UNIVERSIDADE DO PORTO

FACULDADE DE ENGENHARIA

O TESTE DE CARTAS DE CIRCUITO IMPRESSO COM BST: ARQUITECTURA DE UM CONTROLADOR RESIDENTE, E GERAÇÃO AUTOMÁTICA DO PROGRAMA DE TESTE

José Manuel Martins Ferreira

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

DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

PORTO - ABRIL DE 1992

O Teste de Cartas de Circuito Impresso com BST: Arquitectura de um controlador residente, e Geração Automática do Programa de Teste

José Manuel Martins Ferreira

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

Departamento de Engenharia Electrotécnica e de Computadores
Faculdade de Engenharia
Universidade do Porto
Porto 1992

À Margarida e à Ritinha

Agradecimentos

A realização deste trabalho não teria sido possível sem a colaboração de um grupo de amigos que me acompanharam ao longo do tempo, e cujo apoio se concretizou das mais variadas formas.

O enquadramento proporcionado pelo consórcio do projecto ESPRIT 2478 foi igualmente valioso, com realce para a Philips (Eindhoven, Holanda), através do grupo *Electronic Design and Tools* (ED&T), e para o INESC, que me proporcionou os recursos indispensáveis para o trabalho a realizar. Manifesto o meu reconhecimento ao Ir. Frans de Jong (Philips ED&T, Eindhoven), pelas muitas conversas que mantivemos ao longo deste tempo, e por toda a colaboração prestada. O meu reconhecimento vai também para todos os elementos do grupo de CAD e Microelectrónica (INESC Porto), e nomeadamente para os Engºs Canas Ferreira e Filipe Pinto. Agradeço em particular ao Engº Filipe Pinto, tanto pela colaboração como pela amizade.

A conclusão deste trabalho dentro dos prazos estabelecidos só foi possível pelo contributo de várias pessoas, entre as quais quero salientar o Professor Borges Gouveia, a quem agradeço igualmente a minha introdução ao INESC. Agradeço também ao Professor Raul Vidal por toda a confiança demonstrada desde o início, e por uma relação profissional que contribuiu para consolidar uma amizade que vem de há já vários anos. A minha iniciação ao tema deste trabalho fica a dever-se ao Professor Silva Matos. O acompanhamento que me dispensou ao longo destes anos, aliado ao empenho e confiança demonstrados, constituíram para mim motivos de admiração que se estendem muito para lá do plano profissional.

Finalmente, agradeço à minha família todo o apoio e disponibilidade manifestados, e em particular à minha esposa e à minha filha, a quem foram subtraídas muitas das horas envolvidas na realização deste trabalho.

Nota ao leitor

A escrita de um documento técnico enfrenta frequentemente um problema para o qual é necessária uma solução de compromisso, e que consiste no conflito entre a pureza da linguagem e a clareza da exposição. Embora não se possa afirmar que é de todo impossível evitar a utilização de termos estrangeiros, a verdade é que a respectiva tradução é em muitos casos dificultada pela inexistência de termos portugueses directamente equivalentes, ou de expressões que reproduzam satisfatoriamente os conceitos envolvidos. A solução mais adequada para este problema consistiria no estabelecimento de uma terminologia comum, adoptada pela comunidade científica que desenvolve a sua actividade num mesmo domínio, o que proporcionaria um conjunto de termos portugueses com o mesmo significado para todos os seus elementos.

Atendendo a que o círculo de leitores a que este trabalho se destina é restrito, e possui de facto uma terminologia comum que inclui um grande número de termos estrangeiros, optou-se por favorecer a clareza da exposição. O conjunto de termos ingleses com utilização habitual na linguagem verbal é deste modo mantido, embora se priviligie a utilização de termos portugueses, sempre que o seu emprego não suscitar dúvidas.

Lista de abreviaturas

ASIC:	*Application Specific Integrated Circuit*
ATE:	*Automatic Test Equipment*
BIST:	*Built-In Self-Test*
BiCMOS:	*Bipolar / Complementary Metal Oxide Semiconductor*
BSDL:	*Boundary Scan Description Language*
BST:	*Boundary Scan Test*
C&O:	Controlabilidade e Observabilidade
CAD:	*Computer Aided Design*
CCI:	Carta de Circuito Impresso
CI:	Circuito Integrado
CL:	(Nós de) Controlabilidade das Ligações
CMOS:	*Complementary Metal Oxide Semiconductor*
DFT:	*Design for Testability*
E/S:	Entrada / Saída
EBST:	*Extended Boundary Scan Test*
ECL:	*Emitter Coupled Logic*
EDIF:	*Electronic Data Interchange Format*
ESPRIT:	*European Strategic Programme for Research and Development in Information Technology*
FPGA:	*Field-Programmable Gate Array*
GAPT:	Geração Automática do Programa de Teste
HCMOS:	*High-Speed Complementary Metal Oxide Semiconductor*
IEEE:	*Institute of Electrical and Electronics Engineers*
JEDEC:	*Joint Electron Device Engineering Council*
JETAG:	*Joint European Test Action Group*
JTAG:	*Joint Test Action Group*
LFSR:	*Linear Feedback Shift Register*
MCM:	*Multi-Chip Module*
MISR:	*Multiple Input Signature Register*

MSI: *Medium Scale Integration*

MTM: *Module Test and Maintenance (bus)*

OL: (Nós de) Observabilidade das Ligações

RAM: *Random Access Memory*

SSI: *Small Scale Integration*

Std: *Standard*

TAP: *Test Access Port*

TCK: *Test Clock*

TDI: *Test Data Input*

TDO: *Test Data Output:*

TMS: *Test Mode Select*

TRST: *Test Reset*

TTL: *Transistor Transistor Logic*

VHDL: *VHSIC Hardware Description Language*

VHSIC: *Very High Speed Integrated Circuits*

VLSI: *Very Large Scale Integration*

VRS: Vector de Resposta Sequencial

VTS: Vector de Teste Sequencial

Glossário

O conjunto de termos descritos a seguir não pretende representar a totalidade das expressões empregues nesta tese, mas apenas aquelas que são usadas com maior frequência, ou que estão directamente relacionadas com o trabalho aqui descrito.

Algoritmo adaptativo: Algoritmo que efectua a geração de vectores de teste em duas etapas. A geração do segundo conjunto de vectores pode ou não ser orientada pela análise das respostas obtidas para o primeiro conjunto.

Algoritmo de um passo só: Algoritmo em que a geração do conjunto de vectores de teste é efectuada de uma só vez.

Barramento: Grupo de ligações na CCI em que existem nós do tipo CL que partilham um sinal de controlo de estado comum.

Controlador de teste: Bloco responsável por cadenciar a sequência de operações que permite a realização do teste. À escala da CCI, representa um componente concebido especificamente para controlar a infraestrutura de teste disponível.

Controlador do TAP: Bloco interior aos componentes com BST, responsável por gerar o conjunto de sinais que permitem o controlo desta infraestrutura, a partir da sequência de valores lógicos aplicados aos pinos do TAP.

E/S primárias: Conjunto de nós de teste que pertencem aos conectores de uma CCI.

Instrução BST: Representa um comando para o registo de instrução da infraestrutura BST de um componente.

Instrução do controlador de teste: Representa um comando executável pelo controlador de teste, e que tem correspondência directa com uma operação elementar necessária para controlar a infraestrutura BST disponível na CCI.

Ligação BST: Ligação na CCI em que todos os seus pinos são pinos BST, ou pinos de E/S primária da CCI.

Nó de teste: Ponto do circuito a que é possível o acesso (série ou paralelo) para permitir a aplicação de valores (controlabilidade) ou a captura de respostas (observabilidade).

Nós (do tipo) CL: Nós de controlabilidade das ligações. Constituem o conjunto de nós de teste a partir dos quais é possível o controlo dos valores lógicos a aplicar nas ligações da CCI.

Nós (do tipo) OL: Nós de observabilidade das ligações. Constituem o conjunto de nós de teste a partir dos quais é possível a captura dos valores lógicos presentes nas ligações da CCI.

Pino BST: Pino de um componente BST, ao qual se encontram associadas uma ou mais células BST.

Vector de teste: Conjunto de valores aplicados em simultâneo a todos os nós de teste. É também usado para referir o conjunto de valores deslocados para o interior de uma cadeia BST.

VRS (vector de resposta sequencial): Sequência de valores lógicos capturados numa ligação, durante o teste à CCI.

VTS (vector de teste sequencial): Sequência de valores lógicos aplicados numa ligação, durante o teste à CCI.

Índice

Capítulo 1

Introdução

Panorâmica global do teste de CCIs

A importância do projecto testável

A importância do BST

Conteúdo da tese

Estrutura da tese

Enquadramento proporcionado pelo projecto ESPRIT 2478

1. INTRODUÇÃO

Este capítulo realiza o enquadramento deste trabalho, situando o aparecimento do *Boundary Scan Test* (BST) no contexto das dificuldades crescentes experimentadas pelas tecnologias de teste funcional, e *in-circuit*. Os progressos verificados nos domínios da miniaturização, e da complexidade, são responsabilizados por estas dificuldades, e a importância que neste contexto assumem as metodologias de projecto testável (DFT, *Design for Testability*) é discutida. A tecnologia BST é então introduzida, e a sua importância é apresentada por comparação das características principais, relativamente às tecnologias funcional, e *in-circuit*. O conteúdo e a organização desta tese são então apresentados, após o que se descreve o enquadramento que foi proporcionado pelo projecto ESPRIT 2478.

A leitura deste capítulo permite situar o leitor no panorama global que originou o aparecimento do BST, e introduzir o tema tratado nesta tese. Uma leitura abreviada deste capítulo poderá ser efectuada se se passar directamente para as três últimas secções, embora a sua leitura integral permita compreender melhor o enquadramento que deu origem a este trabalho.

1.1. PANORÂMICA GLOBAL DO TESTE DE CCIS

O teste de cartas de circuito impresso (CCIs) tem lugar essencialmente em três enquadramentos distintos, que se encontram directamente relacionados com as correspondentes etapas no ciclo de vida de um produto: desenvolvimento e verificação de protótipos, produção, e manutenção. Este ciclo de vida pode representar-se como se ilustra na figura 1.1 (Miles et al., 91, p. 377), onde se encontram igualmente assinaladas as correspondentes iterações principais.

Cada um dos enquadramentos cria um cenário de teste com características próprias, quer em termos do espectro de defeitos presentes, quer no que respeita a outros factores, tais como os requisitos de diagnóstico, ou as ferramentas de teste a utilizar. De entre estes três cenários, o teste de produção é aquele que apresenta maior importância, sobretudo em consequência do peso que detém relativamente ao custo final do produto. Os requisitos a este nível consistem essencialmente numa taxa de detecção de defeitos tão elevada quanto possível, e em tempos de teste reduzidos. Uma estratégia que é frequentemente empregue com o objectivo de satisfazer a estes requisitos é a que se ilustra na figura 1.2 (Maunder et al., 90, p. 5), onde estão referidos os dois tipos de equipamentos de teste automático (ATE,

Automatic Test Equipment) mais comuns neste domínio: *in-circuit* (com acesso físico aos nós interiores da CCI a testar), e funcional (com acesso apenas aos conectores de entrada/saída da CCI a testar).

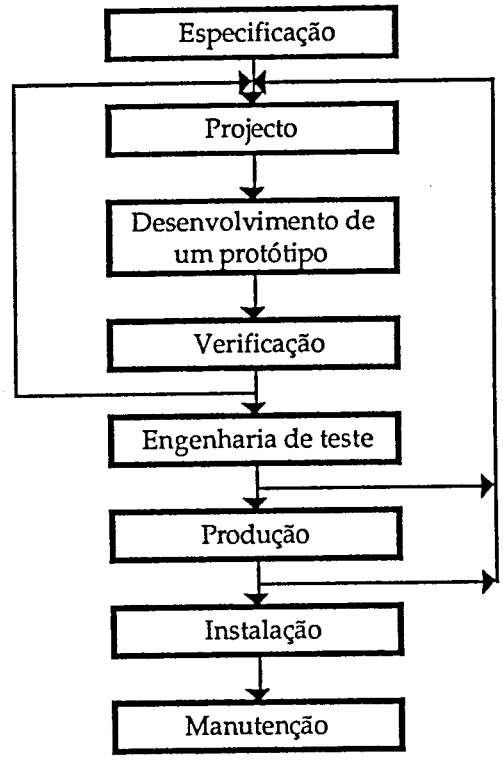


Fig.1.1: Etapas principais no ciclo de vida de um produto.

O teste do tipo *in-circuit* permite a detecção de defeitos do tipo *estrutural*, que correspondem a defeitos nas ligações (curto-circuitos, circuitos abertos), ou nos componentes (componentes mal inseridos, componentes que não funcionam, etc.). Uma vez concluído este teste do tipo estrutural, deverá existir um grau de confiança elevado em que não existam problemas nem com os componentes, nem com as ligações. Apesar de um teste deste tipo poder em determinadas circunstâncias revelar-se suficiente (Schlotzhauer et al., 87), a realização posterior de um teste do tipo funcional é frequente.

O teste do tipo funcional permite igualmente a detecção de defeitos estruturais, mas torna também possível a detecção de defeitos que não se manifestem num teste de baixa velocidade, e que poderão ter origem em deficiências de projecto (margens de segurança mal estabelecidas, por exemplo), em erros do operador, em defeitos que se manifestem de forma intermitente, etc. Adicionalmente, a geração de vectores de teste é efectuada com base num *modelo de faltas* em que se assume uma correspondência entre o espectro de defeitos presente, e um conjunto de faltas que definem o comportamento esperado para o circuito (o

termo *falta* é empregue relativamente ao comportamento lógico, enquanto o termo *defeito* se usa relativamente à causa física que lhe é subjacente). Questões de eficiência no processo de geração dos vectores de teste, e na determinação da percentagem de faltas detectadas (*cobertura de faltas*), podem impor restrições ao modelo de faltas adoptado, o que significa que mesmo um conjunto de vectores de teste com 100 % de cobertura de faltas possa não garantir a completa ausência de defeitos.

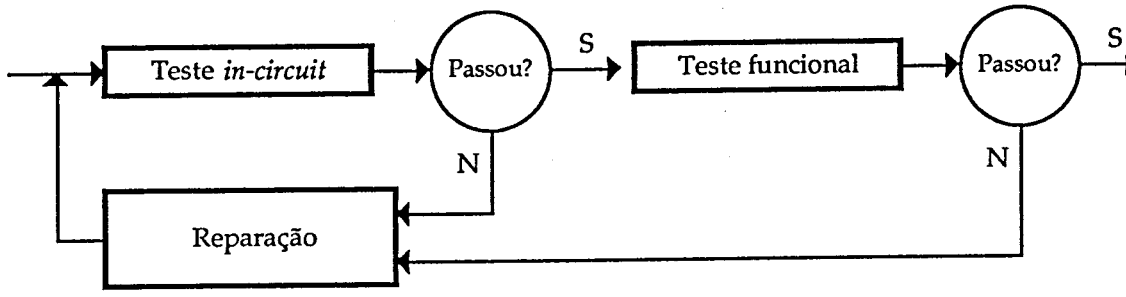


Fig.1.2: Teste de produção em duas etapas.

A estratégia ilustrada na figura 1.2 só começou no entanto a ser empregue após o aparecimento dos equipamentos de teste do tipo *in-circuit*, que surgiram em resposta às dificuldades crescentes que os equipamentos do tipo funcional começaram a experimentar. As razões que motivaram o aparecimento destas duas tecnologias, e as suas características fundamentais, serão brevemente apresentadas nas secções seguintes.

1.1.1. O teste funcional

Tendo constituído a tecnologia de teste fundamental até meados da década de 70, este tipo de equipamentos realizam o teste à CCI apenas a partir dos seus conectores de entrada e saída (Bennetts, 82), (Cortner, 87). A geração do programa de teste recorre normalmente a um modelo de faltas em que se considera cada pino funcional de um componente preso a valores lógicos fixos (*stuck-at-1, stuck-at-0*), e a cobertura de faltas é determinada por recurso à simulação. Deste modo, quer para a geração do programa de teste, quer para a simulação de faltas, deve estar disponível um modelo de simulação para cada componente.

A tecnologia de teste funcional começou no entanto a experimentar problemas crescentes com o aumento da densidade de integração. A presença de componentes cada vez mais complexos reduziu drasticamente os níveis de controlabilidade e de observabilidade (C&O) nos nós interiores das CCIs, dificultando cada vez mais a geração dos programas de teste. A falta de modelos de simulação adequados para estes

componentes veio contribuir para agravar esta situação, levando ao aparecimento da tecnologia de teste *in-circuit*.

1.1.2. O teste *in-circuit*

Os equipamentos de teste *in-circuit* surgiram em finais da década de 70, e apresentaram como vantagem principal eliminar a simulação como etapa necessária à geração do programa de teste (Raymond, 80), (Bateson, 85), (Parker, 87), (Russel, 90). Esta tecnologia baseia-se na utilização de uma matriz-de-agulhas (*bed-of-nails*) que possibilita o acesso físico a todos os nós interiores da CCI, garantindo deste modo a completa C&O que lhe permite dispensar o recurso à simulação.

A realização do teste tem neste caso lugar através de duas etapas principais, a primeira destinada ao teste de ligações, e a segunda ao teste de componentes. O teste de ligações é realizado com a alimentação da CCI desligada, e pretende detectar a ocorrência de curto-circuitos, e de circuitos abertos (os níveis de tensão empregues situam-se nas décimas de Volt). Uma vez confirmada a inexistência de faltas daquele tipo, a alimentação da CCI é ligada, e os componentes presentes são testados. O teste de cada componente tem lugar por aplicação de um conjunto de vectores extraídos de uma biblioteca, pelo que não se torna necessária a existência de modelos de simulação para os componentes presentes. O facto de se utilizarem vectores de teste pré-definidos para cada componente faz no entanto com que seja frequentemente necessário forçar-se num nó o valor lógico oposto ao que é aplicado por uma saída a ele ligada. Este procedimento resulta no que é designado por *backdriving*, e requer que se permita (durante um tempo reduzido) a corrente de circulação necessária para impor o nível lógico pretendido.

A aplicabilidade dos equipamentos de teste *in-circuit* pressupõe como condição fundamental o acesso aos nós interiores da CCI, o que colide necessariamente com os progressos verificados nos domínios do encapsulamento e montagem de componentes. A miniaturização associada à tecnologia de montagem superficial pode facilmente inviabilizar o emprego desta tecnologia de teste, ou torná-la incomportavelmente cara (Kakani, 87), (Bullock, 87). Esta situação conduziu por sua vez a um renovar do interesse pelas técnicas de teste funcional, e contribuiu para o aparecimento de equipamentos de teste híbridos, em que ambas as tecnologias são proporcionadas (*combinational ATEs*). Em qualquer dos casos, as dificuldades experimentadas face a situações mais complexas, como os módulos multi-componente (MCM, *Multichip module*), por exemplo, contribuíram definitivamente para consolidar a necessidade de uma utilização sistemática de metodologias de projecto testável, que proporcionem uma infraestrutura de teste residente na própria CCI.

1.2. A IMPORTÂNCIA DO PROJECTO TESTÁVEL

O emprego de metodologias estruturadas de projecto testável começou por ter lugar à escala do circuito integrado (CI), em que a geração de vectores de teste tem necessariamente que recorrer à simulação de faltas (Eichelberger et al., 77), (Williams et al., 83), (DasGupta et al., 84). A transposição destas metodologias para a escala da CCI foi no entanto atrasada pelos elevados níveis de C&O proporcionados pela tecnologia de teste *in-circuit*. Este facto permitiu que os fabricantes de componentes não encontrassem uma justificação suficientemente forte para proporcionar uma infraestrutura de testabilidade extensível à escala da CCI, o que por sua vez contribuiu para que o acesso às funções de auto-teste existentes nos componentes mais complexos não fosse normalmente proporcionado ao utilizador (Parker, 86).

A aplicabilidade das metodologias de projecto testável à escala da CCI esteve deste modo inicialmente restrita a um número reduzido de casos (Goel et al., 82), (Hansen, 83), e só a partir de meados da década de 80 é que a perspectiva das dificuldades resultantes do aumento na densidade de integração, e das novas técnicas de encapsulamento e montagem de componentes, deu origem ao movimento que conduziria a uma mudança neste domínio (Bennetts, 84), (Beenker et al., 86).

A criação do JETAG (*Joint European Test Action Group*) em Novembro de 1985, por iniciativa da Philips, e que passou em 1986 a integrar igualmente companhias da América do Norte (passando então a designar-se por JTAG), constituiu o ponto de partida para a aceitação generalizada de uma metodologia estruturada de projecto testável, que recebeu a designação de *Boundary Scan Test* — BST (Maunder et al., 91).

1.3. A IMPORTÂNCIA DO BST

O BST foi desenvolvido a partir de metodologias de projecto testável com êxito já comprovado (LeBlanc, 84), (Laurent, 85), e suscitou rapidamente elevados níveis de aceitação entre fabricantes de semicondutores, fabricantes de sistemas, e fabricantes de equipamentos de teste automático (Bennetts et al., 91). O quadro 1.1 sintetiza as características principais que permitem estabelecer termos de comparação com as tecnologias de teste *in-circuit*, e funcional.

A infraestrutura de teste proporcionada pelo BST encontra aplicabilidade em qualquer dos três cenários referidos na secção 1.1 (desenvolvimento e verificação de protótipos, teste de produção, e operações de manutenção), mas é particularmente no domínio do teste de produção que se verifica o seu maior impacto. É já actualmente possível encontrar

exemplos de aplicação desta tecnologia de teste nos domínios em que o acesso directo se revela particularmente difícil, como no teste de módulos multi-componente (Bassett et al., 91), (Posse, 91), ou no teste de *wafers* (Landis, 92). A aprovação do BST como norma do IEEE (IEEE Std 1149.1, 90) veio no entanto criar as condições necessárias para a sua rápida expansão a muitos outros domínios.

Tecnologia	*In-circuit*	Funcional	BST
Controlabilidade / observabilidade dos nós interiores	Directa. Requer o uso de *backdriving* para forçar os nós a testar.	Indirecta. Pode tornar-se muito difícil.	Directa, sem uso de *backdriving*.
Acesso à carta a testar	Por matriz-de-agulhas. Requer contacto físico directo com cada nó.	Por conector, sem contacto físico com nós interiores.	Por conector, sem contacto físico com nós interiores.
Geração de estímulos de teste	Simples, por recurso a bibliotecas, ou a algoritmos de baixa complexidade.	Complexa, pode tornar-se muito demorada.	Simples, por recurso a bibliotecas, ou a algoritmos de baixa complexidade.
Cobertura de faltas	Muito elevada.	Estimada por recurso à simulação de faltas.	Muito elevada.
Resolução no diagnóstico	Elevada.	Pouco elevada. Pode exigir o recurso a técnicas adicionais.	Elevada.

Quadro 1.1: Características principais das tecnologias de teste *in-circuit*, funcional, e BST.

A infraestrutura BST de um CI proporciona também um canal de acesso a funções de auto-teste inseridas no interior dos componentes. Esta possibilidade é particularmente importante, dado que o projecto testável é prática corrente em componentes VLSI, embora a infraestrutura de teste resultante nem sempre seja acessível ao utilizador (Gelsinger, 87), (Gallup et al., 90), (Bishop et al., 90), (Lyon et al., 91). Refira-se no entanto que a presença de uma infraestrutura de teste não significará automaticamente o acesso às funções de auto-teste existentes nestes componentes, sobretudo durante o período inicial em que esta tecnologia vai aumentando a sua implantação (Bruce et al., 91).

Uma infraestrutura BST presente em CCIs cria por sua vez condições para o uso de um controlador de teste dedicado, residente na própria CCI a testar. Esta solução encontra aplicabilidade em qualquer dos três cenários de teste descritos anteriormente (Fleming, 90),

(Sterba et al., 91), mas é em particular nos domínios do teste de produção, e em operações de manutenção, que apresenta maiores potencialidades. O teste de produção será simplificado se os requisitos ao equipamento de teste exterior se restringirem ao acesso às entradas e saídas (E/S) primárias da CCI, e à interacção com o controlador de teste residente. A capacidade de auto-teste que é proporcionada pela existência deste controlador tem igualmente interesse no domínio das operações de manutenção, sobretudo se for suportada uma estratégia de teste hierárquica, em que o controlador de teste presente ao nível do sistema/subsistema possa coordenar as operações de auto-teste nas CCIs que o constituem.

1.4. CONTEÚDO DA TESE

O tema principal desta tese consiste na arquitectura de um controlador de teste para CCIs com BST, e na geração automática do respectivo programa de teste.

A arquitectura proposta para o controlador suporta um conjunto de instruções que implementam directamente as operações elementares identificadas para controlar a infraestrutura BST presente numa CCI. Esta arquitectura admite a existência de uma ou duas cadeias BST na CCI a testar, e proporciona igualmente os recursos necessários para estabelecer o sincronismo com um equipamento de teste exterior, responsável pelo acesso aos pinos de E/S primária.

O próprio controlador pode também dispor de uma infraestrutura BST, incluindo os blocos destinados a permitir que a inicialização e controlo tenham lugar a nível local (na CCI), ou a partir de um outro controlador residente a um nível hierárquico superior. Esta estratégia permite que o mesmo controlador seja empregue a vários níveis, sendo a comunicação entre controladores em níveis diferentes efectuada através das respectivas infraestruturas BST.

A arquitectura do controlador é apresentada de forma modular, de modo a tornar possível a distinção entre o seu núcleo central (constituído por um processador dedicado), e os blocos que permitem a interacção com um barramento de teste a nível hierárquico superior. Esta aproximação permite uma identificação clara dos blocos que deverão sofrer modificações, caso se pretenda suportar outro barramento de teste a nível do sistema/subsistema. Por outro lado, e nos casos em que não se pretenda uma estratégia de teste hierárquica, a implementação do controlador de teste poderá restringir-se ao seu núcleo central, com reduções em termos de complexidade. Esta possibilidade pode revelar-se com interesse, sobretudo se a implementação como um componente programável

localmente (do tipo FPGA, *Field Programmable Gate Array*) for viável.

A geração do programa de teste é feita de forma automática, e toma como pressuposto que os modos de operação proporcionados pela infraestrutura BST presente em cada componente não ultrapassam as três instruções obrigatórias (*Bypass, Sample/Preload, Extest*), e as quatro instruções opcionais (*Idcode, Usercode, Intest, Runbist*), descritas na norma BST (IEEE Std 1149.1, 90). O impacto provocado pela existência de modos de operação mais poderosos é também discutido.

A geração automática do programa de teste (GAPT) suporta as três etapas principais do protocolo de teste para CCIs com BST, que consistem no teste da infraestrutura BST da CCI, no teste das ligações, e no teste de componentes. A etapa correspondente ao teste de ligações inclui igualmente o teste das ligações pertencentes a grupos de componentes não BST, para os quais devem ser gerados exteriormente os respectivos vectores de teste.

O fluxo de dados para cada uma das etapas referidas é descrito, e identifica-se a informação mínima que permite a geração dos respectivos segmentos no programa de teste. Os procedimentos principais empregues em cada uma destas etapas são especificados através de uma pseudo-linguagem de alto nível.

1.5. ESTRUTURA DA TESE

Após o enquadramento geral que foi proporcionado pelo capítulo 1, efectua-se no capítulo 2 uma apresentação sumária dos conceitos principais relativos à tecnologia BST. Esta apresentação inclui uma abordagem às suas limitações principais, e termina com uma referência à sua importância relativamente aos três cenários principais considerados: desenvolvimento e verificação de protótipos, teste de produção, e operações de manutenção.

O teste de CCIs com BST é descrito no capítulo 3, e tem por objectivo principal a caracterização das três etapas principais do respectivo protocolo: teste da infraestrutura BST, das ligações, e dos componentes. A presença de grupos de componentes não BST é considerada, e são discutidas as estratégias possíveis para a realização do respectivo teste.

O capítulo 4 é exclusivamente dedicado ao teste das ligações em CCIs com BST. O facto de ter constituído o objectivo principal que presidiu ao desenvolvimento desta tecnologia de teste justifica que o tratamento deste tema tenha lugar em capítulo próprio. Este capítulo inclui como parte principal uma descrição comparativa dos algoritmos destinados à geração de vectores para a detecção de curto-circuitos entre ligações.

O capítulo 5 efectua a apresentação global do trabalho realizado, quer no que respeita à arquitectura do controlador, quer relativamente à geração automática do programa de

teste. A apresentação que é efectuada neste capítulo serve como introdução para uma descrição mais detalhada em cada um destes domínios principais, que terá lugar nos dois capítulos seguintes.

A arquitectura do controlador desenvolvido constitui o tema principal do capítulo 6. Os blocos fundamentais são aqui descritos, sendo também apresentada a especificação da sequência de transição de estados correspondente a cada uma das instruções suportadas.

O capítulo 7 efectua a apresentação da geração automática do programa de teste (GAPT). A caracterização do fluxo de informação para cada uma das correspondentes etapas, a especificação dos procedimentos principais empregues, e a integração do conjunto de vectores gerados, de modo a produzir um programa de teste final, constituem as partes principais deste capítulo.

Os resultados obtidos, e as perspectivas de trabalho futuro, são apresentados no capítulo 8.

O controlador de teste proposto neste trabalho foi fabricado como um componente LCC com 68 pinos (CMOS, 1.5μ), cujas características são apresentadas no anexo A.

O anexo B ilustra uma aplicação concreta para este controlador, relativamente a uma CCI simples, mas que exemplifica os casos mais complexos presentes em CCIs de maior dimensão (duas cadeias BST, e grupos de componentes não BST). O programa de teste gerado para este caso está igualmente incluído neste anexo.

1.6. ENQUADRAMENTO PROPORCIONADO PELO PROJECTO ESPRIT 2478

O projecto ESPRIT 2478 (*Research into Boundary Scan Test Implementation*) decorreu entre Janeiro de 1989 e Março de 1992, tendo sido liderado pela Philips (Holanda), e incluído as seguintes companhias / instituições: Siemens (Alemanha), Thomson Sintra (França), SGS Thomson Microelectronics (França), Silicon and Software Systems (Irlanda), Elektronikcentralen (Dinamarca), Thomson CSF (França), Matra (França) e INESC (Portugal). O trabalho desenvolvido por este consórcio estruturou-se em torno dos oito subprojectos seguintes:

1: Implementação do BST à escala do CI.

2: Auto-teste com acesso pela infraestrutura BST.

3: Adaptação de ferramentas de CAD.

4: Geração de vectores de teste para a CCI.

5: Formatos de representação (informação de projecto e informação de teste).

6: Equipamento de teste automático.

7: Diagnóstico de faltas.

8: Integração de resultados.

Embora apenas no capítulo 4, e no anexo B, se descreva trabalho que foi parcialmente realizado no âmbito do subprojecto 4 referido acima (Jong et al., 89), (Ferreira et al., 89, 90, 91b, 92c), os resultados obtidos em todos os subprojectos, bem como as reuniões e visitas frequentes que permitiram a coordenação das actividades em curso, constituíram um enquadramento valioso para o trabalho aqui descrito. Neste contexto, assumem uma importância particular os resultados obtidos no âmbito dos subprojectos 1 e 2, quer no que respeita à implementação de uma infraestrutura BST à escala do CI (Siemens, 90b), quer no domínio dos controladores de teste à escala da CCI (Siemens, 90a).

Capítulo 2

O Boundary Scan Test (BST)

Conceitos fundamentais

Custos e benefícios

Limitações principais

Áreas de aplicação

2. O *BOUNDARY SCAN TEST* (BST)

Este capítulo constitui um tutorial sobre os aspectos principais do *Boundary Scan Test* (BST), com o objectivo de proporcionar uma base de conhecimento nos seguintes domínios: conceitos fundamentais, análise de custos e benefícios, limitações principais, e áreas de aplicação.

Os conceitos fundamentais expostos na secção 2.1 proporcionam ao leitor menos familiarizado com esta tecnologia de teste uma aprendizagem/revisão das noções principais. Embora a consulta a várias publicações neste domínio seja frequentemente referida para consolidar questões de pormenor, ou permitir um desenvolvimento mais aprofundado em aspectos específicos, esta secção contem o núcleo de conhecimento essencial para a compreensão do desenvolvimento deste trabalho. As secções restantes têm por objectivo discutir o alcance e a aplicabilidade das potencialidades proporcionadas por esta tecnologia de teste.

A leitura deste capítulo poderá restringir-se à secção 2.1, embora a leitura de todas as secções permita obter uma visão integrada das potencialidades e limitações do BST. O leitor já familiarizado com esta tecnologia de teste poderá passar directamente para o capítulo 3, onde se discutem as questões relacionadas com a sua aplicação prática.

2.1. CONCEITOS FUNDAMENTAIS

O BST constitui uma tecnologia de teste destinada fundamentalmente à escala da CCI, cujo processo de normalização teve início a partir de meados da década de 80 (Beenker, 85), (Laurent, 85), (Maunder et al., 87), (Pradhan et al., 87), e que foi aprovada como norma do IEEE em Fevereiro de 1990 (IEEE Std 1149.1, 90). Apesar de as técnicas fundamentais subjacentes ao BST serem empregues há já mais de uma década (Eichelberger et al., 77), (Goel et al., 82), (LeBlanc, 84), o facto de requererem uma infraestrutura especial disponível no interior do CI (não existente nos componentes da gama comercial) restringiu a sua utilização apenas às grandes companhias verticais, cuja área de actividades se estendesse desde o silício até ao sistema.

2.1.1. O BST à escala da CCI

O teste de CCIs com BST é efectuado a partir de uma infraestrutura composta pelo

encadeamento série de um conjunto de células especiais (células BST), associadas a cada pino funcional dos componentes que suportam esta tecnologia, e que atravessa toda a CCI. Esta infraestrutura está parcialmente ilustrada na figura 2.1, onde se representam igualmente os quatro pinos necessários à sua utilização (por razões de simplicidade, os pinos de alimentação e massa de cada componente não estão representados na figura 2.1). Este conjunto de pinos recebe a designação habitual de TAP (*Test Access Port*), e inclui a entrada (TDI) e saída (TDO) do percurso série, um sinal de relógio (TCK), e um sinal de controlo de modo (TMS). Opcionalmente, pode estar presente um quinto pino, destinado a permitir a inicialização assíncrona da lógica de teste (/TRST). Ao registo formado pelo conjunto de células BST, no interior de cada CI, é dada a designação de *registo BST*.

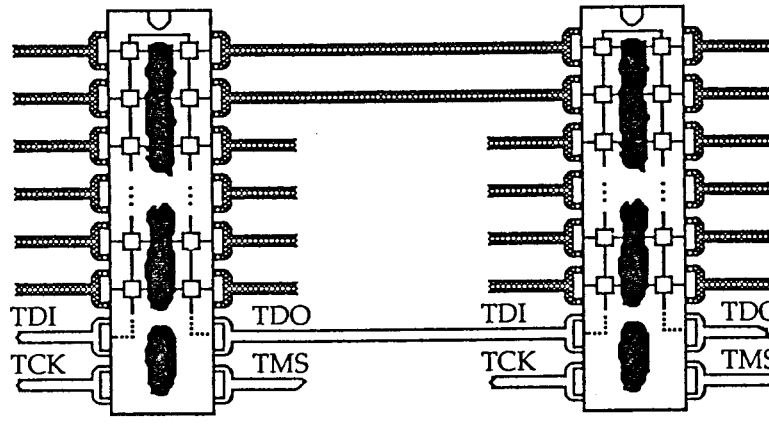
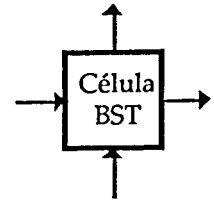


Fig.2.1: Infraestrutura de teste numa CCI com BST.

Cada célula BST tem quatro terminais principais (duas entradas e duas saídas), e a sua integração no circuito tem lugar como se descreve na figura 2.2.

Entrada paralela. Proveniência:

- Pino de entrada (célula BST associada a um sinal de entrada do CI).
- Saída da lógica interna (célula BST associada a um sinal de saída do CI).



Saída paralela. Destino:

- Entrada da lógica interna (célula BST associada a um sinal de entrada do CI).
- Pino de saída (célula BST associada a um sinal de saída do CI).

Fig.2.2: Terminais principais de uma célula BST.

A funcionalidade das células BST tem por objectivo proporcionar uma completa C&O

dos sinais presentes nos pinos a elas associados, sem no entanto interferir no funcionamento normal dos circuitos, de acordo com os requisitos apresentados no quadro 2.1.

- **Transparência:** sempre que não se pretendam efectuar operações de teste, a entrada paralela da célula BST deve estar ligada à saída paralela.
- **Deslocamento** (estímulos/respostas): deve ser possível promover uma operação de deslocamento série ao longo da cadeia formada pelo conjunto das células BST.
- **Observabilidade:** deve ser possível capturar o valor presente na entrada paralela da célula BST, com o objectivo de o deslocar para o exterior.
- **Controlabilidade:** deve ser possível forçar um valor na saída paralela (previamente inserido na célula através de uma operação de deslocamento), independentemente do valor presente na respectiva entrada paralela.

Quadro 2.1: Requisitos de funcionalidade para as células BST.

Os requisitos de funcionalidade descritos no quadro 2.1 permitem um desacopulamento completo entre a periferia e a lógica interna de um CI, o que tem por consequência principal possibilitar que o teste às ligações da CCI seja efectuado por execução cíclica do conjunto de operações descritas no quadro 2.2.

- Deslocar para o interior da cadeia BST os valores lógicos que se pretendem aplicar às ligações (deslocamento).
- Aplicar às ligações os valores presentes no interior das células BST associadas a pinos de saída (controlabilidade). Aplicar também os valores presentes nos canais de teste associados aos nós da CCI a que exista acesso paralelo.
- Capturar os valores presentes nas ligações para o interior das células BST associadas aos pinos de entrada (observabilidade). Capturar também os valores presentes nos canais de teste associados aos nós com acesso paralelo.
- Deslocar para o exterior da cadeia BST os valores capturados (deslocamento).

Quadro 2.2: Sequência de operações para o teste de ligações na CCI.

A existência de ligações que incluam nós com acesso paralelo (por exemplo, pinos de E/S primária da CCI) torna necessária a presença de um conjunto de canais de teste exteriores (independentes da infraestrutura BST) que permitam a aplicação de estímulos, e

a captura de respostas, nestes nós. Repare-se que o deslocamento de novos vectores para o interior da cadeia BST poderá naturalmente ser acompanhado pelo deslocamento para o exterior das respostas capturadas para o vector aplicado anteriormente, pelo que o primeiro e último passos do quadro 2.2 terão lugar em simultâneo.

A figura 2.3 ilustra a sequência de operações descritas no quadro 2.2, relativamente à realização de um teste à continuidade das ligações.

Uma primeira operação de deslocamento envia para as células BST associadas aos pinos de saída os valores que permitem colocar a saída inferior no estado de alta impedância (à esquerda), enquanto o valor a aplicar à ligação é enviado para a célula BST associada ao outro pino de saída (a respectiva célula de controlo receberá o valor que activa o andar de saída). A captura do valor lógico presente nos dois pinos de entrada permitirá então verificar o estado dos percursos assinalados. O deslocamento para o exterior dos valores capturados é acompanhado pela inserção de novos estímulos, que permitirão o teste dos percursos por verificar (à direita).

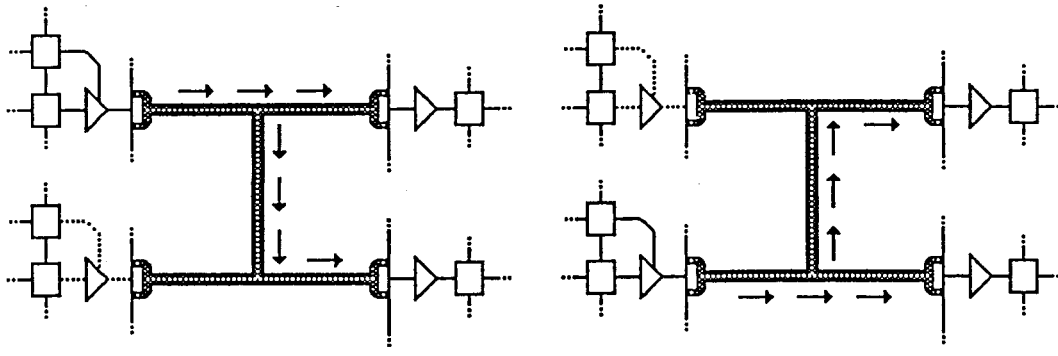


Fig.2.3: Detecção de faltas de continuidade através da infraestrutura BST.

O teste para a detecção de curto-circuitos será igualmente realizado de acordo com a sequência de operações descritas no quadro 2.2, mas agora a partir de um conjunto de vectores que assegura a aplicação de valores lógicos complementares a qualquer par de ligações. Esta condição garantirá a detecção de qualquer curto-circuito, uma vez que os valores capturados nos pinos de entrada de ligações em curto-circuito serão sempre iguais. Resulta daqui que em caso de curto-circuito entre ligações haverá sempre pelo menos um vector no qual existem valores observados que diferem dos valores esperados.

A figura 2.4 ilustra a detecção de curto-circuitos de acordo com o algoritmo da partição binária (Kautz, 74). Os vectores gerados por este algoritmo efectuam uma partição sucessiva do conjunto de ligações em duas metades, permitindo cada vector a

detecção de um curto-circuito entre ligações pertencentes a metades diferentes. A título de exemplo, repare-se que um curto-circuito entre as duas primeiras ligações da figura 2.4 só seria detectado pelo terceiro vector, uma vez que os dois primeiros vectores incluíram estas ligações numa mesma partição.

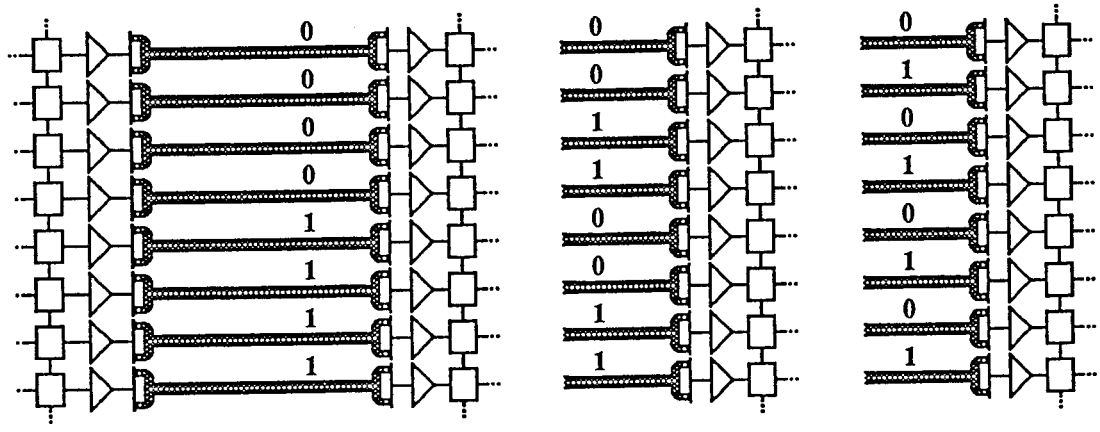


Fig.2.4: Detecção de curto-circuitos através da infraestrutura BST.

2.1.2. O BST à escala do CI

Uma vez caracterizada a funcionalidade do BST à escala da CCI, terá agora lugar a apresentação da infraestrutura lógica correspondente, que deverá estar presente em cada componente.

A figura 2.5 ilustra como a configuração comum para uma célula BST é capaz de garantir os requisitos de funcionalidade descritos no quadro 2.1. Esta configuração chama a atenção para a necessidade de a infraestrutura BST num CI incluir igualmente os blocos responsáveis pela geração dos sinais que permitem cadenciar a sequência de operações a ter lugar, e que controlam o modo de funcionamento pretendido para as células BST. Esta infraestrutura incluirá ainda um conjunto de registos adicionais (para além do registo BST), e pode representar-se como se ilustra na figura 2.6.

Repare-se que qualquer dos registos de dados representados (registo BST, registos do utilizador, registo de identificação, ou registo de *bypass*) pode ser colocado entre TDI e TDO (através do *multiplexer* superior), de acordo com o conteúdo do registo de instrução. Por sua vez, o bloco designado como *Controlador do TAP* é o que determina (através do *multiplexer* inferior) se o registo colocado entre TDI e TDO é um dos registos de dados, ou o registo de instrução. Este controlador do TAP é constituído por uma máquina de estados

comandada pela entrada TMS, e cujo diagrama de estados contem dois percursos fundamentais (para deslocamento de dados, ou de instruções), tal como se representa na figura 2.7.

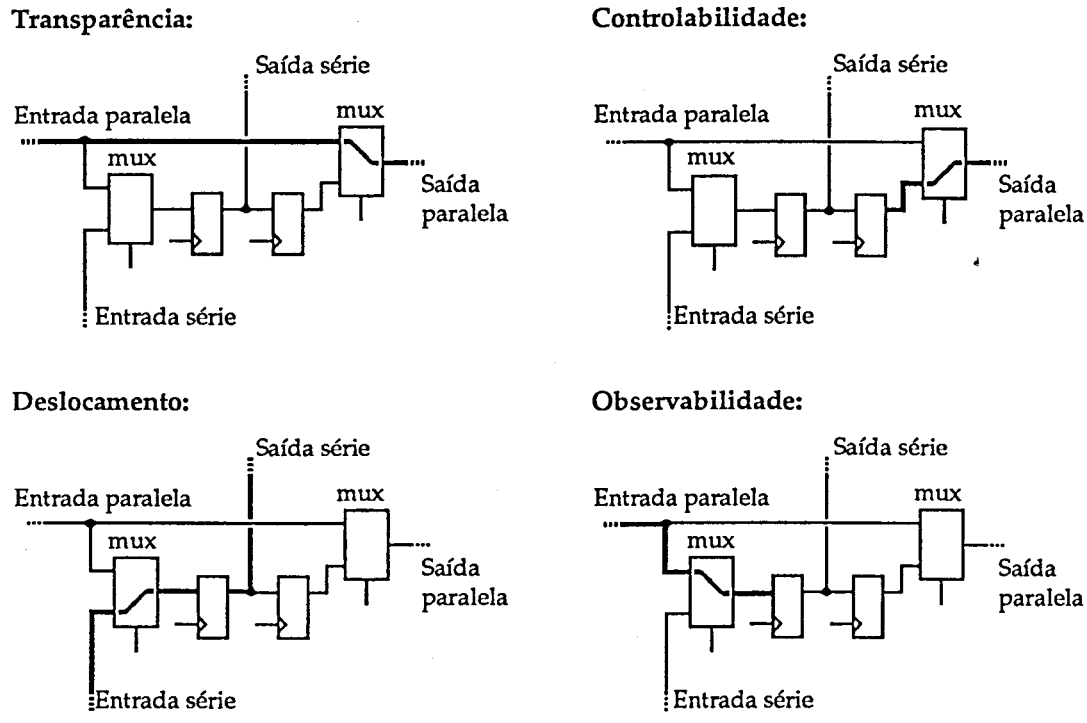


Fig.2.5: Célula BST que garante os requisitos de funcionalidade descritos no quadro 2.1.

De acordo com os elementos apresentados, o acesso a qualquer dos registos de dados pode ser realizado através da sequência de operações descrita no quadro 2.3.

- O controlador do TAP é colocado no estado *Shift-IR*.
- A instrução requerida é deslocada para o interior do registo de instrução.
- O controlador do TAP é colocado no estado *Shift-DR*.
- O acesso ao registo de dados pretendido tem lugar através de uma operação de deslocamento.

Quadro 2.3: Controlador do TAP: protocolo de acesso a um registo de dados.

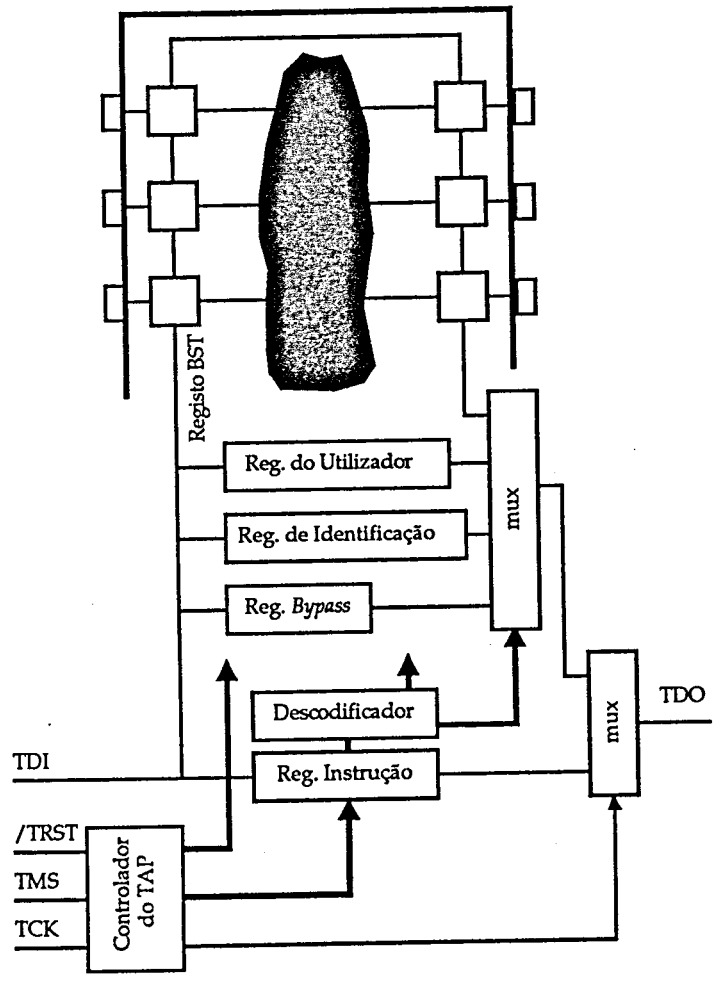


Fig.2.6: Infraestrutura BST num CI.

Os registos de dados ilustrados na figura 2.6 relacionam-se com as operações principais possíveis através da infraestrutura BST de um CI, e a sua utilidade pode ser resumida da seguinte forma:

- O registo BST destina-se fundamentalmente a permitir o teste das ligações na CCI. Repare-se no entanto que a funcionalidade enunciada no quadro 2.1 permite igualmente que o registo BST seja usado para testar a lógica interna do CI, por deslocamentos sucessivos de estímulos e respostas. Este processo, embora perfeitamente aceitável para o teste de ligações (onde o número de vectores se situará muito provavelmente na ordem das poucas dezenas), revela-se porém pouco eficiente para o teste da lógica interna (onde o número de vectores pode facilmente atingir as dezenas de milhar).
- Os registos do utilizador possibilitam o acesso a funções de auto-teste do CI, através da sua infraestrutura BST. Esta possibilidade ultrapassa as limitações

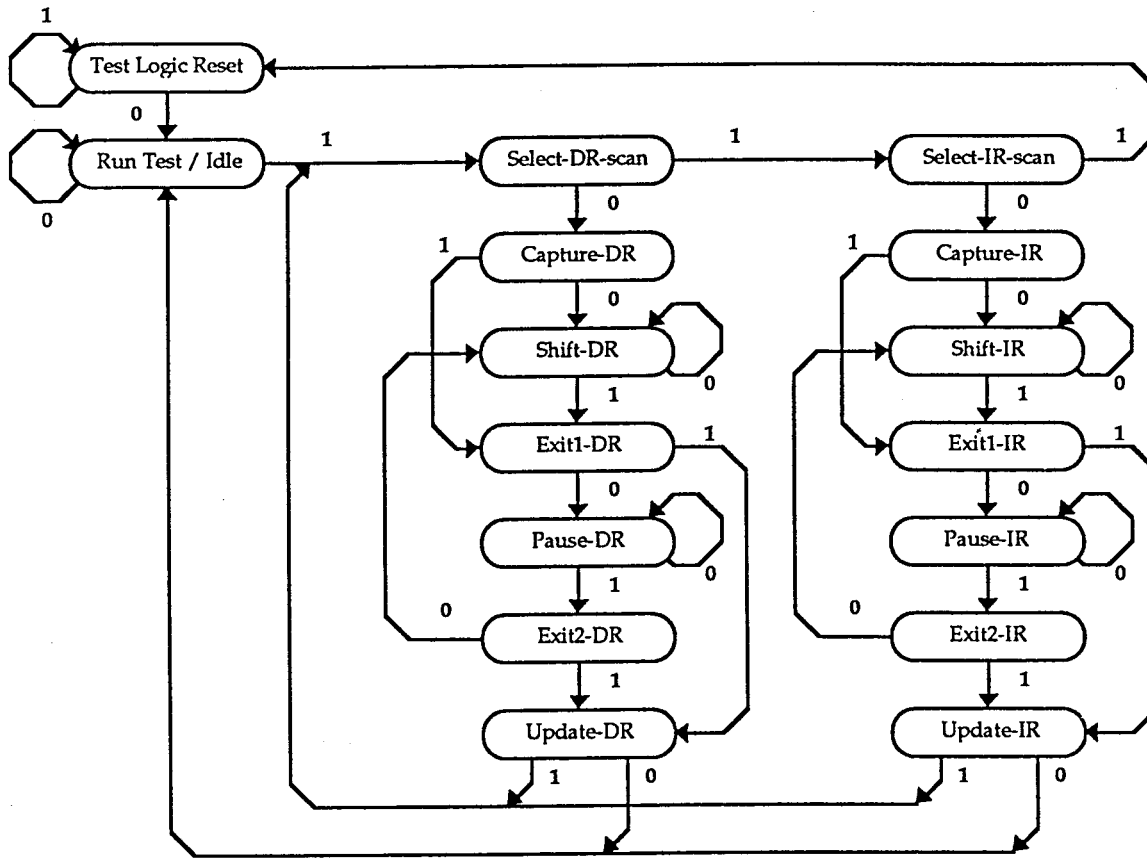


Fig.2.7: Diagrama de estados do controlador do TAP.

referidas para o teste interno através do registo BST, uma vez que o auto-teste é realizado por aplicações sucessivas de vectores (gerados internamente) e compactação de respostas, à razão de um vector por cada ciclo de relógio. O facto de a infraestrutura BST proporcionar um mecanismo para acesso a funções de auto-teste é tanto mais importante quanto é sabido que estas funções estão presentes na maior parte dos componentes mais complexos, apesar de não serem normalmente acessíveis ao utilizador. Estes registos têm existência opcional.

- O registo de identificação possibilita a leitura de um código derivado da especificação JEDEC (JEDEC, 86), com o objectivo de permitir verificar se o componente inserido em cada posição é o correcto. Também este registo tem existência opcional.
- O registo de *bypass* permite reduzir a uma única célula o percurso série entre TDI e TDO. Esta possibilidade é sobretudo importante se atendermos a que podem existir dezenas ou centenas de células por registo BST. O acesso a um número

reduzido de componentes seria nestas circunstâncias moroso e ineficiente, se não houvesse possibilidade de seleccionar o registo de *bypass* nos componentes a que não é pretendido o acesso.

2.2. CUSTOS E BENEFÍCIOS

Uma análise pormenorizada de custos e benefícios para uma tecnologia de teste requereria a consideração de modelos complexos, que integrassem o seu impacto ao longo do ciclo de vida de um produto (Dislis et al., 89), (Miles et al., 91), e que excede o âmbito desta secção. Existe no entanto um conjunto de aspectos gerais que não devem deixar de ser referidos, e que se encontram sintetizados no quadro 2.4 (Sedmak et al., 90).

Benefícios:
• Reutilização de testes. Os testes de produção de um CI (incluindo os que exercitam funções de auto-teste), acessíveis através da sua infraestrutura BST, podem ser reutilizados com o CI inserido na CCI, e com a CCI inserida no sistema. • Os índices de C&O proporcionados pelo BST permitem o desenvolvimento de equipamentos de teste mais eficientes, e mais económicos. • A infraestrutura BST de uma CCI pode constituir um auxiliar valioso na fase de verificação de protótipos. • Tempos de teste inferiores, sobretudo para as CCIs cujo teste só fosse possível por técnicas de teste funcional.
Custos:
• É necessária uma área de silício adicional para a lógica BST. • São requeridos (4 ou 5) pinos adicionais para o TAP. • Aumento no tempo de projecto, quer à escala do CI, quer da CCI. • A inclusão de células BST pode degradar algumas características de funcionamento, nomeadamente os tempos de propagação. • Consumo adicional devido à lógica BST. • Pode reduzir a fiabilidade, e o rendimento no fabrico dos componentes (*Yield*).

Quadro 2.4: Custos e benefícios principais da tecnologia BST.

A área de silício requerida para a lógica de teste, e o número de pinos adicionais, são os custos que mais frequentemente são referidos como contrapartida às vantagens

proporcionadas pela inclusão da infraestrutura BST. No entanto, é interessante referir a este propósito que um número significativo de componentes vê a sua área limitada pelo número de pinos (devido ao número de *bonding pads* necessários), e não pela área correspondente à sua lógica interna. Nestes casos, o custo da inclusão do BST seria essencialmente imposto pelo tempo adicional de projecto necessário, e que pode ser reduzido a apenas algumas horas, se forem empregues ferramentas destinadas a efectuar esta inclusão de forma automática (Muris, 90), (Chiles et al., 91). Por outro lado, e atendendo ao elevado número de pinos que cada vez mais caracterizam as novas gerações de componentes (parte dos quais seriam eventualmente dedicados a operações de teste, se o BST não existisse), a inclusão de um TAP com quatro pinos representará na maior parte dos casos um encargo menor.

2.3. LIMITAÇÕES PRINCIPAIS

A tecnologia BST apresenta no entanto um conjunto de limitações que convém referir, e que são sobretudo consequência de ter sido desenvolvida explicitamente para ultrapassar um conjunto de dificuldades num domínio específico: o teste estrutural de CCIs digitais. Isto significa que, para além das limitações inerentes à própria tecnologia BST, a infraestrutura de teste proporcionada encontra limitações adicionais quando confrontada com questões específicas de domínios de teste complementares.

2.3.1. Capacidade de diagnóstico

Apesar dos excelentes níveis de C&O proporcionados pela infraestrutura BST, existe uma limitação de diagnóstico que é inerente ao facto de esta infraestrutura fazer parte dos próprios componentes.

O facto de um teste de continuidade se estender desde uma célula BST até outra célula BST, tal como se ilustrou na figura 2.3, faz com que a detecção de um defeito deste tipo possa requerer testes adicionais para localizar a interrupção, uma vez que esta pode ter lugar no interior dos componentes (*bonding wire*, por exemplo).

Também as operações de diagnóstico relativamente a curto-circuitos podem encontrar dificuldades, uma vez que um defeito deste tipo resulta normalmente num valor analógico de tensão, comum às ligações envolvidas. Este valor será capturado através de uma operação de leitura digital (na célula BST associada aos pinos de entrada), de modo que estará sempre presente um grau de incerteza quanto ao valor esperado.

2.3.2. Velocidade

A necessidade de efectuar uma operação de deslocamento ao longo da cadeia BST, por cada vector a aplicar, torna ineficiente este procedimento para o teste de blocos que requeiram um elevado número de vectores. Esta limitação far-se-á sobretudo sentir quando existirem componentes BST sem auto-teste interno, ou ainda grupos de componentes não BST, casos em que o número de vectores necessários ao respectivo teste pode atingir as dezenas de milhar.

Repare-se no entanto que esta limitação é sobretudo consequente da existência de blocos com reduzidas características de testabilidade, que serão normalmente evitados se forem empregues metodologias de projecto testável (DFT). Alternativamente, e para os casos em que esta situação não puder ser evitada, começam a surgir no mercado componentes BST que permitem efectuar localmente o teste dos blocos de lógica que não disponham desta tecnologia (Whetsel, 91), (Kritter et al., 91), (Raghavachari, 91).

2.3.3. Teste analógico

O teste de blocos analógicos constitui uma das áreas em que o BST encontra maiores limitações, embora existam alguns trabalhos neste domínio (Fasang, 89), (Vefling et al., 90), (Hirzer, 90).

