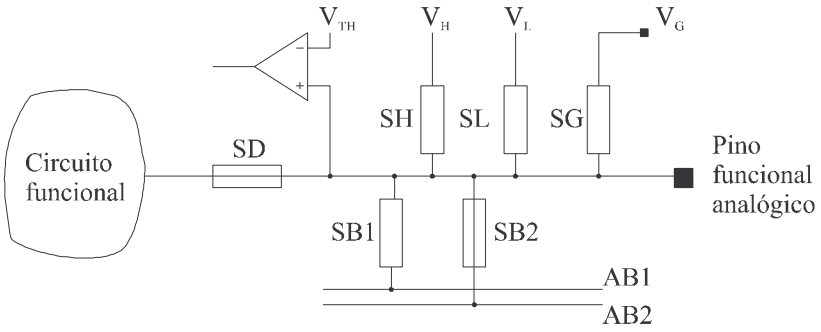


Figura 5-20: Estrutura de comutação do ABM associado ao sinal analógico a converter, nas condições definidas pela instrução *CΣΔ1P*.

Modificações no TBIC e na sua estrutura de controlo

As modificações a introduzir no TBIC[59], por forma a suportar a funcionalidade pretendida para a instrução opcional *CΣΔ1P*, resumem-se praticamente à inclusão na sua estrutura de comutação do modulador de 1ª ordem para o conversor ΣΔ, que deverá possuir como entrada analógica a linha AB2 do barramento analógico de teste interno, conforme se ilustra na Figura 5-21. Para além disso, a saída digital do modulador é ligada à célula CALIBRATE do registo de controlo do TBIC, conforme se ilustra na Figura 5-22.

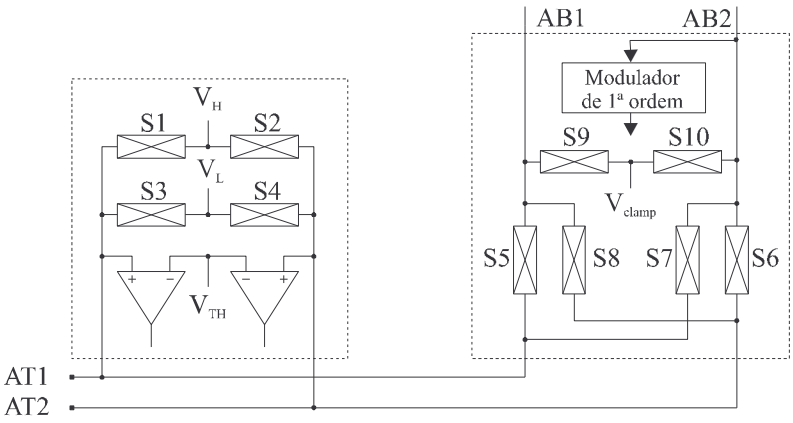


Figura 5-21: Modificações necessárias na estrutura de comutação do TBIC, para passar a suportar a instrução opcional *CΣΔ1P* (restringem-se à inclusão do modulador de 1ª ordem).

[58] $\lfloor(n/20)\rfloor$ representa o maior inteiro não superior a (n/20).

[59] O TBIC é composto por duas partes: a estrutura de comutação e a estrutura de controlo.

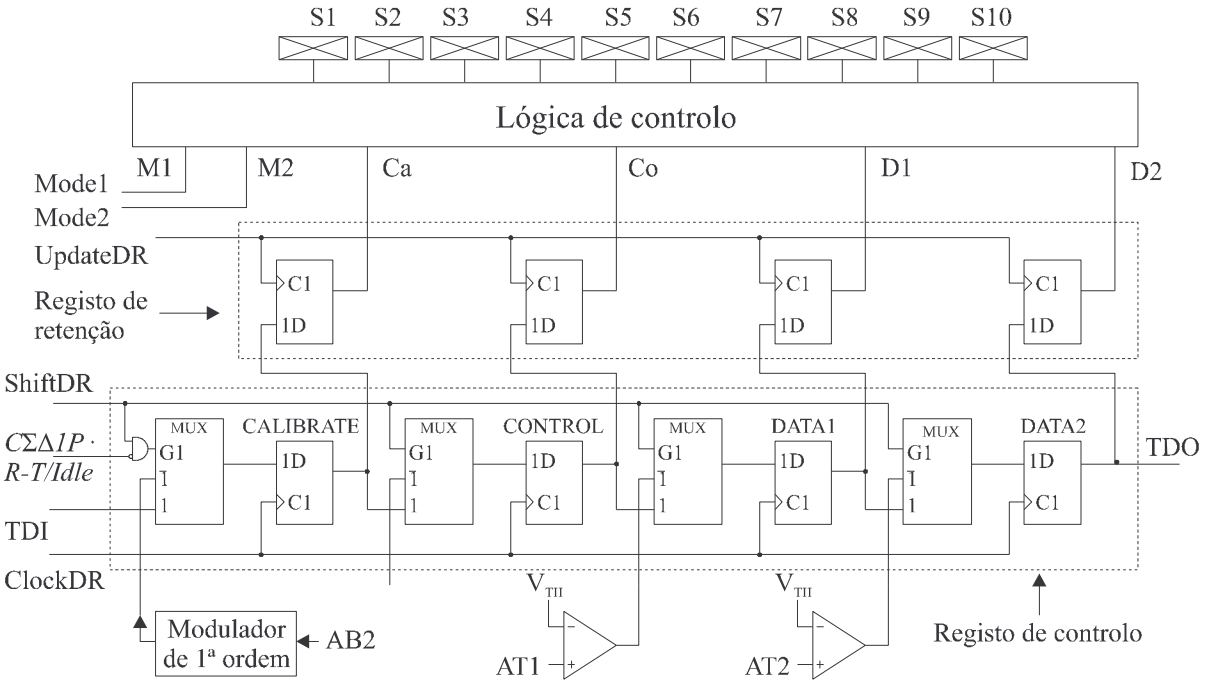


Figura 5-22: Estrutura de controlo do TBIC modificada para suportar a instrução opcional *CΣΔ1P*.

Alterações nos padrões de comutação do TBIC e ABM, e sinais de controlo do registo BS

O suporte da nova instrução opcional *CΣΔ1P* implica um novo padrão de comutação para o TBIC, a reutilização de um padrão de comutação para o ABM – sendo apenas necessário descodificar a instrução *CΣΔ1P* – e a alteração dos sinais ClockDR e ShiftDR. Todos os restantes sinais de controlo do registo BS mantêm as equações lógicas sugeridas na proposta de norma IEEE P1149.4 para o caso de um CI que suporte as quatro instruções obrigatórias (inclui agora a instrução *PROBE*, obrigatória para o caso de CI mistos).

O novo padrão de comutação para o TBIC possibilita que apenas a linha AB1 se encontre ligada à tensão de referência, representada pelo acrónimo V_{CLAMP}, através do interruptor S9. Todos os restantes interruptores deverão estar na posição de aberto. A Tabela 5-9 resume o estado dos interruptores do TBIC para suportarem a instrução opcional *CΣΔ1P*, seguindo o formato utilizado em [IEEE98, pág. 47]. A atribuição deste padrão ao TBIC não depende dos valores presentes nos A_R das células do seu registo de controlo, mas apenas da instrução actual, neste caso *CΣΔ1P* [IEEE98, pág. 48].

No caso dos ABM, aplica-se o padrão identificado pelo número dezassete, relativamente àquele que se encontra associado ao pino cujo valor se pretende converter, aplicando-se o padrão dezasseis aos restantes [IEEE98, pág. 66].

A Tabela 5-10 apresenta as novas equações lógicas dos sinais de controlo do registo BS, onde se destacam as alterações introduzidas para os sinais ClockDR e ShiftDR, que passam a incluir a instrução *CΣΔ1P*, de forma a cumprir os requisitos necessários para implementar o modo de funcionamento proposto.

Tabela 5-9: Padrão de comutação adicional para o TBIC, para suportar a instrução opcional *CΣΔ1P*.

Padrão	Estado dos interruptores										Ligações e função
	1	2	3	4	5	6	7	8	9	10	
P10	0	0	0	0	0	0	0	0	1	0	AB1/2 desligados de AT1/2. AB1 ligado à tensão de referência.

Tabela 5-10: Modificações nas equações lógicas dos sinais de controlo do registo BS, para implementar a instrução opcional *CΣΔ1P*.

Sinal	Infraestrutura obrigatória	Suporte da instrução *CΣΔ1P*
UpdateDR	!TCK·*Update-DR*	Mantém-se
ClockDR	TCK·(*Shift-DR* + *Capture-DR*)	TCK·(*Shift-DR* + *Capture-DR*·!*CΣΔ1P* + *Run-Test/Idle*·*CΣΔ1P*)
ShiftDR	*Shift-DR* (registado)	*Shift-DR* + *Run-Test/Idle* (registado)
Mode	*EXTEST*	Mantém-se
Mode1	Depende da instrução actual	Mantém-se
Mode2	Depende da instrução actual	Mantém-se

Questões a observar na implementação

Do anteriormente exposto decorrem naturalmente algumas questões de pormenor, referentes ao modo de implementação, nomeadamente:

- Modificar os valores presentes no A_D, durante *Run-Test/Idle*, com a instrução *CΣΔ1P* activa, é tolerado pela proposta IEEE P1149.4 [IEEE98, pág. 22 [60]].
- Manter o valor presente no A_D, durante *Capture-DR*, com a instrução *CΣΔ1P* activa, é tolerado pela proposta IEEE P1149.4 [IEEE98, pág. 22].

[60] *"... Rule 4.1.1(b) – The interpretation of the rules governing those structures and facilities that are common to this standard and to IEEE Std. 1149.1 shall be determined by IEEE Std. 1149.1."*

5.3 Detecção de tempos de atraso

O último modo de funcionamento opcional proposto tem por objectivo permitir a detecção de tempos de atraso em ligações entre pinos de CI com uma infraestrutura 1149.1, através da aplicação de vectores de teste e captura das respectivas respostas, durante o estado *Run-Test/Idle*. O intervalo entre o momento da aplicação do vector e o momento da captura da resposta reduz-se a meio período de TCK (distância entre os flancos descendente e ascendente). Este modo de funcionamento permite reduzir o intervalo normal entre aqueles dois eventos, que é de 2,5 ciclos de TCK [61] (repare-se que, se o *duty cycle* for inferior a 50%, esta redução será maior do que cinco vezes). Supondo um TCK com um período na ordem das poucas dezenas de nanosegundos (*ns*) [62], o primeiro intervalo equivale sensivelmente a uma centena de *ns* e o segundo a uma ou duas dezenas de *ns*. Para se determinar se este último valor permite ou não a detecção de faltas do tipo atraso, é necessário obter alguma informação relativa ao tempo de propagação máximo admissível para cada ligação. Este valor depende da frequência com que são trocados os valores lógicos, não sendo geralmente idêntico para todas as ligações. Na realidade, existem partes da CCI que funcionam a uma frequência muito menor que a frequência máxima de trabalho de um determinado CI. O processo de determinação do atraso máximo admissível para cada interligação não é simples e requer geralmente a utilização de ferramentas especializadas. Assume-se porém, para efeitos de apresentação do modo de funcionamento proposto, que na fase de projecto é determinado o atraso máximo admissível para cada ligação, que se designa na forma abreviada por ATR_{MAX}, dando-se a designação de TCK_{MIN} ao valor mínimo admissível para o período de TCK.

Procedimento / sequência de utilização

Se ½ TCK_{MIN} (assumindo um *duty cycle* de 50%) for inferior a ATR_{MAX}, utiliza-se directamente o sinal TCK para a detecção de faltas do tipo atraso. Caso contrário, é necessário reduzir o espaçamento entre os dois flancos, mantendo no entanto a mesma frequência para TCK, o que corresponde a reduzir o *duty cycle* deste sinal. Atendendo no entanto a que é suposto um *duty cycle* de 50% e devido aos atrasos nos sinais internos da lógica de teste e ao processo de deslocamento dos valores contidos nas células BS, é necessário restringir esta modificação ao nível do estado *Run-Test/Idle*. Esta restrição implica o controlo do relógio que regula o funcionamento do controlador residente de teste e depuração.

---

[61] Referido no capítulo 3.

[62] Uma assunção aceitável dado ser comum encontrar CI que especificam frequências para TCK na ordem dos 20/30 MHz.

Para o primeiro caso (½ TCK_{MIN} < $ATR_{MAX)}$ o método a seguir para a detecção de atrasos em ligações engloba as acções que se passam a descrever:

- Ajustar o período de TCK para o valor $TCK_{MIN.}$
- Deslocar a instrução *SAMPLE / PRELOAD* para todos os CI.
- Deslocar o primeiro vector de teste para o interior da cadeia BS da CCI.
- Deslocar a instrução opcional designada por *Detecção de Atrasos* (*DETRA*, na forma abreviada), para todos os CI que suportem o modo de funcionamento proposto. Esta instrução não permite a actualização dos A_R à passagem por *Update-DR*, nem a captura para os A_D à passagem por *Capture-DR*, reservando a ocorrência destes eventos para o estado *Run-Test/Idle*.
- Colocar o controlador do TAP no estado *Run-Test/Idle* durante um único ciclo de TCK, no decurso do qual ocorrerão a aplicação do vector (ao flanco descendente de TCK) e a captura das respostas (ao flanco ascendente de TCK).
- Deslocar a resposta para o exterior da cadeia BS e deslocar o próximo vector de teste para o seu interior. Se a resposta obtida for diferente da resposta esperada, assume-se que a ligação em que foi detectada a diferença possui um atraso superior a ATR_{MAX}[63].
- Repetir as duas acções anteriores até ser deslocada para o exterior a resposta ao último vector de teste.

Para o segundo caso (½ TCK_{MIN} > ATR_{MAX}) é necessário substituir o quinto passo anterior pelo seguinte conjunto de passos, que estão ilustrados na Figura 5-23:

- Aplicar uma transição ascendente em TCK, para colocar o controlador do TAP no estado *Run-Test/Idle*.
- Aplicar um impulso a '0' em TCK com a duração de ATR_{MAX} *ns*.
- Retomar o ciclo de funcionamento normal de TCK.

A solução para o segundo caso implica que o circuito de geração do relógio (TCK) permita satisfazer os seguintes requisitos:

- Parar o fornecimento de impulsos.
- Provocar uma transição ascendente isolada.
- Aplicar um impulso a '0' de duração programável, com uma resolução o mais elevada possível.

[63] Assume-se que o teste de ligações, seguindo os modelo de faltas do tipo SS@ e curto-circuito, foi efectuado anteriormente, utilizando a instrução *EXTEST*, não tendo sido detectada nenhuma falta.

- Retomar o estado normal de funcionamento, i.e. aplicar ciclos de relógio com uma frequência pré-estabelecida e comunicar esse estado para o exterior (através de um sinal de indicação de estado).

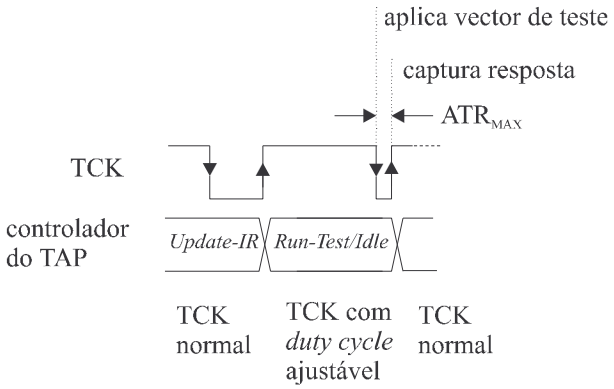


Figura 5-23: Alteração do *duty cycle* do TCK, para a detecção de atrasos em ligações.

A Figura 5-24 ilustra em maior pormenor a aplicação da instrução opcional *DETRA*. A captura da resposta, efectuada ½ TCK_{MIN} ou ATR_{MAX} *ns* após a aplicação na ligação de uma transição, permite verificar se o atraso existente na ligação é inferior ou idêntico ao máximo admissível. O deslocamento do vector seguinte permite que o valor lógico presente na ligação estabilize, por forma a que se teste em seguida a resposta à transição oposta. No caso de existirem capacidades parasitas, a situação complica-se porém um pouco mais. Supondo a existência de uma capacidade parasita entre duas ligações, é necessário que no momento em que se aplica uma transição numa das ligações, a outra mantenha o mesmo valor anterior, para que a capacidade seja carregada e a carga vista pela saída da primeira ligação seja maior. Este tipo de teste é equivalente ao teste de curto-circuitos (as capacidades parasitas comportam-se desta forma para frequências elevadas). Neste caso é necessário utilizar um dos algoritmos existentes para o efeito, como o algoritmo da partição binária, ou outros [Yau89].

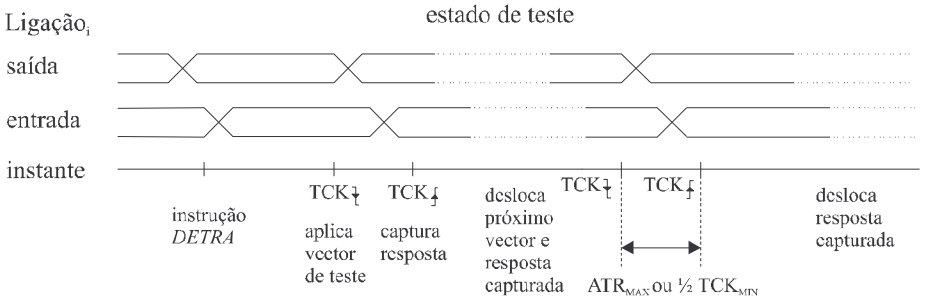


Figura 5-24: Diagrama temporal da detecção de atrasos em ligações.

Modificações na estrutura das células BS

Supondo que os CI possuem a infraestrutura mínima obrigatória definida na norma IEEE 1149.1, é necessário unicamente proceder a alterações na lógica dos sinais de controlo do registo BS, uma vez que o modo de funcionamento opcional proposto não implica qualquer modificação na estrutura das células BS.

Alterações nos sinais de controlo das células BS

Tomando como ponto de referência as equações lógicas sugeridas na norma IEEE 1149.1 para o caso de um CI que suporte as três instruções obrigatórias, apresentam-se na Tabela 5-11 as novas equações lógicas para os sinais ClockDR, UpdateDR e Mode. A equação lógica do sinal Shift mantém-se idêntica. As novas equações lógicas implementam o modo de funcionamento pretendido para a instrução opcional *DETRA*, mantendo a funcionalidade anterior das três instruções obrigatórias.

Tabela 5-11: Modificações nas equações lógicas dos sinais de controlo das células BS, para implementar a instrução opcional *DETRA*.

Sinal	Infraestrutura obrigatória	Suporte da instrução *DETRA*
UpdateDR	!TCK·*Update-DR*	!TCK·(*Update-DR*·!*DETRA* + *Run-Test/Idle*·*DETRA*)
ClockDR	TCK·(*Shift-DR* + *Capture-DR*)	TCK·(*Shift-DR* + *Capture-DR*·!*DETRA* + *Run-Test/Idle*·*DETRA*)
Shift	*Shift-DR* (registado)	Mantém-se
Mode	*EXTEST*	*EXTEST* + *DETRA*

Questões a observar na implementação

Do anteriormente exposto decorrem naturalmente algumas questões de pormenor, referentes ao modo de implementação, nomeadamente:

- Modificar os valores presentes no A_D e no A_R, durante *Run-Test/Idle*, com a instrução *DETRA* activa, é tolerado pela norma IEEE 1149.1 [IEEE93, pág. 5-3].
- Manter o valor presente no A_D, durante *Capture-DR*, com a instrução *DETRA* activa, é tolerado pela norma IEEE 1149.1 [IEEE93, pág. 5-4].
- Manter o valor presente no A_R, durante *Update-DR,* com a instrução *DETRA* activa, não parece constituir uma violação da norma [IEEE93, pág. 5-5], dado não ser obrigatório que o A_R capture o valor presente no A_D, para todas as instruções.

5.4 Conclusão

Os modos de funcionamento opcionais propostos, para infraestruturas de teste baseadas nas normas IEEE 1149.1 e P1149.4, têm associadas determinadas vantagens e desvantagens, que se passam a analisar de uma forma qualitativa.

A detecção em tempo real de pontos de paragem por condição possui duas vantagens principais: o facto de permitir a implementação de pontos de paragem em tempo real e o facto de permitir que a condição englobe valores referentes a todos os pinos do CI. Estas duas vantagens são cruciais, primeiro porque determinadas condições só ocorrem quando o circuito sob depuração se encontra a trabalhar à sua frequência máxima, e em segundo porque se todos os CI suportarem este modo de funcionamento, as condições podem eventualmente estender-se a todas as ligações do circuito. As principais desvantagens decorrem do custo associado à implementação das duas novas instruções opcionais, nomeadamente:

- O registo STC.
- A inclusão de um pino adicional (SCD).
- Eventualmente, a inclusão de um segundo pino adicional (ECD).
- A lógica de detecção de condição verdadeira.

A quantidade de lógica necessária depende do número de condições suportadas, tendo-se restringido este número a oito, na descrição efectuada na secção 5.1. As alterações nas equações lógicas do sinais de controlo das células BS são pequenas, dado que apenas a equação lógica do sinal UpdateDR sofre uma ligeira alteração. Refira-se porém que o número de condições suportadas pode ser sempre aumentado à custa de mais lógica adicional. O momento em que se deve considerar válido o valor lógico da condição, depende do atraso na geração do sinal SCD, que por sua vez depende do modo de implementação, tendo-se relegado este assunto para a funcionalidade a implementar no controlador residente de teste e depuração. Esta relação poderá ser desvantajosa, no caso de se optar por não ter este controlador residente no circuito sob depuração.

A amostragem em tempo real dos valores presentes nos pinos funcionais do CI, constitui uma das técnicas de depuração mais utilizadas na detecção daqueles problemas que se manifestam unicamente quando o circuito se encontra a trabalhar, ininterruptamente, à sua frequência máxima. As várias instruções opcionais propostas possuem como vantagem principal a possibilidade de dispensar as ferramentas tradicionais utilizadas para este efeito, que se baseiam no acesso físico. Do ponto de vista do custo de implementação, pode-se afirmar que os recursos necessários para a instrução opcional *MSEQ* são praticamente irrelevantes, quando comparados

com a quantidade de lógica associada à infraestrutura de teste obrigatória: mais uma porta lógica AND e uma linha para cada célula BS, e mais algumas portas lógicas para geração dos sinais UpdateDR, ClockDR e Select. As principais desvantagens decorrem do número reduzido de amostras, limitadas a duas para uma célula BS sem A_R adicionais, e da violação da norma IEEE 1149.1, no que respeita à modificação dos valores existentes no registo BS, durante o estado *Exit2-DR*. A velocidade de amostragem pode no entanto atingir valores superiores à máxima frequência de TCK especificada para o CI, dado que não existe lógica interposta entre a saída do A_D e a entrada do A_R de uma célula BS. O sincronismo entre o funcionamento do circuito sob depuração e o momento da amostragem é novamente assegurado pelo controlador residente, podendo considerar-se mais uma vez esta relação como uma desvantagem do modo de funcionamento opcional proposto. A instrução opcional *MSATC* tem ainda como desvantagem adicional o facto de obrigar à implementação do pino de entrada ECD e de uma MEF (embora de dimensões reduzidas). Na instrução opcional *MSAPC* pode-se optar entre a inclusão do pino ECD (condições externas ao CI) ou a implementação da funcionalidade associada à detecção em tempo real de pontos de paragem por condição (condições internas do CI). A dimensão da MEF cresce ligeiramente (passa de dois para quatro estados), embora continue a não ter um peso significativo em termos de lógica adicional.

A instrução opcional *MSEQ1P* permite aumentar o número de amostras em função da redução do número de pinos amostrados a um só, seleccionável através de um registo de dados adicional. Apesar de não se utilizar o A_R, o número de células BS existentes num CI permite desde logo o armazenamento de um número significativo de amostras. A velocidade de amostragem não deverá ultrapassar a frequência máxima especificada para TCK, dado que se utiliza o A_D em modo de deslocamento série, para armazenar as amostras efectuadas. A lógica adicional associada à instrução opcional *SELPIN*, o registo SP, o descodificador e a inclusão de mais uma porta AND em cada célula BS (na célula BS mais próxima de TDI é ainda necessário mais um multiplexador de 2:1), constituem a principal desvantagem deste modo de funcionamento. As alterações nos sinais de controlo das células BS reduzem-se aos sinais ClockDR e Shift, podendo considerar-se mínimas. A sincronização entre o momento da amostragem e o funcionamento do circuito sob depuração continua a depender do controlador residente. Uma vantagem adicional desta instrução opcional, em comparação com as três anteriores, consiste na leitura dos valores amostrados, que se baseia simplesmente no seu deslocamento para o exterior no estado *Shift-DR*, evitando-se assim a anterior violação da norma IEEE 1149.1.

A instrução dual da anterior, para os CI mistos com infraestrutura de teste compatível com a proposta IEEE P1149.4, *CΣΔ1P*, requer a implementação de um conversor A/D para possibilitar a amostragem / conversão / retenção do valor em qualquer pino analógico do CI. Optou-se por um conversor ΣΔ de 1-bit, pelas seguintes razões:

- A resolução é elevada.
- A frequência de sobre-amostragem pode-se aproximar da frequência máxima de TCK.
- Pode-se particionar o conversor no modulador e no filtro decimador, sendo o primeiro implementado no interior do CI e o segundo no seu exterior.
- O facto de a saída digital ser apenas de 1-bit permite a aplicação directa do modo de funcionamento opcional suportado pela instrução *MSEQ1P*, com algumas vantagens, uma vez que se assume que a localização desta saída é fixa, não sendo necessário implementar o registo SP, a instrução *SELPIN* e o descodificador.

Este modo de funcionamento opcional possui como vantagem principal o facto de permitir obter, através do percurso TDI - TDO, o valor presente num qualquer pino analógico. A instrução obrigatória *PROBE* serve já este propósito com eventual superioridade (por permitir a observação do valor analógico em tempo real, no pino AT2 do ATAP), mas tem a limitação de só tornar possível a observação de um único pino em cada instante. Face a esta limitação, a instrução *CΣΔ1P* apresenta a vantagem de permitir obter uma representação digital dos valores presentes simultaneamente em vários pinos analógicos (em CI diferentes). Devem porém referir-se algumas desvantagens, nomeadamente:

- O modulador do conversor ΣΔ ocupa uma área de silício não desprezável. Em contrapartida, refira-se que são mínimas as alterações nas equações lógicas dos sinais de controlo do registo BS e que apenas é necessário acrescentar uma porta lógica AND à célula CALIBRATE, mantendo-se inalteradas todas as outras células.
- A linha AB2 do barramento interno de teste analógico é utilizada para ligar o pino a observar à entrada do modulador, ficando indisponível para outros efeitos.
- É necessário um padrão de comutação adicional para o TBIC.
- O resultado da conversão apenas está disponível após o deslocamento para o exterior dos valores amostrados / armazenados no registo BS, e da sua passagem pelo filtro decimador. O intervalo de tempo entre o momento da amostragem e a visualização do resultado da conversão depende assim do tamanho da cadeia BS do circuito sob depuração.

A detecção de atrasos em ligações, com a aplicação de vectores de teste através das células BS, permite aumentar o *índice de cobertura de defeitos* final do teste estrutural efectuado com base nesta infraestrutura, de onde resulta um aumento da confiança na qualidade do protótipo. O modo de funcionamento opcional proposto constitui uma tentativa de passar para o domínio do teste estrutural uma classe de erros que é normalmente detectada através de um teste funcional, facilitando-se assim a sua detecção e diagnóstico, de modo a poupar tempo na depuração

deste tipo de problemas. Se os atrasos máximos admissíveis nas várias ligações forem inferiores a metade do período mínimo admissível para TCK, torna-se no entanto necessário controlar o *duty cycle* deste sinal, num determinado momento, o que resulta numa capacidade acrescida a suportar pelo circuito de geração de relógio (TCK). A quantidade de lógica adicional é insignificante, dado que a estrutura das células BS se mantém inalterada e as alteração nas equações lógicas dos sinais de controlo se reduzem a UpdateDR e ClockDR.

Capítulo 6

Arquitectura do controlador de teste e depuração

No capítulo 4 foram identificados os requisitos principais a que deveria obedecer um controlador de teste e depuração de sistemas que incluam componentes com uma infraestrutura de teste compatível com a norma IEEE 1149.1. Com base nestes requisitos desenvolveu-se o controlador denominado PRODEP (PROcessador de DEPuração), que é descrito no presente capítulo. A sua arquitectura assenta em duas unidades principais que endereçam o controlo da infraestrutura de teste e o controlo da lógica funcional em simultâneo, de uma forma sincronizada, dado que ambas as unidades partilham o mesmo relógio principal – assegurando a execução de operações em sincronismo. Para sincronizar a execução de grupos de instruções existe um canal de sincronismo, que interliga as duas unidades, comandado através de instruções próprias de uma e outra unidades. A arquitectura é apresentada de uma forma modular, iniciando-se com a descrição do nível de topo e a definição dos pinos de Entrada / Saída, seguida da descrição mais detalhada de cada uma das unidades, primeiro com uma visão geral da constituição interna, depois dos blocos principais, em seguida do conjunto de instruções e finalmente de cada uma das instruções, com a apresentação dos diagramas de transição de estados. No final refere-se em maior pormenor a questão do sincronismo entre as duas unidades, encerrando-se o capítulo com a abordagem de algumas questões inerentes à implementação do controlador de teste e depuração.

6.1 Arquitectura de topo e definição de pinos de Entrada / Saída

A necessidade de controlar em paralelo, e de uma forma síncrona, a lógica de teste e a lógica funcional do sistema sob depuração, deu origem à escolha de uma arquitectura baseada em dois processadores para implementar o PRODEP. Uma das unidades é responsável pelo controlo da lógica de teste (CLT), sendo a outra responsável pelo controlo da lógica funcional (CLF). Esta

decisão permitiu desde logo dividir o esforço de desenvolvimento, facilitando a reutilização de um processador anteriormente desenvolvido para o controlo do teste estrutural de CCI com uma ou duas cadeias BST [Fer92a, Fer92b, Fer92c, Mat92]. A disponibilidade de uma descrição em *Hardware Description Language* (HDL) de uma versão deste processador, desenvolvida para permitir a sua implementação em lógica programável [Alv93, Alv95, Fer93] serviu de base para o desenvolvimento da unidade CLT. Ao partir-se de uma base já suficientemente utilizada e testada, acelerou-se o processo de desenvolvimento do PRODEP.

A Figura 6-1 ilustra a arquitectura de topo e os pinos de Entrada / Saída do PRODEP. Cada uma das unidades corresponde a um processador dedicado que executa um programa armazenado em memória. A unidade CLT partiu do núcleo já anteriormente desenvolvido e permite controlar directamente até dois TAP, possuindo um canal de sincronismo com equipamentos de teste exteriores e um conjunto de saídas para indicação do estado interno. A unidade CLF é a que se responsabiliza pelo controlo e observação da lógica funcional, possuindo igualmente um canal de sincronismo com o exterior. Um conjunto de entradas e saídas genéricas permite ainda o controlo / observação directo de pinos de componentes.

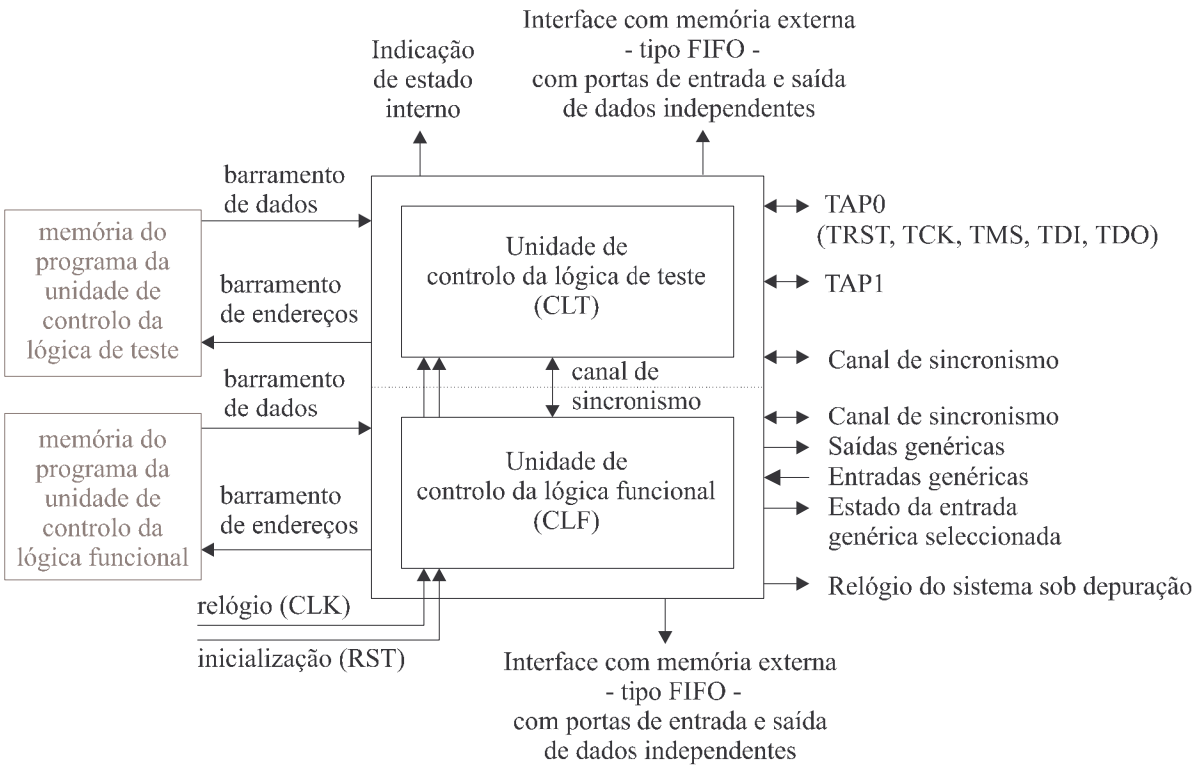


Figura 6-1: Arquitectura de topo e pinos de entrada / saída do PRODEP.

Ambas as unidades possuem ainda um interface com uma memória externa do tipo FIFO com portas de entrada e saída de dados independentes, para passagem de informação para o

exterior. As duas unidades estão interligadas por um canal comum de sincronismo e pela partilha de linhas de relógio (CLK) e de inicialização (RST) comuns.

6.2 A unidade de controlo da lógica de teste

Esta secção descreve a constituição interna, o conjunto de instruções, e cada uma das instruções, individualmente, da unidade CLT do PRODEP. Esta descrição apresenta uma versão actualizada da arquitectura que tinha sido anteriormente desenvolvida para uma unidade deste tipo, tal como foi descrita em [Alv95]. A Figura 6-2 ilustra o diagrama de blocos da arquitectura interna onde se destacam os seguintes elementos:

- Bloco de controlo e descodificação de instruções.
- Registo de programa (PC, *Program Counter*).
- Contadores (CNT, *Counters*).
- Registos de retenção temporária (TMPREG, *Temporary Registers*).
- Interface com memória FIFO externa (desserializador).
- Bloco de controlo da saída série (TDO) para cada TAP.
- Selecção do TAP da cadeia activa.
- Bloco de controlo da entrada série (TDI) proveniente de cada TAP.
- Bloco das saídas dos canais de sincronismo, e de indicação do estado interno.

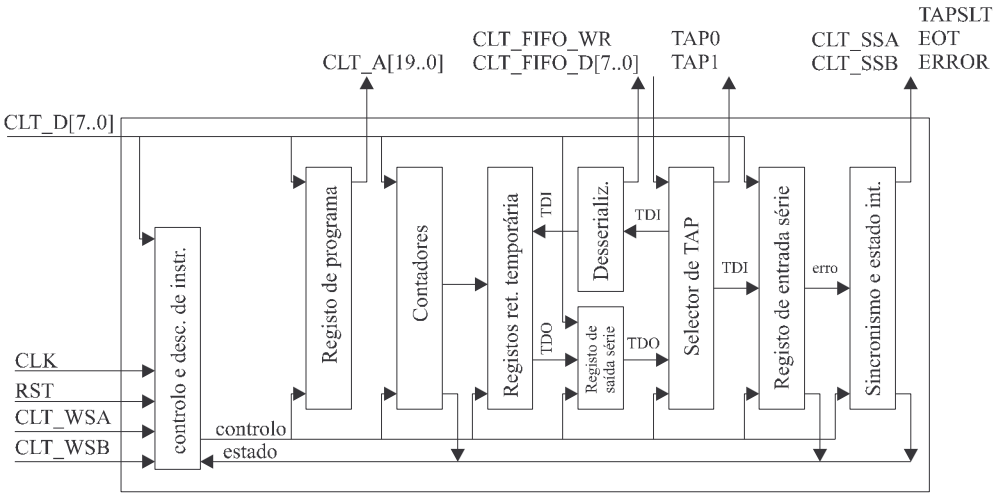


Figura 6-2: Diagrama de blocos da arquitectura interna da unidade CLT.

A Figura 6-2 ilustra ainda, a um nível mais pormenorizado, a constituição dos pinos de entrada e saída da unidade CLT, destacando-se um barramento de dados com oito bits (CLT_D[7..0]), um barramento de endereços com vinte bits (CLT_A[19..0]), que permite e n-

dereçar memórias com até 1 Mbyte, os sinais de interface com a memória externa tipo FIFO com entrada e saída de dados independentes, igualmente com oito bits (CLT_FIFO_D[7..0] e CLT_FIFO_WR), os sinais existentes nos dois TAP independentes (TAP0 e TAP1 – TRST0, TMS0, TCK0, TDO0, TDI0, similar para o TAP1), os sinais de entrada e saída dos canais de sincronismo A e B (CLT_WSA, CLT_WSB, CLT_SSA, CLT_SSB), e os sinais de indicação do estado interno (TAPSLT – TAP *Selected*, EOT – *End of Test*, e ERROR).

6.2.1 Blocos principais

Cada um dos blocos identificados na Figura 6-2 é em seguida descrito, ilustrando-se o diagrama de topo com a identificação dos sinais de entrada e saída, número mínimo de FF necessários, e referindo-se ainda em breves palavras qual a funcionalidade implementada.

Bloco de controlo e descodificação de instruções

A Figura 6-3 ilustra o diagrama interno deste bloco, onde se distinguem três componentes principais:

- O registo de instrução, destinado a receber o código de cada instrução a executar, que é constituído por cinco FF, permitindo codificar 32 (2^5) instruções.
- Uma MEF responsável por gerar a necessária sequência de estados, de acordo com o código presente no registo de instrução e com os sinais de estado provenientes do exterior ou de alguns dos blocos da unidade CLT. Esta MEF é constituída por cinco FF, que indicam o estado actual, e por um bloco combinatório que determina qual o estado seguinte.
- Um circuito combinatório responsável pela geração dos sinais de controlo para todos os blocos da unidade CLT, de acordo com o estado actual e com o código presente no registo de instrução.

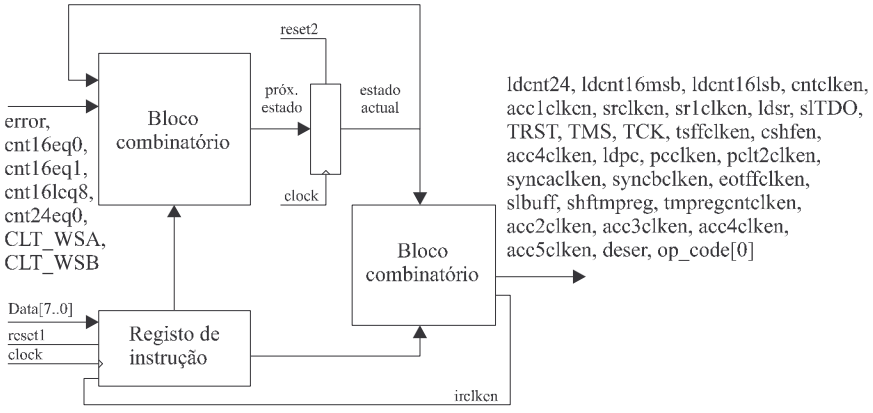


Figura 6-3: Diagrama interno do bloco de controlo e descodificação de instruções da unidade CLT.

Este bloco é igualmente responsável pela leitura do código da instrução seguinte, que tem lugar sempre que a MEF retorna ao estado inicial, a que se atribui por convenção a designação de estado zero (0).

Registo de programa

A Figura 6-4 ilustra o diagrama de topo deste bloco, responsável pelo controlo do barramento de endereços para a memória que contém o programa da unidade CLT.