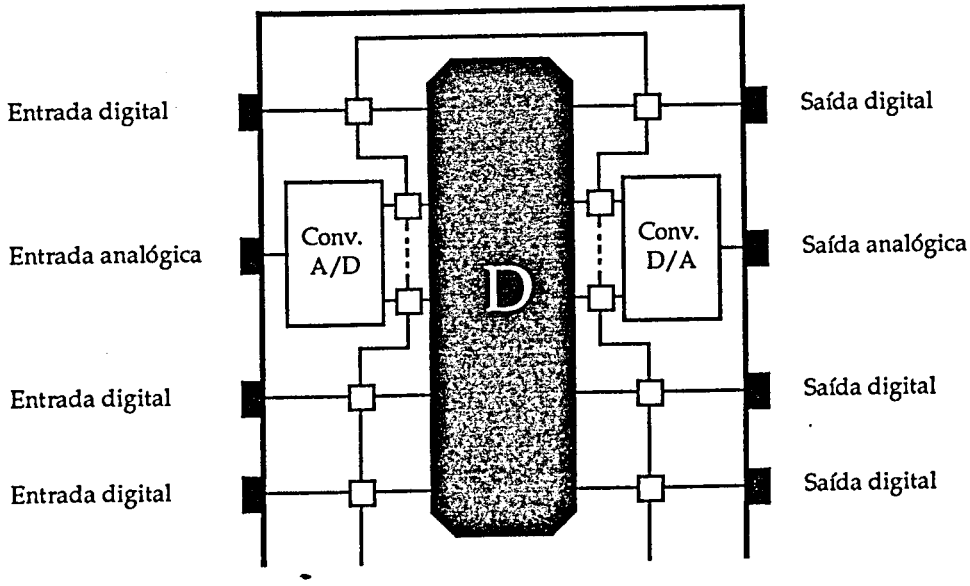


Fig.2.8: Acesso aos pinos analógicos de um componente, através da sua infraestrutura BST.

O acesso às entradas e saídas analógicas de um componente é possível através da sua infraestrutura BST, tal como se ilustra na figura 2.8, embora esta configuração se restrinja ao caso de a função do componente ser predominantemente digital.

O caso mais geral de se pretender dispor de C&O em nós interiores de um bloco analógico é bastante mais complexo, quer por questões de economia (área de silício), quer devido à influência exercida sobre os sinais presentes nestes nós. A figura 2.9 ilustra uma aproximação descrita em (Vefling et al., 90, cap. 4), mas com reduzida aplicabilidade prática, sobretudo devido à área adicional de silício que seria requerida.

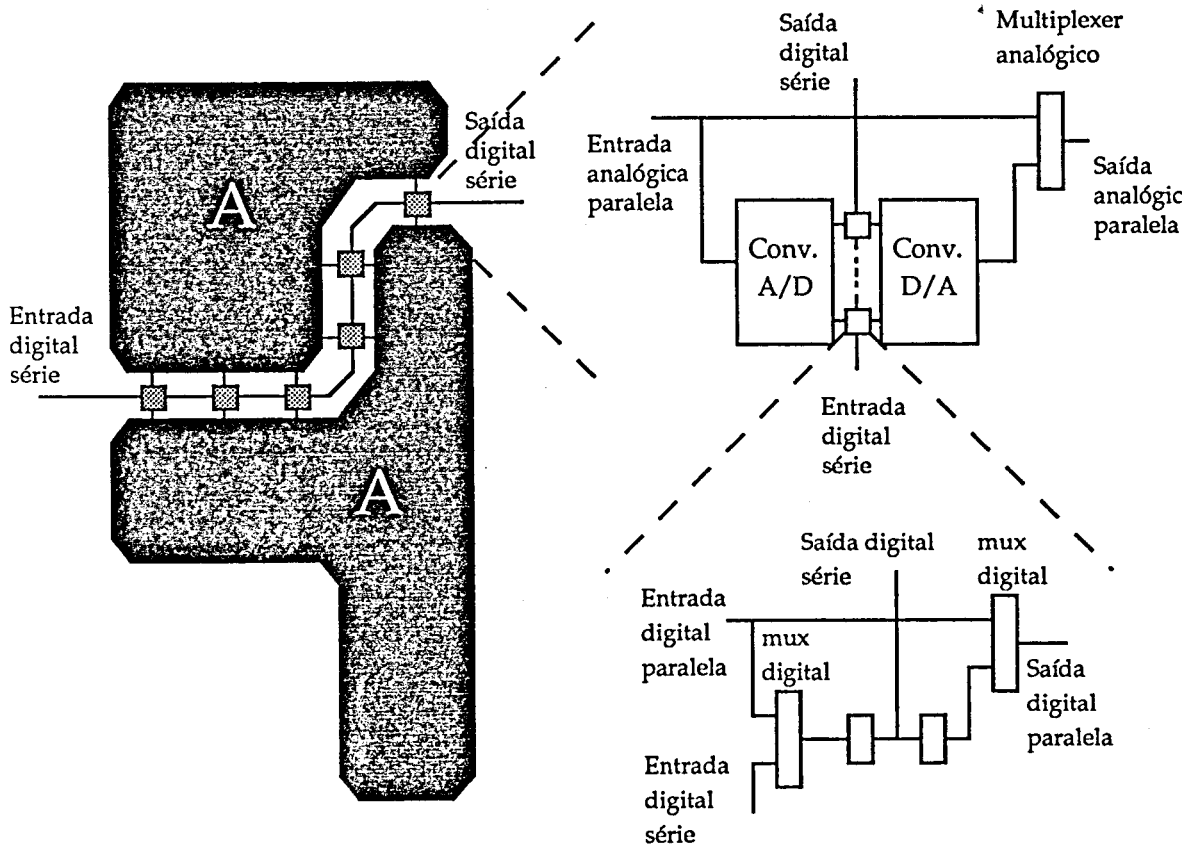


Fig.2.9: Acesso ao interior de blocos analógicos, por técnicas BST.

Repare-se no entanto que esta aproximação poderá já apresentar interesse se for considerada à escala da CCI, com o objectivo de incrementar os níveis de C&O no interior de grupos de componentes analógicos complexos.

2.3.4. Teste hierárquico

A infraestrutura de teste proporcionada pelo BST não é verdadeiramente hierárquica, no sentido em que a sua utilização aos diferentes níveis não é directa. Apesar do reconhecido mérito que esta infraestrutura apresenta para a realização de um teste estrutural à escala da CCI, já a realização de testes eficientes à escala do CI requer mecanismos alternativos aos que são proporcionados para a CCI, e que se traduzem em controladores de auto-teste interno, acessíveis por intermédio da infraestrutura BST (Riessen et al., 89), (Maierhofer, 90). Também a realização de testes ao nível do sistema seria pouco prática se fosse necessário controlar a longa cadeia BST formada pela série das cadeias em cada CCI, ou se fosse necessário proporcionar TAPs individuais para cada CCI. Embora estas limitações possam ser ultrapassadas pela existência de controladores de teste residentes em cada CCI (à semelhança dos controladores de auto-teste nos CIs), a existência de uma proposta de norma para um barramento de teste ao nível do sistema (IEEE P1149.5 Std, 92) serve para ilustrar que o impacto do BST será essencialmente à escala da CCI.

2.4. ÁREAS DE APLICAÇÃO

O impacto principal da tecnologia BST ocorre no domínio do teste de produção, e em particular relativamente ao teste estrutural de CCIs digitais com reduzidos índices de testabilidade (Parker, 89). As potencialidades proporcionadas pela correspondente infraestrutura de teste permitem-lhe no entanto encontrar aplicabilidade em todas as fases do ciclo de vida de um produto (Sterba et al., 91). Esta secção efectuará uma apresentação sucinta do impacto desta tecnologia relativamente a três cenários de teste principais: desenvolvimento e verificação de protótipos, teste de produção, e operações de manutenção (Fleming, 90).

2.4.1. Desenvolvimento e verificação de protótipos

A aplicabilidade do BST no desenvolvimento e verificação de protótipos encontra-se ainda pouco explorada, mas será com certeza uma área em que esta tecnologia assumirá um papel de relevo (Halliday et al., 89).

O desenvolvimento de um novo produto implica frequentemente iterações entre as fases de projecto e verificação, no decorrer das quais se torna necessário o emprego de diverso tipo de equipamentos, tais como sistemas de desenvolvimento, analisadores lógicos, etc. Os requisitos relativamente aos nós a observar enfrentam no entanto o mesmo tipo de

dificuldades já referidas para o teste *in-circuit*, pelo que o acesso (virtual) através da infraestrutura BST se pode revestir de grande interesse. Este acesso pode proporcionar a mesma (ou mesmo superior) capacidade de observação sobre o estado dos circuitos, mas agora apenas através dos quatro pinos do TAP da CCI.

A aplicabilidade do BST neste domínio depende naturalmente do aparecimento de um suporte lógico (*software*) adequado, que permita um uso eficiente da infraestrutura BST a partir de uma estação de trabalho, ou de um computador pessoal. Repare-se ainda que as potencialidades proporcionadas pela infraestrutura BST podem igualmente tornar possível o acesso a registos internos de vários componentes, não directamente acessíveis do exterior, do mesmo modo que possibilitarão abrir janelas de visualização sóbre o estado de diferentes partes do circuito, o que por acesso directo obrigaria ao reposicionamento das pontas de captura.

2.4.2. Teste de produção

O teste de produção constitui o domínio onde a tecnologia BST apresenta o seu maior impacto, e são já conhecidas aplicações que se estendem desde o teste de *wafers* (Landis, 92), (Schwederski et al., 91), até ao teste estrutural (Hansen, 89), (Robinson et al., 90), (Lefebvre, 91), (Moore, 91) e ao teste funcional (Lefebvre, 90), (Sweeney, 90), (Wagner et al., 91) de CCIs.

A aplicabilidade do BST para a realização de testes paramétricos na produção de CIs é também discutida em (Fleming, 90, p. 134), onde se realça a importância de a infraestrutura BST dispensar a propagação através da lógica interna dos componentes, e que em certos casos pode chegar a requerer sequências com dezenas de milhar de vectores.

2.4.3. Operações de manutenção

A aplicabilidade do BST em operações de manutenção pode igualmente revelar-se de grande importância, desde que estejam presentes os recursos que permitam tirar partido da infraestrutura de teste disponível nos equipamentos. Estes recursos podem corresponder a controladores de teste residentes nos próprios equipamentos (à escala da CCI, e do sistema), circunstância em que todos os módulos poderão dispor de capacidade de auto-teste. No entanto, e mesmo que esta possibilidade não seja explorada, o aparecimento de equipamentos de teste baseados em computadores portáteis permitirá uma capacidade de detecção e de diagnóstico mais completa, e com menores custos (TI ASSET, 90), (Goor et

al., 91), (Ferreira et al., 91c).

Uma outra área a ser explorada consiste na possibilidade de se utilizar a infraestrutura BST para permitir a aquisição regular de informação sobre o estado do sistema, durante o respectivo funcionamento. O facto de as operações de observabilidade (restritas à captura e deslocamento de valores através da infraestrutura BST) não impedirem o normal funcionamento dos componentes abre excelentes perspectivas para aplicações no domínio dos sistemas tolerantes a avarias, em que o estado de funcionamento deve ser permanentemente observado.

Capítulo 3

O teste de CCIs com BST

Terminologia

Caracterização de um modelo de faltas

Protocolo de teste

Avaliação da cobertura de faltas

3. O TESTE DE CCIs COM BST

O tema principal deste capítulo consiste na discussão dos procedimentos que permitam um uso eficiente da infraestrutura BST de uma CCI. Esta discussão é precedida pela caracterização de um modelo de faltas, que inclui a análise da distribuição percentual dos defeitos mais comuns à escala da CCI, e também uma breve abordagem aos modelos empregues nas tecnologias de teste *in-circuit* e funcional.

Um protocolo para o teste de CCIs com BST será então apresentado, onde se consideram os aspectos relativos à sincronização das cadeias BST com recursos de teste exteriores, e à sincronização entre cadeias BST. O teste de grupos de componentes não BST será também abordado nesta secção, quer no que respeita a estratégias alternativas para a sua realização, quer relativamente à integração dos respectivos vectores, no sentido de permitir o teste através da infraestrutura BST. A parte final deste capítulo discutirá a avaliação da cobertura global de faltas resultante para o protocolo descrito, e identificará as dificuldades que lhe são inerentes.

A apresentação do protocolo de teste para CCIs com BST, efectuada na secção 3.3, constitui um dos passos fundamentais no sentido de definir uma metodologia que permita explorar convenientemente as potencialidades proporcionadas por esta tecnologia de teste. Como tal, a leitura desta secção deverá preceder a passagem para o capítulo seguinte. A leitura da secção 3.1 é também necessária, sendo então possível efectuar uma passagem mais rápida por este capítulo através de uma leitura superficial das secções restantes.

3.1. TERMINOLOGIA

O objectivo desta secção consiste em descrever um conjunto de conceitos principais, que servirão de base à terminologia adoptada na descrição deste trabalho. A definição destes conceitos torna-se necessária em consequência das características específicas da tecnologia BST, mantendo-se os significados de todos os restantes termos com utilização consagrada no domínio do teste em electrónica. Por conveniência de leitura, limitar-se-à esta primeira abordagem à definição de um conjunto de termos principais, ao qual se irão adicionando novos elementos, à medida que a descrição de novos assuntos o for solicitando.

Os termos *nó de teste, ligação,* e *barramento,* traduzem conceitos que se relacionam hierarquicamente, e que serão frequentemente empregues nos capítulos seguintes. No contexto deste trabalho, designa-se por *nó de teste* a entrada ou a saída (paralelas) de uma

célula BST, ou ainda a entrada ou a saída dos canais de teste destinados a permitir o acesso paralelo. Por sua vez, o termo *ligação* (associado a "pista" da CCI) refere-se ao conjunto de percursos elementares que asseguram a conectividade entre um conjunto de nós de teste electricamente ligados entre si.

O conjunto de nós de teste divide-se em dois grupos principais, no que respeita a funções de controlabilidade e observabilidade (C&O) das ligações, que se designam por *nós de controlabilidade das ligações* (nós do tipo CL), e *nós de observabilidade das ligações* (nós do tipo OL). Os nós do tipo CL permitem controlar o nível lógico a aplicar a uma ligação, e os nós do tipo OL permitem capturar (observar) o nível lógico presente numa ligação. A figura 3.1 ilustra uma ligação com nós destes dois tipos, que aqui surgem associados a diversos tipos de pinos.

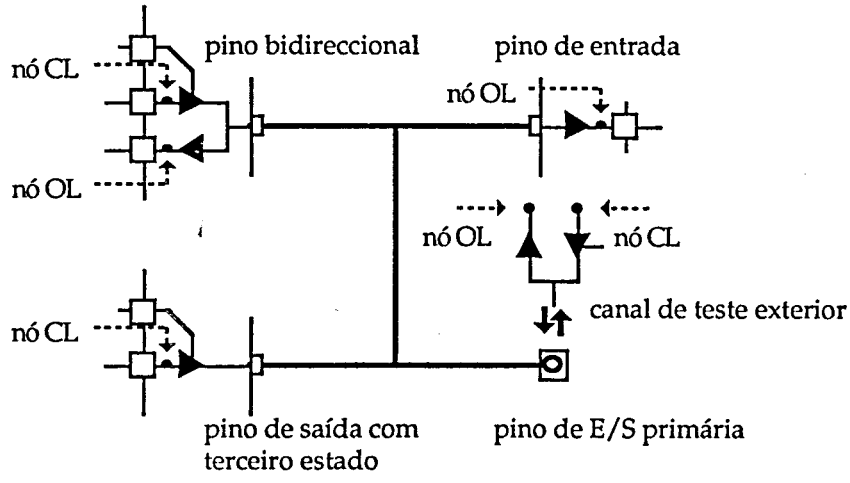


Fig.3.1: Ligação na CCI, com nós dos tipos CL e OL.

Repare-se que, de acordo com as definições apresentadas, os andares de entrada e de saída (*buffers*) passam a fazer parte das ligações entre componentes, tal como os fios de ligação (*bonding wires*) entre os terminais de entrada e saída (*I/O pads*) de um CI, e os seus pinos de comunicação com o exterior. Tal como já se referiu no capítulo anterior, esta definição resulta do facto de a infraestrutura BST fazer parte dos próprios CIs, pelo que as ligações entre componentes incluem os elementos referidos.

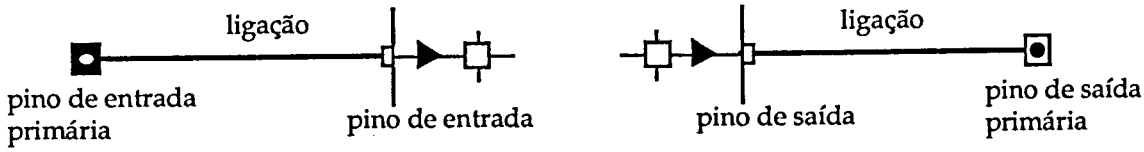
Deste modo, o conceito de nó de teste não coincide com o de *pino*, mantendo-se para este último o entendimento generalizadamente aceite, e que tanto se pode referir aos pinos de um CI, como aos pinos de um conector (E/S primária da CCI). Resulta deste modo que um pino pode estar associado a mais do que um nó de teste, e.g. um pino bidireccional, tal como se ilustra na figura 3.1. O conjunto de pinos existentes numa CCI divide-se por sua

vez em vários tipos, de acordo com a sua funcionalidade, tal como se descreve no quadro 3.1:

Pinos de CIs	de entrada
	de saída, sem terceiro estado
	de saída, com terceiro estado
	bidireccionais
Pinos de entrada/saída (E/S) primária	de entrada
	de saída

Quadro 3.1: Tipos de pinos considerados numa CCI.

Repare-se que a classificação de pinos como entradas, ou como saídas, é feita sob o ponto de vista da CCI, quando se tratem de pinos de E/S primária (E/S da CCI), ou sob o ponto de vista do CI, quando se tratem de pinos dos componentes (E/S dos CIs), tal como se ilustra na figura 3.2.



(a) Pinos de entrada. (b) Pinos de saída.

Fig.3.2: Classificação dos pinos como entradas, ou como saídas.

A expressão *pino BST* será por sua vez usada para designar um pino de um componente a que estejam associadas uma ou mais células BST, e pretende distinguir entre este tipo de pinos, e os pinos de E/S primária da CCI, ou os que pertençam a componentes não BST.

A expressão *ligação BST* refere-se a uma ligação em que todos os seus pinos sejam pinos BST, ou pinos de E/S primária da CCI. Deste modo, as ligações ilustradas nas figuras 3.1 a 3.3 são todas ligações BST, mas nem todas contêm apenas pinos BST.

A um grupo de ligações em que existam nós do tipo CL a partilhar um sinal de controlo comum é dada a designação de *barramento*. A figura 3.3 ilustra um conjunto de ligações nestas circunstâncias.

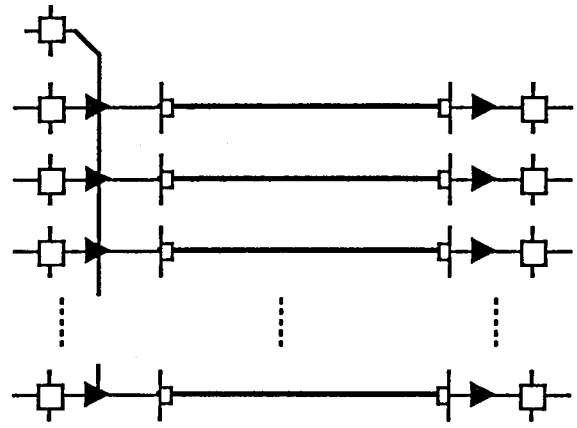


Fig.3.3: Exemplo de um barramento.

3.2. CARACTERIZAÇÃO DE UM MODELO DE FALTAS

Tendo a tecnologia BST sido desenvolvida em consequência dos problemas crescentes enfrentados pelo teste à escala da CCI, assume particular importância a formulação da relação existente entre a capacidade de detecção e diagnóstico proporcionada por esta tecnologia, e o espectro de defeitos mais comuns a este nível. Este espectro de defeitos será relacionado com os modelos de faltas frequentemente adoptados nas tecnologias anteriores (*in-circuit*, funcional), e servirá de base à descrição do modelo de faltas adoptado neste trabalho.

3.2.1. Espectro de defeitos numa CCI

A caracterização apresentada nesta secção pretende essencialmente identificar os tipos de defeitos mais comuns à escala da CCI, e discutir os aspectos mais relevantes envolvidos na transposição entre a causa física (defeito) e o modelo lógico (falta) que lhe será associado.

A obtenção de elementos que permitam caracterizar o espectro de defeitos no teste de CCIs revela-se difícil, sobretudo em consequência de dois factores principais:

- O espectro de defeitos em CCIs está relacionado com o cenário de teste em questão, sendo de esperar que a incidência de cada tipo de falta varie de caso para caso.
- O espectro de defeitos em CCIs está frequentemente relacionado com aspectos do controlo de qualidade, o que significa que estão em causa factores susceptíveis de afectar directamente a competitividade de cada companhia. Neste sentido, é

natural que surjam reservas quanto à publicação de resultados neste domínio.

No que respeita ao primeiro factor, convirá recordar que a tecnologia BST encontra aplicabilidade em praticamente todos os cenários associáveis ao teste de CCIs, desde a verificação/validação de protótipos (Halliday et al., 89), com passagem pelo teste estrutural (Hansen, 91), e estendendo-se mesmo ao domínio do teste funcional (Lefebvre, 90), (Wagner et al., 91). A aplicabilidade do BST em operações de manutenção pode também assumir grande interesse, seja pela possibilidade de se utilizarem testadores compactos baseados em computadores portáteis, seja pelas potencialidades que esta infraestrutura de teste proporciona para a implementação de soluções de auto-teste à escala da CCI. A caracterização de um espectro de defeitos deverá deste modo levar em conta que a respectiva distribuição percentual será naturalmente influenciada pelo cenário de teste em questão. A título de exemplo, e embora seja plausível que os componentes incorrectamente inseridos na CCI assumam uma importância não desprezável no espectro de defeitos característico de um teste de produção, esta situação já não será plausível em termos dos defeitos encontrados em operações de manutenção.

Também as características individuais de cada situação têm naturalmente influência sobre a distribuição percentual referida. No teste de produção, por exemplo, a utilização de diferentes processos de soldadura terá naturalmente influência sobre a distribuição percentual das faltas nas ligações, do mesmo modo que a inserção automática de componentes deverá contribuir para reduzir o número de componentes trocados. Estas variações contribuem por sua vez para dar origem a um intervalo de variação no espectro de defeitos, para cada cenário de teste.

Se a análise a efectuar se restringir ao teste de produção, existem disponíveis alguns elementos que permitem prever que o espectro de defeitos para este cenário não deverá afastar-se da caracterização seguinte (Bateson, 85, p. 7), (Bagge et al., 89, pp. 18 e 19), (Rohde & Schwarz, 89, p. 4), (Hansen, 90, pp. 82 e 83):

- Defeitos estruturais: cerca de 85 a 95 %. Os defeitos estruturais estão principalmente relacionados com as ligações na CCI (curto-circuitos, ou ligações interrompidas, correspondentes a cerca de 65 % do total), mas os relacionados com os componentes assumem também uma importância de relevo. Entre estes, apresentam maior incidência os que dizem respeito a componentes trocados, componentes mal inseridos (com a orientação errada, por exemplo), e ainda a componentes avariados.
- Defeitos nas especificações de funcionamento: cerca de 5 a 15 %. Este tipo de defeitos corresponde frequentemente aqueles que não são detectados por um primeiro teste estrutural da CCI (tradicionalmente realizado pela tecnologia de

teste *in-circuit*), mas podem também relacionar-se com outras causas, como por exemplo com a existência de blocos projectados com margens de segurança mínimas, ou mesmo inexistentes (em termos de tempos de propagação, tempos de transição, margens de ruído, etc.).

A utilização da tecnologia BST na detecção do segundo tipo de defeitos está ainda muito pouco explorada, embora comecem a aparecer trabalhos publicados neste domínio (Whetsel, 90), (Lefebvre, 90), (Wagner et al., 91). A dificuldade principal neste campo consiste na necessidade de sincronizar o funcionamento da lógica BST, em modo de amostragem (instrução *Sample/Preload*, que é uma das três instruções obrigatórias exigidas pela norma IEEE 1149.1), com o funcionamento real do circuito, e que constitui uma condição necessária para que este funcionamento possa ser testado por operações de captura e análise, realizadas de forma cíclica.

Já relativamente à detecção dos defeitos estruturais, que constituem a grande maioria dos defeitos que ocorrem na produção de CCIs, a tecnologia BST reúne excelentes características de detecção. A verificação da funcionalidade da infraestrutura BST da CCI (a realizar em primeiro lugar), que incluirá o acesso aos registos de identificação nos componentes que dispuserem desta facilidade, é capaz por si só de detectar a maior parte dos defeitos estruturais referidos para os componentes. Os restantes serão por sua vez detectáveis por uma etapa posterior dedicada explicitamente ao teste de componentes. Mas é na detecção dos defeitos estruturais presentes nas ligações da CCI que a tecnologia BST apresenta a sua maior importância, e relativamente às quais se levanta a questão da caracterização de um modelo de faltas.

3.2.2. Modelos de faltas nas tecnologias *in-circuit* e funcional

A utilização de técnicas de teste funcional, que precederam o aparecimento das técnicas *in-circuit*, recorre normalmente a um modelo em que o universo de faltas adoptado se restringe à consideração de faltas do tipo *stuck-at* (abreviadamente, s@) em todos os pinos funcionais de cada componente (Hansen, 91, p. 393). Este facto está relacionado com a boa cobertura de defeitos que se constatou ser possível atingir com este modelo de faltas (Parker, 87, p. 94), mas também com as dificuldades que estão associadas à utilização (geração de vectores, simulação) de modelos em que se considerem igualmente faltas de outros tipos. A simulação de faltas é usada neste domínio como auxílio para a geração dos vectores de teste, e para avaliar a cobertura de faltas resultante. Esta cobertura de faltas pode ser expandida para permitir a consideração de curto-circuitos, mas com restrições no universo de defeitos considerados (por exemplo, considerando apenas curto-circuitos entre

pinos adjacentes).

O aparecimento da tecnologia de teste *in-circuit*, que surgiu em consequência das dificuldades enfrentadas pela geração do programa de teste funcional para CCIs com componentes cada vez mais complexos (e eventualmente sem modelos de simulação disponíveis), introduziu modificações profundas nesta situação. A possibilidade de se dispor de um acesso físico directo a qualquer nó interior da CCI veio dispensar a simulação de faltas como etapa fundamental na geração dos vectores de teste, e a possibilidade de se reutilizarem os mesmos vectores empregues no teste de produção dos componentes contribuiu por sua vez para relegar para segundo plano a importância de um modelo que se restringia à consideração de faltas do tipo s@ em cada pino funcional dos componentes.

O acesso directo a cada nó permite por sua vez que a realização de testes destinados à detecção de curto-circuitos entre ligações, e de quebras de continuidade, tenha lugar *antes* que a alimentação da CCI seja ligada. A utilização posterior dos vectores disponíveis para o teste individual dos componentes tornou possível a obtenção de coberturas de defeitos na proximidade dos 100 % (Schlotzhauer et al., 87).

A geração de vectores para a detecção de curto-circuitos, no teste *in-circuit*, baseia-se na aplicação sucessiva de uma tensão reduzida (décimas de Volt) a cada ligação, enquanto todas as restantes ligações são mantidas ao mesmo potencial. Para além de evitar correntes de circulação entre saídas em curto-circuito (a alimentação está desligada), este procedimento tem a vantagem adicional de dispensar a definição de um modelo lógico para os defeitos deste tipo, e que seria necessariamente particular a cada tecnologia (família lógica). Esta vantagem é extensiva às quebras de continuidade nas ligações, cujo modelo seria igualmente dependente da família lógica empregue, e provavelmente ainda menos determinístico.

Os problemas de acesso que nos anos recentes têm vindo a comprometer a aplicabilidade da tecnologia *in-circuit* conduziram por sua vez ao aparecimento de equipamentos de teste híbridos (*combinational ATEs*), em que são empregues técnicas de ambos os tipos (funcional e *in-circuit*). Estes equipamentos recorrem ao teste funcional para os grupos de componentes sem acesso directo aos nós interiores, pelo que coexistem os modelos de faltas descritos para os dois casos.

3.2.3. Descrição do modelo de faltas adoptado

A definição de um modelo de faltas para as ligações de CCIs com BST implica a consideração de factores semelhantes aos que dizem respeito aos equipamentos de teste

híbridos referidos em último lugar, uma vez que a infraestrutura BST da CCI apresenta uma forte analogia com a matriz-de-agulhas que constitui a infraestrutura de acesso aos nós de teste, na tecnologia *in-circuit*. O teste das ligações BST será realizado em colaboração entre a infraestrutura BST e o equipamento de teste exterior responsável pelo acesso aos pinos de E/S primária da CCI, estando disponíveis várias alternativas para a realização do teste aos grupos de componentes não BST (Hansen, 90). Estas alternativas diferem essencialmente no grau de acesso aos nós pertencentes a estes grupos, merecendo especial destaque a técnica designada por *teste virtual* (Hansen, 90, pp. 92 a 94), em que o teste dos grupos de componentes não BST é integralmente realizado através da infraestrutura BST, e dos pinos de E/S primária que a eles estejam ligados. Esta situação representa um paralelo perfeito com as técnicas de teste híbridas anteriormente descritas, em que a infraestrutura de teste *in-circuit* (matriz-de-agulhas) é substituída pela infraestrutura BST.

A realização do teste através da infraestrutura BST influencia directamente a definição de um modelo para as faltas nas ligações, uma vez que as etapas para a detecção de curto-circuitos, e de quebras de continuidade, terão agora que ser realizadas com a alimentação ligada. Este facto tem por consequência a confluência das condições que determinaram os modelos de faltas empregues nas tecnologias funcional, e *in-circuit*. Com efeito, a consideração de um modelo de faltas mais completo, que é viabilizado pelo acesso que a infraestrutura BST possibilita aos nós interiores, surge em simultâneo com a necessidade de se utilizar uma vez mais a simulação de faltas relativamente aos grupos de componentes não BST. Isto significa que a geração de vectores para o teste (virtual) destes grupos voltará a ser feita através das técnicas comuns ao teste funcional, recorrendo a um modelo em que se consideram apenas faltas do tipo s@ em cada pino dos respectivos componentes. Os vectores assim gerados serão acrescentados aos que disserem especificamente respeito ao teste das ligações BST.

A geração de vectores para o teste das ligações BST apresenta por sua vez diferenças substanciais, relativamente à tecnologia *in-circuit*. A primeira razão para estas diferenças consiste no facto de a aplicação de vectores e captura de respostas ser feita através dos andares de saída, e de entrada, dos componentes com BST, pelo que o comportamento em situação de defeito é naturalmente dependente da tecnologia usada (Jong et al., 91a).

A detecção e o diagnóstico de curto-circuitos constituem o primeiro exemplo de uma situação em que a tecnologia empregue constitui o factor principal que condiciona o comportamento em caso de defeito. O procedimento genérico para a detecção de curto-circuitos entre duas ligações consiste em forçar (através dos respectivos nós CL) valores lógicos complementares a estas ligações, e analisar as respostas capturadas nos correspondentes nós OL.

Nos casos em que as saídas em curto-circuito apresentarem menores impedâncias para o estado lógico **0**, este será o valor lógico dominante, que será capturado nos nós OL das ligações envolvidas. Os andares de saída do tipo *totem-pole*, na família lógica TTL, constituem um exemplo concreto desta situação. Deve no entanto referir-se que, mesmo para os casos em que exista um valor lógico dominante, não é garantido que as respostas capturadas pelos nós OL de um conjunto de ligações em curto-circuito coincidam com este valor dominante. Este facto é consequência de o número, e os estados lógicos, das saídas em curto-circuito, constituírem igualmente um factor importante.

O resultado de um curto-circuito entre ligações corresponderá frequentemente a um valor analógico (superior a V_{IL}†, e inferior a V_{IH}††), que será medido de forma digital (através de operações de captura nas células BST). Este valor analógico é imposto por um divisor resistivo (não linear), em que a impedância para a alimentação corresponde ao paralelo das impedâncias oferecidas pelas saídas em **1**, e a impedância para a massa corresponde ao paralelo das impedâncias oferecidas pelas saídas em **0**. O resultado desta situação está ilustrado na figura 3.4, onde se representam, para algumas das tecnologias mais comuns, um conjunto de curvas que permitem determinar o valor de tensão resultante para curto-circuitos envolvendo um número variado de saídas.

As curvas apresentadas na figura 3.4 foram obtidas experimentalmente, por traçado da característica tensão-corrente de cada saída, para cada nível lógico. O valor de tensão para cada situação de curto-circuito (V_d, tensão de defeito) corresponde à abcissa da intercessão entre as curvas que representam o número de saídas em curto-circuito, para cada nível lógico.

O caso apresentado na figura 3.4.(d) é de particular interesse, por corresponder a medidas efectuadas com componentes BST que estão disponíveis comercialmente. Este caso permitiu confirmar que o nível lógico capturado nas células de entrada de ligações em curto-circuito tanto pode ser **0** como **1**, dependendo do nível lógico que tiver sido aplicado à maior parte das ligações envolvidas. É no entanto razoável admitir que o valor lógico capturado será o mesmo em todas as ligações curto-circuitadas entre si, desde que a tecnologia empregue, e os níveis de tensão resultantes, sejam também coincidentes em todas.

---

† Valor máximo da tensão presente numa entrada digital, para o qual se garante a equivalência ao nível lógico **0**.

†† Valor mínimo da tensão presente numa entrada digital, para o qual se garante a equivalência ao nível lógico **1**.

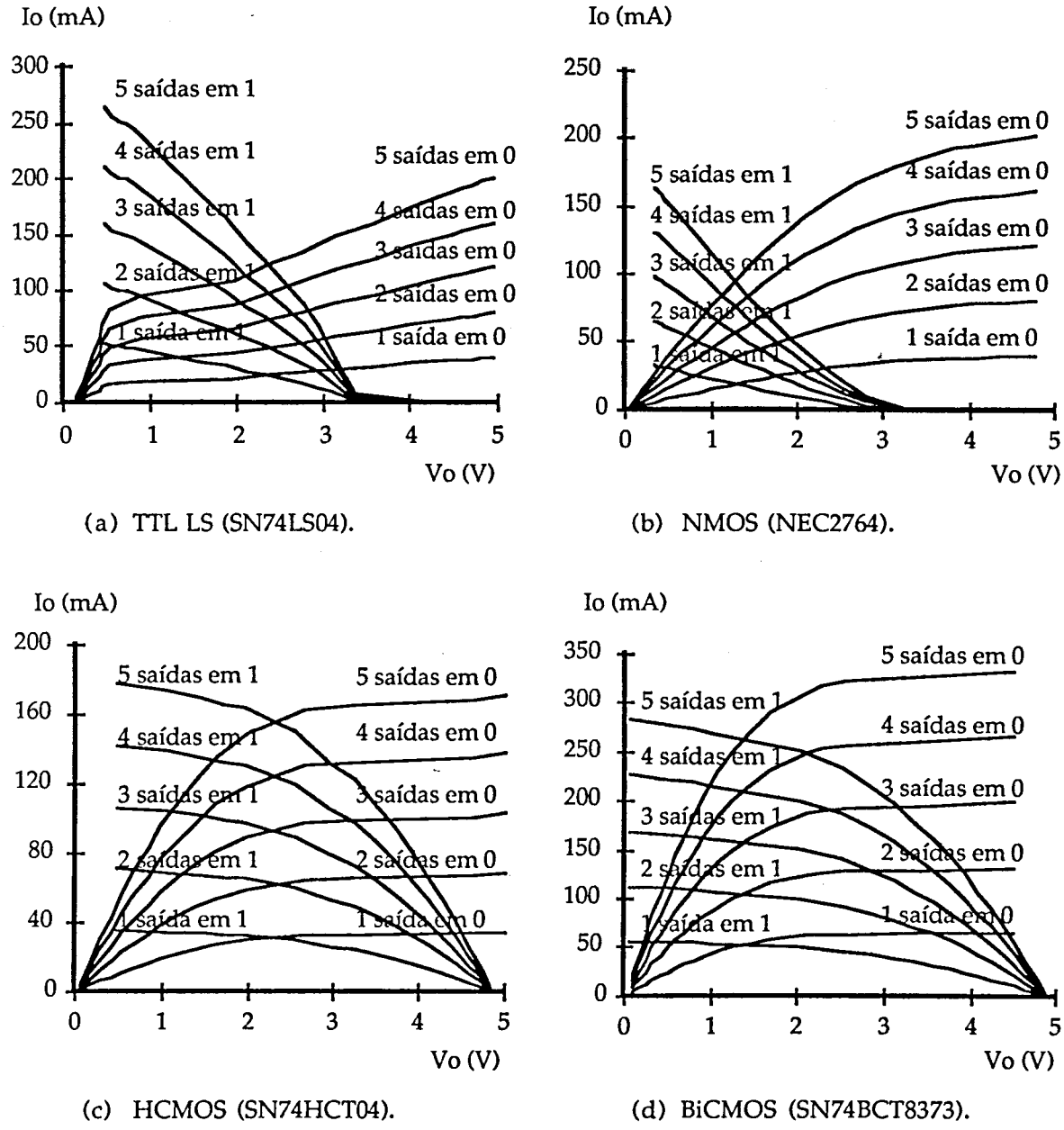


Fig.3.4: Tensão de defeito (V_d) como função do número e estado das saídas em curto-circuito.

Também a ocorrência de quebras de continuidade pode ou não corresponder a situações em que o resultado capturado seja previsível. A família lógica TTL corresponde uma vez mais a um exemplo em que se pode definir um modelo previsível para o comportamento na presença de defeitos, uma vez que uma entrada deixada no ar tende a polarizar os andares a jusante da mesma forma que se a esta entrada tivesse sido aplicado o nível lógico 1. Apesar de esta situação poder também ocorrer com outras tecnologias (experiências conduzidas com componentes BiCMOS com BST evidenciaram o mesmo

comportamento), também se pode dar o caso de o valor a capturar não ser previsível (em CMOS, por exemplo).

As considerações anteriores, relativas ao comportamento de entradas isoladas em resultado de quebras de continuidade, sugerem que não é razoável admitir para este tipo de defeitos um modelo em que se assuma a captura de um valor lógico fixo. O processo mais seguro para a respectiva detecção consistirá deste modo em garantir a aplicação dos dois níveis lógicos, a partir de cada nó CL presente na ligação. Este procedimento torna redundante a consideração explícita de faltas do tipo s@, uma vez que os vectores de teste necessários para este tipo de faltas terão nestas circunstâncias sido já gerados para as faltas de continuidade.

O modelo de faltas adoptado neste trabalho, relativamente às ligações BST, incluirá deste modo os dois tipos de faltas descritos a seguir:

- Faltas do tipo curto-circuito. A estas faltas é atribuído um comportamento determinístico, embora se assuma não ser possível conhecer com antecedência qual o valor capturado, que se considera dependente da tecnologia empregue, e do número, e estado lógico, das saídas em curto-circuito.
- Faltas de continuidade. A estas faltas é igualmente atribuído um comportamento determinístico, embora não se considere conhecido qual o valor lógico que será capturado.

Apesar de em certos casos ser possível uma previsibilidade melhor do que a que corresponde a este modelo de faltas, a geração de vectores de teste baseada nestes pressupostos conduzirá a resultados mais robustos, e com uma aplicabilidade mais ampla.

Recorde-se que o modelo de faltas descrito diz respeito às ligações BST, e não é extensível às ligações pertencentes a grupos de componentes não BST. Aliás, e uma vez que a geração dos vectores destinados ao teste destes grupos será feita por recurso às técnicas de teste funcional, o modelo de faltas a considerar será naturalmente ditado pela ferramenta usada para este fim.

Esta situação é já diferente relativamente ao conjunto de ligações que conduzem os sinais da própria infraestrutura BST da CCI (TDI, TDO, TMS, TCK, e /TRST), e relativamente às quais a quase totalidade das considerações estabelecidas para as ligações BST encontra aplicabilidade. A excepção consiste no comportamento das entradas TDI, TMS, e /TRST, quando existirem interrupções de continuidade nas respectivas ligações. Com efeito, e de acordo com as regras estabelecidas em (IEEE Std 1149.1, 90, pp. 3-2 a 3-5), estas entradas deverão nestas circunstâncias comportar-se da mesma forma que se lhes tivesse sido aplicado o valor lógico 1. Apesar de o universo de faltas ser muito semelhante

nos dois casos, as características específicas de C&O para as ligações da infraestrutura BST fazem com que os procedimentos de geração e aplicação de vectores de teste a elas destinados sejam substancialmente diferentes. Estas diferenças tornar-se-ão claras com a apresentação do protocolo de teste para CCIs com BST, que terá agora lugar.

3.3. PROTOCOLO DE TESTE

O protocolo de teste que será apresentado nesta secção consiste em três etapas principais, correspondentes ao teste da infraestrutura BST da CCI, ao teste das ligações, e ao teste dos componentes BST presentes. O teste das ligações inclui duas fases distintas, destinadas às ligações BST, e às ligações pertencentes a grupos de componentes não BST. Cada uma destas etapas será agora descrita em maior detalhe.

3.3.1. Teste da infraestrutura BST da CCI

O teste da infraestrutura BST da CCI tem por objectivo principal verificar a funcionalidade do conjunto de ligações que conduzem os sinais de teste (TDI, TDO, TMS, TCK, /TRST), e detectar as faltas estruturais nos componentes (componentes trocados, componentes mal inseridos, etc.) que disponham de registo de identificação (Tulloss et al., 89, 90a, 90b), (Heyden, 90), (Jong et al., 91b). Esta etapa assume que os componentes montados na CCI se encontram funcionais.

O primeiro passo consiste numa operação de inicialização, em resultado da qual todos os componentes BST deverão permanecer no estado *Test Logic Reset* (ver figura 2.7). Este estado constitui o ponto de partida para as restantes acções que terão lugar nesta etapa, e será atingido (a partir de qualquer estado) em resultado da aplicação de cinco impulsos no relógio de teste (TCK), enquanto a linha TMS é mantida no nível lógico 1.

O passo que terá lugar a seguir permite detectar a quase totalidade das faltas na infraestrutura BST da CCI, e consiste em promover um deslocamento através dos registos de instrução. A elevada cobertura de faltas proporcionada por esta etapa é garantida pelo facto de existir uma disposição na norma BST (IEEE Std 1149.1, 90, p. 6-1) que força o registo de instrução em cada componente a capturar os valores lógicos 01 nos seus dois bits menos significativos (o 1 no bit menos significativo, que ocupa a posição mais próxima de TDO). Esta situação está ilustrada na figura 3.5, e garante a comutação dos níveis lógicos em todas as ligações TDO-TDI.

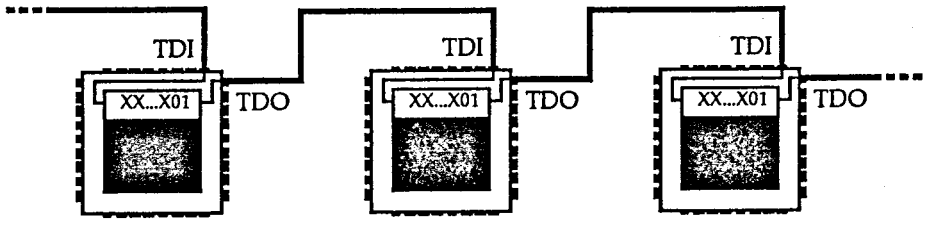


Fig.3.5: Teste da infraestrutura BST da CCI: deslocamento através dos registos de instrução.

Os dois primeiros bits deslocados para a entrada TDI do primeiro componente devem igualmente conter a sequência **01**, de modo a que também esta primeira ligação seja testada. Uma vez que estes dois primeiros bits devem igualmente ser deslocados para o exterior e comparados, o número total de impulsos necessários no relógio de teste (TCK) passará a ser dado por (S_{RI}+2), em que S_{RI} representa o somatório dos comprimentos dos registos de instrução na cadeia BST. Repare-se que este procedimento garante uma total cobertura das faltas de continuidade associadas às ligações TDO-TDI, e às que conduzem os sinais TCK e TMS, já que uma falta deste tipo em qualquer destas ligações impediria a execução com sucesso deste passo. A detecção de faltas de continuidade na ligação que conduz o sinal /TRST terá ainda que ser considerada, uma vez que esta ligação não foi até agora exercitada.

O procedimento descrito detectará a maior parte dos curto-circuitos que envolvam as ligações da infraestrutura BST, mas não dispõe de uma completa capacidade de detecção. A título de exemplo, se todos os componentes apresentassem registos de instrução com o mesmo comprimento, e com o mesmo conteúdo em resultado de uma operação de captura, então um curto-circuito entre os pinos TDI e TDO de um componente poderia não ser detectado. Em termos gerais, a detecção de faltas deste tipo terá sempre lugar se se garantir que em todos os componentes haverá pelo menos um ciclo de relógio em que o valor presente em TDI é diferente do valor presente em TDO. Uma solução referida em (Jong et al., 91, p. 109) consiste em forçar o deslocamento de uma sequência adicional (para além da sequência **01** já referida antes), diferente do conteúdo de qualquer registo de instrução após uma operação de captura, e cujo comprimento deverá ser igual ao do maior registo de instrução presente. Uma sequência com este comprimento, e em que todos os bits tenham o mesmo valor lógico (**0** ou **1**) satisfaz estes requisitos. O número total de ciclos de relógio necessários passa nestas circunstâncias a ser dado por (S_{RI}+M_{RI}+2), em que M_{RI} representa o comprimento do maior registo de instrução presente.

A detecção de faltas de continuidade na ligação que conduz o sinal /TRST, para os componentes que recorrerem a esta possibilidade, pode também ser garantida, desde que a aplicação de um impulso descendente nesta linha seja precedida por uma sequência de

transição de estados que permita posteriormente verificar quais destes componentes é que não transitaram para o estado *Test Logic Reset* (Jong et al., 91b, pp. 110 e 111).

Repare-se que os procedimentos anteriores garantiram já a detecção de parte das faltas estruturais referidas para os componentes, uma vez que os componentes BST mal inseridos provocariam uma quebra de continuidade que impediria o sucesso desta etapa. No entanto, não está garantida a detecção de componentes trocados, mas em que a parte da lógica BST usada apresente as mesmas características (a mesma localização de pinos, e o mesmo comportamento). A detecção destas situações pode ser garantida se os componentes dispuserem de um registo de identificação, pelo que o teste da infraestrutura BST da CCI deve incluir um passo dedicado à verificação dos códigos de identificação disponíveis. Este passo deve ser estendido à verificação dos códigos de identificação adicionais (IEEE Std 1149.1, 90, p. 7-24), quando existentes.

A sequência de operações descrita é independente de existir ou não mais do que uma cadeia BST na CCI a testar. Para os casos de CCIs em que estejam presentes múltiplas cadeias BST, esta sequência de operações deve ter lugar de forma independente em cada uma das cadeias existentes.

3.3.2. Teste das ligações

Após ter sido verificada a operacionalidade da infraestrutura BST, torna-se possível passar ao teste das ligações, ou dos componentes. Uma vez que a existência de saídas em curto-circuito pode conduzir ao dano permanente de alguns componentes, o teste de ligações deve ser o primeiro a ter lugar. Esta etapa tanto diz respeito ao teste das ligações BST, como das ligações pertencentes a grupos de componentes não BST, pelo que ambas as fases serão apresentadas.

Constituindo um dos objectivos principais que presidiram ao desenvolvimento da tecnologia BST, o teste de ligações tem sido alvo de um grande número de publicações, com ênfase no estudo de algoritmos para a detecção e diagnóstico deste tipo de faltas. Uma abordagem minimamente completa deste domínio tornaria demasiado extensa esta secção, pelo que se optou por efectuar uma apresentação restrita à consideração dos aspectos mais importantes a referir. A análise pormenorizada das questões relativas ao teste de ligações BST, e a apresentação dos algoritmos destinados à detecção e diagnóstico de curto-circuitos, serão por sua vez apresentados no capítulo seguinte.

3.3.2.1. Ligações BST

De acordo com o modelo de faltas que foi apresentado para as ligações BST, o conjunto de vectores aplicados nesta etapa tem por objectivo detectar faltas de continuidade, e curto-circuitos. A aplicação destes vectores será feita através da infraestrutura BST (cuja lógica se encontrará em modo de teste externo — instrução *Extest*), em colaboração com os recursos de teste exteriores responsáveis pelo acesso aos pinos de E/S primária da CCI.

A detecção de faltas de continuidade obriga a verificar todos os percursos existentes em cada ligação, o que pode ser efectuado activando cada nó CL em sequência. Uma vez que o modelo de faltas adoptado não permite prever qual o valor a capturar nos nós OL, cada nó CL deverá aplicar os dois níveis lógicos à ligação, pelo que o número de vectores necessários para efectuar o teste de continuidade numa ligação corresponde ao dobro do número de nós CL presentes. Uma vez que todas as ligações podem ser testadas em paralelo, o número de vectores necessários para a detecção de faltas de continuidade será dado pelo dobro do maior número de nós CL existentes numa única ligação.

A detecção de curto-circuitos é mais complexa, sobretudo devido ao número muito elevado de combinações de defeito que podem ter lugar. O número de curto-circuitos diferentes envolvendo **i** ligações, pertencentes a um conjunto com um total de N ligações, é dado por

$$\binom{N}{i} = \frac{N!}{i!\cdot(N-i)!} \tag{3.1}$$

No total, o número de curto-circuitos possíveis, entre qualquer número de ligações, é dado por

$$\sum_{i=2}^{N} \binom{N}{i} = 2^N - (N+1) \tag{3.2}$$

Esta evolução exponencial permite desde logo antever alguma dificuldade, sobretudo para fins de diagnóstico. Mesmo considerando apenas curto-circuitos entre duas ligações, o número de combinações possíveis é dado por

$$\frac{N\cdot(N-1)}{2} \tag{3.3}$$

Atendendo a que uma CCI pode facilmente conter algumas centenas de ligações, o número de combinações dado pela equação (3.3) pode atingir as centenas de milhar. O recurso a considerar apenas os curto-circuitos entre ligações adjacentes simplificaria significativamente as tarefas de diagnóstico, embora se tornasse então necessário definir um critério de adjacência.

O problema torna-se bastante mais simples se o objectivo do teste se restringir à etapa de detecção, uma vez que bastará então garantir que, qualquer que seja o par de ligações considerado, haverá pelo menos um vector de teste que aplica a estas ligações valores lógicos complementares. Repare-se que isto é equivalente a requerer que a sequência de valores lógicos aplicados para o teste de curto-circuitos seja única para cada ligação, i.e. não existam duas ligações a que seja aplicada a mesma sequência. Qualquer algoritmo de geração de vectores que satisfaça este requisito poderá ser empregue, e o número mínimo de vectores necessários para realizar o teste de um conjunto com N ligações será então dado por (Kautz, 74)

$$\lceil \log_2 (N) \rceil \text{ †} \qquad (3.4)$$

Contrariamente ao que acontece com a etapa correspondente ao teste da infraestrutura BST, a realização do teste de ligações em CCIs que contenham mais do que uma cadeia BST não pode ter lugar de forma independente para cada cadeia. Este facto deve-se à existência de ligações com pinos BST em mais do que uma cadeia, que devem estar sincronizadas para permitir a aplicação do conjunto de vectores gerados para aquele fim (Lien et al., 91b).

Por outro lado, e de acordo com o que foi já referido no capítulo 2, a aplicação de cada vector através das cadeias BST existentes tem que ser realizada em sincronismo com os canais de teste exteriores, associados aos nós com acesso paralelo. O teste das ligações BST terá assim lugar por execução cíclica da sequência de passos descrita no quadro 3.2.

i)	Deslocar para o interior de cada cadeia BST o novo vector a aplicar. Esta operação de deslocamento transporta para o exterior as respostas capturadas para o vector anterior.
ii)	Aplicar às ligações os valores deslocados para o interior das cadeias BST, e os valores presentes nos canais de teste associados aos nós com acesso paralelo.
iii)	Capturar as respostas em todas as cadeias BST, e nos canais de teste associados aos nós com acesso paralelo.

Quadro 3.2: Sequência de acções para o teste de ligações em CCIs com múltiplas cadeias BST.

† $\lceil x \rceil$ representa o menor inteiro não inferior a x (tecto de x).

3.3.2.2. Ligações pertencentes a grupos de componentes não BST

O teste das ligações pertencentes aos grupos de componentes não BST pode ter lugar de acordo com mais do que uma estratégia, conforme o grau de acesso que for possível às ligações pertencentes a estes grupos (Hansen, 90).

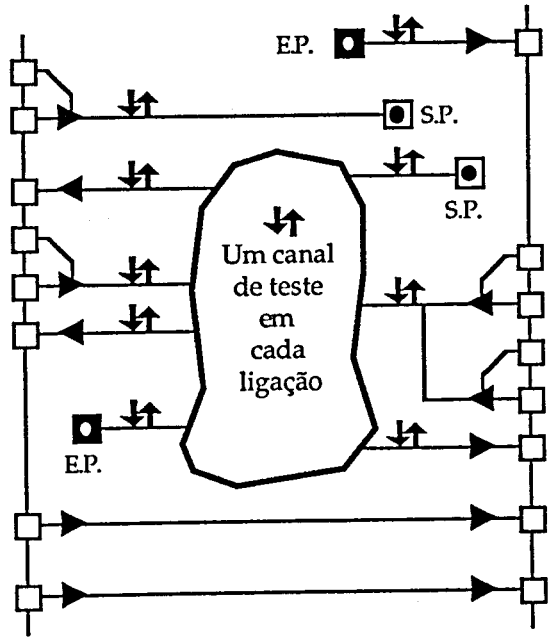


Fig.3.6: Teste de grupos de componentes não BST, com acesso a todas as suas ligações.

A figura 3.6 ilustra a possibilidade de se dispor de acesso directo a todas as ligações pertencentes ao grupo de componentes não BST (o símbolo ↓↑ é usado para representar os canais de teste exteriores). Esta situação corresponde a efectuar-se o teste destes grupos de acordo com técnicas *in-circuit*, e não requer qualquer contribuição da infraestrutura BST para a aplicação de estímulos ou captura de respostas. Nestas circunstâncias, a interacção entre a infraestrutura BST, e os canais de teste exteriores, tal como se descreve no quadro 3.2, não terá aqui lugar.

Esta alternativa tem interesse para as situações em que o acesso *in-circuit* não constitua um problema, caso em que a rapidez (a aplicação de estímulos e captura de respostas são efectuadas em paralelo em todos os nós), e o facto de não requerer a existência de modelos de simulação para os componentes, a podem tornar recomendável.

Uma outra alternativa possível consiste na situação ilustrada na figura 3.7, em que o acesso directo ao grupo não BST se restringe às ligações situadas na sua periferia. Esta estratégia corresponde à que foi descrita quando se referiram os equipamentos de teste

híbridos (*combinational ATEs*), em que o acesso *in-circuit* à periferia do grupo permite a realização de um teste do tipo funcional. Também para este caso o teste dos grupos não BST será realizado exclusivamente através da infraestrutura *in-circuit*, pelo que a interacção descrita no quadro 3.2 não será aplicável.

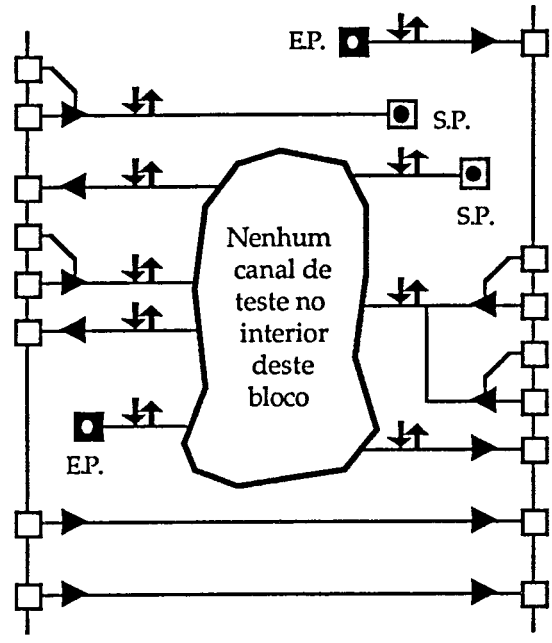


Fig.3.7: Teste de grupos de componentes não BST, com acesso restrito às suas ligações periféricas.

Repare-se que esta alternativa tem sobretudo interesse quando existirem restrições para o acesso *in-circuit*, mas continue a pretender-se a maior rapidez proporcionada pelo acesso paralelo às entradas e saídas do grupo de componentes a testar. Os modelos de simulação para cada componente do grupo devem neste caso estar disponíveis, para permitir a geração dos vectores de teste.

Ambas as alternativas descritas anteriormente requerem a possibilidade de acesso *in-circuit* ao grupo de componentes a testar. Atendendo às dificuldades crescentes que este acesso tem vindo a experimentar, o teste virtual dos grupos de componentes não BST constituirá uma estratégia com grande interesse, sobretudo para os casos em que a respectiva complexidade for reduzida. A realização do teste terá neste caso lugar através da infraestrutura BST, tal como se ilustra na figura 3.8, sendo o acesso exterior limitado aos pinos de E/S primária da CCI.

Esta alternativa requer a presença de canais de teste apenas nas entradas e saídas primárias do grupo de componentes a testar, mas a rapidez do teste será necessariamente inferior, já que se torna agora necessário efectuar uma operação de deslocamento através da

infraestrutura BST, por cada vector a aplicar. A existência de modelos de simulação para os componentes de cada grupo será também necessária, para permitir a geração dos respectivos vectores de teste.

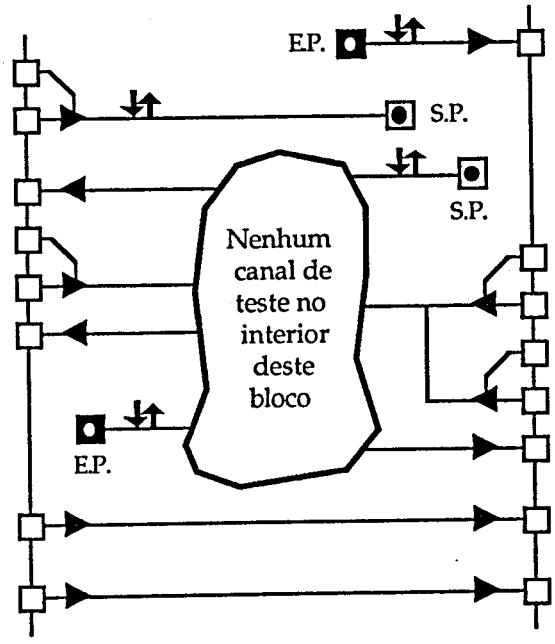


Fig.3.8: Teste de grupos de componentes não BST, por acesso virtual.

A infraestrutura BST será neste caso usada em modo de teste externo (instrução *Extest*), tal como sucedeu relativamente ao teste das ligações BST. Nestas circunstâncias, a realização do teste terá lugar exactamente de acordo com a sequência de acções descritas no quadro 3.2, e a distinção entre as ligações BST, e as ligações pertencentes a grupos de componentes não BST, corresponderá apenas à diferente proveniência dos conjuntos de vectores empregues.

Refira-se ainda que, por razões de ordem prática, o teste virtual de grupos de componentes não BST está limitado a situações com complexidade e dimensão reduzidas. Estas limitações estão essencialmente relacionadas com os seguintes factores:

- Tempo de execução: a aplicação de M vectores de teste, através de uma infraestrutura BST com N células de comprimento, requer cerca de N·M ciclos de relógio.
- Requisitos de armazenamento: o armazenamento de vectores de teste, já serializados, dos respectivos valores esperados, e máscaras de comparação, pode exigir capacidades de armazenamento que atinjam vários *Mbytes* (Hansen, 89).

O segundo destes factores é particularmente importante para CCIs que disponham de controladores residentes, já que os vectores a empregar para o teste de grupos de componentes não BST terão que estar armazenados na memória com o programa de teste. A título de exemplo, um conjunto de 4.000 vectores destinados ao teste de um grupo de componentes não BST, inserido numa cadeia com apenas 100 células de comprimento, requer um espaço de armazenamento correspondente a cerca de 150 *Kbytes*.

Os problemas referidos poderão ser ultrapassados através do emprego de componentes que permitam a geração e aplicação local (na periferia do grupo de componentes não BST) dos vectores de teste, seja por geração pseudo-aleatória de vectores e análise de assinatura, seja por aplicação de vectores gerados de acordo com algoritmos pré-definidos, aplicáveis a determinados tipos de componentes. Para além de dispensar o armazenamento dos vectores a empregar no teste dos grupos de componentes não BST, esta estratégia permite ainda uma grande redução no tempo de execução do teste, uma vez que é necessário apenas um ciclo de relógio por cada vector.

Como exemplos do primeiro caso, podem referir-se *latches* e *buffers* já disponíveis comercialmente, e cuja infraestrutura BST possibilita modos de operação adicionais, destinados a permitir a geração pseudo-aleatória de vectores, e a análise de assinatura (TI SCL162, 90). Outros componentes BST, com modos de funcionamento mais sofisticados, foram já também anunciados (Siemens, 90a), (TI DBM, 90), (Ballew et al., 92).

O teste de blocos de memória de leitura e escrita (RAM) constitui um exemplo do segundo caso, sendo descritos em (Kritter et al., 91), (Raghavachari, 91) componentes BST que realizam este teste de acordo com algoritmos pré-definidos. Também em (Bhavsar, 91a), (Rijk et al., 91) se apresentam métodos alternativos para o teste deste tipo de memórias, quando não estiverem disponíveis recursos como os que foram anteriormente referidos.

3.3.3. Teste dos componentes BST

A última etapa do protocolo descrito para o teste de CCIs com BST corresponde ao teste de componentes, e é essencialmente destinada aos componentes BST que disponham de auto-teste incorporado (BIST, *Built-In Self-Test*). A execução das funções de auto-teste terá início pelo envio da instrução correspondente (*Runbist*) para os componentes que disponham desta facilidade, devendo os restantes componentes ser colocados em modo de *bypass* (instrução *Bypass*). A duração deste passo será imposta pelo componente que requerer o maior número de impulsos no relógio de teste (TCK).

Os componentes BST sem auto-teste poderão também ser testados nesta etapa, mas voltam a estar presentes nestas circunstâncias as limitações já referidas para o teste virtual de grupos de componentes não BST (tempo de execução, e requisitos de armazenamento), uma vez que a realização deste tipo de teste terá igualmente lugar por deslocamentos sucessivos de estímulos e respostas. A lógica BST estará nestas circunstâncias a funcionar em modo de teste interno (instrução *Intest*). Quando a dimensão ou a complexidade do componente inviabilizarem a realização de um teste completo por este processo, o modo de teste interno poderá ainda assim ter interesse para permitir a aplicação de um pequeno conjunto de vectores, com o objectivo principal de verificar que o componente se encontra activo, e responde da forma esperada. Os componentes em que não for pretendida a aplicação de vectores por este processo deverão encontrar-se em modo de *bypass*.

A sequência de operações necessária para realizar o teste aos componentes BST é independente de existir ou não mais do que uma cadeia BST na CCI. Para as CCIs que disponham de múltiplas cadeias BST, esta etapa terá lugar de forma independente em cada uma delas.

3.4. AVALIAÇÃO DA COBERTURA DE FALTAS

A avaliação da cobertura de faltas diz respeito principalmente às faltas estruturais presentes nas ligações da CCI, e deve levar em consideração o facto de a geração dos vectores de teste para a infraestrutura BST da CCI, para as ligações BST, e para as ligações pertencentes a grupos de componentes não BST, ter lugar de forma independente. A título de exemplo, e se existir um curto-circuito entre uma ligação TDO-TDI, e uma ligação BST que se mantenha num valor lógico não dominante (ou em alta impedância) durante a realização da etapa correspondente ao teste da infraestrutura BST, esta falta será provavelmente apenas detectada na etapa seguinte, quando se realizarem os primeiros testes às ligações. O teste de grupos de componentes não BST constitui outro exemplo, uma vez que aos vectores de teste disponíveis para este fim (gerados externamente) estará associada uma cobertura de faltas que não leva em conta a possível existência de curto-circuitos entre ligações BST, e ligações interiores a estes grupos. Em termos gerais, o problema que se considera nesta secção consiste na necessidade de se obter uma estimativa global para a cobertura de faltas atingida pelo protocolo de teste descrito, a partir das coberturas de faltas conhecidas para cada uma das suas etapas.

Este problema não dispõe ainda de uma solução satisfatória, embora seja reconhecido por vários autores (Robinson et al., 90), (Hansen, 91), (Lefebvre, 91). Uma aproximação sugerida em (Hansen, 91, p. 396), consiste em determinar apenas a cobertura de faltas

relativamente a um universo mais restrito, tais como curto-circuitos entre pinos adjacentes, e faltas do tipo s@ em cada pino funcional de todos os componentes. Esta lista de faltas, e o conjunto completo de vectores de teste gerados, seriam então processados por um simulador de faltas, que produziria uma estimativa para a cobertura de faltas global.

Refira-se ainda que uma medida possível para aumentar esta cobertura de faltas consiste em manter activas, e com um mesmo valor lógico aplicado, todas as ligações BST da CCI, durante o teste das ligações pertencentes a grupos de componentes não BST. Em particular, e se a tecnologia empregue tiver um valor lógico não dominante, este deverá ser o valor escolhido para aplicar às ligações BST. O interesse deste procedimento consiste em que um curto-circuito que envolva uma ligação BST, e uma ligação pertencente a um grupo de componentes não BST, será provavelmente detectado nos nós OL da ligação BST. Este pressuposto baseia-se no facto de os vectores usados para o teste dos grupos de componentes não BST aplicarem supostamente ambos os níveis lógicos em todos os pinos funcionais destes componentes, caso o modelo de faltas usado corresponda à consideração de faltas do tipo s@ em cada um destes pinos. Uma estratégia semelhante é adoptada em (Robinson et al., 90), numa etapa descrita como *teste de interacções*.

Capítulo 4

O teste de ligações BST na CCI

Factores que condicionam a detecção de faltas

Conflitos no comando de barramentos

Detecção e diagnóstico de faltas de continuidade

Detecção e diagnóstico de curto-circuitos

4. O TESTE DE LIGAÇÕES BST NA CCI

Este capítulo tem por assunto principal a geração de vectores para o teste de ligações BST na CCI. Os factores que condicionam a detecção de faltas são analisados, e conduzem à consideração da possível ocorrência de conflitos no comando de barramentos, em resultado da aplicação de vectores destinados ao teste das ligações BST.

A detecção e o diagnóstico de faltas de continuidade são então abordados, sendo o restante do capítulo dedicado aos algoritmos para a detecção e diagnóstico de curto-circuitos. A apresentação de cada algoritmo inclui uma descrição sucinta do procedimento de geração dos vectores de teste, um quadro que ilustra a sua aplicação, e um conjunto de comentários relativos à capacidade de diagnóstico, à complexidade, e ao número de vectores gerados. A descrição destes algoritmos conclui com a apresentação de quadros de síntese das suas características principais, e por uma breve análise comparativa. A apresentação efectuada evidencia o facto de a caracterização dos vários algoritmos se poder fazer essencialmente em termos da solução de compromisso que apresentam relativamente à capacidade de diagnóstico, à complexidade na análise das respostas, e ao número de vectores de teste.

Os assuntos tratados neste capítulo, e em particular a análise dos algoritmos destinados à detecção e diagnóstico de curto-circuitos, estão intimamente relacionados com parte do trabalho realizado no âmbito do subprojecto 4 do projecto ESPRIT 2478, e que se encontra descrito em (Jong et al., 89), (Ferreira et al., 89, 90, 91b).

A compreensão das questões principais para a detecção e diagnóstico de faltas, nas ligações BST de uma CCI, assume uma importância de primeiro plano para a exploração na totalidade das potencialidades proporcionadas pela infraestrutura BST. Já a análise e caracterização da diversidade de algoritmos destinados à detecção e diagnóstico de curto-circuitos apresenta um interesse mais académico do que prático, quer pela (pequena) proporção que o tempo necessário para o teste das ligações BST representa, face ao tempo total de teste, quer pelas dificuldades de implementação relativas a alguns destes algoritmos. Estão neste último caso os algoritmos do tipo adaptativo, em que se requer uma segunda fase de geração de vectores de teste, de acordo com os resultados obtidos para o conjunto de vectores aplicados na primeira fase, e também os algoritmos que requeiram a existência de informações adicionais relativas ao processo de fabrico, ou a questões de *layout* (relações de vizinhança entre ligações). Estas considerações tornam opcional a leitura completa da secção 4.4 (Detecção e diagnóstico de curto-circuitos),

permitindo uma leitura mais rápida por passagem imediata ao capítulo seguinte.

4.1. FACTORES QUE CONDICIONAM A DETECÇÃO DE FALTAS

Existem três factores principais que condicionam a detecção dos tipos de faltas considerados, e em resultado dos quais se definem diferentes procedimentos para a respectiva detecção e diagnóstico. Estes factores consistem no comportamento definido para cada falta, no número de possíveis combinações de defeito, e nas condições necessárias para a aplicação dos respectivos vectores de teste.

O comportamento definido para o circuito, quando na presença de alguma das faltas consideradas no modelo adoptado, constitui naturalmente o primeiro factor que estabelece condições particulares para a respectiva detecção. As implicações decorrentes deste comportamento serão analisadas em pormenor nas secções 4.3 e 4.4, em que serão apresentados os factores que condicionam as operações de detecção e diagnóstico.

O número de possíveis situações de defeito, para cada um dos tipos de faltas considerados, constitui o segundo factor que condiciona a respectiva detecção. As faltas de continuidade apresentam um número possível de situações de defeito que depende da topologia particular de cada ligação, mas que aumenta linearmente com o número de ligações na CCI. Esta variação linear entre o número de ligações, e o número de possíveis situações de defeito, não encontra porém paralelo no que respeita às faltas do tipo curto-circuito, situação em que esta dependência assume uma variação de carácter exponencial. Com efeito, e como foi já referido no capítulo anterior, o número total de curto-circuitos possíveis numa CCI com N ligações é dado por

$$2^N - (N+1)\ ^{\dagger} \qquad (4.1)$$

o que desde logo evidencia uma maior complexidade, já que as condições para a respectiva detecção têm que levar em conta que o número de possíveis situações de defeito é muito superior para este caso. As implicações decorrentes do número de possíveis situações de defeito, para cada tipo de falta, serão também analisadas em pormenor nas secções 4.3 e 4.4.

As condições para a aplicação dos vectores de teste, relativamente aos dois tipos de faltas referidos, constituem o terceiro factor a considerar, e a sua importância advém da necessidade de se efectuar uma escolha cuidadosa do conjunto de nós do tipo CL que

---

† As (N+1) combinações excluídas correspondem à inexistência de curto circuitos que envolvam apenas 1 ligação (N casos), ou 0 ligações.

permitirão a aplicação dos vectores às ligações. Esta escolha assume particular importância quando se requerer que todas as ligações estejam simultaneamente sob teste, no sentido de se evitar a possível ocorrência de conflitos no comando de barramentos.

Uma vez que uma falta de continuidade afecta apenas uma ligação, o respectivo teste poderá ser conduzido de forma independente entre ligações. Isto significa que embora a realização deste teste seja mais rápida quando todas as ligações são testadas ao mesmo tempo, a condição de simultaneidade não é de facto necessária. Em particular, a ligação que impuser o maior número possível de faltas de continuidade será a última a ver concluído este teste, podendo muitas das restantes encontrar-se já em estado de alta impedância. Esta situação não se repete no que respeita aos curto-circuitos, já que este tipo de faltas afecta *conjuntos de ligações*. Uma vez que se considera possível a ocorrência de qualquer combinação de defeito, é desejável que todas as ligações estejam simultaneamente sob teste, para este tipo de faltas.

No que respeita ao nó do tipo CL a seleccionar, as condições para aplicação dos vectores de teste para faltas de continuidade requerem que todos os nós deste tipo existentes numa ligação sejam sucessivamente seleccionados, no sentido de cobrir a totalidade dos percursos que compõem a ligação. Já para as faltas do tipo curto-circuito bastará a selecção de um destes nós, em cada ligação, para permitir a aplicação dos respectivos vectores de teste. Esta selecção apresenta apenas por condicionantes que todas as ligações tenham simultaneamente um nó CL activo, e que não ocorram conflitos no comando de barramentos.

As condições descritas para a aplicação dos vectores de teste às ligações BST estão sintetizadas no quadro 4.1, e realçam a preocupação dominante em evitar que uma ligação tenha mais do que um nó do tipo CL activo simultaneamente, i.e. em que exista mais do que um pino a forçar valores lógicos (eventualmente diferentes) na mesma ligação. Esta preocupação deve-se à possibilidade de a infraestrutura BST provocar conflitos no comando de barramentos, já que a geração e aplicação de vectores para o teste de ligações não depende da funcionalidade do circuito, o que justifica uma análise mais cuidada deste problema.

Tipo de falta	Ligações sob teste	Nó do tipo CL seleccionado
Continuidade	Uma ou mais de cada vez	Todos, um de cada vez
Curto-circuitos	Todas em-simultâneo	Um só, qualquer

Quadro 4.1: Condições para aplicação dos vectores de teste às ligações BST.

4.2. CONFLITOS NO COMANDO DE BARRAMENTOS

Pretende-se nesta secção analisar de forma pormenorizada o problema da eventual ocorrência de conflitos no comando de barramentos, e discutir soluções que o permitam ultrapassar. Este problema assume maior importância no caso de circuitos com funcionamento orientado à palavra (grupo de bits), tal como sucede com circuitos baseados em microprocessadores. A situação apresentada na figura 4.1 ilustra um caso destes, em que a passagem ao estado activo de mais do que um dos sinais de selecção provocaria um conflito no acesso ao barramento de dados comum.

Apesar de em funcionamento normal, e partindo do princípio de que não existem problemas de projecto, a não ocorrência de conflitos ser garantida pela própria funcionalidade definida para o circuito, a infraestrutura BST de um CI torna possível que se force qualquer combinação de estados nos seus pinos de saída, e em particular também aquelas que forçariam mais do que um pino activo na mesma ligação.

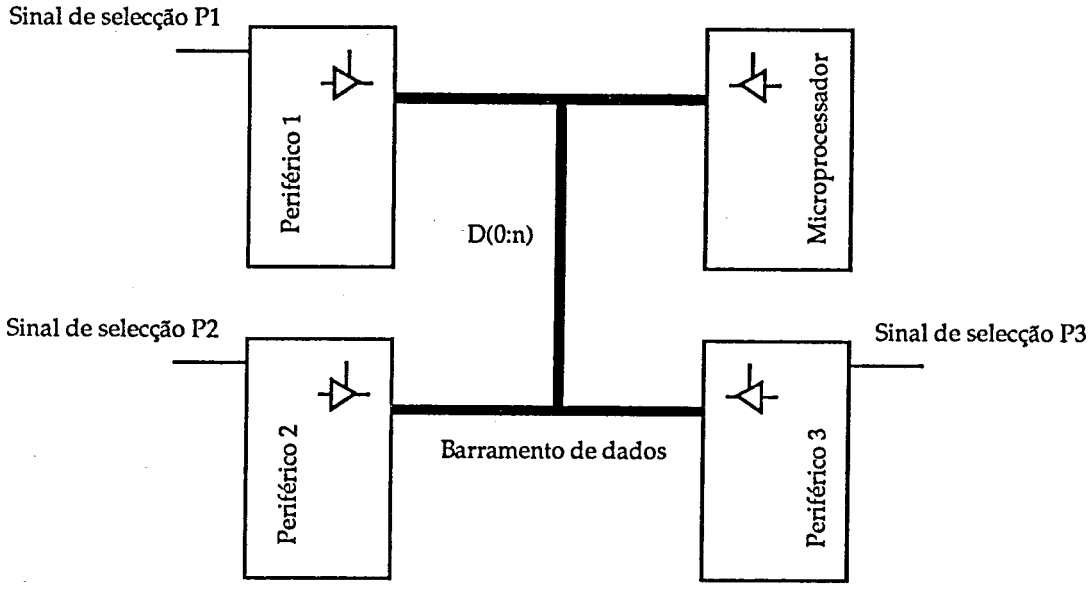


Fig.4.1: Exemplo de um barramento.

Retomando as condições para a aplicação dos vectores de teste às ligações, descritas no quadro 4.1, torna-se fácil constatar que a verificação destas condições constitui uma tarefa simples, no que respeita ao exemplo apresentado na figura 4.1, onde cada ligação do barramento será do tipo que se ilustra na figura 4.2.

Relativamente às condições para a aplicação dos vectores de teste para as faltas de continuidade, a figura 4.2 torna claro que se o teste dos percursos associados aos quatro

nós do tipo CL tiver lugar pela mesma ordem em todas as ligações, tal como se ilustra na figura 4.3, então todas elas podem ser colocadas sob teste em simultâneo, sem que ocorram conflitos. A título de exemplo, se numa das ligações se optasse por uma ordem diferente da que corresponde à sequência ilustrada nas figuras 4.3.(a) a 4.3.(d), resultariam dois nós do tipo CL activos em cada ligação, pelo que a colocação desta ligação sob teste não poderia ser simultânea com as restantes.

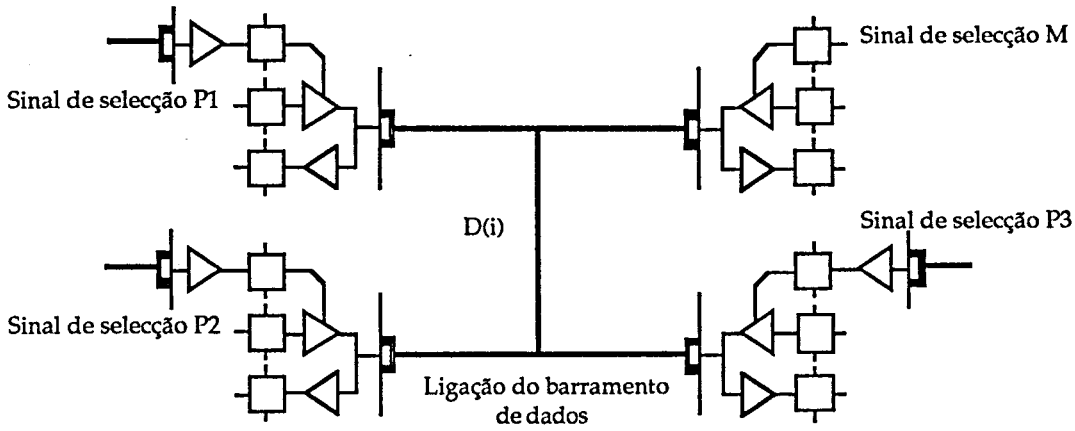


Fig.4.2: Topologia das ligações consideradas no exemplo da figura 4.1.

Apesar de existir uma solução intuitiva para o exemplo da figura 4.1, as condições para a aplicação dos vectores de teste para faltas de continuidade terão que se traduzir num conjunto de regras que integrem o procedimento de geração dos vectores de teste, e que permitam tratar o problema de uma forma geral.

Relativamente às faltas do tipo curto-circuito, e continuando com o exemplo da figura 4.1, é fácil constatar que a selecção de qualquer um dos quatro nós do tipo CL ilustrados na figura 4.2 permitirá que todas as ligações sejam colocadas sob teste em simultâneo, sem que ocorram quaisquer conflitos. Esta condição de simultaneidade é no entanto agora um requisito, de acordo com as condições descritas no quadro 4.1 para este tipo de faltas. Isto significa que a selecção de um nó do tipo CL diferente, numa dada ligação, deixa agora de ser possível, uma vez que o conflito daí decorrente não se pode resolver pelo adiamento da colocação sob teste desta ligação. Uma vez mais, as condições descritas no quadro 4.1 terão que se traduzir num conjunto de regras que integrem o procedimento de geração dos vectores de teste, neste caso para as faltas do tipo curto-circuito.

Repare-se que, relativamente às faltas de continuidade, a resolução do problema descrito é sempre possível. Esta afirmação baseia-se na possibilidade de se adiar a colocação sob teste de uma dada ligação, quando não for possível encontrar uma solução

não conflituosa, com todas as ligações simultaneamente sob teste. No entanto, já não é possível garantir a existência de uma solução relativamente à aplicação dos vectores para o teste de curto-circuitos, uma vez que deixou agora de ser possível o adiamento da colocação de uma ligação sob teste, como recurso para resolver um problema de conflito.

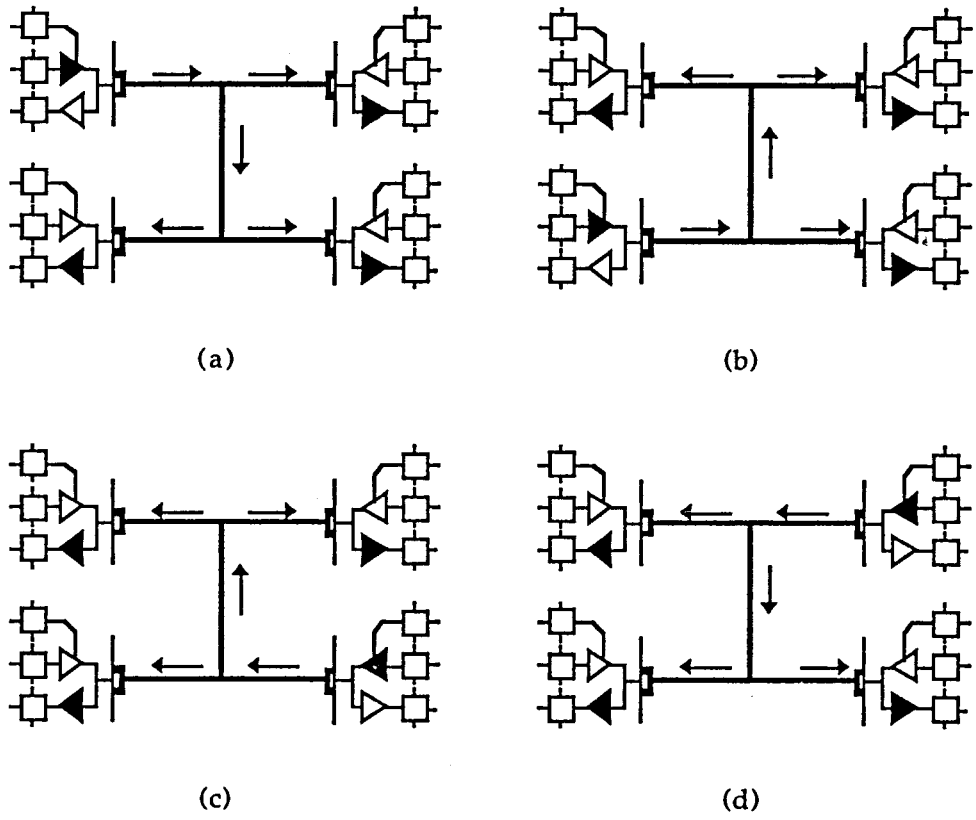


Fig.4.3: Sequência possível para a realização de testes de continuidade.

A figura 4.4 ilustra um conjunto de ligações que se encontram nas circunstâncias referidas no parágrafo anterior (o padrão de preenchimento dos andares de saída indica quais as células BST de controlo comuns). Repare-se que a colocação simultânea das ligações {2,3} sob teste faria com que existissem dois nós do tipo CL activos ao mesmo tempo nas restantes ligações, o que poderia levar à ocorrência de conflitos.

O exemplo apresentado na figura 4.4 merece dois reparos. Por um lado, é pouco provável que se encontrem circuitos correctamente projectados, e que incluam conjuntos de ligações nas circunstâncias ilustradas nesta figura. Esta afirmação decorre da observação de que o funcionamento normal de um tal circuito teria sempre uma parte das suas ligações em estado de alta impedância, embora pudesse alternar o conjunto de ligações nestas circunstâncias. Por outro lado, este exemplo evidencia a necessidade de se definir uma metodologia para lidar com este tipo de situações, uma vez que não existe a garantia de

que as condições descritas no quadro 4.1, relativamente às faltas do tipo curto-circuito, se possam sempre verificar.

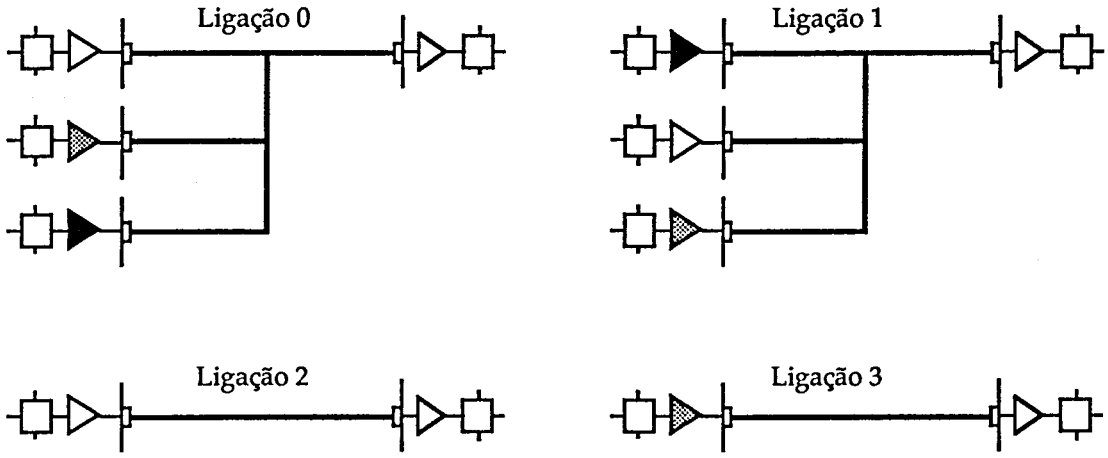


Fig.4.4: Conjunto de ligações que não podem ser colocadas sob teste em simultâneo.

A propósito da metodologia a adoptar para lidar com conjuntos de ligações nas circunstâncias ilustradas na figura 4.4, não está naturalmente colocada de parte a hipótese de se permitir que exista mais do que um nó do tipo CL activo na mesma ligação, desde que os valores lógicos que estes nós apliquem à ligação sejam coincidentes. Esta afirmação baseia-se no facto de as situações de conflito só existirem na realidade quando estes valores forem diferentes, situação em que a passagem de correntes de circulação pode danificar as saídas envolvidas. No entanto, esta possibilidade não constitui a única alternativa possível, e nem invalida o interesse de se procurar uma solução que verifique as condições pretendidas, e cuja existência terá lugar na maioria dos casos. Estas considerações justificam o interesse em se proceder a uma análise mais detalhada deste problema, e à discussão de soluções possíveis.

4.2.1. Formulação do problema

O problema da selecção dos nós do tipo CL pode ser formulado através da relação entre o conjunto de células BST de controlo, existentes numa CCI, e o correspondente conjunto de ligações. Esta relação está ilustrada na figura 4.5 para um exemplo genérico, onde se identifica, para cada célula BST de controlo, o conjunto de ligações que a ela estão associadas.

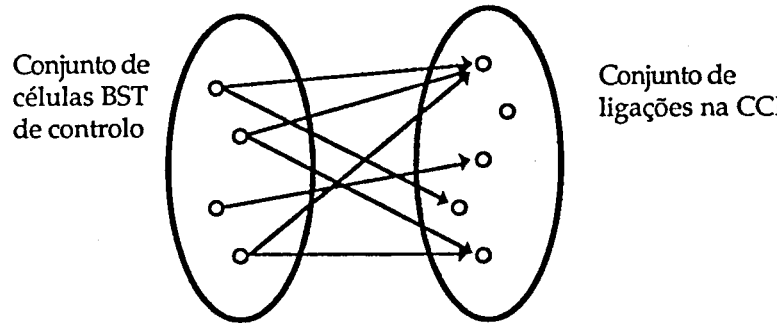


Fig.4.5: Relação entre um conjunto de células BST de controlo, e um conjunto de ligações.

Repare-se que podem existir ligações a que não está associada nenhuma célula BST de controlo, e.g. ligações a cujos nós do tipo CL estejam apenas associadas saídas do tipo colector aberto, mas que todas as células devem estar associadas a pelo menos uma ligação. De acordo com esta representação do problema, a selecção dos nós do tipo CL passa pela solução das duas questões seguintes:

i) Como determinar o menor número de subconjuntos de células BST de controlo que permitam uma cobertura completa do conjunto destas células, i.e. de tal modo que todos os nós do tipo CL existentes na CCI estejam no estado activo pelo menos uma vez, mas garantindo que em nenhum destes subconjuntos exista mais do que uma célula associada à mesma ligação, i.e. que em nenhuma ligação exista mais do que um nó do tipo CL activo ao mesmo tempo?

Repare-se que a solução para esta questão permitirá satisfazer as condições pretendidas relativamente à aplicação dos vectores de teste para as faltas de continuidade, garantindo que não ocorrerão conflitos no comando de barramentos.

ii) Como determinar um subconjunto de células BST de controlo que permita uma cobertura completa do conjunto de ligações (excluindo as que não estão associadas a nenhuma célula de controlo), mas garantindo que nenhuma ligação está associada a mais do que uma célula neste subconjunto, i.e. de modo a que em todas as ligações haja um e um só nó do tipo CL activo?

Repare-se que a solução para esta questão permitirá satisfazer as condições pretendidas relativamente à aplicação dos vectores de teste para as faltas do tipo curto-circuito, garantindo que não ocorrerão conflitos no comando de barramentos.

Esta forma de representação está ilustrada para um caso concreto na figura 4.6, onde se descreve o exemplo apresentado na figura 4.1, assumindo um barramento de dados com largura de 8 bits, e onde se apresentam ainda soluções particulares para este caso, relativamente às questões i) e ii) que foram formuladas anteriormente.

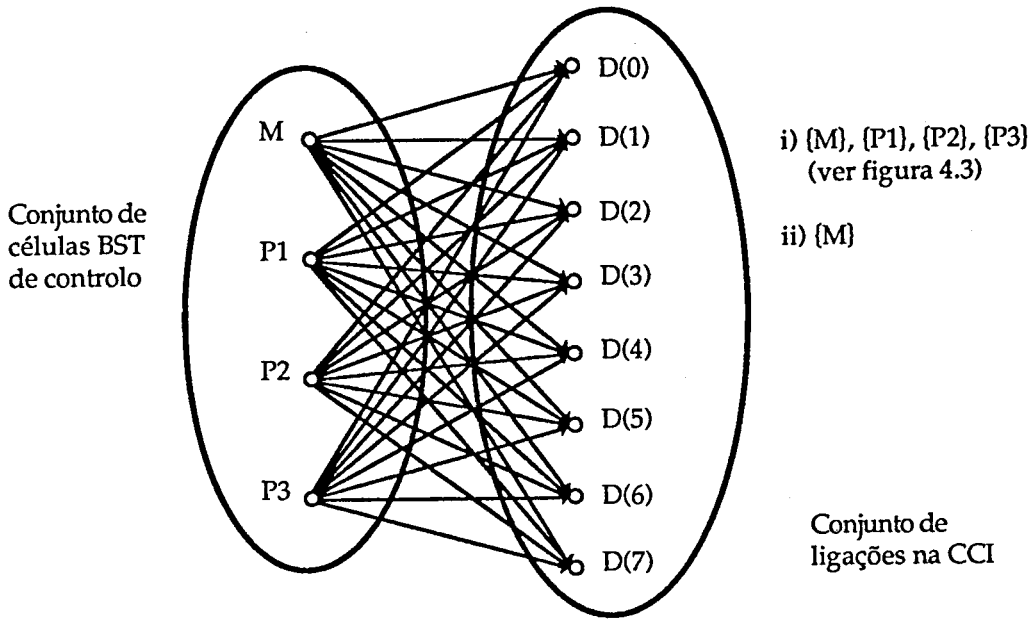


Fig.4.6: Representação correspondente ao exemplo da figura 4.1.

Já o exemplo apresentado na figura 4.7, onde se descreve o caso ilustrado na figura 4.4, mostra não ser possível encontrar uma solução para a segunda questão.

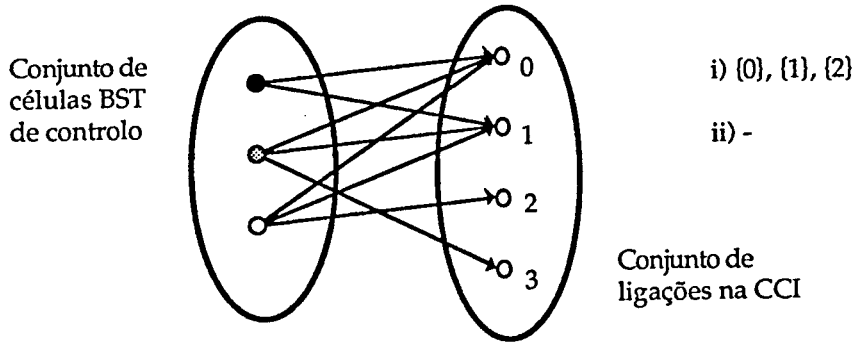


Fig.4.7: Representação correspondente ao exemplo da figura 4.4.

Uma vez formulado o problema da ocorrência de conflitos no comando de barramentos, convém agora discutir a metodologia a empregar, quando não for possível encontrar uma solução para as questões enunciadas.

4.2.2. Discussão

A procura de uma solução que satisfaça as condições descritas na questão ii) da

secção anterior (relativamente às faltas do tipo curto-circuito) deverá ser orientada por um algoritmo que leve em consideração factores como a extensão do espaço de pesquisa, e a eficiência do respectivo procedimento. A análise de problemas semelhantes pode proporcionar uma primeira orientação no sentido de se especificar um algoritmo optimizado para este fim (Christofides, 75), (Garey et al., 79), mas a implementação de um algoritmo não optimizado (por exemplo, por pesquisa exaustiva) pode igualmente constituir uma solução satisfatória.

O objectivo principal desta secção consiste no entanto não em especificar um procedimento que descreva um algoritmo possível (que será apresentado no capítulo 7), mas antes em definir a metodologia a empregar quando não for possível encontrar uma solução para este problema.

Recorde-se que o impedimento à obtenção da solução consistirá eventualmente em não ser possível determinar um conjunto de células BST de controlo que permita colocar todas as ligações sob teste simultaneamente, com um só nó do tipo CL activo em cada ligação, pelo que existem essencialmente duas alternativas disponíveis nestas circunstâncias.

Uma primeira alternativa já referida anteriormente consiste em permitir a existência de mais do que um nó do tipo CL activo na mesma ligação, quando tal for necessário, tendo-se nestas circunstâncias o cuidado de garantir que os valores lógicos forçados pelas respectivas saídas serão sempre os mesmos. Esta alternativa permite que todas as ligações sejam colocadas sob teste simultaneamente, embora não verifique a segunda das condições expressas no quadro 4.1, relativamente às faltas do tipo curto-circuito.

Uma outra alternativa consiste em isolar o conjunto das ligações relativamente às quais não foi possível resolver os conflitos no acesso a barramentos, e adiar para uma segunda fase a respectiva aplicação de vectores para o teste de curto-circuitos. O conjunto de nós do tipo CL associados a estas ligações, e que eram responsáveis pela existência das situações de conflito, serão mantidos no estado de alta impedância, enquanto decorrer o teste das restantes ligações. Uma vez terminada esta primeira fase, proceder-se-á ao teste individual de cada uma das ligações cujo teste foi adiado, pela passagem ao estado activo de apenas um dos nós do tipo CL que lhe estejam associados. Embora venham de facto a existir ligações que não se encontrarão no estado de alta impedância (já que as situações de conflito eram precisamente devidas à existência de células BST de controlo comuns), temos agora a garantia de que não existirão ligações com mais do que um nó do tipo CL activo ao mesmo tempo. Esta alternativa funciona como a dual da anterior, no que respeita às condições expressas no quadro 4.1 relativamente às faltas do tipo curto-circuito, já que não coloca todas as ligações sob teste simultaneamente, embora garanta que não haverá mais do que um nó do tipo CL activo em cada ligação.

A segunda alternativa é no entanto mais complexa do que a primeira, tornando necessária a definição de um critério que permita identificar quais as ligações cujo teste deva ser adiado para uma segunda fase. Por outro lado, a manutenção de ligações em estado de alta impedância introduz um factor adicional de incerteza relativamente aos valores capturados nos respectivos nós do tipo OL. Estas razões, associadas à necessidade da utilização de um maior número de vectores, tornam preferível a primeira alternativa descrita.

Uma vez concluída a análise dos factores que condicionam a aplicação dos vectores para o teste das ligações BST, e discutido o problema da ocorrência de conflitos no comando de barramentos, devem agora analisar-se em pormenor as questões relativas à detecção e diagnóstico dos dois tipos de faltas consideradas.

4.3. DETECÇÃO E DIAGNÓSTICO DE FALTAS DE CONTINUIDADE

Para uma ligação com p nós, em topologia de estrela, mostra-se em (Jong et al., 89, p. 22) que o número possível de faltas de continuidade é dado por

$$\frac{1}{2}\cdot\sum_{k=2}^{p-1}\frac{p!}{k!\cdot(p-k)!} \qquad (4.2)$$

Embora as localizações possíveis para uma falta deste tipo estejam directamente relacionadas com a topologia (*layout*) da ligação afectada, que não se pode por sua vez inferir de um ficheiro que contenha a descrição da lista de ligações (*netlist*), é fácil concluir que a detecção de todas as faltas deste tipo se pode garantir pela verificação da continuidade entre cada nó do tipo CL (associados a pinos de saída com ou sem terceiro estado, pinos bidireccionais, entradas primárias da CCI, ou ainda nós a que exista acesso paralelo), e o conjunto de nós do tipo OL (associados a pinos de entrada, pinos bidireccionais, saídas primárias da CCI, ou ainda nós a que exista acesso paralelo) que a ele estão ligados.

O exemplo apresentado na figura 4.8 apresenta uma ligação simples, com quatro nós, sendo dois do tipo CL e dois do tipo OL. De acordo com as considerações apresentadas no parágrafo anterior, dois testes (de CL1 para OL1 e OL2, e de CL2 para OL1 e OL2) serão suficientes para detectar as sete faltas de continuidade que a equação 4.2 indica para este caso (p=4), e que estão ilustradas na figura 4.9.

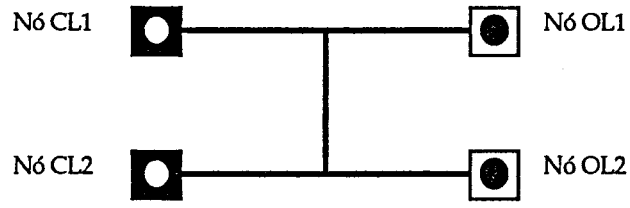


Fig. 4.8: Ligação com dois nós do tipo CL, e dois nós do tipo OL.

Uma vez que o modelo de faltas adoptado não permite conhecer qual o valor que será capturado por uma entrada isolada, a verificação de continuidade deverá realizar-se pela aplicação de ambos os níveis lógicos, a partir de cada nó CL presente na ligação. A detecção da falta será então possível por comparação dos valores lógicos capturados nos nós do tipo OL, com os valores esperados.

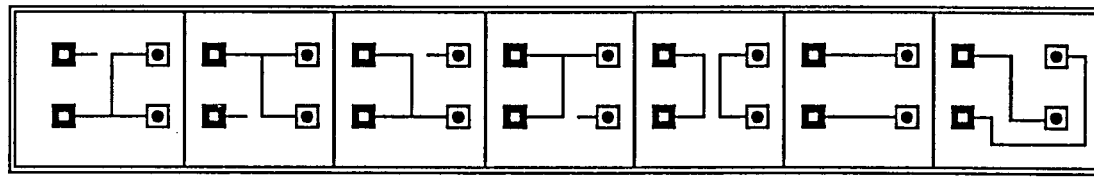


Fig.4.9: Faltas de continuidade consideradas na equação 4.2, para uma ligação com quatro nós.

Tal como foi já referido antes, e dado que as faltas de continuidade afectam as ligações individualmente, a realização de testes para a respectiva detecção pode ser conduzida simultaneamente em mais do que uma ligação. Admitindo que não ocorrem conflitos no comando de barramentos, e uma vez que cada nó CL deverá aplicar os dois níveis lógicos a cada ligação, o número de vectores de teste necessários para a detecção de faltas de continuidade será dado pelo dobro do maior número de nós CL existentes numa única ligação.

4.4. DETECÇÃO E DIAGNÓSTICO DE CURTO-CIRCUITOS

Tal como foi já referido na secção 4.1 (Factores que condicionam a detecção de faltas), o teste relativamente às faltas do tipo curto-circuito é consideravelmente mais complicado do que o correspondente às faltas de continuidade, essencialmente pelo número muito superior de possíveis combinações de defeito, que então se referiu ser dado por

$$2^N - (N+1) \qquad \text{(N: número de ligações)} \qquad (4.3)$$

A detecção de um curto-circuito entre um par de ligações efectua-se pela aplicação simultânea a estas ligações de valores lógicos complementares, o que significa que em ambas as ligações deve existir um nó do tipo CL activo. Uma vez que se considera possível a ocorrência de qualquer combinação de defeito, todas as ligações devem ser colocadas sob teste simultaneamente, sendo a inexistência de conflitos em barramentos a única condição que limita a escolha do nó do tipo CL a activar em cada ligação.

A geração de vectores para o teste de curto-circuitos diz deste modo respeito à definição da sequência de valores lógicos a aplicar a cada ligação (VTS, Vector de Teste Sequencial), de modo a tornar possível a detecção de qualquer curto-circuito em que esta ligação possa estar envolvida. Como se verá mais adiante, quando se apresentarem os vários algoritmos disponíveis para este fim, a capacidade de diagnóstico varia substancialmente de algoritmo para algoritmo, e depende da informação que se consegue inferir a partir da sequência de valores lógicos capturada em cada ligação (VRS, Vector de Resposta Sequencial).

Convirá nesta altura esclarecer que a expressão *vector de teste* será aqui empregue relativamente ao conjunto de valores lógicos que são aplicados em simultâneo (paralelo) ao conjunto de ligações sob teste. A designação VTS refere-se por sua vez à sequência de valores lógicos que surgem numa ligação, por aplicação sucessiva de um conjunto de vectores de teste (p vectores de teste aplicam VTSs com p bits). Tem-se assim que:

i) Cada ligação só poderá ter um VTS único se p verificar a

$$2^P \geq N \tag{4.4}$$

ii) Cada uma das combinações de defeito dadas pela equação 4.3 só poderá ter um VRS único se p verificar a

$$2^P \geq 2^N - (N+1) \tag{4.5}$$

A primeira destas equações dá-nos um minorante para o número de vectores de teste a gerar, de modo a tornar possível a completa detecção de faltas do tipo curto-circuito, enquanto a segunda especifica um número de vectores de teste que tornaria possível a completa capacidade de diagnóstico (com um único conjunto de vectores, e sem recurso a outra informação adicional). Repare-se no entanto que o número de vectores especificado pela equação 4.5 não constitui um minorante, uma vez que o número de combinações de defeito consideradas nesta equação inclui igualmente curto-circuitos não disjuntos, em relação aos quais não se torna necessária a existência de VRSs distintos.

Uma descrição sumária dos vários algoritmos para a geração de vectores para o teste de curto-circuitos entre ligações será agora apresentada, distinguindo-se entre os algoritmos de um passo só, e os algoritmos adaptativos. Os algoritmos de um passo só geram um

único conjunto de vectores, enquanto os algoritmos adaptativos geram dois conjuntos. O segundo conjunto gerado por um algoritmo adaptativo pretende tirar partido das respostas obtidas para um primeiro conjunto com um número reduzido de vectores, no sentido de conseguir uma maior capacidade de diagnóstico, com um reduzido número total de vectores. Estas considerações apontam desde já para o facto de cada algoritmo corresponder a uma solução de compromisso particular entre a capacidade de diagnóstico, e o número de vectores de teste gerados. Como se verá mais adiante, quando se descreverem os algoritmos adaptativos, também a complexidade da análise necessária para as respostas ao primeiro conjunto de vectores representa um factor importante na solução de compromisso referida.

A apresentação de cada algoritmo será feita de acordo com um enquadramento que inclui a apresentação sucinta do procedimento de geração dos vectores de teste, a apresentação de um quadro que ilustra a sua aplicação relativamente a um exemplo simples, e ainda um conjunto de comentários sobre a capacidade de diagnóstico, a complexidade, e o número de vectores de teste gerados.

Os modelos de faltas subjacentes a estes algoritmos apresentam em geral ligeiras diferenças. No entanto, e no sentido de uniformizar a apresentação, considera-se que o resultado de um curto-circuito entre ligações corresponde à **disjunção** dos valores lógicos que lhes estão aplicados. Ainda a este respeito, e de acordo com a equação 4.4, o número mínimo de vectores a gerar para a detecção de curto-circuitos entre N ligações é dado por

$$p = \lceil \log_2(N) \rceil \quad ^{\dagger} \tag{4.6}$$

A equação 4.6 indica que, quando for $\lceil \log_2(N) \rceil = \log_2(N)$, i.e. quando N for uma potência inteira de 2, então todos os 2^p VTSs com p bits serão usados, e em particular os dois VTSs em que todos os bits são **0**, e em que todos os bits são **1**. Repare-se no entanto que estes dois VTSs não permitiriam detectar as faltas do tipo s@ que forçam na ligação o mesmo nível lógico, pelo que vários autores optam por excluí-los do conjunto de vectores gerados. O número mínimo de vectores será nestas circunstâncias dado por

$$p = \lceil \log_2(N+2) \rceil \quad ^{\dagger\dagger} \tag{4.7}$$

---

† $\lceil x \rceil$ representa o menor inteiro não inferior a x.

†† Repare-se que esta expressão equivale a considerar que o circuito tem na realidade (N+2) ligações, reservando-se para as duas ligações fictícias a atribuição dos VTSs referidos.

4.4.1. Algoritmos de um passo só

Esta secção fará a apresentação de um grupo de oito algoritmos que geram um único conjunto de vectores de teste. Os algoritmos apresentados estão ordenados cronologicamente de acordo com a data da sua publicação.

4.4.1.1. Algoritmo da partição binária (Kautz, 74)

A cada ligação é atribuído um VTS com $\lceil \log_2(N) \rceil$ bits, gerado a partir de uma sequência de contagem binária ascendente, com início no VTS 0...0. O conjunto de vectores de teste gerados corresponderia, para o caso de $\lceil \log_2(N) \rceil = \log_2(N)$, a efectuar a partição sucessiva do conjunto de ligações em duas metades, até que esta partição resultasse em conjuntos singulares. A aplicação de valores lógicos opostos às ligações em cada uma destas metades, para cada partição efectuada, permitirá detectar a ocorrência de um curto-circuito que envolva quaisquer ligações.

O conjunto de vectores gerados por este algoritmo está ilustrado no quadro 4.2, onde se representam também as respostas esperadas, assumindo a existência de curto-circuitos independentes entre as ligações {1,4}, {2,9}, e {3,8}. Os VRSs com defeito, i.e. que resultam diferentes dos respectivos VTSs (o que evidencia a existência de uma falta), estão assinalados com o sinal <.

Ligação	VTS	VRS
0	0000	0000
1	0001	0101 <
2	0010	1011 <
3	0011	1011 <
4	0100	0101 <
5	0101	0101
6	0110	0110
7	0111	0111
8	1000	1011 <
9	1001	1011 <
10	1010	1010

Quadro 4.2: Algoritmo da partição binária.

O conjunto de VRSs ilustrados no quadro 4.2 será usado para introduzir os conceitos de síndromas dos tipos *aliasing* e *confounding* (Jarwala et al., 89, p. 66), que caracterizam os dois tipos de ambiguidade que afectam a capacidade de diagnóstico de um algoritmo.

Um síndroma do tipo *aliasing* ocorre quando, num conjunto com três ou mais ligações que apresentam um VRS comum e igual ao VTS atribuído a uma delas, a disjunção dos VTSs atribuídos às restantes ligações deste conjunto continua a produzir o mesmo VRS. Nestas circunstâncias, torna-se impossível distinguir se a ligação cujo VRS coincide com o respectivo VTS estará ou não envolvida no curto-circuito. É este o caso das ligações {1,4,5} no quadro 4.2, relativamente às quais não se pode concluir se a ligação {5} estará ou não curto-circuitada com as ligações {1,4}.

Um síndroma do tipo *confounding* ocorre quando os VRSs produzidos por curto-circuitos independentes são iguais. Nestas circunstâncias, torna-se impossível distinguir se as ligações que apresentam este VRS comum estarão ou não todas curto-circuitadas entre si. É este o caso das ligações {2,3,8,9}, relativamente às quais não se pode concluir se existirão dois curto-circuitos independentes, entre as ligações {2,9} e {3,8}, ou um único curto-circuito entre estas quatro ligações. Repare-se que um síndroma pode ser simultaneamente dos tipos *aliasing* e *confounding*, desde que se verifiquem simultaneamente as respectivas condições. Define-se ainda o *grau* de um síndroma do tipo *confounding*, como sendo o número máximo de curto-circuitos independentes que podem ter lugar (o exemplo apresentado tem grau 2).

4.4.1.2. Algoritmo de Wagner (Wagner, 87)

De acordo com este algoritmo, a segunda metade do conjunto de vectores gerados é obtida por complemento da primeira metade, sendo esta determinada pela atribuição a cada ligação de um VTS com $\lceil \log_2(N+2) \rceil$ bits, gerados a partir de uma sequência de contagem binária ascendente, com início no VTS **0...01**. O número total de vectores gerados será deste modo dado por $2 \cdot \lceil \log_2(N+2) \rceil$.

O conjunto de vectores gerados por este algoritmo está ilustrado no quadro 4.3, onde estão também representadas as respostas esperadas, assumindo a existência de curto-circuitos independentes entre as ligações {3,6} e {4,5}.

Relativamente ao algoritmo da partição binária, o algoritmo de Wagner aumenta a capacidade de diagnóstico, por recurso a um maior número de vectores de teste. Repare-se que o procedimento de geração dos vectores de teste faz com que todos os VTSs tenham o

mesmo número de 1s (p/2), o que tem por consequência que os VRSs apresentados por ligações em curto-circuito passem a ter um maior número de 1s, relativamente aos respectivos VTSs. Isto significa que todas as ligações envolvidas em curto-circuitos apresentarão um VRS distinto do seu VTS, i.e. este algoritmo impede a ocorrência de síndromas do tipo *aliasing*. No entanto, e tal como se ilustra pelo exemplo dos curto-circuitos entre as ligações {3,6} e {4,5}, a possibilidade de ocorrência de síndromas do tipo *confounding* permanece.

Ligação	VTS	VRS
0	00011110	00011110
1	00101101	00101101
2	00111100	00111100
3	01001011	01111011 <
4	01011010	01111011 <
5	01101001	01111011 <
6	01111000	01111011 <
7	10000111	10000111
8	10010110	10010110
9	10100101	10100101
10	10110100	10110100

Quadro 4.3: Algoritmo de Wagner.

4.4.1.3. Algoritmo da sequência caminhante (Hassan et al., 88)

O algoritmo da sequência caminhante atribui a cada ligação um VTS que é obtido a partir do VTS atribuído à ligação anterior, por deslocamento do **1** uma posição para a direita. À primeira ligação é atribuído um VTS com um **1** na primeira posição da esquerda, sendo todos os restantes bits **0**.

O conjunto de vectores gerados por este algoritmo está ilustrado no quadro 4.4, onde estão também representadas as respostas esperadas, assumindo a existência de curto-circuitos independentes entre as ligações {3,6} e {4,5}.

O algoritmo da sequência caminhante tem como propriedade fundamental permitir uma completa capacidade de diagnóstico, embora consiga esta característica à custa de um número de vectores igual ao número de ligações. Esta propriedade está relacionada com o

conceito de *independência diagonal* de um conjunto de vectores de teste, tal como é definido em (Jarwala et al., 89, p. 67).

Ligação	VTS	VRS
0	10000000000	10000000000
1	01000000000	01000000000
2	00100000000	00100000000
3	00010000000	00010010000 <
4	00001000000	00001100000 <
5	00000100000	00001100000 <
6	00000010000	00010010000 <
7	00000001000	00000001000
8	00000000100	00000000100
9	00000000010	00000000010
10	00000000001	00000000001

Quadro 4.4: Algoritmo da sequência caminhante.

Seja S_{NxM} a matriz correspondente ao conjunto de vectores de teste (N linhas, M colunas, M≥N), e seja b_{ij}, onde 0≤i≤N-1, 0≤j≤M-1, um elemento de S. Então, se S ou a matriz obtida de S por trocas sucessivas de linhas e/ou colunas, for tal que

$$b_{ij} = \begin{cases} 1, & \text{para todo o } i=j \\ 0, & \text{para todo o } i>j \\ x, & \text{para todo o } i<j \end{cases} \qquad (4.8)$$

onde $x \in \{0,1\}$, então S diz-se diagonalmente independente. Repare-se que nenhum par de linhas de uma matriz diagonalmente independente pode ter o primeiro **1** na mesma posição, pelo que não é possível que um curto-circuito entre ligações provoque um VRS em que o primeiro **1** surja na mesma posição do VTS atribuído a uma ligação não envolvida neste curto-circuito, i.e. nunca ocorrerão síndromas do tipo *aliasing*. Do mesmo modo, não é possível que dois curto-circuitos independentes produzam VRSs cujo primeiro **1** surja na mesma posição, pelo que também nunca ocorrerão síndromas do tipo *confounding*.

Tal como se verá quando se apresentar o algoritmo do diagnóstico máximo (Lien et al., 91a), a existência de faltas múltiplas numa mesma ligação pode produzir resultados insatisfatórios, mesmo com um conjunto de vectores diagonalmente independente. O

algoritmo da sequência caminhante manterá no entanto uma completa capacidade de detecção e diagnóstico, mesmo nestas circunstâncias.

4.4.1.4. Algoritmo do peso mínimo (Yau et al., 89)

Este algoritmo atribui sequencialmente VTSs com o menor peso possível, i.e. com o menor número possível de 1s. Uma vez que o número de VTSs de p bits, com peso w, é dado por

$$\binom{p}{w} = \frac{p!}{w!\cdot(p-w)!} \tag{4.9}$$

então o peso máximo (w_{max}) dos VTSs atribuídos a um circuito com N ligações será dado pelo menor valor de k que satisfaça a

$$\sum_{i=1}^{k} \binom{p}{i} \geq N \tag{4.10}$$

O conjunto de vectores gerados por este algoritmo, considerando p=5, está ilustrado no quadro 4.5, onde estão também representadas as respostas esperadas, assumindo a existência de curto-circuitos independentes entre as ligações {0,2}, {1,7} e {3,5}.

Ligação	VTS	VRS
0	10000	10100 <
1	01000	11010 <
2	00100	10100 <
3	00010	11010 <
4	00001	00001
5	11000	11010 <
6	10100	10100
7	10010	11010 <
8	10001	10001
9	01100	01100
10	01010	01010

Quadro 4.5: Algoritmo do peso mínimo.

Uma consequência deste algoritmo é que w_{max} é sempre menor ou igual que o peso de qualquer um dos (2^p-N) VTSs não atribuídos, à excepção do VTS 0...0. Uma vez que o

curto-circuito entre ligações produz VRSs cujo peso é sempre maior ou igual que o peso dos respectivos VTSs, este procedimento de geração de vectores torna menos provável a ocorrência de síndromas do tipo *aliasing*. Aliás, esta probabilidade será tanto menor quanto maior for o valor arbitrado para p, dentro do intervalo

$$\lceil \log_2(N+2) \rceil \leq p \leq N \tag{4.11}$$

Repare-se que se $p=\lceil \log_2(N+2) \rceil$, e se N for tal que $\lceil \log_2(N+2) \rceil = \log_2(N+2)$, i.e. se todos os VTSs possíveis forem atribuídos, então o conjunto de vectores gerados coincide com o gerado pelo algoritmo da partição binária (diferindo apenas na ordem de atribuição dos VTSs às ligações), pelo que a ambiguidade no diagnóstico será também a mesma. Por outro lado, quando for p=N, o conjunto de vectores gerados será exactamente o mesmo que é gerado pelo algoritmo da sequência caminhante, e permitirá total capacidade de diagnóstico. O exemplo apresentado no quadro 4.5, em que se assumiu o valor mínimo para p, ilustra a ocorrência de síndromas dos tipos *aliasing* {0,2,6} e *confounding* {1,3,5,7}.

4.4.1.5. **Algoritmo da independência máxima** (Yau et al., 89)

O algoritmo da independência máxima minimiza o número de vectores a gerar, mantendo completa capacidade de diagnóstico. Este objectivo é atingido pelo recurso a informação adicional, que caracteriza a adjacência entre ligações, e a extensão máxima para as faltas do tipo curto-circuito, i.e. a indicação do número máximo de ligações que poderão estar curto-circuitadas entre si. O procedimento para a geração dos vectores de teste correspondentes a este algoritmo recorre aos conceitos de *peso potencial* ($\tilde{w}$), e de conjunto de ligações *ordenadas por adjacência*, descritos em (Yau et al., 89, p. 72).

Dado um VTS em que i e j correspondam à primeira e última posições em que exista um 1, define-se o peso potencial ($\tilde{w}$) deste VTS como sendo igual a j-i+1. Para o caso particular do VTS 0...0, define-se $\tilde{w}=0$. De acordo com esta definição, o número $N_{\tilde{w}}$ de VTSs a que corresponde um peso potencial $\tilde{w}$ é dado por

$$N_{\tilde{w}} = \begin{cases} 1, \text{ para } \tilde{w}=0 \\ p, \text{ para } \tilde{w}=1 \\ (p-\tilde{w}+1)\cdot 2^{\tilde{w}-2}, \text{ para } \tilde{w}>1 \end{cases} \tag{4.12}$$

Diz-se que um conjunto com N ligações $\{n_1,n_2,...,n_N\}$ está ordenado por adjacência, se

n_i for mais adjacente, i.e. mais provável que venha a estar em curto-circuito, com n_{i+1} do que com n_{i+2}, para $1 \leq i \leq N-2$, e se n_i for mais adjacente com n_{i-1} do que com n_{i-2}, para $3 \leq i \leq N$.

Com base nos conceitos definidos anteriormente, o procedimento de geração dos vectores de teste consiste na seguinte sequência:

i) Determine-se o menor p (número de bits de cada VTS) que permite uma completa capacidade de diagnóstico, para um número máximo de ligações em curto-circuito que não exceda um limite E pré-determinado, de acordo com

$$p = \begin{cases} \lceil E + \log_2(N+1) - \log_2 E - 1 \rceil \\ \text{se } E>2 \text{ ou } \log_2(N+1) < \lceil \log_2(N+1) \rceil \\ \\ \lceil \log_2(N+2) \rceil \\ \text{se } E=2 \text{ e } \log_2(N+1) = \lceil \log_2(N+1) \rceil \end{cases} \qquad (4.13)$$

ii) Construa-se uma lista de todas as ligações presentes na CCI, ordenadas por adjacência.

iii) Construam-se subconjuntos de VTSs com p bits (excluindo os VTSs **0...0** e **1...1**), tais que cada subconjunto seja constituído por todos os VTSs que tenham o mesmo peso potencial, e o mesmo peso de Hamming, i.e. o mesmo número de elementos não nulos (Blahut, 83, p. 46).

iv) A partir do subconjunto de VTSs com o peso potencial 1, acrescente-se repetidamente o próximo subconjunto de VTSs com peso potencial mínimo, até que o conjunto assim formado contenha N VTSs. Quando dois ou mais subconjuntos de VTSs tiverem o mesmo peso potencial, use-se em primeiro lugar aquele que tiver menor peso de Hamming.

O conjunto de vectores de teste gerados por este algoritmo, considerando E=4, está ilustrado no quadro 4.6, onde estão também representadas as respostas esperadas, assumindo a existência de um curto-circuito entre as ligações {0,1}.

A propriedade fundamental do algoritmo da independência máxima é que, desde que a extensão de um curto-circuito não exceda o limite superior que foi admitido (E), e desde que não existam faltas múltiplas numa mesma ligação, existirá sempre total capacidade de diagnóstico. A razão para este facto decorre de o conjunto de ligações estar ordenado por adjacência, e de quaisquer E VTSs consecutivos possuírem a propriedade da independência

diagonal. Apesar de, no exemplo ilustrado no quadro 4.6, os VRSs das ligações {0,1} coincidirem com o VRS da ligação {5}, não tem no entanto lugar um síndroma do tipo *aliasing*, já que o facto de o conjunto de ligações estar ordenado por adjacência elimina a possibilidade de esta última ligação estar envolvida no curto-circuito, sem que as ligações {2,3,4} o estejam.

Ligação	VTS	VRS
0	10000	11000 <
1	01000	11000 <
2	00100	00100
3	00010	00010
4	00001	00001
5	11000	11000
6	01100	01100
7	00110	00110
8	00011	00011
9	10100	10100
10	01010	01010

Quadro 4.6: Algoritmo da independência máxima.

A importância deste algoritmo na redução do número de vectores de teste gerados pode ser ilustrada para o caso de um circuito com 1.000 ligações, relativamente ao qual se assuma um E=20. Nestas circunstâncias, o conjunto de vectores de teste gerados terá apenas 25 elementos, face aos 1.000 que o algoritmo da sequência caminhante produziria, para a mesma capacidade de diagnóstico. A dificuldade principal na aplicação deste algoritmo consiste porém na obtenção do conjunto de ligações ordenado por adjacência, embora se possa obter uma adjacência parcial razoável se se identificar a adjacência de ligações com a adjacência de pinos.