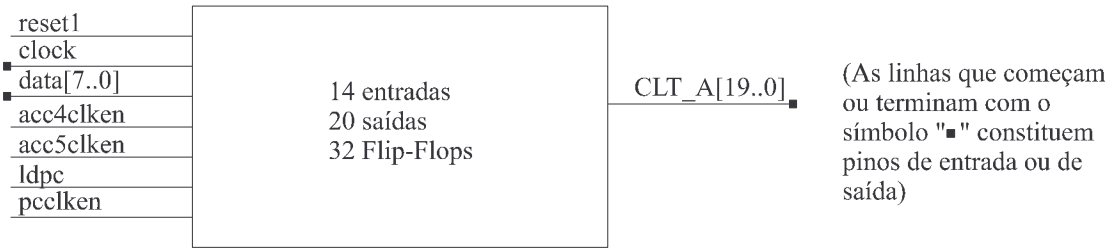


Figura 6-4: Diagrama de topo do registo de programa da unidade CLT.

O registo de programa é constituído por um contador síncrono de vinte bits, com possibilidade de carga paralela, para suportar instruções de salto. Uma vez que a largura do barramento de dados é de oito bits, torna-se necessária a existência de dois acumuladores, um com oito e outro com quatro bits, de modo a permitir que a carga dos três andares do registo de programa tenha lugar em paralelo – andar menos significativo (oito bits) do registo de programa carregado com o conteúdo do barramento de dados, andar intermédio (oito bits) carregado com o conteúdo do acumulador de oito bits e andar mais significativo (quatro bits) carregado com o conteúdo do acumulador de quatro bits. As saídas do barramento de endereços são colocadas num estado de alta-impedância quando a linha de inicialização interna (*reset1*) se encontra activa ao nível lógico baixo '0'.

Contadores

A Figura 6-5 ilustra o diagrama de topo deste bloco, constituído por dois contadores descendentes síncronos, com carga paralela, de 24 e dezasseis bits. O contador interno seleccionado é decrementado pela unidade de controlo e descodificação de instruções, durante a execução de instruções que requerem um determinado número de ciclos do relógio de teste (TCK). Em virtude de cada uma destas instruções apenas necessitar de um contador activo, os dois contadores partilham os dezasseis bits menos significativos, resultando numa economia de recursos. A

carga dos contadores é efectuada sem auxílio de acumuladores, uma vez que os possíveis estados intermédios não influenciam o comportamento pretendido para este bloco.

Figura 6-5: Diagrama de topo do bloco dos contadores da unidade CLT.

Este bloco proporciona à unidade de controlo e descodificação vários sinais que indicam qual o seu estado interno, nomeadamente se o conteúdo de ambos os contadores é igual a zero (fim de contagem), se o conteúdo do contador de dezasseis bits é igual ou inferior a oito, ou se é igual a um.

Registos de retenção temporária

A Figura 6-5 ilustra o diagrama de topo deste bloco, responsável pelo armazenamento temporário dos valores capturados na entrada série (TDI) da unidade CLT, durante a execução de um determinado tipo de instruções de deslocamento (a apresentar adiante, na subsecção 6.2.2).

Figura 6-6: Diagrama de topo do bloco de registos de retenção temporária da unidade CLT.

Conforme se ilustra na Figura 6-7, este bloco é constituído internamente por dois módulos parametrizados (*Library of Parameterized Modules*, LPM) de memória, um circuito de selecção do módulo activo, um contador descendente, síncrono, de onze bits, que funciona como apontador de memória, e um multiplexador de saída. Dependendo dos recursos disponíveis no

circuito onde se implementa o PRODEP, é possível modificar os parâmetros dos LPM, por forma a configurar a sua dimensão. Refira-se neste ponto que na implementação efectuada, a descrever mais adiante, cada LPM foi configurado como uma memória de 2 Kbit, com barramentos de dados de entrada e saída independentes, registados, de largura igual a um bit, e com sinal de permissão de escrita síncrono. Com base neste parâmetros, projectou-se um contador com onze bits (2^{11} = 2048 = 2 Kbits) que permite apontar a célula activa num dado momento, embora a sua dimensão possa ser rapidamente alterada, pelo facto de o projecto se encontrar especificado numa HDL.

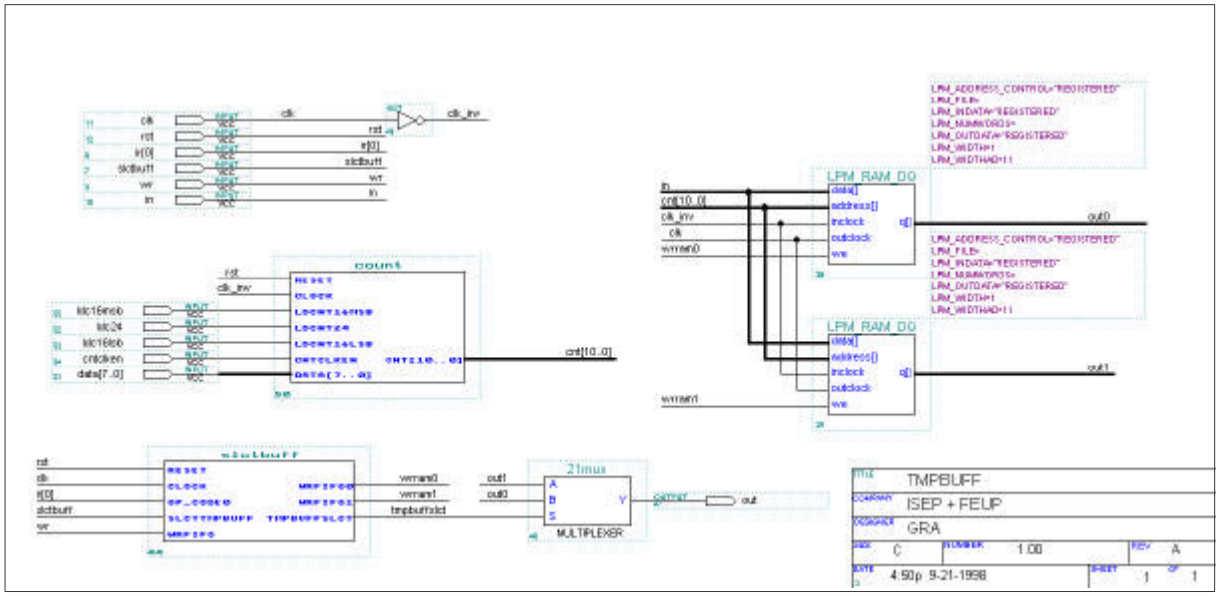


Figura 6-7: Diagrama interno do bloco de registos de retenção temporária da unidade CLT.

Interface com uma FIFO externa (desserializador)

A Figura 6-8 ilustra o diagrama de topo deste bloco, responsável pelo armazenamento numa memória externa dos valores capturados na entrada série (TDI) activa da unidade CLT. Com o objectivo de facilitar o desenvolvimento deste bloco optou-se por uma memória externa do tipo FIFO com barramentos de entrada e saída de dados independentes. Desta forma, evitou-se a necessidade de implementar um apontador de memória (repare-se que a escrita é sequencial, pelo que não existem inconvenientes nesta escolha), e um barramento de dados com controlo de estado, reduzindo-se a interface de controlo a uma única linha de indicação de escrita (CLT_FIFO_WR). Dado que se optou ainda por uma memória externa com um barramento de dados com oito bits de largura, os valores capturados em TDI são agrupados de oito em oito bits no interior do bloco, para efeitos de escrita.

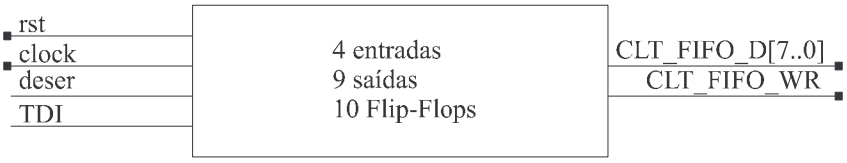


Figura 6-8: Diagrama de topo da interface com uma FIFO externa da unidade CLT.

Controlo da saída série (TDO) para cada TAP

A Figura 6-9 ilustra o diagrama de topo deste bloco, constituído por um registo de deslocamento de oito bits, com carga paralela, que se destina a permitir a inserção de vectores no interior da cadeia activa do sistema sob depuração.

Figura 6-9: Diagrama de topo do bloco de controlo da saída série (TDO) para cada TAP da unidade CLT.

O registo de deslocamento é carregado através de um acumulador de oito bits, que tem por objectivo permitir que a unidade CLT execute concorrentemente a carga dos próximos oito bits, durante o deslocamento dos actuais. Esta possibilidade tem um interesse particular no que respeita às operações de deslocamento com comparação, onde as operações de leitura dos valores a deslocar, valores a comparar e máscaras de comparação, podem deste modo ser efectuadas durante o deslocamento dos oito bits actuais, o que optimiza a velocidade de inserção de vectores na cadeia activa. Este bloco possui ainda um multiplexador que permite controlar a origem dos vectores a deslocar: memória do programa ou registo de retenção temporária actualmente seleccionado.

Controlo da entrada série (TDI) proveniente de cada TAP

Este bloco destina-se a efectuar a comparação dos bits deslocados para o exterior da cadeia activa, relativamente ao seu valor esperado. Nem todos os bits possuem um valor esperado conhecido, pelo que é usada uma máscara para indicar aqueles cuja comparação é relevante. A Figura 6-10 ilustra o diagrama de topo deste bloco, constituído por dois registos de deslocamento de oito bits, com carga paralela, e por um circuito de comparação.

Figura 6-10: Diagrama de topo do bloco de controlo da entrada série (TDI) proveniente de cada TAP da unidade CLT.

Cada registo de deslocamento é carregado a partir de um acumulador de oito bits, com o objectivo já descrito de executar concorrentemente os acessos à memória, para a leitura dos novos conteúdos de cada registo, durante a própria operação de deslocamento. Cada bit em '0', na máscara de comparação, indica que o conteúdo da posição respectiva não é relevante para a comparação.

Selecção do TAP da cadeia activa

A Figura 6-11 ilustra o diagrama de topo deste bloco, que se destina a efectuar a multiplexagem entre os recursos proporcionados pela unidade CLT e os dois TAP disponíveis. Uma das linhas de entrada deste bloco permite controlar a origem dos valores a deslocar para o interior da cadeia activa: linha de saída do bloco de controlo da saída série[64] (TDO) ou a linha de entrada série (TDI) do TAP activo da unidade CLT. Repare-se que esta última possibilidade permite efectuar um deslocamento circular do conteúdo da cadeia activa ou seja, o valor deslocado para o exterior da cadeia é igualmente deslocado para o seu interior.

[64] Onde já era possível seleccionar entre vectores armazenados na memória do programa ou no registo de retenção temporária activo.

Figura 6-11: Diagrama de topo do bloco de selecção do TAP activo da unidade CLT.

Canais de sincronismo e sinais de estado

A Figura 6-12 ilustra o diagrama de topo deste bloco, que agrupa o controlo das saídas dos canais de sincronismo e a geração dos sinais de estado exteriores da unidade CLT, que incluem a indicação do estado de fim-de-teste (EOT - actuado pela execução da instrução que termina o programa), a indicação de qual dos dois TAP se encontra activo (TAPSLD) e a indicação da ocorrência de uma diferença entre um valor esperado e um valor lido, com a máscara de comparação activa (ERROR).

Figura 6-12: Diagrama de topo do bloco dos canais de sincronismo e sinais de estado da unidade CLT.

A parte relativa aos canais de sincronismo é essencialmente constituída por dois registos de um bit, que permitem memorizar o valor lógico especificado pela execução das instruções que controlam os pinos de saída destes canais. Os correspondentes pinos de entrada destes canais encontram-se directamente ligados ao bloco de controlo e descodificação de instruções.

6.2.2 Conjunto de instruções

Apresenta-se em seguida o conjunto de instruções suportadas pela unidade CLT, que permitem o controlo da infraestrutura de teste do sistema sob depuração, quer para efeitos de operações de teste quer para efeitos de operações de depuração. A Tabela 6-1 descreve sucintamente as instruções para controlo da infraestrutura de teste, que fazem parte do conjunto original de instruções do processador referido em [Fer92a, Fer92b, Fer92c, Mat92]. A Tabela 6-2 descreve sucintamente as instruções que lhe foram adicionadas por forma a expandir as suas capacidades no apoio a operações de depuração através da infraestrutura de teste. Deve-se salientar que a unidade CLT corresponde à reunião deste processador com o conjunto das novas capacidades, que correspondem às funções permitidas pelas instruções referidas nesta tabela. Finalmente, a Tabela 6-3 descreve sucintamente as instruções para o controlo dos recursos internos, canais de sincronismo e fluxo de programa da unidade CLT. Algumas das instruções fazem parte do conjunto original de instruções do processador (LD C16, N; SSA0; SSA1; SSB0; SSB1; WSA0; WSA1; WSB0; WSB1; HALT), tendo outras sido adicionadas ou ligeiramente modificadas (LD C24, N; JPE Endereço; e JPNE Endereço) no momento em que se passou de uma implementação em ASIC para uma implementação em lógica programável, de que resultou um aumento da capacidade de endereçamento do processador (de dezasseis para vinte linhas) e o acrescento de mais um contador interno, com 24 bits, para suportar a execução de funções de auto-teste[65] em CI.

Tabela 6-1: Instruções para controlo da infraestrutura de teste.

Instruções para controlo da infraestrutura de teste	
SELTAP0, SELTAP1	Selecciona qual o TAP activo.
TRST	Aplica um impulso ao nível lógico baixo ‘0’ na linha TRST do TAP activo.
TMS0, TMS1	Aplica um impulso em TCK, mantendo a linha TMS no valor pretendido.
NSHF	Efectua o deslocamento, sem comparação, de um vector para a cadeia activa.
NSHFCP	Efectua o deslocamento, com comparação, de um vector para a cadeia activa.
NTCK	Aplica N impulsos em TCK, mantendo a linha TMS no nível lógico baixo ‘0’.

[65] No processador original utilizava-se um contador de dezasseis bits para este efeito. Contudo, componentes como o microprocessador 80486 da INTEL, necessitam de cerca de um milhão ($\approx 2^{20}$) de impulsos de relógio de teste (TCK) para executar as suas funções de auto-teste, pelo que se justificou a necessidade de implementar um contador com mais capacidade, neste caso com 24 bits.

Tabela 6-2: Instruções adicionais para suporte de operações de depuração.

Instruções adicionais para suporte de operações de depuração	
STCK	Aplica impulsos em TCK enquanto a entrada de sincronismo A se mantiver no nível lógico alto '1'.
STMPB0, STMPB1	Selecciona o registo de retenção temporária pretendido.
NCSHF	Efectua um deslocamento circular, sem comparação, da cadeia activa. O valor capturado na entrada série (TDI) é armazenado no registo de retenção temporária seleccionado e simultaneamente aplicado na saída série (TDO).
NCSHFCP	Efectua um deslocamento circular, com comparação, da cadeia activa. O valor capturado na entrada série (TDI) é armazenado no registo de retenção temporária seleccionado e simultaneamente aplicado na saída série (TDO).
NSHFB2C	Efectua o deslocamento do vector contido no registo de retenção temporária seleccionado, para o interior da cadeia activa.
NSHFCPB2C	Efectua o deslocamento, com comparação, do vector contido no registo de rete n-ção temporária seleccionado, para o interior da cadeia activa.

Tabela 6-3: Instruções para o controlo dos recursos internos, canais de sincronismo e fluxo do programa.

Instruções para controlo dos recursos internos, canais de sincronismo e fluxo do programa	
LD C16, N	Carrega o contador interno de dezasseis bits com o número N.
LD C24, N	Carrega o contador interno de 24 bits com o número N.
SSA0, SSA1, SSB0, SSB1	Coloca valor pretendido na saída de sincronismo (A ou B).
WSA0, WSA1, WSB0, WSB1	Espera até que a entrada de sincronismo (A ou B) se encontre no valor lógico pretendido.
JPE Endereço, JPNE Endereço	Implementa um salto condicional, de acordo com o estado do pino de saída de erro (ERROR).
HALT	Conclui a execução do programa.

Cada uma das instruções apresentadas nas tabelas anteriores, é descrita em seguida de uma forma mais pormenorizada, nomeadamente através da apresentação do seu diagrama de transição de estados.

Instruções para controlo da infraestrutura de teste

Apresenta-se em seguida a caracterização individual das instruções da unidade CLT descritas na Tabela 6-1, que permitem controlar a infraestrutura de teste do sistema sob depuração.

SELTAP0, SELTAP1

Estas instruções permitem seleccionar qual dos dois TAP da unidade CLT será controlado pelas instruções seguintes. A Figura 6-13 ilustra o diagrama de transição de estados para estas instruções.

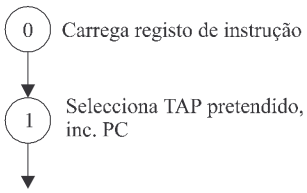


Figura 6-13: Diagrama de transição de estados para a instrução SELTAP.

TRST

Esta instrução aplica um impulso ao nível lógico baixo '0' na linha TRST do TAP seleccionado, permitindo a inicialização da lógica de teste de todos os componentes que disponham de um pino /TRST ligado a esta saída do PRODEP. A inicialização de toda a infraestrutura de teste do sistema sob depuração, por esta via, implica que todos os componentes compatíveis com a norma IEEE 1149.1 disponham deste pino opcional (caso contrário será necessário aplicar uma sequência de cinco impulsos em TCK, mantendo a linha TMS ao nível lógico alto '1'). A Figura 6-14 ilustra o diagrama de transição de estados para esta instrução.

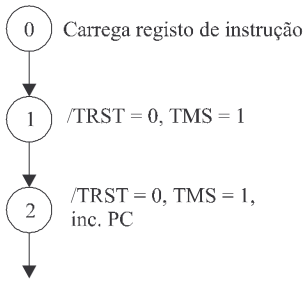


Figura 6-14: Diagrama de transição de estados para a instrução TRST.

TMS0, TMS1

Estas instruções aplicam um impulso na linha TCK, com a linha TMS activa ao nível lógico especificado, permitindo comandar as transições dos controladores do TAP dos componentes inseridos na cadeia activa. A Figura 6-15 ilustra o diagrama de transição de estados para esta instrução.

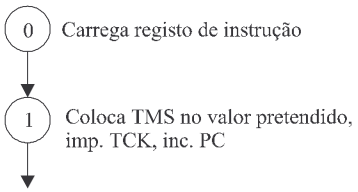


Figura 6-15: Diagrama de transição de estados para a instrução TMS.

NSHF

Esta instrução desloca através da linha TDO do TAP seleccionado, uma sequência de N bits armazenada nas posições de memória de programa que se seguem ao respectivo código de instrução. N representa o conteúdo do contador interno de dezasseis bits, que deverá ter sido previamente carregado com o número de bits a deslocar. A Figura 6-16 ilustra o diagrama de transição de estados para esta instrução.

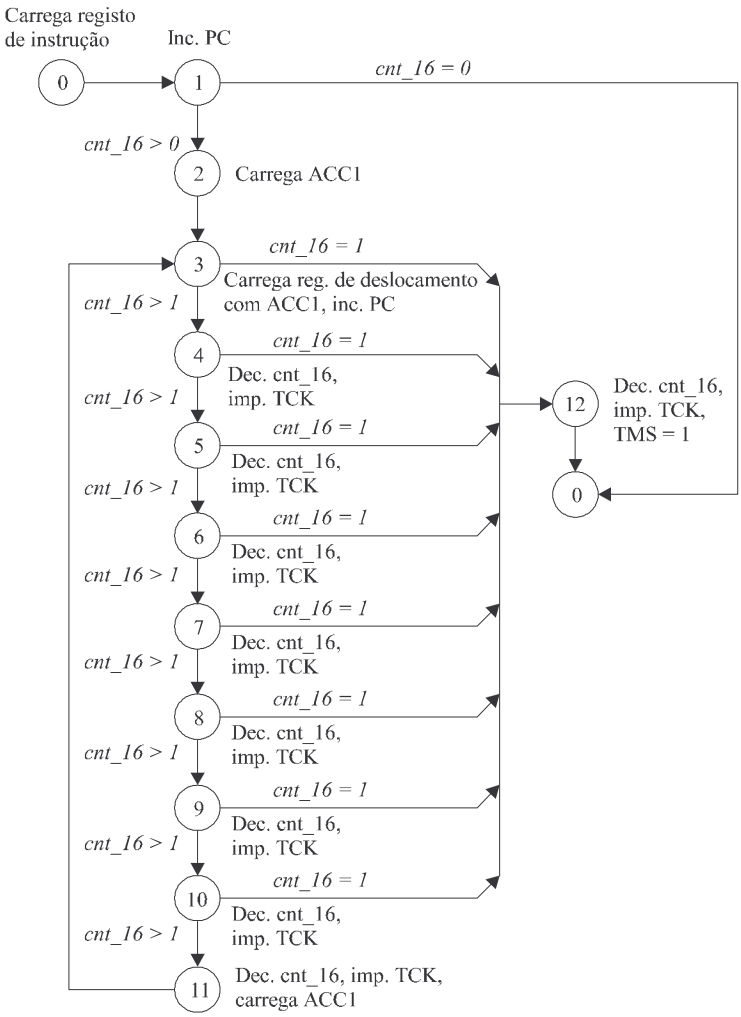


Figura 6-16: Diagrama de transição de estados para a instrução NSHF.

A linha TMS é mantida ao nível lógico baixo '0' durante os (N-1) impulsos aplicados em TCK, de modo a manter os controladores do TAP dos componentes inseridos na cadeia activa, no estado *Shift-DR* (*Shift-IR*). O último ciclo de TCK, aplicado durante o último estado desta instrução (estado 12), provoca uma transição dos controladores do TAP para o estado *Exit1-DR* (*Exit1-IR*), em virtude do valor lógico alto '1' especificado para a linha TMS. As actualizações dos valores a deslocar para a cadeia activa ocorrem nas transições descendentes do sinal TCK. Os dados referentes à sequência a deslocar são armazenados como se segue: o primeiro byte a deslocar encontra-se armazenado imediatamente após a posição de memória ocupada pelo código da instrução NSHF, e o primeiro bit a deslocar corresponde ao bit mais à direita, ou menos significativo (CLT_D[0]), desse byte.

As operações de deslocamento e de leitura de novos valores a deslocar são efectuadas em paralelo, de forma a ser apenas necessário um único ciclo de relógio, correspondente a um estado, para carregar o registo de deslocamento (estado 3), permitindo assim um atraso de apenas um ciclo de relógio por cada oito bits a deslocar (são necessários nove ciclos de relógio – CLK – por cada oito ciclos de TCK).

NSHFCP

Esta instrução desloca através da linha TDO do TAP seleccionado, uma sequência de N bits armazenada nas posições de memória de programa que se seguem ao respectivo código de instrução. N representa o conteúdo do contador interno de dezasseis bits, que deverá ter sido previamente carregado com o tamanho da sequência de bits a deslocar. O armazenamento desta sequência deve no entanto ser alternado com o de duas outras sequências, byte a byte, destinadas respectivamente à indicação dos valores esperados na correspondente entrada série (TDI) e das máscaras de comparação a utilizar. A primeira comparação de um valor (bit) capturado em TDI que difira do seu valor esperado, e cuja máscara permita a comparação, actuará o registo de memorização de erro, fazendo com que a respectiva saída da unidade CLT, que indica a ocorrência de erro (ERROR), passe ao estado activo (no nível lógico alto '1'). A instrução NSHFCP termina a sua execução, em qualquer dos casos, no final do deslocamento da totalidade da sequência, embora o estado do registo de memorização de erro possa ser posteriormente testado por uma operação de salto condicional. A Figura 6-17 ilustra o diagrama de transição de estados para esta instrução.

A linha TMS é mantida ao nível lógico baixo '0' durante os (N-1) impulsos aplicados em TCK, de modo a manter os controladores do TAP dos componentes inseridos na cadeia activa, no estado *Shift-DR* (*Shift-IR*). O último ciclo de TCK, aplicado durante o último estado desta instrução (estado 17), provoca uma transição dos controladores do TAP para o estado *Exit1-DR*

(*Exit1-IR*), em virtude do valor lógico alto '1' especificado para a linha TMS. As actualizações dos valores a deslocar para a cadeia activa ocorrem nas transições descendentes do sinal TCK. Os dados referentes à sequência a deslocar são armazenados de forma idêntica à especificada para a instrução NSHF, sendo intercalados, byte a byte, com os dados referentes aos valores esperados e máscaras de comparação. Durante a execução desta instrução, o sinal interno *Deser* é activado por forma a permitir o armazenamento na memória externa (tipo FIFO) dos valores capturados na entrada série (TDI) activa da unidade CLT.

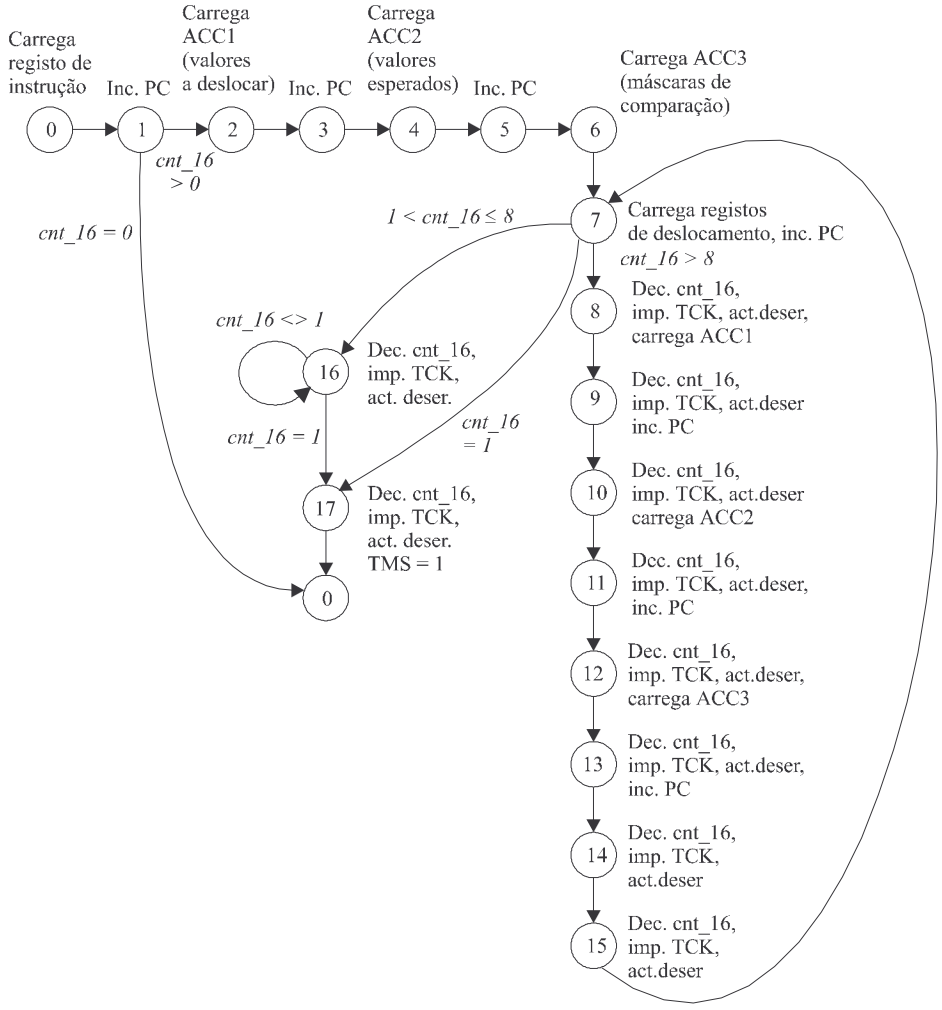


Figura 6-17: Diagrama de transição de estados para a instrução NSHFCP.

Também na instrução NSHFCP, a exemplo do que sucedia já na instrução NSHF, a execução concorrente da operação de deslocamento, e da leitura dos novos valores a deslocar, valores esperados e máscaras, permite ter um atraso de apenas um ciclo de relógio por cada oito bits a deslocar (são necessários nove ciclos de relógio – CLK – por cada oito ciclos de TCK).

NTCK

Esta instrução destina-se a permitir a execução das funções de auto-teste em componentes compatíveis com a norma IEEE 1149.1, que disponham da instrução opcional *RUNBIST*, ou outra similar, tendo por resultado a aplicação de N impulsos em TCK, enquanto a linha TMS do TAP seleccionado é mantida ao nível lógico baixo (de modo que os controladores do TAP dos componentes seleccionados se mantenham no estado *Run-Test/Idle*). N representa o conteúdo do contador interno de 24 bits, que deverá ter sido previamente carregado com o número de impulsos de relógio necessários para executar as funções de auto-teste. A Figura 6-18 ilustra o diagrama de transição de estados para esta instrução.

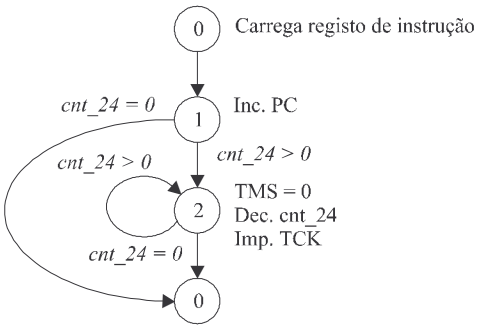


Figura 6-18: Diagrama de transição de estados para a instrução NTCK.

De acordo com o diagrama de transição de estados anterior, a frequência de TCK será igual à frequência do relógio principal (CLK) da unidade CLT (ou seja, do PRODEP).

Instruções para suporte de operações de depuração

Apresenta-se em seguida a caracterização individual das instruções descritas na Tabela 6-2, que permitem suportar as operações de depuração efectuadas através (ou com a ajuda) da unidade CLT.

STCK

Esta instrução destina-se a permitir a aplicação de impulsos de relógio de teste (TCK) enquanto a entrada (da unidade CLT) do canal de sincronismo que interliga as duas unidades (canal A) se mantiver no nível lógico alto '1'. Repare-se que neste caso não se sabe à partida qual o número de impulsos a aplicar em TCK. A Figura 6-19 ilustra o diagrama de transição de estados para esta instrução.

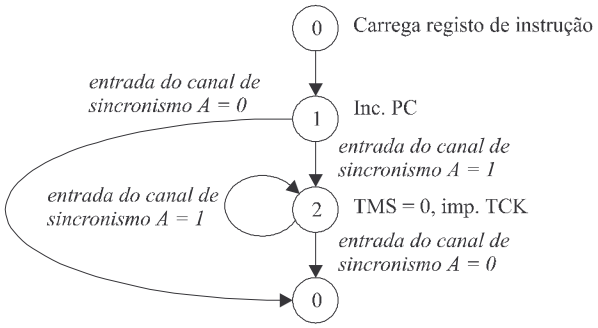


Figura 6-19: Diagrama de transição de estados para a instrução STCK.

STMPB0, STMPB1

Estas instruções destinam-se a permitir a selecção de um dos registos de retenção temporária que existem na unidade CLT. A Figura 6-20 ilustra o diagrama de transição de estados para esta instrução.

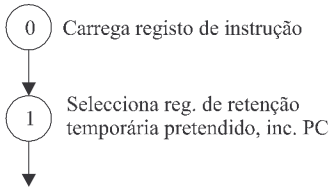


Figura 6-20: Diagrama de transição de estados para a instrução STMPB.

NCSHF

Esta instrução desloca a sequência de N bits capturados (bit a bit) na linha TDI do TAP seleccionado, para a linha TDO do mesmo TAP. Internamente, a unidade CLT armazena os valores capturados no registo de retenção temporária seleccionado. N representa o conteúdo do contador interno de dezasseis bits, que deverá ter sido previamente carregado com o número de bits a deslocar. Se o valor N igualar o comprimento da cadeia activa, no final da execução desta instrução o conteúdo da cadeia activa terá sido deslocado circularmente através dos pinos TDI – TDO do PRODEP, e memorizado num dos seus registos de retenção temporária. A Figura 6-21 ilustra o diagrama de transição de estados para esta instrução.

A linha TMS é mantida ao nível lógico baixo '0' durante os (N-1) impulsos aplicados em TCK, de modo a manter os controladores do TAP dos componentes inseridos na cadeia activa, no estado *Shift-DR* (*Shift-IR*). O último ciclo de TCK, aplicado durante o último estado desta instrução (estado 11), provoca uma transição dos controladores deoTAP para o estado *Exit1-DR* (*Exit1-IR*), em virtude do valor lógico alto '1' especificado para a linha TMS.

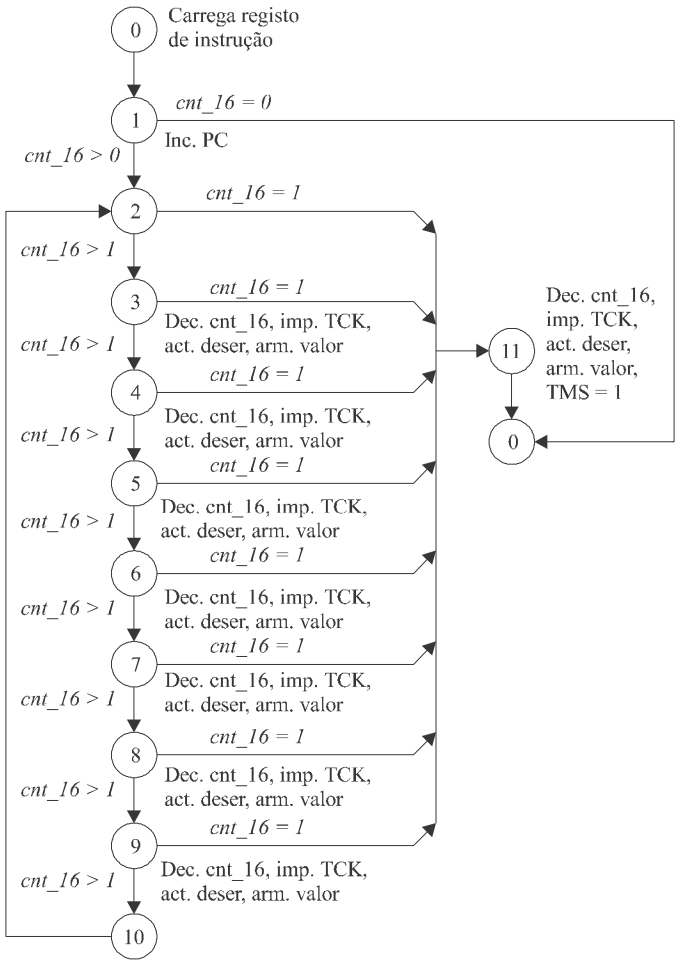


Figura 6-21: Diagrama de transição de estados para a instrução NCSHF.

NCSHFCP

Esta instrução desloca a sequência de N bits capturados (bit a bit) na linha TDI do TAP seleccionado, para a linha TDO do mesmo TAP, comparando em simultâneo, através de uma máscara, cada valor lido com um valor esperado. As sequências de valores esperados e das máscaras de comparação a utilizar, devem estar armazenadas alternadamente (byte a byte) nas posições de memória seguintes ao código de instrução. A primeira comparação de um valor (bit) capturado em TDI que difira do seu valor esperado, e cuja máscara permita a comparação, actuará o registo de memorização de erro, fazendo com que a respectiva saída da unidade CLT, que indica a ocorrência de erro (ERROR), passe ao estado activo no nível lógico alto '1'. A instrução NCSHFCP termina a sua execução, em qualquer dos casos, no final do deslocamento da totalidade da sequência, podendo o estado do registo de memorização de erro ser posteriormente testado por uma operação de salto condicional. Internamente, a unidade CLT armazena os valores capturados no registo de retenção temporária seleccionado. N representa o conteúdo

do contador interno de dezasseis bits, que deverá ter sido previamente carregado com o número de bits a deslocar. Se o valor N igualar o comprimento da cadeia activa, no final da execução desta instrução o conteúdo da cadeia activa terá sido deslocado circularmente através dos pinos TDI – TDO do PRODEP, comparado com uma sequência esperada, e memorizado num dos seus registos de retenção temporária. A Figura 6-22 ilustra o diagrama de transição de estados para esta instrução.

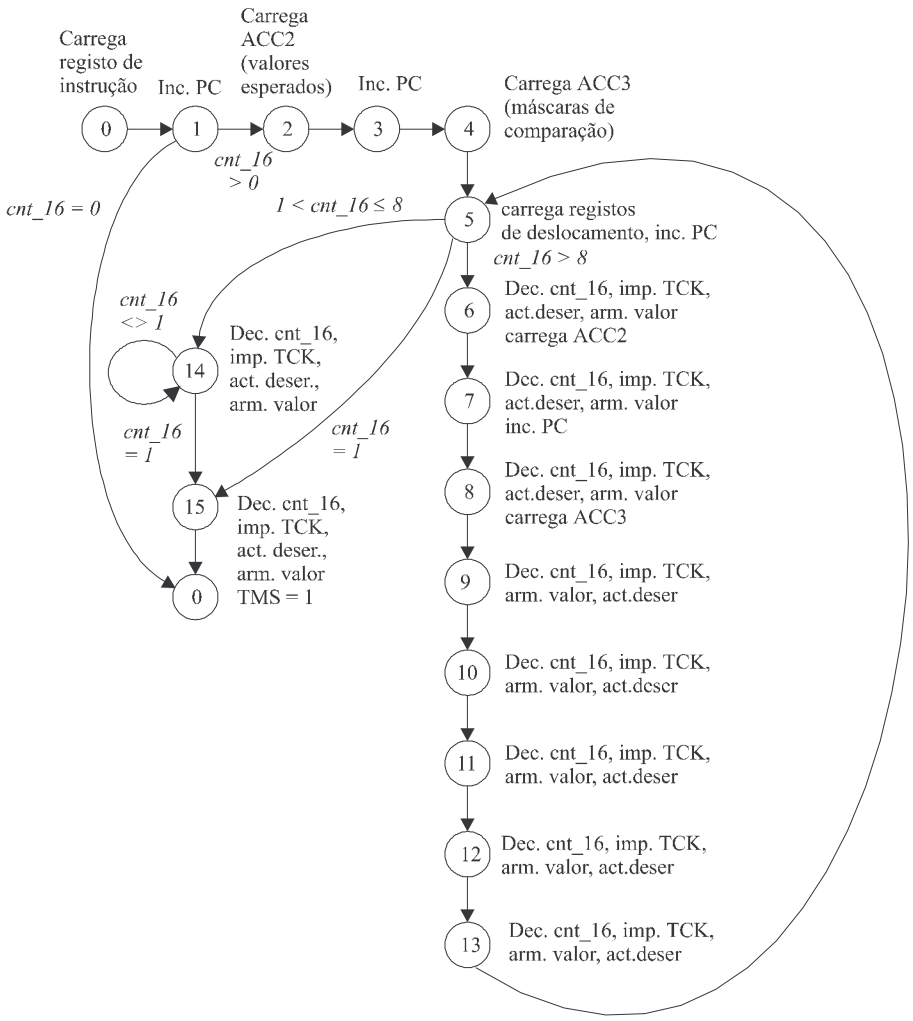


Figura 6-22: Diagrama de transição de estados para a instrução NCSHFCP.

A linha TMS é mantida ao nível lógico baixo ‘0’ durante os (N-1) impulsos aplicados em TCK, de modo a manter os controladores do TAP dos componentes inseridos na cadeia activa, no estado *Shift-DR* (*Shift-IR*). O último ciclo de TCK, aplicado durante o último estado (estado 15) desta instrução, provoca uma transição dos controladores do TAP para o estado *Exit1-DR* (*Exit1-IR*), em virtude do valor lógico alto ‘1’ especificado para a linha TMS.

NSHFB2C

Esta instrução desloca a sequência de N bits armazenados internamente no registo de retenção temporária seleccionado, para a entrada série (TDI) da cadeia activa. N representa o conteúdo do contador interno de dezasseis bits, que deverá ter sido previamente carregado com o número de bits a deslocar. Se o valor N igualar o comprimento da cadeia activa, no final da execução desta instrução, o seu conteúdo será idêntico ao anteriormente memorizado através da execução de uma instrução NCSHF, ou NCSHFCP, permitindo desta forma repô-la num estado anterior desconhecido. A Figura 6-23 ilustra o diagrama de transição de estados para esta instrução.