4.4.1.6. Algoritmo do auto-diagnóstico (Cheng et al., 90)

Tal como acontecia já com o conjunto de vectores gerado pelo algoritmo de Wagner, o conjunto de vectores gerado por este algoritmo garante que qualquer ligação envolvida num curto-circuito apresentará um VRS diferente do seu VTS.

Este algoritmo é apresentado com base no conceito de *cobertura* entre vectores (Cheng

et al., 90, p. 566), entendendo-se que um VTS **a** cobre um VTS **b**, sse **a** tem um **1** em todas as posições em que **b** tem um **1**. Um conjunto de VTSs diz-se *independente*, se nenhum VTS cobrir outro VTS. Repare-se que um conjunto de VTSs que disponha desta propriedade garantirá a inexistência de síndromas do tipo *aliasing*, uma vez que o VRS produzido por um conjunto de ligações em curto-circuito só poderia coincidir com o VTS atribuído a uma outra ligação, se o VTS desta ligação cobrisse os VTSs das ligações envolvidas no curto-circuito.

Ligação	VTS	VRS
0	111000	111100 <
1	110100	111100 <
2	110010	110010
3	110001	110001
4	101100	111100 <
5	101010	101010
6	101001	101001
7	100110	100110
8	100101	100101
9	100011	100011
10	011100	111100 <

Quadro 4.7: Algoritmo do auto-diagnóstico.

O procedimento para a geração de vectores de teste deste algoritmo gera todos os VTSs de p bits, com (p/2) † 0s. Deste modo, o número mínimo de vectores de teste gerados para um circuito com N ligações será dado pelo mínimo valor de p que satisfizer a

$$\binom{p}{p/2} \geq N \qquad (4.14)$$

O conjunto de vectores de teste gerados por este algoritmo está ilustrado no quadro 4.7, onde estão também representadas as respostas esperadas, assumindo a existência de curto-circuitos independentes entre as ligações {0,1} e {4,10}.

Apesar de garantir a inexistência de síndromas do tipo *aliasing*, este algoritmo não impede a ocorrência de síndromas do tipo *confounding*, como o ilustra o VRS resultante do curto-circuito entre as ligações {0,1,4,10}. Possuindo a mesma capacidade de diagnóstico que o algoritmo de Wagner, este algoritmo apresenta no entanto a vantagem de gerar um

† p/2 representa o quociente da divisão inteira de p por 2.

menor número de vectores de teste, para o mesmo número de ligações.

4.4.1.7. Algoritmo do diagnóstico global (Cheng et al., 90)

O algoritmo do diagnóstico global garante a inexistência de síndromas do tipo *aliasing*, embora não garanta que qualquer ligação envolvida num curto-circuito apresente um VRS diferente do seu VTS. Apesar de conservar a mesma capacidade de diagnóstico característica dos algoritmos do auto-diagnóstico, e de Wagner, este algoritmo é o que, de entre os três, gera o menor número de vectores de teste. Esta redução torna-se possível pelo facto de os restantes dois algoritmos conterem condições suficientes, mas não necessárias, para garantir a inexistência dos síndromas do tipo *aliasing*.

A apresentação deste algoritmo é feita com base no conceito de *esparsidade* de um conjunto de vectores de teste (Cheng et al., 90, p. 567), entendida como a propriedade pela qual um conjunto de ligações em curto-circuito apresentará um VRS que será diferente de qualquer VTS atribuído a uma ligação não envolvida no curto-circuito.

O procedimento para a geração dos vectores de teste é apresentado de forma recursiva, sobre dois conjuntos designados por T(p,1), definido como o conjunto de todos os VTSs com p bits gerados pelo algoritmo do auto-diagnóstico, e por T(p,2), definido como o conjunto de todos os VTSs com p bits gerados por este algoritmo do diagnóstico global. Nestas circunstâncias, T(p+1,2) será obtido pela reunião de dois subconjuntos disjuntos, X(p+1) e Y(p+1). X(p+1) contem todos os VTSs de T(p,2), com um 0 acrescentado à direita, e Y(p+1) contem todos os VTSs de T(p,1), com um 1 acrescentado à direita. Os conjuntos iniciais são T(1,1)={0} e T(1,2)={0,1}. Uma vez que o tamanho de T(p+1,2) é dado pela soma do tamanho de T(p,2) com $\binom{p}{p/2}$, resulta que p+1 vectores de teste serão suficientes para testar

$$2 + \sum_{i=1}^{p} \binom{i}{i/2} \tag{4.15}$$

ligações.

O conjunto de vectores de teste gerados por este algoritmo está ilustrado no quadro 4.8, onde estão também representadas as respostas esperadas, assumindo a existência de curto-circuitos independentes entre as ligações {0,2,10}, {5,9} e {6,8}.

Apesar de este algoritmo garantir a inexistência de síndromas do tipo *aliasing*, pode tornar-se necessária uma análise mais complexa do conjunto de VRSs resultantes. Com efeito, torna-se necessário verificar se o VRS produzido por um conjunto de ligações em

curto-circuito coincide com algum dos VTSs que lhes foram atribuídos. Só desta forma se torna possível concluir se nenhuma das ligações que apresentam VRSs correctos estará envolvida num curto-circuito, tal como o ilustra o caso das ligações {0,2,10}, apresentado no quadro 4.8. A possibilidade de ocorrerem síndromas do tipo *confounding* continua a existir, tal como o ilustra o VRS produzido pelas ligações {5,6,8,9}.

Ligação	VTS	VRS
0	00000	01101 <
1	10000	10000
2	01000	01101 <
3	01100	01100
4	10100	10100
5	00110	01111 <
6	01010	01111 <
7	10010	10010
8	00111	01111 <
9	01011	01111 <
10	01101	01101

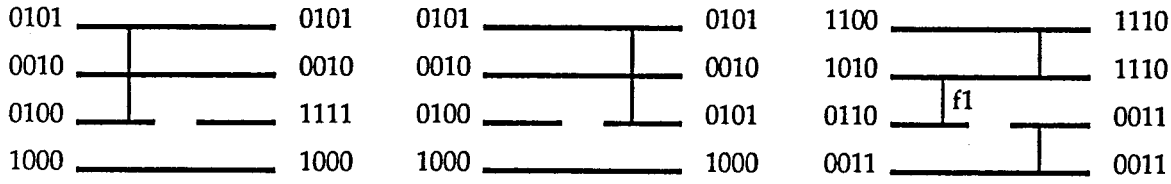
Quadro 4.8: Algoritmo do diagnóstico global.

Este algoritmo ilustra uma questão que será retomada mais do que uma vez no seguimento deste capítulo, e que consiste na necessidade de se encontrar uma solução de compromisso entre o número de vectores de teste, e a complexidade requerida na análise das respostas, para uma mesma capacidade de diagnóstico.

4.4.1.8. Algoritmo do diagnóstico máximo (Lien et al., 91a)

Este algoritmo requer uma informação de vizinhança entre as ligações existentes, e faz uso de um modelo em que se admite a existência de faltas múltiplas numa mesma ligação. A existência de faltas múltiplas pode ter por consequência que algumas delas não sejam detectáveis pelos algoritmos anteriormente descritos. Os exemplos apresentados na figura 4.10, extraídos de (Lien et al., 91a, p. 100), ilustram deficiências na capacidade de detecção de faltas, quer por um conjunto de vectores que apresenta a propriedade da independência diagonal (figuras 4.10 (a) e (b)), quer por um conjunto em que nenhum VTS cobre qualquer

outro (figura 4.10 (c)).†



(a) Curto-circuito não detectável.

(b) Falta de continuidade não detectável.

(c) Curto-circuito (f1) não detectável.

Fig.4.10: Problemas de detecção relacionados com a existência de faltas múltiplas na mesma ligação.

O algoritmo do diagnóstico máximo estende a noção de cobertura entre vectores para a escala do conjunto de vectores (*set-cover independent*). Um conjunto $S=\{VTS_1, VTS_2, ...,VTS_n\}$ diz-se com independência de cobertura quando, para qualquer i=1,2, ..., n, o VTS_i não cobre, nem é coberto, pela união de qualquer subconjunto de vectores em $S-\{VTS_i\}$. Repare-se que o conjunto de vectores gerados pelo algoritmo da sequência caminhante goza naturalmente desta propriedade, e garante completa capacidade de detecção, relativamente às faltas ilustradas na figura 4.10.

Considerando um conjunto de ligações W, e designando por $N(w_i) \in W$ o conjunto de ligações que são vizinhas de w_i, o algoritmo do diagnóstico máximo descreve-se como se segue:

i) Construa-se um grafo de vizinhanças NG=(V,E), em que:

- Para cada $w_i \in W$ existe um nó $v_i \in V$.
- O conjunto de transições é formado por $E=(E_1 \cup E_2)$, onde

$$E_1 = \{e \mid e=(v_i, v_j), \forall w_j \in N(w_i), \forall w_i \in W\}\ ^{††}, \text{e}$$

$$E_2 = \{e \mid e=(v_j, v_k), \forall v_j, v_k \in N(w_i), \forall v_i \in V\}^{†††}$$

ii) Atribua-se a cada nó v_i de NG uma cor $C(v_i)$, tal que $C(v_i) \neq C(v_j)$ se $e=(v_i,v_j) \in E$, e que o número de cores seja mínimo.

iii) Associe-se a cada cor uma sequência binária (VTS) única, de modo a que apenas um bit em cada VTS tome o valor lógico **1**.

---

† O modelo de faltas adoptado neste algoritmo assume que uma entrada no ar captura o valor lógico **1**.

†† Conjunto das transições entre cada ligação e cada uma das suas ligações vizinhas.

††† Conjunto das transições entre ligações que são vizinhas de uma mesma ligação.

iv) Associe-se a cada ligação $w_i \in W$ o VTS correspondente à cor que foi atribuída ao respectivo nó v_i de NG.

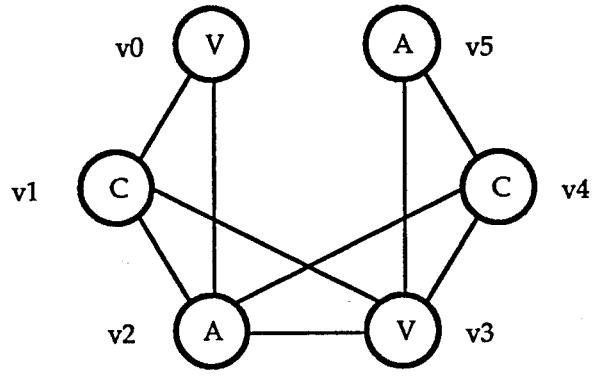


Fig.4.11: Grafo de vizinhanças, e atribuição de cores, para um exemplo com 6 ligações.

A figura 4.11 ilustra o exemplo apresentado em (Lien et al., 91a, pp. 104 e 105), para um circuito composto por 6 ligações, em que $N(w_0)=\{w_1\}$, $N(w_1)=\{w_0,w_2\}$, $N(w_2)=\{w_1,w_3\}$, $N(w_3)=\{w_2,w_4\}$, $N(w_4)=\{w_3,w_5\}$, $N(w_5)=\{w_4\}$. A aplicação do passo ii) resulta na atribuição de três cores, representadas por V, C, e A.

A atribuição dos VTSs {100, 010, 001} às cores {V, C, A} resulta no conjunto de vectores ilustrado no quadro 4.9, onde estão também representadas as respostas esperadas, assumindo a existência de curto-circuitos entre as ligações {1,2} e {4,5}.

Ligação	VTS	VRS
0	100	100
1	010	011 <
2	001	011 <
3	100	100
4	010	011 <
5	001	011 <

Quadro 4.9: Algoritmo do diagnóstico máximo.

Repare-se que o facto de as ligações {1,2,4,5} apresentarem o mesmo VRS não corresponde de facto a uma situação de ambiguidade, o mesmo sucedendo com as ligações {0,3}. A consulta às relações de vizinhança características deste circuito permite concluir que as ligações {0} e {3} não podem estar em curto-circuito, e que os curto-circuitos {1,2} e {4,5} são independentes.

Apesar de garantir uma completa capacidade de detecção e diagnóstico com um conjunto de vectores de tamanho mínimo, este algoritmo apresenta o inconveniente de requerer a elaboração de uma lista de vizinhanças para cada ligação presente. Para além da dificuldade inerente à definição de um conceito de vizinhança, e à extracção da relação de vizinhanças para cada circuito, deve também referir-se que a aplicação do passo ii) requer a resolução de um problema do tipo NP-completo.

4.4.2. Algoritmos adaptativos

O objectivo global dos algoritmos adaptativos consiste em empregar um primeiro conjunto de vectores com completa capacidade de detecção, e tamanho reduzido, reservando para uma segunda fase a clarificação de situações de diagnóstico impreciso. Para a mesma capacidade de detecção, o número total de vectores gerados por um algoritmo adaptativo é inferior ao que seria necessário se fosse empregue um algoritmo de um passo só. Esta redução é possível pelo facto de a geração do segundo conjunto de vectores dispor já da informação correspondente a uma primeira operação de diagnóstico.

Esta secção fará a apresentação de um grupo de seis algoritmos que permitem a geração de um segundo conjunto de vectores de teste, caso a análise das respostas obtidas para o primeiro conjunto assim o determine.

4.4.2.1. Algoritmo de Goel e McMahon (Goel et al., 82)

O primeiro conjunto de vectores é determinado pela atribuição a cada ligação de um VTS com $\lceil \log_2(N+2) \rceil$ bits, gerados através de uma sequência de contagem binária ascendente, com início no VTS **0...01**.

Assumindo que a aplicação deste primeiro conjunto de vectores resultou na detecção de um subconjunto de W ligações que produziram VRSs com defeito, a geração de um segundo conjunto terá lugar pela atribuição sucessiva de um **1** a cada uma destas ligações, enquanto a todas as restantes é atribuído um **0**.

Os conjuntos de vectores gerados por este algoritmo estão ilustrados no quadro 4.10, onde estão também representadas as respostas esperadas, assumindo a existência de curto-circuitos independentes entre as ligações {0,3}, {1,8} e {2,7}.

	Primeiro conjunto		Segundo conjunto	
Ligação	VTS	VRS	VTS	VRS
0	0001	0101 <	100000	100100 <
1	0010	1011 <	010000	010001 <
2	0011	1011 <	001000	001010 <
3	0100	0101 <	000100	100100 <
4	0101	0101	000000	000000
5	0110	0110	000000	000000
6	0111	0111	000000	000000
7	1000	1011 <	000010	001010 <
8	1001	1011 <	000001	010001 <
9	1010	1010	000000	000000
10	1011	1011	000000	000000

Quadro 4.10: Algoritmo de Goel e McMahon.

Repare-se que o segundo conjunto de vectores, relativamente às W ligações cujo VRS diferiu do respectivo VTS, goza da propriedade de independência diagonal, pelo que seria possível uma completa capacidade de detecção e diagnóstico, desde que este conjunto de vectores atribua VTSs não nulos a todas as ligações com defeito. No entanto, e tal como se exemplificará quando se apresentar o algoritmo adaptativo A1 (Lien et al., 91a), a existência de faltas múltiplas pode levar a que ligações com defeito exibam VRSs correctos, pelo que os resultados obtidos nestas circunstâncias não serão satisfatórios.

4.4.2.2. Algoritmo do vector único (Jarwala et al., 89)

O primeiro conjunto de vectores é determinado pela atribuição a cada ligação de um VTS com $\lceil \log_2(N+2) \rceil$ bits, gerados através de uma sequência de contagem binária ascendente, com início no VTS **0...01**.

Se os VRSs correspondentes a este primeiro conjunto incluírem síndromas do tipo *aliasing*, terá então lugar a geração de um único vector de teste, que constituirá o segundo conjunto. Este vector de teste terá por objectivo aplicar um **1** apenas aquelas ligações que, estando envolvidas num síndroma do tipo *aliasing*, produziram um VRS igual ao respectivo VTS. A todas as restantes ligações será aplicado um **0**.

	Primeiro conjunto		Segundo conjunto	
Ligação	VTS	VRS	VTS	VRS
0	0001	0101 <	0	0
1	0010	1011 <	0	0
2	0011	1011 <	0	0
3	0100	0101 <	0	0
4	0101	0101	1	1
5	0110	0110	0	0
6	0111	0111	0	0
7	1000	1011 <	0	0
8	1001	1011 <	0	0
9	1010	1010	0	0
10	1011	1011	0	0

Quadro 4.11: Algoritmo do vector único.

Os conjuntos de vectores gerados por este algoritmo estão ilustrados no quadro 4.11, onde estão também representadas as respostas esperadas, assumindo a existência de curto-circuitos independentes entre as ligações {0,3}, {1,8} e {2,7}.

Este algoritmo não permite resolver os síndromas do tipo *confounding*, pelo que é apresentado em (Jarwala et al., 89, p. 68) como equivalente ao algoritmo de Wagner. Repare-se que mais uma vez se assiste a uma solução de compromisso entre o número de vectores de teste e a complexidade da análise das respostas, para uma mesma capacidade de diagnóstico, e que neste caso permite resolver todos os síndromas do tipo *aliasing* com não mais de $\lceil \log_2(N+2) \rceil + 1$ vectores de teste.

4.4.2.3. Algoritmo dos C vectores óptimos (Jarwala et al., 89)

O primeiro conjunto de vectores é determinado pela atribuição a cada ligação de um VTS com $\lceil \log_2(N+2) \rceil$ bits, gerados através de uma sequência de contagem binária ascendente, com início no VTS **0...01**.

A existência de síndromas dos tipos *aliasing* ou *confounding* dará então lugar à geração do segundo conjunto de vectores, de acordo com o seguinte procedimento:

i) Se existirem apenas síndromas do tipo *aliasing*, terá lugar a geração de um único vector que constituirá o segundo conjunto, de acordo com o procedimento já

descrito para o algoritmo do vector único.

ii-a) Se os síndromas forem do tipo *confounding*, ou de ambos os tipos, e designando por C o máximo grau encontrado nos síndromas do tipo *confounding*, bastará a geração de C-1 vectores para resolver o síndroma de grau máximo. Estes C-1 vectores serão também usados para resolver todos os síndromas deste tipo, já que o diagnóstico de faltas com síndromas distintos pode ser realizado em paralelo.

ii-b) Finalmente, e se existirem ainda síndromas do tipo *aliasing*, terá lugar a geração de um último vector de teste, de acordo com o procedimento atrás descrito para o esclarecimento deste tipo de síndromas.

Os conjuntos de vectores gerados por este algoritmo estão ilustrados no quadro 4.12, onde estão também representadas as respostas esperadas, assumindo a existência de curto-circuitos independentes entre as ligações {0,9}, {1,8} e {2,7}.

	Primeiro conjunto		Segundo conjunto	
Ligação	VTS	VRS	VTS	VRS
0	0001	1011 <	10	10
1	0010	1011 <	01	01
2	0011	1011 <	00	00
3	0100	0100	00	00
4	0101	0101	00	00
5	0110	0110	00	00
6	0111	0111	00	00
7	1000	1011 <	00	00
8	1001	1011 <	00	01 <
9	1010	1011 <	00	10 <
10	1011	1011	00	00

Quadro 4.12: Algoritmo dos C vectores óptimos.

Repare-se que para cada síndroma do tipo *confounding*, de grau c, terão que ser considerados todos os subconjuntos de ligações que produzam o mesmo VRS, a partir dos quais se determinarão os que correspondem aos curto-circuitos independentes que verifiquem a condição de manter o próprio VRS do síndroma. Em particular, poderá também existir mais do que uma combinação de c subconjuntos de ligações nestas circunstâncias. Todas estas possibilidades terão que ser analisadas, para permitir a construção de um subconjunto com c ligações, que terá a propriedade de conter pelo menos

uma das ligações envolvidas em cada um dos subconjuntos considerados. A geração de um conjunto com c-1 vectores, que aplicarão sucessivamente um 1 a c-1 destas ligações, e um 0 a todas as restantes, permitirá então implementar a etapa ii-a), definida para este algoritmo.

A dificuldade principal deste algoritmo consiste em determinar este conjunto de c ligações para cada síndroma do tipo *confounding* presente, o que constitui um problema de cobertura de conjuntos, NP-completo (Winter et al., 90, p. 37 e pp. 51-53).

Este algoritmo constitui outro exemplo relativamente à necessidade de um equilíbrio entre o número de vectores de teste, e a complexidade requerida para a análise das respostas. Com efeito, e apesar de se garantir o esclarecimento de todos os síndromas dos tipos *aliasing* e *confounding*, com não mais de $\lceil \log_2(N+2) \rceil + C$ vectores de teste, a determinação do segundo conjunto de vectores envolve o procedimento mais complexo dos que foram até agora apresentados.

Refira-se por fim que, a exemplo do que acontece igualmente com o algoritmo de Goel e McMahon, a presença de faltas múltiplas numa mesma ligação pode produzir resultados insatisfatórios com este algoritmo.

4.4.2.4. Algoritmo do diagnóstico em dois passos (Cheng et al., 90)

Este algoritmo apresenta a propriedade de permitir que ambos os conjuntos de vectores de teste sejam gerados de início, sendo o segundo conjunto independente das respostas obtidas para o primeiro.

O primeiro conjunto de vectores é determinado pela aplicação a cada ligação de um VTS com $\lceil \log_2(N) \rceil$ bits, gerados por uma sequência de contagem binária ascendente, com início no VTS **0...0**. O segundo conjunto será constituído por $\lceil \log_2(\lceil \log_2(N) \rceil + 1) \rceil$ vectores, em que cada VTS representa o número de bits em **0**, no correspondente VTS do primeiro conjunto.

Os conjuntos de vectores gerados por este algoritmo estão ilustrados no quadro 4.13, onde estão também representadas as respostas esperadas, assumindo a existência de curto-circuitos independentes entre as ligações {2,5} e {3,4}.

	Primeiro conjunto		Segundo conjunto	
Ligação	VTS	VRS	VTS	VRS
0	0000	0000	100	100
1	0001	0001	011	011
2	0010	0111 <	011	011
3	0011	0111 <	010	011 <
4	0100	0111 <	011	011
5	0101	0111 <	010	011 <
6	0110	0110	010	010
7	0111	0111	001	001
8	1000	1000	011	011
9	1001	1001	010	010
10	1010	1010	010	010

Quadro 4.13: Algoritmo do diagnóstico em dois passos.

Este algoritmo garante a inexistência de síndromas do tipo *aliasing*. Com efeito, se um VTS **a** cobrir um VTS **b**, no primeiro conjunto, então **a** terá mais **1**s do que **b**. Consequentemente, no segundo conjunto, **b** representará um número maior do que **a**, pelo que a relação de cobertura deixará de existir. No entanto, a ocorrência de síndromas do tipo *confounding* continua a ser possível, como o ilustra o VRS produzido pelas ligações {2,3,4,5}, no quadro 4.13.

Este algoritmo é apresentado como idêntico ao algoritmo de Wagner (Cheng et al., 90, p. 568), em termos de possuírem a mesma capacidade de diagnóstico. Com efeito, também o algoritmo de Wagner poderia ser encarado como adaptativo, uma vez que a geração da segunda metade do conjunto de vectores poderá ter lugar apenas quando se pretender proceder a operações de diagnóstico (Wagner, 87, p. 56). Nestas circunstâncias, ambos os algoritmos apresentam a mesma capacidade de diagnóstico, e permitem que a geração do segundo conjunto de vectores ocorra logo de início, independentemente das respostas obtidas para o primeiro conjunto.

4.4.2.5. Algoritmo adaptativo A1 (Lien et al., 91a)

Do mesmo modo que o algoritmo do diagnóstico máximo já descrito, também o algoritmo adaptativo A1 faz uso de um modelo em que se admite a existência de faltas múltiplas numa mesma ligação. O procedimento para a geração do conjunto de vectores de

teste correspondente a este algoritmo pode ser descrito como se segue:

i) O primeiro conjunto de vectores (S_M) é gerado do mesmo modo que foi já anteriormente descrito para o algoritmo do auto-diagnóstico (Cheng et al., 90). Uma vez que todos os VTSs gerados apresentam o mesmo número de **0**s e de **1**s, o número de vectores de teste gerados para um circuito com N ligações será dado pelo mínimo valor de p que satisfizer a

$$\binom{p}{p/2} \geq N \qquad (4.16)$$

ii) O conjunto de ligações é dividido em dois grupos, colocando-se no grupo 0 todas as ligações que apresentem um VRS único, igual ao correspondente VTS. Todas as restantes ligações são incluídas no grupo 1.

iii) O segundo conjunto de vectores (S_F) é gerado pela atribuição sucessiva de um **1** a cada ligação pertencente ao grupo 1, enquanto às restantes se atribui um **0**. Às ligações pertencentes ao grupo 0 é sempre atribuído um **0**.

Os conjuntos de vectores gerados por este algoritmo estão ilustrados no quadro 4.14, onde estão também representadas as respostas esperadas, assumindo a existência de curto-circuitos independentes entre as ligações {1,5}, {2,6}, e {7,10}.

	Primeiro conjunto		Segundo conjunto	
Ligação	**VTS**	**VRS**	**VTS**	**VRS**
0	111000	111000	000000	000000
1	110100	111110 <	100000	101000 <
2	110010	111011 <	010000	010100 <
3	110001	110001	000000	000000
4	101100	101100	000000	000000
5	101010	111110 <	001000	101000 <
6	101001	111011 <	000100	010100 <
7	100110	111110 <	000010	000011 <
8	100101	100101	000000	000000
9	100011	100011	000000	000000
10	011100	111110 <	000001	000011 <

Quadro 4.14: Algoritmo adaptativo A1.

Este algoritmo é semelhante ao algoritmo de Goel e McMahon, diferindo essencialmente no que se refere ao procedimento empregue para gerar o primeiro conjunto

de vectores. No entanto, e tal como se ilustra no exemplo apresentado na figura 4.12† (Lien et al., 91a, p. 103), a existência de faltas múltiplas numa mesma ligação poderá produzir resultados insatisfatórios com o algoritmo de Goel e McMahon. Repare-se que o facto de as duas primeiras ligações ilustradas na figura 4.12 produzirem VRSs únicos, e iguais aos correspondentes VTSs, conduziria a que as faltas f_1, f_2, e f_3, não fossem detectadas. O emprego de um primeiro conjunto de vectores em que não existe uma relação de cobertura entre qualquer par de VTSs garante no entanto que o algoritmo adaptativo A1 mantem uma completa capacidade de detecção e diagnóstico, mesmo em presença de faltas múltiplas.

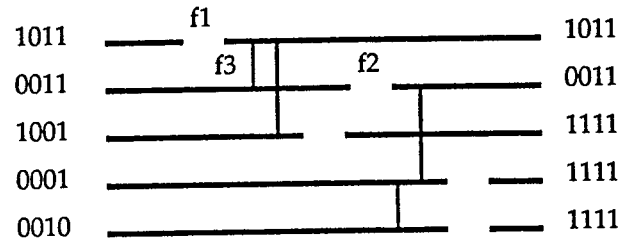


Fig.4.12: Exemplo para o qual o algoritmo de Goel e McMahon não produz resultados satisfatórios.

4.4.2.6. Algoritmo adaptativo A2 (Lien et al., 91a)

O algoritmo adaptativo A2 faz igualmente uso de um modelo em que se admite a existência de faltas múltiplas numa mesma ligação. O procedimento que gera o correspondente conjunto de vectores pode ser descrito como se segue:

i) O primeiro conjunto de vectores (S_M) é gerado de acordo com o procedimento descrito para o algoritmo do auto-diagnóstico (Cheng et al., 90). O número de vectores de teste gerados para um circuito com N ligações será dado pelo mínimo valor de p que satisfizer a

$$\binom{p}{p/2} \geq N \qquad (4.17)$$

ii) O conjunto de ligações é dividido em dois grupos, colocando-se no grupo 0 todas as ligações que apresentem um VRS único, igual ao correspondente VTS. Todas as restantes ligações são incluídas no grupo 1.

iii) O conjunto de ligações presentes no grupo 1 é dividido, de modo a agrupar as ligações que apresentem o mesmo VRS. Numerem-se estes grupos de 1 a G, e

† Admite-se que uma entrada no ar captura um valor lógico **1**.

designe-se por K a cardinalidade do maior destes grupos.

iv) A primeira parte (S_G) do segundo conjunto de vectores é gerada pela atribuição de um **1** a cada ligação presente nos G grupos formados no passo anterior, enquanto às restantes é atribuído um **0**. Uma vez que este processo decorre em paralelo para todos os grupos, K vectores serão gerados neste passo. Às ligações presentes no grupo 0 é sempre atribuído um **0**.

v) A segunda parte (S_K) do segundo conjunto de vectores é gerada pela atribuição sucessiva de um **1** a cada um dos G grupos formados no passo ii) (a todas as ligações dentro de cada grupo é sempre atribuído o mesmo valor). Às ligações presentes no grupo 0 é sempre atribuído um **0**. G vectores serão gerados neste passo.

Os conjuntos de vectores gerados por este algoritmo estão ilustrados no quadro 4.15, onde estão também representadas as respostas esperadas, assumindo a existência de curto-circuitos independentes entre as ligações {1,5}, {2,6}, e {7,10}.

	Primeiro conjunto		Segundo conjunto			
Ligação	VTS	VRS	VTS		VRS	
0	111000	111000	00	0000	00	0000
1	110100	111110 <	10	1000	10	1100 <
2	110010	111011 <	01	1000	01	1100 <
3	110001	110001	00	0000	00	0000
4	101100	101100	00	0000	00	0000
5	101010	111110 <	10	0100	10	1100 <
6	101001	111011 <	01	0100	01	1100 <
7	100110	111110 <	10	0010	10	0011 <
8	100101	100101	00	0000	00	0000
9	100011	100011	00	0000	00	0000
10	011100	111110 <	10	0001	10	0011 <

Quadro 4.15: Algoritmo adaptativo A2 †.

Apesar de o número total de vectores gerados, para o exemplo escolhido (foi usado o mesmo exemplo já apresentado no quadro 4.14), ser o mesmo que é produzido pelo

† Para o segundo conjunto de vectores, a primeira metade está representada à direita, e a segunda metade à esquerda.

algoritmo adaptativo A1, o número de vectores gerados pelo algoritmo adaptativo A2 será em geral inferior. Este algoritmo mantem uma completa capacidade de detecção e diagnóstico, mesmo em presença de faltas múltiplas. Repare-se que a segunda metade do segundo conjunto de vectores gerados tem por objectivo detectar a presença de faltas entre ligações pertencentes a grupos diferentes, e que pudessem ter sido mascaradas pela existência de outras faltas nas mesmas ligações (veja-se a este propósito o exemplo ilustrado na figura 4.11 (c)).

4.4.3. Síntese das características principais

Esta secção apresenta dois quadros em que se sintetizam as características principais dos algoritmos descritos, com o objectivo de facilitar uma comparação em termos globais. Os algoritmos de um passo só estão descritos no quadro 4.16, e os algoritmos do tipo adaptativo no quadro 4.17, sendo nestes dois quadros feita referência ao seguinte conjunto de notas:

Nota 1: $\lceil \log_2(N+2) \rceil \leq p \leq N$

Nota 2: p é determinado em função do máximo número de ligações que podem estar curto-circuitadas entre si (E), e do número de ligações existentes (N).

Nota 3: Valores aproximados, assumindo a existência de 500 ligações, 1.000 células BST, e um relógio de teste (TCK) de 1 MHz.

Nota 4: Assumindo que o máximo número de ligações que podem estar em curto-circuito entre si (E) é de 20.

Nota 5: W corresponde ao número de ligações que produziram VRSs com defeito, i.e. VRSs diferentes dos respectivos VTSs.

Nota 6: C representa o máximo número de curto-circuitos independentes que podem ter lugar em qualquer dos síndromas do tipo *confounding* presentes, i.e. o máximo grau encontrado nos síndromas deste tipo.

Nota 7: G representa o número de grupos de ligações que apresentam o mesmo VRS, e K a cardinalidade do maior destes grupos.

Algoritmo	Partição binária (Kautz 74)	Wagner (Wagner 87)	Sequência caminhante (Hassan et al. 88)	Peso mínimo (Yau et al. 89)	Independência máxima (Yau et al. 89)	Auto-diagnóstico (Cheng et al. 90)	Diagnóstico global (Cheng et al. 90)	Diagnóstico máximo (Lien et al. 91a)
Número de vectores	$\lceil \log_2(N) \rceil$	$2\lceil \log_2(N+2) \rceil$	N	p (nota 1)	p=f(E,N) (nota 2)	Mínimo p que satisfizer a $\binom{p}{p/2} \geq N$	p+1, onde p assume o valor mínimo que satisfizer a $2+\sum_{i=1}^{p}\binom{i}{i/2} \geq N$	Depende da relação de vizinhanças para o circuito.
Tempo de aplicação (nota 3)	9 ms	18 ms	500 ms	p ms	24 ms (nota 4)	12 ms	11 ms	Depende da relação de vizinhanças para o circuito.
Capacidade de diagnóstico	Pobre. Possível ocorrência de *aliasing* e *confounding*	*Aliasing* inexistente. *Confounding* pode ter lugar.	Completa.	Ambiguidade tanto menor quanto maior for o valor admitido para p.	Completa.	*Aliasing* inexistente. *Confounding* pode ter lugar.	*Aliasing* inexistente. *Confounding* pode ter lugar.	Completa, mesmo em presença de faltas múltiplas na mesma ligação.
Comentários	Gera o menor número possível de vectores.	Todas as ligações com defeito produzirão um VRS diferente do seu VTS.	Gera o número máximo de vectores.	A atribuição de VTSs às ligações procura minimizar a ocorrência de *aliasing* e de *confounding* .	Requer a disponibilidade de informações adicionais (vizinhança de ligações, características do processo).	Todas as ligações com defeito produzirão um VRS diferente do seu VTS.	Ligações com defeito podem produzir VRSs iguais aos respectivos VTSs.	Requer a extracção da relação de vizinhança entre ligações.

Quadro 4.16: Detecção e diagnóstico de curto-circuitos: Algoritmos de um passo só.

Algoritmo	Goel e McMahon (Goel et al. 82)	Vector único (Jarwala et al. 89)	C vectores óptimos (Jarwala et al. 89)	Diagnóstico em dois passos (Cheng at al. 90)	Algoritmo adptativo A1 (Lien et al., 91a)	Algoritmo adptativo A2 (Lien et al., 91a)
Número de vectores	$\lceil \log_2(N+2) \rceil + W$ (nota 5)	$\lceil \log_2(N+2) \rceil + 1$	$\lceil \log_2(N+2) \rceil + C$ (nota 6)	$\lceil \log_2(N) \rceil + \lceil \log_2(\lceil \log_2(N) \rceil + 1) \rceil$	$\binom{p}{p/2} + W$ onde p assume o mínimo valor que satisfizer a $\binom{p}{p/2} \geq N$. (nota 5)	$\binom{p}{p/2} + G + K$ onde p assume o mínimo valor que satisfizer a $\binom{p}{p/2} \geq N$. (nota 7)
Tempo de aplicação (nota 3)	(9 + W) ms	10 ms	(9 + C) ms	13 ms	(12 + W) ms	(12 + G + K) ms
Capacidade de diagnóstico	Completa.	Síndromas do tipo *confounding* ficam por resolver.	Completa.	Síndromas do tipo *confounding* ficam por resolver.	Completa, mesmo em presença de faltas múltiplas.	Completa, mesmo em presença de faltas múltiplas.
Comentários	Todas as ligações com VRSs diferentes dos respectivos VTSs são testadas individualmente.	Só um VTS é gerado no segundo conjunto. Reduzida análise de respostas.	Análise de respostas muito complexa.	Segundo conjunto de vectores é independente da análise das respostas.	O primeiro conjunto de vectores garante que a presença de faltas múltiplas não impedirá uma completa capacidade de detecção e diagnóstico.	O segundo conjunto de vectores testa em paralelo os grupos de ligações com o mesmo VRS, procedendo então a um teste entre ligações de diferentes grupos.

Quadro 4.17: Detecção e diagnóstico de curto-circuitos: Algoritmos adaptativos.

4.4.4. Análise comparativa

A diversidade de algoritmos apresentados sugere a necessidade de uma reflexão relativamente à aplicabilidade de cada um, e ao significado das suas diferenças principais.

A aplicabilidade de um algoritmo é essencialmente condicionada pelas dificuldades de implementação que apresente, e que são especialmente relevantes para os algoritmos que requerem informações adicionais, tais como as relações de vizinhança entre ligações. Também os algoritmos adaptativos podem encontrar dificuldades de implementação, tornando-se necessária a definição de um protocolo que permita aos módulos responsáveis pela análise das respostas a especificação do novo conjunto de vectores a gerar. Repare-se ainda que os algoritmos adaptativos terão apenas utilidade quando for efectivamente necessária uma maior resolução no diagnóstico, o que não terá lugar em todos os cenários de teste.

A maior dificuldade inerente a um algoritmo adaptativo, face a um algoritmo de um passo só, tem por contrapartida um menor número total de vectores, que se traduz por um teste mais rápido. Torna-se no entanto necessário avaliar de forma concreta até que ponto é que esta vantagem não é mais aparente do que real, uma vez que o ganho em tempo de execução pode não compensar a maior complexidade envolvida. Com efeito, e apesar de os elementos apresentados nos quadros 4.16 e 4.17 apontarem para que um algoritmo de um passo só possa requerer um tempo de aplicação significativamente superior ao que corresponde a um algoritmo adaptativo, importa igualmente considerar o ganho efectivo relativamente à duração total do teste (incluindo todas as restantes etapas, e nomeadamente o teste de componentes). Repare-se a este propósito que quer o teste de grupos de componentes não BST (e das respectivas ligações na CCI), quer o teste de componentes BST (com ou sem auto-teste interno), poderão exceder largamente o tempo correspondente ao teste das ligações BST, caso em que a utilização de um algoritmo de um passo só se tornaria naturalmente preferível.

Por outro lado, e dependendo da resolução de diagnóstico proporcionada, os algoritmos de um passo só apresentam uma larga diversidade no número de vectores gerados, no tempo de aplicação, e na capacidade de memória necessária para armazenamento dos vectores. Os elementos apresentados nos quadros 4.18 a 4.20 ilustram a variação destes parâmetros com o número de ligações, assumindo para cada ligação uma média de 3 células BST, e um relógio de teste (TCK) com frequência de 1 MHz.

	Número de vectores gerados por cada algoritmo					
Número de ligações	Partição binária	Wagner	Sequência caminhante	Auto diagnóstico	Diagnóstico global	Diagnóstico em 2 passos
100	7	14	100	9	9	10
200	8	16	200	10	10	12
300	9	18	300	11	11	13
400	9	18	400	11	11	13
500	9	18	500	12	11	13

Quadro 4.18: Número de vectores x Número de ligações, para vários algoritmos de um passo só.

Os elementos apresentados nestes quadros não incluem os algoritmos que requerem informações adicionais sobre o circuito (algoritmos da independência máxima, e do diagnóstico máximo), uma vez que o número de vectores gerados depende de parâmetros adicionais. Também o algoritmo do peso mínimo não está representado nos quadros 4.18 a 4.20, uma vez que os valores correspondentes não são directamente dependentes do número de ligações. O número de vectores gerados por este algoritmo pode ser especificado pelo utilizador, dentro dos limites impostos pela equação 4.11, pelo que os valores correspondentes, nos quadros 4.18 a 4.20, serão limitados pelos que são apresentados para os algoritmos da partição binária, e da sequência caminhante.

	Tempos de aplicação por cada algoritmo [ms]					
Número de ligações	Partição binária	Wagner	Sequência caminhante	Auto diagnóstico	Diagnóstico global	Diagnóstico em 2 passos
100	2,1	4,2	30	2,7	2,7	3
200	4,8	9,6	120	6	6	7,2
300	8,1	16,2	270	9,9	9,9	11,7
400	10,8	21,6	480	13,2	13,2	15,6
500	13,5	27	750	18	16,5	19,5

Quadro 4.19: Tempos de aplicação x Número de ligações, para vários algoritmos de um passo só.

Os elementos apresentados no quadro 4.19 sugerem que, com excepção do algoritmo da sequência caminhante, a escolha de qualquer um dos restantes algoritmos não produzirá diferenças significativas na duração total do teste. Mesmo no que respeita ao algoritmo da sequência caminhante, a importância que a sua duração terá sobre a duração total do teste dependerá essencialmente da dimensão e complexidade dos grupos de componentes não

BST presentes, e dos teste a realizar relativamente aos componentes BST.

	Capacidade de memória por cada algoritmo [Kbytes]					
Número de ligações	Partição binária	Wagner	Sequência caminhante	Auto diagnóstico	Diagnóstico global	Diagnóstico em 2 passos
100	2,1	4,1	29,3	2,6	2,6	2,9
200	4,7	9,4	117,2	5,9	5,9	7
300	7,9	15,8	263,7	9,7	9,7	11,4
400	10,5	21,1	468,8	12,9	12,9	15,2
500	13,2	26,4	732,4	17,6	16,1	19

Quadro 4.20: Capacidade de memória x Número de ligações, para vários algoritmos de um passo só.

A comparação dos algoritmos referidos, no que respeita à capacidade de memória necessária para armazenar os vectores de teste gerados, pode por sua vez constituir um factor de maior importância na selecção de um algoritmo, sobretudo nos casos em que estes vectores devam estar armazenados na própria CCI a testar. A importância deste factor é reforçada pela necessidade de se indicarem as respostas esperadas, e máscaras de comparação, para cada vector (o que triplica a capacidade de memória indicada para cada caso, no quadro 4.20). O algoritmo da sequência caminhante não seria naturalmente a melhor opção para a inclusão de funções de auto-teste em CCIs com BST, sendo nestas circunstâncias preferível trocar uma maior resolução no diagnóstico por menores requisitos em capacidade de memória local. O algoritmo do auto-diagnóstico constituirá neste caso uma boa escolha, cujo interesse é reforçado pelo facto de o conjunto de vectores gerados não permitir que a existência de faltas múltiplas impeça uma completa capacidade de detecção e diagnóstico (Lien et al., 91a).

Os algoritmos adaptativos apresentados distinguem-se essencialmente no que respeita à solução de compromisso adoptada entre o número de vectores de teste gerados para o segundo conjunto, e a complexidade requerida para a análise das respostas. O algoritmo de Goel e McMahon, e o algoritmo adaptativo A1, requerem apenas que a análise de respostas identifique as ligações que apresentaram VRSs com defeito, enquanto o algoritmo dos C vectores óptimos requer a mais complexa das análises de respostas. Uma situação intermédia consistiria em restringir a análise de respostas à identificação das ligações que partilhem VRSs comuns. Os grupos de ligações nestas circunstâncias poderão ser testados em paralelo na segunda fase, o que permitirá um teste com menor duração (Ferreira et al., 91a). O algoritmo adaptativo A2 adopta uma solução idêntica, mas a consideração de um

modelo de faltas múltiplas confere-lhe melhores características de detecção e diagnóstico.

Uma vez analisadas as características principais dos algoritmos para a detecção e diagnóstico de faltas nas ligações BST de CCIs, estão reunidas todas as condições para a apresentação do tema principal deste trabalho, que consiste na especificação de um controlador de teste que permita uma exploração eficiente da infraestrutura BST disponível na CCI, e na geração automática do respectivo programa de teste.

Capítulo 5

Um controlador residente para CCIs com BST:

Arquitectura, e geração automática do programa de teste

5. UM CONTROLADOR RESIDENTE PARA CCIS COM BST: ARQUITECTURA, E GERAÇÃO AUTOMÁTICA DO PROGRAMA DE TESTE

Este capítulo discute os requisitos a que deve obedecer um controlador residente, destinado a permitir a execução do auto-teste em CCIs com BST, ou a apoiar a realização do teste, em colaboração com um equipamento exterior. A funcionalidade que um tal controlador deve proporcionar é apresentada, e discutem-se os aspectos de maior relevo na geração automática do programa de teste (GAPT), o qual deverá ser especificado por recurso ao conjunto de instruções suportado por este controlador. A apresentação destes requisitos é precedida pela discussão das questões principais que se colocam pela existência de um controlador residente, e que pretende clarificar a aplicabilidade desta solução, e caracterizar os resultados que ela proporciona. As características deste controlador serão então apresentadas, com o objectivo principal de identificar o conjunto de operações elementares que deverão estar disponíveis, e que permitirão implementar as etapas correspondentes ao protocolo de teste para CCIs com BST. A caracterização da GAPT será apresentada a seguir, com relevo para a identificação do fluxo de dados para cada etapa do protocolo de teste, e para a identificação dos procedimentos principais que deverão estar presentes. Este capítulo encerra com a discussão da influência que esta solução tem sobre o protocolo de teste para CCIs com BST, de que resulta uma extensão para este protocolo, a empregar quando um controlador residente está disponível.

Este capítulo efectua a apresentação global do trabalho descrito nesta tese, pelo que a sua leitura deverá necessariamente ter lugar. Os conceitos aqui apresentados serão aprofundados nos capítulos 6 (arquitectura do controlador) e 7 (geração automática do programa de teste). Uma leitura mais rápida deste trabalho poderá deste modo ter lugar por passagem directa do capítulo 5 para o capítulo 8 (conclusão), circunstância em que a compreensão da solução proposta se resume aos aspectos principais descritos neste capítulo.

5.1. BENEFÍCIOS E REQUISITOS

A existência de um controlador residente, com o objectivo de permitir uma melhor exploração da infraestrutura de testabilidade incluída numa CCI, é uma estratégia que tem ganho aceitação crescente, quer através de trabalhos realizados em universidades e outras

instituições de I&D (Budde, 88), (Wang et al., 89), (Lien et al., 89), quer através de projectos levados a cabo em vários sectores da indústria electrónica (Avra, 87), (Dervisoglu, 90), (Jarwala et al., 91b).

A disponibilidade de componentes destinados a melhorar os índices de C&O em CCIs não é novidade (Turino, 84, 89). Contudo, a ausência de uma infraestrutura de testabilidade residente nos próprios componentes da CCI contribuiu para restringir o seu alcance, quer pelo custo adicional (proporcional ao número de pontos de teste a suportar), quer pela insuficiência inerente a uma estratégia que só permitia aceder a uma percentagem reduzida dos nós existentes.

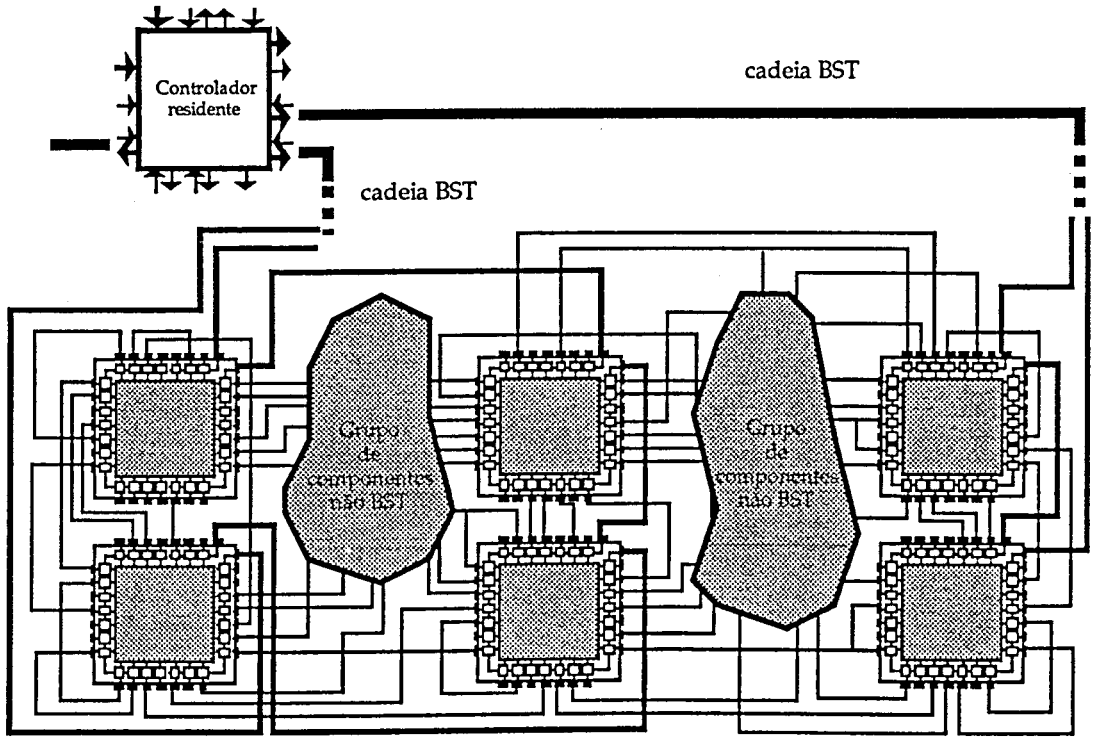


Fig.5.1: Controlador residente numa CCI com BST.

A aprovação do BST como norma do IEEE (IEEE Std 1149.1, 90) veio modificar radicalmente esta situação, uma vez que foi especificada uma infraestrutura de testabilidade generalizadamente aceite, e com um custo adicional que é minimizado pelo facto de a lógica de teste estar incluída nos próprios componentes. A esta vantagem deve ainda acrescentar-se o facto de a infraestrutura BST permitir o acesso a funções de auto-teste incorporadas nos componentes mais complexos (Maierhofer, 90), (Riessen et al., 91), o que lhe confere capacidades superiores àquelas que a frequente analogia com uma *matriz-de-agulhas electrónica* permitiria inferir. As potencialidades proporcionadas pela

infraestrutura BST contribuem para estimular o interesse pela existência de um controlador residente em CCIs que disponham desta tecnologia de teste, tal como se ilustra na figura 5.1, pelo que se justifica uma análise dos benefícios e requisitos inerentes a esta opção.

Uma das razões principais que justificam o interesse pela existência de um controlador residente consiste em ocultar o conjunto de operações elementares que são necessárias para o teste de uma CCI através da sua infraestrutura BST, relativamente ao controlador do sistema, ou a um equipamento de teste exterior (Yau et al., 90, p. 311), (Dervisoglu, 90, pp. 582 e 590), (Jarwala et al., 91a, p. 1). Esta possibilidade é tanto mais importante quanto maior for o número de CCIs existentes, uma vez que o controlador do sistema passará deste modo a dispor de uma visão homogénea de todas as CCIs, em que o acesso à infraestrutura de teste será sempre efectuado da mesma forma (por interpelação do controlador local), permitindo que a realização de testes decorra simultaneamente em várias CCIs. Esta estratégia conduz naturalmente a uma estrutura hierárquica, tal como se ilustra na figura 5.2, em que o controlador residente em cada CCI se responsabiliza por realizar os testes, e reportar os resultados, de acordo com os comandos emitidos pelo controlador do sistema.

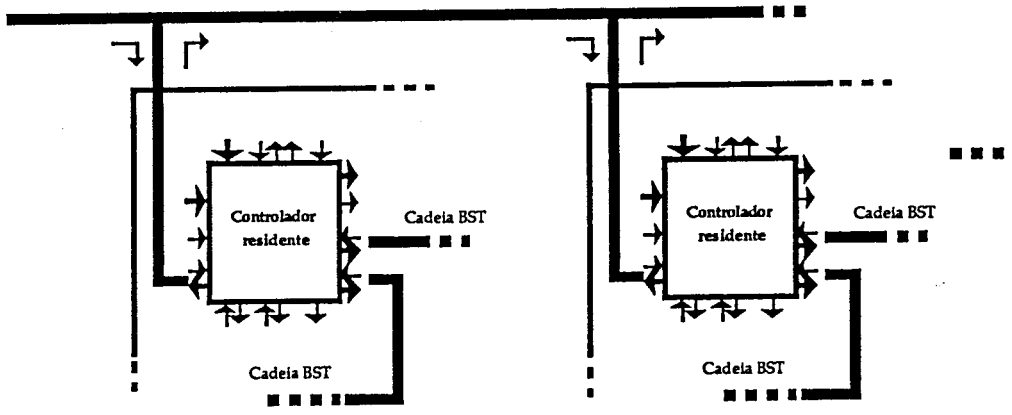


Fig.5.2: Configuração hierárquica para o teste de CCIs com controladores residentes.

Uma outra razão que suporta o interesse pela existência de um controlador residente consiste no facto de a sua aplicabilidade se estender aos vários cenários de teste referidos no capítulo 2 (desenvolvimento e verificação de protótipos, produção, manutenção). Com efeito, a existência de um controlador que se ocupe de todos os detalhes relativos à realização de testes através da infraestrutura BST permitirá que o equipamento de teste exterior passe a ter por funções principais a aplicação de vectores e captura de respostas, relativamente ao conjunto de nós com acesso paralelo, de acordo com um protocolo de sincronismo estabelecido com o controlador residente. Esta possibilidade permite reduzir

significativamente os requisitos exigidos ao equipamento de teste exterior, relativamente a aplicações nos domínios do desenvolvimento e verificação de protótipos, e no teste de produção. No que respeita a operações de manutenção, deve referir-se que a existência de um controlador residente é compatível com a tendência crescente que se verifica no sentido de uma descentralização das funções de diagnóstico. A localização da CCI com defeito poderá nestas circunstâncias ser realizada através de um primeiro teste de curta duração, que produzirá o resultado do tipo sim/não pretendido. Repare-se que será normalmente necessário realizar testes mais pormenorizados posteriormente à detecção das faltas, com o objectivo de proceder ao respectivo diagnóstico. A possibilidade de continuar a utilizar os recursos do controlador residente poderá no entanto manter todo o seu interesse, desde que estejam disponíveis ferramentas computacionais que permitam a realização dos novos testes por seu intermédio, continuando deste modo a ocultar os detalhes de operação ao mais baixo nível, se tal for desejado.

O apoio computacional à geração do programa de teste, referido no parágrafo anterior, assume uma importância de primeiro plano, e constitui um dos requisitos principais para viabilizar a utilização de controladores residentes em CCIs com BST. Com efeito, e considerando que uma das razões mais importantes para justificar o interesse por esta opção consiste em ocultar as operações elementares necessárias para controlar a infraestrutura BST, a existência de ferramentas para a GAPT assume uma importância fundamental para se concretizarem os benefícios pretendidos.

O conjunto de programas que permitem a GAPT apresentará um fluxo de dados como se ilustra na figura 5.3, em que a informação de entrada se divide em três tipos principais:

- Descrição das ligações na CCI (*netlist*).
- Descrição da infraestrutura BST existente nos componentes. Esta descrição incluirá elementos relativos ao teste destes componentes, quando disponíveis.
- Descrição dos grupos de componentes não BST existentes. Para além da descrição relativa ao acesso a estes grupos, através da infraestrutura BST disponível, deverá igualmente estar presente a descrição dos vectores necessários ao respectivo teste.

Repare-se que os conjuntos de vectores de teste para os grupos de componentes não BST serão gerados exteriormente, como se ilustra na figura 5.3. Estes vectores de teste deverão ser integrados na descrição do respectivo grupo de componentes, que será aceite pela GAPT.

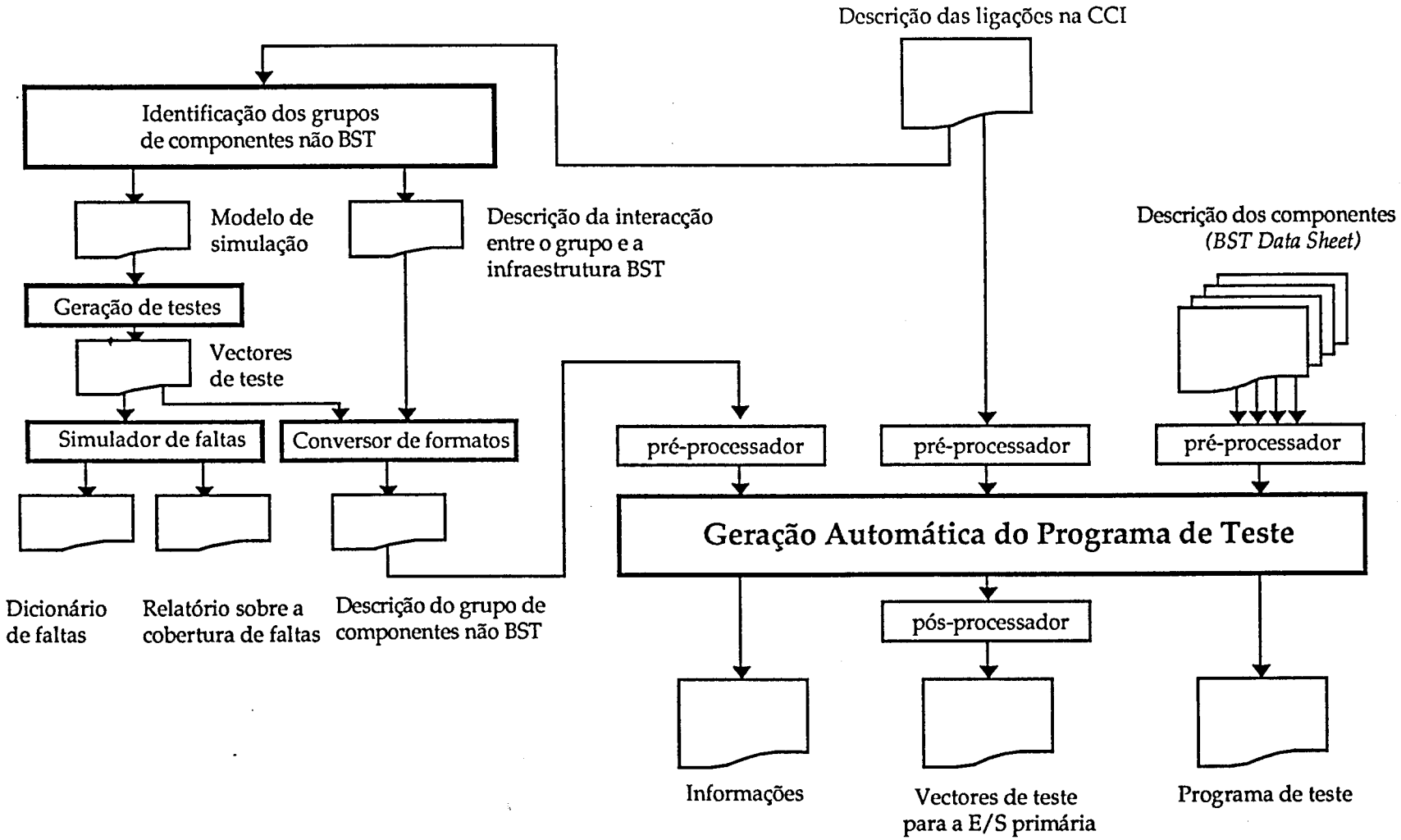


Fig.5.3: Fluxo de dados na GAPT.

O programa de teste gerado deverá implementar todas as etapas necessárias para a realização do teste da CCI, utilizando para o efeito o conjunto de instruções suportadas pelo controlador residente. Para os casos em que essa solução se torne economicamente viável, a geração automática do programa de teste poderá produzir o ficheiro de especificação para a memória (ROM) a incluir no próprio controlador, de acordo com o formato suportado pelo ambiente de projecto utilizado. A implementação de uma solução para o auto-teste da CCI, baseada num único componente, tornar-se-á então possível nestas circunstâncias (Ferreira et al., 92a). A viabilidade desta alternativa dependerá de factores como o volume de produção estimado para a CCI em questão, o número e a complexidade dos CIs presentes, e também a taxa de cobertura de faltas que a infraestrutura BST existente permitir.

Um volume de produção elevado constitui um requisito no plano económico, sobretudo para justificar a opção de se incluir a memória com o programa de teste no próprio controlador. Com efeito, se o programa de teste for armazenado numa memória exterior, o mesmo controlador poderá ser utilizado em CCIs diferentes, o que lhe permitirá uma redução de custos por aumento do volume de fabrico. Repare-se porém que este factor não é determinante, já que a inclusão de um controlador residente se pode justificar em CCIs que, apesar de não serem produzidas em grandes volumes, apresentem características especiais no que respeita à complexidade dos componentes presentes, e a questões de fiabilidade. Os sistemas desenvolvidos para fins militares constituem um exemplo em que estas circunstâncias estão presentes (Avra, 87), e onde a importância assumida pelo factor custo é inferior.

A taxa de cobertura de faltas possível através da infraestrutura BST constitui também um factor de importância a considerar (Lefebvre, 90, pp. 295 e 303), tendo em vista que a inclusão de um controlador residente inclui o objectivo de minimizar os recursos necessários em equipamentos de teste exteriores. A existência de uma percentagem tão elevada quanto possível de componentes com BST deverá portanto ser considerada como um dos requisitos necessários para justificar a inclusão de um controlador residente, sobretudo atendendo a que a introdução gradual do BST dará numa primeira fase lugar a CCIs que apresentarão apenas uma implementação parcial desta tecnologia de teste (Hansen, 91, p. 394), (Bagge et al., 89, pp. 17 e 18).

Uma vez discutidos os principais aspectos relacionados com a inclusão de um controlador residente em CCIs com BST, terá lugar na secção seguinte uma apresentação de requisitos, em termos das características pretendidas, a partir da qual se tornará possível a especificação de uma arquitectura que os satisfaça, a apresentar no capítulo 6.

5.2. ESPECIFICAÇÃO DE CARACTERÍSTICAS PARA UM CONTROLADOR RESIDENTE

As considerações apresentadas na secção anterior permitiram concluir que os objectivos a atingir consistem fundamentalmente em explorar todas as potencialidades proporcionadas pela infraestrutura BST para o teste de CCIs, ao mesmo tempo que se pretende proporcionar uma interacção eficiente com um controlador existente a um nível hierárquico superior (subsistema, sistema). Esta secção apresentará uma especificação para o conjunto de características de um controlador residente em CCIs com BST, tendo em vista que se pretendem optimizar os objectivos referidos.

5.2.1. Descrição da funcionalidade pretendida

A exploração das potencialidades proporcionadas pelo BST deverá ser assegurada por um conjunto de instruções que permita implementar directamente as operações elementares necessárias para a realização do teste através desta infraestrutura, e cuja identificação terá lugar na secção seguinte. As características de funcionalidade que serão apresentadas nesta secção contribuem igualmente para a especificação deste conjunto de instruções, pelo que a sua apresentação será efectuada em primeiro lugar.

Um primeiro factor a ser levado em conta, no que respeita a concretizar as potencialidades proporcionadas pela infraestrutura BST, consiste em prever a possibilidade de existir mais do que uma cadeia BST na CCI a testar, e cujo interesse se deve sobretudo à necessidade de limitar superiormente o número de células (comprimento da cadeia). Esta necessidade assenta em razões de carácter prático, tais como limitar o número máximo de entradas ligadas ao pino que comanda as linhas TMS e TCK (considerações de *fan-out*), ou facilitar a definição dos percursos físicos para estas ligações (problemas de *routing*). Também a duração do teste será inferior se existirem duas cadeias BST, uma vez que o tempo necessário ao deslocamento dos vectores é proporcional ao comprimento do percurso série. Este argumento deverá no entanto ser confrontado com a acrescida complexidade para o controlo da infraestrutura BST, e levada em conta a possibilidade de esta medida ter um reduzido impacto sobre o tempo total do teste (por exemplo, se este for dominado pelo teste de grupos de componentes não BST, efectuado por um equipamento exterior). Por fim, a dimensão e a complexidade estimadas para CCIs com BST (projectos de média complexidade, formato duplo *Eurocard*, 223x160 mm^2) permite prever que raramente se justificaria a existência de mais do que duas cadeias (Bagge et al., 89, p. 18).

O controlador residente deverá deste modo suportar a existência de uma ou duas

cadeias BST. O controlo das duas cadeias será efectuado alternadamente (por multiplexagem dos recursos internos), mas deverá estar disponível um TAP independente para cada cadeia.

Um outro factor importante em termos de funcionalidade consiste em permitir o acesso exterior à(s) cadeia(s) BST da CCI a testar, o que pode justificar-se para a realização de testes específicos, não incluídos no programa executado pelo controlador residente. A disponibilização desta infraestrutura pode no entanto ter também interesse para permitir a utilização de sistemas de desenvolvimento, ou de analisadores lógicos, que suportem esta tecnologia. Uma das características requeridas consistirá deste modo em permitir a colocação no estado de alta impedância do conjunto de pinos usados para comandar a(s) cadeia(s) BST da CCI.

O teste de uma CCI com BST deverá naturalmente abranger o conjunto de ligações associadas aos nós com acesso paralelo, sejam os que constituem as entradas e saídas primárias da CCI a testar, ou os que foram explicitamente previstos para permitir a C&O de blocos não BST. O comando destes nós terá que ser feito a partir de um equipamento de teste exterior, pelo que se torna necessário prever a existência de um protocolo de sincronismo entre este equipamento e o controlador residente, no sentido de permitir a aplicação de estímulos e captura de respostas. Este sincronismo pode ser garantido através de um protocolo assíncrono que envolve duas linhas (Driessen et al., 90, p. 12), tal como se ilustra na figura 5.4.

A indicação para o exterior do resultado do teste, através de uma saída do tipo sim / não, é também uma das características requeridas para um controlador residente, e que tem sobretudo interesse no que respeita às situações em que se pretenda apenas identificar quais as CCIs com defeito. Repare-se que esta saída não invalida o interesse de se conservar internamente uma informação que permita caracterizar melhor o tipo de falta(s) encontrada(s), e que seja acessível a um nível hierárquico superior. Esta informação deverá permitir a distinção entre várias faltas possíveis (falta na infraestrutura BST, falta de continuidade, falta no resultado do auto-teste de componentes, etc.), e a identificação da cadeia BST em que cada uma destas faltas foi detectada. Isto significa que os resultados do teste deverão ser de dois tipos, sendo fornecida para o exterior apenas uma indicação do tipo ausência ou presença de faltas, mas possibilitando-se a memorização interna dos tipos de faltas detectadas, e da cadeia BST em que a sua detecção teve lugar.

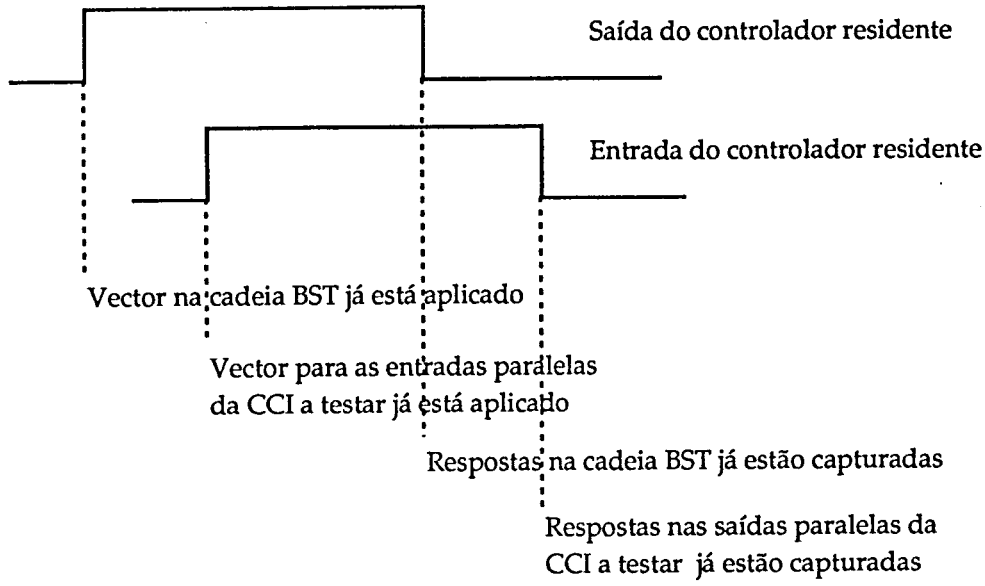


Fig.5.4: Protocolo de sincronismo com um equipamento de teste exterior.

A interacção com um nível hierárquico superior assume uma importância de relevo na caracterização da funcionalidade pretendida para um controlador residente, embora não exista ainda um consenso quanto ao modo como esta interacção se deva processar. Existe com efeito alguma dispersão nos trabalhos publicados, que incluem soluções *ad-hoc* para a ligação a equipamentos com características específicas (Vining, 89), (Jarwala et al., 91a), (Dervisoglu, 91), (Brown, 91), e soluções baseadas em barramentos de teste com um alcance mais genérico (Avra, 87), (Breuer et al., 88), incluindo a extensão do BST ao nível da interligação entre CCIs (Siemens, 90a), (Bhavsar, 91b). Esta indefinição relaciona-se com a inexistência, à data, de uma especificação generalizadamente aceite para um barramento de teste ao nível do sistema/subsistema, e que será ultrapassada pela aprovação pelo IEEE do barramento MTM (*Module Test and Maintenance Bus*), cujo desenvolvimento se encontra actualmente em fase final (IEEE P1149.5 Std, 92).

A proposta P1149.5 (MTM) descreve uma solução para um barramento de teste série, destinado ao nível do sistema/subsistema, que pode ser empregue em configurações hierárquicas com outros barramentos de teste, suportados ou não por uma norma internacional. Este barramento usa uma topologia do tipo *multi-drop*, tal como se ilustra na figura 5.5, em que um mestre pode comunicar com até 250 escravos, através de um conjunto de quatro ou cinco linhas (o sinal MPR é opcional). O mestre comunica com os escravos através de mensagens que consistem na transferência de pacotes, contendo cada mensagem um pacote de cabeçalho, um pacote de reconhecimento opcional, e um número variável de pacotes de dados.

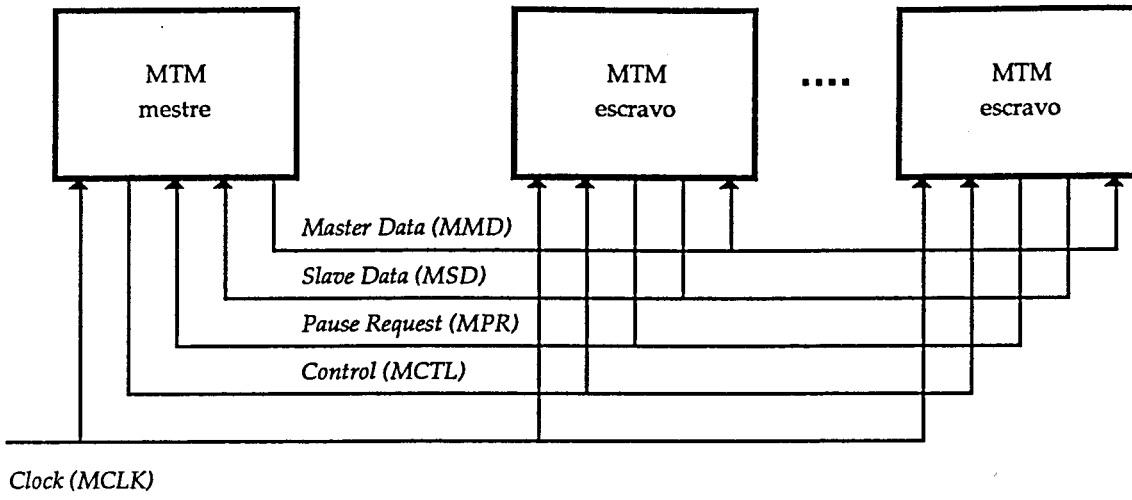


Fig.5.5: O barramento de teste MTM (IEEE P1149.5 Std, 92).

Repare-se porém que a aprovação deste barramento pelo IEEE não significa que esta passe a ser necessariamente a escolha para um barramento de teste entre CCIs. Por outro lado, a inclusão de um controlador residente torna possível a extensão do BST como barramento de teste a um nível hierárquico superior, desde que este controlador disponha de uma infraestrutura BST. Nestas circunstâncias, um controlador a nível do sistema verá apenas a infraestrutura BST de cada controlador residente nas CCIs, desaparecendo o inconveniente principal que consistia no tamanho resultante da concatenação das cadeias BST em cada uma. A execução do programa de teste relativo a cada CCI terá então lugar por ordem do controlador do (sub)sistema, que a enviará como um comando para o registo de instrução da infraestrutura BST do respectivo controlador residente. Esta solução tem o interesse adicional de tornar possível uma configuração hierárquica em que o mesmo controlador é usado em diferentes níveis, tal como se ilustra na figura 5.6. Repare-se que esta solução apresenta o inconveniente de a ausência de uma CCI interromper o barramento de teste, pelo que deverá estar prevista uma solução que restabeleça a continuidade física das ligações interrompidas, nestas circunstâncias.

Esta solução é especialmente viável para um número reduzido de CCIs, em que não seja necessária a maior sofisticação proporcionada pelo barramento MTM. As seguintes razões podem ser consideradas em favor da solução ilustrada na figura 5.6:

i) A extensão do BST para barramento de teste ao nível do (sub)sistema resulta numa configuração hierárquica que usa apenas uma norma de testabilidade, o que proporciona uma solução mais simples.

ii) Esta solução permite que o mesmo controlador seja empregue em diferentes níveis hierárquicos.

iii) A inclusão de uma infraestrutura BST num CI é um processo simples, que é ainda mais facilitado pelo número de fabricantes de ASICs que proporcionam já bibliotecas de células para este fim (Whetsel, 88), (Muris, 89), (Plessey, 90). Aliás, existem já ferramentas computacionais destinadas a automatizar este processo, a partir da descrição das características pretendidas para esta infraestrutura (Eerenstein, 89), (Muris, 90), (Chiles et al., 91).

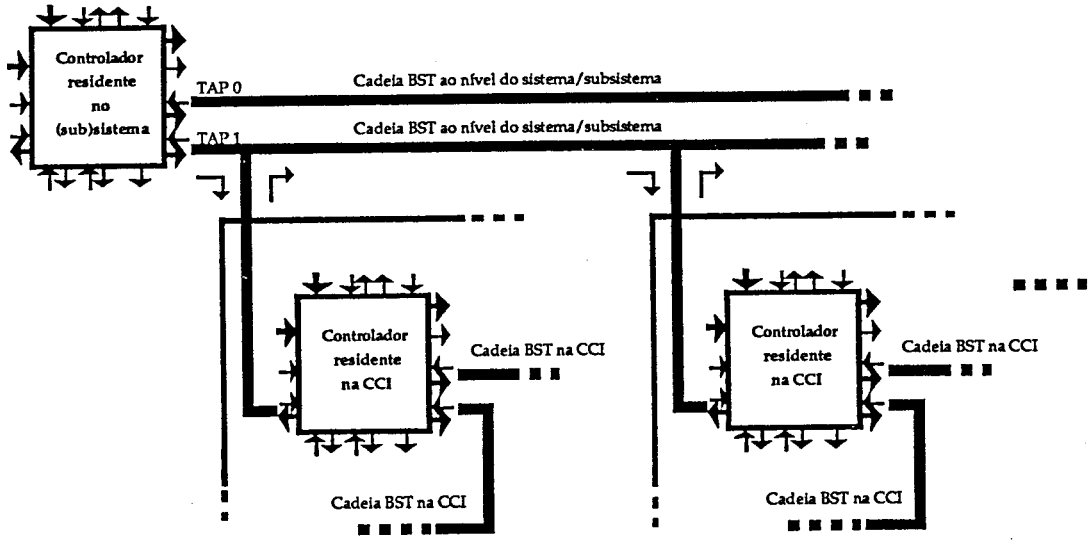


Fig.5.6: Extensão do BST para barramento de teste ao nível do sistema/subsistema.

As considerações apresentadas justificam a selecção do BST como barramento de teste para interligar CCIs com controladores residentes, sendo a interacção entre os dois níveis efectuada pela infraestrutura BST dos próprios controladores. Esta infraestrutura permitirá ao controlador a nível superior dar início à realização dos testes nas CCIs em que tal seja pretendido, e aceder à informação relativa ao resultado parcial ou total dos testes já realizados.

A caracterização da funcionalidade pretendida para um controlador residente deverá finalmente incluir a possibilidade de interrogar cada controlador acerca da identificação da respectiva CCI, de modo a verificar que todas as CCIs pretendidas estão presentes.

O quadro 5.1 apresenta uma síntese dos requisitos de funcionalidade que foram descritos para um controlador residente, distinguindo-se entre os que dizem respeito à exploração das potencialidades proporcionadas pela infraestrutura BST existente na CCI, e os que se referem à interacção com o nível hierárquico superior.

Exploração das potencialidades proporcionadas pela infraestrutura BST da CCI a testar
• O controlador residente deverá suportar um conjunto de instruções que implementem directamente as operações necessárias para o controlo da infraestrutura BST (incluindo as que se referem ao protocolo de sincronismo com um equipamento de teste exterior). • Deverá ser suportada a existência de duas cadeias BST na CCI a testar. • Deverá ser possível o acesso exterior à(s) cadeia(s) BST existente(s) na CCI a testar. • Deverá existir uma saída do tipo sim/não que indique para o exterior o resultado do teste.
Interacção com o nível superior
• Deverá ser suportada uma configuração hierárquica, através da existência de uma infraestrutura BST no controlador residente. • A infraestrutura BST de um controlador residente deverá permitir que um outro controlador, a nível hierárquico superior, dê início à realização de testes, e aceda aos resultados parciais ou totais. • Os resultados acessíveis pelo controlador a nível hierárquico superior deverão permitir a discriminação entre diferentes tipos de faltas, e identificar a cadeia BST em que estas faltas foram detectadas. • Deverá ser possível ao controlador a nível hierárquico superior inquirir sobre a identificação das CCIs presentes.

Quadro 5.1: Síntese da funcionalidade pretendida para um controlador residente em CCIs com BST.

5.2.2. Identificação das operações elementares requeridas

As operações elementares requeridas determinam o conjunto de instruções a ser suportado pelo controlador, e serão apresentadas em três grupos. Estes grupos dizem respeito ao controlo da infraestrutura BST, ao protocolo de sincronismo com um equipamento de teste exterior, e ao controlo de recursos internos ao controlador.

As operações elementares destinadas ao controlo da infraestrutura BST devem proporcionar os mecanismos básicos que permitam implementar as etapas definidas no protocolo de teste para CCIs com BST (apresentado no capítulo 3), sendo a identificação destas operações elementares efectuada por análise individual das etapas do protocolo.

i) **Teste da infraestrutura BST**

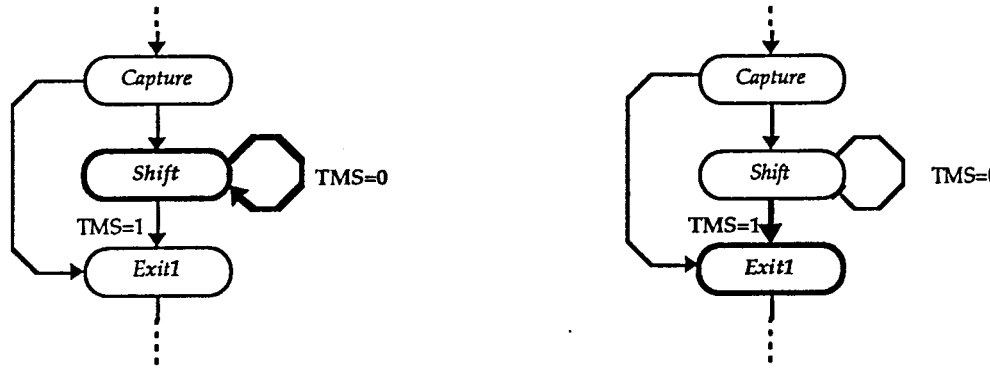
- Inicialização da infraestrutura BST

A concretização desta etapa será possível pela existência de uma operação elementar que provoque um impulso a **0** na linha /TRST associada à cadeia BST que se pretenda inicializar. Repare-se que esta etapa pode alternativamente ser garantida pela aplicação de cinco impulsos no relógio de teste (TCK) do TAP dos CIs cuja lógica BST se pretende inicializar, enquanto a respectiva linha TMS é mantida em **1** (IEEE Std.1149.1, 90, p. 5-16). Esta alternativa identifica a necessidade de uma outra operação elementar, que consiste em aplicar um impulso no relógio de teste (TCK), enquanto a linha TMS é mantida num nível lógico pré-definido.

- Teste à operacionalidade da cadeia BST

Esta etapa será concretizada através dos registos de instrução, sendo a sequência de bits que surgem na saída série (TDO) da CCI a testar comparada com a sequência esperada. A concretização desta etapa requer operações elementares de dois tipos, em que o primeiro corresponde ao já descrito anteriormente para o controlo da linha TMS. Esta operação será aqui requerida para seleccionar os registos de instrução, já que estes registos não ficam automaticamente seleccionados (entre TDI e TDO de cada CI) após a inicialização.

A outra operação elementar requerida consiste em efectuar o deslocamento através da cadeia série formada pelos registos seleccionados na lógica BST de cada CI, sendo a sequência de bits que surge na saída série (TDO) da CCI comparada com a sequência esperada. De acordo com a especificação apresentada em (IEEE Std 1149.1, 90, cap. 5), o deslocamento de N bits através da cadeia BST terá lugar pela manutenção do controlador do TAP (na lógica BST interna a cada CI) no estado *Shift-DR* (*Shift-IR*) durante N-1 impulsos no relógio de teste (TCK), devendo o deslocamento do último bit ser acompanhado pela transição do controlador do TAP para o estado *Exit1-DR* (*Exit1-IR*). Isto significa que a linha TMS deverá ser mantida em **0** durante o deslocamento dos primeiros N-1 bits, passando a **1** para o deslocamento do último bit, tal como se ilustra na figura 5.7. A operação elementar que promova o deslocamento com comparação deve garantir o controlo da linha TMS, a colocação na linha TDI da CCI a testar de cada novo bit a deslocar, e a comparação do bit surgido na linha TDO da CCI com o seu valor esperado, por cada impulso no relógio de teste (TCK). Adicionalmente, esta operação elementar deve permitir o uso de uma máscara individual para cada um dos N bits a deslocar, já que poderá não ser conhecido o valor esperado para alguns destes bits.



(a) Deslocamento dos primeiros (N-1) bits. (b) Deslocamento do último bit.

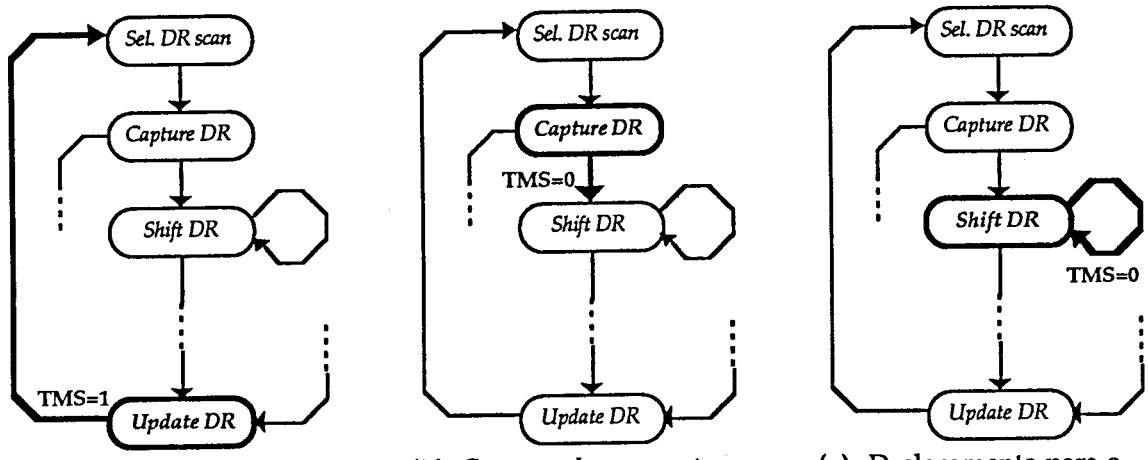
Fig.5.7: Estados do controlador do TAP no deslocamento de N bits através da cadeia BST da CCI.

- Leitura dos registos de identificação dos componentes

Este passo tem por objectivo deslocar para o exterior o conteúdo dos registos de identificação existentes (código de identificação, e código de identificação adicional), e compará-los com o valor esperado. Voltam a ser necessárias as operações elementares correspondentes ao controlo individual da linha TMS, e à realização de deslocamentos com comparação, mas identifica-se aqui a necessidade de uma outra operação elementar, correspondente ao deslocamento sem comparação. Com efeito, o deslocamento para os registos de instrução do comando que selecciona os registos de identificação dos componentes é uma operação de deslocamento em que não se torna necessário analisar a sequência de bits que surge na linha TDO da CCI. Aliás, o deslocamento com comparação obriga ao armazenamento adicional do valor esperado, e da respectiva máscara, por cada bit, o que significa triplicar a capacidade de memória necessária por vector. A operação de deslocamento sem comparação é no restante idêntica à de deslocamento com comparação, devendo verificar a transição de estados ilustrada na figura 5.7, e garantindo a presença na linha TDI da CCI de cada novo bit a deslocar, e o controlo da linha TMS.

ii) Teste de ligações na CCI

Esta etapa inclui o teste das ligações BST, e o teste das ligações pertencentes a grupos de componentes não BST, e requer a selecção dos registos de instrução, o envio do comando que selecciona os registos BST em modo de teste externo (*Extest*), e o deslocamento sucessivo de vectores, com comparação das respostas obtidas relativamente aos seus valores esperados. Repare-se que o deslocamento para o interior da cadeia BST de um novo vector é feito em simultâneo com o deslocamento para o exterior das respostas capturadas *para o vector anterior*, tal como se ilustra na figura 5.8.



(a) Aplicação do vector deslocado para o interior da cadeia BST.

(b) Captura das respostas ao vector aplicado.

(c) Deslocamento para o interior de novo vector, e para o exterior das respostas ao anterior.

Fig.5.8: Estados do controlador do TAP para o teste de ligações.

As operações elementares requeridas consistem portanto no controlo individual da linha TMS, e nas operações de deslocamento sem e com comparação, todas já anteriormente identificadas.

iii) Teste dos componentes BST

Esta etapa do protocolo inclui a verificação da operacionalidade dos componentes BST com auto-teste incorporado, e dos componentes BST sem auto-teste. Enquanto a realização de testes relativamente a este segundo caso é idêntica ao teste de ligações, do ponto de vista das operações elementares requeridas (deslocamento sucessivo de vectores e respostas), já o mesmo não sucede com o teste de componentes BST que dispuserem de auto-teste. A verificação de funcionalidade para este tipo de componentes tem início pelo envio para os respectivos registos de instrução do comando que activa as funções de auto-teste, tendo então lugar a transição do controlador do TAP para o estado *Run Test / Idle*, após o que será aplicado um número de impulsos no relógio de teste (TCK) não inferior ao que o fabricante especifica para este fim, tal como se ilustra na figura 5.9. O resultado do auto-teste estará então disponível num registo de dados, para ser comparado com o valor esperado.

Para além de operações elementares já anteriormente identificadas, como o controlo individual da linha TMS, e operações de deslocamento sem e com comparação, surge nesta etapa a identificação de uma nova operação elementar, que consiste na aplicação de um

número pré-definido de impulsos no relógio de teste (TCK), mantendo a linha TMS em **0**. Esta operação elementar destina-se explicitamente a permitir a execução da instrução de auto-teste (*Runbist*), tal como definida em (IEEE Std 1149.1, 90, p. 7-20).

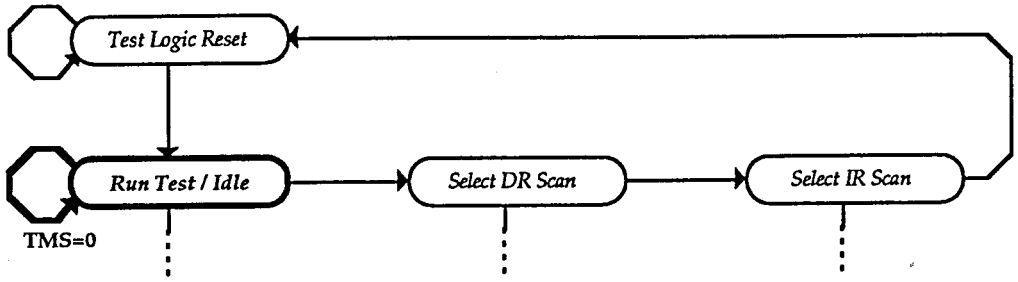


Fig.5.9: Estado do controlador do TAP, durante a execução do auto-teste em componentes BST.

Finalmente, deve estar também disponível uma operação elementar que permita seleccionar qual das duas cadeias BST deve ser controlada. Esta operação elementar é necessária para o caso de CCIs com duas cadeias BST, em que a realização do teste requererá a comutação frequente da cadeia activa.

Operações elementares para o controlo da infraestrutura BST
• Aplicar um impulso a **0** na linha /TRST. • Aplicar um impulso no relógio de teste (TCK), enquanto a linha TMS é mantida no valor pretendido. • Deslocar um conjunto de N bits para o interior da cadeia BST, sem comparar os bits deslocados para o exterior. A linha TMS deverá ser mantida em **0** durante o deslocamento dos primeiros N-1 bits, e colocada em **1** para o deslocamento do último bit. • Deslocar N bits para o interior da cadeia BST, comparando os bits deslocados para o exterior com o seu valor esperado, e fazendo uso de uma máscara que indique quais os bits cuja comparação é relevante. A linha TMS deverá ser mantida em **0** durante o deslocamento dos primeiros N-1 bits, e colocada em **1** para o deslocamento do último bit. • Aplicar N impulsos no relógio de teste (TCK), mantendo a linha TMS em **0**. • Seleccionar a cadeia BST a que serão aplicadas as operações elementares definidas anteriormente.

Quadro 5.2: Síntese das operações para o controlo da infraestrutura BST.

O quadro 5.2 apresenta uma síntese das operações elementares identificadas para

permitir o controlo da infraestrutura BST.

O protocolo de sincronismo entre a infraestrutura BST e o equipamento de teste exterior identifica a necessidade de mais duas operações elementares, destinadas a controlar a saída de sincronismo, e a observar a entrada de sincronismo, tal como se ilustra na figura 5.4.

O controlo da saída de sincronismo pode ser assegurado por uma operação elementar que coloque nesta saída o valor lógico pretendido, de modo a permitir ao controlador residente indicar ao equipamento de teste exterior que já terminou o processo de aplicação de um novo vector através da infraestrutura BST (destinada a suceder à operação elementar que provoca a transição de estado ilustrada na figura 5.8.(a)), ou que já capturou as respostas através desta mesma infraestrutura (destinada neste caso a suceder à operação elementar que provoca a transição de estado ilustrada na figura 5.8.(b)).

A outra operação elementar que se identifica para efeitos de sincronismo deve permitir que se retenha o controlador durante um número indefinido de impulsos de relógio (relógio do controlador), até que a entrada de sincronismo se encontre no valor pretendido. Esta operação elementar permitirá ao controlador determinar quando é que o equipamento de teste exterior já terminou o processo de aplicação de um novo vector às entradas primárias da CCI a testar (destinada a preceder a operação elementar que provoca a passagem para o estado *Capture-DR* (*Capture-IR*), na figura 5.8.(b)), ou quando é que este equipamento exterior já capturou as respostas presentes nas saídas primárias da CCI a testar (destinada a preceder o deslocamento de um novo vector para o interior da cadeia BST, ilustrado na figura 5.8.(c)).

As operações elementares para a implementação deste protocolo de sincronismo estão identificadas no quadro 5.3.

Operações elementares para o protocolo de sincronismo com equipamentos exteriores
• Colocar na saída de sincronismo o valor pretendido. • Reter o controlador durante um número indefinido de ciclos de relógio (relógio do controlador), até que a entrada de sincronismo se encontre no valor pretendido.

Quadro 5.3: Síntese das operações para o protocolo de sincronismo com um equipamento exterior.

A identificação do último grupo de operações elementares surge em resultado da necessidade de se controlarem os recursos internos do próprio controlador, e o fluxo do programa de teste.

As operações elementares destinadas a permitir o deslocamento de N bits, com ou sem comparação, ou a aplicação de N impulsos no relógio de teste (TCK), descritas no quadro 5.2, evidenciam a necessidade de existir um contador interno ao controlador, cujo conteúdo indique o número de impulsos (N) pretendidos no relógio de teste (TCK). Este facto torna necessária uma operação elementar para carregar este contador com o valor pretendido, e que se destina a preceder as operações elementares referidas acima.

A caracterização da funcionalidade pretendida para o controlador, descrita no quadro 5.1, especifica que os resultados acessíveis por um controlador existente a um nível hierárquico superior deverão permitir a discriminação entre os diferentes tipos de faltas, e identificar a cadeia BST em que estas faltas foram detectadas. Este requisito sugere a existência de um registo de estado que disponha de dois conjuntos de elementos de memória (*flags*), associados às duas cadeia BST da CCI a testar. A selecção da *flag* que deverá ser activada se ocorrer a detecção de uma falta constituirá deste modo outra das operações elementares requeridas. Esta operação elementar destina-se a anteceder a operação de deslocamento com comparação que efectua a aplicação de um vector para a detecção de um novo tipo de faltas, ou em quaisquer outras circunstâncias em que tal se revele adequado.

A existência de um conjunto de *flags* destinadas a sinalizar a ocorrência de um determinado tipo de falta conduz por sua vez à identificação de uma outra operação elementar, destinada a controlar o fluxo do programa de teste, e que consiste num salto condicional, baseado no estado da *flag* de erro seleccionada. Esta operação elementar permitirá que o fluxo normal do programa seja interrompido, após a realização de uma operação de deslocamento com comparação, se ocorrer a detecção de uma falta. Recorde-se que o teste através da infraestrutura BST requer que a CCI a testar se encontre alimentada, o que pode tornar necessária a interrupção do teste, caso se detecte a ocorrência de curto-circuitos entre ligações, já que um vector que force valores lógicos complementares em saídas curto-circuitadas se manterá activo até que seja aplicado um novo vector (Bhavsar, 90). Finalmente, deverá existir uma outra operação elementar também relacionada com o controlo do fluxo do programa de teste, e destinada a concluir a sua execução.

O quadro 5.4 apresenta uma síntese do último grupo de operações elementares identificadas, destinadas ao controlo dos recursos internos do controlador, e do fluxo do programa de teste.

O conjunto total de operações elementares descritas proporciona todos os mecanismos básicos necessários para a concretização das etapas que constituem o protocolo de teste para CCIs com BST.

Operações elementares para o controlo dos recursos internos, e do fluxo do programa de teste
• Carregar um contador interno com um valor que indica o número de impulsos a aplicar no relógio de teste (TCK). • Seleccionar a *flag* de erro que deve ser actuada pela posterior detecção de uma falta. • Permitir um salto condicional no fluxo do programa de teste, de acordo com o estado da *flag* de erro seleccionada. • Concluir a execução do programa de teste.

Quadro 5.4: Síntese das operações para o controlo de recursos internos, e do fluxo do programa de teste.

5.3. ESPECIFICAÇÃO DE CARACTERÍSTICAS PARA A GAPT

A identificação da informação requerida por cada etapa do protocolo de teste, e a especificação dos procedimentos principais que as constituem, requerem uma análise cuidada das questões principais que se prendem com a GAPT. Esta análise dará lugar a uma especificação de características neste domínio, que será apresentada nesta secção, e que permitirá enquadrar a apresentação a efectuar no capítulo 7.

A GAPT descrita neste trabalho assume que a funcionalidade proporcionada pela infraestrutura BST disponível não ultrapassa a que corresponde às três instruções obrigatórias (*Bypass*, *Sample/Preload*, e *Extest*), e às quatro instruções opcionais (*Idcode*, *Usercode*, *Intest*, e *Runbist*), que são definidas em (IEEE Std 1149.1 90, cap. 7). A implementação do conjunto de passos que corresponde a cada etapa do protocolo de teste é definida em termos das três instruções obrigatórias referidas. As quatro instruções opcionais estão essencialmente relacionadas com o teste de componentes, sendo o correspondente conjunto de passos implementado, quando estas instruções estiverem disponíveis.

O fluxo de dados na GAPT foi já anteriormente referido, estando ilustrado na figura 5.3. A apresentação então efectuada chamou a atenção para o facto de a geração de vectores para o teste de grupos de componentes não BST ser efectuada exteriormente, por recurso a ferramentas computacionais adequadas. A descrição de cada grupo de componentes não BST deve deste modo incluir os respectivos vectores de teste, e constitui parte da informação de entrada requerida pela GAPT, cujo fluxo de dados pode então representar-se, de forma simplificada, como se ilustra na figura 5.10.

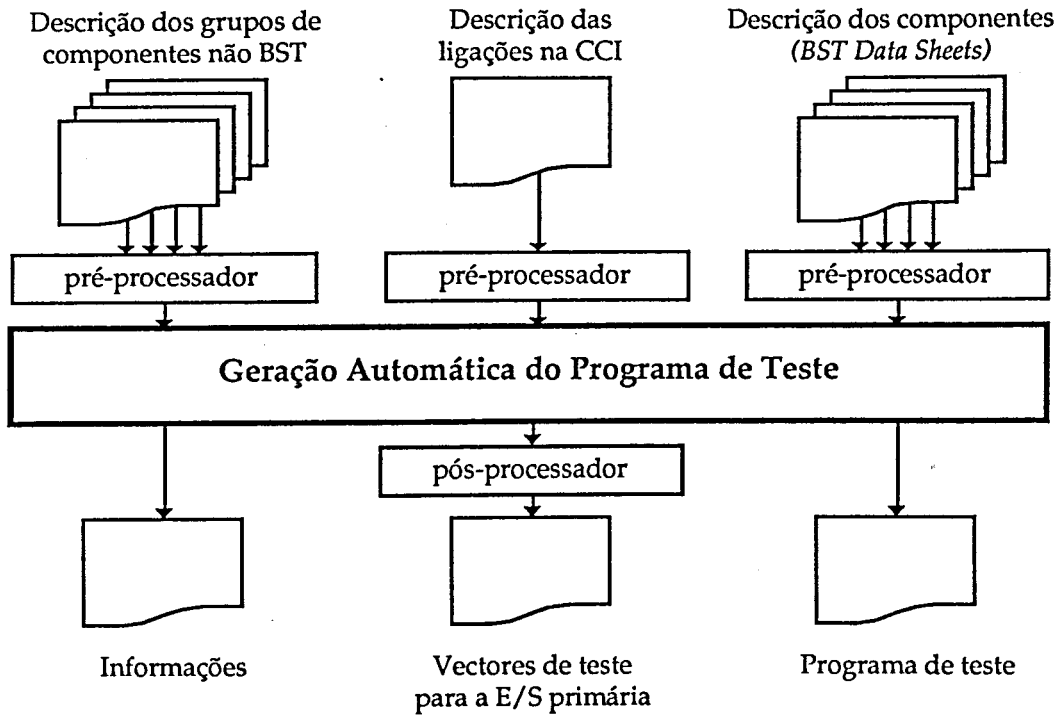


Fig.5.10: Representação simplificada do fluxo de dados na GAPT.

O suporte a diferentes formatos para a informação contida nos ficheiros de entrada poderá ser conseguido pela utilização de diferentes pré-processadores, uma vez que se verifica a existência de mais do que uma alternativa neste domínio (Hines, 87), (Etherington, 87), (Plassart et al., 90).

Quer a linguagem VHDL (IEEE Std 1076, 87), quer o formato EDIF (EIA EDIF, 87), são normalmente suportados por sistemas de CAD, e existem já propostas nestes dois domínios, com o objectivo de permitir a descrição da infraestrutura BST presente num CI. O formato EBST † está proposto como extensão do EDIF para representar a informação que descreve a infraestrutura BST num CI (Mortensen et al., 91), (Wijngaarden, 90). Também a linguagem BSDL †† (Parker et al., 90, 91) está proposta com este mesmo objectivo (no domínio do VHDL), e foi recentemente adoptada pelo *1149.1 Working Group*, no sentido de vir eventualmente a integrar a norma IEEE 1149.1 (Maunder, 91).

A especificação de vectores de teste pode também ser efectuada de acordo com mais do que uma alternativa (Verhelst, 89), (Sebesta et al., 90), (Moran et al., 90), (Plassart et al., 91), pelo que a existência do pós-processador referido na figura 5.10 pode igualmente

---

† *Extended Boundary Scan Test*

†† *Boundary Scan Description Language*

revelar-se com interesse.

A multiplicidade de alternativas respeitantes aos formatos de entrada e de saída para a informação processada pela GAPT justifica que a apresentação que é efectuada neste trabalho se restrinja à consideração de um conjunto de estruturas de dados internas, aptas a representar toda a informação manipulada em cada etapa do protocolo de teste. A tradução entre a informação de entrada e de saída, nos vários formatos possíveis, e o conteúdo das estruturas de dados referidas, não faz portanto parte do trabalho descrito.

O fluxo de dados simplificado que corresponde a cada uma das etapas principais do protocolo de teste será agora apresentado nas secções seguintes, conjuntamente com a identificação dos procedimentos principais envolvidos.

5.3.1. Teste da infraestrutura BST na CCI

O teste da infraestrutura BST tem por objectivo verificar a operacionalidade das ligações que conduzem os sinais de teste (/TRST, TMS, TCK, TDI, TDO). O fluxo de dados correspondente à geração do respectivo segmento do programa de teste está ilustrado na figura 5.11.

Apesar de ser requerida a descrição das ligações existentes na CCI (*netlist*), para determinar a ordem pela qual os componentes surgem na cadeia BST, a componente principal da informação necessária é proveniente da descrição da infraestrutura BST em cada componente. Este facto sugere a utilização de uma estrutura de dados orientada para o componente, e onde se representem todos os elementos necessários à geração do programa de teste para esta etapa.

A geração do segmento do programa para o teste da infraestrutura BST constitui um dos procedimentos principais na GAPT, e tem por objectivo permitir a detecção de faltas nesta infraestrutura, de acordo com as considerações efectuadas quando se descreveu o protocolo para o teste de CCIs com BST (ver capítulo 3). A realização deste teste pelo controlador residente, utilizando os registos de instrução presentes em cada componente com BST, está ilustrada na figura 5.12.

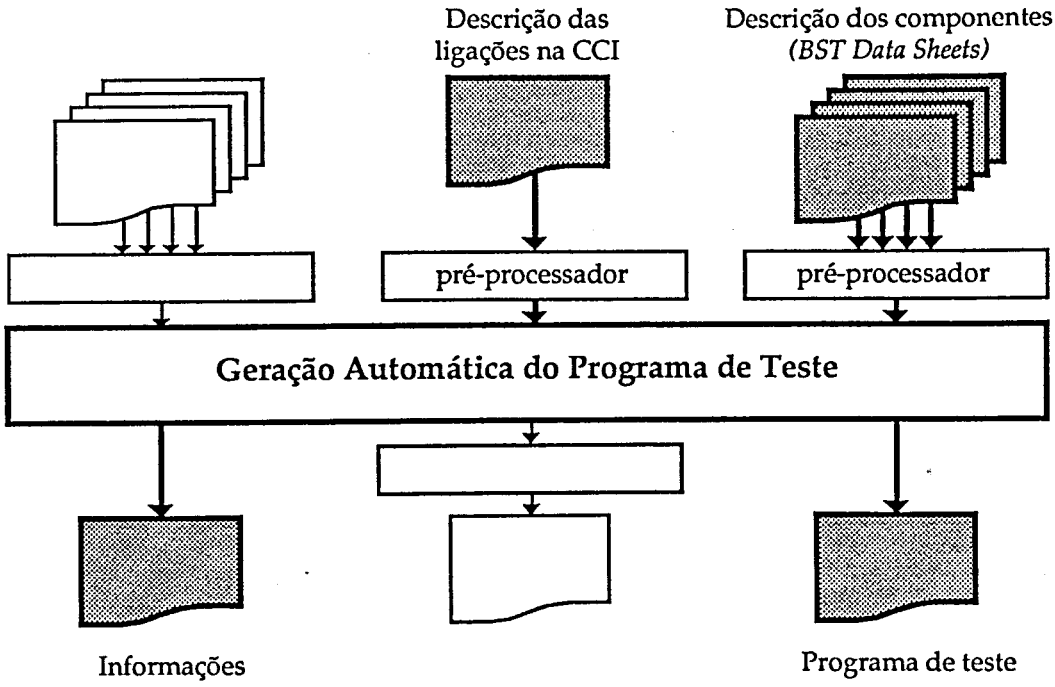


Fig.5.11: Fluxo de dados na GAPT para a infraestrutura BST.

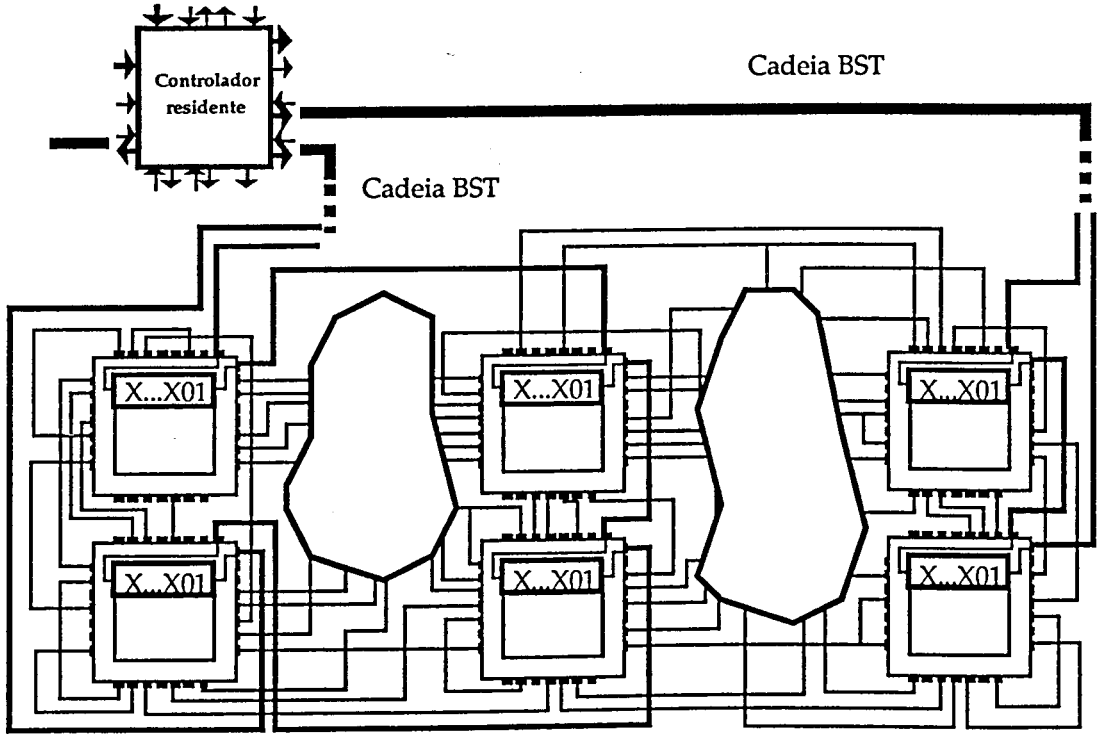


Fig.5.12: Teste da infraestrutura BST na CCI.

5.3.2. Teste de ligações na CCI

O teste de ligações na CCI tem por objectivo detectar a ocorrência de faltas de continuidade, e de curto-circuitos, quer nas ligações BST, quer nas ligações pertencentes a grupos de componentes não BST. Estes dois tipos de testes serão agora descritos nas secções seguintes.

5.3.2.1. Teste das ligações BST

A geração do segmento do programa para o teste das ligações BST requer a descrição das ligações na CCI (*netlist*), e da infraestrutura BST em cada componente, tal como se ilustra no fluxo de dados representado na figura 5.13.

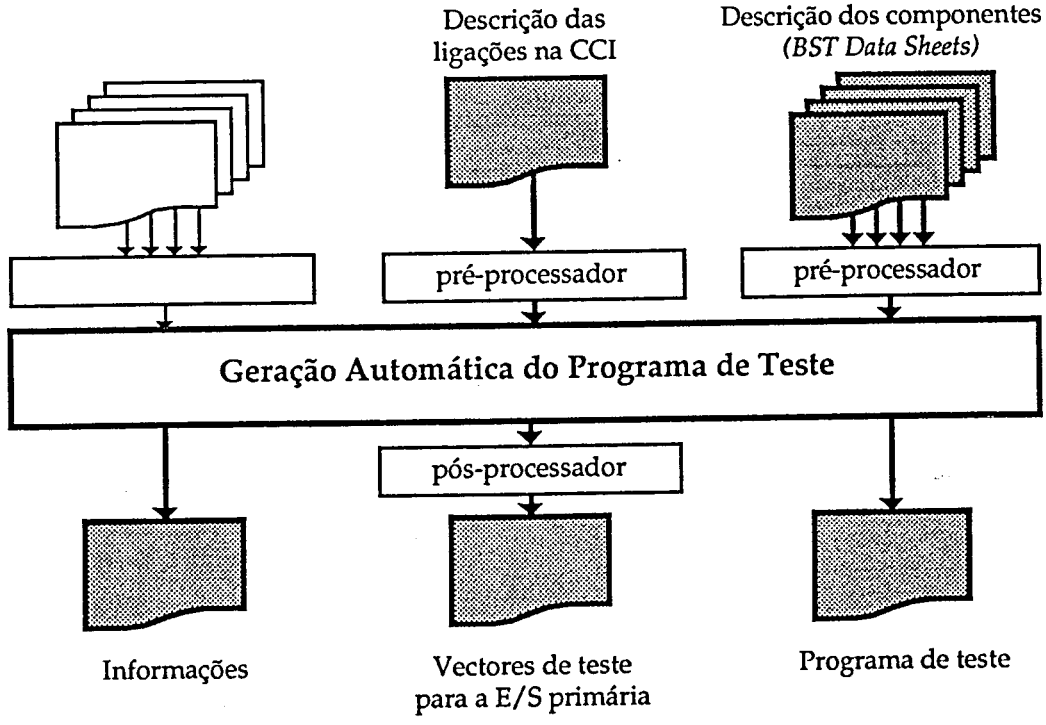


Fig.5.13: Fluxo de dados na GAPT para as ligações BST..

Apesar de não ser agora possível identificar uma destas componentes como principal, é plausível considerar para este caso a utilização de uma estrutura de dados orientada para a ligação, e onde se combinem os elementos principais necessários à geração do programa para este tipo de teste. Estes elementos deverão permitir uma descrição completa dos conjuntos de nós associados a cada ligação, quer os que correspondem a pinos BST, quer os

que fazem parte da E/S primária da CCI, incluindo a informação que descreve como forçar o estado de alta impedância, para os nós que dispuserem desta possibilidade. A estrutura de dados assim constituída deverá proporcionar toda a informação necessária aos procedimentos para a geração do programa para o teste de ligações na CCI.

A GAPT para as ligações BST é constituída por mais do que um procedimento principal, dada a especificidade da geração dos segmentos do programa de teste para cada tipo de faltas, e a necessidade de se efectuar uma selecção dos nós CL a activar (tal como se descreveu no capítulo 4). O teste destas ligações é efectuado através dos registos BST presentes em cada componente, tal como se ilustra na figura 5.14, cujas células deverão estar configuradas em modo de teste externo.

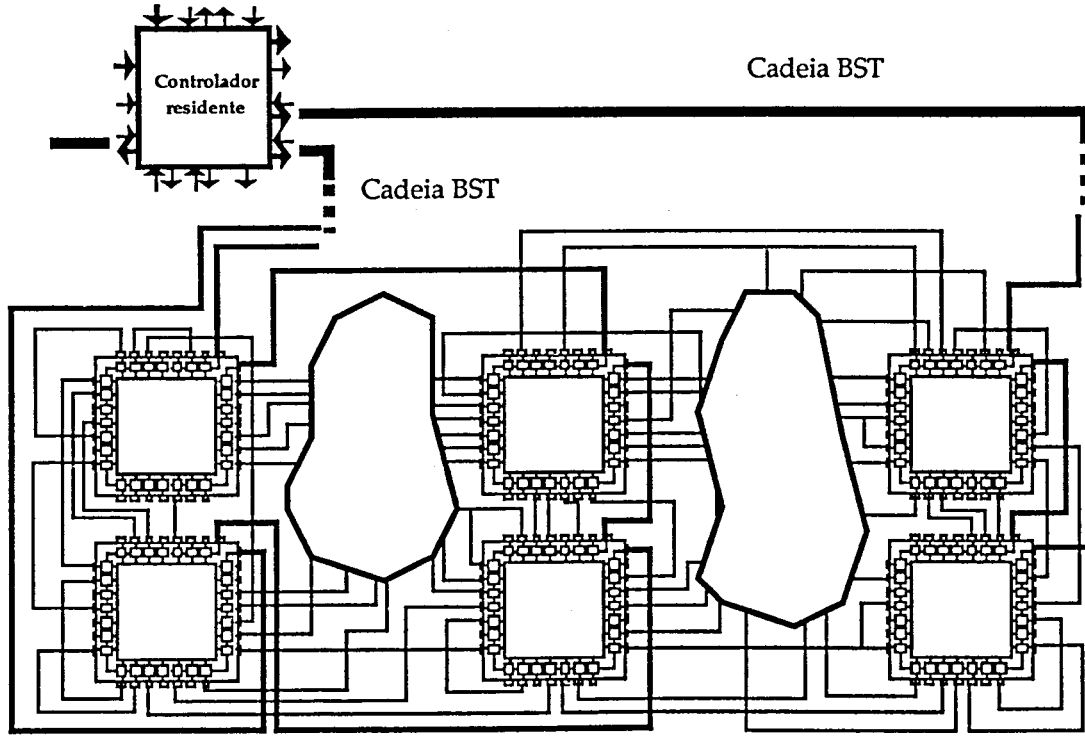


Fig.5.14: Teste das ligações BST.

5.3.2.2. Teste das ligações pertencentes a grupos de componentes não BST

A geração do segmento de programa para o teste das ligações pertencentes a grupos de componentes não BST tem lugar a partir da informação que caracteriza cada um destes grupos, e que inclui os respectivos vectores de teste. Recorde-se que estes vectores de teste

deverão ser gerados exteriormente, de acordo com o fluxo de dados ilustrado na figura 5.3. A caracterização resultante para cada grupo de componentes não BST permitirá então a GAPT para este passo, de acordo com o fluxo de dados ilustrado na figura 5.15.

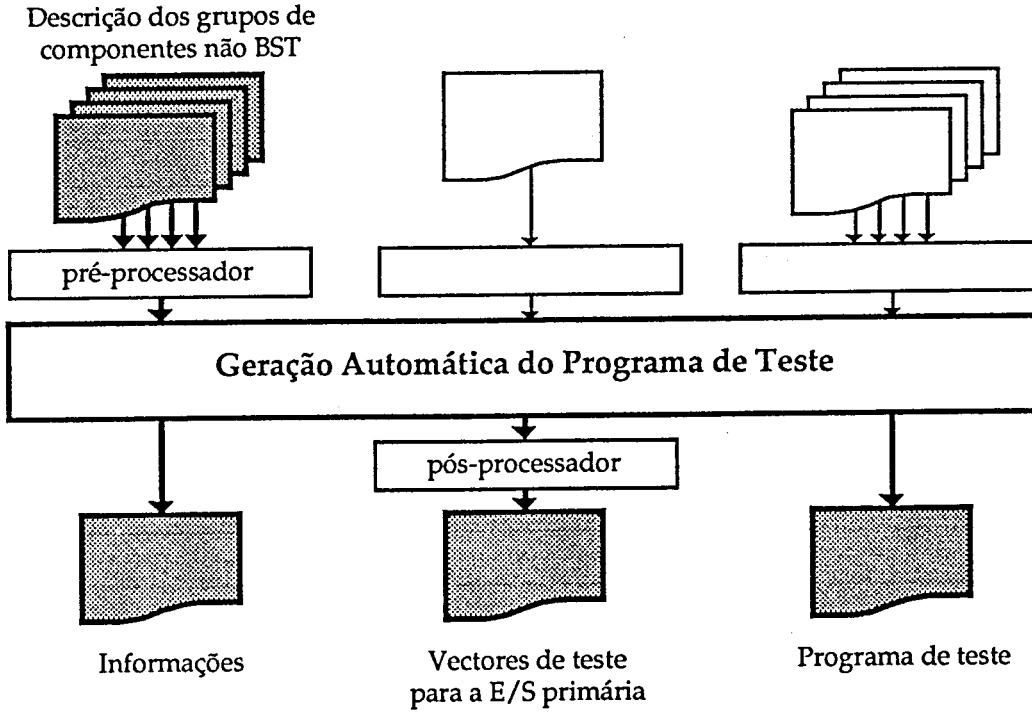


Fig.5.15: Fluxo de dados na GAPT para as ligações pertencentes a grupos de componentes não BST.

O procedimento que efectua a geração do segmento de programa de teste para este passo responsabiliza-se por integrar na(s) cadeia(s) BST o conjunto de vectores correspondentes a cada grupo de componentes não BST. Repare-se que esta operação implica a serialização do conjunto de vectores especificados para cada grupo, de acordo com a descrição do conjunto de células BST que os rodeiam. A execução do segmento de programa gerado permitirá então o teste destes grupos através da infraestrutura BST da CCI, tal como se ilustra na figura 5.16.

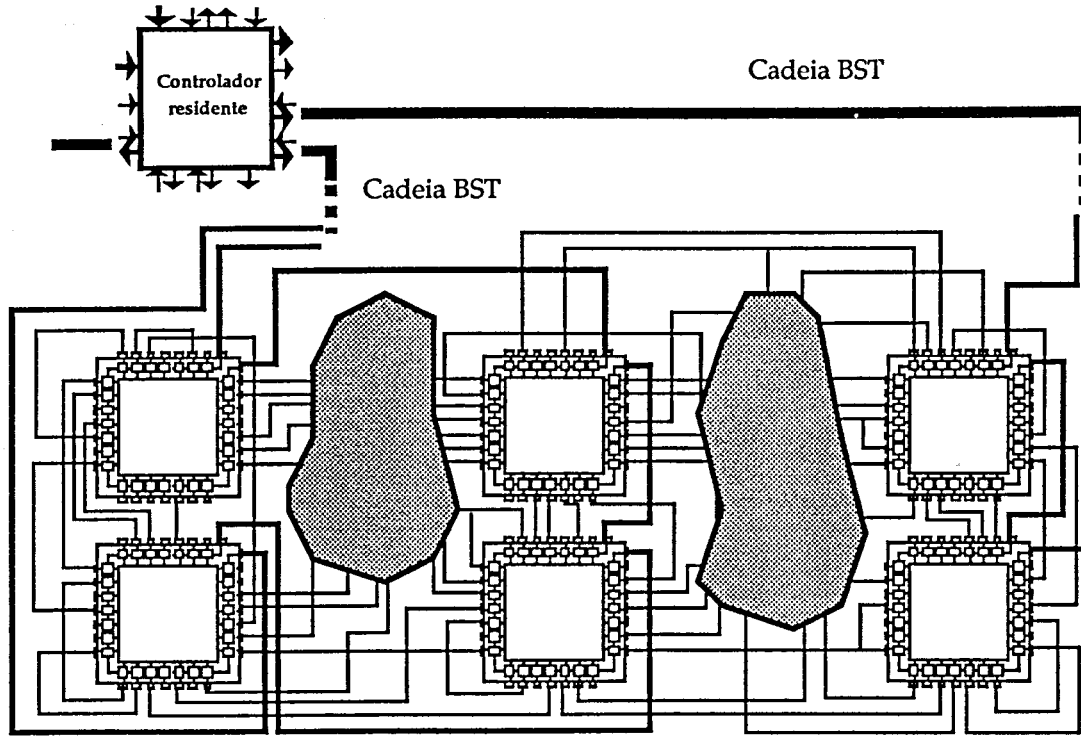


Fig.5.16: Teste das ligações pertencentes a grupos de componentes não BST.

5.3.3. Teste dos componentes BST

Esta etapa tem por objectivo verificar a operacionalidade dos componentes BST presentes na CCI, e é principalmente destinada aos componentes que disponham de auto-teste incorporado (BIST). Será também nesta fase que terá lugar o teste de componentes BST que não disponham de auto-teste, que será realizado por deslocamentos sucessivos de vectores e respostas. Este último procedimento corresponde a um teste de alcance limitado, dada a morosidade do processo, pelo que se pretenderá essencialmente confirmar que os componentes em questão estão activos e respondem da forma esperada, mais do que efectuar um teste que cubra de forma satisfatória a totalidade das faltas possíveis.

A ordem dos componentes em cada cadeia BST deve também ser conhecida, pelo que o fluxo de dados correspondente a esta etapa pode ser apresentado como se ilustra na figura 5.17.