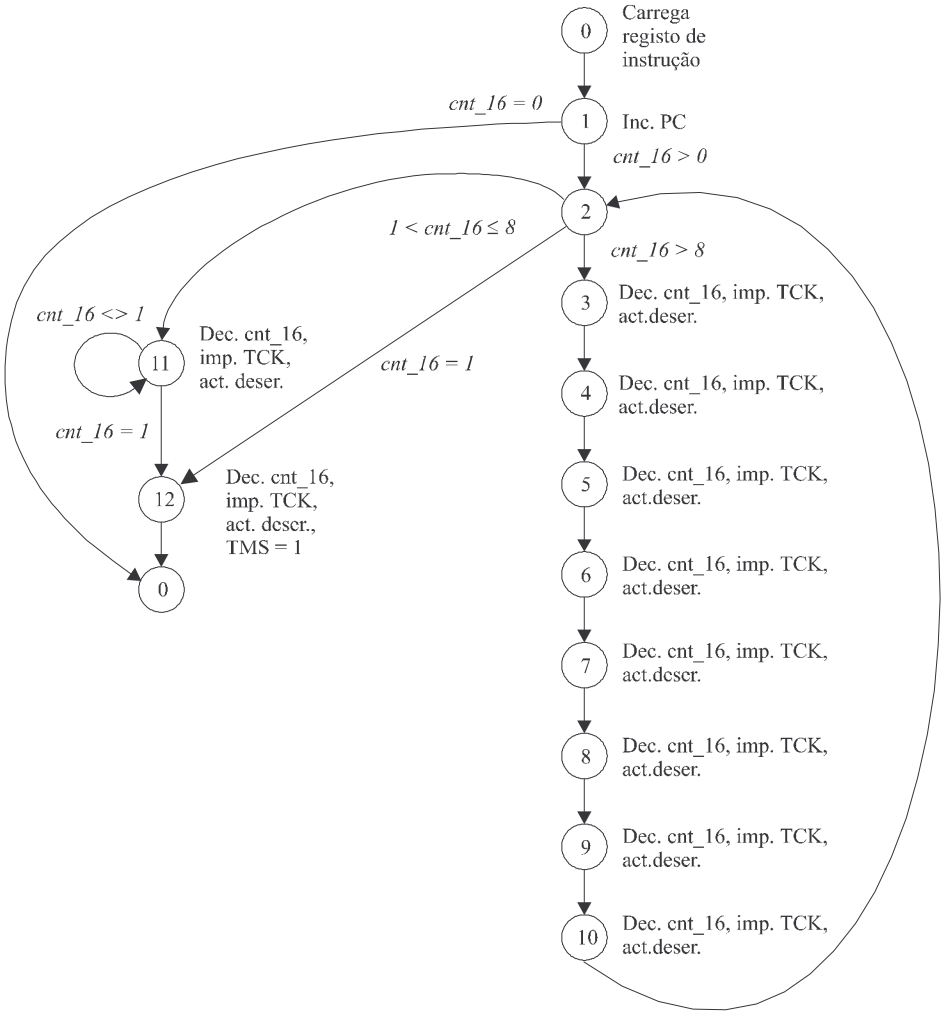


Figura 6-23: Diagrama de transição de estados para a instrução NSHFB2C.

A linha TMS é mantida ao nível lógico baixo '0' durante os (N-1) impulsos aplicados em TCK, de modo a manter os controladores do TAP dos componentes inseridos na cadeia activa, no estado *Shift-DR* (*Shift-IR*). O último ciclo de TCK, aplicado durante o último estado desta instrução (estado 12), provoca uma transição dos controladores do TAP para o estado *Exit1-DR* (*Exit1-IR*), em virtude do valor lógico alto '1' especificado para a linha TMS.

NSHFCPB2C

Esta instrução desloca a sequência de N bits armazenados internamente no registo de retenção temporária seleccionado, para a entrada série (TDI) da cadeia activa, comparando em simultâneo, através de uma máscara, o valor lido na saída séria (TDO) dessa cadeia com um valor esperado. A Figura 6-24 ilustra o diagrama de transição de estados para esta instrução.

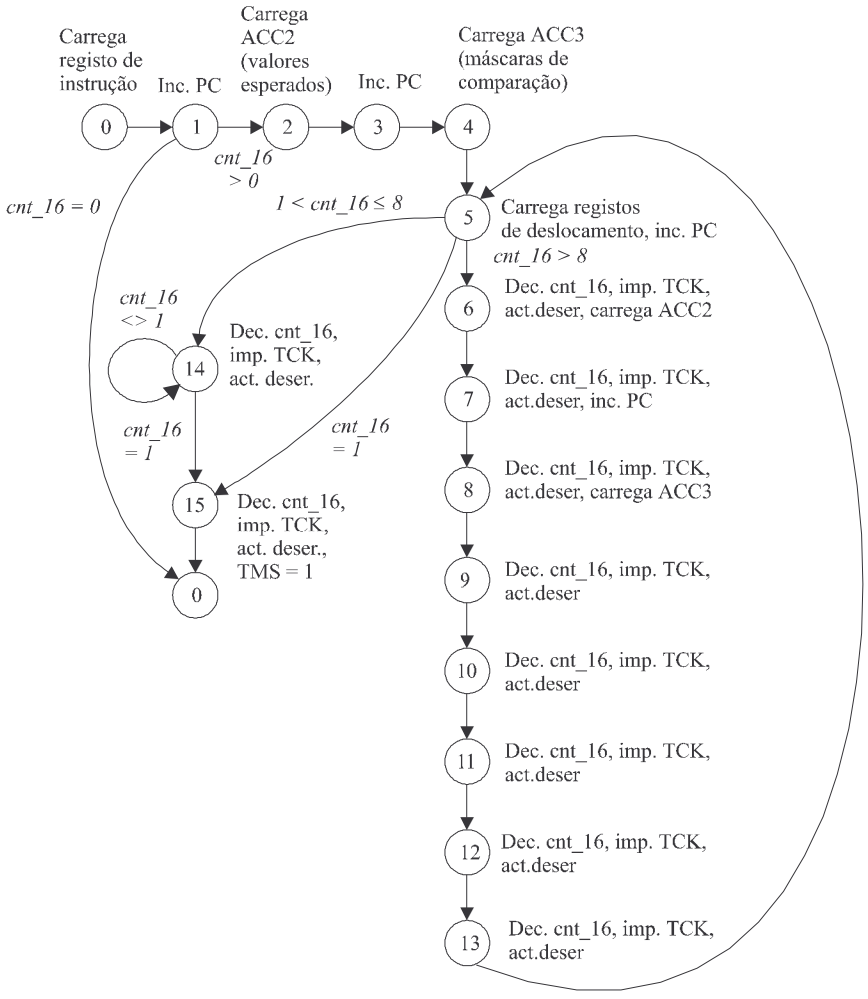


Figura 6-24: Diagrama de transição de estados para a instrução NSHFCPB2C.

As sequências de valores esperados e das máscaras de comparação a utilizar, devem estar armazenadas alternadamente (byte a byte) nas posições de memória seguintes ao código de instrução. A primeira comparação de um valor (bit) capturado em TDI, que difira do seu valor esperado, e cuja máscara permita a comparação, actuará o registo de memorização de erro, fazendo com que a respectiva saída da unidade CLT, que indica a ocorrência de erro (ERROR), passe ao estado activo no nível lógico alto '1'. A instrução NSHFCPB2C termina a sua execução, em qualquer dos casos, no final do deslocamento da totalidade da sequência, podendo o estado do registo de memorização de erro ser posteriormente testado por uma operação de salto condicional. N representa o conteúdo do contador interno de dezasseis bits, que deverá ter sido previamente carregado com o número de bits a deslocar. Se o valor N igualar o comprimento da cadeia activa, no final da execução desta instrução, o seu conteúdo será idêntico ao anteriormente memorizado através da execução de uma instrução NCSHF, ou NCSHFCP, permitindo desta forma repô-la num estado anterior desconhecido.

Instruções para controlo dos recursos internos, canais de sincronismo e fluxo do programa

Apresenta-se em seguida a caracterização individual das instruções descritas na Tabela 6-3, que permitem controlar os recursos internos, os canais de sincronismo e o fluxo de execução do programa da unidade CLT.

LD C16, N

Esta instrução carrega no contador interno de dezasseis bits o conteúdo armazenado nas duas posições de memória seguintes à que contém o código de instrução. A primeira destas posições contém os oito bits mais significativos do valor a carregar e a segunda posição contém os oito bits menos significativos. A Figura 6-25 ilustra o diagrama de transição de estados para esta instrução.

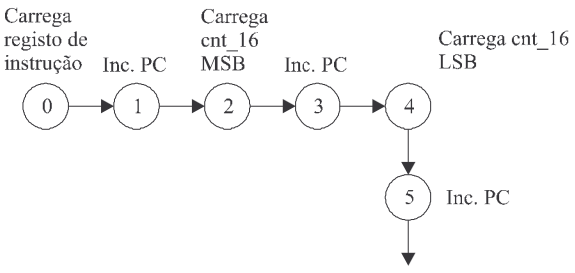


Figura 6-25: Diagrama de transição de estados para a instrução LD C16, N.

LD C24, N

Esta instrução carrega no contador interno de 24 bits o conteúdo armazenado nas três posições de memória seguintes à que contém o código de instrução. A primeira destas posições contém os oito bits mais significativos do valor a carregar, a segunda posição contém os oito bits intermédios, e a terceira posição contém os oito bits menos significativos. A Figura 6-26 ilustra o diagrama de transição de estados para esta instrução.

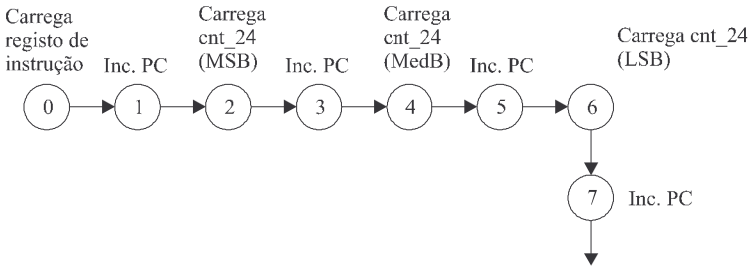


Figura 6-26: Diagrama de transição de estados para a instrução LD C24, N.

SSA0, SSA1, SSB0, SSB1

Estas instruções permitem controlar o nível lógico (‘0’, ‘1’) presente nas saídas dos canais de sincronismo (A, B) da unidade CLT. Estas instruções são usadas em conjunto com as instruções WS (a descrever em seguida) para implementar um protocolo de sincronismo assíncrono com equipamentos de teste externos ou para sincronizar a execução de grupos de instruções com a unidade CLF. A Figura 6-27 ilustra o diagrama de transição de estados desta instrução.

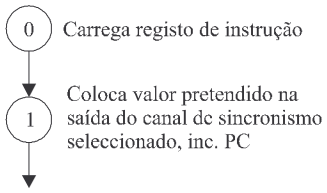


Figura 6-27: Diagrama de transição de estados para a instrução SS.

WSA0, WSA1, WSB0, WSB1

Estas instruções retêm a unidade CLT no mesmo estado por um número indefinido de ciclos de relógio, até que a entrada do canal de sincronismo seleccionado (A, B) se encontre no valor lógico especificado (‘0’, ‘1’). Estas instruções são usadas em conjunto com as instruções SS para implementar um protocolo de sincronismo assíncrono com equipamentos de teste ex-

ternos ou para sincronizar a execução de grupos de instruções com a unidade CLF. A Figura 6-28 ilustra o diagrama de transição de estados para esta instrução.

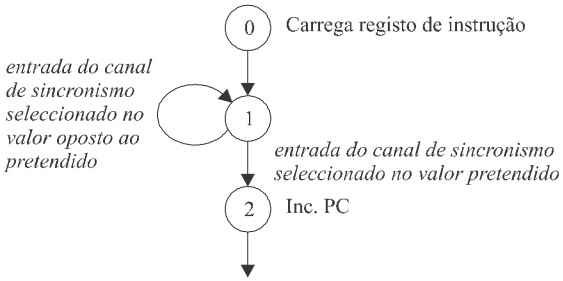


Figura 6-28: Diagrama de transição de estados para a instrução WS.

JPE Endereço

Esta instrução provoca um salto para o endereço especificado, se o registo de memorização de erro estiver no nível lógico alto '1', em consequência da detecção anterior de um erro. Se a saída deste registo se encontrar no nível lógico baixo '0', o próximo código de instrução será lido na primeira posição de memória que se segue ao operando desta instrução, que para o caso de um barramento de endereços com vinte linhas corresponde a três bytes. A Figura 6-29 ilustra o diagrama de transição de estados para esta instrução.

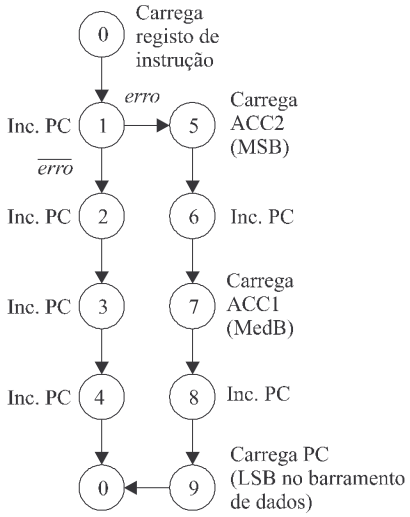


Figura 6-29: Diagrama de transição de estados para a instrução JPE Endereço.

JPNE Endereço

Esta instrução provoca um salto para o endereço especificado, se o registo de memorização de erro estiver no nível lógico baixo '0', em consequência de não ter ainda sido detectado nenhum erro. Se a saída deste registo se encontrar no nível lógico alto '1', o próximo código de instrução será lido na primeira posição de memória que se segue ao operando desta instrução, que para o caso de um barramento de endereços com vinte linhas corresponde a três bytes. A Figura 6-30 ilustra o diagrama de transição de estados para esta instrução.

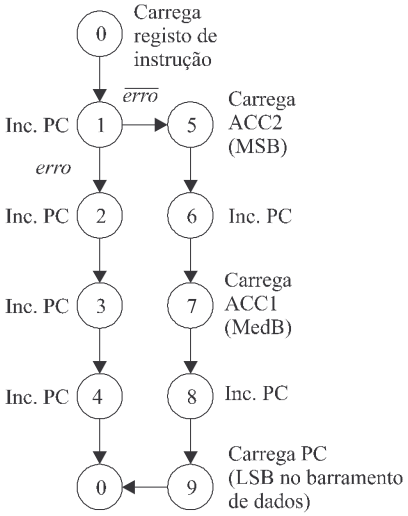


Figura 6-30: Diagrama de transição de estados para a instrução JPNE Endereço.

HALT

Esta instrução é usada para terminar a execução do programa. A Figura 6-31 ilustra o diagrama de transição de estados para esta instrução.

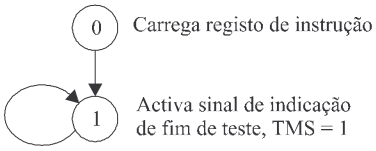


Figura 6-31: Diagrama de transição de estados para a instrução HALT.

6.3 A unidade de controlo da lógica funcional

Esta secção descreve a constituição interna, o conjunto de instruções e cada uma das instruções, individualmente, da unidade CLF do PRODEP (ver Figura 6-1). A Figura 6-32 ilustra o diagrama de blocos da sua arquitectura interna onde se destacam os seguintes elementos:

- Bloco de controlo e descodificação de instruções.
- Registo de programa (PC, *Program Counter*).
- Interface com memória FIFO externa.
- Contadores (CNT, *Counter*).
- Registos de uso genérico.
- Bloco de entradas e saídas de uso genérico.
- Circuito de geração do relógio do sistema.
- Bloco das saídas dos canais de sincronismo.

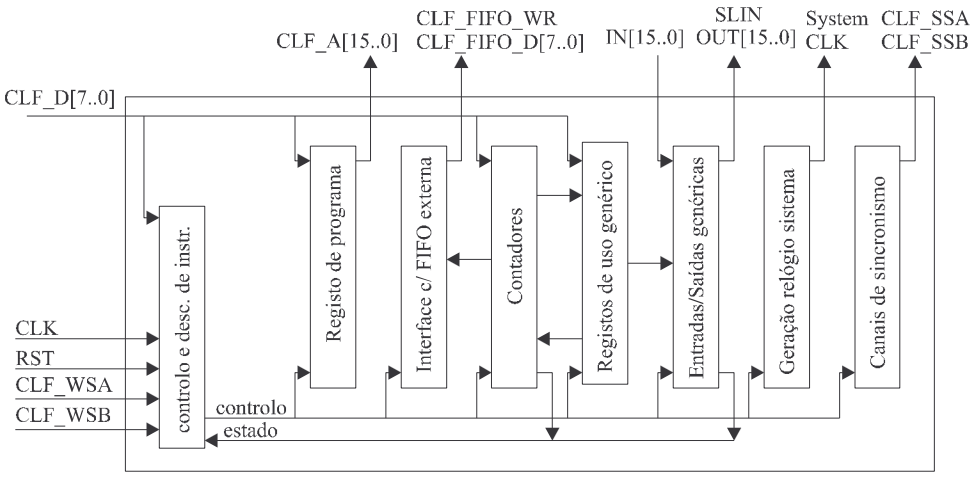


Figura 6-32: Diagrama de blocos da arquitectura interna da unidade CLF.

A Figura 6-32 ilustra ainda, a um nível mais pormenorizado, a constituição dos pinos de entrada e saída da unidade CLF, destacando-se um barramento de dados com oito bits (CLF_D[7..0]), um barramento de endereços com dezasseis bits (CLF_A[15..0]), que permite endereçar memórias com até 64 Kbytes, os sinais de interface com uma memória externa tipo FIFO com entrada e saída de dados independentes, com oito bits (CLF_FIFO_D[7..0] e CLF_FIFO_WR), um conjunto de dezasseis saídas para usos genéricos (OUT[15..0]), um conjunto de dezasseis entradas para usos genéricos (IN[15..0]), uma saída (SLIN, *Selected Input*) que contém o valor actualizado de uma qualquer das dezasseis entradas anteriores, o relógio do sistema sob depuração (System_CLK), e os sinais de entrada e saída dos dois canais de sincronismo A e B (CLF_WSA, CLF_WSB, CLF_SSA, CLF_SSB).

6.3.1 Blocos principais

Cada um dos blocos identificados na Figura 6-32 é em seguida descrito, ilustrando-se o diagrama de topo com a identificação dos sinais de entrada e saída, número mínimo de FF necessários, e referindo-se ainda em breves palavras qual a funcionalidade implementada.

Unidade de controlo e descodificação de instruções

A Figura 6-33 ilustra o diagrama interno deste bloco, onde se distinguem três componentes principais:

- O registo de instrução, destinado a receber o código de cada instrução a executar, que é constituído por cinco FF, permitindo codificar 32 (2^5) instruções.
- Uma MEF responsável por gerar a necessária sequência de estados, de acordo com o código presente no registo de instrução e com os sinais de estado provenientes do exterior ou de alguns dos blocos da unidade CLF. Esta MEF é constituída por três FF, que indicam o estado actual, e por um bloco combinatório que determina qual o estado seguinte.
- Um circuito combinatório responsável pela geração dos sinais de controlo para todos os blocos da unidade CLF, de acordo com o estado actual e com o código de instrução presente.

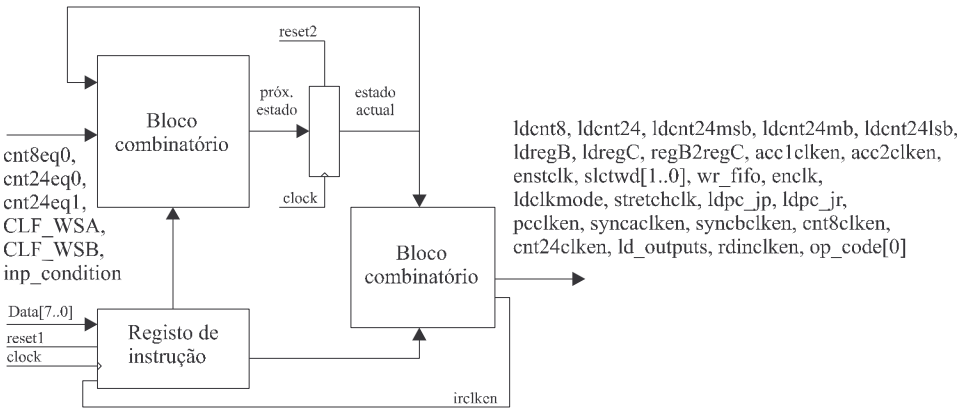


Figura 6-33: Diagrama interno do bloco de controlo e descodificação de instruções da unidade CLF.

Este bloco é igualmente responsável pela leitura do código da instrução seguinte, que tem lugar sempre que a MEF retorna ao estado inicial, a que se atribui por convenção a designação de estado zero (0).

Registo de programa

A Figura 6-34 ilustra o diagrama de topo deste bloco, responsável pelo controlo do barramento de endereços, constituindo o único apontador para a memória com o programa da unidade CLF, pelo que todos os ciclos de acesso à memória (leitura de dados, ou de códigos de instrução) devem posteriormente incrementar este apontador, de modo a preparar o próximo acesso.

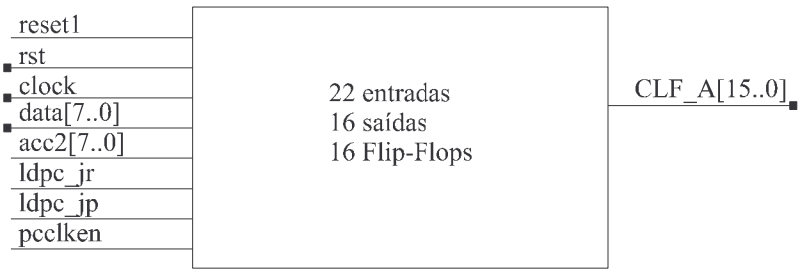


Figura 6-34: Diagrama de topo do registo de programa da unidade CLF.

O registo de programa é constituído por um contador síncrono de dezasseis bits, com possibilidade de carga paralela e soma do conteúdo com um valor (oito bits) em complemento para dois, para suportar instruções de salto absoluto e relativo, respectivamente. Uma vez que a largura do barramento de dados é de oito bits, torna-se necessária a existência de um acumulador com oito bits, de modo a permitir que a carga dos dois andares do registo de programa tenha lugar em paralelo – andar menos significativo (oito bits) do registo de programa carregado com o conteúdo do barramento de dados e andar mais significativo (oito bits) carregado com o conteúdo do acumulador, o qual faz parte do bloco de registos de uso genérico. Para suportar saltos relativos existe um bloco combinacional que soma o conteúdo do registo do programa com um valor binário codificado em complemento para dois (o que permite saltos entre –127 e 128). Aos oito bits mais significativos do operando da soma atribui-se o valor correspondente ao bit mais significativo do salto a efectuar (presente em D[7]).

As saídas do barramento de endereços são colocadas num estado de alta-impedância quando a linha de inicialização externa (RST) se encontra activa ao nível lógico baixo ‘0’.

Bloco de interface com uma FIFO externa

A Figura 6-35 ilustra o diagrama de topo deste bloco, responsável pelo armazenamento numa memória externa do valor actual do contador interno de 24 bits, que é utilizado na implementação do algoritmo de alargamento de um impulso de relógio do sistema (System_CLK), a descrever adiante. Com o objectivo de facilitar o desenvolvimento deste bloco optou-se por uma memória externa do tipo FIFO com barramentos de entrada e saída de dados independentes.

Assim, evitou-se a necessidade de implementar um apontador de memória (repare-se que a escrita é sequencial, pelo que não existem inconvenientes nesta escolha), e um barramento de dados com controlo de estado, reduzindo-se a interface de controlo a uma única linha de indicação de escrita (CLF_FIFO_WR). Dado que se optou ainda por uma memória externa com um barramento de dados com oito bits de largura, o conteúdo do contador de 24 bits é escrito em três fases, que correspondem a cada um dos três bytes que o formam.

Figura 6-35: Diagrama de topo da interface com a FIFO externa da unidade CLF.

Contadores

A Figura 6-36 ilustra o diagrama de topo deste bloco, constituído por dois contadores descendentes síncronos, com carga paralela, de 24 e oito bits. Este bloco proporciona à unidade de controlo e descodificação vários sinais que indicam qual o seu estado interno, nomeadamente quando o conteúdo de ambos os contadores é igual a zero (fim de contagem) e quando o conteúdo do contador de 24 bits é igual a um.

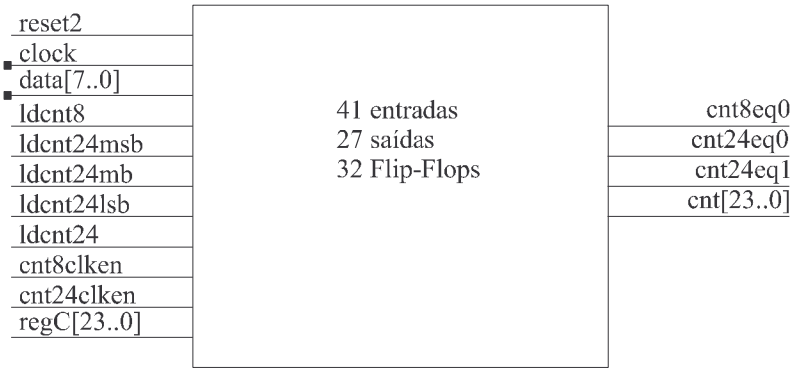


Figura 6-36: Diagrama de topo do bloco dos contadores da unidade CLF.

O contador interno de oito bits é decrementado pela unidade de controlo e descodificação de instruções, durante a execução de instruções que requerem um determinado número de impulsos de relógio de sistema (System_CLK). O contador interno de 24 bits é utilizado na im-

plementação de ciclos numerados (do tipo de I = 1 até N, faça …), e na implementação do algoritmo de alargamento de um impulso de relógio, em que o seu conteúdo indica qual o impulso de relógio que irá ser 'esticado' (a apresentar adiante, na subsecção 6.3.2). Em virtude de algumas instruções necessitarem dos dois contadores, não se implementou a partilha de andares entre estes. A carga dos contadores é efectuada sem auxílio de acumuladores, uma vez que os possíveis estados intermédios não influenciam o comportamento pretendido para este bloco.

Registos de uso genérico

A Figura 6-37 ilustra o diagrama de topo deste bloco, constituído por três acumuladores de 24 bits cada e dois acumuladores de oito bits cada. Os primeiros servem de registos de retenção temporária do conteúdo inicial e actual e da diferença entre estes dois valores, do contador de 24 bits, para efeitos de implementação do algoritmo de alargamento de um impulso do relógio do sistema. Os segundos servem de registos de retenção para algumas operações internas, nomeadamente a carga do registo de programa em paralelo (em que valores intermédios poderiam causar problemas na leitura da posição de memória seguinte) e a memorização de qual a entrada, ou saída, genérica seleccionada.

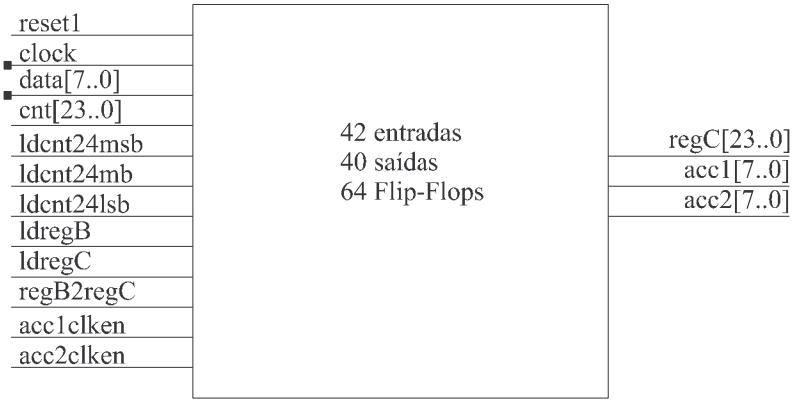


Figura 6-37: Diagrama de topo do bloco dos registos de uso genérico da unidade CLF.

Bloco de Entradas e Saídas de uso genérico

A Figura 6-38 ilustra o diagrama de topo deste bloco, que liga a um conjunto de dezasseis pinos de entrada e a um conjunto de dezasseis pinos de saída, para uso genérico. Este bloco possui ainda um pino de saída que permite visualizar o valor actual de uma qualquer das entradas. Os pinos de entrada podem ser utilizados para operações de salto condicional ou para a detecção de condições externas, nomeadamente nos casos em que se encontrem ligados a saídas de componentes que suportem a instrução opcional *COMP* (neste caso, o pino SCD). Os pinos de

saída permitem controlar directamente linhas ou sinais do sistema sob depuração, que sirvam directamente ou indirectamente objectivos de teste ou depuração, como é o caso de pinos de controlo de compatibilidade com a norma IEEE 1149.1 [IEEE93, pág. 3-7].

Figura 6-38: Diagrama de topo do bloco de entradas e saídas de uso genérico da unidade CLF.

Circuito de geração do relógio do sistema

A Figura 6-39 ilustra os diagramas interno e de topo deste bloco, responsável pelo controlo do relógio do sistema sob depuração. A funcionalidade deste bloco permite a aplicação de:

- Um único impulso.
- Um trem de impulsos com a mesma frequência do relógio que alimenta o PRODEP (em que o número aplicado é determinado pelo valor carregado no contador de oito bits).
- A aplicação contínua de impulsos (com a mesma frequência anterior) enquanto um registo interno de um bit se encontrar carregado com o valor lógico alto '1'.
- A aplicação de um impulso estendido (i.e., com a largura de um ciclo completo do relógio principal).
- Nenhum impulso de relógio.

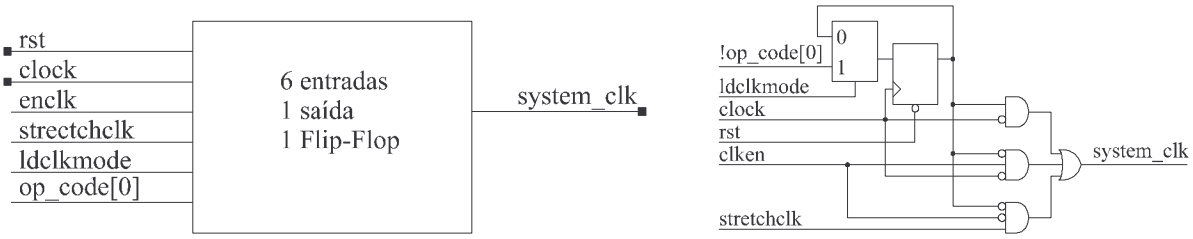


Figura 6-39: Diagramas de topo e interno do circuito de geração de relógio do sistema.

Um aspecto crucial no desenvolvimento deste bloco consiste na garantia de não existirem desvios significativos entre o relógio do sistema (System_CLK, da unidade CLF) e o relógio de teste (TCK0 ou TCK1, da unidade CLT), devido a percursos lógicos com extensões diferentes para um e outro. Repare-se que, neste caso, o relógio principal passa por duas portas lógicas

antes de alimentar o pino de relógio do sistema, pelo que na geração do relógio de teste se deverá garantir um percurso idêntico. Para efeitos de implementação em lógica programável, este aspecto depende ainda da constituição de cada macrocélula. A localização dos pinos não deve apresentar problemas, devendo utilizar-se o pino dedicado de entrada de relógio do componente programável, que garante tempos de atraso idênticos para todos os seus pinos de saída.

Canais de sincronismo

A Figura 6-40 ilustra o diagrama de topo deste bloco, responsável pelo controlo das saídas dos canais de sincronismo. Internamente, é constituído por dois registos de um bit que permitem memorizar o valor lógico especificado pela execução das instruções que controlam os pinos de saída destes canais. Os pinos de entrada correspondentes a cada canal encontram-se directamente ligados ao bloco de controlo e descodificação de instruções.

Figura 6-40: Diagrama de topo do bloco das saídas dos canais de sincronismo da unidade CLF.

6.3.2 Conjunto de instruções

Apresenta-se em seguida o conjunto de instruções suportadas pela unidade CLF, que permitem implementar as operações identificadas e posteriormente agrupadas no modelo simplificado de depuração, apresentado no capítulo 2 e ilustrado na Figura 2.14. A Tabela 6-4 descreve sucintamente as instruções destinadas ao suporte de operações de COV em pinos directamente acessíveis do sistema sob depuração, que se encontrem ligados ao PRODEP. A Tabela 6-5 descreve sucintamente as instruções destinadas ao suporte de operações passo-a-passo, através do controlo directo do relógio do sistema sob depuração. A Tabela 6-6 descreve sucintamente as instruções destinadas ao suporte de operações de pontos de paragem por condição, através da aplicação condicional de impulsos de relógio do sistema. A Tabela 6-7 descreve sucintamente as instruções destinadas ao suporte de operações de depuração em tempo real, nomeadamente para controlo da aplicação contínua de impulsos de relógio ao sistema sob depuração e para implementação do algoritmo de alargamento de um impulso do relógio do sistema. Finalmente, a Tabela 6-8 descreve sucintamente as instruções destinadas ao controlo dos recursos internos, canais de sincronismo e fluxo de execução do programa da unidade CLF.

Tabela 6-4: Instruções para suporte de operações de COV em pinos directamente acessíveis.

Instruções para suporte de operações de COV em pinos directamente acessíveis	
RSTOUT [i]	Coloca o valor lógico baixo '0' na saída genérica [i].
SETOUT [i]	Coloca o valor lógico alto '1' na saída genérica [i].
READ IN,[i]	Liga a entrada genérica [i] ao pino de saída denominado SLIN. Permite visualizar o valor lógico presente numa das dezasseis entradas genéricas.
JZ [i], Endereço JNZ [i], Endereço	Carrega o registo de programa com <endereço> (salto absoluto) se a entrada genérica seleccionada exibir o valor lógico pretendido.
WAIT_WHL_Z [i] WAIT_WHL_NZ [i]	Espera enquanto a entrada genérica seleccionada se encontrar no valor lógico pretendido.

Tabela 6-5: Instruções para suporte de operações passo-a-passo.

Instruções para suporte de operações passo-a-passo	
CLK	Aplica um impulso na linha do relógio do sistema.
CLK_N <N>	Aplica N impulsos na linha do relógio do sistema.

Tabela 6-6: Instruções para suporte de operações de pontos de paragem por condição.

Instruções para suporte de operações de pontos de paragem por condição	
CLK_WHL_Z [i]	Aplica impulsos na linha de relógio do sistema enquanto a entrada genérica [i] se mantiver no nível lógico baixo '0'.
CLK_WHL_NZ [i]	Aplica impulsos na linha de relógio do sistema enquanto a entrada genérica [i] se mantiver no nível lógico alto '1'.

Tabela 6-7: Instruções para suporte de operações de depuração em tempo real.

Instruções para suporte de operações de depuração em tempo real	
START_CLK	Inicia a aplicação contínua de impulsos na linha de relógio do sistema.
STOP_CLK	Pára a aplicação contínua de impulsos na linha de relógio do sistema.
CSTRETCH_N	Executa o algoritmo de alargamento de um impulso de relógio. O algoritmo é repetido *n* vezes, sendo alargado o *i-ésimo* impulso por cada trem de *n* impulsos aplicados, em que *i* denota a ordem de repetição do trem aplicado.

Tabela 6-8: Instruções para controlo dos recursos internos, canais de sincronismo e fluxo do programa.

Instruções para controlo dos recursos internos, canais de sincronismo e fluxo do programa	
LD C24, N	Carrega o contador interno de 24 bits com o número N.
STORE C24	Coloca o conteúdo do contador interno de 24 bits na memória FIFO externa.
SSA0, SSA1, SSB0, SSB1	Coloca o valor lógico pretendido na saída de sincronismo (A, B)
WSA0, WSA1, WSB0, WSB1	Espera até que a entrada de sincronismo (A ou B) se encontre no valor lógico pretendido.
DJNZ Endereço_relativo	Decrementa o contador interno de 24 bits, e se o seu conteúdo for diferente de zero adiciona <endereço_relativo> ao valor actual do registo de programa.
JP Endereço	Carrega o registo do programa com <endereço>.
HALT	Conclui a execução do programa.

Cada uma das instruções apresentadas nas tabelas anteriores é descrita em seguida de uma forma mais pormenorizada, nomeadamente através da apresentação do seu diagrama de transição de estados.

Instruções para suporte de operações de COV em pinos directamente acessíveis

Apresenta-se em seguida a caracterização individual das instruções da unidade CLF descritas na Tabela 6-4, que permitem suportar operações de COV em pinos directamente acessíveis do sistema sob depuração, que estejam ligados às entradas ou saídas de uso genérico do PRODEP.

SETOUT [i]

Esta instrução coloca ao nível lógico alto '1', a saída genérica indicada pelo operando que se segue, na memória do programa, ao código de instrução. A Figura 6-41 ilustra o diagrama de transição de estados para esta instrução.

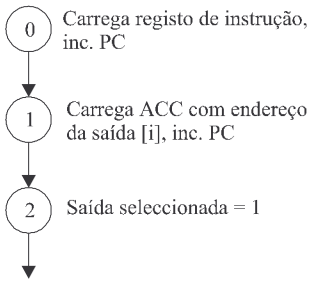


Figura 6-41: Diagrama de transição de estados para a instrução SETOUT [i].

A codificação das saídas é efectuada em binário, de forma que à saída identificada pelo número zero corresponde o operando zero, à saída um o operando um (palavra "00000001", em que o bit mais à direita constitui o bit menos significativo), e assim por diante até se atingir a saída dezasseis. Dado que o operando é constituído por oito bits, pode-se aumentar o número de pinos de saída da unidade CLF, até um máximo de 256 (2^8), sem necessidade de modificar o diagrama de transição de estados para esta instrução.

RSTOUT [i]

Esta instrução coloca ao nível lógico baixo '0', a saída genérica indicada pelo operando que se segue, na memória do programa, ao código da instrução. A Figura 6-42 ilustra o diagrama de transição de estados para esta instrução.

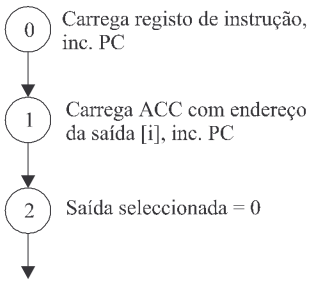


Figura 6-42: Diagrama de transição de estados para a instrução RSTOUT [i].

READ IN, [i]

Esta instrução liga, para efeitos de observação, a entrada genérica indicada pelo operando [i], ao pino de saída denominado SLIN, da unidade CLF. O operando, com oito bits, ocupa a posição de memória seguinte ao código de instrução. A Figura 6-43 ilustra o diagrama de transição de estados para esta instrução.

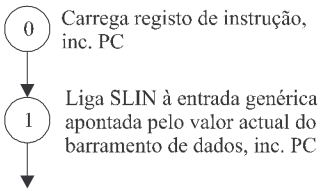


Figura 6-43: Diagrama de transição de estados para a instrução READ IN, [i].

O objectivo desta instrução, e do pino SLIN que lhe está associada, consiste em permitir visualizar (por recurso a multiplexagem controlada pelo código do programa) o valor lógico de

uma qualquer das dezasseis entradas, que assuma num determinado momento funções especiais, como por exemplo o controlo da aplicação condicional de impulsos de relógio do sistema.

JZ [i], Endereço

Esta instrução testa o valor lógico da entrada genérica indicada pelo primeiro operando, e efectua um salto para o endereço indicado pelo segundo operando, caso aquele valor seja igual a '0'. O primeiro operando, com oito bits, ocupa a posição de memória que se segue ao código de instrução, e o segundo operando, com dezasseis bits (indica um salto absoluto, pelo que o número de bits iguala o número de linhas do barramento de endereços), ocupa a segunda e terceira posições que se seguem. A Figura 6-44 ilustra o diagrama de transição de estados para esta instrução.

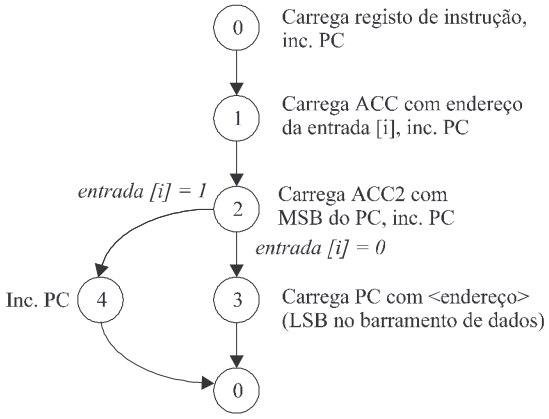


Figura 6-44: Diagrama de transição de estados para a instrução JZ [i], Endereço.

JNZ [i], Endereço

Esta instrução testa o valor lógico da entrada genérica indicada pelo primeiro operando, e efectua um salto para o endereço indicado pelo segundo operando, caso aquele valor seja igual a '1'. O primeiro operando, com oito bits, ocupa a posição de memória que se segue ao código de instrução, e o segundo operando, com dezasseis bits (indica um salto absoluto, pelo que o número de bits iguala o número de linhas do barramento de endereços), ocupa a segunda e terceira posições que se seguem. A Figura 6-45 ilustra o diagrama de transição de estados para esta instrução.

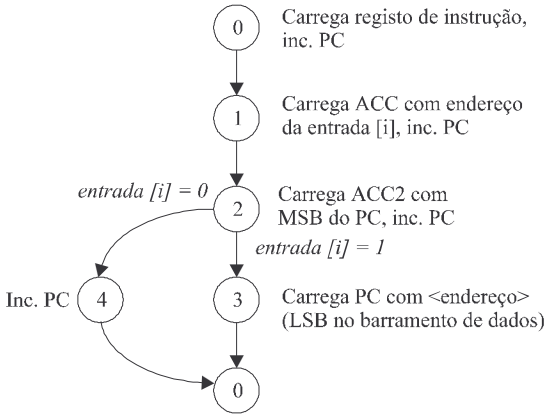


Figura 6-45: Diagrama de transição de estados para a instrução JNZ [i], Endereço.

WAIT_WHL_Z [i]

Esta instrução testa o valor lógico da entrada genérica indicada pelo operando e interrompe a execução do programa, caso esse valor seja igual a '0'. A execução do programa é retomada após a detecção do valor lógico oposto (neste caso '1'), na entrada seleccionada, na primeira transição ascendente do relógio que comanda a unidade de controlo e descodificação de instruções. O operando, com oito bits, ocupa a posição de memória que se segue ao código de instrução. A Figura 6-46 ilustra o diagrama de transição de estados para esta instrução.

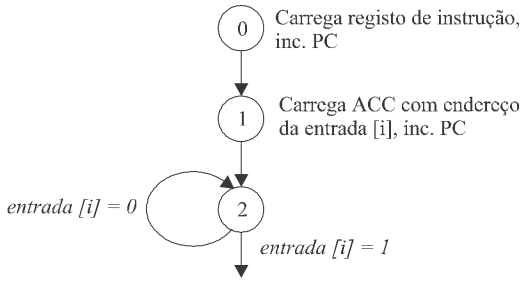


Figura 6-46: Diagrama de transição de estados para a instrução WAIT_WHL_Z [i].

WAIT_WHL_NZ [i]

Esta instrução testa o valor lógico da entrada genérica indicada pelo operando e interrompe a execução do programa, caso esse valor seja igual a '1'. A execução do programa é retomada após a detecção do valor lógico oposto (neste caso '0'), na entrada seleccionada, na primeira transição ascendente do relógio que comanda a unidade de controlo e descodificação de instruções. O operando, com oito bits, ocupa a posição de memória que se segue ao código de instrução. A Figura 6-47 ilustra o diagrama de transição de estados para esta instrução.

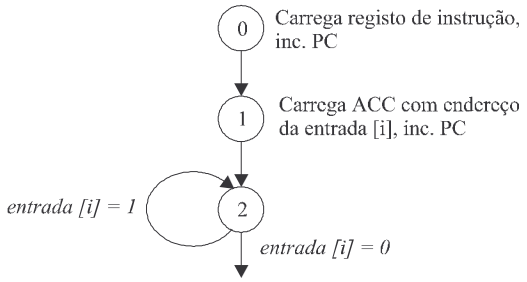


Figura 6-47: Diagrama de transição de estados para a instrução WAIT_WHL_NZ [i].

Instruções para suporte de operações passo-a-passo

Apresenta-se em seguida a caracterização individual das instruções da unidade CLF, destinadas ao suporte de operações de depuração do tipo passo-a-passo, que foram sucintamente descritas na Tabela 6-5.

CLK

Esta instrução aplica um impulso isolado na saída de relógio de sistema. Destina-se a mover o estado do sistema sob depuração um passo, nos casos em que para tal apenas é necessário um único impulso de relógio. A Figura 6-48 ilustra o diagrama de transição de estados para esta instrução.

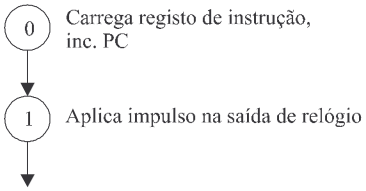


Figura 6-48: Diagrama de transição de estados para a instrução CLK.

CLK_N <N>

Esta instrução aplica um trem de N impulsos na saída de relógio de sistema, em que N constitui o valor do operando com oito bits (permitindo especificar valores entre 0 e 255) que se segue ao código de instrução. Permite mover o estado do sistema sob depuração um passo, nos casos em que são necessários vários impulsos de relógio, como por exemplo nos sistemas baseados em microprocessadores, em que um ciclo-máquina possa corresponder a vários impulsos de relógio. O valor N é carregado no contador interno de oito bits da unidade CLF, sendo decrementado de uma unidade por cada impulso aplicado na saída de relógio. A Figura 6-49 ilustra o diagrama de transição de estados para esta instrução.

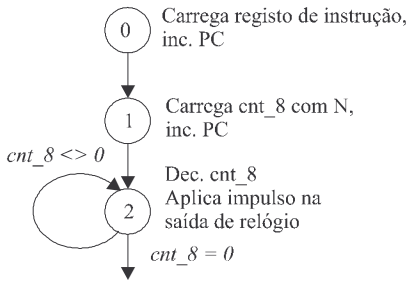


Figura 6-49: Diagrama de transição de estados para a instrução CLK_N <N>.

Instruções para suporte de operações de pontos de paragem por condição

Apresenta-se em seguida a caracterização individual das instruções da unidade CLF destinadas ao suporte de operações de depuração do tipo pontos de paragem por condição, que foram s u-cintamente descritas na Tabela 6-6.

CLK_WHL_Z [i]

Esta instrução aplica continuamente impulsos na saída de relógio do sistema, enquanto o valor lógico da entrada genérica seleccionada pelo operando desta instrução, for igual a '0'. A execução desta instrução permite a implementação de pontos de paragem por condição num sistema com componentes que suportem a instrução opcional *COMP* (descrita no capítulo anterior), através da ligação do pino SCD a uma das entradas genéricas da unidade CLF, que se pode seleccionar através do operando com oito bits, que ocupa a posição de memória seguinte à que contém o código de instrução. A Figura 6-50 ilustra o diagrama de transição de estados para esta instrução.

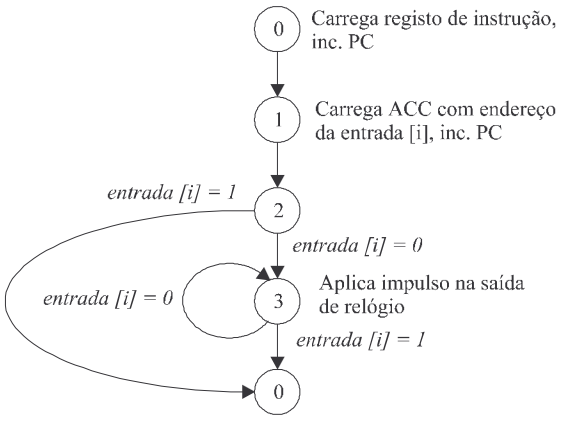


Figura 6-50: Diagrama de transição de estados para a instrução CLK_WHL_Z [i].

CLK_WHL_NZ [i]

Esta instrução aplica continuamente impulsos na saída de relógio do sistema, enquanto o valor lógico da entrada genérica seleccionada pelo operando desta instrução, for igual a '1'. Destina-se a permitir uma maior flexibilidade na especificação da condição de paragem e constitui a dual da instrução anterior. A Figura 6-51 ilustra o diagrama de transição de estados para esta instrução.

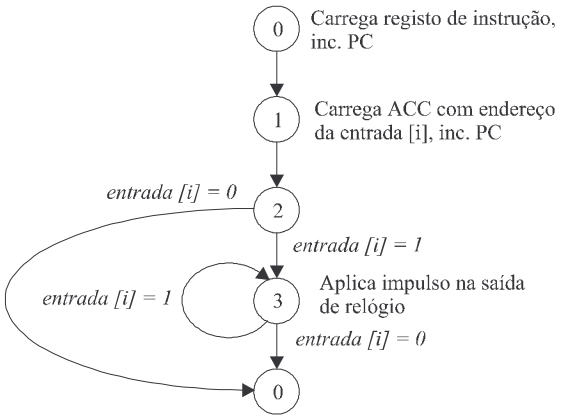


Figura 6-51: Diagrama de transição de estados para a instrução CLK_WHL_NZ [i].

Instruções para suporte de operações em tempo real

Apresenta-se em seguida a caracterização individual das instruções da unidade CLF, que foram sucintamente descritas na Tabela 6-7, destinadas ao suporte de operações de depuração em tempo real.

START_CLK

Esta instrução permite a aplicação contínua de impulsos na saída de relógio do sistema, ao carregar o valor lógico '1' no registo que controla o modo de funcionamento do circuito de g e-ração de relógio. Este modo de funcionamento só pode ser alterado através da execução da instrução STOP_CLK, ou da inicialização da unidade CLF. A Figura 6-52 ilustra o diagrama de transição de estados para esta instrução.

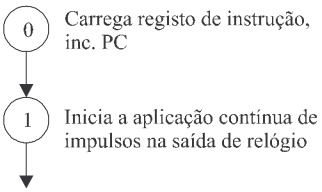


Figura 6-52: Diagrama de transição de estados para a instrução START_CLK.

STOP_CLK

Esta instrução pára a aplicação contínua de impulsos na saída de relógio do sistema, ao carregar o valor lógico '0' no registo que controla o modo de funcionamento do circuito de ger ação de relógio. Repare-se que as instruções CLK, CLK_N, CLK_WHL_Z e CLK_WHL_NZ, anteriormente apresentadas, só fazem sentido quando o circuito de geração de relógio se encontra neste modo de funcionamento. A execução da instrução CSTRETCH_N (a descrever em seguida) requer igualmente o modo de funcionamento especificado por esta instrução. A Figura 6-53 ilustra o diagrama de transição de estados para esta instrução.

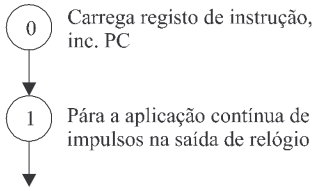


Figura 6-53: Diagrama de transição de estados para a instrução STOP_CLK.

CSTRETCH_N

Esta instrução executa parcialmente o algoritmo de alargamento de um impulso do relógio do sistema. Assumindo que o contador interno de 24 bits foi carregado inicialmente com o valor N, e que o seu valor actual é M (em virtude de se ter executado N - M vezes a instrução DJNZ Endereço_relativo, a descrever em seguida), a execução desta instrução corresponde à aplicação de um trem de M - 1 impulsos na saída de relógio, seguido de um impulso alargado (i.e. aplicação de um '0' durante um ciclo de relógio, seguido da aplicação de um '1' durante o mesmo período de tempo), terminando com a aplicação de N - M impulsos de relógio, com a mesma frequência do relógio principal que alimenta o PRODEP. Repare-se que no total foram aplicados N impulsos de relógio [(M –1) + 1 + (N - M) = N], tendo sido alargado o M-ésimo (que corresponde ao conteúdo actual do contador de 24 bits) impulso de relógio do sistema. A Figura 6-54 ilustra o diagrama de transição de estados para esta instrução.

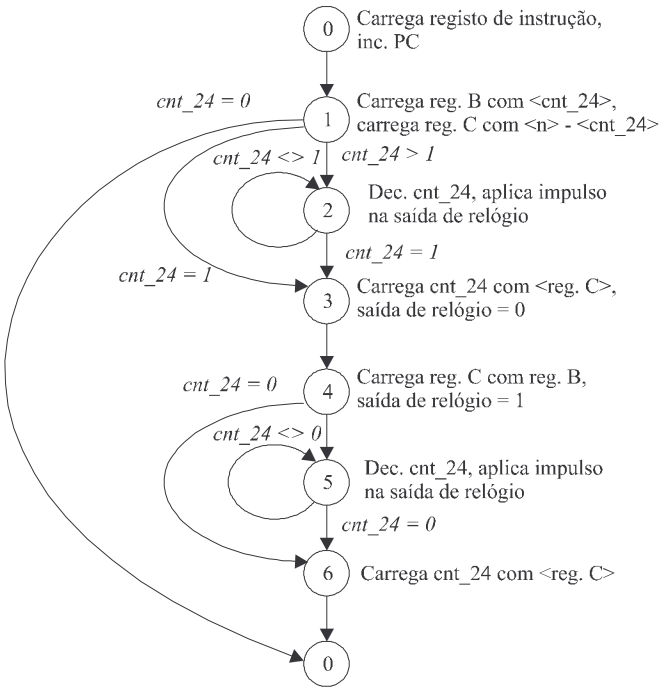


Figura 6-54: Diagrama de transição de estados para a instrução CSTRETCH_N.

Em virtude de se utilizar o contador de 24 bits na implementação desta instrução, armazena-se temporariamente o seu conteúdo num dos registos de uso genérico, carregando-se no final esse valor no contador, por forma a repô-lo no mesmo estado anterior. A Figura 6-55 ilustra um exemplo de execução desta instrução, supondo que o contador de 24 bits foi carregado inicialmente com o valor dez, e que o seu valor actual é oito.

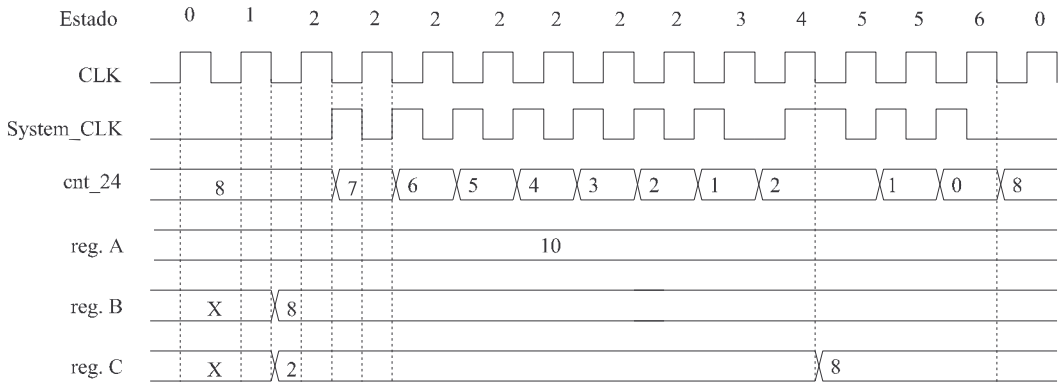


Figura 6-55: Diagrama temporal de execução da instrução CSTRETCH_N para um exemplo em que os conteúdos inicial e actual do contador de 24 bits são iguais a dez e oito, respectivamente, correspondendo assim à extensão do oitavo impulso de relógio do sistema.

Instruções para controlo dos recursos internos, canais de sincronismo e fluxo do programa

Apresenta-se em seguida a caracterização individual das instruções da unidade CLF que foram sucintamente descritas na Tabela 6-8 e que se destinam ao controlo dos recursos internos, canais de sincronismo e fluxo do programa.

LD C24, N

Esta instrução carrega no contador interno de 24 bits o conteúdo armazenado nas três posições de memória seguintes à que contém o código de instrução. A primeira destas posições contém os oito bits mais significativos do valor a carregar, a segunda posição contém os oito bits intermédios, e a terceira posição contém os oito bits menos significativos. A Figura 6-56 ilustra o diagrama de transição de estados para esta instrução.

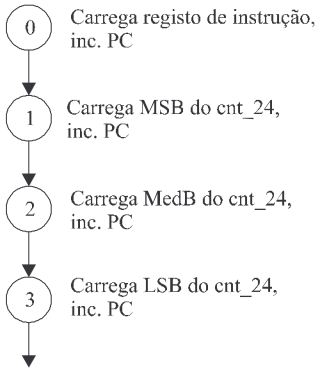


Figura 6-56: Diagrama de transição de estados para a instrução LD C24, N.

STORE C24

Esta instrução coloca o conteúdo do contador interno de 24 bits na memória externa do tipo FIFO. O processo de escrita é faseado, em consequência da largura do barramento de dados (oito bits) da memória seleccionada. O primeiro byte a ser transferido para a memória corresponde ao byte mais significativo do contador de 24 bits, o segundo byte ao byte intermédio e o último byte ao byte menos significativo. A Figura 6-57 ilustra o diagrama de transição de estados para esta instrução.

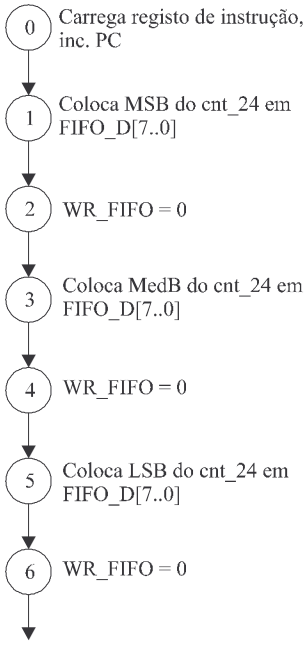


Figura 6-57: Diagrama de transição de estados para a instrução STORE C24.

DJNZ Endereço_relativo

Esta instrução decrementa o contador interno de 24 bits e verifica se o respectivo conteúdo chegou a zero. Se assim não for, soma-se ao registo de programa o valor representado pelo operando que se segue ao código da instrução, o que provoca um salto relativo com amplitude entre −127 e +128 bytes. A Figura 6-58 ilustra o diagrama de transição de estados para esta instrução.

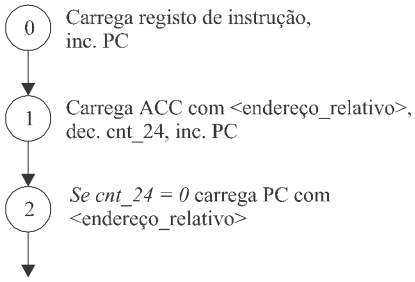


Figura 6-58: Diagrama de transição de estados para a instrução DJNZ Endereço_relativo.

Esta instrução permite a implementação de ciclos de programa do tipo para i = 1 até N faça <conjunto de instruções>, em que N corresponde ao valor carregado no contador interno de 24 bits, através da instrução LD C24, N.

SSA0, SSA1, SSB0, SSB1

Estas instruções permitem controlar o nível lógico (‘0’, ‘1’) presente nas saídas dos canais de sincronismo (A, B) da unidade CLF, sendo idênticas às instruções com o mesmo nome que foram anteriormente apresentadas para a unidade CLT. Estas instruções são usadas em conjunto com as instruções WS, para implementar um protocolo de sincronismo assíncrono com equipamentos de teste externos ou para sincronizar a execução de grupos de instruções com a unidade CLT. A Figura 6-27 ilustra o diagrama de transição de estados destas instruções.

WSA0, WSA1, WSB0, WSB1

Estas instruções retêm a unidade CLF no mesmo estado por um número indefinido de ciclos de relógio, até que a entrada do canal de sincronismo (A, B) se encontre no valor lógico especificado (‘0’, ‘1’). Estas instruções são usadas em conjunto com as instruções SS para implementar um protocolo de sincronismo assíncrono com equipamentos de teste ou para sincronizar a execução de grupos de instruções com a unidade CLT. A Figura 6-59 ilustra o diagrama de transição de estados para esta instrução, que difere ligeiramente da instrução equivalente que foi anteriormente apresentada para a unidade CLT (na Figura 6-28 constata-se a existência de mais um estado, devido a incrementar-se o registo de programa logo no estado 0).

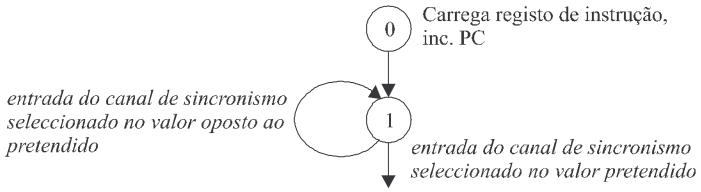


Figura 6-59: Diagrama de transição de estados para a instrução WS.

JP Endereço

Esta instrução permite carregar o registo de programa, incondicionalmente, com o endereço armazenado a seguir ao código de instrução. Este endereço ocupa os dois bytes (dezasseis bits) seguintes, dada a largura do respectivo barramento nesta unidade CLF. A Figura 6-60 ilustra o diagrama de transição de estados para esta instrução.

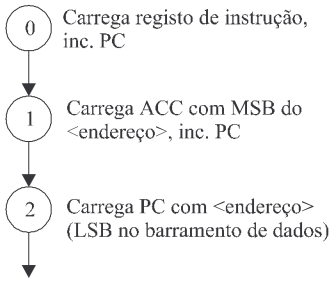


Figura 6-60: Diagrama de transição de estados para a instrução JP Endereço.

HALT

Esta instrução é usada para terminar a execução do programa, sendo idêntica à instrução com o mesmo nome, que faz parte do conjunto de instruções da unidade CLT. O diagrama de transição de estados é igual ao ilustrado na Figura 6-31.

6.4 Sincronismo / paralelismo entre as duas unidades principais

Esta secção analisa a questão do sincronismo entre operações, e execução de instruções, das duas unidades do PRODEP. Numa primeira fase descreve-se, através dos diagramas de estados das instruções que controlam os pinos de saída de relógio de cada uma das unidades[66], o sincronismo que existe entre a aplicação de impulsos num e noutro. Esta descrição cobre a aplicação de um único impulso de relógio – através das instruções CLK, e TMS0 (ou TMS1) –, a aplicação de um determinado número de impulsos – através das instruções CLK_N e NTCK - e a aplicação de vários impulsos de relógio, onde não é conhecido à partida o número a aplicar. Numa segunda fase descreve-se como se pode sincronizar a execução de instruções, ou grupos de instruções, utilizando o canal de sincronismo A, que interliga as duas unidades. Dado que se pode em qualquer instante sincronizar a execução de instruções entre as duas unidades, é possível a execução em paralelo de diferentes tarefas, resultando numa economia de tempo em termos de depuração.

6.4.1 Sincronismo entre operações de aplicação de impulsos de relógio

Apresenta-se em seguida, através dos diagramas temporais de execução das instruções que aplicam impulsos de relógio (de teste e do sistema), o mecanismo de funcionamento pelo qual

[66] Saídas de relógio de teste (TCK0 e TCK1), no caso da unidade CLT, e saída de relógio do sistema (System_CLK) , no caso da unidade CLF.

se garante a existência de sincronismo entre operações respeitantes à infraestrutura de teste e operações respeitantes à lógica funcional do sistema sob depuração.

Aplicação de um único impulso de relógio

Para aplicar um único impulso nas saídas de relógio de teste e do sistema, utilizam-se as instruções TMS0 (ou TMS1) e CLK, respectivamente da unidade CLT e da unidade CLF. Em termos de infraestrutura de teste, a aplicação de um impulso em TCK com a linha de TMS a um determinado nível lógico, pode corresponder a uma operação de captura de valores (na transição ascendente de TCK, com o controlador do TAP no estado *Capture-DR*[67]) ou a uma operação de aplicação de valores (na transição descendente de TCK, com o controlador do TAP no estado *Update-DR)*.

Em termos de lógica do sistema, a aplicação de um único impulso pode corresponder à transição entre dois estados, pelo que uma e outra operações devem estar sincronizadas, por forma a que se obtenha uma imagem correcta do funcionamento do sistema. A Figura 6-61 ilustra os diagramas temporais de execução destas instruções, onde é possível verificar que caso se garanta o mesmo tempo de atraso entre a entrada do relógio principal de ambas as unidades e as saídas de relógio de cada uma delas, se obtém um sincronismo perfeito na aplicação de um único impulso de relógio.

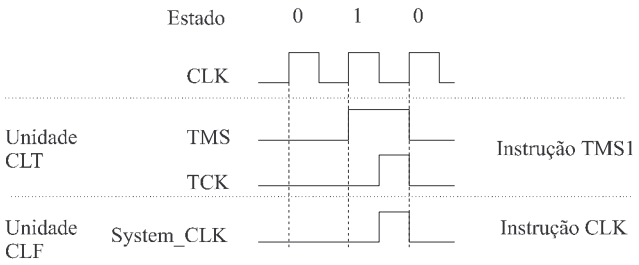


Figura 6-61: Aplicação síncrona de um impulso nas saídas de relógio de teste e de relógio do sistema.

Aplicação de N impulsos de relógio

Para aplicar um determinado número de impulsos nas saídas de relógio de teste e de sistema, utilizam-se as instruções NTCK e CLK_N, respectivamente da unidade CLT e da unidade CLF. Em termos de infraestrutura de teste, a aplicação de um determinado número de impulsos em TCK, com a linha de TMS ao nível lógico baixo '0', pode corresponder a uma operação de

[67] Corresponde geralmente à execução de uma instrução TMS0, dado que esta transição leva o controlador do TAP do estado *Capture-DR* para o estado *Shift-DR*, para deslocamento série dos valores capturados.

captura de valores em tempo real (na transição ascendente de TCK, com o controlador do TAP no estado *Run-Test/Idle*[68]). Em termos de lógica do sistema, a aplicação de vários impulsos corresponde à transição entre N estados, pelo que uma e outra operações devem estar sincronizadas, por forma a que também para este caso se obtenha uma imagem correcta do funcionamento do sistema. A Figura 6-62 ilustra os diagramas temporais de execução destas instruções[69], onde é possível verificar que caso se garanta o mesmo tempo de atraso, entre a entrada do relógio principal de ambas as unidades e as saídas de relógio de cada uma delas, se obtém um sincronismo perfeito na aplicação de vários impulsos de relógio (neste caso, para um exemplo com cinco impulsos).

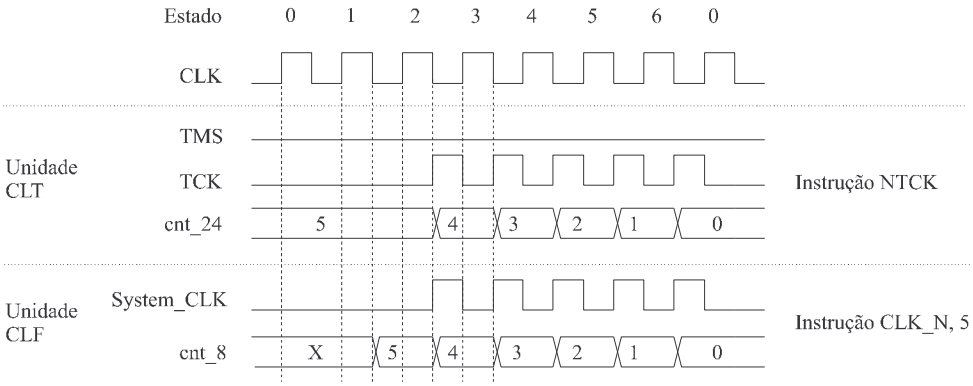


Figura 6-62: Aplicação síncrona de N impulsos nas saídas de relógio de teste e de relógio do sistema.

Aplicação de um número indeterminado de impulsos de relógio

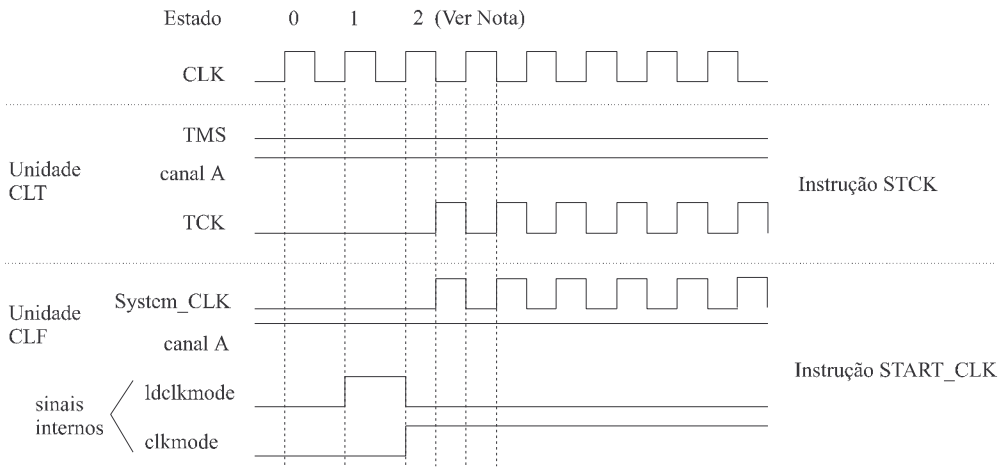
Para aplicar um número indeterminado de impulsos nas saídas de relógio de teste e de sistema, utilizam-se as instruções STCK e START_CLK (STOP_CLK), respectivamente, da unidade CLT e da unidade CLF. Em termos de infraestrutura de teste, a aplicação de um número indeterminado de impulsos em TCK, com a linha de TMS ao nível lógico baixo '0', pode corresponder a uma operação de captura de valores em tempo real (na transição ascendente de TCK, com o controlador do TAP no estado *Run-Test/Idle*[70]). Em termos de lógica do sistema, a aplicação de vários impulsos corresponde à transição entre estados, pelo que uma e outra opera-

[68] Corresponde à funcionalidade das instruções opcionais *MSEQ* e *MSEQ1P*, descritas no capítulo anterior.

[69] Assume-se que no caso da unidade CLT, o contador interno de 24 bits foi previamente carregado com o número N, que é utilizado como operando da instrução CLK_N, da unidade CLF.

[70] Corresponde à funcionalidade das instruções opcionais *MSATC* e *MSAPC*, descritas no capítulo anterior. Repare-se que, para estas instruções, o número de impulsos aplicados em TCK é irrelevante, dado que as MEF garantem que apenas o número correcto de vectores são armazenados no registo BS, antes e após a ocorrência de uma condição, que deverá ser detectada pela unidade CLF, através de uma qualquer das suas entradas genéricas.

ções devem estar sincronizadas, por forma a que uma vez mais se obtenha uma imagem correcta do funcionamento do sistema. A Figura 6-63 ilustra os diagramas temporais de execução destas instruções, onde é possível verificar que caso se garanta o mesmo tempo de atraso, entre a entrada do relógio principal de ambas as unidades e as saídas de relógio de cada uma delas, se obtém um sincronismo perfeito na aplicação de um número indeterminado de impulsos de relógio (supondo neste caso que o canal de sincronismo A se encontra ao nível lógico alto '1').



Nota: A unidade CLT mantém-se no estado 2, enquanto o canal A se mantiver em '1'.
A unidade CLF regressa ao estado 0, podendo executar outras instruções.

Figura 6-63: Aplicação síncrona de um número indeterminado de impulsos nas saídas de relógio de teste e de relógio do sistema.

6.4.2 Sincronismo entre execução de instruções / grupos de instruções

Apresenta-se em seguida a forma de utilização das instruções que controlam o canal de sincronismo A (SSA0, SSA1, WSA0 e WSA1), em ambas as unidades, para sincronizar o início de execução de instruções. Esta apresentação explica como é possível garantir que os blocos de controlo e descodificação de instruções de ambas as unidades, arrancam a partir do estado 0 após um determinado instante. Apenas a unidade CLF pode iniciar o processo de sincronização, que deverá resultar de um procedimento normalizado do módulo de geração automática do programa de teste e depuração, por forma a que exista uma correspondência directa entre as instruções de uma e outra unidades. O diagrama ilustrado na Figura 6-64 descreve o processo em que a unidade CLF inicia o pedido de sincronismo, ao colocar a sua saída do canal A em '1', aguardando que a unidade CLT confirme a aceitação do pedido, através da colocação do valor lógico '1' na sua saída do canal A (ligada à entrada de sincronismo do mesmo canal da unidade CLF). As duas unidades ficam sincronizadas a partir do momento em que a unidade

CLT coloca o valor lógico '0' (ou seja, o oposto do anterior) na saída do canal A, entretanto monitorizada pela unidade CLF através da execução da instrução WSA0.

Com base nos diagramas de transição de estado das instruções WSA0, da unidade CLF, e SSA0, da unidade CLT, ilustrados na Figura 6-59 e Figura 6-27, respectivamente, apresenta-se na Figura 6-65 o diagrama temporal do processo de sincronização de execução de instruções, entre estas duas unidades.

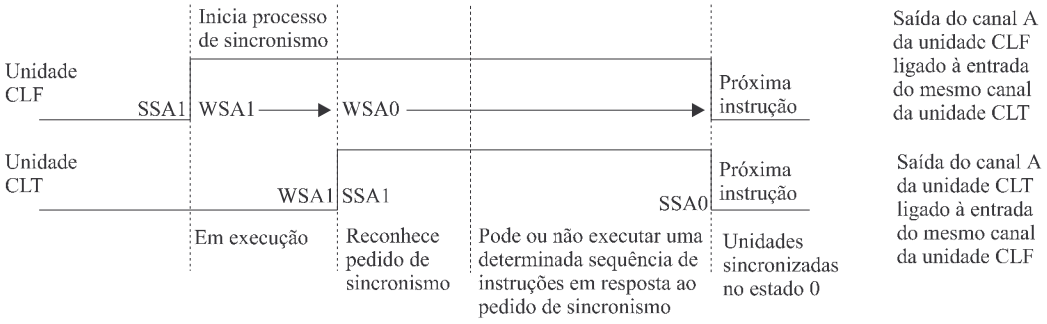


Figura 6-64: Processo de sincronização de execução de instruções entre as unidades CLT e CLF.

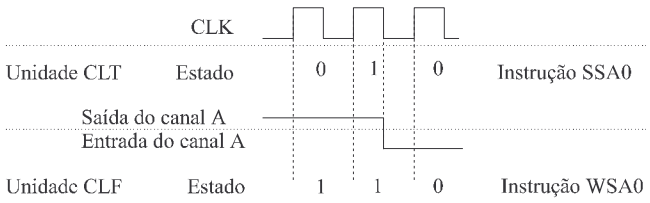


Figura 6-65: Diagrama temporal da entrada em sincronismo, no estado 0, das unidades CLT e CLF.

6.5 Conclusão

O controlador de teste e depuração descrito implementa os requisitos identificados no capítulo 4. Nos casos em que o controlador seja parte integrante do sistema sob depuração, adopta-se a expressão de controlador residente, por forma a salientar este facto. Repare-se que a inclusão do controlador no sistema sob depuração permite a execução posterior de operações de manutenção ou de verificação de funcionamento, no ambiente de utilização final.

A implementação do controlador num dispositivo lógico programável complexo, da família Flex 10K da Altera [Alt99], encontra-se descrita em anexo[71], incluindo-se ainda os ficheiros que contêm a especificação de cada um dos blocos principais das duas unidades. O nível de

[71] Esquemáticos no anexo em formato impresso e restantes elementos no anexo em CD-ROM.

topo do projecto, correspondente ao PRODEP, é constituído por um ficheiro (em esquemático) onde se interligam as duas unidades e se definem os pinos de Entrada e Saída deste controlador de teste e depuração.

Capítulo 7

Metodologia e geração automática do programa de teste e depuração

Neste capítulo descreve-se a metodologia e a geração automática do programa de teste e depuração, para cada um dos três tipos de sistemas utilizados ao longo deste documento:

- Sistemas baseados em lógica dedicada.
- Sistemas baseados em microprocessadores.
- Sistemas híbridos.

Para os sistemas baseados em lógica dedicada, descrevem-se as etapas consideradas para a verificação do projecto e do primeiro protótipo, identificando-se o fluxo de dados utilizado pela ferramenta computacional de Geração Automática do Programa de Teste e Depuração (GAPTD), nomeadamente a informação de entrada e de saída. A primeira secção termina com uma apresentação mais detalhada do algoritmo principal da ferramenta desenvolvida, incluindo a construção das estruturas de dados internas e a geração dos programas a executar pelas unidades CLF e CLT do PRODEP. Para os sistemas baseados em microprocessadores, colocam-se novamente as questões apontadas no capítulo 2, que reflectem a dificuldade da aplicação de uma metodologia estruturada de depuração a este tipo de sistemas, onde predomina a utilização de técnicas tradicionais baseadas na execução passo-a-passo do programa executado pelo microprocessador, na aplicação de pontos de paragem por condição para quebrar a totalidade da execução em blocos menos extensos, ou ainda na aplicação de técnicas de amostragem em tempo real, para depuração de problemas que ocorrem quando o sistema se encontra a trabalhar ininterruptamente à sua frequência máxima de operação. A utilização dos modos de operação opcionais para a infraestrutura de teste, descritos no capítulo 5, e dos recursos existentes no PRODEP, descritos no capítulo 6, permitem uma implementação eficiente e não intrusiva destas técnicas de depuração, conforme se descreve na segunda secção deste capítulo. Finalmente, para o caso dos sistemas híbridos, aborda-se a questão de definir qual a metodologia de depuração mais adequada, com base no exposto nas duas secções anteriores.

7.1 Sistemas baseados em lógica dedicada

Esta secção descreve o processo de verificação de sistemas baseados em lógica dedicada, considerando os níveis hierárquicos do CI e da CCI. Níveis hierárquicos inferiores (caso dos macroblocos), intermédios (caso dos módulos multi-pastilha), ou superiores (caso de sistemas formados pela ligação de várias CCI) são apenas referidos na secção de conclusão, em virtude do compromisso assumido entre a dimensão deste capítulo e a profundidade de análise pretendida para cada nível hierárquico.

7.1.1 Etapas e metodologia de depuração

Etapas e metodologia de depuração encontram-se intimamente ligadas, dado que a metodologia adoptada no início do projecto determina quais as etapas que deverão ocorrer (incluindo o seu faseamento) ao longo de todo o ciclo de desenvolvimento. Do ponto de vista da metodologia de depuração assume particular importância a etapa da *inserção das infraestruturas de teste*. Sendo este o veículo principal da aplicação dos vários procedimentos de teste e depuração, torna-se vital que esta ocorra o mais cedo possível no ciclo de desenvolvimento do sistema, o que depende naturalmente do nível hierárquico considerado: CI ou CCI. Partindo do pressuposto que as infraestruturas de teste se encontram já especificadas no momento em que se inicia a *simulação funcional* (fazendo portanto parte do modelo do sistema utilizado), é possível no final desta etapa proceder-se em paralelo à *simulação temporal*[72] e à *simulação da verificação funcional* do sistema. Esta última etapa corresponde à simulação do funcionamento do conjunto formado pela ligação do sistema ao PRODEP, que deverá executar o programa de teste e depuração, gerado automaticamente com base em diversa informação de entrada, que inclui, entre outros ficheiros, o resultado da simulação funcional do sistema.

Com base no modelo estrutural (ao nível da porta lógica) do sistema, procede-se à geração do teste estrutural, utilizando ferramentas de geração automática. A inclusão do programa de teste estrutural (nomeadamente dos vectores de teste) no programa a executar pelo PRODEP (em ambiente de simulação), depende da existência de módulos de conversão apropriados, sendo no entanto útil efectuar-se esta etapa, designada por *simulação do teste estrutural*, dadas as

[72] Considerando já cumpridas as etapas de síntese, colocação (*placement*), ligação (*routing*) e geração do modelo temporal, para o caso dos CI. Para o caso das CCI, considerando as etapas de colocação, ligação e geração do modelo temporal, com base nos atrasos inerentes ao comprimento e carga das linhas. É importante salientar neste ponto que os modelos individuais dos CI deverão na medida do possível ser do tipo temporal, embora possam existir casos em que coexistam vários tipos de modelos.

vantagens associadas aos ambientes de simulação (facilidade de detectar problemas na ligação entre as interfaces do sistema e do testador – neste caso, do PRODEP).

No final da simulação temporal do sistema, e com base nos resultados obtidos, dá-se início ao fabrico dos primeiros protótipos. Após a sua disponibilização é possível proceder-se de imediato, tendo em atenção os resultados entretanto obtidos por simulação, à execução do programa do PRODEP, que inclui a verificação da infraestrutura de teste, *o teste estrutural* e a *verificação funcional* e *temporal* do sistema desenvolvido. Atente-se neste ponto que o programa executado é idêntico ao programa utilizado nos ambientes de simulação, pelo que se abreviou o ciclo de desenvolvimento, ao efectuar-se anteriormente a tarefa da sua geração automática. Analisa-se em seguida em maior detalhe, para os níveis hierárquicos considerados (CI e CCI), cada uma das etapas de depuração mencionadas:

- Inserção das infraestruturas de teste.
- Simulação funcional.
- Simulação temporal.
- Simulação da verificação funcional.
- Simulação do teste estrutural.
- Teste estrutural.
- Verificação funcional.
- Verificação temporal.

Inserção das infraestruturas de teste

Para o caso de um CI, esta etapa depende da disponibilidade de ferramentas de inserção automática de infraestruturas de teste. A utilização de uma ferramenta deste tipo implica que o circuito se encontre descrito num determinado formato de entrada [Man96], existindo ainda actualmente diversas restrições, particularmente no que se refere à ligação de cadeias de varrimento internas à lógica BST, suporte de instruções opcionais definidas na norma IEEE 1149.1 e suporte de instruções e registos de dados de teste definidos pelo projectista. A inserção manual de infraestruturas de teste permite um maior grau de liberdade, à custa de um maior dispêndio de tempo e trabalho, que porém será inevitável em consequência das limitações das ferramentas de inserção automática. Repare-se que é possível reutilizar alguns módulos, como o controlador do TAP, registo de instrução, registo *bypass* e outros, qualquer que seja o tipo de circuito a desenvolver, pelo que a inserção manual não parte de uma base nula, mas antes beneficia de trabalho anteriormente efectuado.