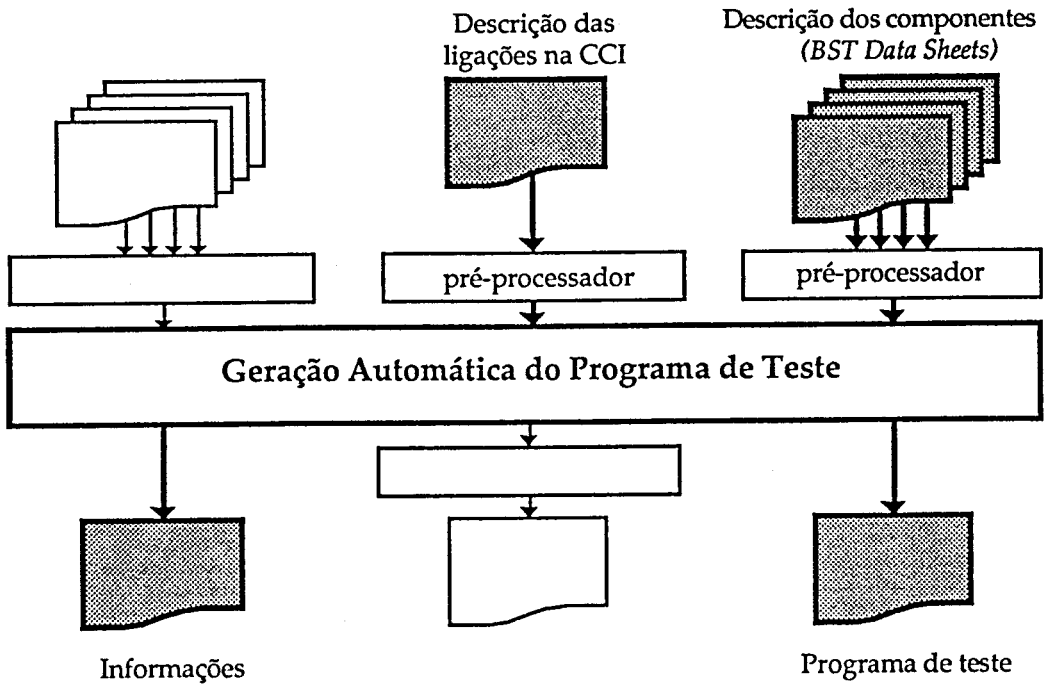


Fig.5.17: Fluxo de dados na GAPT para os componentes BST.

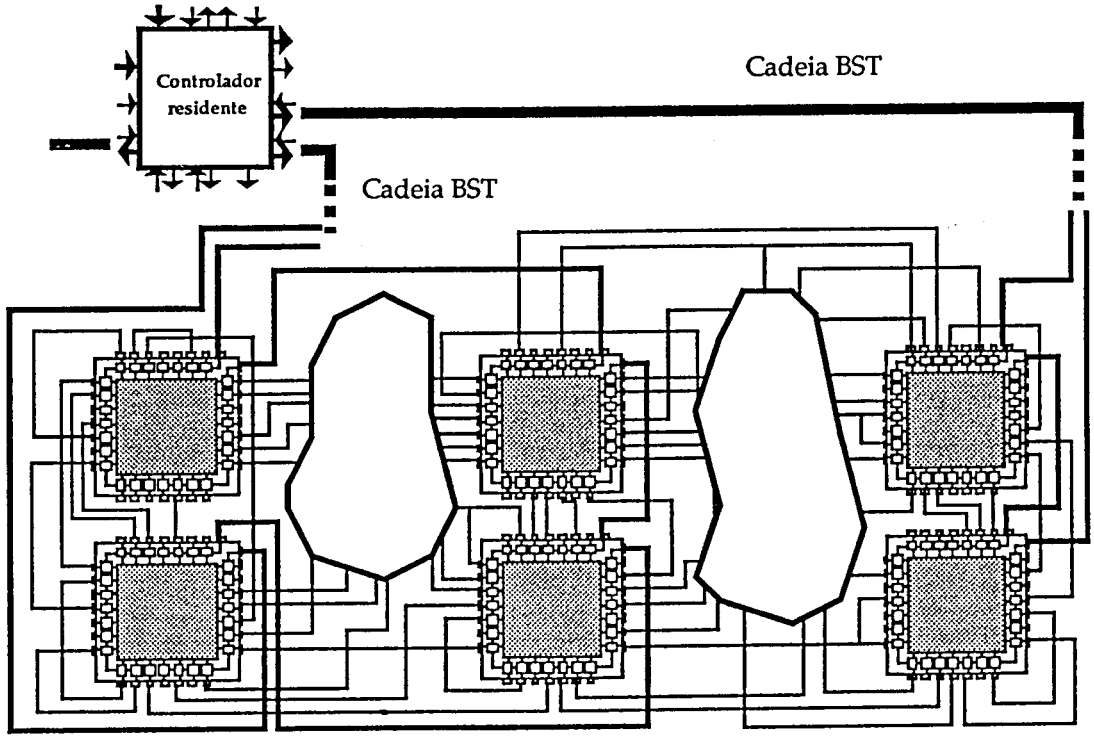


Fig.5.18: Teste aos componentes BST.

Repare-se que o fluxo de dados correspondente ao teste da infraestrutura BST na CCI envolve os mesmos tipos de informação ilustrados na figura 5.17, embora os elementos requeridos nesta etapa, relativamente à descrição dos componentes BST, sejam diferentes. Isto significa que à estrutura de dados orientada para o componente, então referida, deve ser adicionada a informação referente ao teste de componentes, quando este for pretendido.

O procedimento responsável pela geração deste segmento do programa de teste deverá distinguir entre os componentes BST que suportam auto-teste, e os que requerem a aplicação de um conjunto pré-determinado de vectores. Relativamente a este segundo caso, assume-se que a especificação dos vectores de teste consta da descrição do respectivo componente. Este procedimento produzirá a sequência de código que permite exercitar as funções de auto-teste disponíveis, e efectuará a integração dos vectores de teste especificados para os restantes casos. Os elementos colocados sob teste nesta etapa do protocolo estão assinalados na figura 5.18.

5.3.4. Síntese dos procedimentos principais na GAPT

O quadro 5.5 apresenta uma síntese dos procedimentos principais que foram identificados para permitir a GAPT, e cuja especificação será pormenorizada no capítulo 7.

Procedimentos principais na geração automática do programa de teste
Geração do segmento do programa de teste para a infraestrutura BST.
Geração do segmento do programa de teste para as ligações na CCI: • Selecção dos nós do tipo CL a activar para permitir a aplicação dos vectores de teste, de acordo com os requisitos apresentados no quadro 4.1. • Detecção de faltas de continuidade nas ligações BST. • Detecção de curto-circuitos nas ligações BST. • Detecção de faltas nas ligações pertencentes a grupos de componentes não BST.
Geração do segmento do programa de teste para os componentes BST.

Quadro 5.5: Síntese dos procedimentos principais na geração automática do programa de teste.

Uma vez apresentada a especificação de características para um controlador residente, e a caracterização da GAPT, torna-se necessário avaliar as modificações que serão introduzidas no protocolo de teste pela existência do controlador na CCI a testar, e que serão analisadas na secção seguinte.

5.4. PROTOCOLO DE TESTE PARA CCIS COM CONTROLADORES RESIDENTES

O protocolo de teste para CCIs com BST, apresentado no capítulo 3, identificou três etapas principais, que consistem na verificação da operacionalidade da infraestrutura BST, das ligações na CCI, e dos componentes presentes. A existência de um controlador residente numa CCI com BST tem por primeira consequência fazer com que a infraestrutura de teste disponível deixe de coincidir com a infraestrutura BST, e passe também a incluir o próprio controlador residente, bem como a memória que contem o programa de teste, se esta lhe for exterior. Este controlador residente assegurará todas as operações necessárias para a realização do teste através da infraestrutura BST, pelo que se tornará possível que o equipamento de teste exterior passe a ter uma visão da CCI a testar que se restringe às E/S primárias, e à interacção com o controlador residente.

Verificação da operacionalidade da infraestrutura de teste

• Teste do controlador residente (e da memória com o programa de teste, se esta lhe for exterior) †
• Teste da infraestrutura BST
  • Inicialização da lógica BST em todos os componentes
  • Verificação da operacionalidade do percurso série
  • Leitura da identificação dos componentes

Verificação da operacionalidade das ligações na CCI
• Teste das ligações BST • Teste das ligações pertencentes a grupos de componentes não BST
Verificação da operacionalidade dos componentes BST
• Teste dos componentes BST com auto-teste incorporado • Teste dos componentes BST sem auto-teste incorporado

Quadro 5.6: Etapas principais no protocolo de teste para CCIs com controladores residentes.

A infraestrutura de teste disponível é composta pela infraestrutura BST que atravessa a CCI, e cuja operacionalidade deverá ser verificada pelo controlador residente, da forma já

† Realizada pelo equipamento de teste exterior.

anteriormente referida, e pelo próprio controlador, cuja operacionalidade deverá ser verificada pelo equipamento de teste exterior. Isto significa que o protocolo de teste para CCIs com controladores residentes mantem as etapas já anteriormente descritas, mas que deverão agora ser precedidas pela verificação da operacionalidade do controlador, realizada pelo equipamento de teste exterior, tal como se descreve no quadro 5.6.

O teste ao controlador residente será facilitado se este componente dispuser de auto-teste incorporado, circunstância em que o acesso a estas funções poderá ter lugar através de pinos previstos para este efeito, ou por uma interacção em que o equipamento de teste exterior desempenhe as funções de um controlador a nível hierárquico superior. O número de nós a que é requerido o acesso directo será mínimo nestas circunstâncias, tal como se ilustra na figura 5.19 (o símbolo ↓↑ representa os canais de teste exteriores).

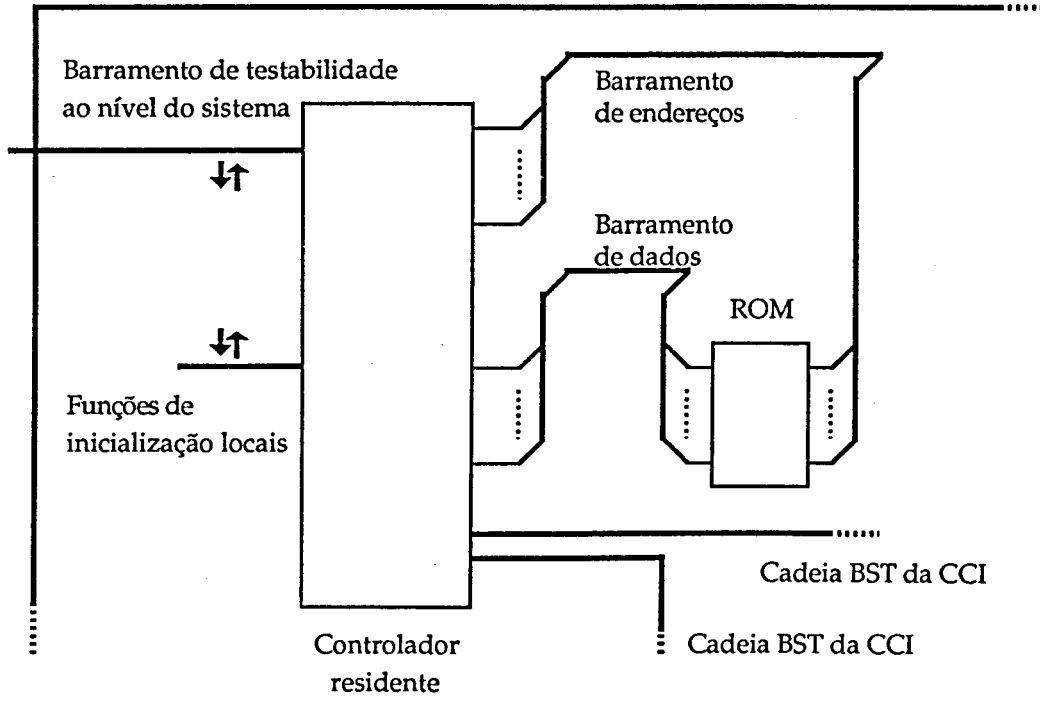


Fig.5.19: Interacção entre o controlador residente, e o equipamento de teste exterior.

5.5. SUMÁRIO

Este capítulo teve por objectivos principais apresentar uma especificação de características para um controlador de teste, residente em CCIs com BST, e para a geração automática do programa de teste (GAPT) que será executado por este controlador.

Os requisitos principais a que deve satisfazer a arquitectura do controlador de teste foram analisados, tendo sido apresentada a funcionalidade pretendida, e identificadas as operações elementares necessárias para uma exploração eficiente da infraestrutura BST da CCI. A apresentação de uma arquitectura apta a satisfazer estes requisitos será efectuada no capítulo 6.

A especificação de características para a GAPT foi então apresentada, partindo-se do pressuposto que a funcionalidade proporcionada pela infraestrutura BST em cada componente se restringe às três instruções obrigatórias, e às quatro instruções opcionais, descritas em (IEEE Std 1149.1, 90, cap. 7). Esta especificação de características teve início com uma identificação sumária do fluxo de informação em cada etapa do protocolo de teste, e dos procedimentos principais necessários à respectiva concretização. Estes elementos serão desenvolvidos no capítulo 7, onde os aspectos principais da GAPT serão discutidos em detalhe.

As alterações necessárias ao protocolo de teste para CCIs com BST, devidas à inclusão de um controlador residente, foram por fim consideradas. Estas alterações dizem apenas respeito ao equipamento de teste exterior, pelo que a GAPT para o controlador residente mantem o protocolo descrito no capítulo 3.

Capítulo 6

Arquitectura do controlador residente

Caracterização global

Arquitectura do processador de teste

Especificação das instruções suportadas

Testabilidade hierárquica: A infraestrutura BST

Controlo do processador

6. ARQUITECTURA DO CONTROLADOR RESIDENTE

Este capítulo descreve a arquitectura de um controlador residente para CCIs com BST, de acordo com as especificações que foram apresentadas no capítulo anterior. Esta arquitectura é apresentada de forma modular, de modo a permitir a identificação do núcleo correspondente ao processador de teste, e dos blocos que permitem a interacção com um barramento de testabilidade a nível hierárquico superior. A especificação do conjunto de instruções a suportar é então efectuada, a partir do conjunto de operações elementares descritas no capítulo anterior, e das condicionantes impostas pela arquitectura descrita. Esta especificação inclui a descrição da sequência de transição de estados para a execução de cada instrução, que constitui por sua vez a informação necessária para caracterizar a funcionalidade dos blocos que integram a unidade de descodificação e controlo. A discussão da infraestrutura BST a incluir no controlador residente, com o objectivo de permitir uma testabilidade hierárquica através da extensão do BST à escala da interligação entre CCIs, precede a última secção deste capítulo, onde se descrevem os dois modos possíveis para o funcionamento do processador, que consistem no controlo através da infraestrutura BST, e no funcionamento em modo autónomo.

6.1. CARACTERIZAÇÃO GLOBAL

Esta secção tem por objectivo efectuar a caracterização global de uma arquitectura que implemente a funcionalidade pretendida para um controlador residente em CCIs com BST, tal como foi descrita no quadro 5.1. De acordo com as especificações apresentadas no capítulo anterior, este controlador é constituído por três partes principais:

i) Um núcleo que constitui o processador de teste, com uma arquitectura dedicada especificamente ao controlo da infraestrutura BST existente na CCI a testar. Este processador suporta directamente o conjunto de operações elementares descritas nos quadros 5.2 a 5.4, e que será descrito nas secções 6.2 e 6.3 deste capítulo.

ii) Uma infraestrutura BST que tem por objectivo principal permitir a interacção com um nível hierárquico superior, por extensão do BST como barramento de teste entre CCIs com controladores residentes, e que será descrita na secção 6.4 deste capítulo.

iii) Uma memória que conterá o programa de teste a ser executado, e que poderá ou não ser exterior ao controlador. Tal como foi referido no capítulo anterior, a opção por uma memória exterior terá a vantagem de permitir uma redução de custos,

embora o controlador residente deixe de ser constituído por um único componente.

Quando a memória com o programa de teste for exterior, a expressão *controlador residente* será também usada para referir a reunião dos restantes blocos (i e ii), como se ilustra na figura 6.1. Esta figura representa igualmente dois outros blocos (*ClockMux* e *ResetGen*), destinados a permitir que o controlo do processador de teste tenha lugar localmente, ou através da infraestrutura BST, e que serão descritos na secção 6.5.

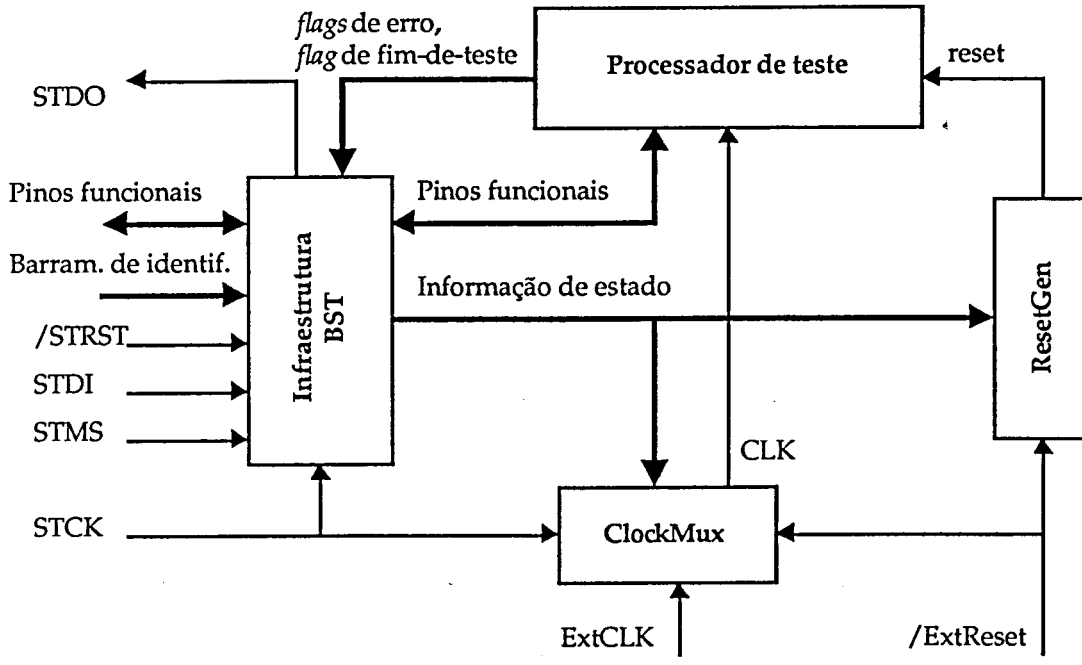


Fig.6.1: Diagrama de blocos global, para o controlador residente.

O TAP do próprio controlador (que será comandado por um controlador residente a nível hierárquico superior) será indicado pelo prefixo S, passando a designar-se por STAP (/STRST, STMS, STCK, STDO, STDI). A representação ilustrada na figura 6.1 indica que os dois TAPs proporcionados pelo controlador para comandar a infraestrutura BST da CCI a testar fazem parte do conjunto de pinos funcionais deste componente.

Repare-se que esta representação permite a distinção clara entre o processador de teste, destinado a suportar as operações elementares descritas nos quadros 5.2 a 5.4, e o bloco que se responsabiliza pela interacção com o nível hierárquico superior. Isto significa que qualquer alteração nesta interacção (tal como a adopção do barramento MTM, em vez de BST), envolverá apenas modificações neste bloco, desde que se continue a pretender a funcionalidade global descrita.

A definição da largura para o barramento de dados do processador merece uma

análise cuidada, em que sejam equacionados os diversos factores em jogo. Apesar da maior velocidade no deslocamento de vectores para o interior da cadeia BST, que seria proporcionada por um barramento de dados com um maior número de bits, os argumentos seguintes devem também ser considerados:

i) O número de operações elementares que se identificaram no capítulo anterior é reduzido, pelo que o número de bits necessários para codificar as instruções correspondentes será também reduzido.

Atendendo a que a codificação de uma instrução requer uma palavra de memória, a utilização de um maior número de bits por palavra corresponderia a uma menor eficiência na utilização da capacidade de memória disponível. Este facto é tanto mais relevante quanto maior for o número de palavras de memória destinadas ao armazenamento de códigos de instrução, face às que se destinam ao armazenamento de dados. Apesar de esta razão depender naturalmente do tamanho da(s) cadeia(s) BST presente(s), e do número total de vectores a aplicar, a utilização frequente que é previsível para algumas das operações elementares descritas (controlo da linha TMS, selecção da cadeia BST a controlar, operações de sincronismo, etc.) recomenda que se leve este factor em consideração.

ii) Apesar de a opção por um barramento de dados com um menor número de bits implicar um maior número de acessos à memória, para a mesma dimensão dos vectores a aplicar, a degradação que isto implica em termos de velocidade pode ser minimizada pela execução concorrente das operações de deslocamento, e de preparação dos novos bits a deslocar.

Com efeito, e embora as operações de deslocamento com comparação requeiram três acessos à memória por cada palavra a deslocar (valores a deslocar, valores esperados, máscaras de comparação), estes acessos podem ter lugar durante o deslocamento dos valores anteriormente carregados, tornando-se necessário interromper esta operação apenas durante o tempo requerido para actualizar o conteúdo dos blocos que a controlam.

iii) Um barramento de dados com um maior número de bits obrigará a um maior número de pinos no controlador residente, quando a memória com o programa de teste lhe for exterior.

Ponderados os factores descritos, concluiu-se que um barramento de dados com 8 bits constitui uma solução de compromisso satisfatória, pelo que foi este o valor adoptado.

A determinação da largura recomendável para o barramento de endereços enfrenta no entanto mais dificuldades, sobretudo pela falta de uma experiência de projecto neste domínio, o que torna difícil quantificar a capacidade de memória que será requerida para os programas de teste. Esta capacidade depende de diversos factores, tais como o número de

componentes BST na CCI, o número das ligações, o tamanho da(s) cadeia(s) BST existentes, as facilidades disponíveis para o teste dos componentes BST, e também a dimensão e complexidade dos grupos de componentes não BST presentes. Este último factor é particularmente importante, podendo mesmo inviabilizar a utilização de um controlador residente, se a capacidade de memória requerida para o teste de grupos de componentes não BST for demasiado elevada (Hansen, 89, p. 171).

Por outro lado, e para o caso de CCIs em cujo projecto se tenha desde o início adoptado a inclusão de um controlador residente, e levados em conta os respectivos requisitos, a capacidade de memória necessária para armazenar o programa de teste será compatível com um barramento de endereços com 16 bits (Ferreira et al., 91c, pp. 17-22).

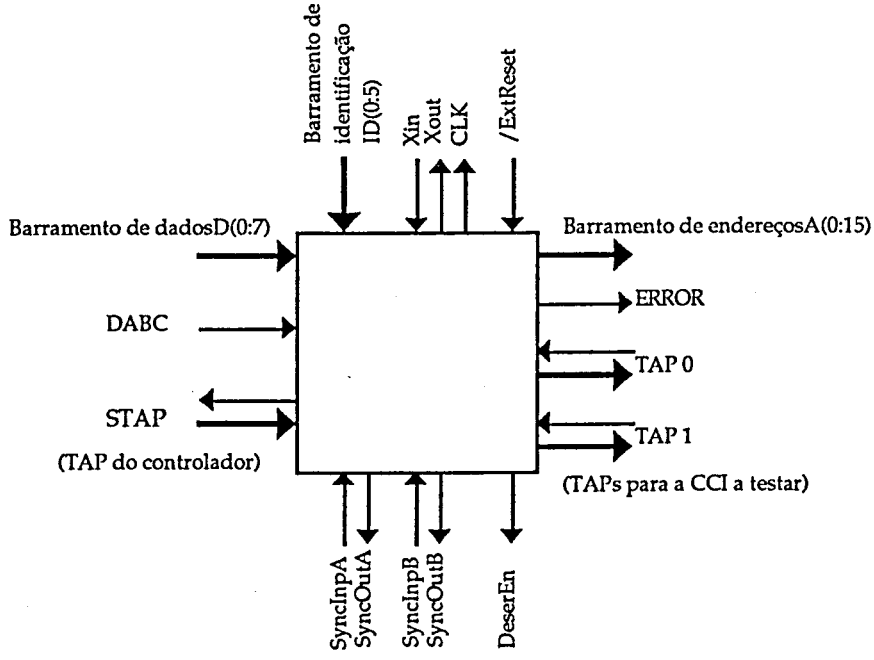


Fig.6.2: O conjunto de pinos do controlador residente.

A questão do encapsulamento adoptado para o controlador residente deverá também ser levada em consideração, uma vez que um aumento na largura do barramento de endereços poderá tornar necessário um encapsulamento com um maior número de pinos.

Os factores descritos conduzem à conclusão que a largura mínima recomendável para o barramento de endereços será de 16 bits, sendo de considerar a hipótese de aumentar este valor, caso o encapsulamento adoptado o permita.

Finalmente, e de acordo com as considerações efectuadas quando se optou pela extensão do BST como barramento de teste entre CCIs, a identificação de cada uma não

requererá mais do que 6 bits, pelo que esta é a largura seleccionada para o barramento de identificação (*ID bus*).

A figura 6.2 mostra o conjunto de pinos que fazem parte do controlador residente, e cuja descrição se encontra sintetizada no quadro 6.1.

Funcionalidade definida para o conjunto de pinos do controlador residente		
Designação	Núm. de pinos	Funcionalidade
A(0:15)	16	Barramento de endereços
D(0:7)	8	Barramento de dados
ID(0:5)	6	Barramento de identificação
TAP 0	5	Controlo de uma cadeia BST da CCI a testar
TAP 1	5	Controlo da outra cadeia BST da CCI a testar
STAP	5	TAP da infraestrutura BST do controlador
Sync...	4	Entradas e saídas de sincronismo
Xin, Xout, CLK	3	Relógio do controlador (Xin e Xout para um cristal exterior, CLK como saída do relógio interno)
/ExtReset	1	Entrada de inicialização para o núcleo que constitui o processador dedicado
ERROR	1	Saída que indica o resultado do teste
DeserEn	1	Saída que indica quando tem lugar uma operação de deslocamento com comparação
DABC	1	Entrada que permite colocar as saídas que controlam o TAP 0 e o TAP 1 em terceiro estado. O acesso directo às cadeias BST da CCI a testar é então possível (*Direct Access to the Board Chains*)

Quadro 6.1: Síntese da funcionalidade definida para os pinos do controlador residente.

Deve notar-se que a caracterização global do controlador residente apresentado distingue entre o processador de teste propriamente dito, dedicado à exploração das potencialidades proporcionadas pela infraestrutura BST da CCI a testar, e a infraestrutura BST do próprio controlador, destinada a permitir a interacção com o nível hierárquico superior.

6.2. ARQUITECTURA DO PROCESSADOR DE TESTE

Esta secção apresenta uma arquitectura para o processador de teste referido na figura 6.1, destinada a garantir a completa exploração das potencialidades proporcionadas pela infraestrutura BST. Para além dos blocos genéricos, tais como o apontador de programa, o registo de instrução, ou a unidade de descodificação e controlo de execução para cada instrução, a identificação dos restantes blocos que constituirão esta arquitectura pode ser feita por análise das operações elementares descritas nos quadros 5.2 a 5.4, tal como se apresenta a seguir:

i) As operações elementares para o controlo da infraestrutura BST (quadro 5.2) requerem a existência de blocos que se responsabilizem pelo deslocamento de vectores para a cadeia BST (controlo da linha TDO para os dois TAPs comandados pelo processador), pela recepção das respostas aos vectores aplicados anteriormente (controlo da linha TDI proveniente dos dois TAPs comandados pelo processador), e ainda de um outro bloco que se responsabilize por seleccionar qual a cadeia BST que deverá ser controlada (TAP 0 ou TAP 1).

ii) As operações elementares para o protocolo de sincronismo com equipamentos exteriores (quadro 5.3) requerem por sua vez a existência de um bloco que controle as saídas de sincronismo existentes, enquanto as entradas de sincronismo deverão proporcionar a informação necessária ao bloco responsável pelo controlo de execução para cada instrução.

iii) Finalmente, as operações elementares para o controlo dos recursos internos, e controlo do fluxo do programa de teste (quadro 5.4), identificam a necessidade de um contador interno, e de um registo de estado, que incluirá o conjunto de *flags* de erro destinadas a memorizar a ocorrência de diferentes tipos de faltas, e da cadeia BST em que foram detectadas.

A figura 6.3 ilustra a arquitectura definida para este processador, onde se identificam todos os blocos anteriormente referidos. Repare-se que o conteúdo do registo de estado deverá ser acessível através da infraestrutura BST do controlador residente, pelo que este acesso está também representado.

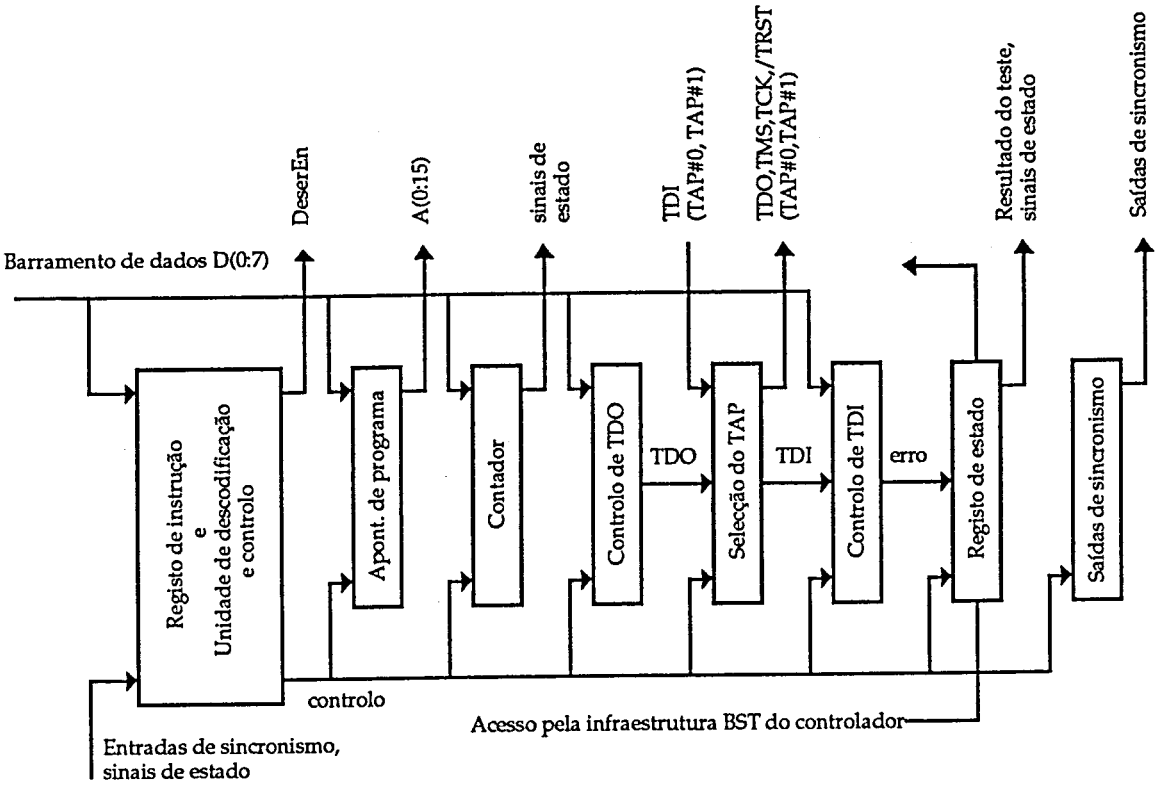


Fig.6.3: Arquitectura do processador dedicado ao controlo da infraestrutura BST da CCI a testar.

Os parágrafos apresentados a seguir efectuarão uma caracterização individual dos blocos ilustrados na figura 6.3.

Registo de instrução, e unidade de descodificação e controlo

Este bloco pode representar-se como se ilustra na figura 6.4, onde se distinguem três componentes principais:

i) O registo de instrução, destinado a receber o código de cada instrução a executar, e que é constituído por uma *latch* de 5 bits.

ii) Uma máquina de estados, que se responsabiliza por gerar a necessária sequência de estados, de acordo com o código presente no registo de instrução. Esta máquina de estados é constituída por uma *latch* de 5 bits, que indica o estado actual, e por um circuito combinacional que determina qual o estado seguinte (*nextst*).

iii) Um circuito combinacional responsável pela geração dos sinais de controlo para todos os blocos do processador, de acordo com o estado actual, e com o código de instrução presente (*exectr*).

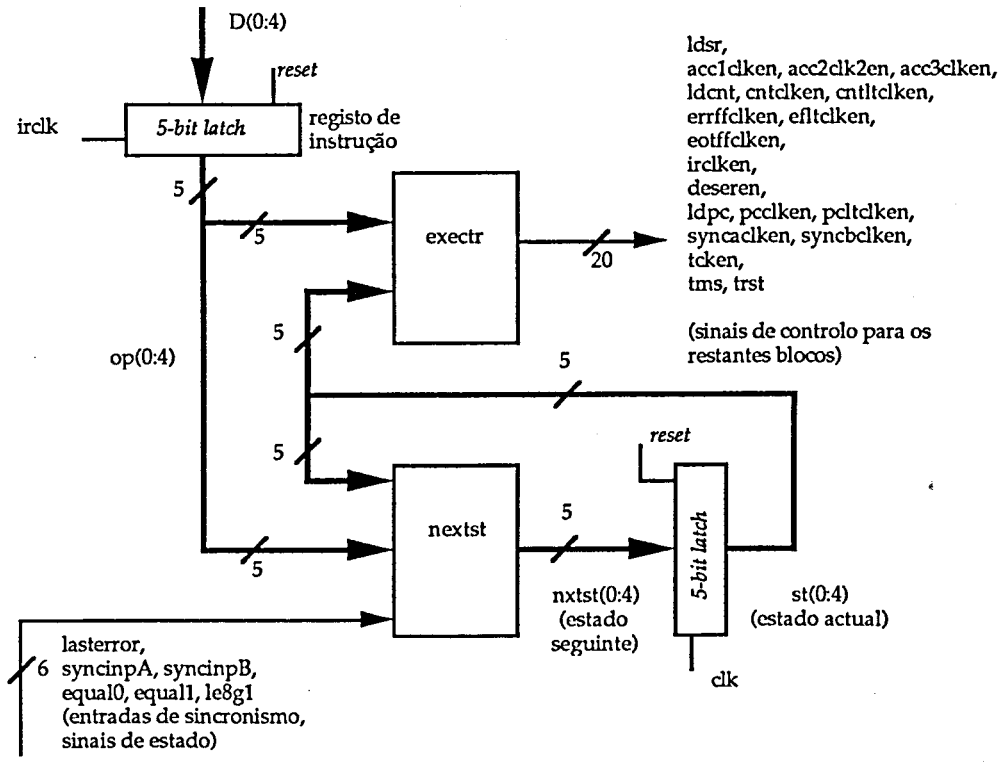


Fig.6.4: Registo de instrução, e unidade de descodificação e controlo.

Repare-se que este bloco deve igualmente responsabilizar-se pela leitura do código da instrução seguinte, o que deve ter lugar sempre que a máquina de estados que controla a execução de cada instrução retornar ao estado inicial.

Apontador do programa (PC, *Program Counter*)

Este bloco é responsável pelo controlo do barramento de endereços, e constitui o único apontador para a memória com o programa de teste. Isto significa que todos os ciclos de acesso à memória (leitura de dados, ou de códigos de instrução) deverão posteriormente incrementar este apontador, de modo a preparar o próximo acesso.

O apontador do programa está ilustrado na figura 6.5, e é constituído por um contador síncrono de 16 bits, com possibilidade de carga paralela, para suportar as instruções de salto condicional. Uma vez que a largura do barramento de dados é de 8 bits, torna-se necessária a existência da *latch* ilustrada na figura 6.5, de modo a permitir que a carga dos dois andares de 8 bits (PC LSB, PC MSB) tenha lugar em simultâneo. A constituição típica para cada um dos 16 andares que constituem o apontador de programa está ilustrada na figura 6.6, onde i representa a ordem do andar.

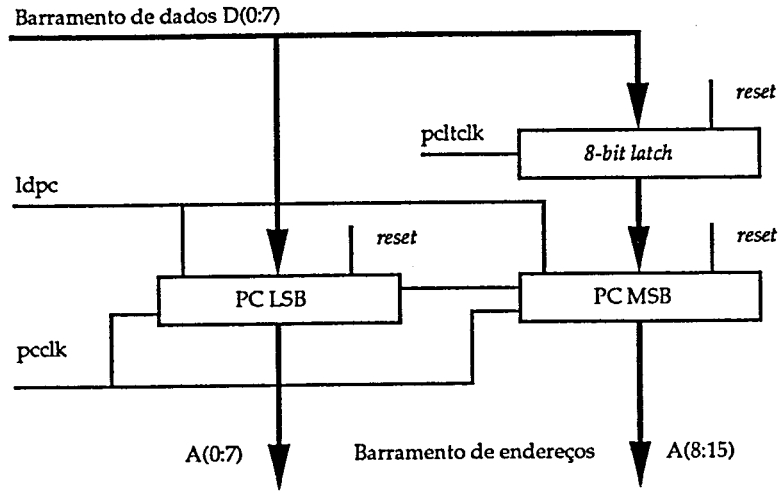


Fig.6.5: Apontador do programa.

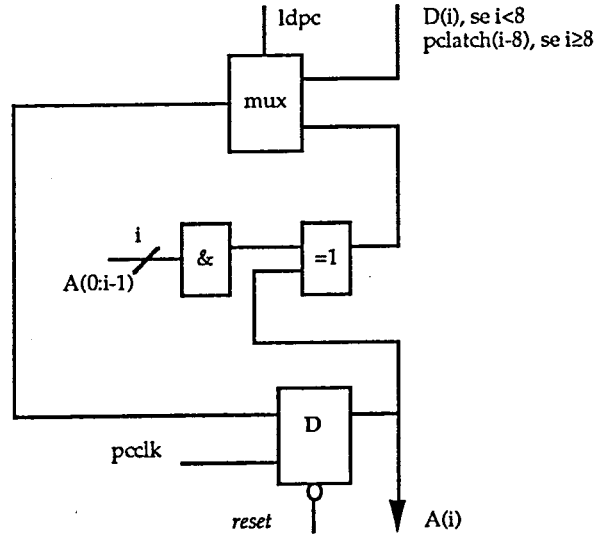


Fig.6.6: Constituição típica de cada andar, no apontador de programa.

Contador interno

O contador interno é decrementado directamente pela unidade de descodificação e controlo, durante a execução das instruções que requerem um determinado número de impulsos no relógio de teste (TCK) da cadeia BST a controlar.

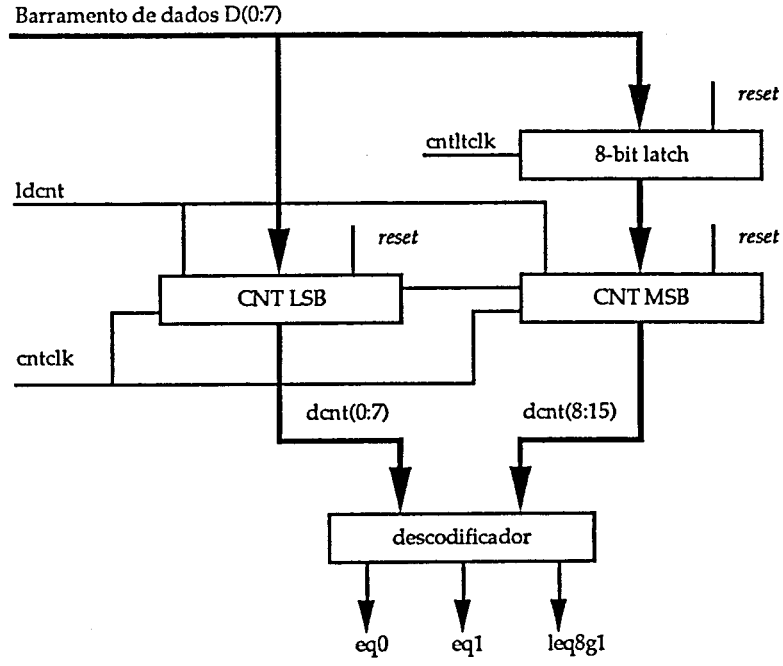


Fig.6.7: Contador interno.

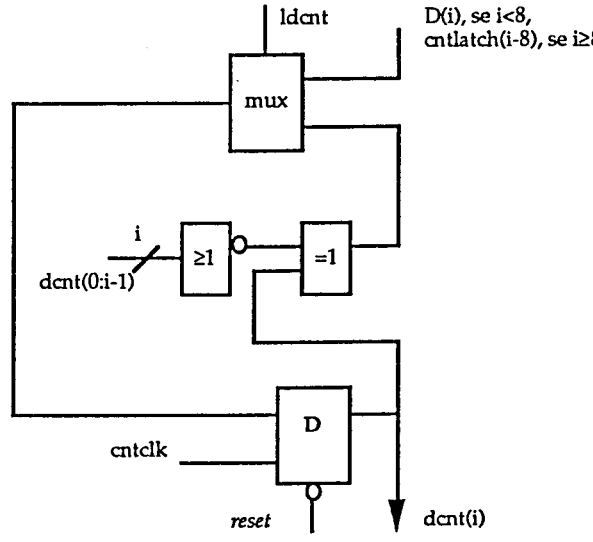


Fig.6.8: Constituição típica de cada andar, no contador interno.

Este bloco está ilustrado na figura 6.7, e é constituído por um contador descendente síncrono de 16 bits, com carga paralela. A *latch* ilustrada na figura 6.7 permite que a carga dos dois andares de 8 bits seja efectuada em paralelo. Repare-se que este bloco proporciona à unidade de descodificação e controlo vários sinais que indicam a proximidade ao fim da

contagem. A constituição típica para cada um dos 16 andares que constituem este contador está ilustrada na figura 6.8, onde **i** representa a ordem do andar.

Controlo da saída série (TDO) para cada TAP

Este bloco destina-se a permitir a inserção de vectores no interior da cadeia BST da CCI a testar, e é constituído por um registo de deslocamento de 8 bits, com carga paralela, tal como se ilustra na figura 6.9.

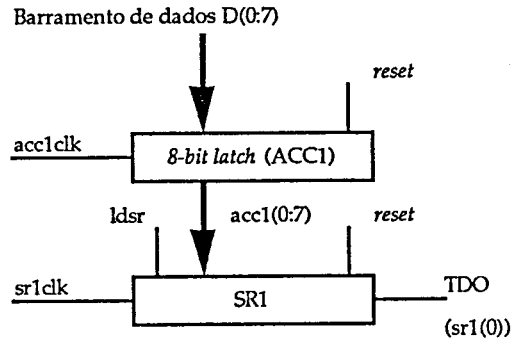


Fig.6.9: Controlo da saída série (TDO) para cada TAP.

Este registo é carregado através de uma *latch* de 8 bits, que tem por objectivo permitir que o processador execute concorrentemente a carga dos próximos 8 bits, durante o deslocamento dos actuais. Esta possibilidade tem um interesse particular no que respeita às operações de deslocamento com comparação, e onde os três acessos à memória (valores a deslocar, valores a comparar, e máscaras de comparação) podem deste modo ser efectuados durante o deslocamento dos 8 bits actuais, o que optimiza a velocidade de inserção de vectores na cadeia BST. A constituição típica para cada um dos 8 andares que constituem este registo de deslocamento está ilustrada na figura 6.10.

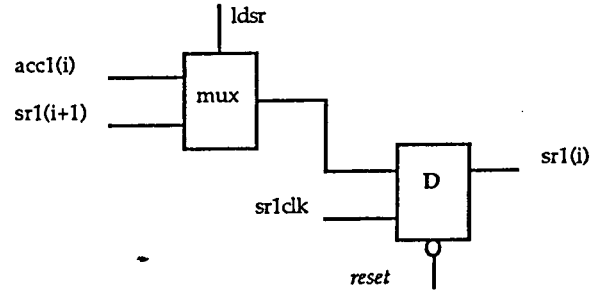


Fig.6.10: Constituição típica de cada andar, no registo de deslocamento.

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 BST, 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 funcionalidade pretendida para este bloco é garantida por dois registos de deslocamento com carga paralela, e por um circuito de comparação, tal como se ilustra na figura 6.11.

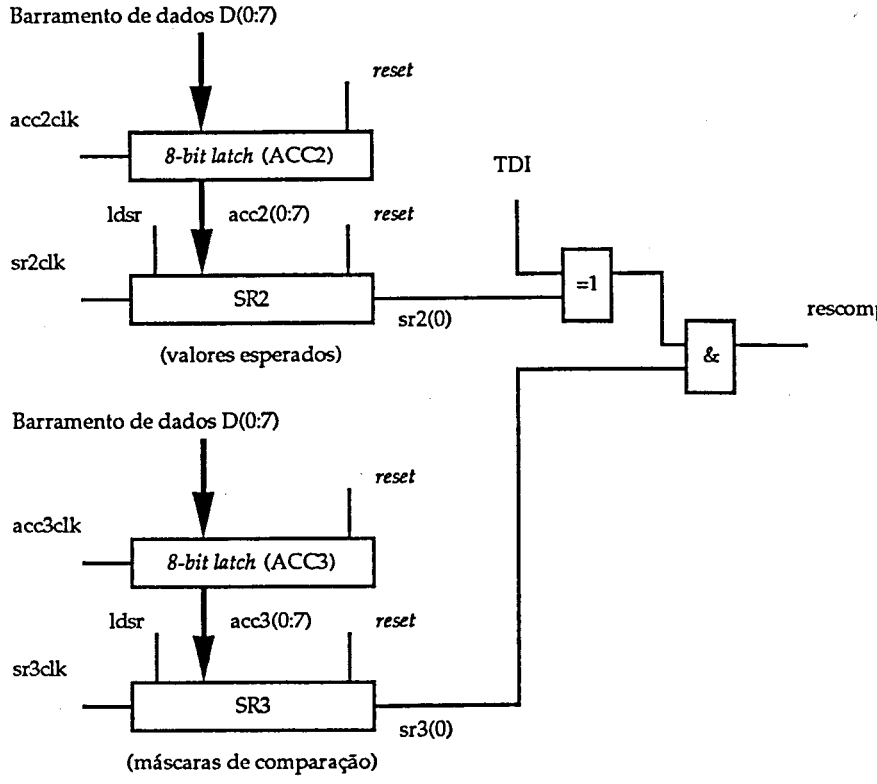


Fig.6.11: Controlo da entrada série (TDI) proveniente de cada TAP.

Cada registo de deslocamento é carregado a partir de uma *latch* de 8 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. Repare-se que a colocação de um bit em **0**, na palavra que constitui a máscara de comparação, tem por efeito impedir a detecção de uma diferença entre o valor esperado e o valor lido. A constituição típica para cada um dos 8 andares de cada registo de deslocamento está ilustrada na figura 6.12.

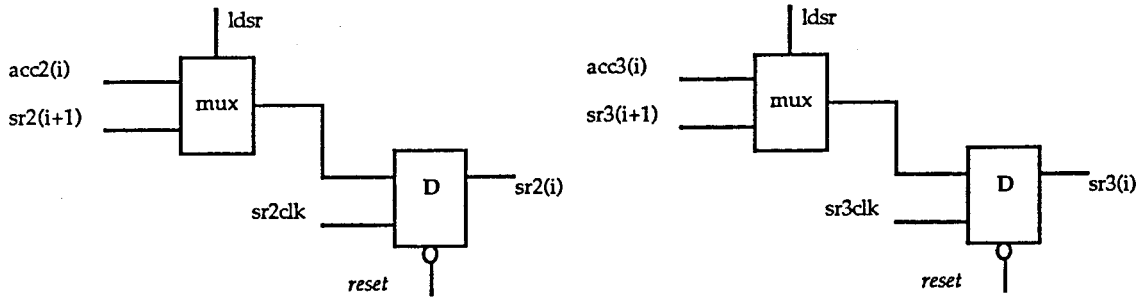
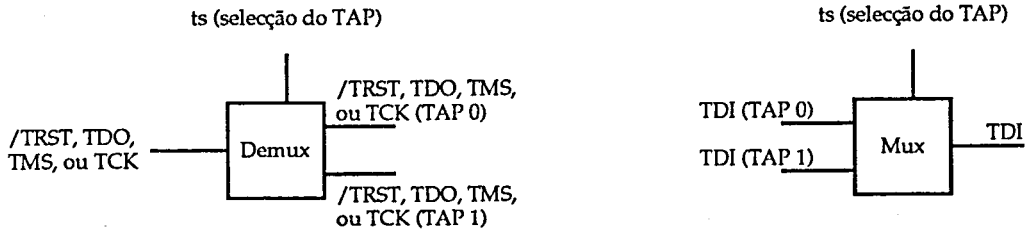


Fig.6.12: Constituição típica de cada andar, no bloco de controlo da entrada série (TDI).

Selecção da cadeia BST a controlar

Este bloco destina-se a efectuar a multiplexagem entre os recursos proporcionados pelo processador, e os dois TAPs disponíveis, e pode ser representado como se ilustra na figura 6.13.



(a) Selecção do conjunto /TRST, TDO, TMS, TCK.

(b) Selecção de TDI.

Fig.6.13: Selecção da cadeia BST a controlar.

Registo de estado

Este bloco contem dois conjuntos de 8 *flags* de erro, e uma *flag* de fim-de-teste (actuada pela execução da instrução que termina o programa de teste), e tem por objectivo proporcionar esta informação a um controlador residente a nível hierárquico superior. Cada grupo de 8 *flags* de erro diz respeito exclusivamente a um dos dois TAPs comandados pelo processador, efectuando-se automaticamente a comutação de um grupo para o outro, quando é executada a instrução que selecciona a cadeia BST a controlar. A *flag* de erro actuada pela ocorrência de uma diferença válida (não mascarada) entre o valor esperado e o valor lido (numa operação de deslocamento com comparação) será a que foi previamente seleccionada pela instrução correspondente (a inicialização do processador selecciona a *flag*

de erro 0, no TAP 0). A constituição deste bloco, no que respeita à memorização de erros, está ilustrada na figura 6.14.

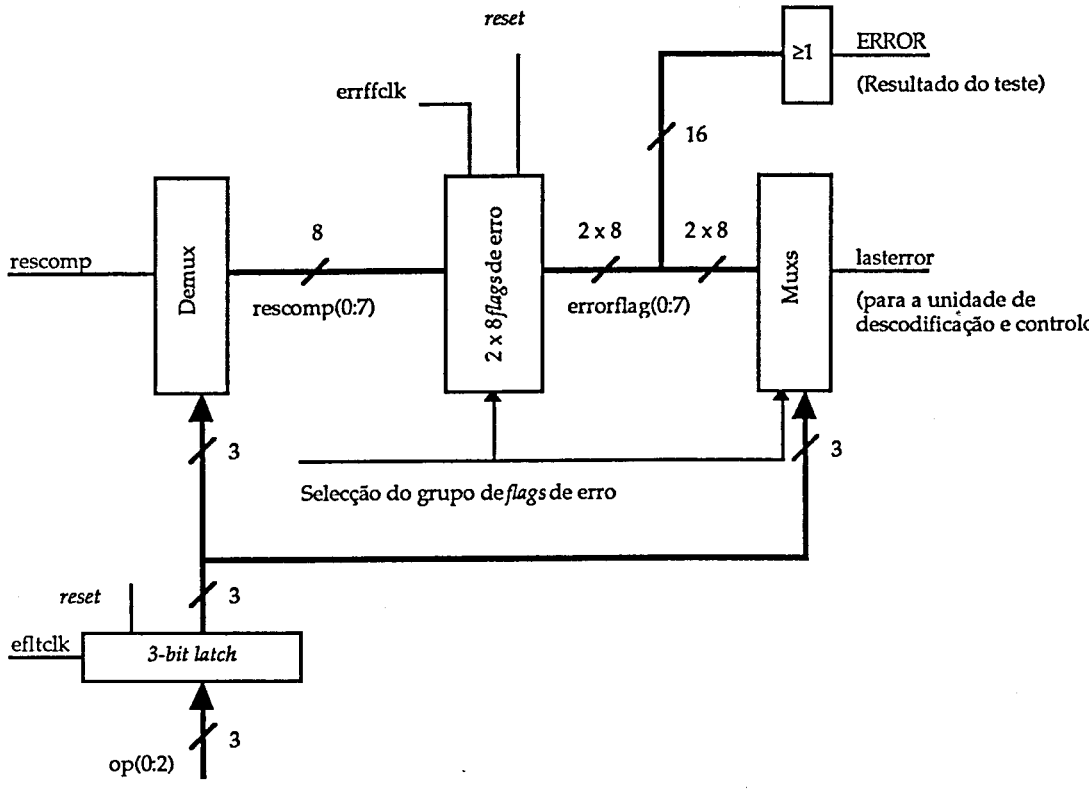


Fig.6.14: Memorização de erros no registo de estado.

Repare-se que basta que uma das *flags* de erro tenha sido actuada, para que a saída que apresenta o resultado do teste produza a indicação de erro. Por sua vez, a unidade de descodificação e controlo tem sempre disponível a informação que indica o estado da *flag* de erro seleccionada, na cadeia BST activa.

Saídas de sincronismo

Este bloco é essencialmente constituído por dois *flip-flops* que permitem memorizar, nos pinos de sincronismo, o valor lógico especificado pela execução da operação elementar que controla estas saídas. As correspondentes entradas de sincronismo proporcionam esta informação directamente à unidade de descodificação e controlo.

6.3. ESPECIFICAÇÃO DAS INSTRUÇÕES SUPORTADAS

Esta secção efectuará a apresentação do conjunto de instruções suportadas pelo processador, de acordo com as operações elementares que foram identificadas nos quadros 5.2 a 5.4.

6.3.1. Definição do conjunto de instruções

O conjunto de instruções suportado tem uma correspondência directa com os três tipos de operações elementares que foram identificadas no capítulo anterior, e que está traduzida nos quadros 6.2 a 6.4.

Instruções para o controlo da infraestrutura BST	
Operação elementar	**Instrução**
Aplicar um impulso a 0 na linha /TRST	TRST
Aplicar um impulso no relógio de teste, mantendo a linha TMS no valor pretendido.	TMS0, TMS1
Efectuar o deslocamento, sem comparação, de um vector para a cadeia BST.	NSHF
Efectuar o deslocamento, com comparação, de um vector para a cadeia BST.	NSHFCP
Aplicar N impulsos no relógio de teste, mantendo a linha TMS em 0.	NTCK
Seleccionar qual a cadeia BST a controlar.	SELTAP0, SELTAP1

Quadro 6.2: Instruções para o controlo da infraestrutura BST.

Instruções para implementação do protocolo de sincronismo com o exterior	
Operação elementar	**Instrução**
Colocar na saída de sincronismo o valor pretendido.	SSA0, SSA1, SSB0, SSB1
Esperar até que a entrada de sincronismo se encontre no valor pretendido.	WSA0, WSA1, WSB0, WSB1

Quadro 6.3: Instruções relativas ao protocolo de sincronismo com o exterior.

Instruções para o controlo dos recursos internos, e do fluxo do programa de teste	
Operação elementar	Instrução
Carregar o contador interno com o número de impulsos pretendidos no relógio de teste.	LD CNT, N
Seleccionar a *flag* de erro a ser actuada pela posterior detecção de uma falta.	SERFLG0, ..., SERFLG7
Implementar um salto condicional, de acordo com o estado da *flag* de erro seleccionada.	JPE Endereço, JPNE Endereço
Concluir a execução do programa de teste.	HALT

Quadro 6.4: Instruções para o controlo dos recursos internos, e do fluxo do programa de teste.

6.3.2. Caracterização individual

A caracterização individual de cada instrução inclui uma descrição sucinta do seu efeito, e a apresentação do respectivo diagrama de transição de estados. A caracterização de algumas instruções poderá ainda incluir elementos adicionais, tais como a descrição do armazenamento de dados em memória, ou a ilustração de diagramas temporais. Após a caracterização individual de todas as instruções, será apresentado um quadro que indicará o número de palavras de memória requeridas por cada uma, e o número de ciclos de relógio necessários à respectiva execução.

6.3.2.1. Instruções para o controlo da infraestrutura BST

Apresenta-se nesta secção a caracterização individual das instruções destinadas a controlar a infraestrutura BST da CCI, que foram descritas no quadro 6.2.

TRST

Esta instrução provoca uma inicialização da lógica BST em todos os componentes que disponham de um pino /TRST ligado a esta saída, na cadeia BST seleccionada. Repare-se que a inicialização de toda a infraestrutura BST só será possível por este processo se todos os componentes BST dispuserem de um pino /TRST ligado a esta saída. O correspondente diagrama de transição de estados está ilustrado na figura 6.15.

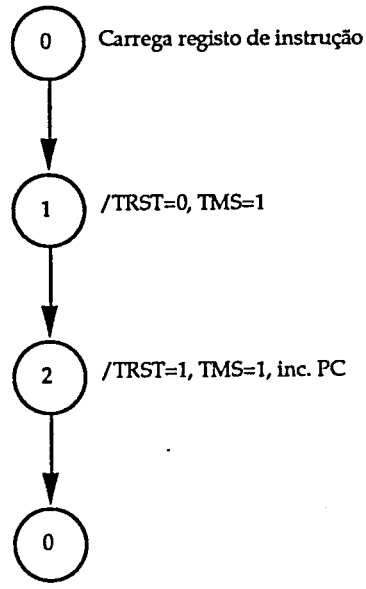


Fig.6.15: Diagrama de transição de estados para a instrução TRST.

De acordo com a especificação apresentada em (IEEE Std 1149.1, 90, p. 3-5), a linha TMS deve ser mantida em **1** durante a transição ascendente da linha /TRST, pelo que o diagrama temporal correspondente à execução desta instrução é o que se ilustra na figura 6.16.

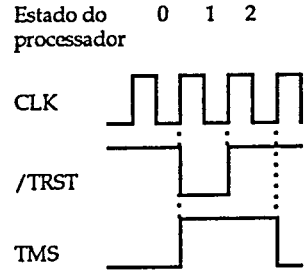


Fig.6.16: Diagrama temporal para a execução da instrução TRST.

O nível lógico **0** é temporariamente aplicado nas linhas /TRST dos dois TAPs comandados pelo processador sempre que a alimentação é ligada, com o objectivo de permitir que se provoque a inicialização da lógica BST (*power-up reset*), nos componentes em que esta não ocorra automaticamente (IEEE Std 1149.1, 90, p. 5-16).

TMS0, TMS1

Esta instrução aplica um impulso no relógio de teste (TCK), enquanto a linha TMS é

mantida em 0 (TMS0), ou em 1 (TMS1). O correspondente diagrama de transição de estados está ilustrado na figura 6.17.

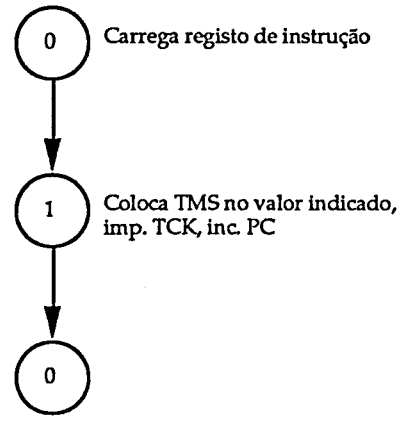


Fig.6.17: Diagrama de transição de estados para a instrução TMS.

Tal como especificado em (IEEE Std 1149.1, 90, p. 5-2), todas as transições de estado deverão ocorrer com a transição ascendente do relógio de teste (TCK), pelo que o diagrama temporal que caracteriza a execução desta instrução é o que se ilustra na figura 6.18.

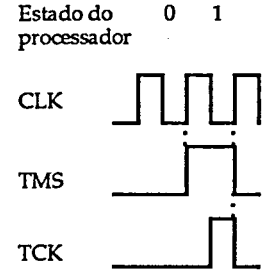


Fig.6.18: Diagrama temporal para a execução da instrução TMS1.

NSHF

Esta instrução faz com que a sequência de N bits que se segue ao respectivo código de instrução, na memória do programa, seja deslocada através da saída série (TDO) para o TAP seleccionado. N representa o conteúdo do contador interno (16 bits), pelo que esta instrução deverá ser precedida por uma outra que carregue este contador com o valor pretendido (LD CNT, N). O diagrama de estados para a execução desta instrução está ilustrado na figura 6.19.