Para o caso de uma CCI, esta etapa depende das infraestruturas de teste dos vários CI existentes. Nos casos em que exista um número significativo de componentes sem qualquer tipo de infraestrutura de teste, e para facilitar das tarefas de teste e depuração, pode-se justificar a troca

daqueles componentes por outros equivalentes, que possuam uma infraestrutura de teste, ou a inserção de CI adicionais destinados unicamente ao suporte destas tarefas [Mau92a].

Simulação funcional

Esta etapa consome normalmente uma larga parte do tempo associado à fase de depuração, dado ser frequente proceder-se a alterações na funcionalidade do CI em função dos resultados obtidos, que ajudam o projectista[73] a obter uma melhor e mais aprofundada visão do comportamento inicialmente especificado, geralmente numa linguagem de tipo não formal. A velocidade de simulação funcional (em virtude dos modelos utilizados serem menos complexos comparativamente com os modelos utilizados na simulação temporal) permite que se proceda a sessões de simulação envolvendo a execução de largos períodos de operação do circuito sob desenvolvimento. A realização desta etapa é ainda pouco frequente no caso de CCI, dadas as seguintes razões:

- Inexistência de modelos para todos os CI.
- Complexidade e dimensão do modelo da CCI, o que leva a tempos de simulação mais ou menos alongados, consoante o número de sinais que se pretendem visualizar (só entradas / saídas primárias, ou incluindo ligações internas).
- Fraca motivação e preparação dos projectistas para esta etapa.

A primeira razão tende a diluir-se à medida que aumenta o número de CI desenvolvidos a partir de linguagens de descrição de hardware. Para CI do tipo LSI (*Large Scale Integration*), ou MSI (*Medium Scale Integration*), que correspondem a CI projectados à cerca de uma / duas décadas atrás, mas que ainda são frequentemente utilizados, é já comum encontrarem-se bases de dados com modelos em VHDL, integradas em sistemas de desenvolvimento de CCI, como é o caso de [OrC99]. A segunda razão tende igualmente a diluir-se com o aumento da capacidade e da rapidez de processamento e de armazenamento, que é possível constatar nos computadores onde são actualmente instalados os sistemas de desenvolvimento. A terceira razão tende a desaparecer, à medida que aumenta a preocupação pela qualidade em geral e a quantidade de informação disponível na área do projecto para o teste e depuração.

Para efeitos da metodologia de depuração proposta, deve-se incluir no ficheiro que contém os resultados da simulação, a visualização dos valores presentes nos vários elementos sequenciais que se encontrem inseridos na(s) cadeia(s) de varrimento interna(s). Parte-se do pressuposto que as respostas obtidas nos pinos de saída e os estímulos aplicados aos pinos de entrada

[73] E o cliente final, nos casos em que a tarefa de projecto surge como uma actividade subcontratada.

estão sempre presentes neste ficheiro. No final desta etapa, ou seja, após se ter obtido um modelo suficientemente simulado, que corresponda à funcionalidade pretendida, atribui-se ao ficheiro que contém os resultados da última simulação, a designação de *ficheiro dourado (golden file)*, que servirá de base para dois objectivos: geração do programa de verificação funcional e comparação com o ficheiro que contém os resultados da simulação temporal.

Simulação temporal

A etapa da simulação temporal aproxima-se mais ou menos do funcionamento real do circuito consoante o nível de detalhe existente no modelo utilizado. A Figura 7-1 ilustra uma possível métrica de avaliação, onde se constata que um dos extremos (menor aproximação à realidade) corresponde à utilização de modelos com atrasos unitários, idênticos para todos os tipos de portas lógicas, e que o outro extremo corresponde à utilização de modelos ao nível da porta lógica, com atrasos específicos para cada tipo de porta e com anotação dos tempos de atraso decorrentes do percurso (análise geométrica em três dimensões) e da carga (dinâmica) dos sinais. Para o caso dos CI é frequente utilizarem-se modelos ao nível da porta lógica, com anotação dos tempos de atraso decorrentes do percurso dos sinais. Estes modelos possuem uma dimensão e complexidade mais elevada, que influenciam negativamente o tempo de simulação, em comparação com o tempo de simulação funcional. Para os casos das CCI, tendo presente o exposto na etapa anterior, é actualmente pouco frequente a realização desta tarefa.

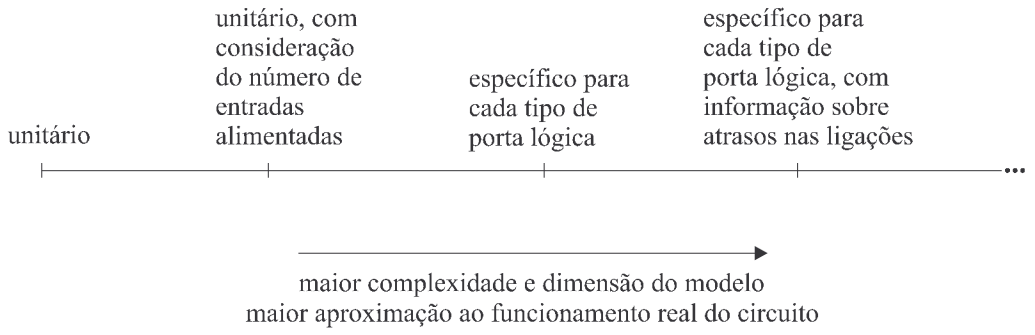


Figura 7-1: Caracterização do tipo de modelo utilizado na simulação temporal.

Para efeitos da metodologia de depuração proposta, procede-se no final desta tarefa à comparação entre o ficheiro dourado e o ficheiro que contém os resultados da simulação temporal, nos momentos em que ocorrem as transições ascendentes do relógio do circuito sob depuração. Repare-se que se está implicitamente a restringir o tipo de sistemas considerados aos circuitos puramente síncronos, alimentados por uma única entrada de relógio. Uma forma simples de efectuar este procedimento consiste em filtrar a informação existente no ficheiro com os resul-

tados da simulação temporal, eliminando todos os vectores que não correspondem a transições ascendentes do relógio e comparando-o no final com o ficheiro dourado (por simples comparação de ficheiros). A Figura 7-2 ilustra este procedimento para um exemplo obtido durante o desenvolvimento de um circuito implementado em lógica programável.

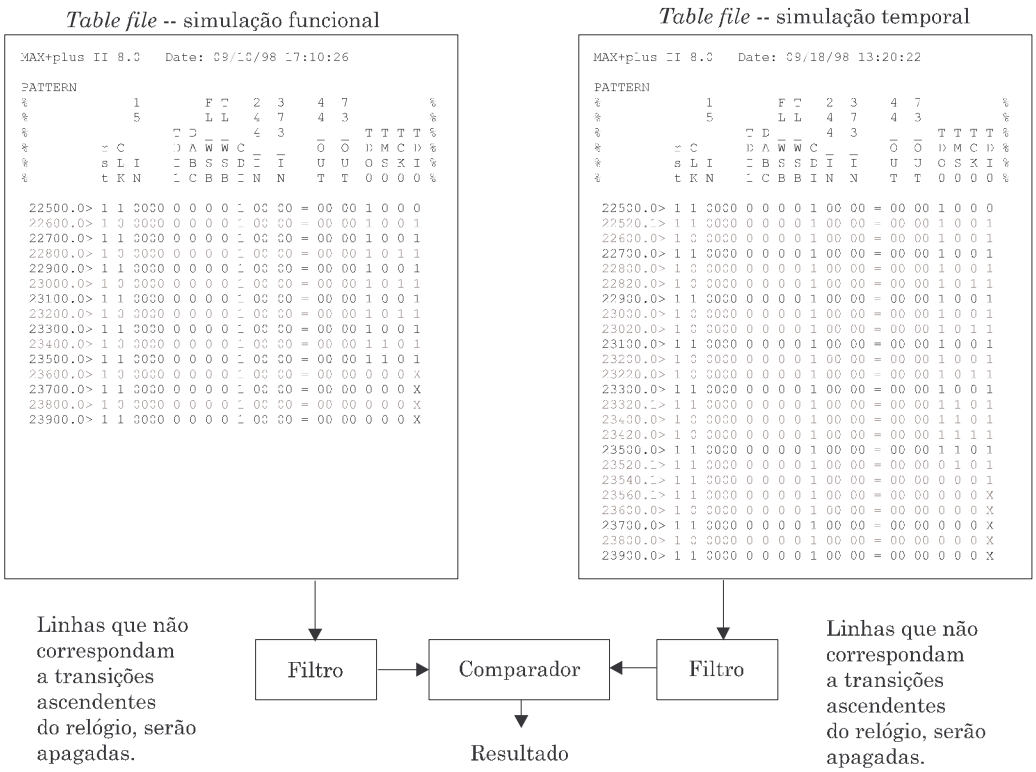


Figura 7-2: Comparação entre os ficheiros que contêm os resultados da simulação funcional e temporal.

Simulação da verificação funcional

Com base nos resultados obtidos na simulação funcional, procede-se à GAPTD (a descrever em seguida). Uma vez que o modelo do PRODEP (funcional e temporal) já se encontra disponível, é possível simular a execução do programa entretanto gerado, através da criação de um modelo correspondente à junção do modelo funcional do CI em desenvolvimento, ao modelo do PRODEP e das duas memórias (com os programas das unidades CLT e CLF). Esta situação[74] permite, ainda antes do desenvolvimento de qualquer protótipo, detectar possíveis erros na aplicação do próprio programa de teste e depuração, poupando assim tempo precioso, e aumentar a confiança no processo de verificação funcional dos primeiros protótipos.

[74] A que se atribui geralmente a designação de *co-simulação*.

Simulação do teste estrutural

Os vectores de teste, obtidos através das ferramentas de geração automática do programa de teste estrutural, podem ser integrados no programa executado pelo PRODEP, caso se garanta a existência de módulos de conversão entre os dois formatos utilizados (a representação dos vectores de teste gerados e o conjunto de instruções do CLT). Esta situação permite detectar possíveis erros na aplicação do próprio programa de teste estrutural, aumentando assim a confiança nesta etapa de verificação dos primeiros protótipos.

Teste estrutural

A verificação do primeiro protótipo inicia-se com a aplicação do teste estrutural, que engloba as fases descritas na Tabela 7-1 (para CI e CCI).

Tabela 7-1: Fases do teste estrutural de CI e CCI.

CI	CCI
Teste da infraestrutura de teste	Teste da infraestrutura de teste
Teste da lógica interna Teste de faltas do tipo *sempre-a*. Teste de faltas do tipo circuito aberto. Teste de faltas do tipo curto-circuito. Nota: Preferencialmente através da utilização da instrução opcional *INTEST* e da instrução‡ de acesso à cadeia de varrimento interna. A não existência de uma cadeia de varrimento interna dificulta a geração do teste estrutural (circuito sequencial em comparação com circuito combinatório). O não suporte da instrução *INTEST* obriga a utilizar equipamentos de teste automático, ou a rodear o CI por um conjunto externo de células BS, pertencentes a um dos componentes descritos em [Alv93, Fer93].	Teste das interligações: Entre grupos de componentes BST Teste de faltas do tipo *sempre-a*. Teste de faltas do tipo circuito aberto. Teste de faltas do tipo curto-circuito. Teste de faltas do tipo atraso†. Entre grupos de componentes não BST Teste de faltas do tipo *sempre-a*. Teste de faltas do tipo circuito aberto. Teste de faltas do tipo curto-circuito. Teste de faltas do tipo atraso. Entre grupos de componentes BST e não BST
	Teste dos componentes: Activação de funções de auto-teste. Teste da lógica interna, através da reutilização de vectores de teste, aplicados simultaneamente através das cadeias de varrimento periféricas e internas.

‡Ou instruções, no caso de existirem várias cadeias de varrimento internas.

†Através da utilização da instrução opcional *DETRA*.

A separação entre teste estrutural e verificação funcional permite aumentar a capacidade de diagnóstico de faltas detectadas, apesar de existir uma sobreposição de vectores, ou seja, a aplicação nestas duas etapas de vectores de teste idênticos, o que faz aumentar o tempo total de teste.

Verificação funcional

Esta etapa consiste na aplicação do programa de teste e depuração, gerado automaticamente a partir do ficheiro com o resultado da simulação funcional. No caso de CI é possível verificar os valores presentes nos pinos de saída e nos elementos sequenciais internos, que satisfaçam simultaneamente as seguintes condições:

- Pertencerem à cadeia de varrimento interna.
- Terem sido incluídos como canais de saída no ficheiro com os resultados da simulação.

No caso de CCI é possível verificar os valores presentes nos pinos de saídas primárias, nas ligações internas e nos elementos sequenciais internos (referente a CI), desde que se garantam simultaneamente as seguintes condições:

- As ligações internas incluírem um pino com uma célula BS associada.
- Os elementos sequenciais internos pertencerem a cadeias de varrimento internas (de CI).
- Ambos corresponderem a canais de saída no ficheiro com os resultados da simulação.

A aplicação dos estímulos de entrada é efectuada a partir das células BS associadas aos pinos de entrada do CI, ou das células BS associadas aos pinos de componentes com BST que correspondam a entradas primárias da CCI, desde que se verifique o requisito do suporte da instrução opcional *INTEST*. Caso contrário, será necessário utilizar equipamentos de teste automático ou, opcionalmente, ligar os pinos de entrada a um colar externo de células BS, conforme se ilustra na Figura 7-3. Esta solução foi já descrita em [Alv93, Alv95, Fer93], onde se refere a utilização de componentes que internamente possuem unicamente um conjunto de 26 células BS bidireccionais (i.e. associadas a pinos bidireccionais com controlo de estado individual) e um conjunto de dez células BS de entrada (i.e. associadas a pinos de entrada), mais o resto do circuito exigido pela norma IEEE 1149.1. A lógica funcional deste componente é inexistente, ou seja, possui um núcleo funcional vazio.

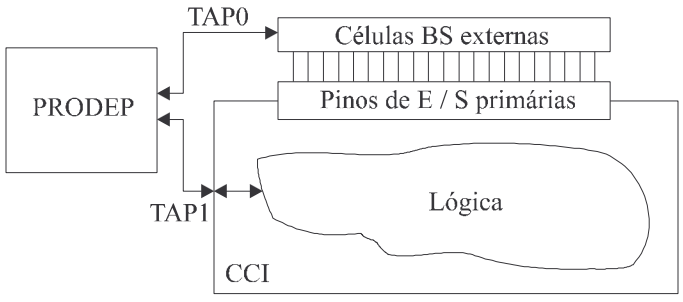


Figura 7-3: Ligação dos pinos de E/S primárias da CCI a um conjunto de células BS externas.

Verificação temporal

Esta etapa de depuração consiste na utilização da instrução CSTRETCH_N, da unidade CLF do PRODEP, nas situações em que se detectem faltas no comportamento do sistema sob depuração, quando este se encontre a trabalhar à sua frequência máxima de operação. Tipicamente, uma falta é externamente detectada *n* ciclos após ter sido propagada a partir da sua origem num determinado ponto do circuito. O procedimento para a localização e diagnóstico da falta consiste na execução dos seguintes passos:

- Parar o sistema imediatamente após a sua detecção.
- Determinar qual o estado actual, através do deslocamento para o exterior do valor existente em todos os pinos e elementos sequenciais acessíveis através da infraestrutura de teste. Os valores armazenados em memórias poderão igualmente ser obtidos através dos processos de leitura descritos no capítulo 3, supondo que as entradas / saídas destas (endereços, dados e controlo) se encontram ligadas a pinos de componentes BST, ou a elementos sequenciais acessíveis por varrimento.
- Determinar, através de um processo designado por *análise regressiva* (*backtracking*), um estado anterior (afastado *m* ciclos) ao actual, onde se suponha ainda não ter ocorrido a falta. Este processo pode ser efectuado com base na simulação funcional do sistema.
- Aplicar o algoritmo de alargamento de um impulso de relógio, utilizando as capacidades existentes no PRODEP, nomeadamente a instrução CSTRETCH_N da unidade CLF. O contador interno de 24 bits desta unidade deve ser previamente carregado com o número *m*. No final da aplicação de cada trem de *m* impulsos, verificar se a falta ainda persiste, ou seja, se ainda é externamente detectada.

Caso a falta continue a ser detectada, voltar a aplicar novo trem de *m* impulsos, alargando agora o impulso seguinte (àquele que foi anteriormente alargado), e assim sucessivamente até que a falta deixe de ser detectada, ou se tenham aplicado *m* trens de *m* impulsos. Nesta última situação (a mais desfavorável), deve-se repetir todos os passos anteriores, com um número de impulsos superior ao anteriormente utilizado (*m*). Na situação em que se deixe de detectar a falta, verificar qual o valor (*i*) do contador de 24 bits. Colocar o sistema no estado inicial considerado (ou seja o estado *m* ciclos anterior ao ciclo em que a falta é detectada) e aplicar um número de impulsos idêntico ao valor *i*. Dado que não se alargou nenhum impulso de relógio, a falta é activada neste mesmo ciclo, conforme se pode verificar pelo exemplo ilustrado na Figura 7-4. Deslocar e comparar os valores capturados com os valores esperados, obtidos através da simulação temporal do sistema, para as mesmas condições de funcionamento. A detecção de uma diferença permite localizar o pino ou elemento sequencial que captura o valor contrário ao esperado. Através da análise do sistema, identifica-se o cone de influência do ponto em causa, ou seja, quais os pinos ou elementos sequenciais que influenciam o valor existente

nesse ponto, efectuando-se o diagnóstico com base numa ferramenta de análise temporal estática, que determine qual o percurso lógico que introduz o atraso superior ao que d everia existir.

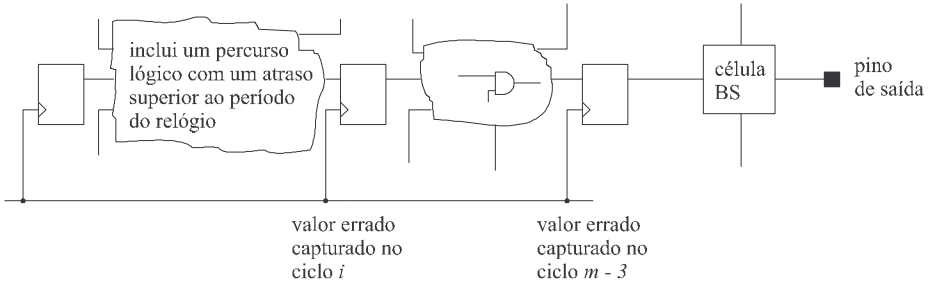


Figura 7-4: Exemplo de um percurso lógico com um atraso superior ao previsto, activado no ciclo *i*.

7.1.2 Identificação do fluxo de dados

As etapas de depuração descritas na secção anterior são endereçadas de formas distintas em função do nível hierárquico do sistema considerado, CI ou CCI.

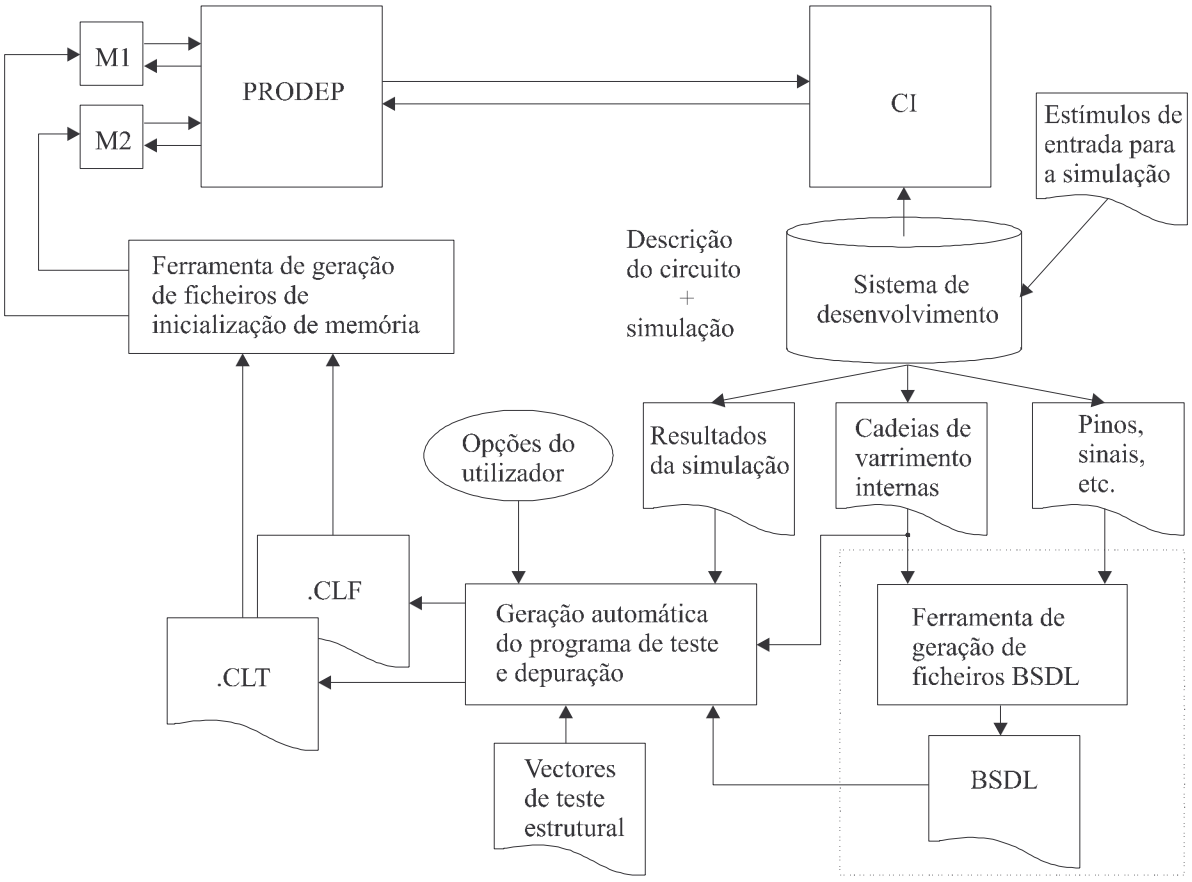


Figura 7-5: Fluxo de dados no desenvolvimento, simulação e verificação de um CI.

A Figura 7-5 ilustra uma representação simplificada do fluxo de dados existente no processo de desenvolvimento, simulação e verificação de um CI, considerando a metodologia apresentada na secção anterior. É possível através da Figura 7-5 identificar rapidamente qual a informação de entrada e de saída da ferramenta de GAPTD. Nas duas subsecções seguintes descreve-se em maior detalhe o conteúdo da informação de entrada, e de saída, respectivamente, sendo a última subsecção dedicada à descrição do algoritmo utilizado pela ferramenta desenvolvida. A Figura 7-6 refere-se ao caso de uma CCI, sendo igualmente apresentado nas subsecções sequintes toda a informação relativa a este nível hierárquico.

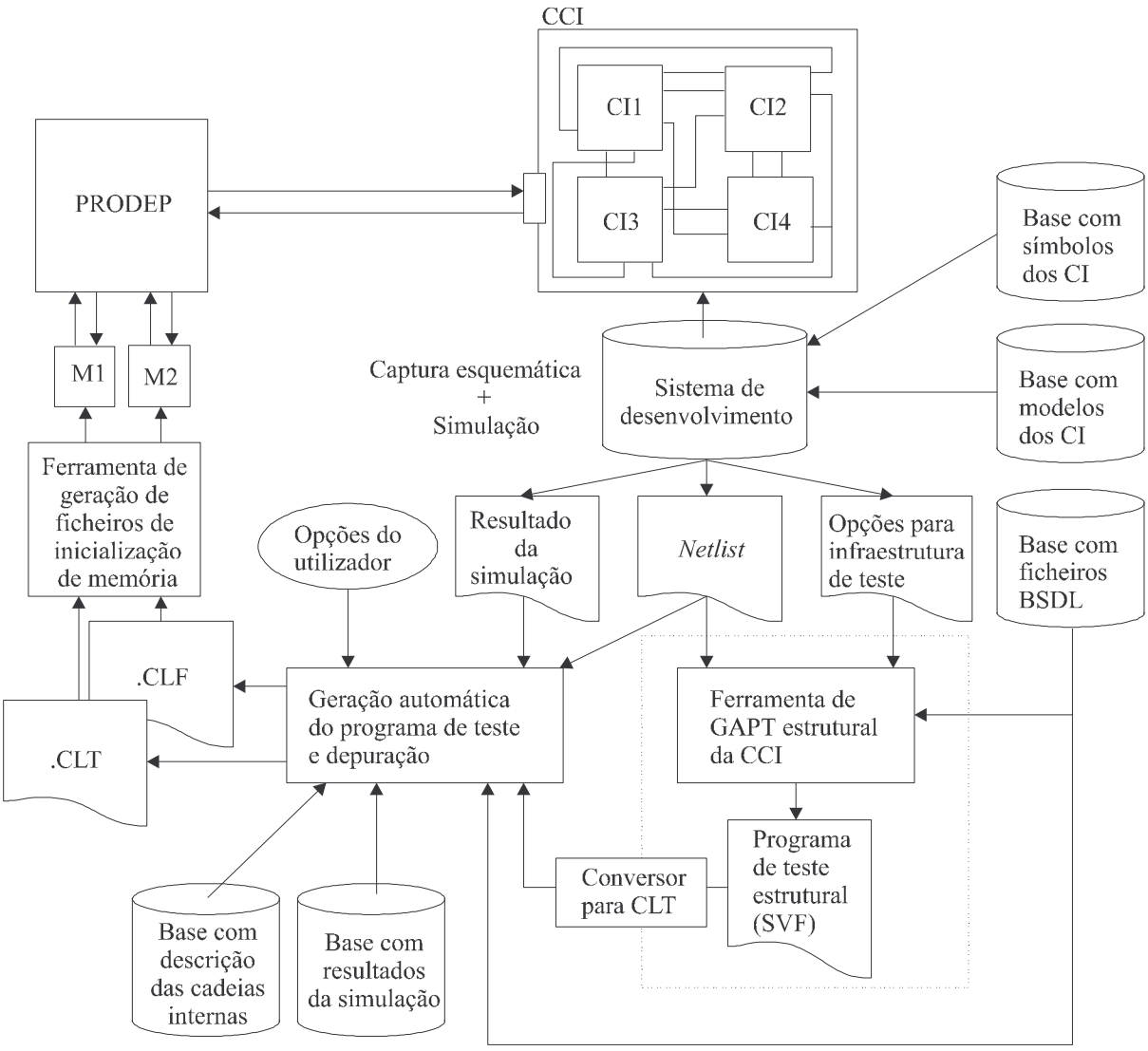


Figura 7-6: Fluxo de dados no desenvolvimento, simulação e verificação de uma CCI.

7.1.3 Informação de entrada

A informação de entrada da ferramenta de GAPTD difere, consoante se trate da verificação de um CI ou de uma CCI. Existe ainda todo um conjunto de situações referentes ao tipo, extensão e quantidade de infraestruturas de teste existentes no sistema sob depuração, que influenciam o processo de verificação. Tentar-se-á, na medida do possível, considerar essas situações ao longo desta subsecção.

Com base na Figura 7-5 e na Figura 7-6, é possível listar toda a informação de entrada utilizada pela ferramenta computacional de GAPTD, referindo-se em cada caso as diferenças que decorrem de se considerar o caso de um CI ou de uma CCI:

- Ficheiro BSDL, no caso de um único CI com BST. Ficheiros BSDL de todos os componentes BST, no caso de uma CCI com vários componentes deste tipo.
- Ficheiro com o resultado da simulação funcional do sistema (CI e CCI). Deve ter-se em atenção quais os canais de simulação[75] existentes ou seleccionados. Para o caso da CCI, considera-se a existência de uma base de dados com os ficheiros de simulação individual de cada um dos componentes inseridos na carta.
- Descrição das cadeias de varrimento internas (CI). Para o caso da CCI, considera-se a existência de uma base de dados com os ficheiros que contêm a descrição das cadeias de varrimento internas de cada um dos componentes inseridos na carta.
- Ficheiro que contém os vectores (ou o programa) de teste estrutural do CI, ou da CCI. Este ficheiro é produzido por uma ferramenta externa de geração automática de programas de teste e posteriormente integrado no programa executado pelo PRODEP.
- Ficheiro que reúne todas as opções e informações introduzidas pelo utilizador.
- Descrição das ligações entre os componentes inseridos na CCI (*Netlist*).

Cada uma destas componentes da informação de entrada é em seguida descrita com maior detalhe, referindo-se ainda qual a sua contribuição para a estrutura de informação interna da ferramenta de GAPTD.

Ficheiro(s) BSDL

Este ficheiro contém a descrição da infraestrutura BST do CI, no formato definido na norma IEEE 1149.1 [IEEE93]. A correspondência entre número / nome dos pinos funcionais do CI e as células BS que lhes estão associadas, encontra-se descrita neste ficheiro, que contém igualmente o código das instruções opcionais, bem como a indicação de qual o registo de dados de

[75] Entende-se por canal de simulação um pino, elemento sequencial, ou mesmo um sinal de saída de uma porta lógica, cujo valor seja possível visualizar no ficheiro que contém as respostas da simulação.

teste seleccionado (i.e. colocado no percurso TDI - TDO interno) e qual o seu comprimento. A instrução opcional, definida pelo utilizador, que permite aceder à cadeia de varrimento interna (bem como o seu comprimento), faz parte da informação contida neste ficheiro. Idealmente, o ficheiro BSDL deverá ser produzido automaticamente, embora algumas ferramentas actuais não suportem todas as opções possíveis, nomeadamente algumas instruções opcionais definidas na norma, ou pelo utilizador [Man96].

Resultado da simulação

Existem actualmente diversos formatos para o ficheiro que contém o resultado da simulação, não sendo possível afirmar-se que exista uma norma estabelecida ou *de facto*, para este tipo de ficheiros. É possível no entanto encontrar vários exemplos de ficheiros cujo conteúdo se baseia unicamente na utilização de caracteres do tipo ASCII. Este tipo de ficheiros possui como vantagem principal a facilidade de leitura e de compreensão do seu conteúdo, o que favorece o rápido desenvolvimento de um *módulo de pré-processamento,* para a extracção da informação pretendida. No anexo em formato electrónico apresenta-se o ficheiro que contém o resultado da simulação do CI usado no exemplo de utilização, que possui um formato tabular com as linhas verticais a corresponderem a canais de simulação (ou seja, sinais de entrada, saída e internos, usados na sessão de simulação) e as linhas horizontais a corresponderem a momentos no tempo. A intersecção entre uma linha vertical e uma linha horizontal é preenchida com o valor lógico do respectivo sinal, naquele momento.

Descrição das cadeias de varrimento interno

Este ficheiro complementa a informação contida no ficheiro BSDL, relativa às cadeias de varrimento internas. Nele se indica a correspondência entre o número de ordem de uma dada célula da cadeia e o nome lógico do sinal associado, que deverá ser idêntico ao nome utilizado no ficheiro de resultado da simulação (ou seja, um canal de simulação que corresponda a um elemento acessível por varrimento interno deverá possuir o mesmo nome em ambos os ficheiros).

Programa ou vectores de teste estrutural

Existem diversos formatos para o ficheiro que contém o programa de teste estrutural. No que se refere ao teste estrutural através da infraestrutura BST, pode no entanto considerar-se o formato SVF (*Serial Vector Format*) como uma norma de facto, dada a sua utilização generalizada. O formato STIL (*Standard Test Interface Language*), ou norma IEEE 1450, deve igualmente ser destacado, dado que no momento em que se escreve esta tese, o processo de publicação se aproxima a passos largos do seu epílogo, conforme se pode verificar por uma mensagem re-

centemente enviada pelo responsável do grupo de trabalho, Mr. Greg Maston, a todos os participantes no processo de normalização, incluída em nota de rodapé[76]. Este formato é particularmente importante pelo facto de, como tudo indica, se tornar uma norma facilmente acessível a todos os utilizadores. Estes dois formatos possuem ainda a vantagem adicional de se basearem em caracteres do tipo ASCII, o que favorece novamente o rápido desenvolvimento de um módulo de pré-processamento para extracção da informação pretendida. No caso de um programa de teste escrito em SVF, e dadas as semelhanças existentes com a linguagem da unidade CLT, torna-se fácil incorporar o programa de teste estrutural no programa a executar por esta unidade, após uma breve passagem pelo respectivo módulo de pré-processamento, ilustrado na Figura 7-6.

Opções e informações entradas pelo utilizador

Este ficheiro contém a descrição da correspondência entre os nomes lógicos das entradas e saídas primárias do sistema sob depuração, e as células BS externas, conforme a situação ilustrada na Figura 7-3. Contém ainda diversas opções e informações introduzidas pelo utilizador, nomeadamente:

- Se o pino de relógio funcional do circuito sob depuração se encontra (ou não) ligado ao pino System_clk do PRODEP.
- Qual o período do sinal de relógio do circuito, utilizado na sessão de simulação.
- Qual o período do sinal de relógio que alimenta o PRODEP.

O facto de se concentrar num único ficheiro toda a informação introduzida pelo utilizador, facilita a futura expansão do tipo e extensão desta informação.

Descrição estrutural da CCI

Este ficheiro contém o mapa das ligações entre os vários componentes, BST e não BST, inseridos na CCI sob depuração. Existem vários formatos para este tipo de ficheiro, sendo possível encontrar programas comerciais que convertem entre si algumas dezenas de formatos [RSI99]. Formatos que só utilizem caracteres em ASCII, apresentam mais uma vez a vantagem referida

---

[76] *"To all involved with 'P'1450,*
Congratulations! At 8:20 am Eastern Standard Time on St Patty's Day, 1999, IEEE Revcom accepted our work on P1450, as a "consent agenda" item (which means we crossed enough T's that they had no questions or issues about our work. There were 3 efforts on consent approval at this meeting - and a dozen or so efforts with questions. We done good.) The effort is now officially recognized as 1450. Thank you all for the work you put into this! Of course, you are aware that the work is not complete. Now we start working with the IEEE Editorial staff, to cross the final T's. I'm hopeful that we can complete this task in a timely fashion and get our effort published and available to the industry."

anteriormente. Dada a existência e facilidade de obtenção de conversores de formatos, pode-se optar pelo suporte de um único formato e, com base nele, construir o módulo de pré-processamento para a extracção da informação pretendida. Nas situações em que o ficheiro com a descrição estrutural da CCI se encontre num formato diferente do aceite pelo referido módulo, efectua-se uma conversão de formatos e procede-se então à fase de extracção de informação.

7.1.4 Informação de saída

Esta subsecção descreve em detalhe a informação de saída produzida pela ferramenta computacional de GAPTD, que consiste em dois ficheiros: o programa a executar pela unidade CLT e o programa a executar pela unidade CLF, ambos em linguagem *assembly*[77] (sufixo .asm). Estes ficheiros são lidos por uma aplicação *shareware* denominada TASM[78] [Tho98], que gera os respectivos ficheiros em código-máquina (sufixo .obj), i.e. com os códigos de instrução e operandos, bem como um ficheiro-relatório (sufixo .lst) que contém os endereços, o código fonte em *assembly*, o código-máquina e uma lista com as etiquetas (*labels*) utilizadas ao longo do programa para instruções de salto, com a indicação do respectivo valor. Os ficheiros-relatório são posteriormente lidos por um módulo que os converte em modelos ou ficheiros de inicialização de memórias (sufixo .mif), para serem utilizados no ambiente de desenvolvimento no qual se efectua a simulação da execução do programa do PRODEP. Esta conversão depende no entanto de qual o formato de entrada aceite, para este tipo de ficheiros, pelo ambiente de simulação. A identificação de cada uma das instruções de uma dada unidade encontra-se por sua vez armazenada num ficheiro em formato tabular (sufixo .tab), lido pelo TASM no início do processo de geração do código-máquina.

Programa da unidade CLF

A Figura 7-7 ilustra a geração do ficheiro com o código a ser executado pela unidade CLF e a geração do ficheiro de inicialização da memória utilizada no ambiente de simulação de um sistema de desenvolvimento para lógica programável [Alt99].

---

[77] As mnemónicas utilizadas foram já apresentadas no capítulo anterior, nas tabelas que descrevem o conjunto de instruções de cada uma destas unidades do PRODEP.

[78] O TASM é um *table-driven cross-assembler*, uma vez que se baseia numa tabela (com a correspondência entre mnemónicas e código-máquina) e porque é executado por um microprocessador diferente daquele para o qual gera código.

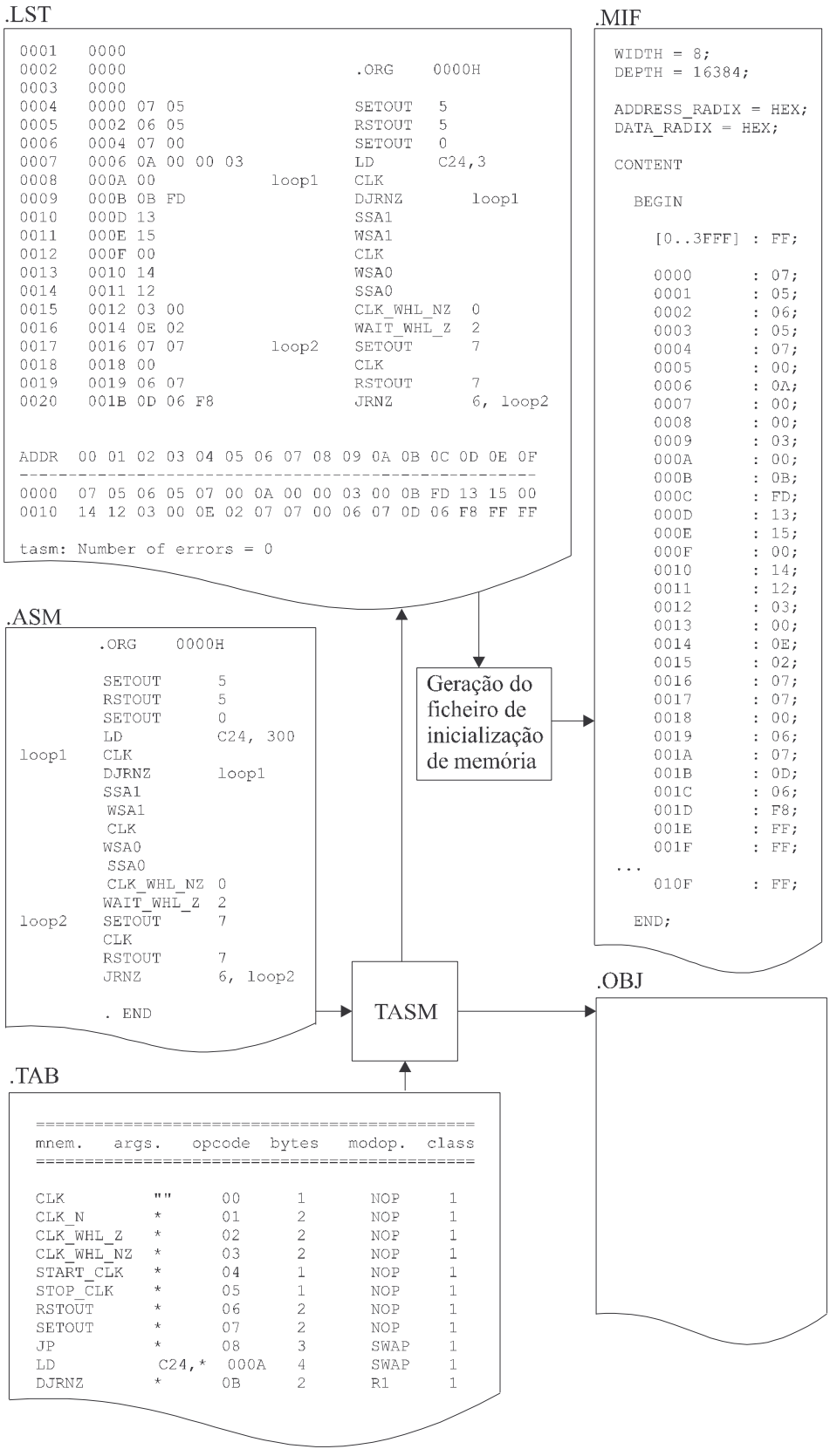


Figura 7-7: Exemplo de geração do ficheiro com o código a ser executado pela unidade CLF e do ficheiro de inicialização da memória utilizada num ambiente de simulação.

Programa da unidade CLT

A geração do ficheiro com o código a ser executado pela unidade CLT e a sua posterior conversão no ficheiro utilizado para a inicialização da memória usada no ambiente de simulação, é idêntico ao referido para o exemplo anterior.

7.1.5 Geração automática do programa de teste e depuração

Esta subsecção descreve o processo de GAPTD, nomeadamente a formação da estrutura de dados interna e a criação dos programas a executar pelas unidades CLT e CLF. De forma a ilustrar mais detalhadamente este processo, recorre-se a um exemplo de GAPTD para um único CI com uma infraestrutura de teste compatível com a norma IEEE 1149.1, e com uma cadeia de varrimento interna acessível através do TAP.

Formação da estrutura de dados interna

O ficheiro BSDL é o primeiro a ser lido pela ferramenta computacional de GAPTD. A informação extraída[79] da secção "*port*" deste ficheiro, é armazenada na estrutura de informação de *nós* do circuito, denominada *Nodes*. Inicialmente, esta estrutura de informação contém um elemento por cada pino funcional do CI, com a respectiva identificação e a indicação de que se trata de uma entrada, saída, ou ambos (para o caso de pinos bidireccionais). O atributo *boundary_register*, existente no BSDL, identifica as células BS associadas, nomeadamente o seu número de ordem no registo BS e o seu tipo, de acordo com a topologia [80] exemplificada na norma IEEE 1149.1. É ainda extraído deste ficheiro outra informação, que inclui:

- Comprimento do registo BS (através do atributo *boundary_length*).
- Comprimento do registo de varrimento interno e o nome da instrução que permite colocá-lo no percurso TDI - TDO (através do atributo *register_access*).
- Comprimento do registo de instrução (através do atributo *instruction_length*).
- Os códigos das instruções *SAMPLE / PRELOAD*, *INTEST* e de acesso ao registo de varrimento interno (através do atributo *instruction_opcode*).

---

[79] Refira-se neste ponto que, tal como acontece com todos os ficheiros lidos, a extracção de informação é precedida pelo seu armazenamento na memória do computador e por um pré-processamento em que se normaliza o conteúdo do ficheiro, reduzindo todos os caracteres alfabéticos a letras minúsculas (o BSDL é insensível ao tipo de carácter, *case-insensitive*) e eliminando todos os caracteres que sejam opcionais, nomeadamente do tipo espaços ou tabulações.

[80] Por exemplo, uma célula BS completa, i.e. com um andar de captura / deslocamento e um andar de retenção, é designada por BC_1 [Ble93, capítulo 4].

No final deste processo, a estrutura de informação de nós do circuito terá um formato idêntico ao ilustrado na Figura 7-8.

```
Nodes
_________________           _________________
id: clock1                  id: strobe
type: input                 type: output
boundary_cell: 37, BC_4     boundary_cell: 22, BC_1
_________________           _________________
id: sel_d3                  id: shutdown
type: input                 type: output
boundary_cell: 36, BC_4     boundary_cell: 21, BC_1
...                         ...
```

Figura 7-8: Extracto da estrutura de dados interna referente aos nós do circuito, após a leitura do ficheiro BSDL, para o exemplo de um único CI.

O ficheiro com as opções e informação dadas pelo utilizador é o segundo a ser lido pela ferramenta de GAPTD. Nele se identificam os pinos do CI que são acessíveis através do registo de varrimento externo[81] e a correspondência com o número de ordem das células BS do(s) componente(s) utilizados(s). No final deste processo, a estrutura de informação de nós do circuito é expandida, tal como se ilustra na Figura 7-9. Refira-se ainda que é extraída deste ficheiro informação sobre: i) a ligação do pino de relógio do CI (ligado / não ligado ao pino System_clk do PRODEP); ii) o período do relógio funcional do CI, utilizado durante a última simulação; e iii) o período do relógio funcional do PRODEP. A leitura do ficheiro seguinte, referente à descrição das cadeias de varrimento internas do CI, permite completar a estrutura de informação de nós do circuito, com os dados correspondentes aos elementos sequenciais que formam cada um destes registos, incluindo a sua numeração. A Figura 7-10 ilustra o conteúdo da estrutura de dados no final da leitura deste ficheiro.

A estrutura de informação de nós do circuito constitui assim uma base de informação completa sobre a acessibilidade (em termos de controlabilidade / observabilidade) dos nós periféricos e internos do circuito, independente de qualquer forma ou tipo de simulação. A informação relativa ao ficheiro de simulação é armazenada numa segunda estrutura de dados, denominada canais de simulação, ou simplesmente *Channels*. A leitura deste ficheiro permite, num primeiro passo, criar um elemento por cada identificador (*id*) presente nas secções de entradas (*inputs*),

[81] De uma forma análoga à ilustrada na Figura 7-3, pode-se rodear os pinos do CI sob depuração de um conjunto de células BS externas. Este processo é porém redundante e desnecessário quando o CI suporta a instrução opcional *INTEST*, com as células do registo periférico a serem capazes de aplicar valores nas entradas da lógica funcional e de capturar valores nas saídas dessa mesma lógica, supondo que todas as células são completas, ou seja, do tipo BC_1.

saídas (*outputs*) e elementos sequenciais internos (*buried*) acessíveis por varrimento, tal como se ilustra na Figura 7-11. Nesta estrutura de informação é relevante a ordem pela qual aparecem os elementos, contrariamente ao que sucede na estrutura de informação referente aos nós do circuito. Concretamente, deve respeitar-se a ordem pela qual os identificadores surgem no ficheiro com o resultado da simulação, dado que esta traduz a forma como os canais de simulação estão organizados na tabela / matriz de valores, que geralmente constitui o corpo do ficheiro. A estrutura *Channels* constitui assim uma guia para a decifração de cada linha (horizontal) da tabela de vectores de simulação: à medida que se vão extraindo valores de uma dada linha (em bits individuais ou valores hexadecimais, posteriormente convertidos em valores binários), esta estrutura vai permitindo identificar inequivocamente qual o canal correspondente a cada bit. Uma vez identificado um dado canal de simulação, a estrutura de informação de nós do circuito permite determinar se este corresponde a um nó acessível ao PRODEP e, se sim, de que forma é acessível.

```
Nodes
_________________           _________________
id: clock1                  id: strobe
type: input                 type: output
boundary_cell: 37, BC_4     boundary_cell: 22, BC_1
external_cell: 28           external_cell: 9
_________________           _________________
id: sel_d3                  id: shutdown
type: input                 type: output
boundary_cell: 36, BC_4     boundary_cell: 21, BC_1
external_cell: 26           external_cell: 8
...                         ...
```

Figura 7-9: Extracto da estrutura de dados interna referente aos nós do circuito, após a leitura do ficheiro que contém as opções e informação definidas pelo utilizador, para o exemplo de um único CI.

```
Nodes
_________________   _________________   _________________
id: clock1          id: strobe          id: core\count1\cnt_bits1
type: input         type: output        type: buried
boundary_cell: 37, BC_4  boundary_cell: 22, BC_1  internal_cell: 24
external_cell: 28   external_cell: 9
_________________   _________________   _________________
id: sel_d3          id: shutdown        id: core\count1\cnt_bits2
type: input         type: output        type: buried
boundary_cell: 36, BC_4  boundary_cell: 21, BC_1  internal_cell: 25
external_cell: 26   external_cell: 8
...                 ...                 ...
```

Figura 7-10: Extracto da estrutura de dados interna referente aos nós do circuito, após a leitura do ficheiro de descrição das cadeias de varrimento internas, para o exemplo de um único CI.

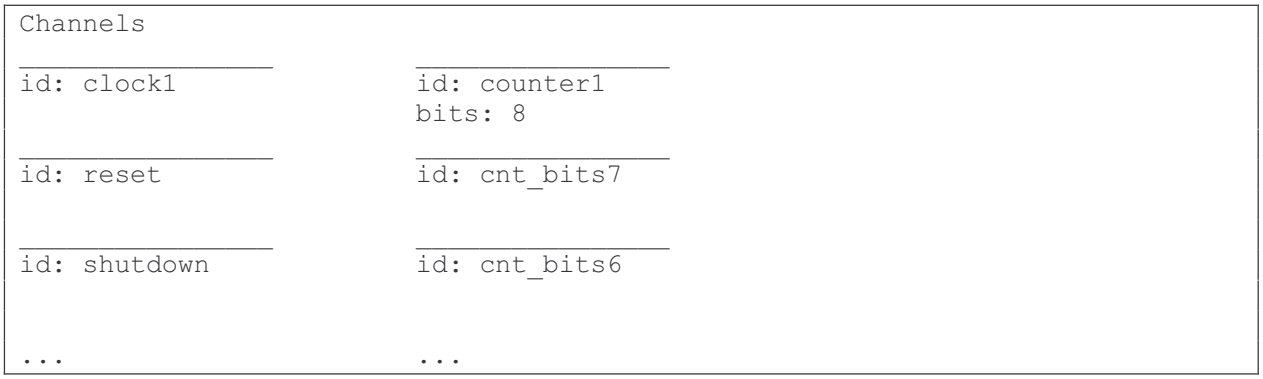


Figura 7-11: Extracto da estrutura de dados interna referente aos canais de simulação, após a leitura do ficheiro com o resultado da simulação, para o exemplo de um único CI.

Refira-se neste ponto que, no caso do identificador corresponder a um grupo de canais de simulação (por exemplo, `id: counter1` na Figura 7-11), é necessário expandi-lo, ou seja, registar no próprio elemento o número de canais individuais que o constituem, e inserir posteriormente nas posições imediatamente seguintes, os novos elementos identificadores desses canais (`id: cnt_bits7, …, id: cnt_bits0,` para o caso ilustrado na Figura 7-11).

Criação dos programas (de teste e depuração) a executar pelas unidades CLT e CLF

A fase seguinte da GAPTD consiste basicamente na leitura linha-a-linha do corpo do ficheiro com o resultado da simulação (i.e. da tabela com os valores estipulados / obtidos para cada canal de simulação) e posterior escrita nos ficheiros de saída, que contêm o programa a executar pela unidade CLF e pela unidade CLT. O código gerado levará o PRODEP a verificar os valores indicados para os canais correspondentes a pinos de saída e a elementos sequenciais, e ainda a aplicar os valores indicados nos canais correspondentes a pinos de entrada, para todos aqueles que são acessíveis por varrimento ou directamente, através dos pinos de entrada / saída genéricos da unidade CLF. Na verificação funcional são apenas seleccionadas as linhas do ficheiro de simulação em que ocorre uma transição ascendente do relógio funcional, que se podem detectar com base na identificação, fornecida pelo utilizador, do pino de relógio funcional.

Para cada valor (em binário ou hexadecimal) extraído de uma dada linha da tabela de vectores, a estrutura de dados referente aos canais de simulação permite identificar o correspondente canal ou grupo de canais, e para cada canal identificado, a estrutura de dados referente aos nós do circuito permite saber se existe um nó correspondente e qual a sua acessibilidade. Deste modo, a leitura de uma linha da tabela de vectores de simulação resulta no preenchimento (em memória) de um conjunto de vectores binários, que servem posteriormente como base para a obtenção dos argumentos das instruções de deslocamento, no código a gerar para a unidade CLT do PRODEP. Os vectores binários utilizados são dimensionados de acordo com a

informação previamente reunida sobre os comprimentos das várias cadeias de varrimento. Na Figura 7-12 ilustram-se os vectores binários obtidos após a leitura de uma linha do corpo do ficheiro com o resultado da simulação, para o exemplo de um único CI que suporte a instrução opcional *INTEST* – em que os valores referentes aos seus pinos podem ser aplicados / capturados através das células do registo BS do próprio CI.

```
registo de varrimento periférico (BS)
vector a deslocar:
10010000101000011000000001111110111111
vector esperado:
00000000000000011000000001111110111111
máscara:
00000000000000011111111111111111111111

registo de varrimento interno       .
vector esperado:
00000000000000000000000000
máscara:
11111111000000000000011110
```

Figura 7-12: Construção dos argumentos das instruções de deslocamento da unidade CLT, com base na leitura das linhas do ficheiro com o resultado da simulação e da informação contida nas estruturas de dados internas da ferramenta de GAPTD.

Em todos os vectores o bit mais à direita corresponde à célula BS identificada com o número zero (0), ou seja, a mais próxima de TDO. No caso do registo BS do CI sob depuração, os valores esperados nas saídas são copiados para o vector de valores a deslocar, o que permite emular o correcto funcionamento do CI durante a execução do programa de teste e depuração. Isto significa que, após a detecção de uma diferença entre um valor esperado (obtido por simulação) e um valor capturado (obtido durante o funcionamento real do circuito), o valor que será colocado no pino de saída do CI será igual ao que foi deslocado para a célula BS a ele associada, ou seja, ao valor obtido por simulação.

A transição entre estados do CI, correspondente à aplicação de um impulso no seu relógio funcional, é obtida através do pino System_clk da unidade CLF, ou a partir da própria instrução *INTEST*, dependendo das capacidades implementadas no CI. Refira-se, em seguimento do que já foi descrito anteriormente, que o controlo das operações a efectuar (leitura dos valores, aplicação de um impulso de relógio e aplicação dos valores) são sempre da responsabilidade da unidade CLF, uma vez que é esta unidade que controla o sincronismo com a unidade CLT. O facto da responsabilidade do arranque das acções pertencer à unidade CLF, não invalida porém que ambas as unidades trabalhem em paralelo, ou mesmo que a unidade CLT execute um maior número de operações, como aliás acontece no exemplo apresentando.

7.2 Sistemas baseados em microprocessadores

A existência de uma infraestrutura de teste no interior do próprio microprocessador facilita a aplicação das tarefas de depuração que constam do modelo apresentado no capítulo 2. É difícil no entanto estabelecer uma correspondência precisa entre tipos de microprocessadores e tipos de infraestruturas de teste, dado o panorama heterogéneo que se verifica nesta área, tema aliás já abordado no capítulo 3. Considera-se pois que um microprocessador poderá:

- Não possuir qualquer infraestrutura de teste.
- Possuir a infraestrutura mínima descrita na norma IEEE 1149.1.
- Suportar uma ou várias das instruções opcionais igualmente definidas na norma anterior.
- Suportar um subconjunto ou o conjunto total das instruções opcionais definidas no capítulo 5 deste documento.

Nesta secção analisa-se a implementação das técnicas tradicionais de depuração baseadas na execução passo-a-passo, na aplicação de pontos de paragem por condição, ou na aplicação de técnicas de amostragem em tempo real, considerando o caso de o microprocessador suportar o conjunto das instruções opcionais definidas neste documento (a última das quatro situações referidas acima). Não se consideram os casos em que o microprocessador já possui capacidades de apoio à depuração, acessíveis através da sua infraestrutura de teste, dado que constituem um exemplo já consumado da implementação de uma metodologia de projecto para o teste e depuração. Serão ainda pontualmente referidos os casos em que: i) o microprocessador não possui qualquer tipo de infraestrutura de teste; ii) possui a infraestrutura BST mínima; ou iii) suporta algumas das instruções opcionais definidas na norma IEEE 1149.1. Estes casos serão considerados nos momentos em que for importante realçar o modo como o PRODEP comanda a execução ou implementação da técnica de depuração em análise.

7.2.1 Identificação da informação de entrada e de saída

A informação de entrada da ferramenta de GAPTD, para o caso da depuração de sistemas baseados em microprocessadores, consiste fundamentalmente na informação referida na secção anterior, para o caso de uma CCI. Repare-se que se assume implicitamente que este tipo de sistemas se baseia na interligação, numa carta, entre o microprocessador, a memória com o programa a executar e os vários componentes que interagem com o microprocessador. Recentemente, tem-se assistido à migração deste tipo de sistemas para o interior de um único CI. Nesta situação, o microprocessador surge como um macrobloco do CI, pelo que apenas se referirá este nível hierárquico na conclusão do capítulo.

A informação de entrada é pois constituída, para o nível da CCI, por:

- Ficheiros BSDL de todos os componentes BST.
- Descrição das ligações entre os componentes inseridos na CCI (*Netlist*).
- Ficheiro com o resultado da simulação funcional do sistema (neste caso, da CCI). Deve ter-se em atenção quais os canais de simulação existentes ou seleccionados. Considera-se ainda a existência de uma base de dados com os ficheiros de simulação individual de cada um dos componentes inseridos no sistema.
- Descrição das cadeias de varrimento internas de cada componente. Considera-se a existência de uma base de dados com este tipo de ficheiros.
- Ficheiro que reúne todas as opções e informações introduzidas pelo utilizador.
- Ficheiro que contém os vectores (ou o programa) de teste estrutural da CCI. Estes ficheiros são produzidos por uma ferramenta externa de geração automática de programas de teste e posteriormente integrados no programa executado pelo PRODEP.

Cada uma destas componentes de informação de entrada foi já anteriormente descrita em detalhe, tendo sido referido qual a sua contribuição para a estrutura de informação interna da ferramenta de GAPTD. A informação de saída consiste mais uma vez nos programas, a executar pelas unidades CLF e CLT do PRODEP, que permitem implementar cada uma das operações de depuração a seguir analisadas.

7.2.2 Implementação de operações passo-a-passo

A implementação de operações passo-a-passo consiste basicamente na aplicação controlada de impulsos de relógio ao sistema sob depuração. Com base na análise efectuada no capítulo 3 e com base nas capacidades do PRODEP apresentadas no capítulo 6, pode-se implementar este tipo de operação de duas formas, tal como se ilustra na Figura 7-13:

- Aplicação de *n* impulsos de relógio ao microprocessador, através da ligação do seu pino de entrada de relógio (clk_in) ao pino System_clk do PRODEP, e posterior utilização da instrução CLK ou CLK_N, tal como se ilustra na Figura 7-13a).
- Utilização da infraestrutura de teste do microprocessador, controlada através de um dos TAP da unidade CLT do PRODEP, para aplicar impulsos na entrada funcional de relógio, através da célula BS associada ao respectivo pino, ou através de um circuito de geração de relógio que suporte a instrução opcional *INTEST*, tal como se ilustra na Figura 7-13b).

Estas duas formas de implementação são seguidamente descritas com maior pormenor.

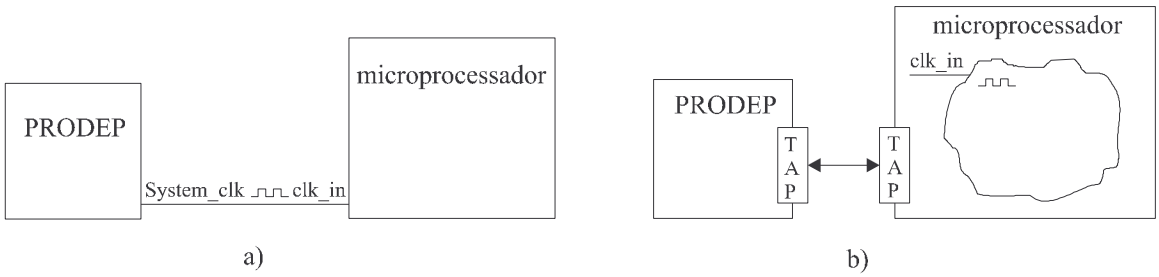


Figura 7-13: Ligação e utilização do PRODEP para implementação de operações passo-a-passo.

Aplicação de *n* impulsos de relógio através do pino System_clk do PRODEP

A execução de um único passo implica o conhecimento de quantos ciclos de relógio são necessários para completar um ciclo-máquina do microprocessador. Esta informação deve constar do ficheiro que reúne as opções e informações entradas pelo utilizador, de modo a que a ferramenta de GAPTD a possa converter em código a executar pelas duas unidades do PRODEP. A Figura 7-14 ilustra um exemplo do código gerado para o caso de um microprocessador, sem infraestrutura de teste, em que um ciclo-máquina corresponde a doze impulsos de relógio. A mesma figura inclui o código gerado para a unidade CLT, na situação em que se pretende, no final da execução da operação de passo, capturar os valores existentes num conjunto de ligações do sistema sob depuração, através da infraestrutura de teste dos componentes BST que possuam pinos pertencentes a essas ligações. Assume-se neste caso que esses componentes foram anteriormente carregados com a instrução *SAMPLE / PRELOAD*.

```
Unidade CLF                                  Unidade CLT

...                                          ...

CLK_N 12        ; executa um passo           WSA1        ; espera por pedido
SSA1            ; pede à unidade CLT         SSA1        ; responde
WSA1            ; para capturar valores      TMS0        ; Capture-DR
WSA0            ; espera por fim             TMS0        ; Shift-DR
SSA0            ; de tarefa                  LD C16, <comprimento_da_cadeia>
WAIT_WHL_Z   2  ; espera que seja dada       NCSHF <…>   ; guarda vectores
WAIT_WHL_NZ  2  ; ordem para nova            TMS1        ; Update-DR
                ; operação                   TMS1        ; Select-DR
...                                          SSA0        ; fim de tarefa
                                             WSA1        ; espera por pedido

                                             ...
```

Figura 7-14: Extracto do código a executar pelas unidades CLF e CLT, para implementação de uma operação passo-a-passo, seguida pela observação dos valores existentes num conjunto de ligações.

Utilização da infraestrutura BST do microprocessador

O modo de utilização da infraestrutura BST do microprocessador, para implementar uma operação passo-a-passo, foi já analisada no capítulo 3. Pretende-se neste ponto exemplificar a forma como o PRODEP comanda a infraestrutura de teste do microprocessador, supondo que esta suporta a instrução opcional *INTEST*, de forma a completar uma operação passo-a-passo. Refira-se que a forma de obter internamente o relógio para a lógica funcional do microprocessador pode variar entre componentes, sendo possível encontrar quatro exemplos na norma IEEE 1149.1 [IEEE93, págs. 7-16 a 7-19[82]]. No caso ilustrado na Figura 7-15 assume-se que é necessário aplicar doze impulsos no relógio de teste enquanto o controlador do TAP do microprocessador se encontrar no estado *Run-Test/Idle*[83]. Esta e outras informações deverão constar do ficheiro que reúne as opções e informação entradas pelo utilizador.

```
Unidade CLT

...

WSA1          ; espera por pedido
SSA1          ; responde
TMS1          ; Select-DR, supondo Run-Test/Idle como estado anterior
TMS1          ; Select-IR
TMS0          ; Capture-IR
TMS0          ; Shift-IR
LD C16, <comprimento_dos_registos_de_instrução>
NSHF     <vector que contém o código da instrução INTEST do microprocessador
         e da instrução BYPASS para os restantes componentes>
TMS1          ; Update-IR
TMS0          ; Run-Test/Idle
LD C24, 12
NTCK          ; executa um passo
SSA0          ; fim de tarefa
WSA1          ; espera por novo pedido

...
```

Figura 7-15: Extracto do código a executar pela unidade CLT, para implementação de uma operação passo-a-passo, através da utilização da instrução opcional *INTEST*.

---

[82] "*Typically, the on-chip system logic will receive a sequence of clock events between application of the stimulus and capture of the responses such that single-step operation is achieved. The specification of BS cells for system clock input pins allows the clocks for the on-chip system logic to be obtained in several ways while the* INTEST *instruction is selected. The following are offered as examples: 7.8.2(a) through 7.8.2(d).*"

[83] Corresponde à situação prevista em 7.8.2(b) – ver nota de rodapé anterior.

7.2.3 Implementação em tempo real de pontos de paragem por condição

A implementação em tempo real de pontos de paragem por condição pressupõe que a infraestrutura de teste do microprocessador, ou de outros componentes BST do sistema, suporta as instruções opcionais *SELCOND* e *COMP*. De acordo com este modo de funcionamento opcional descrito no capítulo 5, a detecção de condições é efectuada através do registo BS e sinalizada para o exterior via SCD. Este pino opcional deverá ser ligado a uma das entradas genéricas do PRODEP, de forma a que no momento em que for detectada uma condição verdadeira (SCD = nível lógico '1'), seja imediatamente interrompido o fornecimento de impulsos de relógio ao sistema sob depuração. Esta funcionalidade corresponde precisamente à instrução CLK_WHL_Z [i] da unidade CLF, descrita no capítulo anterior.

A Figura 7-16 e a Figura 7-17 ilustram duas possíveis situações de implementação de pontos de paragem por condição, em tempo real, num sistema baseado num microprocessador. A primeira situação, ilustrada na Figura 7-16, corresponde à existência de uma infraestrutura de teste no microprocessador, que suporta as duas instruções BST opcionais e o pino opcional SCD. Este encontra-se ligado a uma das entradas genéricas do PRODEP, que por sua vez fornece impulsos de relógio ao microprocessador, através do pino System_clk (ligado ao pino clk_in). Repare-se que nesta situação se restringem as condições a vectores definidos exclusivamente pelos valores presentes nos pinos do microprocessador.

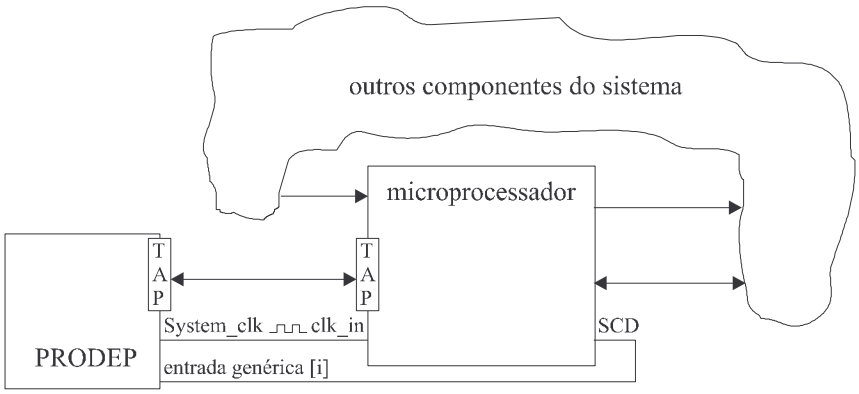


Figura 7-16: Ligação do PRODEP ao microprocessador para implementação de operações de pontos de paragem por condição.

A segunda situação, ilustrada na Figura 7-17, corresponde à existência de uma infraestrutura de teste nos vários componentes que rodeiam totalmente o microprocessador. Estes componentes suportam as duas instruções BST opcionais e os pinos opcionais ECD e SCD, que se encontram ligados em cadeia, de forma que o pino SCD do último componente da cadeia se

encontra ligado a um dos pinos de entradas genéricas do PRODEP. Este, por sua vez, fornece impulsos de relógio ao microprocessador, através do pino System_clk (ligado ao pino clk_in). Repare-se que nesta situação as condições podem estender-se a pinos não directamente ligados ao microprocessador, embora todos os pinos deste se encontrem cobertos, em termos de possibilidade de especificar condições que englobem valores neles presentes. A situação ideal corresponde à existência de uma infraestrutura de teste que suporte as duas instruções opcionais, em todos os componentes do sistema, incluindo o microprocessador, dado que possibilita a especificação de condições formadas com base nos valores presentes em qualquer combinação de pinos (ou ligações) do sistema. Nesta situação, todos os pinos ECD – SCD, dos vários componentes, seriam ligados em cadeia, de forma análoga à ilustrada na Figura 7-17.

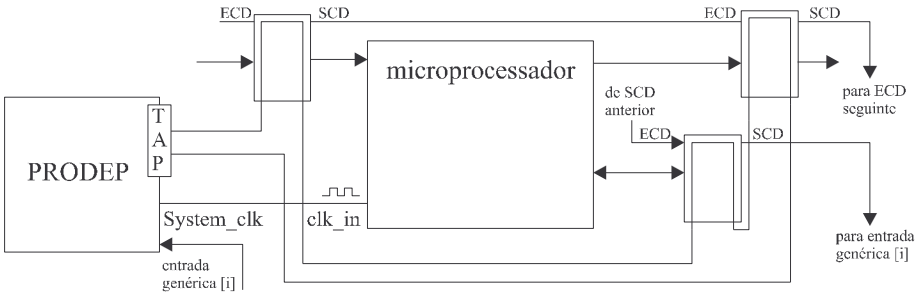


Figura 7-17: Ligação do PRODEP aos componentes BST que interagem com o microprocessador, para implementação de operações de pontos de paragem por condição.

Com base no esquema ilustrado na Figura 7-16, apresenta-se em seguida um exemplo do código a executar pelas duas unidades, por forma a implementar um ponto de paragem por condição, neste caso referente ao aparecimento de um determinado vector no barramento de endereços do microprocessador. Assume-se que o pino SCD do microprocessador se encontra ligado à entrada genérica 3 do PRODEP. A Figura 7-18 e a Figura 7-19 ilustram o código a executar pela unidade CLF e pela unidade CLT, respectivamente, para o exemplo proposto.

```
Unidade CLF

...
SSA1            ; pede à unidade CLT para deslocar para o interior do
WSA1            ; registo BS do microprocessador, o vector que especifica a
WSA0            ; detecção do vector pretendido no barramento de endereços
SSA0            ; espera por fim de tarefa
CLK_WHL_Z    3  ; fornece impulsos de relógio enquanto condição for falsa
SSA1            ; ,ou seja, enquanto SCD = '0'
...
```

Figura 7-18: Extracto do código a executar pela unidade CLF, para implementação de uma operação de ponto de paragem por condição, detectada em tempo real através da infraestrutura BST e sinalizada para o exterior através do pino opcional SCD.

```
Unidade CLT
...
WSA1          ; espera por pedido
SSA1          ; responde
TMS0          ; Capture-IR supondo Select-IR como estado anterior
TMS0          ; Shift-IR
LD C16, <comprimento_dos_registos_de_instrução>
NSHF    <vector que contém o código da instrução SELCOND do microprocessador
         e da instrução BYPASS para os restantes componentes>
TMS1          ; Update-IR
TMS1          ; Select-DR
TMS0          ; Capture-DR
TMS0          ; Shift-DR
LD C16, <comprimento_reg_STC_microprocessador + registos_BP_restantes_comp.>
NSHF    <vector que selecciona condição do tipo IGUAL, comparação através de
         máscara, para o microprocessador, '1' para restantes componentes>
TMS1          ; Update-DR
TMS1          ; Select-DR
TMS1          ; Select-IR
TMS0          ; Capture-IR
TMS0          ; Shift-IR
LD C16, <comprimento_dos_registos_de_instrução>
NSHF    <vector que contém o código da instrução SAMPLE/PRELOAD do micro-
        processador e da instrução BYPASS para os restantes componentes>
TMS1          ; Update-IR
TMS1          ; Select-DR
TMS0          ; Capture-DR
TMS0          ; Shift-DR
LD C16, <comprimento_registo_BS_processador + registos_BP_restantes_comp.>
NSHF    <vector que coloca valor pretendido nas células BS associadas aos
        pinos do barramento de endereços do microprocessador, '1' para as
        restantes células do registo BS e registos BP dos restantes comp o-
        nentes>
TMS1          ; Update-DR
TMS1          ; Select-DR
TMS1          ; Select-IR
TMS0          ; Capture-IR
TMS0          ; Shift-IR
LD C16, <comprimento_dos_registos_de_instrução>
NSHF    <vector que contém o código da instrução COMP do microprocessador e
        da instrução BYPASS para os restantes componentes>
TMS1          ; Update-IR
TMS1          ; Select-DR
TMS0          ; Capture-DR
TMS0          ; Shift-DR
LD C16, <comprimento_reg_BS_microprocessador + registos_BP_restantes_comp.>
NSHF    <vector que coloca máscara activa '1' nas células BS associadas aos
        pinos do barramento de endereços do processador, '0' para as resta n-
        tes células do registo BS e registos BP dos restantes comp onentes>
TMS1          ; Update-DR
TMS0          ; Run-Test/Idle - SCD passa a exibir resultado da comparação
SSA0          ; fim de tarefa
WSA1          ; espera por novo pedido
...
```

Figura 7-19: Extracto do código a executar pela unidade CLT, para implementação de uma operação de ponto de paragem por condição, detectada em tempo real através da infraestrutura BST e sinalizada para o exterior através do pino opcional SCD.

Repare-se que para um exemplo referente ao caso ilustrado na Figura 7-17, é apenas necessário modificar o código a executar pela unidade CLT. A ferramenta de GAPTD é neste caso responsável pela criação dos respectivos vectores a deslocar para o interior da cadeia, com base na descrição estrutural da CCI (*Netlist*) – para identificação dos pinos dos componentes BST ligados aos pinos do barramento de endereços do microprocessador – e nos ficheiros BSDL – para identificação das células BS associadas a esses mesmos pinos.

7.2.4 Implementação de operações de amostragem em tempo real

A implementação de operações de amostragem em tempo real pressupõe que a infraestrutura de teste do microprocessador, ou de outros componentes BST do sistema, suporta uma ou várias das instruções opcionais definidas no capítulo 5 para este efeito, nomeadamente:

- *MSEQ* – para memorizar sequências de dois vectores contíguos no registo BS.
- *MSAPC* – para memorizar sequências de dois vectores contíguos no registo BS, após condição detectada através do próprio registo BS, ou do pino opcional ECD.
- *MSATC* – para memorizar sequências de dois vectores contíguos no registo BS, até condição detectada através do pino opcional ECD.
- *MSEQ1P* – para memorizar sequências de *n* bits contíguos no registo BS, capturados num único pino funcional.

Nestes modos de funcionamento opcionais, o registo BS funciona como registo de armazenamento das amostras capturadas nas entradas paralelas das células que o compõem, e que se encontram associadas aos pinos de entrada, ou às saídas da lógica funcional do componente. Desta forma, é o relógio de teste (TCK) que determina o momento de amostragem, pelo que se torna necessário garantir o sincronismo com o relógio funcional do sistema. Este requisito é garantido através da ligação do pino de saída de relógio do PRODEP (System_clk) ao pino de entrada de relógio do microprocessador (clk_in), tal como se ilustra na Figura 7-16 e na Figura 7-17. Repare-se que se assume implicitamente que todas as acções do sistema baseado no microprocessador são síncronas com o seu relógio de entrada.

A necessidade de efectuar todas as operações em paralelo e em tempo real, salienta a vantagem de se ter optado, no caso do PRODEP, por uma arquitectura baseada em dois processadores, a trabalhar sincronamente em paralelo. Este facto ilustra-se com o exemplo que se apresenta a seguir, onde se assume que a infraestrutura de teste do microprocessador suporta a instrução opcional *MSAPC,* e o pino opcional SCD, e onde se pretende efectuar um registo das últimas vinte operações de escrita numa determinada posição (designada por XPTO) da memória de dados do microprocessador, e mais especificamente de quais os valores escritos nessa posição.

Começa-se por apresentar o código executado pela unidade CLF, onde se realça a existência de um ciclo numerado de um até vinte, que permite controlar o número de vezes em que foi detectada uma operação de escrita na posição XPTO da memória de dados do microprocessador. A detecção desta condição é sinalizada para o exterior do microprocessador através do seu pino SCD, que se encontra ligado à entrada genérica 3 do PRODEP. O funcionamento ininterrupto, em tempo real, do microprocessador, é garantido através da instrução START_CLK, que aplica continuamente impulsos na saída de relógio System_clk. O código a executar pela unidade CLF, nestas condições, é idêntico ao ilustrado na Figura 7-20.

```
Unidade CLF

...
        SSA1            ; pede à unidade CLT para deslocar para o registo BS
        WSA1            ; do microprocessador, o vector que especifica a
        WSA0            ; detecção do vector XPTO no barramento de endereços e
        SSA0            ; de uma operação de escrita na sua memória de dados
                        ; - espera por fim de tarefa -
        START_CLK       ; aplica impulsos contínuos em System_clk - Nota: esta
                        ; instrução poderia ter sido executada anteriormente
        SSA1            ; inicia amostragem - activa o relógio TCK -
                        ;  - ver código da unidade CLT -
        LD C24, 20      ; especifica um ciclo de 20
Ciclo   WAIT_WHL_Z 3    ; espera enquanto não é detectada a condição
                        ; especificada através do pino SCD do microprocessador
                        ; ligado à entrada genérica 3 desta unidade
        SSA0            ; fim de amostragem - pára  TCK
                        ; - ver código da unidade CLT -
        SSA1            ; pede à unidade CLT para deslocar para o exterior os
        WSA1            ; vectores armazenados, que deverão incluir
        WSA0            ; o dado escrito na posição XPTO. No final deverá ser
        SSA0            ; deslocado para o interior do registo BS a mesma
                        ; condição anterior
        SSA1            ; inicia amostragem - activa o relógio TCK -
                        ; ver código da unidade CLT
        DJNZ Ciclo      ; decrementa contador
        STOP_CLK        ; fim da operação pretendida
...
```

Figura 7-20: Extracto do código a executar pela unidade CLF, para implementação de uma operação de amostragem em tempo real, dos valores escritos numa determinada posição da memória de dados externa do microprocessador do sistema sob depuração.

O código a executar pela unidade CLT encontra-se ilustrado na Figura 7-21. Repare-se na existência de dois blocos distintos: um bloco inicial para especificação da condição de início de amostragem, delimitado no final pela instrução STCK, que permite efectuar as operações de captura / armazenamento após a detecção da condição, e um segundo bloco que permite a leitura dos vectores armazenados e a reentrada da condição inicialmente especificada.

```
Unidade CLT
...
WSA1          ; espera por pedido
SSA1          ; responde
- - - - - - - - - - - - - - - -  Código a repetir - - - - - - - - - - - - - - -
TMS0          ; Capture-IR supondo Select-IR como estado anterior
TMS0          ; Shift-IR
LD C16, <comprimento_dos_registos_de_instrução>
NSHF     <vector que contém o código da instrução SAMPLE/PRELOAD do micro-
         processador e da instrução BYPASS para os restantes componentes>
TMS1          ; Update-IR
TMS1          ; Select-DR
TMS0          ; Capture-DR
TMS0          ; Shift-DR
LD C16, <comprimento_reg_BS_microprocessador + registos_BP_restantes_comp.>
NSHF     <vector que coloca valor XPTO nas células BS associadas aos pinos do
         barramento de endereços do microprocessador; a combinação referente
         a uma operação de escrita na memória de dados, nas células BS ass o-
         ciadas aos pinos de controlo do processador; '1' para as restantes
         células do registo BS e registos BP dos restantes comp onentes>
TMS1          ; Update-DR
TMS1          ; Select-DR
TMS1          ; Select-IR
TMS0          ; Capture-IR
TMS0          ; Shift-IR
LD C16, <comprimento_dos_registos_de_instrução>
NSHF     <vector que contém o código da instrução MSAPC do microprocessador e
         da instrução BYPASS para os restantes componentes>
TMS1          ; Update-IR
TMS0          ; Run-Test/Idle
SSA0          ; indica fim de tarefa
WSA0          ; espera por confirmação

STCK          ; estado de detecção e captura após ocorrência da condiçã o

WSA1          ; espera por pedido
SSA1          ; responde
TMS1          ; Capture-DR
TMS0          ; Shift-DR
LD C16, <comprimento_reg_BS_microprocessador + registos_BP_restantes_comp.>
NSHFCP   <desloca e armazena na memória FIFO o primeiro vector capturado. T o-
         dos os bits da máscara deverão estar a '0' - comparação inactiva. O
         valor deslocado para o interior é irrelevante>
TMS0          ; Pause-DR
TMS1          ; Exit2-DR - coloca segundo vector no andar de deslocamento
TMS0          ; Shift-DR
LD C16, <comprimento_reg_BS_microprocessador + registos_BP_restantes_comp.>
NSHFCP   <desloca e armazena na memória FIFO o segundo vector capturado. T o-
         dos os bits da máscara deverão estar a '0' - comparação inactiva. O
         valor deslocado para o interior é irrelevante>
TMS1          ; Update-DR
TMS1          ; Select-DR
TMS1          ; Select-IR
- - - - - - - - Repetir código delimitado por estas linhas - - - - - - - - -
...
```

Figura 7-21: Extracto do código a executar pela unidade CLT, para implementação de uma operação de amostragem em tempo real, dos valores escritos numa determinada posição da memória de dados do microprocessador do sistema sob depuração.

7.3 Sistemas híbridos

A depuração de um sistema híbrido depende mais uma vez de qual o nível hierárquico considerado. Para o caso de um CI, as partes relativas ao microprocessador e à lógica dedicada surgem como macroblocos, pelo que se coloca novamente a opção de remeter a análise deste nível hierárquico para a conclusão do capítulo. Para o caso de uma CCI, as partes relativas ao microprocessador e à lógica dedicada surgem no nível hierárquico imediatamente inferior, ou seja, como CI. Nesta situação colocam-se novamente todas as questões relativas à existência, tipo e capacidades da infraestrutura de teste de cada um dos componentes da carta, incluindo microprocessador e lógica dedicada. A regra a seguir consiste em garantir para cada componente, dentro dos limites impostos pelos requisitos a que deve obedecer o sistema – incluindo o requisito da verificação -, o suporte dos modos de funcionamento opcionais definidos no capítulo 5.

A aplicação da metodologia de depuração definida para o caso dos sistemas baseados em lógica dedicada depara com algumas dificuldades, nomeadamente ao nível da simulação, dada a complexidade, não só da lógica dedicada como também da aplicação executada pelo microprocessador. É no entanto plausível[84] que se possa simular o funcionamento de todo o sistema para pequenos intervalos de operação, que embora não garantam uma completa cobertura da funcionalidade do sistema, poderão activar algumas das funções principais. Com base nos resultados destas pequenas sessões de simulação, pode-se aplicar a metodologia estruturada definida na primeira secção para o caso dos sistemas baseados em lógica dedicada. Repare-se que o teste estrutural do sistema beneficia da existência de uma infraestrutura de teste em todos os componentes, pelo que se deve tentar garantir este requisito, na medida do possível. A depuração da aplicação executada pelo microprocessador segue novamente a metodologia proposta na segunda secção deste capítulo, para o caso dos sistemas baseados em microprocessadores. Esta metodologia baseia-se principalmente na utilização de um processo dedutivo do tipo tentativa – erro, com técnicas tradicionais do tipo execução passo-a-passo, aplicação de pontos de paragem por condição e técnicas de amostragem em tempo real. O suporte das instruções opcionais definidas no capítulo 5, principalmente por parte da infraestrutura de teste do microprocessador e da lógica dedicada, permite uma melhor verificação da interacção entres estes dois elementos do sistema sob depuração.