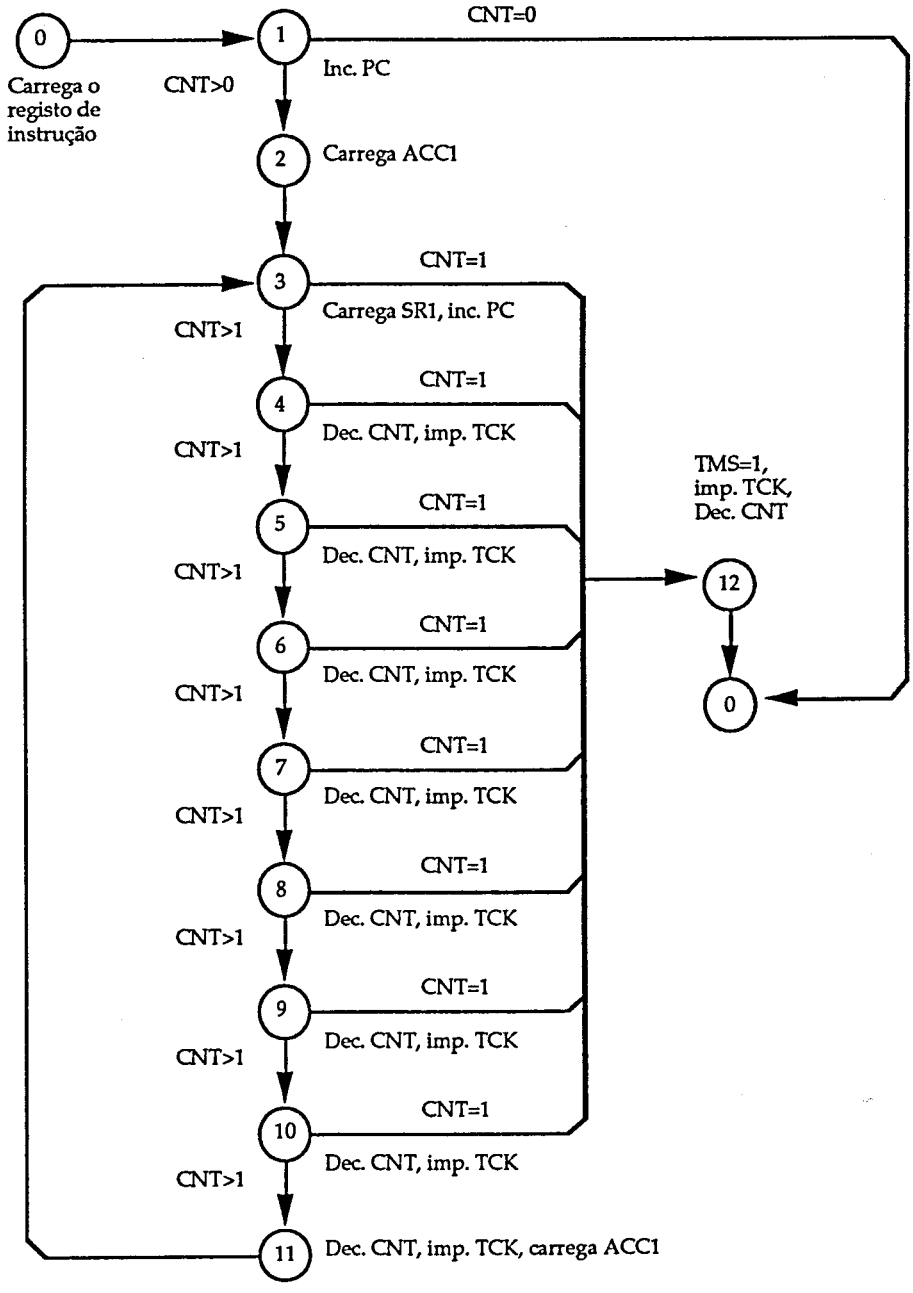


Fig.6.19: Diagrama de transição de estados para a instrução NSHF.

A linha TMS será mantida em **0** durante os primeiros (N-1) impulsos no relógio de teste (TCK), de modo a que o controlador do TAP, na lógica BST de cada componente, se conserve no estado *Shift-DR* (*Shift-IR*). O último impulso aplicado no relógio de teste terá a linha TMS em **1**, de modo a que ocorra a transição para o estado *Exit1-DR* (*Exit1-IR*). A transição ascendente na linha TMS terá lugar com a transição descendente do (N-1)º impulso no relógio de teste (IEEE Std 1149.1, 90, p. 5-2). Repare-se que o controlador do TAP, na lógica BST de cada componente, se deverá já encontrar no estado *Shift-DR* (*Shift-*

IR), anteriormente à execução desta instrução. As transições na linha TDO (actualização do valor a deslocar para a cadeia BST) ocorrem com as transições descendentes no relógio de teste (IEEE Std 1149.1, 90, pp. 5-4 e 5-6), tal como se ilustra na figura 6.20, que apresenta o diagrama temporal para a execução da instrução NSHF \$55,\$CA† (assume-se que o contador tinha sido anteriormente carregado através da instrução LD CNT, 10).

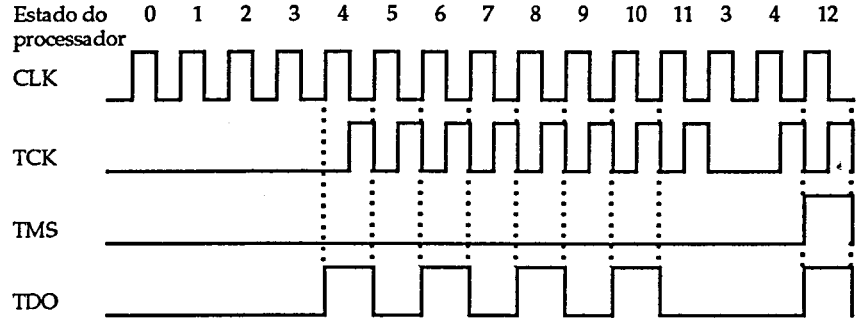


Fig.6.20: Diagrama temporal para a execução da instrução NSHF \$55,\$CA (N=10).

Este diagrama temporal, associado ao diagrama de transição de estados apresentado na figura 6.19, ilustra a execução concorrente da operação de deslocamento, e da leitura de novos valores a deslocar, e que permite ter um atraso de apenas um único impulso no relógio de teste (TCK), por cada 8 bits a deslocar (são necessários 9 ciclos de relógio (CLK) por cada 8 impulsos no relógio de teste (TCK)).

Os dados a deslocar para o interior da cadeia BST deverão estar armazenados como se ilustra na figura 6.21. Estes dados serão deslocados da esquerda para a direita (do bit mais significativo para o menos significativo), pelo que o primeiro bit a deslocar será o bit menos significativo (D0) da palavra que se segue ao código desta instrução.

Endereço:	D7	D6	D5	D4	D3	D2	D1	D0	
(PC)	Código de NSHF								
(PC)+1	0	1	0	1	0	1	0	1	55
(PC)+2	1	1	0	0	1	0	1	0	CA
...									

Fig.6.21: Armazenamento dos dados na instrução NSHF \$55,\$CA.

† O símbolo \$ é aqui usado para indicar a representação hexadecimal.

NSHFCP

Esta instrução faz com que uma sequência de N bits, posterior ao respectivo código de instrução, seja deslocada através da saída série (TDO) para o TAP seleccionado. O armazenamento em memória desta sequência deve no entanto ser alternado com o de duas outras sequências, palavra a palavra, destinadas respectivamente à indicação dos valores esperados na correspondente entrada série (TDI), e das máscaras de comparação a utilizar. N representa o conteúdo do contador interno (16 bits), pelo que esta instrução deverá ser precedida por uma outra que carregue este contador com o valor pretendido. A primeira comparação de um bit que difira do seu valor esperado, e cuja máscara permita a comparação, actuará a *flag* de erro seleccionada, e fará com que a saída do processador que indica a ocorrência de erro passe ao estado activo. A instrução NSHFCP terminará a sua execução em qualquer dos casos, mas o estado da *flag* de erro seleccionada poderá ser posteriormente testado por uma operação de salto condicional.

Os impulsos no relógio de teste (TCK), aplicados durante a execução da instrução NSHFCP, serão acompanhados pela aplicação de um **1** na saída DeserEn, com o objectivo de indicar para o exterior que está a ocorrer uma operação de deslocamento com comparação. O diagrama de transição de estados para a execução desta instrução está ilustrado na figura 6.22.

A linha TMS será mantida em **0** durante os primeiros (N-1) impulsos no relógio de teste, de modo a que o controlador do TAP, na lógica BST de cada componente, se conserve no estado *Shift-DR* (*Shift-IR*). O último impulso aplicado no relógio de teste terá a linha TMS em **1**, de modo a que ocorra a transição para o estado *Exit1-DR* (*Exit1-IR*). A transição ascendente na linha TMS terá lugar com a transição descendente do (N-1)º impulso no relógio de teste (IEEE Std 1149.1, 90, p. 5-2). Repare-se que o controlador do TAP, na lógica BST de cada componente, se deverá já encontrar no estado *Shift-DR* (*Shift-IR*), anteriormente à execução desta instrução.

As transições na linha TDO (actualização do valor a deslocar para a cadeia BST) ocorrem com as transições descendentes no relógio de teste (IEEE Std 1149.1, 90, pp. 5-4 e 5-6), tal como se ilustra na figura 6.23, que apresenta o diagrama temporal para a execução da instrução NSHFCP $0C,$C0,$E7,$0C,$80,$03 (assume-se que o contador interno tinha sido anteriormente carregado através da instrução LD CNT, 10).

Também na instrução NSHFCP, e a exemplo do que sucedia já na instrução NSHF, a execução concorrente da operação de deslocamento, e da leitura dos novos valores a deslocar, permite ter apenas um atraso de um impulso no relógio de teste, por cada 8 bits a deslocar (serão necessários 9 ciclos de relógio (CLK) por cada 8 impulsos no relógio de teste (TCK)).

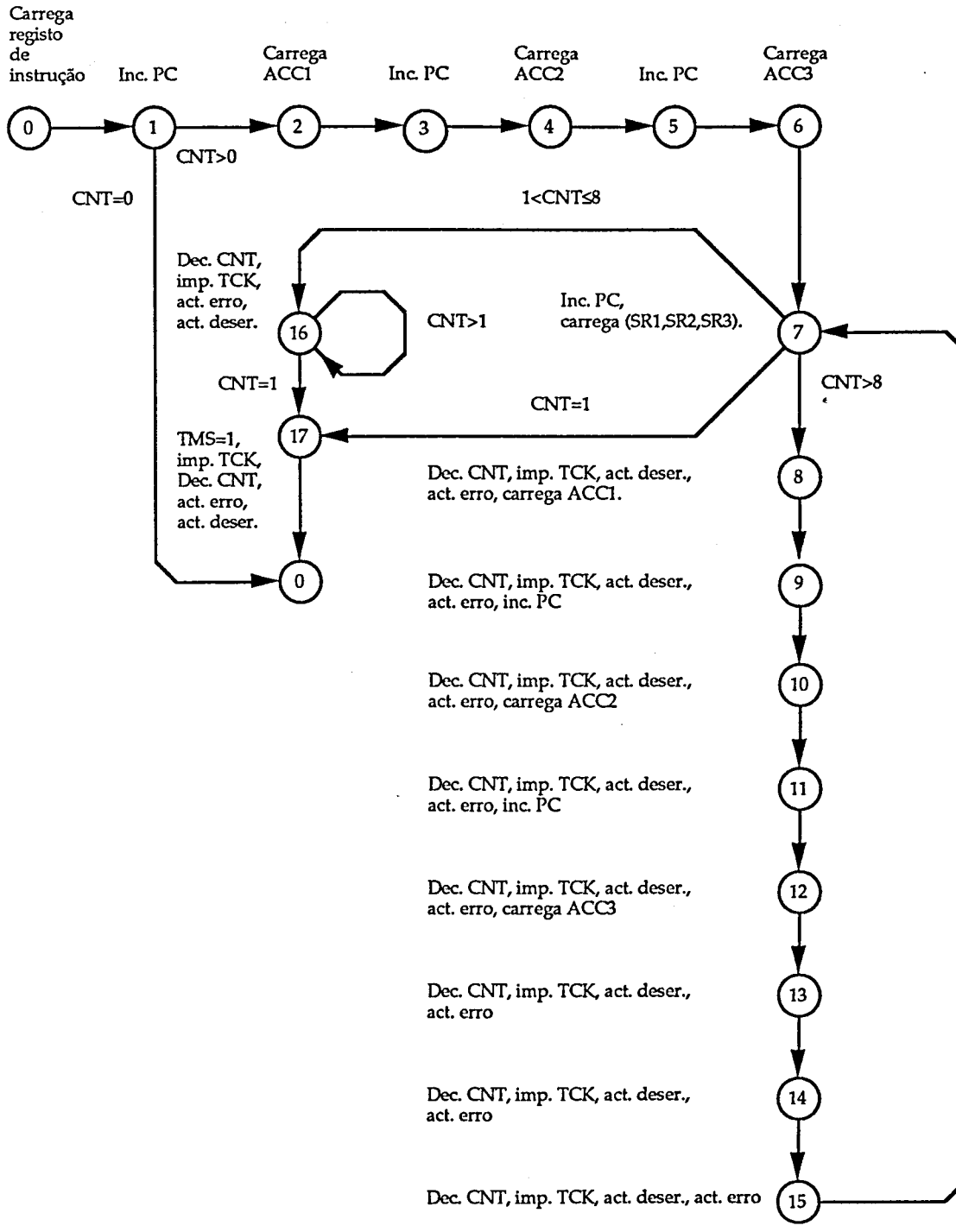


Fig.6.22: Diagrama de transição de estados para a instrução NSHFCP.

Os dados correspondentes a esta instrução deverão estar armazenados em memória de modo a que surja alternadamente uma palavra de cada um dos três tipos (valores a deslocar, valores esperados, e máscaras de comparação), tal como se ilustra na figura 6.24. O deslocamento destes valores procede da esquerda para a direita (do bit mais significativo

para o menos significativo), pelo que o primeiro bit a ser deslocado será o bit menos significativo (D0) da palavra que se segue ao código desta instrução. A primeira palavra que se segue ao código da instrução pertence ao conjunto dos valores a deslocar, a segunda ao conjunto dos valores esperados, e a terceira às máscaras de comparação, mantendo-se daqui por diante esta mesma sequência.

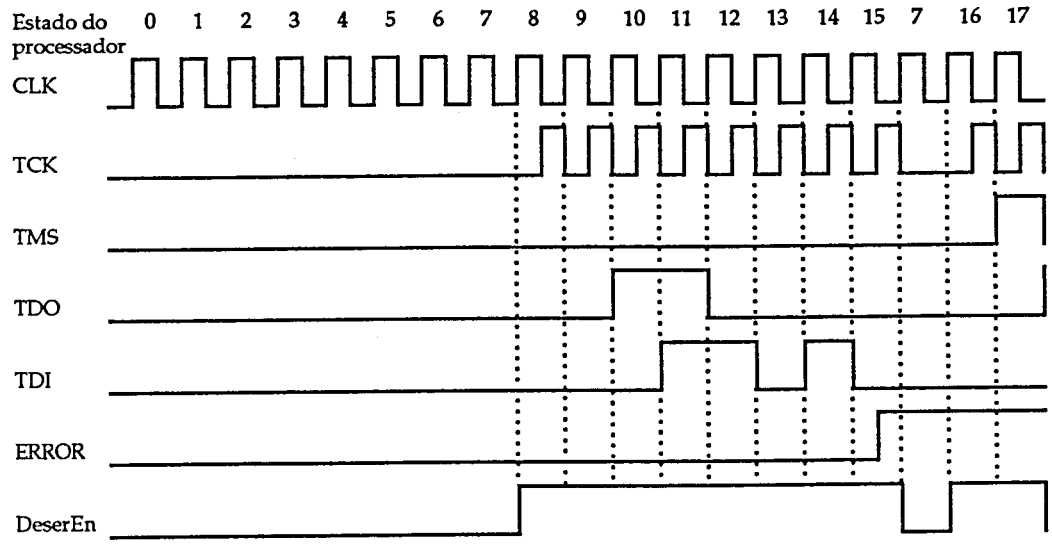


Fig.6.23: Diagrama temporal na execução da instrução NSHFCP $0C,$C0,$E7,$0C,$80,$03 (N=10).

Endereço:	D7	D6	D5	D4	D3	D2	D1	D0		
(PC)	Código de NSHFCP									
(PC)+1	0	0	0	0	1	1	0	0	0C	(a deslocar)
(PC)+2	1	1	0	0	0	0	0	0	C0	(valores esperados)
(PC)+3	1	1	1	0	0	1	1	1	E7	(máscara de comparação)
(PC)+4	0	0	0	0	1	1	0	0	0C	(a deslocar)
(PC)+5	1	0	0	0	0	0	0	0	80	(valores esperados)
(PC)+6	0	0	0	0	0	0	1	1	03	(máscara de comparação)

Fig.6.24: Armazenamento dos dados na instrução NSHFCP $0C,$C0,$E7,$0C,$80,$03 (N=10).

NTCK

Esta instrução destina-se a permitir a execução das funções de auto-teste em componentes BST que disponham desta possibilidade, e terá por resultado a aplicação de N impulsos no relógio de teste (TCK), enquanto a linha TMS é mantida em 0 (de modo a que o controlador do TAP pertencente à lógica BST de cada componente se mantenha no estado *Run Test / Idle*). N representa o conteúdo do contador interno (de 16 bits), pelo que esta instrução deverá ser precedida por uma outra que carregue este contador com o valor pretendido (LD CNT, N). Repare-se que o controlador do TAP, em cada componente BST, se deverá encontrar já no estado *Run Test / Idle*. O diagrama de transição de estados para a execução desta instrução está ilustrado na figura 6.25.

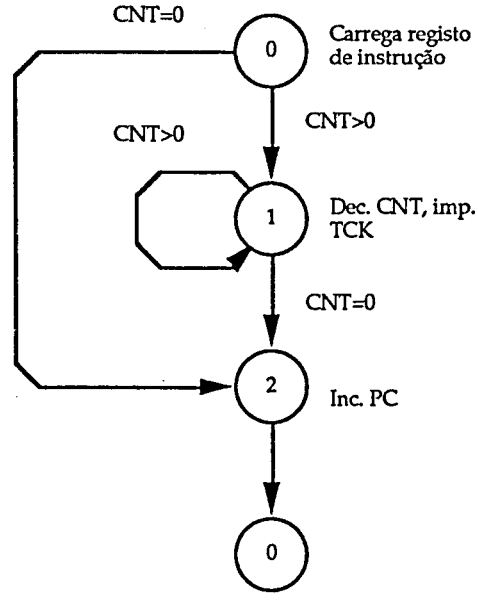


Fig.6.25: Diagrama de transição de estados para a instrução NTCK.

De acordo com o diagrama de transição de estados apresentado, a frequência dos impulsos gerados no relógio de teste (TCK) será a mesma do relógio do processador (CLK).

SELTAP0, SELTAP1

Estas instruções permitem seleccionar qual dos dois TAPs será controlado pelas instruções seguintes, e destinam-se a permitir o controlo de CCIs com duas cadeias BST. O TAP seleccionado poderá apenas ser modificado pela reinicialização do processador, ou por outra instrução SELTAP posterior. O diagrama de transição de estados para esta instrução está ilustrado na figura 6.26.

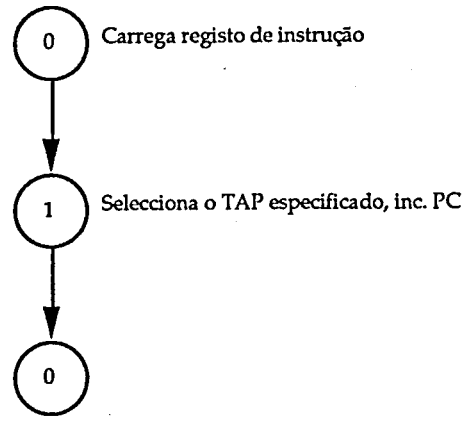


Fig.6.26: Diagrama de transição de estados para a instrução SELTAP.

6.3.2.2. Instruções para implementação do protocolo de sincronismo com o exterior

Apresenta-se nesta secção a caracterização individual das instruções destinadas a implementar o protocolo de sincronismo com o exterior, que foram descritas no quadro 6.3.

SSA0, SSA1, SSB0, SSB1

Estas instruções destinam-se a actuar sobre as saídas de sincronismo, e têm por consequência que o valor especificado (0, 1) será aplicado à saída de sincronismo indicada (A, B).

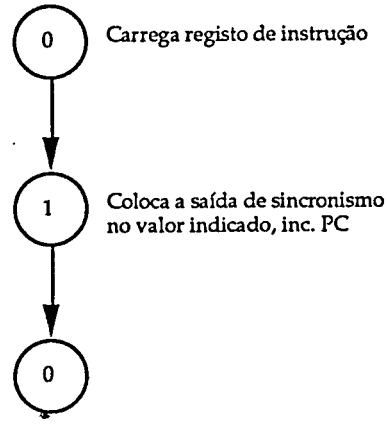


Fig.6.27: Diagrama de transição de estados para as instruções SS.

O valor aplicado na saída de sincronismo poderá apenas ser modificado por uma reinicialização do processador, ou por outra instrução SS posterior. Quando usadas em conjunto com as instruções WS (a descrever a seguir), permitem implementar um protocolo de sincronismo assíncrono com equipamentos exteriores. O diagrama de transição de estados para estas instruções está ilustrado na figura 6.27.

WSA0, WSA1, WSB0, WSB1

Estas instruções retêm o processador no mesmo estado por um número indefinido de ciclos de relógio, até que a entrada de sincronismo (A, B) se encontre no valor especificado (0, 1). Quando usadas em conjunto com as instruções SS, permitem implementar um protocolo de sincronismo assíncrono com equipamentos exteriores. O diagrama de transição de estados para estas instruções está ilustrado na figura 6.28.

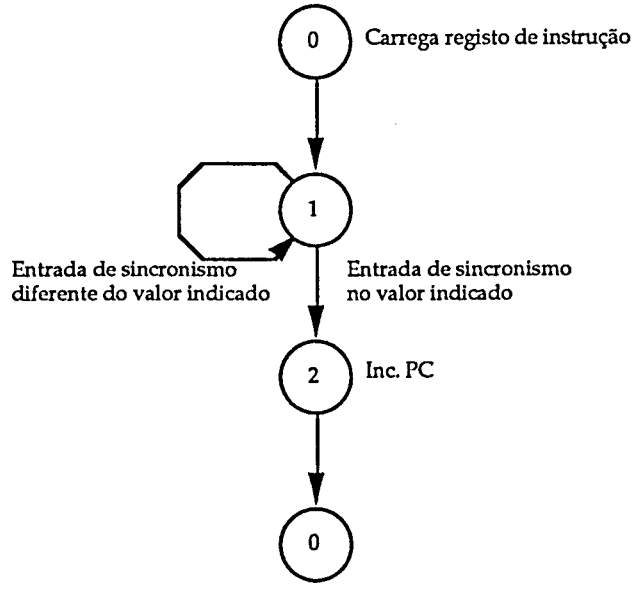


Fig.6.28: Diagrama de transição de estados para as instruções WS.

6.3.2.3. Instruções para o controlo de recursos internos, e do fluxo do programa

Apresenta-se nesta secção a caracterização individual das instruções destinadas ao controlo dos recursos internos do processador, e do fluxo do programa de teste, que foram descritas no quadro 6.4.

LD CNT, Conteúdo

Esta instrução transfere o conteúdo das segunda e terceira palavras de memória, após o código de instrução, para o contador interno. A segunda palavra deverá conter os 8 bits mais significativos do valor a carregar, e a terceira palavra os 8 bits menos significativos. O diagrama de transição de estados para a execução desta instrução está ilustrado na figura 6.29.

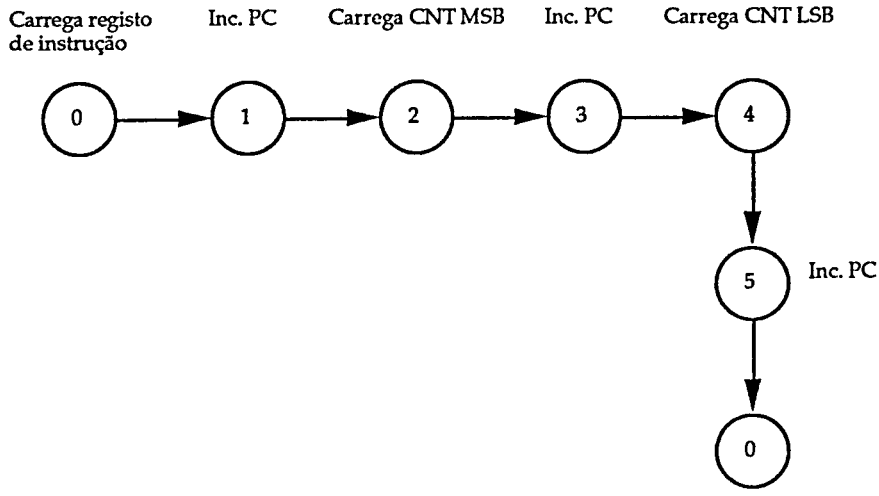


Fig.6.29: Diagrama de transição de estados para a instrução LD CNT, Conteúdo.

SERFLG0, ..., SERFLG7

Este conjunto de 8 instruções permite seleccionar uma de entre as 8 *flags* de erro disponíveis para cada TAP. A ocorrência de um erro, na posterior execução de uma instrução NSHFCP, actuará sobre a *flag* de erro seleccionada. A possibilidade de seleccionar diferentes *flags* de erro, a preceder instruções NSHFCP para diferentes tipos de vectores, permite discriminar entre diferentes tipos de faltas. Existem dois grupos de 8 *flags* de erro, um por cada TAP comandado pelo processador. A execução de uma instrução que selecciona a cadeia BST a controlar (o TAP a controlar) tem automaticamente por efeito que passa a estar seleccionado o respectivo grupo de *flags* de erro, embora se mantenha (dentro do novo grupo) a indicação da *flag* de erro seleccionada. A inicialização do processador tem por consequência que a *flag* de erro seleccionada passa a ser a 0 (equivale a SERFLG0). O diagrama de transição de estados para a instrução SERFLG está ilustrado na figura 6.30.

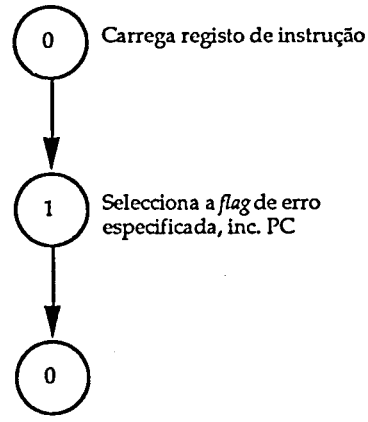


Fig.6.30: Diagrama de transição de estados para as instruções SERFLG.

JPE Endereço

Esta instrução provoca um salto para o endereço especificado, se a *flag* de erro seleccionada tiver sido actuada anteriormente.

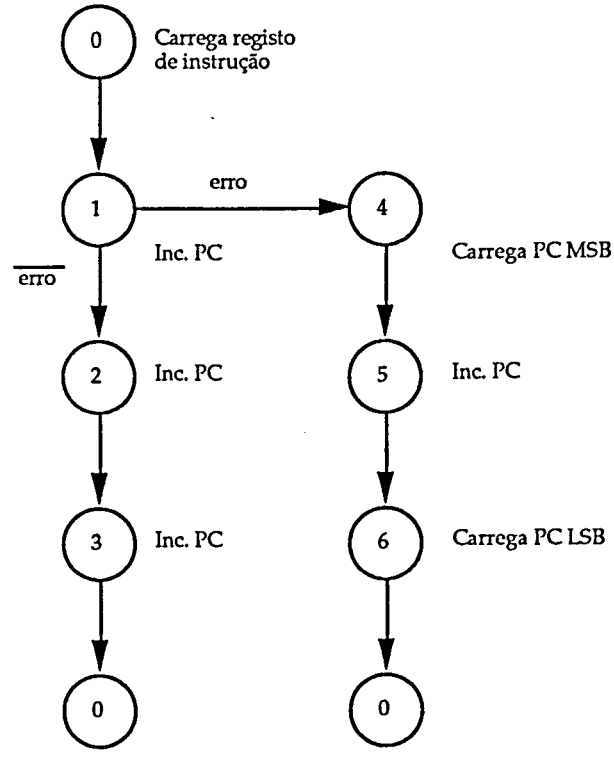


Fig.6.31: Diagrama de transição de estados para a instrução JPE Endereço.

Caso contrário, o próximo código de instrução será lido na primeira posição que se segue ao operando desta instrução. O correspondente diagrama de transição de estados está ilustrado na figura 6.31.

JPNE Endereço

Esta instrução provoca um salto para o endereço especificado, se a *flag* de erro seleccionada não tiver sido actuada anteriormente. Caso contrário, o próximo código de instrução será lido na primeira posição que se segue ao operando desta instrução. O correspondente diagrama de transição de estados está ilustrado na figura 6.32.

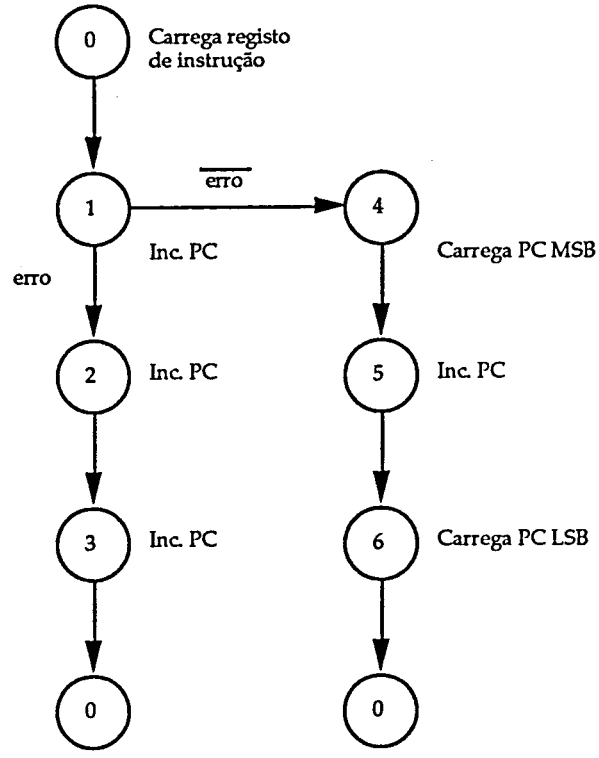


Fig.6.32: Diagrama de transição de estados para a instrução JPNE Endereço.

HALT

Esta instrução é usada para terminar a execução de um programa. O diagrama de transição de estados correspondente está ilustrado a figura 6.33.

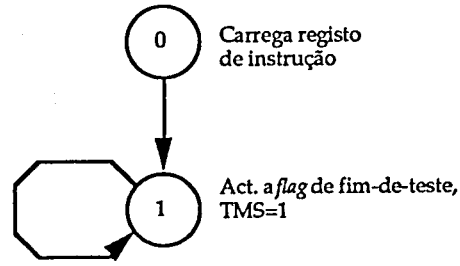


Fig.6.33: Diagrama de transição de estados para a instrução HALT.

6.3.2.4. Requisitos em memória, e tempo de execução

A caracterização das instruções anteriormente descritas, no que respeita ao número de palavras de memória, e ao número de ciclos de relógio, está apresentada no quadro 6.5.

Requisitos em memória, e tempo de execução, para cada instrução		
Instrução	Número de palavras	Número de ciclos de relógio
TRST	1	3
TMS0, TMS1	1	2
NSHF	$1 + \lceil (N/8) \rceil$	2, se N=0
		5, se N=1
		$3 + N + \lceil (N/8) \rceil$, se N>1
NSHFCP	$1 + 3 \cdot \lceil (N/8) \rceil$	2, se N=0
		9, se N=1
		$3 + N + 3 \cdot \lceil (N/8) \rceil$, se N>1
NTCK	1	2 + N
SELTAP0, SELTAP1	1	2
SSA0, SSA1, SSB0, SSB1	1	2
WSA0, WSA1, WSB0, WSB1	1	Indefinido
LD CNT, Conteúdo	3	6
SERFLG0,...,SERFLG7	1	2
JPE Endereço, JPNE Endereço	3	4, se o salto não ocorrer
		5, se o salto ocorrer
HALT	1	Indefinido

Quadro 6.5: Requisitos em memória, e tempo de execução, para cada instrução (N: valor do contador).

6.4. TESTABILIDADE HIERÁRQUICA: A INFRAESTRUTURA BST

O interesse de se incluir uma infraestrutura BST no próprio controlador residente foi já justificado no capítulo anterior, por permitir a extensão do BST como barramento de teste entre CCIs, e também pelo facto de permitir que o mesmo controlador seja empregue em diferentes níveis hierárquicos. Esta secção efectuará a apresentação desta infraestrutura, com relevo para os três registos que são específicos do controlador residente.

A infraestrutura BST do controlador está ilustrada na figura 6.34 (Siemens, 90b), e corresponde ao bloco assim designado, no diagrama global apresentado na figura 6.1.

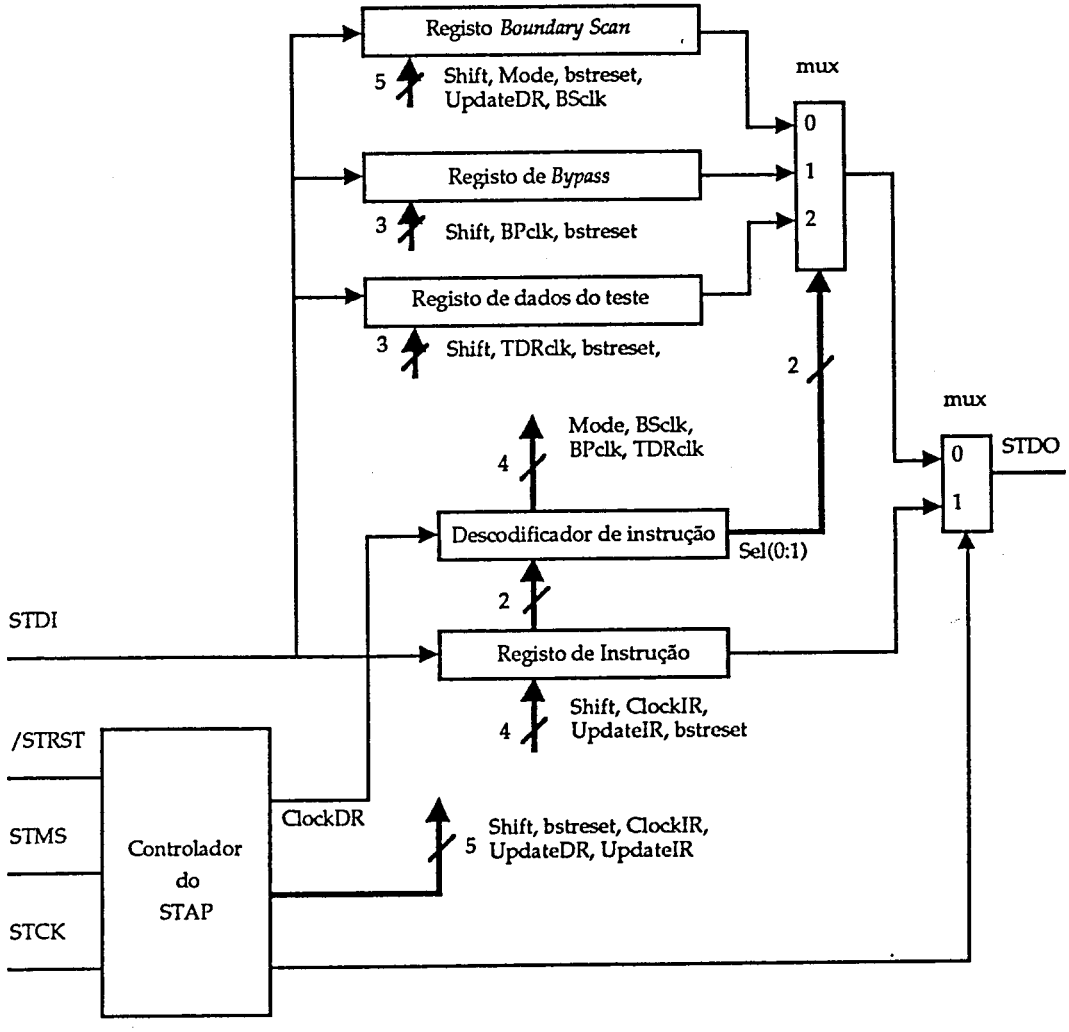


Fig.6.34: Infraestrutura BST do controlador residente.

A inicialização desta infraestrutura é feita automaticamente quando se alimenta o componente (*power-up reset*), sem que seja portanto necessária uma malha RC no respectivo

pino de inicialização (/STRST). Este pino está no entanto disponível, no sentido de proporcionar uma inicialização assíncrona da lógica BST, por ordem de um controlador residente a nível hierárquico superior (através da linha /TRST de um dos seus dois TAPs controlados). Os três registos específicos do controlador residente serão agora descritos em maior detalhe.

6.4.1. O registo de instrução

O registo de instrução é composto por dois bits, e suporta as quatro instruções descritas no quadro 6.6 (o bit mais à direita é o mais próximo de STDO).

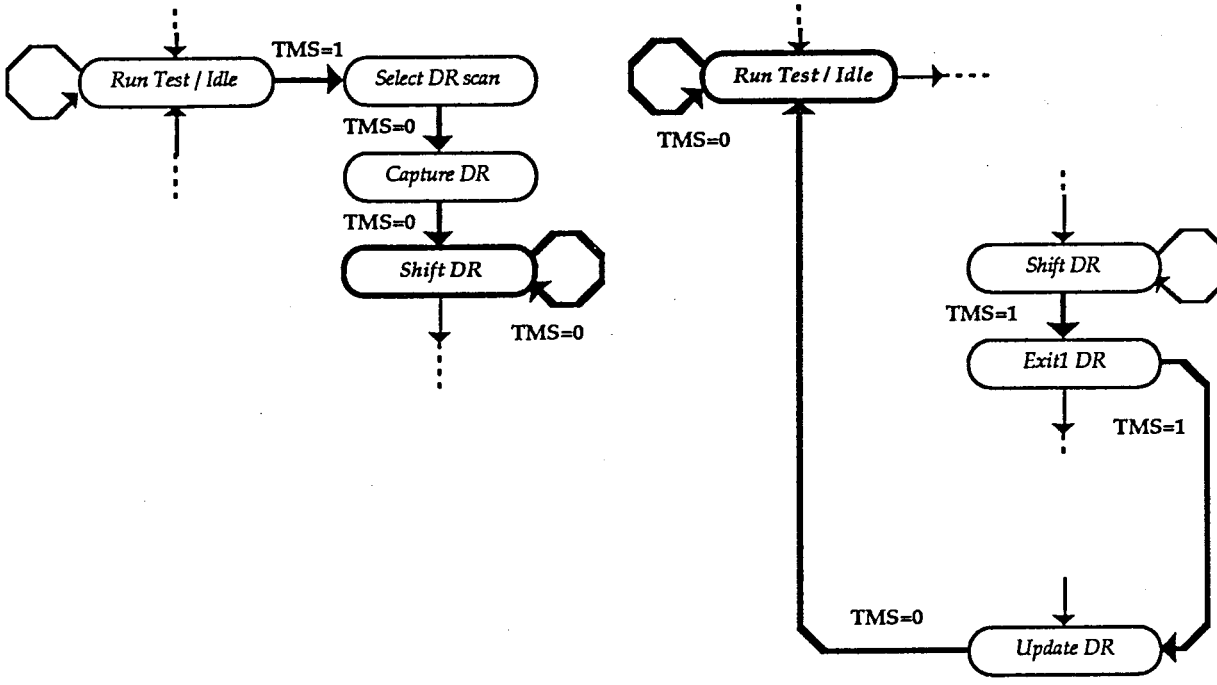
Código da instrução	Instrução
00	*Extest*
01	*Run Board Test*
10	*Sample / Preload*
11	*Bypass*

Quadro 6.6: Instruções suportadas pela infraestrutura BST do controlador residente.

Para além das três instruções obrigatórias (IEEE Std 1149.1, 90, p. 7-2), é também suportada a instrução *Run Board Test*, que permitirá a um controlador residente a um nível hierárquico superior o controlo sobre a execução do teste à CCI. Este teste terá início quando o controlador do STAP for colocado no estado *Run Test / Idle*, após o registo de instrução ter recebido o código correspondente à instrução *Run Board Test*. Nestas circunstâncias, o sinal de relógio que cadencia a execução do teste será o próprio STCK, fornecido pelo controlador a nível superior, que assim disporá de total controlo sobre a execução do teste à CCI.

A interrupção na execução do programa de teste pode ter lugar por interrupção dos impulsos de relógio aplicados em STCK, ou por colocação do controlador do STAP em qualquer outro estado (que não *Run Test / Idle*). A instrução *Run Board Test* selecciona o registo de dados do teste, no percurso dos registos de dados da infraestrutura BST do controlador. Uma vez que este registo captura o conteúdo do registo de estado do processador (ver figura 6.3), a leitura dos resultados parciais ou totais do teste pode em qualquer altura ser efectuada pelo controlador residente a nível hierárquico superior, através da sequência de transição de estados que se ilustra na figura 6.35.(a), permanecendo o controlador do STAP no estado *Shift-DR* até ter sido obtida a informação pretendida. A

continuação da execução do programa de teste poderá então ter lugar por nova transição para o estado *Run Test / Idle*, de acordo com o que se ilustra na figura 6.35.(b).



(a) Transição de estados para a leitura dos resultados (parciais ou totais).

(b) Transição de estados para retomar a execução do programa de teste.

Fig.6.35: Leitura dos resultados proporcionados pela execução da instrução *Run Board Test.*

Refira-se que a execução do teste terá também lugar independentemente de qualquer ordem recebida de um nível hierárquico superior, sempre que ocorrer um impulso em **0** na linha exterior de inicialização do processador (*/ExtReset*, na figura 6.1), sendo nestas circunstâncias a execução do programa de teste cadenciada pelo sinal de relógio do próprio processador (*ExtCLK*, na figura 6.1).

6.4.2. O registo de dados do teste

O registo de dados do teste permite capturar o conteúdo do registo de estado do processador (ver figura 6.3), e é seleccionado quando o registo de instrução receber o código da instrução *Run Board Test.* É constituído por 17 células, tal como se descreve no quadro 6.7.

Célula	Designação
0...7	*Flags* de erro, TAP 0
8...15	*Flags* de erro, TAP 1
16	*Flag* de fim-de-teste

Quadro 6.7: Constituição do registo de dados do teste.

6.4.3. O registo *boundary scan*

O registo *boundary scan* contem as células BST associadas ao conjunto de pinos funcionais do processador, tal como se ilustra na figura 6.1, e inclui também as células associadas ao conjunto de pinos que constituem o barramento de identificação (*ID(0:5)*, figura 6.2). Este registo é constituído por 50 células, tal como se descreve no quadro 6.8.

Célula	Pino	Comentários
0	ExtCLK	Saída interna do oscilador a cristal
1	ERROR	Saída com o resultado do teste
2	ExtReset	Pino de inicialização externa do processador
3...10	D(0:7)	Barramento de dados
11...26	A(0:15)	Barramento de endereços
27,28	SyncInpA, SyncOutA	Entrada e saída de sincronismo, grupo A
29,30	SyncInpB, SyncOutB	Entrada e saída de sincronismo, grupo B
31...35	TDI,TDO,TCK,TMS,/TRST	TAP 0 (controlado pelo processador)
36...40	TDI,TDO,TCK,TMS,/TRST	TAP 1 (controlado pelo processador)
41	CLK	Saída com o relógio interno do processador
42	DABC	Permite o acesso directo às cadeias BST da CCI
43	DeserEn	Para activar um deserializador externo
44...49	ID(0:5)	Barramento de identificação

Quadro 6.8: Constituição do registo *boundary scan*.

6.5. CONTROLO DO PROCESSADOR

Esta secção tem por objectivo caracterizar os blocos ClockMux e ResetGen (ver figura

6.1), destinados a suportar os dois modos de funcionamento já anteriormente referidos: sob controlo da infraestrutura BST, ou em modo autónomo.

O bloco ClockMux tem por objectivo permitir que o relógio do processador (CLK, na figura 6.1) seja proveniente do relógio exterior (ExtCLK), ou do relógio de teste aplicado à infraestrutura BST do controlador (STCK). A comutação entre estas duas fontes de relógio é controlada pelos dois sinais referidos no quadro 6.9.

Sinais de selecção		Proveniência do relógio do processador
RunBoardTest	*RunTestIdle*	
inactivo	inactivo	ExtCLK (relógio exterior)
inactivo	activo	ExtCLK (relógio exterior)
activo	inactivo	Nenhuma (não deve haver relógio)
activo	activo	STCK (relógio de teste)

Quadro 6.9: Comutação da proveniência do relógio do processador.

Os sinais *RunBoardTest* e *RunTestIdle*, quando activos, indicam respectivamente que a instrução *Run Board Test* está presente no registo de instrução, e que o controlador do STAP (na infraestrutura BST do controlador residente) se encontra no estado *Run Test / Idle*.

Tal como se descreve no quadro 6.9, a comutação na proveniência do relógio inicia-se com a recepção do código da instrução *Run Board Test*, passando o relógio interno (CLK) a ser proveniente do relógio de teste (STCK) quando o controlador do STAP se encontrar no estado *Run Test / Idle*. Repare-se que a transição de estados que permite a leitura dos resultados (parciais ou totais) do teste, ilustrada na figura 6.35, deve implicar a inibição do sinal de relógio do processador, de modo a que se retome a execução do programa de teste (figura 6.35.(b)) com o processador no mesmo estado interno. Por esta razão, não deverá haver relógio do processador quando o controlador do STAP deixar o estado *Run Test / Idle*, tal como se ilustra no quadro 6.9.

O bloco ClockMux tem uma constituição interna que está ilustrada na figura 6.36. Repare-se que no funcionamento em modo autónomo se verifica uma das duas primeiras condições descritas no quadro 6.9, pelo que o relógio do processador (CLK) será nestas circunstâncias proveniente do relógio exterior (ExtCLK).

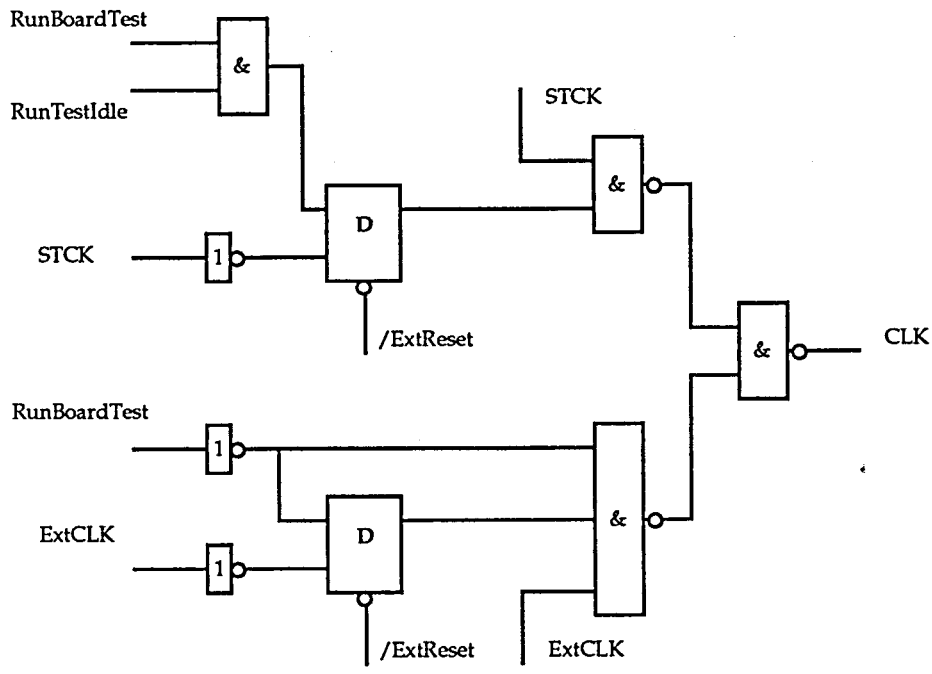


Fig.6.36: Constituição do bloco responsável pela selecção do relógio do processador.

Também a inicialização do processador, controlada pelo bloco ResetGen na figura 6.1, está relacionada com o modo de funcionamento seleccionado. Com efeito, a recepção do código da instrução *Run Board Test* deve dar origem à inicialização do processador, de modo a permitir a execução do programa de teste, sob comando do controlador residente a nível hierárquico superior. Esta inicialização deve ocorrer durante a permanência no estado *Update-IR* (que actualiza o conteúdo do registo de instrução), e deverá desaparecer à passagem para o estado *Run Test / Idle*, onde a execução do programa de teste terá lugar (ver figura 6.35.(a)).

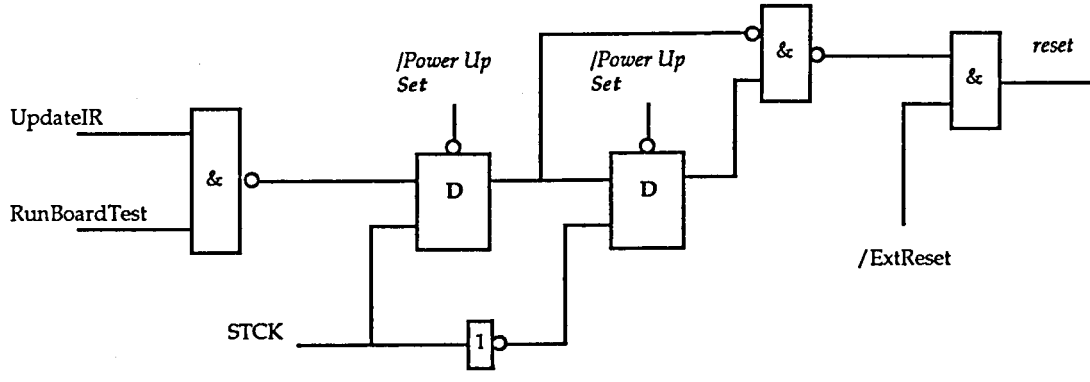
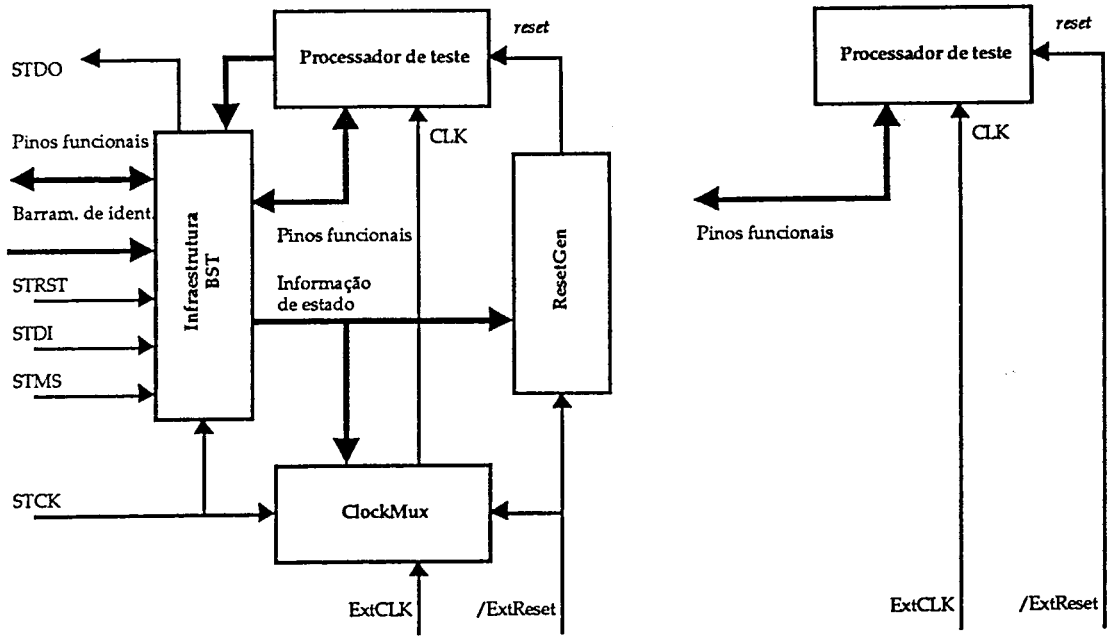


Fig.6.37: Constituição do bloco que controla a inicialização do processador.

A aplicação de um impulso a 0 no pino de inicialização exterior (*/ExtReset*, na figura 6.1) produzirá o mesmo efeito, conforme se verifica na figura 6.37, que ilustra a constituição deste bloco.

As considerações apresentadas nesta secção tornaram claro que a existência destes dois blocos é apenas devida à necessidade de suportar os dois modos de funcionamento possíveis, que consistem no controlo do funcionamento através da infraestrutura BST, ou no funcionamento em modo autónomo. Esta constatação vem realçar o interesse em se considerar o caso particular dos sistemas constituídos por uma única CCI, em que não existam requisitos em termos de testabilidade hierárquica, e onde a infraestrutura BST não seja de facto necessária. Nestas circunstâncias, e tal como se ilustra na figura 6.38, o controlador residente poder-se-ia reduzir ao processador de teste, por eliminação dos três blocos relacionados com a infraestrutura BST, com a consequente redução de complexidade. Repare-se que também o número de pinos seria reduzido (os 5 pinos do STAP deixariam de existir), o que poderia por sua vez representar uma vantagem adicional.



(a) Diagrama de blocos para o controlador residente, com infraestrutura BST.

(b) Diagrama de blocos para o controlador residente, sem infraestrutura BST.

Fig.6.38: O controlador residente, com e sem infraestrutura BST.

Esta alternativa tem ainda um interesse especial pela forma independente como foi apresentado o projecto dos quatro blocos apresentados na figura 6.1, e que permite

facilmente considerar as duas alternativas ilustradas na figura 6.38 (Ferreira et al., 92b).

Esta secção concluiu a apresentação da arquitectura de um controlador residente para CCIs com BST, tal como foi especificada no capítulo anterior.

Capítulo 7

A geração automática do programa de teste

Objectivos e limitações

Estratégia global

A informação necessária à GAPT

Procedimentos principais

Integração do conjunto de vectores gerados

7. A GERAÇÃO AUTOMÁTICA DO PROGRAMA DE TESTE

Este capítulo tem por objectivos principais identificar a informação necessária para cada etapa da geração automática do programa de teste (GAPT), especificar os procedimentos principais que a constituem, e descrever o modo de realizar a integração do conjunto de vectores gerados, de modo a produzir um programa de teste final.

Os objectivos estabelecidos para a GAPT são descritos em primeiro lugar. Segue-se a apresentação da estratégia global adoptada, que permite a identificação da informação necessária a cada etapa do protocolo de teste. A especificação dos procedimentos principais que permitem a geração de vectores em cada uma destas etapas é então efectuada. A conclusão deste capítulo tem lugar com a integração do conjunto de vectores gerados, que dá origem a um programa de teste final, especificado em termos do conjunto de instruções suportadas pelo controlador residente.

Este capítulo surge no seguimento da especificação de características apresentada no capítulo 5 para a GAPT, e concretiza as directrizes principais que foram então definidas.

7.1. OBJECTIVOS E LIMITAÇÕES

Apesar da aparente simplicidade que caracteriza a geração de um programa de teste para o caso ideal de uma CCI em que todos os componentes disponham de BST, e de auto-teste incorporado, esta tarefa pode complicar-se substancialmente pela presença de grupos de componentes não BST. A geração automática dos vectores de teste para estes grupos deverá ser realizada por ferramentas que existam já disponíveis para este fim, e os vectores gerados deverão ser integrados no conjunto total a aplicar através da infraestrutura de teste existente na CCI (Hansen, 90), (Robinson et al., 90).

A dificuldade na geração do programa de teste pode também ter origem na necessidade de se realizarem testes que se baseiem em modos particulares de funcionamento, acessíveis através da infraestrutura BST, mas que transcendam a funcionalidade mínima especificada na respectiva norma (IEEE Std 1149.1, 90). Refira-se como exemplo a possibilidade de se efectuar o teste de grupos de componentes não BST, do tipo combinacional, por aplicação de um conjunto de vectores pseudo-aleatórios, e compactação de respostas (análise de assinatura). Esta possibilidade pode em certos casos ser proporcionada por reconfiguração do modo de funcionamento das células BST, tal como sucede com a série de octais com BST fabricados pela Texas Instruments. Para este

exemplo concreto, existe um conjunto de instruções que reconfiguram as células BST de modo a permitirem a realização das seguintes estruturas (TI BCT8244, 90):

- Registos de deslocamento com realimentação linear (LFSR, *Linear Feedback Shift Register*), destinados a permitir a geração pseudo-aleatória de estímulos de teste.
- Registos de assinatura com entradas múltiplas (MISR, *Multiple Input Signature Register*), destinados a permitir a compactação de respostas por análise de assinatura.

Apesar da sequência de operações elementares que conduzem à reconfiguração das células BST, para os modos de funcionamento referidos, ser relativamente simples, a grande diversidade de situações que podem ter lugar, e até a própria identificação de cada grupo a ser testado por este processo, levantam dificuldades que complicam significativamente a tarefa de automatizar a geração do programa de teste para o controlador residente.

As considerações apresentadas justificam que, tal como foi já referido no capítulo 5, a GAPT assuma para a infraestrutura BST de cada componente uma funcionalidade que se restringe às três instruções obrigatórias (*Bypass, Sample/Preload, Extest*), e às quatro instruções opcionais (*Idcode, Usercode, Intest, Runbist*), descritas em (IEEE Std 1149.1, 90, cap. 7). Esta limitação tem por objectivo restringir a complexidade da GAPT, sem no entanto restringir a complexidade admissível para a CCI a testar. Para os casos em que existam grupos de componentes não BST, a GAPT suportará a realização do respectivo teste por recurso à funcionalidade referida, e de acordo com um conjunto de vectores especificados como parte da informação de entrada. A exploração de potencialidades adicionais, tais como a geração pseudo-aleatória de estímulos, ou a compactação de respostas, terá que ser efectuada por posterior edição do programa de teste gerado.

A resolução no diagnóstico constitui também um factor que condiciona os objectivos e limitações da GAPT, tendo a este propósito sido já anteriormente referido que a realização do teste deverá proporcionar para o exterior apenas uma saída do tipo sim/não, e para o controlador residente a nível hierárquico superior uma indicação do tipo de falta detectada, e a identificação da cadeia BST em que a respectiva detecção teve lugar. Neste sentido, a GAPT deverá manipular o registo de estado do processador (através das instruções SERFLG do controlador residente) de modo a permitir que seja memorizada a identificação do grupo de vectores responsável pela detecção de cada falta, sendo a identificação da cadeia BST garantida pelas instruções que seleccionam qual o TAP activo (SELTAP, que afectam directamente o registo de estado).

A GAPT assume ainda que a CCI a testar poderá dispor de uma ou duas cadeias BST, com linhas próprias de TMS, TCK, TDO, TDI e /TRST. Refira-se que a necessidade de

TAPs independentes para as duas cadeias BST não constitui propriamente uma limitação, uma vez que as restantes configurações possíveis teriam essencialmente interesse quando não existissem recursos de teste que permitissem alimentar de forma independente os dois TAPs.

O quadro 7.1 sintetiza os objectivos e limitações descritos para a GAPT.

Objectivos
• Deverão ser consideradas todas as etapas do teste de uma CCI com BST: o teste da infraestrutura BST, o teste de ligações (incluindo as ligações BST, e as que pertencem a grupos de componentes não BST), e o teste de componentes. • A execução do programa de teste deverá proporcionar para o exterior uma saída do tipo sim/não, e para um controlador a nível superior uma indicação do tipo de falta detectada, e a identificação da cadeia BST em que a respectiva detecção teve lugar. • Deverá ser suportada a existência de duas cadeias BST.
Limitações
• A existência de grupos de componentes não BST deverá ser limitada aos casos de reduzida dimensão e complexidade, e deverá estar disponível a informação que descreve estes grupos (incluindo os respectivos vectores de teste). • É apenas reconhecida a funcionalidade proporcionada pelas três instruções obrigatórias, e pelas quatro instruções opcionais, tal como são descritas em (IEEE Std 1149.1, 90, cap. 7). • Quando existirem duas cadeias BST, os respectivos TAPs deverão ser completamente independentes (não deverão partilhar quaisquer sinais).

Quadro 7.1: Objectivos e limitações da GAPT.

7.2. ESTRATÉGIA GLOBAL

As questões associadas à implementação de cada uma das etapas referidas para o protocolo de teste são analisadas nesta secção, sendo apresentados os problemas principais, e as opções efectuadas. Os problemas referentes à avaliação da cobertura de faltas (qualidade do teste), e à interacção com os recursos de teste exteriores, são também

abordados.

7.2.1. O teste da infraestrutura BST

O teste da infraestrutura BST será realizado considerando-se que esta etapa é independente das restantes. Nestas circunstâncias, e conforme foi já referido no capítulo 3, a existência de faltas entre ligações pertencentes a esta infraestrutura, e outras ligações, poderá não ser detectada nesta etapa. É no entanto seguro admitir que a sua detecção terá lugar numa das etapas seguintes, o que satisfaz os objectivos principais do teste sim/não, pretendido do controlador residente.