---

[84] Na condição de existir um modelo para cada um dos componentes do sistema. Para o caso da lógica dedicada, é provável que assim seja, dado que no desenvolvimento de sistemas híbridos se efectua geralmente em paralelo o desenvolvimento da lógica dedicada e da aplicação a executar pelo microprocessador.

O caso dos sistemas híbridos surge naturalmente como o mais complexo do ponto de vista da depuração. É frequente neste tipo de sistemas proceder-se ao desenvolvimento da aplicação a executar pelo microprocessador, em paralelo com o desenvolvimento do CI que congrega a lógica dedicada, resultando na maior parte das vezes em especificações iniciais incompletas e no deslocamento das alterações a introduzir, nas fases mais adiantadas do desenvolvimento, para o lado da aplicação, dada a versatilidade associada à escrita de código. Esta facto origina no entanto algumas complicações do ponto de vista da verificação, dado que a depuração do código segue uma metodologia não estruturada, em contraste com a depuração do funcionamento da lógica dedicada, que segue uma metodologia estruturada baseada na utilização intensiva de informação, passada entre as várias etapas de depuração – a iniciar na simulação funcional e a terminar na verificação temporal, passando pela simulação temporal, simulação da verificação funcional, teste estrutural e verificação funcional.

7.4 Conclusão

Neste capítulo descreveu-se a metodologia de teste e depuração, bem como a geração automática do respectivo programa, para três tipos de sistemas: i) baseados em lógica dedicada; ii) baseados em microprocessadores; e iii) híbridos.

Para os sistemas baseados em lógica dedicada consideraram-se os níveis hierárquicos do CI e da CCI. O nível hierárquico do macrobloco constitui actualmente um campo de análise e investigação intensiva, dada a sua crescente utilização no desenvolvimento dos CI de maior complexidade. A definição de infraestruturas de teste, ou simplesmente de infraestruturas de acesso, para inserção nos macroblocos, constitui o objectivo principal de um dos grupos de trabalho da VSI *Alliance*, sendo aguardados os primeiros resultados para muito brevemente. A adaptação dos modos de funcionamento opcionais, definidos no capítulo 5, à infraestrutura de teste adoptada para o caso dos macroblocos, pode ajudar na tarefa da verificação do CI onde este se encontra embutido, principalmente por garantir o acesso electrónico às suas entradas / saídas, para efeitos de detecção de condições e de amostragem de valores em tempo real. Este acesso traz maiores vantagens nos casos em que coexistem no mesmo CI macroblocos dos tipos lógica dedica e microprocessador, uma vez que permite verificar a interacção entre estes dois tipos de componentes, durante a realização da aplicação executada pelo microprocessador. Repare-se que esta última situação corresponde a um sistema híbrido para o nível hierárquico imediatamente superior, ou seja, ao nível do CI. O nível hierárquico do módulo multi-pastilha constitui um nível intermédio entre os níveis do CI e da CCI. A utilização do modo de compatibilidade com a norma IEEE 1149.1, permite considerar um módulo multi-pastilha como pertencente ao nível hierárquico do CI, nos casos em que a compatibilidade esteja activada, ou considerá-lo como pertencente ao nível hierárquico da CCI, nos casos em que a compatibilida-

de não esteja activada. Nesta última situação, o TAP do módulo funciona como porto de acesso para a cadeia formada pelos TAP dos vários CI (não encapsulados) existentes no seu interior. Repare-se que se assume implicitamente que estes CI possuem uma infraestrutura de teste compatível com a norma IEEE 1149.1, que se encontra activa quando os pinos de selecção de compatibilidade do módulo se encontram inactivos. O caso dos sistemas formados pela interligação de várias CCI constitui o que apresenta maior dificuldade de análise, em virtude do número de cartas inseridas e da forma de interligação entre elas.

A utilização da norma IEEE 1149.5 [IEEE95], ou *Module Test and Maintenance* (MTM) – *bus*, reduz-se aos sistemas constituídos pela ligação num único barramento comum de várias CCI. Dado que a funcionalidade deste barramento inclui um modo de acesso a barramentos de teste de nível inferior, nomeadamente com as quatro (cinco) linhas que formam o TAP, pode-se idealizar esquemas, baseados na utilização do MTM-*bus*, para suporte de tarefas de ajuda à depuração deste tipo de sistemas. Estes esquemas não se inserem contudo no núcleo central do trabalho apresentado nesta tese, pelo que se consideram antes como uma direcção possível para trabalho futuro.

Capítulo 8

Conclusão e perspectivas de evolução

Nesta dissertação propôs-se uma solução de projecto para o teste e depuração baseada em três componentes principais:

- Novos modos de operação opcionais para as infraestruturas normalizadas IEEE 1149.1 e P1149.4, especificamente concebidos para apoio ao teste e depuração.
- Um controlador (residente) com uma arquitectura optimizada para as tarefas de teste e depuração.
- Uma metodologia e uma ferramenta para a geração automática do respectivo programa.

Os novos modos de operação contribuem para ultrapassar as lacunas das infraestruturas referidas, para a implementação das operações de depuração apresentadas no capítulo 2. Estas lacunas foram identificadas ao longo do capítulo 3, onde se analisou a utilização para este fim dos modos de operação básicos e opcionais definidos nas respectivas normas [Alv99e]. Nesta análise considerou-se igualmente a utilização de componentes comerciais que suportam modos de operação opcionais definidos pelo fabricante, com vista a melhorar a implementação de algumas das operações de depuração. Os modos de operação propostos implicam modificações mínimas, em número e tipo, relativamente à infraestrutura obrigatória definida na norma IEEE 1149.1, ou P1149.4 [Alv99b, Alv99c, Alv99f].

O controlador residente permite controlar a actividade da lógica de teste, em sincronismo com a actividade da lógica funcional do sistema, para implementar as operações de depuração. Esta dupla função (controlo da lógica de teste e sincronização com a lógica funcional) levou à definição de uma arquitectura interna baseada em dois processadores, sincronizados ao nível-máquina através de um relógio de entrada comum. Um dos processadores controla a lógica de teste através das linhas de dois TAP, activos alternadamente. O outro processador controla o fornecimento de impulsos de relógio à lógica funcional do sistema e responsabiliza-se pelas restantes operações de depuração [Alv99b, Alv99c].

O programa de teste e depuração (executado pelo controlador) é gerado automaticamente por uma ferramenta computacional, com base num conjunto de informação de entrada que inclui o modelo do sistema, os resultados da simulação, as características da infraestrutura de teste, vectores para o teste estrutural (importados a partir de outras ferramentas de geração a utomática) e alguma informação especificada pelo utilizador. A geração do programa obedece a uma metodologia que define a sequência de execução de acções (que poderão ocorrer em simultâneo) referentes à lógica funcional e à lógica de teste do sistema. Esta metodologia depende do tipo (sistemas baseados em lógica dedicada, em microprocessadores ou híbridos) e nível hierárquico (CI ou CCI) do sistema sob depuração.

8.1 Resultados obtidos

Os modos de operação opcionais propostos foram implementados numa infraestrutura para o teste e depuração, adicionada a um componente que controla um dispositivo electrónico para a comunicação aumentativa e alternativa [Fer94]. Esta infraestrutura suporta o seguinte conjunto de instruções:

- *EXTEST*, *SAMPLE / PRELOAD* e *BYPASS* (obrigatórias).
- *INTEST* (opcional, definida na norma).
- Uma instrução de acesso à cadeia de varrimento interna (opcional).
- *COMP*, *MSEQ*, *MSAPC*, *MSATC* e *MSEQ1P* (opcionais, definidas nesta tese).

A lógica funcional e a lógica de teste foram descritas utilizando um editor esquemático e a linguagem denominada AHDL (*Altera Hardware Description Language*). O ambiente de desenvolvimento MaxPlus® II da Altera[85] inclui ferramentas de geração de modelos (funcionais e comportamentais), simulação, síntese, colocação e ligação, e programação de componentes [Alt99]. O circuito descrito foi sintetizado para um dispositivo lógico programável complexo pertencente à família Flex 10K, nomeadamente o EPF10K10, que dispõe de aproximadamente 10.000 portas lógicas. Os ficheiros com a descrição do circuito implementado encontram-se em anexo (esquemáticos ilustrados no anexo em formato impresso e todos os ficheiros armazenados no anexo em formato electrónico, que inclui igualmente os ficheiros esquemáticos).

O controlador residente foi igualmente descrito em esquemático e em AHDL, tendo sido sintetizado para o EPF10K30, um dispositivo com uma capacidade de 30.000 portas lógicas e 4 Kbits de memória interna disponível para o utilizador. A complexidade do circuito sintetizado reparte-se de forma aproximadamente igual para cada um dos dois processadores que formam a arquitectura interna do controlador, representada no diagrama esquemático de topo que se ilu s-

[85] Um dos maiores fabricantes de ferramentas de desenvolvimento para FPGA e CPLD.

tra no anexo em formato impresso. Os ficheiros que descrevem os blocos internos de cada um dos processadores encontram-se armazenados no anexo em formato electrónico.

Para verificar o funcionamento do controlador e da infraestrutura para o teste e depuração, criou-se um modelo de topo formado pela interligação do modelo funcional do controlador residente, dos modelos das memórias com o seu programa e do modelo funcional do componente referido inicialmente. Este último modelo foi fornecido ao módulo de geração automática do programa de teste e depuração, em conjunto com a seguinte informação de entrada:

- Ficheiro com os resultados da simulação original do seu funcionamento (i.e. sem considerar o funcionamento da lógica de teste).
- Ficheiro com a descrição da infraestrutura de teste, em BSDL.
- Ficheiro com informação referente à cadeia de varrimento interna.
- Opções definidas pelo utilizador.

O modelo de topo e cada um dos ficheiros da lista anterior encontram-se armazenados no anexo em formato electrónico (o modelo de topo encontra-se ainda ilustrado no anexo em formato impresso), juntamente com os programas produzidos pelo módulo de geração automática, que são criados utilizando as mnemónicas *assembly* do conjunto de instruções de cada processador do controlador residente. Os ficheiros resultantes são convertidos para código-máquina com o auxílio do TASM e posteriormente transformados em ficheiros de inicialização dos modelos de memórias, por sua vez lidos pela ferramenta de simulação integrada no MaxPlus® II, que através do modelo de topo referido simula a execução do programa [Alv99a, Alv99d].

8.2 Perspectivas de desenvolvimento futuro

A automatização do processo de inserção da infraestrutura 1149.1, ou P1149.4, capaz de suportar os modos de operação opcionais propostos, é uma das perspectivas de evolução possíveis. Este processo pode ser efectuado em duas etapas: uma primeira em que se cria um modelo da infraestrutura para simulação funcional e uma segunda destinada à síntese do circuito completo (lógica de teste acoplada à lógica funcional). Para a criação do modelo é necessário dispor de informação acerca dos pinos individuais da lógica funcional. Exclui-se nesta primeira etapa a implementação de instruções de acesso a cadeias de varrimento internas, de execução de funções de auto-teste, ou outras similares, uma vez que apenas fazem sentido quando estes mecanismos estão presentes no circuito. Repare-se que é usual utilizar-se ferramentas de inserção automática destes mecanismos numa fase posterior, ou seja, quando o circuito se encontra descrito a um nível de abstracção mais baixo.

O estudo das potencialidades de utilização dos modos de operação propostos na implementação de tarefas não consideradas nesta tese, é outra das perspectivas de evolução que são possíveis. Os modos propostos tiveram por objectivo principal colmatar lacunas identificadas ao longo de um processo de análise específico, neste caso para as funções de depuração. Da mesma forma que as instruções obrigatórias podem ser utilizadas para objectivos diferentes dos inicialmente estabelecidos na norma IEEE 1149.1, ou P1149.4 (tal como se demonstra na análise efectuada ao longo do capítulo 3), também as instruções opcionais propostas podem ser utilizadas para outros fins, que poderão por sua vez levar à necessidade de implementar novos modos de operação opcionais. Repare-se que a análise efectuada partiu de um modelo com quatro tipos básicos de operações de depuração para circuitos digitais. A análise dos circuitos mistos, nomeadamente da parte analógica e da interface entre esta e a parte digital (ou vice-versa), deverá dar origem a operações de depuração complementares. A observação de outros níveis hierárquicos[86], como o dos macroblocos, ou o dos módulos multi-pastilha (*Multi-Chip Modules*, ou MCM), poderá ainda dar origem a novas linhas de desenvolvimento [Aba94, Bhat96, Fli94, Fli96, Murr96, Nys95, Oli96, Raj96], principalmente se se considerar a possibilidade de utilizar o modo de selecção de compatibilidade com a norma IEEE 1149.1 [Alv97b, Jar94].

Em relação ao controlador residente existem várias vias de desenvolvimento em aberto. A sua interligação e subordinação a um barramento de teste de nível hierárquico superior, por exemplo o MTM-*bus*, ou IEEE 1149.5, constitui uma destas vias. Neste sentido poderá ainda considerar-se a necessidade de desenvolver um controlador para este barramento de teste. Uma outra alternativa consiste em utilizar o 1149.1 em ambos os níveis hierárquicos (sistema e CCI), implementando-se em cada CCI um circuito de interface, baseado por exemplo no SN74ABT8996, ou no SCANPSC110F. Esta alternativa possibilita a reutilização do controlador desenvolvido no nível de topo.

A concluir, vale ainda a pena referir que as perspectivas de evolução apontadas para a infraestrutura de teste e depuração, e para o controlador residente, influenciam igualmente a metodologia e a geração automática do respectivo programa. Em relação a esta ferramenta há ainda diversas frentes em aberto, principalmente no que diz respeito à criação dos módulos de pré-processamento para a importação de informação de entrada proveniente de outras ferramentas computacionais.

[86] Em relação aos níveis hierárquicos principais considerados nesta dissertação (CI e CCI).

Referências

[Aba94] M. Abadir, A. Parikh, P. Sandborn, K. Drake e L. Bal, "Analyzing Multichip Module Testing Strategies," em *IEEE Design & Test of Computers*, pp. 40-52, Spring 1994.

[Abr90] Miron Abramovici, M. A. Breuer e A. D. Friedman, *Digital System Testing and Testable Design*, Computer Science Press, 1990, ISBN 0-7167-8179-4.

[Agr88] V. D. Agrawal e S. C. Seth, *Test Generation for VLSI Chips*, IEEE Computer Society Press, 1988, ISBN 0-8186-4786-8.

[Alt99] Altera Corporation, disponível em http://www.altera.com, 1999.

[Alv93] Gustavo R. Alves, M. G. Gericota, J. L. Ramalho e José M. M. Ferreira, "An HDL Approach to Board-Level BIST," em *Proceedings of the European Design Automation Conference*, pp. 410-415, IEEE Computer S ociety Press, 1993.

[Alv95] Gustavo R. Alves, A Testabilidade e o Teste de Cartas de Circuito Impresso com BST, Tese de Mestrado, FEUP, Jun. 1995.

[Alv96] Gustavo R. Alves e J. M. Martins Ferreira, Relatório nº1, Projecto JNICT PBIC/C/TIT/2474/95 (Ferramentas de Apoio ao Desenvolvimento de Sistemas que Incluam Componentes com BST), Out. 1996.

[Alv97a] Gustavo R. Alves, Daniel Aga, Ovidiu Mosuc e José M. M. Ferreira, "Debug and Test of Microcontroller Based Applications using the Boundary Scan Test Infrastructure," em *Digest of the Student Forum within the IEEE International Symposium on Industrial Electronics*, IEEE Industrial Electronics Society, 1997.

[Alv97b] Gustavo R. Alves e José M. M. Ferreira, "IEEE 1149.1 Compliance-enable Pin(s): A Solution for Embedded Microprocessor-based Systems Debug and Test," em *Digest of the 1st IEEE International Workshop on Testing Embedded Core-based Systems*, paper 4.3, 1997.

[Alv98a] Gustavo R. Alves e J. M. Martins Ferreira, Relatório nº2, Projecto JNICT PBIC/C/TIT/2474/95 (Ferramentas de Apoio ao Desenvolvimento de Sistemas que Incluam Componentes com BST), Fev. 1998.

[Alv98b] Gustavo R. Alves e J. M. Martins Ferreira, "Modern Techniques for System Design", Projecto Leonardo INSIGHT II, May 1998.

[Alv98c] Gustavo R. Alves e J. M. Martins Ferreira, "Techniques for Prototype Debugging and Validation", Projecto Leonardo INSIGHT II, June 1998.

[Alv98d] Gustavo R. Alves e J. M. Martins Ferreira, "Design for Debug and Test Techniques", Projecto Leonardo INSIGHT II, Oct. 1998.

[Alv99a] Gustavo R. Alves, Telmo Amaral e José M. M. Ferreira, "A Framework for System-level co-verification using the BST infrastructure," em *User's Forum in DATE*, pp. 233-236, 1999.

[Alv99b] Gustavo R. Alves, Telmo Amaral e José M. M. Ferreira, "A Built-In Controller and Extended BS architecture for Prototype Debug and Test," em *User's Forum in DATE*, pp. 153-158, 1999.

[Alv99c] Gustavo R. Alves, Telmo Amaral e José M. M. Ferreira, "Board-level Prototype Validation: A Built-in Controller and Extended BST Architecture," a ser publicado em *ISCAS*, IEEE Circuits and Systems Society Press, May 1999.

[Alv99d] Gustavo R. Alves, Telmo Amaral e José M. M. Ferreira, "A System Verification Strategy Based on the BST Infrastructure," a ser publicado em *ISCAS*, IEEE Circuits and Systems Society Press, May 1999.

[Alv99e] Gustavo R. Alves e José M. M. Ferreira, "From Design-for-Test to Design-for-debug-and-Test: Analysis of Requirements and Limitations for 1149.1," a ser publicado em *VLSI Test Symposium*, IEEE Computer Society Press, Apr. 1999.

[Alv99f] Gustavo R. Alves e José M. M. Ferreira, "Using the BS Register for Capturing and Storing n-bit Sequences in Real-Time," a ser publicado em *European Test Workshop*, May 1999.

[And94] John Andrews, "Using SCAN™ Bridge as an IEEE P1149.1 Protocol Addressable, Multi-Drop, Backplane Test Bus," em *International Test Conference*, p. 1019, IEEE Computer Society Press, 1994.

[Apt98] Aptix Corporation, "Aptix System Explorer MP3A," disponível em http://www.aptix.com/Products/mp3a.html, 1998.

[ARM99] Advanced RISC Machines Inc., disponível em http://www.arm.com/, 1999.

[Ass98] Asset-Intertech Inc., "Scanley's Handbook of Boundary-Scan Solutions," disponível em http://www.asset-intertech.com, 1998.

[Bal89] W. D. Ballew e L. M. Streb, "Board-Level Boundary Scan: Regaining Observability With an Additional IC," em *International Test Conference*, pp. 182-189, IEEE Computer Society Press, 1989.

[Bat85] J. Bateson, *In-Circuit Testing*, Van Nostrand Reinhold Company Inc., 1985, ISBN 0-442-21284-4.

[Ben84] R. G. Bennetts, *Design of Testable Logic Circuits*, Addison-Wesley Publishers Ltd, 1984, ISBN 0-8449-1385-0.

[Bhat96] Sandeep Bhatia, Tushar Gheewala e Prab Varma, "A Unifying Methodology for Intellectual Property & Custom Logic Testing," em *International Test Conference*, pp. 639-648, IEEE Computer Society Press, 1996.

[Bhav91] Dilip Bhavsar, "An Architecture for Extending the IEEE Standard 1149.1 Test Access Port to System Backplanes," em *International Test Conference*, pp. 768-776, IEEE Computer Society Press, 1991.

[Bla95] Thomas R. Blakeslee e Jan Liband, "Real-Time Debugging Techniques: Hardware-Assisted Real-Time Debugging Techniques for Embedded Systems," em *Embedded Systems Programming*, vol. 8, nº 4, 1995.

[Ble93] Harry Bleeker, Peter van Den Eijnden e Frans de Jong, *Boundary-Scan Test: A Practical Approach*, Kluwer Academic Publishers, 1993, ISBN 0-7923-9296-5.

[Bru91] W. C. Bruce, Michael G. Gallup, Grady Giles e Tom Munns, "Implementing 1149.1 on CMOS Microprocessors," em *International Test Conference*, pp. 879-886, IEEE Computer Society Press, 1991.

[CAD98] CAD-UL GmbH, "ROM-Monitor Debuggers: An alternative to in-circuit emulation?," disponível em http://www.cadul.de/embedded/public/romm.htm, 1998.

[CED98] CED, "Monitor ROMs," disponível em http://www.ced.co.uk/prog/progmru.htm , 1998.

[Che89] W. -T. Cheng, e T. J. Chakraborty, "Gentest: An Automatic Test-generation System for Sequencial Circuits," em *IEEE Computers*, vol. 22, nº 4, pp. 43-49, Apr. 1989.

[Che91] K-T Cheng, Srinivas Devadas e Kurtz Keutzer, "A Partial Enhanced-Scan Approach to Robust Delay-Fault Test Generation for Sequential Circuits," em *International Test Conference*, pp. 403-410, IEEE Computer Society Press, 1991.

[Coo98] Tony Coomes et al., "JTAG Aids Parallel Processor Projects," disponível em http://www.ahtcl.com/jtag.com, 1998.

[Cor87] J. M. Cortner, *Digital Test Engineering*, John Wiley & Sons, 1987, ISBN 0-471-85135-3.

[Cru94] Claude A. Cruz, "System-Level Boundary Scan Testing via the National Semiconductor SCANPSC110 SCAN™ Bridge," em *European Design & Test Conference*, pp. 203-206, IEEE Computer Society Press, 1994.

[Dea98] Lee Dearmin, "Performance Analysis Tools: An Overview," disponível em http://www.hmi.com/hmipac.html, 1998.

[Der91] B. Dervisoglu e G. Strong, "Design for Testability: Using Scanpath Techniques for Path Delay Test and Measurement," em *International Test Conference*, pp. 365-374, IEEE Computer Society Press, 1991.

[Eri95] Bruce Erickson, "Selecting the Right Debugging Tool," em *Electronic Design*, vol. 43, pp. 83-98, Oct. 1995.

[EST98] Embedded Support Tools Corp., "visionICE", disponível em http://www.estc.com/, 1998.

[Fen96] Billy Fenton, "Emulators Test Complex Boards," em Test & Measurement World magazine, pp. 37-40, Dec. 1996.

[Fer92a] J. M. Ferreira, F. S. Pinto e J. S. Matos, "Automatic Generation of a Single-Chip Solution for Board-level BIST of Boundary Scan Boards," em *Proceedings of the EDAC*, pp. 154-158, IEEE Computer Society Press, 1992.

[Fer92b] J. M. Ferreira, F. S. Pinto e J. S. Matos, "A Modular Architecture for BIST of Boundary Scan Boards," em *Proceedings of the EUROASIC Conference*, pp. 184-188, IEEE Computer Society Press, 1992.

[Fer92c] J. M. Ferreira, O Teste de Cartas de Circuito Impresso com BST: Arquitectura de um Controlador Residente, e Geração Automática do Programa de Teste, Tese de Doutoramento, FEUP, Abr. 1992.

[Fer93] J. M. Ferreira, M. G. Gericota, J. L. Ramalho e G. R. Alves, "BIST for 1149.1-Compatible Boards: A Low-Cost and Maximum-Flexibility Solution," em *International Test Conference*, pp. 536-543, IEEE Computer Society Press, 1993.

[Fer94] J. M. Ferreira, A. C. Teixeira, J. L. Ramalho e M. L. Lourenço, "A Portable Device to Improve the Communication Ability of People with Cerebral Palsy," em *Proceedings of the 6th Biennial Conference of the International Society for Augmentative and Alternative Communication* (ISAAC'94), pp. 446-448, 1994.

[Fli94] Andrew Flint, "Test Strategies for a Family of Complex MCMs," em *International Test Conference*, pp. 436-445, IEEE Computer Society Press, 1994.

[Fli96] Andrew Flint, "Three Different MCMs, Three Different Test Strategies," em *International Test Conference*, pp. 828-833, IEEE Computer Society Press, 1996.

[Fra96] Piero Franco et al., "Analysis and detection of timing failures in an experimental Test Chip," em *International Test Conference*, pp. 691-700, IEEE Computer Society Press, 1996.

[Fuc91] K. Fuchs, F. Fink e M. H. Schulz, "DYNAMITE: An Efficient Automatic Test Pattern Generation System for Path Delay Faults," em *IEEE Transactions on Computer Aided Design*, vol. 10, pp. 1323-1335, Oct. 1991.

[Goe81] P. Goel e B. C. Rosales, "PODEM-X: An Automatic Test Generation System for VLSI Logic Structures," em *18th IEEE Design Automation Conference*, pp. 260-268, IEEE Computer Society Press, 1981.

[Goo91] A. J. van de Goor, *Testing Semiconductor Memories: Theory and Practice*, Wiley & Sons, England, 1991.

[Gou98] Liam Goudge, "Debugging Embedded Systems," White Paper, disponível em http://www.arm.com/Documentation/WhitePapers/DebugEmbSys/, 1998.

[Hab94] Oliver F. Haberl e Thomas Kropf, "Self Testable Boards With Standard IEEE 1149.5 Module Test and Maintenance (MTM) Bus Interface," em *European Design & Test Conference*, pp. 220-225, IEEE Computer Society Press, 1994.

[Hal89] Andy Halliday, Greg Young e Al Crouch, "Prototype Testing simplified by Scannable Buffers and Latches," em *International Test Conference*, pp. 174-181, IEEE Computer Society Press, 1989.

[Han89] Peter Hansen, "Testing Conventional Logic and Memory Clusters using Boundary Scan Devices as Virtual ATE Channels," em *International Test Conference*, pp. 166-173, IEEE Computer Society Press, 1989.

[Hao95] Hong Hao e Rick Avra, "Structered Design-for-Debug - the SuperSPARC™ II Methodology and Implementation," em *International Test Conference*, pp. 175-183, IEEE Computer Society Press, 1995.

[Has89] Abu Hassan, V. K. Agarwal, Janusz Rajski e Benoit N. Dostie, "Testing of Glue Logic Interconnecting using Boundary Scan Architecture," em *International Test Conference*, pp. 700-711, IEEE Computer Society Press, 1989.

[Hay85] J. P. Hayes, "Fault Modeling," em *IEEE Design & Test of Computers*, pp. 88-95, Apr. 1985.

[Hol94] K. Holdbrook, S. Joshid, S. Mitra, J. Petolino, R. Ramon e M. Wong, "microSPARC™: A Case-Study of Scan Based Debug," em *International Test Conference*, pp. 70-75, IEEE Computer Society Press, 1994.

[HP99] Hewlett-Packard Test & Measurement Inc., disponível em http://www.tmo.hp.com/tmo/ , 1999.

[Hsi77] E. P. Hsieh, R. A. Rasmussen, L. J. Vidunas e W. T. Davis, "Delay Test Generation," em *Proc. of the 14th Design Automation Conference*, pp. 486-491, 1977.

[Hug84] Joseph L. A. Hughes e Edward J. McCluskey, "An Analysis of the Multiple Fault Detection Capabilities of the Single Stuck-at Fault Test Sets," em *International Test Conference*, pp. 70-75, IEEE Computer Society Press, 1984.

[Hun94] C. Hunter, E. K. Vida-Torku e Johnny LeBlanc, "Balancing Structured and Ad-hoc Design for Test: Testing of the PowerPC™ 603 Microprocessor," em *International Test Conference*, pp. 76-83, IEEE Computer Society Press, 1994.

[IEEE87] IEEE Standard VHDL Language Reference Manual, IEEE, New York, IEEE Std. 1076, 1987.

[IEEE93] IEEE Standard Test Access Port and Boundary-Scan Architecture, Oct. 1993, IEEE Std. 1149.1 (Includes IEEE Std. 1149.1a), ISBN 1-55937-350-4.

[IEEE95] IEEE Standard for Module Test and Maintenance Bus (MTM-Bus), Aug. 1995, IEEE Std. 1149.5, ISBN 1-55937-558-2.

[IEEE98] IEEE P1149.4/D21 Draft Standard for a Mixed-Signal Test Bus, Oct. 1998.

[Int98] International Test Technologies Inc., disponível em http://ireland.iol.ie/ittdub/, 1998.

[Jar89] Najmi Jarwala e Chi. W. Yau, "A New Framework for Analyzing Test Generation and Diagnosis Algorithms for Wiring Interconnects," em *International Test Conference*, pp. 63-70, IEEE Computer Society Press, 1989.

[Jar91] Najmi Jarwala e Chi W. Yau, "Achieving Board-Level BIST Using the Boundary Scan Master," em *International Test Conference*, pp. 649-658, IEEE Computer Society Press, 1991.

[Jar94] Najmi Jarwala, "Designing 'Dual Personality' IEEE 1149.1 Compliant Multi-Chip Modules," em *International Test Conference*, pp. 446-455, IEEE Computer Society Press, 1994.

[Jon92] F. de Jong, e A. J. de Lind van Wijngaarden, "Memory Interconnection Test at Board-Level," em *International Test Conference*, pp. 328-337, IEEE Computer Society Press, 1992.

[Kat94] Jerry Katz, "A Case-Study in the use of Scan in microSPARC™ Testing and Debug," em *International Test Conference*, pp. 456-460, IEEE Computer Society Press, 1994.

[Kon96] B. Konemann, Ben Bennetts, N. Jarwala e Benoit Nadeau-Dostie, "Built-in Self-Test: Assuring System Integrity," em *IEEE Computer*, pp. 39-45, IEEE Computer Society Press, Nov. 1996.

[Lam96] Gerd Lammers, "Bond-out Technology Helps Debug Real-Time Problems," em *Engineering Software Special Editorial Feature*, Electronic Design magazine, pp. 100-106, Oct. 24, 1996.

[Lau96] Gil Lauer, "Bridge the Emulator Gap for High-Performance Debugging," em *Engineering Software Special Editorial Feature*, Electronic Design magazine, pp. 130-140, Nov. 18, 1996.

[Lea95] Rick Leatherman, "Get the Most from Improved Emulators," em *Test & Measurement Special Editorial Feature,* Electronic Design magazine, pp. 157-164, Nov. 20, 1995.

[Lee96] David Lee e Mihalis Yannakakis, "Principles and Methods of Testing Finite State Machines – A Survey," em *Proceedings of the IEEE*, Vol. 8, n. 8, pp. 1089-1123, Aug. 1996.

[Lef90] M. F. Lefébvre, "Functional Test and Diagnosis: A Proposed JTAG Sample Mode Scan Tester," em *International Test Conference*, pp. 294-303, IEEE Computer Society Press, 1990.

[Leo99] Leonardo INSIGHT II, Projecto número N/96/2/1085/PI/II.1.1.c/FPC, disponível em http://www.hibu.no/avding/data/insight/, 1999.

[Leu91] B. Leung, "Design Methodology of Decimation Filters for Oversampled ADC Based on Quadratic Programming," em *IEEE Transactions on Circuits and Systems*, pp. 1121-1132, 1991.

[Lev95] M. Levitt, S. Nari, S. Naraynan, G. Grewal, L. Youngs, A. Jones, G. Billus e S. Paramanandam, "Testability, Debuggability and Manufacturability Features of the UltraSPARC™-I Microprocessor," em *International Test Conference*, pp. 157-166, IEEE Computer Society Press, 1995.

[Lew96] Jeff Lewis, "Intellectual Property (IP) Components," disponível em http://www.eedesign.com/ic-logic/IPdata/IParticle.html, 1996.

[Lie91] J.-C. Lien e Melvin A. Breuer, "Maximal Diagnosis for Wiring Networks," em *International Test Conference*, pp. 96-105, IEEE Computer Society Press, 1991.

[Lof96] Keith Lofstrom, "Early Capture for Boundary Scan Timing Measurements," em *International Test Conference*, pp. 417-422, IEEE Computer Society Press, 1996.

[Lun92] Dave Lundis, Chuck Hudson e Pat McHugh, "Applications of the IEEE P1149.5 Module Test and Maintenance Bus," em *International Test Conference*, pp. 984-992, IEEE Computer Society Press, 1992.

[Mad95] Vijav Madisetti e Jack Corley, "Rapid Prototyping of Application Specific Signal Processors," em *The RASSP Digest*, Vol. 2, 2nd Quarter 1995.

[Mag97] Peter Magnusson, "Efficient Techniques for Instrumented Instruction Set Simulation," disponível em http://www.sics.se/~psm/sim9705/, 1997.

[Man96] R. T. Maniwa, "Focus Report: Design for Test Tools," disponível em http://www.eedesign.com/EEdesign/FocusReport9609.html, Sep. 1996.

[Mar96] D. Marr, S. Natarajan, S. Thakkar e R. Zucker, "Multiprocessor Validation of the Pentium Pro," em *IEEE Computer*, pp. 47-53, Nov. 1996.

[Mat92] J. Matos, F. Pinto e J. M. Ferreira, "A Boundary Scan Test Controller for Hierarchical BIST," em *International Test Conference*, pp. 217-223, IEEE Computer Society Press, 1992.

[Mat93] J. Matos, A. Leão e J. C. Ferreira, "Control and Observation of Analog Nodes in Mixed-Signal Boards," em *International Test Conference*, pp. 323-331, IEEE Computer Society Press, 1993.

[Mau90] C. M. Maunder e R. E. Tulloss, *The Test Access Port and Boundary Scan Architecture*, IEEE Computer Science Press Tutorial, 1990, ISBN 0-8186-9070-4.

[Mau92a] C. M. Maunder, *The Board's Designer Guide to Testable Logic Circuits*, Addison-Wesley Publishers Ltd, 1992, ISBN 0-201-56513-7.

[Mau92b] C. M. Maunder, " Boundary-Scan: An End-of-Term Report," em *IEEE Design & Test of Computers*, pp. 82-85, June 1992.

[Max93] Peter C. Maxwell, "Let´s Grade ALL Faults," em *International Test Conference*, p. 595, IEEE Computer Society Press, 1993.

[McH94] P. McHugh, "The IEEE P1149.5 MTM-Bus, A Backplane Test & Initialization Interface," em *International Test Conference*, p. 1020, IEEE Computer Society Press, 1994.

[Mei95] Russel D. Meier, "Rapid Prototyping of a RISC Architecture for Implementation in FPGAs," em *IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 190-196, IEEE Computer Society Press, 1995.

[Mel92] Matthew Melton e Franc Brglez, "Automatic Pattern Generation for Diagnosis of Wiring Interconnect Faults," em *International Test Conference*, pp. 389-398, IEEE Computer Society Press, 1992.

[Mit97] Larry Mittag, "Software Debug Options on ASIC Cores," Embedded Systems Programming, disponível em http://www.espmag.com/97/feat9701.htm, 1997.

[Mon95] C. Montemayor et al., "Multiprocessor Design Verification for the PowerPC 620 Microprocessor," em *International Conference on Computer Design*, pp. 188-195, IEEE Computer Society Press, 1995.

[Muri93] M. Muris e A. Biewenga, "Using Boundary Scan Test to Test Random Access Memory Clusters," em *International Test Conference*, pp. 174-179, IEEE Computer Society Press, 1993.

[Murr96] Brian T. Murray e John P. Hayes, "Testing ICs: Getting to the Core of the Problem," em *IEEE Computer*, pp. 32-38, Nov. 1996.

[Nee96] W. Needham e N. Gallakota, "DFT Strategy for INTEL Microprocessors," em *International Test Conference*, pp. 396-399, IEEE Computer Society Press, 1996.

[Nov95] John Novellino, "Emulators Race to Keep Up with New Devices," em *Test & Measurement Special Editorial Feature*, Electronic Design magazine, pp. 151-154, Nov. 20, 1995.

[Nys95] Kjell Nystrom, Anders Astrom e Bo Thunnel, "MCM Test Strategy (a user's experience)," em *Proceedings of the European Design & Test Conference User Forum*, pp. 209-214, IEEE Computer Society Press, 1995.

[Olc95] Serafin Olcoz, Luis Entrena e Luis Berrojo, "VHDL Virtual Prototyping," em *Proceedings of the 6th International Conference on Rapid System Prototyping*, IEEE Computer Society Press, 1995.

[Oli96] J. Oliver e H. Kerkhoff, "Test Structures on MCM Active Substrate: Is it Worthwhile?," em *Proceedings of the European Design & Test Conference*, pp. 126-135, IEEE Computer Society Press, 1996.

[OrC99] OrCAD Corporation, "OrCAD's release 9", disponível em http://www.orcad.com/, 1999.

[Par92] Kenneth Parker, *The Boundary Scan Handbook*, Kluwer Academic Publishers, 1992, ISBN 0-7923-9270-1.

[Patel93] Rajiv Patel e K. Yarlagadda, "Testability Features of the SuperSPARC™ Microprocessor," em *International Test Conference*, pp. 773-781, IEEE Computer Society Press, 1993.

[Pater98] Michael Paterson e John Friedman, "Don't be Afraid of Debugging Symmetric Multiprocessing Systems," em Electronic Design magazine, pp. 42-48, Feb. 23, 1998.

[Pau96] Pierre G. Paulin et al., "Trends in Embedded Systems Technology: An Industrial Perspective," a part of *Hardware / Software Co-design*, disponível em http://tima-cmp.imag.fr/homepages/liem/paper_archive, 1996.

[Pet95] Leo Petropoulos, "Prototyping Beats Simulation for Complex, Real-time Designs," em *EDN*, pp. 136-140, 8 June 1995.

[Pic95] F. Pichon, S. Blanc e B. Candaele, "Mixed-Signal Modeling for System-on-chip Applications," em *Proceedings of the European Design & Test Conference*, pp. 218-222, IEEE Computer Society Press, 1995.

[Pyr95] Carol Pyron e W. C. Bruce, "Implementing 1149.1 in the PowerPC™ RISC Microprocessor Family," em *International Test Conference*, pp. 844-850, IEEE Computer Society Press, 1995.

[Rag91] P. Raghavachari, "Circuit Pack BIST from System to Factory - The MCERT Chip," em *International Test Conference*, pp. 641-648, IEEE Computer Society Press, 1991.

[Raj96] Rochit Rajusman, "Challenge Of The 90's: Testing CoreWare™ Based ASICs," em *International Test Conference*, p. 940, IEEE Computer Society Press, 1996.

[Rob90] Gordon D. Robinson e John G. Deshayes, "Interconnect Testing of Boards With Partial Boundary Scan," em *International Test Conference*, pp. 572-581, IEEE Computer Society Press, 1990.

[Rot66] J. P. Roth, "Diagnosis of Automata Failures: A calculus and a method," em *IBM Journal on Research and Development,* vol. 10, pp. 278-291, July 1966.

[RSI99] Router Solutions Inc., "Translator & Interface Tools for CAE-CAD-CAM", disponível em http://rsi-inc.com/, 1999.

[RTW98] RASSP Taxonomy Working Group, "VHDL Modeling Terminology and Taxonomy," Revision 2.4, em http://www.atl.external.lmco.com/rassp/taxon/rassp_taxon.html, ou http://rassp.scra.org, Feb. 23, 1998.

[Sav91] Jacob Savir e Robert Berry, "At-Speed Test is not Necessarily an AC Test," em *International Test Conference*, pp. 722-727, IEEE Computer Society Press, 1991.

[SCA94] National Semiconductor, *SCAN™ Data Book*, 1994.

[SCO94] Texas Instruments, *SCOPE™ Logic Products: Application and Data Manual*, 1994, ISBN 3-88078-098-6.

[Sed94] Richard M. Sedmak, "Boundary-Scan: Beyond Production Test," em *International Test Conference*, pp. 415-420, IEEE Computer Society Press, 1994.

[Sid95] J. Sidoran, C. Burns, S. Maethner, D. Spenser e H. Bond, "A Case Study on Rapid Systems Prototyping and its Impact on Systems Evolution," em *Proceedings of the 6th International Conference on Rapid System Prototyping*, IEEE Computer Society Press, 1995.

[Sie95] K. Sievert, Y. Manoli, A. Both e R. Lerch, "On-chip Emulation and Debugging for Embedded Microcontrollers using the IMS ScanDebugger," em *European Design & Test Conference User Forum*, pp. 229-233, IEEE Computer Society Press, 1995.

[Sta96] James Stanbridge e Iain Small, "Boundary Scan Solves Diagnostic Problems," em *Test – The European Test Industry Journal*, pp. 16-18, Sep. 1996.

[Sto77] T. Storey e J. Barry, "Delay Fault Simulation," em *Proc. of the 14th Design Automation Conference,* pp. 492-494, 1977.

[Tem98] Temento Systems S. A., "TemTag™: High Speed Boundary Scan Controller," disponível em http://www.temento.com, 1998.

[Tho98] Thomas N. Anderson, "The Telemark Assembler (TASM) User's Manual," Version 3.1, disponível em www.halcyon.com/squakvly/, Squak Valley Software, Feb. 1998.

[Tuc97] Barbara Tuck, "Complex ASIC straining verification resources," disponível em http://www.computer-design.com/Editorial/1997/01/ASIC/197asdrpt.html , Computer Design, 1997.

[Var94] Prab Varma, "On Path Delay Testing in a Standard Scan Environment," em *International Test Conference*, pp. 164-173, IEEE Computer Society Press, 1994.

[Vid95] E. K. Vida-Torku, Charles H. Malley, Sung Park e Rowland Reed, "Design and Test of the PowerPC™ 603 Microprocessor," em *European Design & Test Conference,* pp. 378-384, IEEE Computer Society Press, 1995.

[Vir98] VirtuaLogic Incorporated, "VirtuaLogic-8 Emulation System," disponível em http://www.ikos.com/products/vsli/index.html, 1998.

[Wag91] Kenneth D. Wagner e T. W. Williams, "Enhancing Board Functional Self-Test by Concurrent Sampling," em *International Test Conference*, pp. 633-640, IEEE Computer Society Press, 1991.

[Was84] R. J. K. Wassinik, *Soldering in Electronics*, Electromechanical Publications, 1984, ISBN 0-901150-14-2.

[Wen96] Thomas Wenzel e I. Krellner, "The ARTEMIS Chip: An On-board Test Solution," em *ARTEMIS Symposium*, ESPRIT III Project 6138, Jan. 1996.

[Whe91] Lee Whetsel, "An IEEE 1149.1 Based Logic / Signature Analyzer in a Chip," em *International Test Conference*, pp. 869-878, IEEE Computer Society Press, 1991.

[Whe92] Lee Whetsel, "A Proposed Method of Accessing 1149.1 Applications in a Backplane Environment," em *International Test Conference*, pp. 206-216, IEEE Computer Society Press, 1992.

[Whe93] Lee Whetsel, "Hierarchically Accessing 1149.1 Applications in a System Environment," em *International Test Conference*, pp. 517-526, IEEE Computer Society Press, 1993.

[Whi96] Bruce E. Whittaker e Nicole A. Doherty, "Boundary Scan Helps Solve Field Failures," em *Evaluation Engineering*, Vol. 35, nº 10, pp. 26-30, Oct. 1996.

[Woo95] Joel Woodward, "Resurgence in Prototyping," Integrated System Design Editorial, disponível em http://www.eedesign.com/Editorial/1995/ASICFeature9505.html, 1995.

[Yau89] C.W. Yau e Najmi Jarwala, "A unified theory for designing optimal test generation and diagnosis algorithms for board interconnects," em *International Test Conference*, pp. 71-77, IEEE Computer Society Press, 1989.

[Zor89] Yervant Zorian e Najmi Jarwala, "Designing Fault-Tolerant, Testable VLSI Processors Using the IEEE P1149.1 Boundary-Scan Architecture," em *Proceedings of the International Conference on Computer Design*, pp. 580-581, IEEE Computer Society Press, 1989.

Anexo

O anexo desta dissertação é constituído por duas partes:

- Uma em formato impresso, que ocupa as páginas seguintes.
- Outra em formato electrónico, armazenada no CR-ROM, que se encontra na bolsa fixada no verso da contracapa.

A seguinte lista descreve os elementos que fazem parte do anexo em formato impresso:

- Arquitectura do componente que inclui a infraestrutura para o teste e depuração (ilustrada em esquemático).
- Descrição da infraestrutura de teste (BSDL).
- Arquitectura de topo do controlador residente (ilustrada em esquemático).
- Arquitectura de topo dos processadores do controlador (ilustrada em e squemático).
- Modelo do sistema utilizado para a experimentação prática do trabalho realizado.

O anexo em formato electrónico inclui:

- Todos os ficheiros que contêm os elementos apresentados em formato impresso.
- Ficheiros com a descrição dos blocos internos dos processadores do controlador (AHDL).
- Ficheiros com a descrição dos blocos internos do componente que inclui a infraestrutura para o teste e depuração (AHDL).
- Ficheiro com a descrição da cadeia de varrimento interna do componente anterior.
- Ficheiro com o resultado da simulação do funcionamento do componente anterior, antes de ser adicionada a infraestrutura para o teste e depuração.
- Ficheiros com os programas produzidos pelo módulo de geração automática (*assembly*).
- Ficheiros de inicialização dos modelos das memórias, utilizados no ambiente de simulação.
- Ficheiro com o resultado da simulação da execução do programa de teste e depuração.
- Versão não funcional do programa de controlo e interface do sistema de demonstração baseado no 80C51 (desenvolvido no âmbito do projecto JNICT PBIC/C/TIT/2474/95).
- Os cursos desenvolvidos para o projecto Leonardo INSIGHT II.
- O questionário utilizado nas entrevistas, a que se fez referência no capítulo introdutório.
- O relatório sobre o estado da arte, a que se fez referência no capítulo introdutório
- Esta dissertação em formato Adobe® Acrobat.

O relatório sobre o estado da arte fornece uma perspectiva geral dos aspectos principais, ou considerados mais importantes, na área do projecto para o teste e depuração. Os materiais contidos no relatório foram recolhidos entre finais de 1997 e inícios de 1998, recorrendo às fontes de informação descritas no seu primeiro capítulo. Estas incluem: livros, revistas e jornais, locais da Internet, listas de distribuição de correio electrónico, panfletos e brochuras de cursos, folhetos e materiais de promoção de companhias comerciais e associações / organizações desta área, e uma série de artigos de conferências, revistas e jornais científicos. Os nove capítulos que se seguem abordam os seguintes assuntos:

- Normas relevantes para a área do projecto para o teste e depuração. Cada norma é apresentada através de um formulário normalizado, que ocupa uma página[87] e que contém os seguintes campos: identificação da norma, objectivo, organização que promove ou mantém a norma e o seu contacto, situação actual e comentários adicionais.
- Produtos comerciais. Campos: nome do produto e informação acerca da companhia que o produz, descrição do produto, preço, avaliação qualitativa, comparação com outros produtos concorrentes e comentários adicionais.
- Produtos de distribuição gratuita ou generalizada (*shareware* / *freeware*). Campos: nome do produto e informação acerca da companhia ou instituição que o produz, descrição do produto, restrições ou limitações conhecidas ou identificadas, avaliação qualitativa, comparação com outros produtos similares e comentários adicionais.
- Consórcios, associações ou organizações. Campos: nome e contacto, objectivos, data de fundação ou constituição, documentos ou produtos suportados e comentários adicionais.
- Companhias comerciais. Campos: nome e contacto, objectivos e nichos de mercado de maior importância, informações acerca da dimensão / orçamento e comentários adicionais.
- Grupos de investigação e centros de excelência. Campos: nome e contacto do centro ou da pessoa responsável, áreas de investigação, informações acerca da dimensão / orçamento e comentários adicionais.
- Projectos relevantes. Campos: título, objectivos, pessoa / instituição responsável, data de início / fim, descrição sumária do projecto e comentários adicionais.
- Conferências, simpósios e reuniões científicas. Campos: título e data de realização, promotores, objectivos, informação acerca da dimensão / conteúdo e comentários adicionais.
- Quem é quem. Cada indivíduo, cujo trabalho seja considerado relevante para a área do teste e depuração, é apresentado através de um formulário com os seguintes campos: nome e contacto, posição actual e qualificações, temas nos quais desenvolve a sua actividade e comentários adicionais.

[87] Todos os elementos dos capítulos seguintes são igualmente apresentados através de um formulário normalizado de uma página, em que apenas mudam os campos em função do objectivo do capítulo.

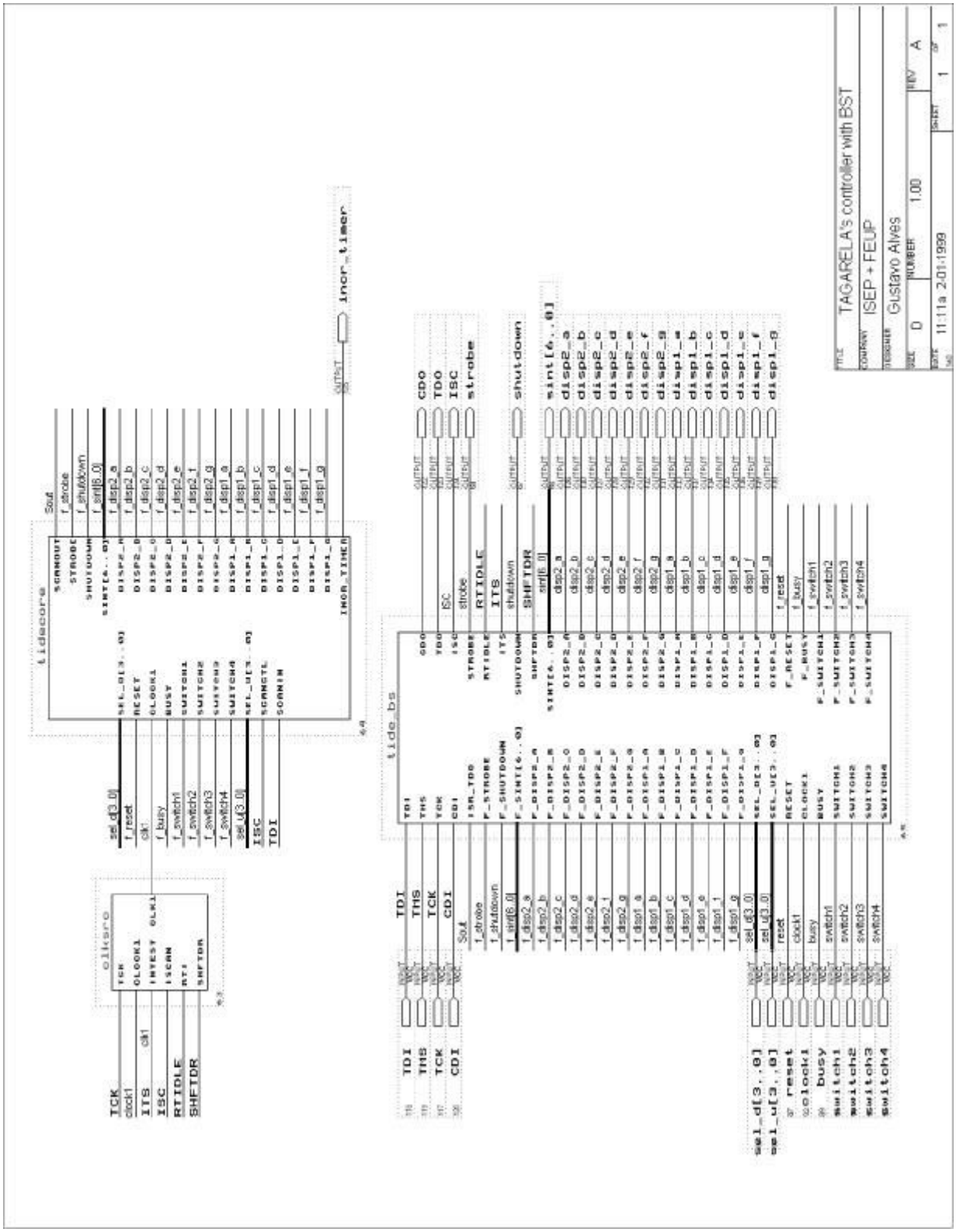
O último capítulo apresenta um conjunto de perspectivas de evolução na área do projecto para o teste e depuração. Este conjunto é divido em quatro categorias de acordo com o nível hierárquico a que se referem: macroblocos ou componentes de propriedade intelectual, CI, CCI e sistemas.

O relatório possui ainda o seguinte conjunto de ligações, que permitem uma navegação entre elementos de informação associados ou interligados.

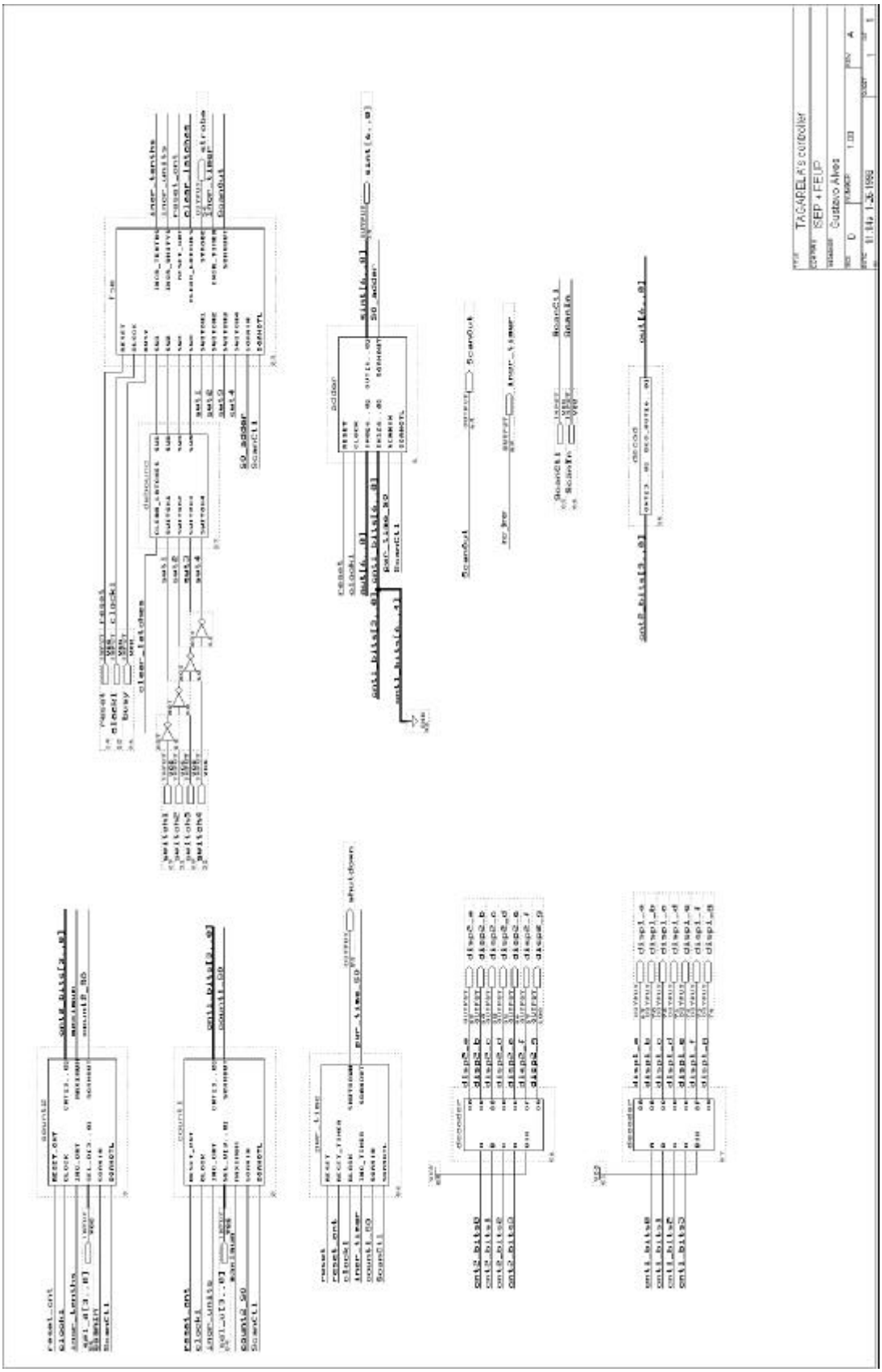
- Ligações entre páginas do relatório.
- Ligações entre páginas do relatório e páginas da Internet.
- Ligações entre páginas do relatório e páginas armazenadas no CD-ROM.

Uma ligação entre páginas do relatório é identificada pelo aparecimento de uma palavra, ou expressão, em cor azul e sublinhada, por exemplo assim. Uma ligação a uma página da Internet é identificada pelo prefixo http://, seguido do respectivo endereço, aparecendo toda a expressão em azul e sublinhada (e.g. http://www.fe.up.pt). Uma ligação a uma página armazenada no CD-ROM é identificada pelo prefixo \TEMP\, seguido do nome do capítulo, ou da secção, e do elemento a que se refere, aparecendo toda a expressão em azul e sublinhada (e.g., \TEMP\Books\digital_system_testing.htm). Para seguir qualquer uma destas ligações coloca-se o apontador do rato em cima da expressão em cor azul e sublinhada (devendo então o apontador mudar para a forma de uma mão com o dedo indicador levantado) e carrega-se no botão esquerdo. Após breves momentos, a nova página é carregada.

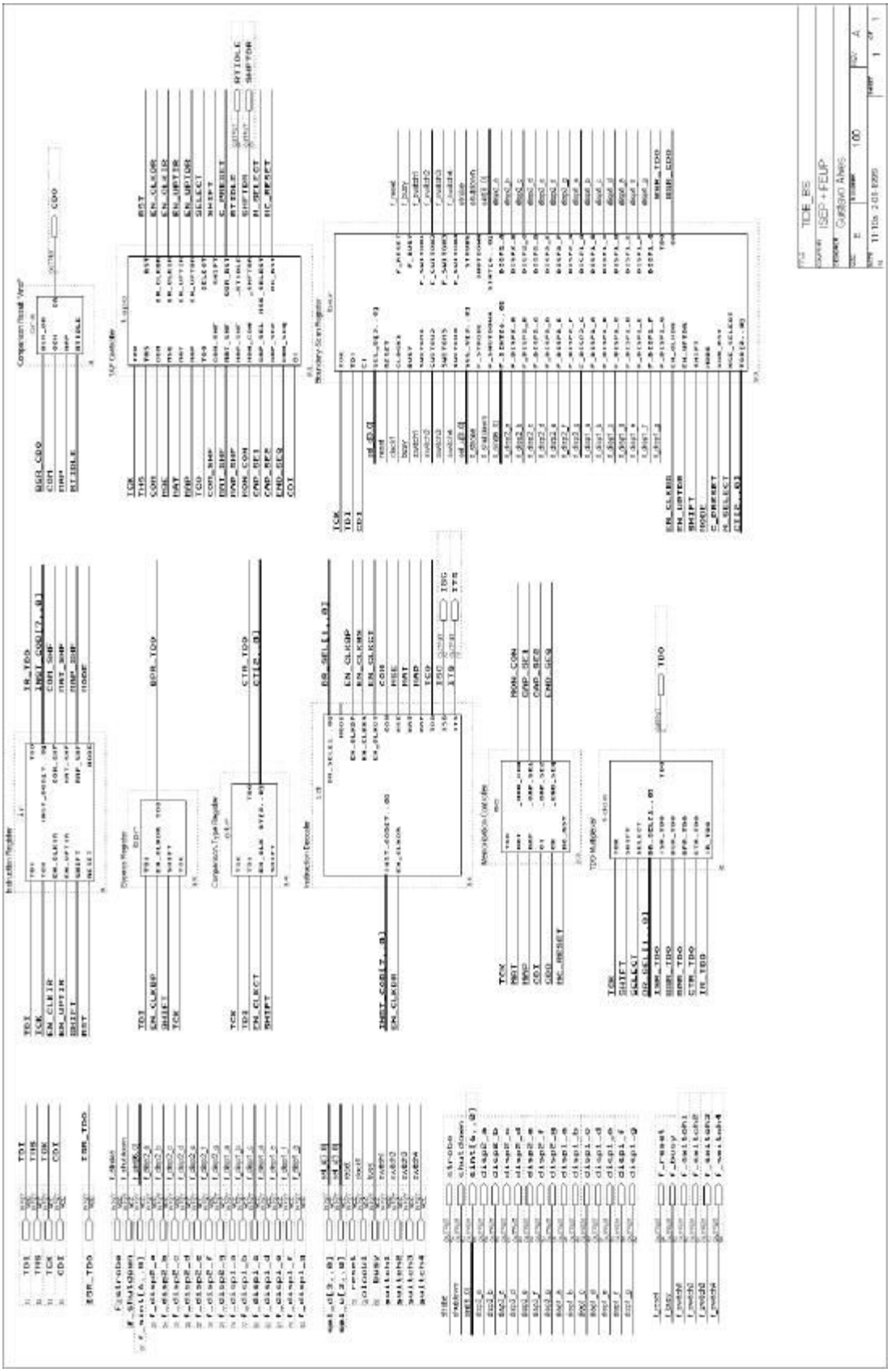
Arquitectura de topo do componente que inclui a infraestrutura para o teste e depuração.



Arquitectura da lógica funcional do componente anterior.



Arquitectura da lógica de teste do componente anterior.

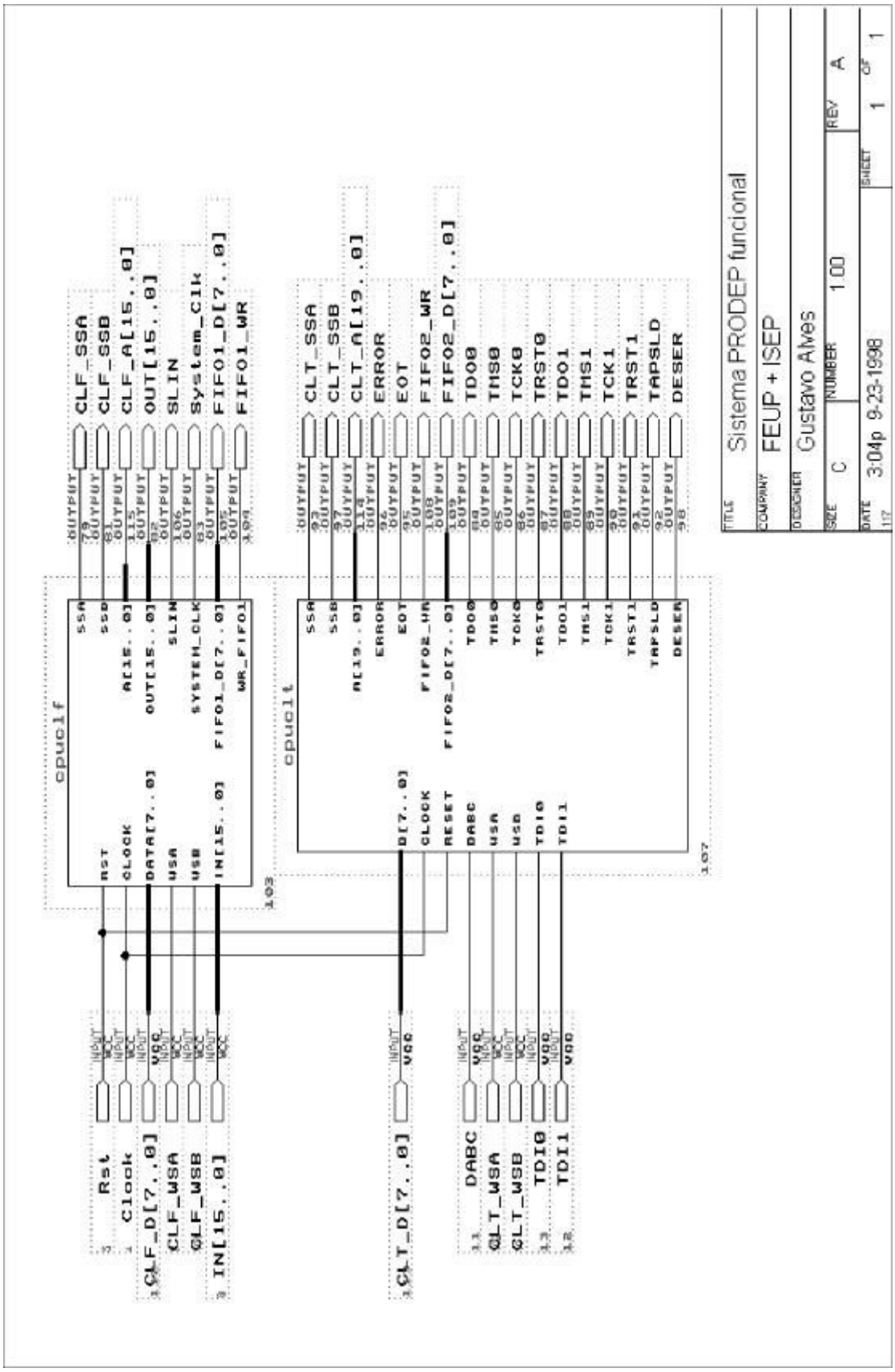


Ficheiro com a descrição da infraestrutura de teste (BSDL) do componente anterior.

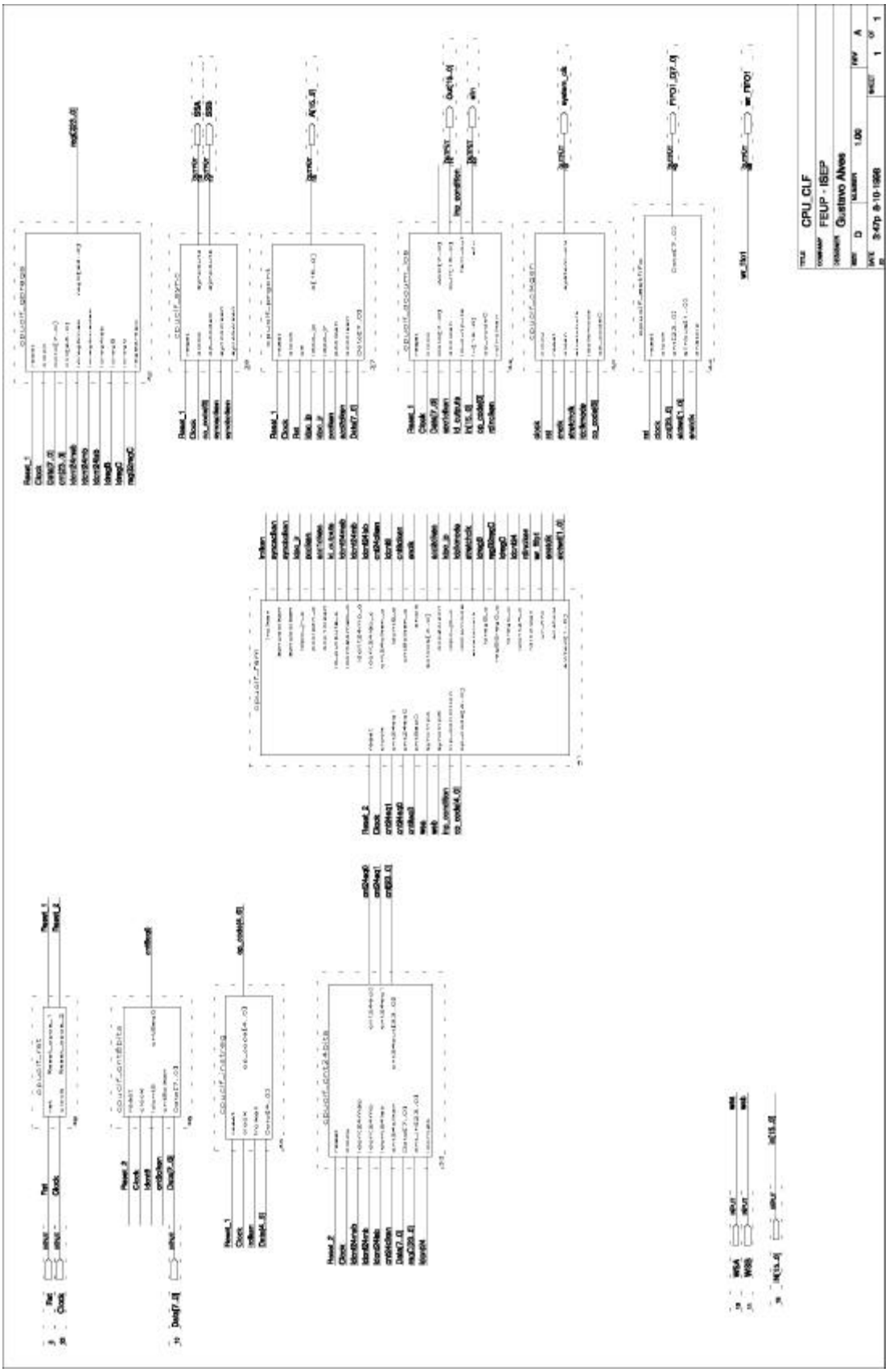
```
-- ---------------------------------------------------------------------------------------------
-- descricao BSDL gerada manualmente, seguindo a descricao fe ita no capitulo 4 de "Boundary-Scan
-- Test: A Pratical Aproach" [Ble93]
-- TIDE, compilado para o componente EPF10K10LC84-3 e com a atribuicao de pinos descrita em
-- tide.acf / tide.rpt
-- ---------------------------------------------------------------------------------------------

entity tide is
   -- generic parameter

   generic (PHYSICAL_PIN_MAP : string := "undefined" );
   -- logical port description

   port (
      CLOCK1   :in      bit;
      SEL_D0   :in      bit;
      SEL_D1   :in      bit;
      SEL_D2   :in      bit;
      SEL_D3   :in      bit;
      SEL_U0   :in      bit;
      SEL_U1   :in      bit;
      SEL_U2   :in      bit;
      SEL_U3   :in      bit;
      RESET    :in      bit;
      BUSY     :in      bit;
      SWITCH1  :in      bit;
      SWITCH2  :in      bit;
      SWITCH3  :in      bit;
      SWITCH4  :in      bit;
      STROBE   :out     bit;
      SHUTDOWN :out     bit;
      SINT0    :out     bit;
      SINT1    :out     bit;
      SINT2    :out     bit;
      SINT3    :out     bit;
      SINT4    :out     bit;
      SINT5    :out     bit;
      SINT6    :out     bit;
      DISP2_G  :out     bit;
      DISP2_F  :out     bit;
      DISP2_E  :out     bit;
      DISP2_D  :out     bit;
      DISP2_C  :out     bit;
      DISP2_B  :out     bit;
      DISP2_A  :out     bit;
      DISP1_G  :out     bit;
      DISP1_F  :out     bit;
      DISP1_E  :out     bit;
      DISP1_D  :out     bit;
      DISP1_C  :out     bit;
      DISP1_B  :out     bit;
      DISP1_A  :out     bit;
      TCK      :in      bit;
      TMS      :in      bit;
      TDI      :in      bit;
      CDI      :in      bit;
      TDO      :out     bit;
      CDO      :out     bit;
      VCC      :linkage bit_vector(1 to 5); -- tal como em 10K10L84.BSD
      GND      :linkage bit_vector(1 to 5)
   );

   -- use statement(s)
   use STD_1149_1_1990.all; -- Get Standard attributes and d efinitions

   -- package pin mapping
   attribute PIN_MAP of tide : entity is PHYSICAL_PIN_MAP;

      constant PLCC44: PIN_MAP_STRING :=
         "disp_g:54,"&
         "CDO:58,"&
         "TDO:60,"&
         "en:62,"&
         "disp_a:64,"&
         "disp_b:66,"&
         "disp_c:70,"&
         "disp_d:72,"&
         "disp_e:78,"&
         "disp_f:80,"&
         "sint0:61,"&
         "sint1:65,"&
         "sint2:67,"&
         "sint3:69,"&
         "sint4:71,"&
         "sint5:73,"&
         "sint6:79,"&
         "shutdown:81,"&
         "strobe:83,"&
         "TMS:10,"&
         "CDI:8,"&
         "TDI:6,"&
         "TCK:2,"&
```

```
         "switch1:39,"&
         "switch2:37,"&
         "switch3:35,"&
         "switch4:29,"&
         "busy:27,"&
         "reset:25,"&
         "sel_d0:23,"&
         "sel_d1:21,"&
         "sel_d2:19,"&
         "sel_d3:17,"&
         "sel_u0:11,"&
         "sel_u2:7,"&
         "sel_u1:9,"&
         "sel_u3:5,"&
         "clock2:3,"&
         "clock1:1,"&
         "VCC:(20, 40, 45, 63, 4),"& -- tal como em 10K10L84.BSD
         "GND:(26, 41, 46, 68, 82)";

   -- scan port identification
   attribute TAP_SCAN_CLOCK of TCK : signal is (20.0e6, BOTH);
      -- o valor real indica a maxima frequencia de funcion amento de TCK; LOW ou BOTH indicam
      -- o(s) estado(s) em que TCK pode ser interrompido sem perda de dados
   attribute TAP_SCAN_MODE  of TMS : signal is true;
   attribute TAP_SCAN_IN    of TDI : signal is true;
   attribute TAP_SCAN_OUT   of TDO : signal is true;

   -- TAP description
   attribute INSTRUCTION_LENGTH of tide : entity is 8;
   attribute INSTRUCTION_OPCODE of tide : entity is
      -- esta descricao nao tem que ser exaustiva; aqui inclui-se apenas 1 codigo por cada instrucao
      "EXTEST (00000000),"&
      "BYPASS (11111111),"&
      "SAMPLE (00000010),"&
      "INTEST (00011101),"&
      "ISCAN  (00011000) "; -- instrucao de acesso 'a cadeia de varrimento interna

   attribute INSTRUCTION_CAPTURE of tide : entity is "00000001";

   -- boundary register description
   attribute BOUNDARY_CELLS    of tide : entity is "BC_1, BC_4";
   attribute BOUNDARY_LENGTH   of tide : entity is 38;
   attribute BOUNDARY_REGISTER of tide : entity is

      " 37 (BC_4, CLOCK1   ,clock, X), " &
      " 36 (BC_4, SEL_U3   ,clock, X), " &
      " 35 (BC_4, SEL_U2   ,clock, X), " &
      " 34 (BC_4, SEL_U1   ,clock, X), " &
      " 33 (BC_4, SEL_U0   ,clock, X), " &
      " 32 (BC_4, SEL_D3   ,clock, X), " &
      " 31 (BC_4, SEL_D2   ,clock, X), " &
      " 30 (BC_4, SEL_D1   ,clock, X), " &
      " 29 (BC_4, SEL_D0   ,clock, X), " &
      " 28 (BC_4, RESET    ,clock, X), " &
       -- estas celulas correspondem 'as figuras f10-11 da no rma; em BSDL sao celulas BC_4 com
       -- funcao clock; o X nao faz aqui sentido, mas tem que haver ou 4 ou 7 argumentos
      " 27 (BC_1, BUSY     ,input, X), " &
      " 26 (BC_1, SWITCH4  ,input, X), " &
      " 25 (BC_1, SWITCH3  ,input, X), " &
      " 24 (BC_1, SWITCH2  ,input, X), " &
      " 23 (BC_1, SWITCH1  ,input, X), " &
      " 22 (BC_1, STROBE   ,output2, X), " &
      " 21 (BC_1, SHUTDOWN ,output2, X), " &
      " 20 (BC_1, SINT6    ,output2, X), " &
      " 19 (BC_1, SINT5    ,output2, X), " &
      " 18 (BC_1, SINT4    ,output2, X), " &
      " 17 (BC_1, SINT3    ,output2, X), " &
      " 16 (BC_1, SINT2    ,output2, X), " &
      " 15 (BC_1, SINT1    ,output2, X), " &
      " 14 (BC_1, SINT0    ,output2, X), " &
      " 13 (BC_1, DISP2_G  ,output2, X), " &
      " 12 (BC_1, DISP2_F  ,output2, X), " &
      " 11 (BC_1, DISP2_E  ,output2, X), " &
      " 10 (BC_1, DISP2_D  ,output2, X), " &
      "  9 (BC_1, DISP2_C  ,output2, X), " &
      "  8 (BC_1, DISP2_B  ,output2, X), " &
      "  7 (BC_1, DISP2_A  ,output2, X), " &
      "  6 (BC_1, DISP1_G  ,output2, X), " &
      "  5 (BC_1, DISP1_F  ,output2, X), " &
      "  4 (BC_1, DISP1_E  ,output2, X), " &
      "  3 (BC_1, DISP1_D  ,output2, X), " &
      "  2 (BC_1, DISP1_C  ,output2, X), " &
      "  1 (BC_1, DISP1_B  ,output2, X), " &
      "  0 (BC_1, DISP1_A  ,output2, X), " ;

   -- register access description

    attribute REGISTER_ACCESS of tide : entity is
       "BOUNDARY       (EXTEST, SAMPLE, INTEST),"&
       "BYPASS         (BYPASS),"&
       "INTERNAL_1[26] (ISCAN)";
          -- declara-se aqui a existencia e o comprimento da cadeia de varrimento interna, bem
          -- como a instrucao que lhe esta' associada (com o pcode ja' definido)

end tide;

-- ---------------------------------------------------------------------------------------------
```

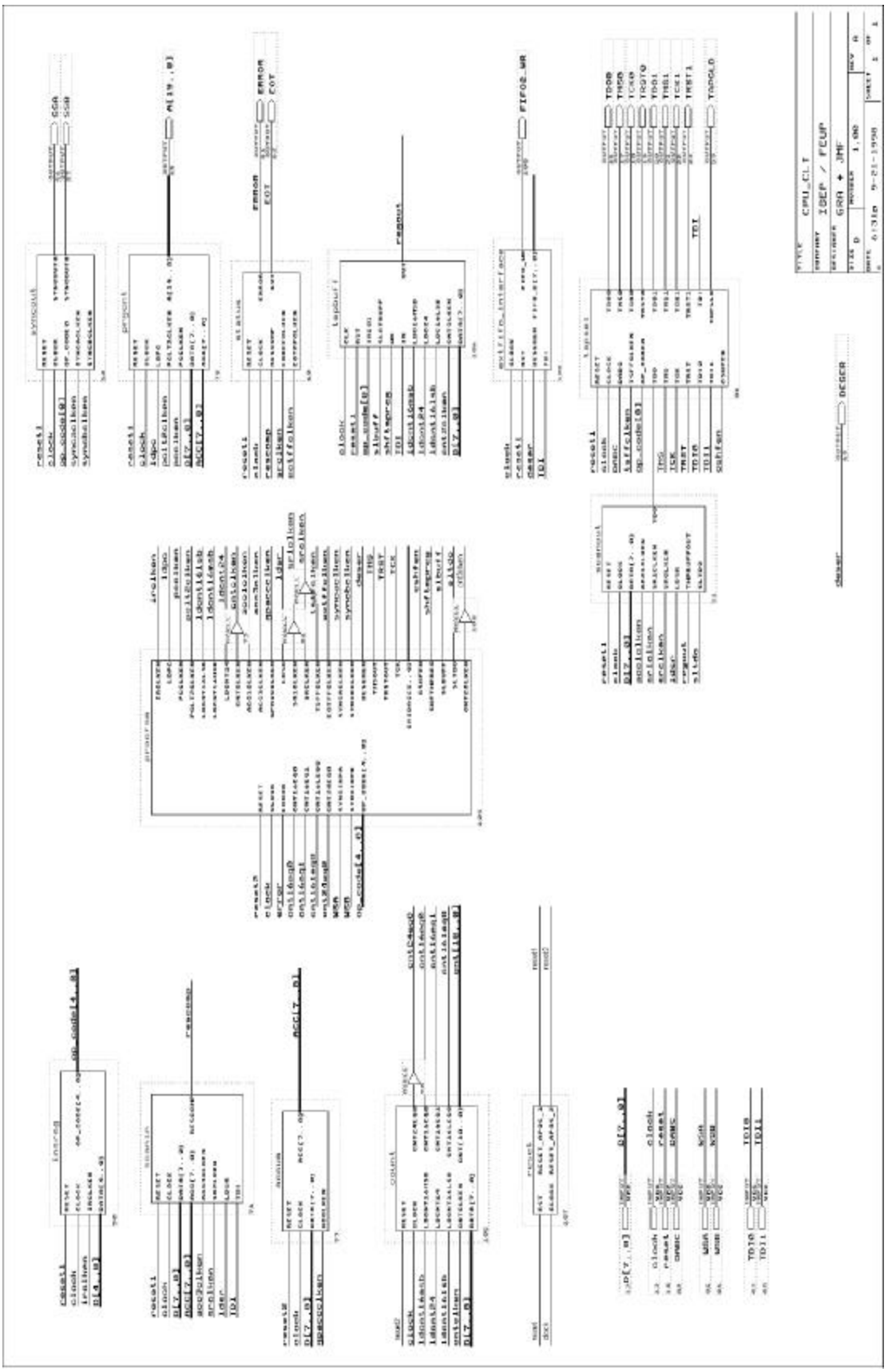
Arquitectura de topo do controlador residente (PRODEP).



Arquitectura do processador de controlo da lógica funcional (CLF).



Arquitectura do processador de controlo da lógica de teste (CLT).



Modelo do sistema usado na experimentação prática do trabalho realizado.