Uma alternativa a este processo consistiria em se começar por admitir a operacionalidade da infraestrutura BST, deslocando-se um primeiro vector que aplicasse o valor dominante a todas as ligações, e só então se procedendo ao teste da infraestrutura. Repare-se no entanto que esta alternativa tem o inconveniente de ser dependente da tecnologia usada (o valor dominante não é sempre o mesmo), o que tornaria esta solução particular para cada caso. Por outro lado, e se existirem grupos de componentes não BST, a implementação desta estratégia (aplicar a todos os restantes nós o valor dominante) poderia simplesmente não ser exequível. Finalmente, a preparação da infraestrutura BST para o deslocamento dos vectores pretendidos implica a realização de operações que constituem por si só a parte principal do teste à infraestrutura BST (deslocamento através dos registos de instrução), pelo que esta medida faria pouco sentido.

A realização do teste à infraestrutura BST terá deste modo lugar de acordo com a sequência de operações que foi descrita no protocolo de teste (ver a secção 3.3.1), e que consiste essencialmente em impor a comutação do nível lógico em todas as ligações TDO-TDI (efectuando um deslocamento através dos registos de instrução de cada componente), a que se seguirá a verificação do conteúdo dos registos de identificação existentes. Em CCIs que disponham de duas cadeias BST, este teste será realizado de forma independente, numa de cada vez.

A detecção de uma falta na infraestrutura BST (ou numa delas, quando existirem duas), deverá determinar a interrupção do teste, já que não existirão condições para a sua continuação.

7.2.2. O teste das ligações na CCI

O teste das ligações na CCI será realizado em duas fases, a primeira dedicada ao teste das ligações BST, e a segunda ao teste das ligações pertencentes a grupos de componentes não BST. Esta segunda fase terá lugar através da infraestrutura BST que rodeia cada grupo a testar (teste virtual), existindo acesso directo apenas aos pinos de E/S primária da CCI, tal como foi já descrito no capítulo 3 (ver a secção 3.3.2.2).

A aplicação de estímulos e captura de respostas terá em ambos os casos lugar de acordo com o protocolo descrito no capítulo 5 (ver figura 5.4), controlado pelos sinais SS (do controlador para o equipamento exterior) e WS (do equipamento exterior para o controlador), e que integra as etapas descritas no quadro 7.2. A transição entre estas etapas deverá ser precedida pelas transições indicadas para os sinais SS e WS (Driessen et al., 90, p. 12).

Etapa	Acção	SS	WS
i)	Aplicar os estímulos através da infraestrutura BST.	Sobe	
ii)	Aplicar os estímulos nos nós de entrada com acesso paralelo.		Sobe
iii)	Capturar as respostas através da infraestrutura BST.	Desce	
iv)	Capturar as respostas nos nós de saída com acesso paralelo.		Desce

Quadro 7.2: Teste das ligações: Protocolo de sincronismo.

A realização do teste será feita por ciclos sucessivos de aplicação de estímulos e captura de respostas, o que equivale a múltiplas passagens pelo percurso *Select-DR-scan/.../Update-DR*, no diagrama correspondente ao controlador de estados do TAP de cada componente (ver figura 2.7). Para o caso em que existam duas cadeias BST, o controlador residente deverá responsabilizar-se por sequenciar o respectivo comando, de modo a que se continue a observar o protocolo descrito no quadro 7.2. A aplicação de estímulos e captura de respostas será então implementada através de múltiplas execuções do ciclo de operações elementares descrito no quadro 7.3 (onde se assume que os controladores do TAP de todos os componentes BST se encontram já no estado *Shift-DR*).

Operação elementar	CCI com uma só cadeia BST	CCI com duas cadeias BST
Inserir o vector na cadeia BST 0	•	•
Prosseguir até *Select-DR-scan*, inclusivé	•	•
Seleccionar a cadeia BST 1		•
Inserir o vector na cadeia BST 1		•
Prosseguir até *Select-DR-scan*, inclusivé		•
Subir o sinal SS	•	•
Esperar pela subida do sinal WS	•	•
Prosseguir até *Shift-DR*, inclusivé	•	•
Seleccionar a cadeia BST 0		•
Prosseguir até *Shift-DR*, inclusivé		•
Descer o sinal SS	•	•
Esperar pela descida do sinal WS	•	•

Quadro 7.3: Teste de ligações: Sequência de operações elementares.

7.2.2.1. Ligações BST

O teste das ligações BST terá início pela aplicação dos vectores destinados à detecção de faltas de continuidade, a que se seguirá a aplicação dos vectores destinados à detecção de curto-circuitos.

Faltas de continuidade

Tal como foi já anteriormente referido, e de acordo com o modelo de faltas adoptado, o teste relativamente às faltas de continuidade será realizado pela aplicação de um 0, e de um 1, a partir de cada nó CL presente em cada ligação. A ordem pela qual estes nós serão seleccionados terá por objectivos permitir que este teste se realize em simultâneo para todas as ligações, e ao mesmo tempo garantir que não existirão ligações com mais do que um nó CL activo ao mesmo tempo.

Curto-circuitos

O algoritmo responsável pela geração de vectores para o teste de curto-circuitos entre ligações é semelhante ao algoritmo do auto-diagnóstico (Cheng et al., 90). No entanto, e atendendo a que não se assume a existência de um valor lógico dominante, é introduzida

uma modificação que tem por objectivo garantir que cada vector manterá sempre metade do número total de ligações a **0**, e a outra metade a **1**. Esta modificação consiste em aplicar às ligações com ordem ímpar um VTS (vector de teste sequencial) que corresponde ao complemento do VTS atribuído à ligação que a precede (Robinson et al., 81), sendo às ligações de ordem par atribuído um VTS que é gerado pelo algoritmo do auto-diagnóstico.

O número de vectores necessário para o teste de uma CCI com N ligações BST, de acordo com o algoritmo do auto-diagnóstico (apresentado na secção 4.4.1.6), é dado pelo mínimo valor de p que satisfizer a †

$$\binom{p}{p/2} \geq N \qquad (7.1)$$

Repare-se que a modificação referida, nos casos em que o número de vectores indicado pela equação 7.1 for ímpar, produz VTSs que não foram gerados pelo algoritmo do auto-diagnóstico. Este facto é consequência de o complemento de cada VTS apresentar um diferente número de 0s (e de 1s) relativamente ao VTS que lhe deu origem. Já nos casos em que o número de vectores indicado pela equação 7.1 for par, o conjunto gerado inclui cada VTS e o seu complemento, pelo que a relação entre o número de ligações BST, e o número de vectores produzidos pelo algoritmo adoptado, é a que se apresenta no quadro 7.4.

Número de ligações	6	20	70	252	924	3432
Número de vectores	4	6	8	10	12	14

Quadro 7.4: Nº de ligações x Nº de vectores, para a detecção de curto-circuitos.

A detecção de uma falta nas ligações BST poderia não implicar necessariamente a interrupção do teste. Pode referir-se a este propósito a detecção de faltas de continuidade, cuja existência não acarreta o perigo de danos adicionais. Deve no entanto atender-se à possibilidade de um curto-circuito que envolva ligações da infraestrutura BST ser detectado apenas quando forem aplicados os primeiros vectores para o teste de continuidade às ligações, situação em que deveria ter lugar a interrupção do teste. Opta-se deste modo por determinar que a detecção de uma falta deve interromper a realização do teste, independentemente do tipo de vector aplicado.

---

† p/2 representa o quociente da divisão inteira de p por 2.

7.2.2.2. Ligações pertencentes a grupos de componentes não BST

Tal como foi já anteriormente referido, o teste destas ligações será efectuado através da infraestrutura BST que rodeia cada grupo, existindo acesso directo apenas aos nós que pertencerem à E/S primária da CCI.

A realização do teste aos grupos de componentes não BST terá lugar de acordo com a especificação de conjuntos de vectores gerados exteriormente, e que serão importados pela GAPT já no formato correspondente à descrição do grupo a que se destinam. Isto significa que a especificação destes vectores não diz respeito estritamente à constituição do grupo a testar, mas deverá igualmente levar em conta o conjunto de nós do tipo CL, e OL, que o rodeiam. A título de exemplo, e para o caso que se ilustra na figura 7.1, a correspondente especificação de vectores deverá levar em conta que existem duas possibilidades para o acesso ao terminal de entrada representado.

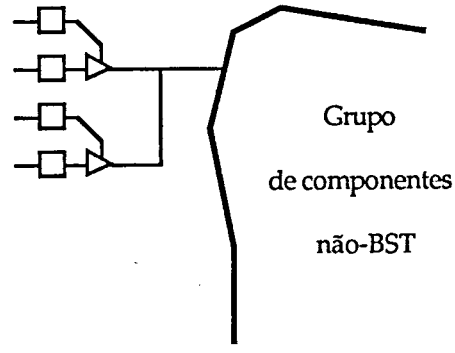


Fig.7.1: Entrada de um grupo de componentes não BST, com duplo acesso.

A geração dos vectores de teste para os grupos de componentes não BST deverá igualmente responsabilizar-se por verificar a inexistência de situações de conflito no acesso a barramentos pertencentes a estes grupos. Este problema foi já discutido no capítulo 4, mas então com respeito apenas às ligações BST. A consideração de circuitos que incluam grupos de componentes não BST requer que se volte a considerar este problema, mas agora na perspectiva de conflitos originados em barramentos pertencentes a estes grupos, tal como se ilustra na figura 7.2.

Os mesmos vectores que permitem a realização do teste aos grupos de componentes não BST deverão aplicar um valor lógico fixo às ligações BST, de modo a aumentar a probabilidade de detecção relativamente aos curto-circuitos que envolvam ligações de ambos os tipos. Nos casos em que exista um valor lógico dominante para as faltas deste tipo, o valor oposto a este deverá ser escolhido para aplicação às ligações BST. Nestas

circunstâncias, e assumindo que os vectores de teste gerados para os grupos de componentes não BST deverão ser capazes de forçar os dois níveis lógicos em todos os seus pinos funcionais, a detecção dos curto-circuitos referidos estaria sempre garantida (Robinson et al., 90, pp. 575 a 577).

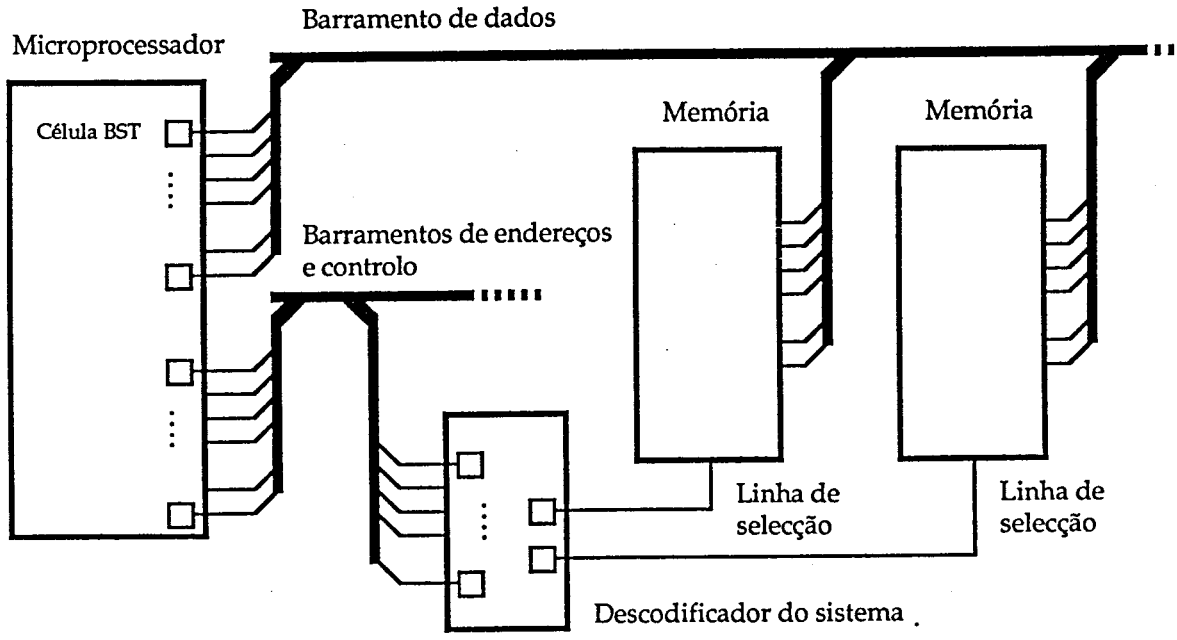


Fig.7.2: Possíveis conflitos no comando de barramentos, com componentes não BST.

Repare-se que se verifica uma dependência da estratégia descrita, relativamente à tecnologia usada (o valor lógico dominante não é sempre o mesmo). No entanto, e se esta dependência devesse ser evitada, poder-se-ia especificar apenas que o valor lógico a aplicar às ligações BST fosse sempre o mesmo. Nestas circunstâncias, quando este valor coincidisse com o valor lógico não dominante, a detecção da falta teria lugar nos nós OL das ligações BST envolvidas. Caso contrário, a respectiva detecção deveria ter lugar nos nós OL que capturam os resultados presentes nas saídas dos grupos de componentes não BST.

Repare-se que a detecção de uma falta, durante a realização do teste correspondente a esta etapa, tanto poderá corresponder a um componente não BST avariado, como a um curto-circuito entre ligações. Nestas circunstâncias, a detecção de uma falta deverá ter por consequência a interrupção do teste.

7.2.3. O teste de componentes BST

O teste dos componentes BST será igualmente realizado em duas fases, a primeira das quais diz respeito aos componentes BST com auto-teste incorporado.

A morosidade inerente ao deslocamento sucessivo de estímulos e respostas não permitirá normalmente a realização de um teste completo aos componentes BST que não dispuserem de auto-teste incorporado. No entanto, a aplicação de um número reduzido de vectores permitirá concluir se estes componentes respondem da forma esperada, pelo que esta possibilidade deverá ser suportada pela GAPT.

O teste de componentes BST constitui uma etapa que é efectivamente independente das restantes, e a sua posição nesta sequência (infraestrutura, ligações, componentes) tem essencialmente a ver com o tempo que a respectiva realização pode requerer (alguns segundos, por exemplo). Considera-se que um componente avariado não corresponde a uma falta que se possa propagar, pelo que a realização do teste não será interrompida nestas circunstâncias.

A realização desta etapa, em CCIs que disponham de duas cadeias BST, terá lugar de forma independente em cada cadeia.

7.3. A INFORMAÇÃO NECESSÁRIA À GAPT

Esta secção identifica a informação mínima que deverá estar disponível para permitir a GAPT. Tal como foi já referido no capítulo 5, esta informação diz essencialmente respeito à descrição da infraestrutura BST em cada componente, das ligações na CCI, e dos grupos de componentes não BST presentes. A geração dos segmentos de programa para cada etapa do protocolo de teste apresenta requisitos próprios neste domínio, que serão analisados nas secções seguintes.

7.3.1. Teste da infraestrutura BST da CCI

O teste da infraestrutura BST da CCI será realizado de acordo com a descrição que foi efectuada quando se apresentou o protocolo de teste para CCIs com BST (ver capítulo 3), e que consiste na seguinte sequência:

i) Inicialização da lógica BST em todos os componentes (*reset*).

ii) Deslocamento através dos registos de instrução.

iii) Leitura dos registos de identificação.

A concretização do passo i) não requer qualquer descrição acerca da infraestrutura BST em cada componente, uma vez que faz exclusivamente uso de recursos, e procedimentos, que estão completamente definidos em (IEEE Std 1149.1, 90), e relativamente aos quais não existem parâmetros que possam variar de implementação para implementação.

A verificação da operacionalidade da cadeia BST, por deslocamento através dos registos de instrução (passo ii), requer a identificação do tamanho (número de bits) do registo de instrução presente em cada componente, e a indicação do conteúdo deste registo, após uma operação de captura (*capture-IR*). Com efeito, e de acordo com a descrição apresentada em (IEEE Std 1149.1, 90, pp. 6-1 e 6-2), só os dois bits menos significativos (mais próximos de TDO) são forçados a capturar a sequência 01 (o bit mais à direita é o menos significativo). Os restantes bits poderão capturar valores lógicos fixos, ou relacionados com informação específica de cada componente, não sendo obrigatória a descrição dos valores capturados. Isto significa que a informação requerida para a realização deste passo consiste no tamanho do registo de instrução em cada componente, e ainda no seu conteúdo após uma operação de captura, para os casos em que tal for conhecido.

A leitura dos códigos de identificação (passo iii) requer por sua vez que se indique se a existência do respectivo registo, que é opcional, tem efectivamente lugar. Relativamente aos componentes que dispuserem desta possibilidade, será ainda necessário conhecer qual o código de instrução correspondente (*Idcode*), e qual a identificação associada ao componente. Esta identificação deriva do código JEDEC atribuído a cada fabricante (JEDEC, 86), e tem um tamanho fixo de 32 bits. É também possível a existência de um código de identificação adicional (para certos componentes programáveis), acessível através do mesmo registo de identificação (IEEE Std 1149.1, 90, p. 7-24), o que obriga à necessidade de se indicar se esta possibilidade tem lugar, para cada componente. Nos casos em que tal suceder, deverá também indicar-se qual o código de instrução correspondente (*Usercode*), e o código de identificação adicional associado ao componente.

Deve ainda referir-se que a realização dos dois últimos passos (ii e iii) obriga a conhecer a ordem dos componentes na cadeia BST da CCI, que terá que ser extraída do ficheiro que contem a descrição das ligações (*netlist*).

O teste à infraestrutura BST termina com a sua preparação para a realização da etapa seguinte (teste de ligações), em que é seleccionado o funcionamento em modo de teste externo. Deve no entanto referir-se que a instrução correspondente (*Extest*) força nos pinos de saída o conteúdo das células BST, cujo valor poderá provocar conflitos no acesso a barramentos (não foram ainda deslocados quaisquer valores para o seu interior). Este

problema poderá ser ultrapassado por recurso à instrução *Sample/Preload* (como se verá na secção 7.4 — Procedimentos principais), que selecciona igualmente o registo BST em cada componente, mas aplicando aos pinos os valores presentes nas saídas da lógica interna (IEEE Std 1149.1, 90, p. 7-9). Isto significa que o código desta instrução deve também ser conhecido, pelo que a informação necessária à geração do segmento do programa para o teste da infraestrutura BST da CCI é o que se apresenta no quadro 7.5.

Qual a identificação desta cadeia BST? Qual a ordem deste componente, na cadeia BST a que pertence?
Qual é o número de bits do registo de instrução? Qual é o código da instrução *Sample/Preload*? Qual é o conteúdo do registo de instrução, após uma operação de captura (**X**, **0**, ou **1**, por cada bit)?

Existe registo de identificação? Em caso afirmativo:
• Qual o correspondente código de instrução (*Idcode*)?
• Qual é a identificação do componente?
• Existe código de identificação suplementar? Em caso afirmativo:
  • Qual o correspondente código de instrução (*Usercode*)?
  • Qual é a identificação suplementar do componente?

Quadro 7.5: Informação necessária à GAPT para a infraestrutura BST.

7.3.2. Teste das ligações

Tal como foi já anteriormente referido, o teste das ligações terá lugar em duas fases, primeiro relativamente às ligações BST, e depois às ligações pertencentes a grupos de componentes não BST. Os requisitos de informação para cada uma destas fases serão agora analisados.

7.3.2.1. Ligações BST

A informação necessária à GAPT para as ligações BST consiste principalmente na descrição dos nós CL e OL presentes em cada ligação, associados quer a pinos BST, quer às E/S primárias da CCI. Esta descrição deverá incluir a identificação dos nós de controlo de estado, e dos valores que permitam forçar o estado de alta impedância, relativamente aos pinos que dispuserem desta possibilidade.

A necessidade de se evitarem conflitos no comando de barramentos merece também um comentário. Apesar de a apresentação que foi efectuada no capítulo 4 (ver secção 4.2) ter já permitido concluir que este problema pode ser evitado por uma escolha cuidadosa dos nós CL a activar, a análise das implicações devidas aos tipos possíveis de andares de saída deve também aqui ser considerada. Do ponto de vista funcional, as alternativas existentes neste domínio são essencialmente independentes da tecnologia presente (TTL, CMOS, ECL, BiCMOS, etc.), e podem ser descritas de forma sucinta como se apresenta a seguir:

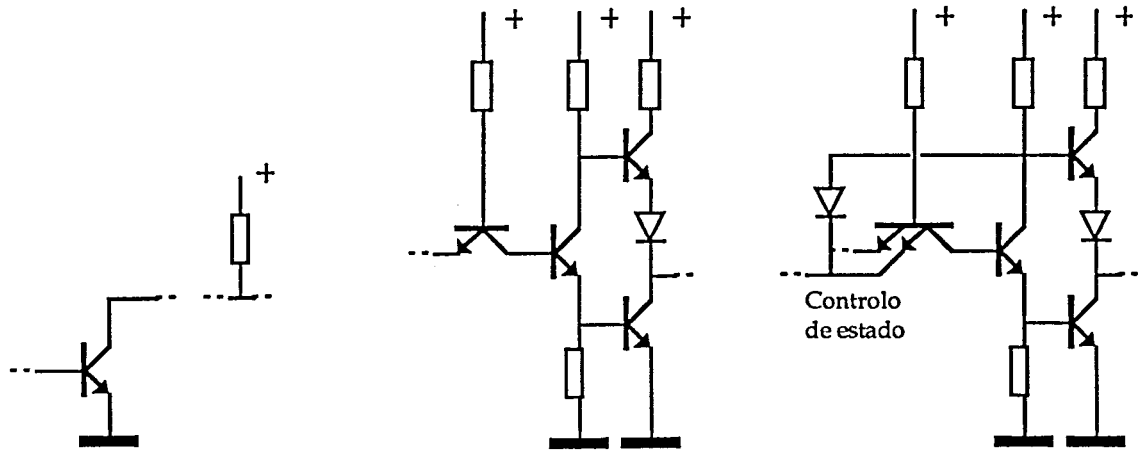
- Andar de saída do tipo I. A circulação de corrente é unidireccional, e requer quase sempre a existência de uma resistência exterior. É o caso dos andares de saída do tipo colector aberto (TTL) e dreno aberto (CMOS), capazes apenas de absorver corrente (*sink*), e ainda o tipo emissor aberto (ECL), capaz apenas de fornecer corrente (*source*). Este tipo de andar não é exclusivo de componentes SSI/MSI, tendo também por vezes lugar em saídas de periféricos de microprocessadores, com o objectivo de permitir pedidos de interrupção através de configurações em *wired-and*.
- Andar de saída do tipo II. A circulação de corrente é bidireccional (*source/sink*), e não é possível a passagem para o estado de alta impedância. É o caso frequente em saídas TTL (*totem-pole*) ou CMOS (quer nos andares de saída simétricos — série B —, quer nos restantes).
- Andar de saída do tipo III. A circulação de corrente é bidireccional, e é possível a passagem para o estado de alta impedância. Trata-se de um andar de saída frequentemente idêntico ao anterior, mas em que ambos os percursos (*source/sink*) podem ser inibidos em simultâneo.

Estes três tipos de andares estão ilustrados na figura 7.3, onde por simplicidade se optou por representar a respectiva configuração para o caso TTL.

Uma mesma ligação pode integrar vários andares de saída do mesmo tipo, mas não é de esperar que integre andares de tipos diferentes, tal como se descreve no quadro 7.6. Repare-se que as quatro combinações em que coexistem diferentes tipos de andares são todas susceptíveis de permitir a ocorrência de correntes de circulação entre saídas em valores lógicos complementares, pelo que não se considera a existência de ligações nestas circunstâncias.

Os elementos apresentados no quadro 7.6 permitem ainda concluir que a ocorrência de conflitos no comando de barramentos só será de considerar relativamente às ligações que incluírem andares de saída do tipo III. A descrição das ligações BST deverá deste modo

indicar se numa dada ligação existem ou não andares de saída deste tipo (com possibilidade de passar para o estado de alta impedância), pelo que esta informação deverá igualmente estar presente.



(a) Andar do tipo I (capaz apenas de absorver corrente).

(b) Andar do tipo II (absorve e fornece corrente, mas sem terceiro estado).

(c) Andar do tipo III (absorve e fornece corrente, e tem terceiro estado).

Fig.7.3: Os três tipos de andares de saída considerados, para o caso TTL.

Tipo de andar			Comentários à funcionalidade da ligação
I	II	III	
		•	Comum em barramentos
	•		Frequente em ligações simples, com um único nó do tipo CL
	•	•	Não (correntes de circulação entre andares de saída)
•			Ligações *wired-and* ou *wired-or*
•		•	Não (correntes de circulação entre andares de saída)
•	•		Não (correntes de circulação entre andares de saída)
•	•	•	Não (correntes de circulação entre andares de saída)

Quadro 7.6: Coexistência de andares de saída numa mesma ligação.

A descrição das ligações BST deve ainda levar em conta a possibilidade de uma ligação se encontrar permanentemente a um valor lógico fixo (V_{CC}, ou massa). As ligações nestas circunstâncias disporão apenas de nós do tipo OL, e a confirmação de que a ligação se encontra ao nível lógico pretendido requer que a indicação deste nível faça parte da

respectiva descrição.

Finalmente, e atendendo a que uma mesma ligação pode incluir nós cujas células BST pertençam a diferentes cadeias, a identificação da cadeia a que pertence cada célula deverá igualmente estar presente.

O quadro 7.7 sintetiza a informação mínima que deverá estar presente relativamente às ligações BST existentes na CCI, de modo a permitir a geração automática do respectivo programa de teste.

Existem apenas andares de saída com terceiro estado?

Qual o número de cadeias BST em que existem células pertencentes a esta ligação? Por cada uma:
• Qual a identificação desta cadeia BST?
• Quantos pinos de saída existem? Por cada um:
  • Qual a identificação da célula BST de saída que lhe está associada?
  • Qual a identificação da célula BST de controlo que lhe está associada?
  • Qual o valor, em cada célula BST de controlo, que força o estado de alta impedância?
• Quantos pinos de entrada existem? Por cada um:
  • Qual a identificação da célula BST de entrada que lhe está associada?
• Quantos pinos bidireccionais existem? Por cada um:
  • Qual a identificação da célula BST de saída que lhe está associada?
  • Qual a identificação da célula BST de controlo que lhe está associada?
  • Qual o valor, em cada célula BST de controlo, que força o estado de alta impedância?
  • Qual a identificação da célula BST de entrada que lhe está associada?

Qual o número de saídas primárias da CCI que pertencem a esta ligação? Por cada uma: • Qual a identificação desta saída primária?
Qual o número de entradas primárias da CCI que pertencem a esta ligação? Por cada uma: • Qual a identificação desta entrada primária? • Qual o valor que força o estado de alta impedância no respectivo canal de teste?
Esta ligação está permanentemente a um valor lógico fixo? Em caso afirmativo, qual?

Quadro 7.7: Informação necessária à GAPT para as ligações BST.

7.3.2.2. Ligações pertencentes a grupos de componentes não BST

O teste de grupos de componentes não BST será realizado através da infraestrutura BST que rodeia cada grupo, e dos pinos de E/S primária que lhes estejam ligados.

Conforme foi já referido, e de acordo com o fluxo de dados ilustrado na figura 5.3, admite-se que os conjuntos de vectores de teste empregues para este fim são gerados exteriormente.

A informação mínima para permitir a GAPT relativamente a estes grupos deverá deste modo incluir a descrição do conjunto de nós CL e OL que os rodeiam, bem como a especificação dos correspondentes vectores de teste. Repare-se que esta especificação deverá indicar os valores a atribuir a cada nó CL (e aos respectivos sinais de controlo de estado, quando existentes), e os valores esperados (e respectivas máscaras) em cada nó OL, pelo que deverá normalmente ter lugar um pré-processamento do conjunto de vectores gerados exteriormente.

Qual o número de vectores especificados para o teste deste grupo?

Qual o número de cadeias BST em que existem células que rodeiam este grupo? Por cada uma:
• Qual a identificação desta cadeia BST?
• Quantas células pertencentes a esta cadeia estão presentes? Por cada uma:
  • Qual é a identificação desta célula?
  • Por cada vector:
  • Qual é o valor a deslocar para esta posição?
  • Qual é o valor esperado nesta posição?
  • Qual é o valor da correspondente máscara de comparação?
  • Se for requerida a aplicação de valores lógicos fixos, durante o teste às ligações BST:
  • Qual é o valor a deslocar para esta posição?

Quantas entradas primárias estão presentes? Por cada uma:
• Qual é a identificação desta entrada primária?
• Por cada vector:
  • Qual é o valor a aplicar a esta entrada primária?
  • Qual é o valor a aplicar ao seu sinal de controlo de estado?
• Se for requerida a aplicação de valores lógicos fixos, durante o teste às ligações BST:
  • Qual é o valor a aplicar nesta entrada primária?
  • Qual é o valor a aplicar ao seu sinal de controlo de estado?

Quantas saídas primárias estão presentes? Por cada uma:
• Qual é a identificação desta saída primária? Por cada vector:
  • Qual é o valor esperado nesta saída primária?
  • Qual é o valor da correspondente máscara de comparação?

Quadro 7.8: Informação necessária à GAPT para as ligações em grupos de componentes não BST.

Dependendo da funcionalidade de cada grupo de componentes não BST, poderá

também tornar-se necessário especificar um conjunto de valores que deverão estar presentes nas células BST que rodeiam o grupo, e nas entradas primárias a ele ligadas, durante a aplicação dos vectores destinados ao teste das ligações BST.

O quadro 7.8 sintetiza a informação mínima que deverá estar presente para permitir a GAPT relativamente às ligações pertencentes a grupos de componentes não BST.

7.3.3. Teste dos componentes BST

Esta etapa permitirá a execução das funções de auto-teste disponíveis em componentes BST, e ainda a aplicação de vectores de teste especificados para os componentes BST que não dispuserem daquela possibilidade, mas relativamente aos quais se pretenda a realização de um teste interno.

Nos casos em que a existência de auto-teste tiver lugar, deverá ser indicado o código da instrução correspondente (*Runbist*), e qual o número de impulsos no relógio de teste (TCK) que são necessários para a sua realização. Também o tamanho do registo que conterá o resultado do auto-teste (e que é seleccionado pela instrução *Runbist*), e o seu conteúdo quando a respectiva execução tiver terminado com sucesso, deverão ser indicados.

Apesar de não ser necessário o prévio deslocamento para o interior dos componentes de quaisquer valores de inicialização (IEEE Std 1149.1, 90, p. 7-20), poderá pretender-se que a realização do auto-teste decorra com valores pré-especificados presentes nos pinos de saída. Nestas circunstâncias, a instrução *Sample/Preload* poderá ser usada para deslocar estes valores para o interior dos registos *boundary scan*, e a necessidade desta operação deverá ser explicitamente indicada. Neste caso, deverão ainda ser conhecidos o código da instrução correspondente, o tamanho do registo *boundary scan*, e os valores a deslocar para o seu interior.

Os componentes que não dispuserem de auto-teste incorporado poderão ser testados pela aplicação de um conjunto de vectores especificados para este fim, sendo para o efeito as células BST colocadas em modo de teste interno (instrução *Intest*). A informação necessária para permitir a GAPT deverá neste caso incluir a identificação das células BST que rodeiam cada componente, bem como a especificação dos valores a deslocar, valores esperados, e máscaras de comparação.

Finalmente, e de acordo com o que foi referido a este propósito no capítulo 5 (secção 5.3.3), a ordem dos componentes em cada cadeia BST deve também ser conhecida.

O quadro 7.9 sintetiza a informação mínima necessária para permitir a GAPT relativamente aos componentes BST.

Qual a cadeia BST a que pertence o componente? Qual a ordem deste componente, na cadeia BST a que pertence?
Qual é o comprimento do registo *boundary scan*?

Existe auto-teste acessível através da infraestrutura BST? Em caso afirmativo:
• Qual o correspondente código de instrução (*Runbist*)?
• Qual o número de impulsos necessários no relógio de teste (TCK)?
• Qual o tamanho do registo que conterá o resultado?
• Qual o resultado, quando o auto-teste termina com sucesso?
• É requerido o prévio deslocamento de valores para os pinos de saída? Em caso afirmativo:
  • Qual o código da instrução *Sample/Preload*?
  • Por cada célula BST:
  • Qual o valor a deslocar para esta posição?

Deve ter lugar um teste interno através da infraestrutura BST? Em caso afirmativo:
• Qual é o número de vectores de teste especificados para o efeito?
• Qual é o código da instrução que selecciona o modo de teste interno (*Intest*)?
• Para cada célula:
  • Para cada vector:
  • Qual é o valor a deslocar para esta posição?
  • Qual é o valor esperado nesta posição?
  • Qual é o valor da correspondente máscara de comparação?

Quadro 7.9: Informação necessária à GAPT para os componentes BST.

7.3.4. Especificação dos vectores de teste

A especificação final dos vectores que permitem realizar o teste da CCI constitui uma das saídas da GAPT, e integra os valores a aplicar através da infraestrutura BST, e os que dizem respeito aos nós de E/S primária.

A informação necessária para especificar os vectores a aplicar através da infraestrutura BST deve identificar a cadeia em questão, e incluir por cada bit desta cadeia a indicação do novo valor a deslocar, do resultado esperado para o valor capturado, e ainda a especificação de uma máscara que permita identificar quando é que a comparação é relevante. A necessidade de uma máscara decorre de ser possível não se conhecer o valor capturado em parte das células BST, já que a funcionalidade da lógica interna de vários componentes pode não ser conhecida. Por outro lado, mesmo nos casos em que esta

funcionalidade seja conhecida, a determinação dos valores capturados nas suas saídas pode não ter sido efectuada, já que isso exigiria a disponibilidade de modelos adequados, e a realização de sessões de simulação.

A especificação dos vectores correspondentes aos nós de E/S primária integra a informação relativa aos estímulos a aplicar nos nós de entrada, e as respostas esperadas nos nós de saída. A informação relativa aos nós de entrada deve incluir a especificação do valor a aplicar através dos correspondentes canais de teste, e ainda a indicação do valor a aplicar ao respectivo controlo de estado (activo, ou em alta impedância). No que respeita aos nós de saída, deverá estar presente a descrição do respectivo valor esperado, acompanhado por uma máscara que identifica quais as saídas cuja comparação é relevante, para cada vector.

A informação necessária para permitir a especificação dos vectores a aplicar à CCI está sintetizada no quadro 7.10.

Por cada cadeia BST da CCI a testar:
• Qual a identificação desta cadeia BST?
• Por cada célula BST existente nesta cadeia:
  • Qual o novo valor a deslocar para o seu interior?
  • Qual o resultado esperado, após uma operação de captura?
  • Qual o valor da respectiva máscara de comparação (sim/não)?

Por cada entrada primária: • Qual o valor do estímulo a aplicar através do respectivo canal de teste? • Qual o valor a aplicar ao respectivo sinal de controlo de estado?
Por cada saída primária: • Qual o valor esperado no correspondente canal de teste? • Qual o valor da respectiva máscara de comparação (sim/não)?

Quadro 7.10: Informação necessária à especificação dos vectores a aplicar à CCI a testar.

7.4. PROCEDIMENTOS PRINCIPAIS

Esta secção apresentará uma especificação em alto nível dos procedimentos principais envolvidos na GAPT, de acordo com a estratégia global anteriormente definida, e com os requisitos de informação que foram apresentados na secção anterior. A descrição de procedimentos que aqui será efectuada pretende caracterizar a sequência de acções que conduza à especificação dos vectores de teste para cada etapa do protocolo. A geração do

programa de teste a executar pelo controlador residente, que permitirá a aplicação dos vectores gerados, será descrita na secção 7.5 (Integração do conjunto de vectores gerados).

A especificação de procedimentos será efectuada através de uma pseudo-linguagem de alto nível, que fará uso de um conjunto de primitivas com significado equivalente ao das primitivas correspondentes na linguagem C, de acordo com a correspondência descrita no quadro 7.11.

Primitiva	Equivalente ANSI C
para	*for*
se / senão	*if / else*
enquanto	*while*
continuar	*continue*
interromper	*break*
e	& &
/* ... */	/* ... */ (comentário)

Quadro 7.11: Correspondência entre as primitivas usadas, e primitivas ANSI C.

No que respeita aos vectores destinados ao teste de ligações, BST ou não (a aplicar através dos registos *boundary scan*, em modo de teste externo), parte-se de um pressuposto comum, que consiste em assumir uma pré-inicialização que produz um vector-base com as seguintes características:

- Todas as células BST de saída contêm o valor lógico 1.
- Todas as células BST de controlo contêm o valor lógico que força no nó CL correspondente o estado de alta impedância.
- Todas as células BST de entrada contêm o valor lógico 1.
- Os valores esperados para cada célula BST (independentemente do seu tipo) contêm o mesmo valor que lhe foi atribuído pela operação de pré-inicialização.
- Todas as máscaras que definem se a comparação é ou não relevante estão inicializadas com o valor lógico que inibe a comparação.
- Todos os canais de teste associados às entradas primárias existentes contêm o valor lógico 1.
- Todos os sinais de controlo correspondentes aos canais de teste existentes contêm o valor lógico que força o estado de alta impedância.
- Todos os valores esperados correspondentes às saídas primárias existentes estão

inicializados com o valor lógico **1**.

- Todas as máscaras que definem se a comparação é ou não relevante, relativamente às saídas primárias existentes, estão inicializadas com o valor lógico que inibe a comparação.

Adicionalmente, e quando existirem grupos de componentes não BST que requeiram a aplicação de valores lógicos especificados na respectiva descrição, durante a realização do teste às ligações BST, deverão ainda ter lugar os passos seguintes:

- Todas as células que rodeiam grupos de componentes não BST devem ser reinicializadas com o valor requerido por cada grupo, quando esta especificação tiver lugar.
- Todas as entradas primárias (canais de teste) ligadas aos grupos de componentes não BST devem ser reinicializadas com o valor requerido por cada grupo, quando esta especificação tiver lugar.
- Todos os sinais de controlo correspondentes às entradas primárias que rodeiam grupos de componentes não BST devem ser reinicializados com o valor requerido por cada grupo, quando esta especificação tiver lugar.

O vector-base produzido pela inicialização descrita permite que a geração dos vectores para o teste de ligações tenha lugar apenas por modificação dos valores lógicos já atribuídos, e garante que os valores que não forem modificados não originarão quaisquer situações de conflito. A geração deste vector-base pode ser realizada pelo procedimento descrito a seguir:

```
/*  Procedimento responsável pela geração de um vector-base, que aplicaria às ligações da CCI um
    conjunto de valores "seguros". Este vector-base é posteriormente usado para efectuar a geração
    dos vectores destinados ao teste de ligações.                                          */

para (cada cadeia BST)
   para (cada célula BST)  {
      se ((esta célula for de saída) ou (de entrada))
         atribuir-lhe o valor lógico 1;
      senão
         /*   trata-se de uma célula de controlo                                          */
         atribuir-lhe o valor lógico que desactiva o(s) correspondente(s) nós CL;
      atribuir ao valor esperado o mesmo valor atribuído à célula BST correspondente;
```

```
        atribuir à máscara respectiva o valor que inibe a comparação;
    }
para (cada entrada primária da CCI)  {
    atribuir ao respectivo canal de teste o valor lógico 1;
    atribuir ao respectivo sinal de controlo o valor que força o estado de alta impedância;
}
para (cada saída primária da CCI)  {
    atribuir ao respectivo valor esperado o valor lógico 1;
    atribuir à máscara respectiva o valor que inibe a comparação;
}
para (cada grupo de componentes não BST)
    se (existir uma especificação dos valores requeridos nas entradas, durante o teste às ligações BST) {
        para (cada cadeia BST que rodeia este grupo)
            para (cada célula BST que rodeia este grupo)
                atribuir-lhe o valor requerido;
        para (cada entrada primária ligada a este grupo)  {
            atribuir ao respectivo canal de teste o valor requerido;
            atribuir ao respectivo sinal de controlo de estado o valor requerido;
        }
        /*      os valores requeridos são especificados na descrição dos grupos de componentes não BST
                                                                                                    */
    }
```

Proc. 7.1: Procedimento que gera o vector-base para o teste de ligações.

7.4.1. Teste da infraestrutura BST

O teste da infraestrutura BST dará lugar à geração de três vectores, que provocarão dois deslocamentos através dos registos de instrução, e um deslocamento através dos registos de identificação, ou dos registos de *BYPASS* (quando aquele não existir). Adicionalmente, deverá ainda ser gerado um quarto vector, que promoverá um deslocamento através dos registos de identificação (com a instrução *Usercode*), quando existirem componentes que disponham de um código de identificação adicional.

O primeiro deslocamento através dos registos de instrução envia para cada um o código da instrução *Idcode* (*Bypass*), de modo a permitir a leitura da identificação dos

componentes que dispuserem deste registo opcional, ao mesmo tempo que força sucessivas comutações do nível lógico nas ligações TDO-TDI, através do conteúdo capturado nos dois bits menos significativos do registo de instrução (01). Tal se justificou quando se discutiu o protocolo de teste para CCIs com BST (capítulo 3), uma sequência adicional será acrescentada a este vector (a ser inserida em primeiro lugar na cadeia BST), com o objectivo aumentar a capacidade de detecção de faltas que lhe está associada. A especificação do procedimento que gera este vector pode ser efectuada como se apresenta a seguir:

/* Procedimento responsável pela geração do vector que envia para cada registo de instrução o código que selecciona o registo de identificação (ou *BYPASS*, quando aquele não existir). Este vector inclui uma sequência adicional destinada a aumentar a capacidade de detecção de faltas na infraestrutura BST. */

para (para componente)
  para (cada bit do respectivo registo de instrução) {
    se (existir registo de identificação)
      o valor a deslocar é o que corresponde à instrução *Idcode*;
    senão
      o valor a deslocar é o que corresponde à instrução *Bypass*;
    se (está especificado o valor a capturar)
      o valor esperado é definido de acordo com o especificado;
    senão
      o valor esperado é (arbitrariamente) posto a 1;
    se (está especificado o valor a capturar)
      o valor da máscara é de modo a permitir a comparação;
    senão
      o valor da máscara é de modo a inibir a comparação;
  }
acrescentar no início† a sequência 01;
acrescentar no início uma sequência de 0s de comprimento igual ao maior registo de instrução presente;

Proc. 7.2: Geração do vector para o teste do percurso série, e envio dos códigos de *Idcode* ou *Bypass*.

---

† Define-se o início como a primeira parte a ser deslocada para o interior da cadeia BST.

A inserção na cadeia BST do vector gerado por este procedimento deverá ser seguida pela inserção de um novo vector, com o objectivo de forçar o deslocamento através dos registos de dados seleccionados (os registos de identificação, ou de *BYPASS*). Os novos valores que este vector deslocará para o interior da cadeia BST são irrelevantes, uma vez que a passagem pelo estado *Update-DR* não provoca efeito sobre o conteúdo dos registos seleccionados (IEEE Std 1149.1, 90, caps. 9 e 11). A especificação do procedimento que gera este vector pode ser apresentada da forma que se segue:

```
/*   Procedimento responsável pela geração do vector que permite a leitura dos registos de
     identificação.                                                                    */

para (cada componente)
   se (existir código de identificação)
      para (cada bit do registo de identificação)  {
         /*    valor a deslocar para o interior da cadeia BST                          */
         o valor a deslocar é (arbitrariamente) posto a 1;
         /*    valor esperado para o bit a deslocar para o exterior                    */
         o valor esperado é definido de acordo com o especificado na descrição do componente;
         /*    máscara                                                                 */
         o valor da máscara é de modo a permitir a comparação;
      }
   senão  {
      /*    valor a deslocar                                                          */
      o valor a deslocar é (arbitrariamente) posto a 1;
      /*    valor esperado (o registo de BYPASS captura sempre um 0)                  */
      o valor esperado é 0;
      /*    máscara                                                                   */
      o valor da máscara é de modo a permitir a comparação;
   }
```

Proc. 7.3: Leitura dos registos de identificação.

Se a CCI contiver componentes que disponham de um código de identificação adicional (IEEE Std 1149.1, 90, p. 11-1 e 11-2), deverão ser gerados mais dois vectores, de modo a permitir que se efectue a respectiva leitura. A identificação adicional será carregada

no registo de identificação, devendo para este efeito a instrução *Usercode* ser enviada para os componentes que dispuserem desta possibilidade. A especificação do procedimento responsável pela geração do vector que envia os correspondentes códigos de instrução pode ser efectuada como se apresenta a seguir:

```
/*  Procedimento responsável pela geração do vector que envia para cada registo de instrução o
    código que selecciona o registo de identificação (para leitura da identificação adicional dos
    componentes), ou de BYPASS (caso não exista código de identificação adicional).        */

para (cada componente)
   para (cada bit do registo de instrução)
      /*   valor a deslocar para o interior da cadeia BST                                   */
      se (existir código de identificação adicional)
         o valor a deslocar é o que corresponde à instrução Usercode;
      senão
         o valor a deslocar é o que corresponde à instrução Bypass;
```

Proc. 7.4: Geração do vector que envia os códigos de *Usercode* ou *Bypass*.

Repare-se que este procedimento não gera os valores esperados para cada bit, nem as máscaras correspondentes, já que esta informação só tem relevância para o teste à operacionalidade do percurso série de dados (TDO-TDI), na infraestrutura BST da CCI. Uma vez que este teste já terá sido realizado anteriormente, o envio de novos códigos de instrução para os componentes BST não requer a identificação de valores esperados, nem das respectivas máscaras (este deslocamento será efectuado pela instrução NSHF, cujo argumento requer apenas um terço da memória correspondente ao da instrução NSHFCP).

O procedimento que gera o vector para a leitura dos registos de identificação pode agora ser reutilizado (procedimento 7.3), mas tendo em conta que se pretende efectuar a leitura dos códigos de identificação adicional.

A transição para a etapa correspondente ao teste de ligações terá lugar com o envio da instrução *Extest*, que fará aparecer nos pinos de saída o conteúdo das respectivas células BST. No entanto, e tal como foi já referido na secção 7.3.1, torna-se necessária uma operação de pré-inicialização, de modo a garantir que os valores lógicos aplicados às ligações não provocarão conflitos. Uma forma de ultrapassar esta dificuldade consiste em

enviar para cada componente o código da instrução *Sample/Preload*, a que se deverá seguir a sequência de transição de estados *Select-DR-scan* / *Capture-DR* / *Exit1-DR* / *Update-DR*, no diagrama de estados dos controladores do TAP (sem envolver qualquer deslocamento através dos registos *boundary scan*). Esta sequência transfere para o interior das células BST os valores presentes nas saídas da lógica interna dos componentes, pelo que se assume que os conflitos referidos não ocorrerão. A instrução *Extest* poderá então ser enviada, o que preparará a infraestrutura BST para a aplicação dos vectores correspondentes ao teste de ligações. Quer a geração do vector correspondente ao envio da instrução *Sample/Preload*, quer a do correspondente ao envio da instrução *Extest*, serão realizadas de acordo com um procedimento genérico para a geração de vectores referentes ao envio de instruções, e que se pode apresentar da seguinte forma:

```
/*   Procedimento genérico, responsável pela geração de um vector que envia para cada registo de
     instrução o código da instrução pretendida (todos os componentes recebem o mesmo código de
     instrução).                                                                              */

para (cada componente)
   para (cada bit do registo de instrução)
      /*   valor a deslocar para o interior da cadeia BST                                    */
      o valor a deslocar é o que corresponde à instrução que se pretende enviar;
```

Proc. 7.5: Geração do vector que envia o código da instrução pretendida.

Repare-se que também este procedimento não gera os valores esperados para cada bit, nem as máscaras correspondentes, pelos mesmos motivos já referidos antes.

Refira-se ainda que a possível existência de duas cadeias BST não tem implicações sobre os procedimentos apresentados, que serão empregues de forma independente para gerar os vectores necessários ao teste de cada uma.

7.4.2. Teste das ligações

A especificação dos procedimentos principais relativos ao teste de ligações será feita em quatro etapas, correspondentes à geração dos vectores de teste para a detecção de faltas de continuidade, à selecção do conjunto de nós do tipo CL para permitirem a aplicação dos

vectores destinados à detecção de curto-circuitos, à geração dos vectores para este tipo de faltas, e à geração dos vectores destinados ao teste de ligações pertencentes a grupos de componentes não BST.

7.4.2.1. Geração de vectores para a detecção de faltas de continuidade em ligações BST

A geração de vectores para a detecção de faltas de continuidade é condicionada pela necessidade de se testarem todos os percursos entre cada nó do tipo CL, e o conjunto de nós do tipo OL, existentes em cada ligação. A realização deste teste pode decorrer em simultâneo em todas as ligações, tal como foi já anteriormente referido quando se discutiu o teste de ligações na CCI (capítulo 4), desde que não ocorram conflitos no acesso a barramentos.

Repare-se que a situação de existirem dois nós do tipo CL activos numa mesma ligação é de evitar (mesmo que apliquem o mesmo valor lógico), uma vez que a geração de vectores é feita precisamente com o objectivo de se testarem todos os percursos com início num nó do tipo CL, um de cada vez. O procedimento especificado nesta secção gera de uma vez só um conjunto de vectores que aplicam um 0 a partir de cada nó CL presente em cada ligação, e pode ser especificado da seguinte forma:

```
/*  Procedimento responsável pela geração de vectores destinados à detecção de faltas de
    continuidade nas ligações BST. Cada nó CL é activado em sequência, para permitir a aplicação
    de um 0 à respectiva ligação. Todas as ligações são testadas em paralelo.          */

começar com o primeiro vector;
enquanto (existirem percursos para serem testados) {
   para (cada ligação) {
      se (esta ligação está a um valor lógico fixo)
         continuar para a ligação seguinte;
      se (já existir um nó do tipo CL activo) {
         se (o percurso correspondente ainda não foi considerado) {
            atribuir a este nó o valor lógico 0;
            atribuir ao valor esperado nos correspondentes nós OL o valor lógico 0;
            modificar a máscara dos correspondentes nós OL para permitir a comparação;
```

```
            retirar o percurso correspondente da lista de percursos por considerar;
        }
        continuar para a ligação seguinte;
    }
    para (cada nó do tipo CL)
        se (o percurso correspondente ainda não foi testado) {
            activar este nó CL;
            se (existir um conflito com outras ligações, por activar este nó CL) {
                desactivar este nó CL;
                continuar para o próximo nó CL;
            }
            senão {
                retirar o percurso correspondente da lista de percursos por testar;
                atribuir a este nó CL o valor lógico 0;
                atribuir ao valor esperado nos correspondentes nós OL o valor lógico 0;
                modificar a máscara dos correspondentes nós OL para permitir a comparação;
            }
        }
    }
    prosseguir para o próximo vector de teste;
}
```

Proc. 7.6: Geração de vectores para a detecção de faltas de continuidade em ligações BST.

Tal como foi discutido no capítulo 4 (ver a secção 4.3), a detecção de faltas de continuidade requer a aplicação de um **0**, e de um **1**, a partir de cada nó CL existente em todas as ligações. Deste modo, o procedimento descrito nesta secção deverá ser repetido, mas agora de modo a atribuir a cada nó CL o valor lógico **1**, em vez de **0**.

O procedimento descrito deverá considerar que os nós do tipo CL presentes numa ligação podem estar associados a qualquer dos tipos de andares de saída representados na figura 7.3, pelo que expressões como *nó do tipo CL activo* deverão ser encaradas de acordo com o correspondente andar de saída. Estes pormenores de implementação foram no entanto evitados na especificação, com o objectivo de facilitar a sua leitura.

Repare-se que a possível existência de duas cadeias BST não implica modificações no procedimento descrito, já que a geração de vectores tem lugar por enumeração de todos os nós CL, para cada ligação. As ligações que incluírem nós CL pertencentes a mais do que

uma cadeia BST contribuirão para a geração de vectores em ambas as cadeias, enquanto as restantes contribuirão apenas para a geração de vectores na cadeia BST a que pertencem.

7.4.2.2. Selecção dos nós do tipo CL para a detecção de curto-circuitos entre ligações BST

O procedimento descrito nesta secção tem por objectivo seleccionar o conjunto de nós CL que serão usados para aplicar os vectores destinados à detecção de curto-circuitos entre ligações BST. De acordo com a discussão apresentada a este respeito no capítulo 4 (secção 4.2), este conjunto de nós deve permitir que todas as ligações sejam colocadas sob teste simultaneamente, e ao mesmo tempo garantir a ausência de conflitos no comando de barramentos.

O procedimento apresentado efectua uma pesquisa em profundidade sobre o espaço de combinações entre nós CL. A detecção de ligações com mais do que um nó CL activo dá origem a um retrocesso (*backtracking*) na pesquisa, pelo qual se desfaz a opção que deu origem a esta situação, e se inicia a procura de uma alternativa. Quando se esgotar o espaço de pesquisa sem que tenha sido encontrada uma solução satisfatória, o procedimento descrito aceita a existência de ligações em que exista mais do que um nó CL activo, devendo nestas circunstâncias garantir-se que os valores aplicados nestes nós são sempre os mesmos. Este procedimento pode especificar-se como se apresenta a seguir:

```
/*  Procedimento que selecciona os nós CL a activar em cada ligação BST, para permitir a
    aplicação dos vectores destinados à detecção de curto-circuitos.                    */

começar com a primeira ligação;
enquanto (todas as ligações não tiverem sido consideradas)  {
   se ((esta ligação está a um valor lógico fixo) ou (os andares de saída dos pinos BST forem todos do
        tipo I) ou (houver entradas primárias))  {
      se (está a recuar)
         recuar para a ligação anterior;
      senão
         prosseguir para a ligação seguinte;
      continuar;
   }
```

```
se((está a recuar) e (esta ligação ainda não é a pretendida))  {
    recuar para a ligação anterior;
    continuar;
}
se (esta ligação ainda não tiver um nó CL seleccionado)  {
    seleccionar o primeiro nó CL (de um pino de saída, ou bidireccional);
    atribuir à célula BST de controlo correspondente o valor que activa este nó CL;
    assinalar esta ligação como tendo já um nó CL seleccionado;
}
se (não ocorrerem conflitos com outras ligações, por se activar este nó CL)
    prosseguir para a ligação seguinte;
senão  {
    se (for possível escolher outro nó CL para activar, nesta ligação)  {
        atribuir à célula de controlo correspondente ao nó CL seleccionado, o valor que o desactiva;
        seleccionar o próximo nó CL (de um pino de saída, ou bidireccional);
        atribuir à célula de controlo correspondente o valor que activa este nó CL;
    }
    senão  {
        se ((esta for a primeira ligação) ou (a ligação em que se detectou o conflito for posterior))  {
            retornar à ligação que deu origem ao conflito;
            atribuir à célula de controlo inicialmente considerada o valor que activa o seu nó CL;
            assinalar esta situação com um aviso;
            prosseguir para a ligação seguinte;
        }
        senão  {
            atribuir à célula de controlo correspondente ao nó CL seleccionado o valor que o
                desactiva;
            assinalar que esta ligação não tem ainda um nó CL seleccionado;
            recuar para a ligação em que se detectou o conflito;
        }
    }
}
}
```

Proc. 7.7: Selecção dos nós CL a activar em cada ligação BST, para o teste de curto-circuitos.

7.4.2.3. Geração de vectores para a detecção de curto-circuitos entre ligações BST

A geração de vectores para a detecção de curto-circuitos recorre à selecção dos nós do tipo CL efectuada pelo procedimento anteriormente descrito, e tem por objectivo permitir a aplicação a cada ligação do respectivo VTS, gerado pelo procedimento que implementa o algoritmo para o teste de curto-circuitos entre ligações BST. Este procedimento será apresentado em primeiro lugar, e será seguido pela apresentação do procedimento correspondente à determinação das respostas esperadas, e respectivas máscaras, para cada vector. Os procedimentos que permitem a geração dos vectores a inserir nas cadeias BST, a partir do conjunto de VTSs gerados, serão então apresentados.

A geração dos vectores destinados à detecção de curto-circuitos entre ligações BST é efectuada de acordo com um algoritmo que constitui uma variante do algoritmo do auto-diagnóstico, já referido anteriormente neste capítulo (ver a secção 7.2.2.1). O procedimento correspondente pode ser especificado como se apresenta a seguir:

```
/*  Procedimento que gera N VTSs com p bits cada um, para o teste de curto-circuitos entre ligações
    BST. Estes VTSs são gerados por uma variante do algoritmo do auto-diagnóstico, e garantem
    que cada vector mantem metade das ligações em 0, e a outra metade em 1.                    */

determinar o número de vectores a gerar;
/*  o número de vectores é determinado de acordo com os valores apresentados no quadro 7.4     */
começar com a primeira ligação;
enquanto (esta ligação estiver a um valor lógico fixo)  {
   atribuir a todos os bits do seu VTS este mesmo valor lógico;
   prosseguir para a ligação seguinte;
}
inicializar o VTS da primeira ligação (primeira metade a 0, segunda metade a 1);
atribuir a uma variável ordem (par / ímpar) o valor par;
para (cada uma das restantes ligações)  {
   se (esta ligação estiver a um valor lógico fixo)  {
      atribuir a todos os bits do seu VTS este mesmo valor lógico;
      continuar para a ligação seguinte;
   }
   complementar a variável ordem;
```

```
se (a variável ordem é ímpar) {
    atribuir a esta ligação o complemento do VTS atribuído à ligação anterior;
    continuar para a ligação seguinte;
}
senão {
    atribuir a esta ligação o mesmo VTS atribuído à última ligação que a precedeu, com ordem par;
    para (cada bit neste VTS, prosseguindo da esquerda para a direita)
        se (este bit é 0) {
            se (este bit não é o bit mais à direita do VTS) {
                atribuir a este bit o valor 1;
                atribuir ao primeiro bit à sua direita o valor 0;
                interromper a consideração de mais bits;
            }
            senão {
                atribuir a uma variável K o número de bits consecutivos à direita, todos a 0;
                enquanto ((não encontrar o primeiro 0) e (não atingir a posição mais à esquerda))
                    prosseguir para a esquerda, a partir dos bits consecutivos à direita em 0;
                se ((atingiu a posição mais à esquerda) e (este bit não é 0)) {
                    atribuir, a partir deste bit inclusivé, e para a direita, um 0 a (k+1) bits;
                    atribuir um 1 a todos os restantes bits;
                    interromper a consideração de mais bits;
                }
                senão {
                    atribuir a este bit o valor 1;
                    atribuir, a partir deste bit inclusivé, e para a direita, um 0 a (k+1) bits;
                    atribuir a todos os restantes bits à direita (se os houver) o valor 1;
                    interromper a consideração de mais bits;
                }
            }
        }
}
```

Proc. 7.8: Geração dos VTSs a atribuir às ligações BST para a detecção de curto-circuitos.

A deslocação de cada novo vector para o interior da cadeia BST é acompanhada pela

deslocação para o exterior das respostas capturadas *para o vector anterior*. Este facto deverá ser levado em consideração para a determinação do argumento da instrução que se responsabilizará pelas operações de deslocamento e comparação para cada vector (NSHFCP, no controlador residente).

A determinação das respostas esperadas, e das respectivas máscaras, correspondentes à aplicação de um determinado vector, pode ser efectuada pelo procedimento descrito a seguir:

```
/*  Procedimento para a determinação das respostas esperadas, e das respectivas máscaras, para
    cada vector. Este procedimento determina as modificações que terão lugar nos valores lógicos
    presentes nos nós OL de cada ligação, assumindo que um vector indicado foi aplicado à CCI a
    testar. As máscaras de comparação são também actualizadas.                              */

para (cada ligação)  {
   se (esta ligação estiver a um valor lógico fixo)
      o valor aplicado é este mesmo valor lógico;
   senão
      para (cada nó CL)
         se (este nó CL está activo)  {
            o valor aplicado é o deste nó CL;
            interromper a consideração de mais nós CL;
         }
   atribuir aos valores esperados nos nós OL o valor aplicado;
   modificar as máscaras correspondentes aos nós OL, para permitir a comparação;
}
```

Proc. 7.9: Determinação dos valores esperados, e máscaras, para cada vector.

A comparação das respostas esperadas para o último vector terá que ser efectuada durante a operação de deslocamento de um (p+1)º vector, mas relativamente a este não será necessário concluir o ciclo de aplicação de estímulos e captura de respostas.

Uma vez gerado o conjunto de VTSs destinados ao teste de curto-circuitos entre ligações BST, e determinados os valores esperados e as respectivas máscaras de comparação, torna-se necessária a geração do conjunto de vectores a deslocar para o interior

da(s) cadeia(s) BST, e que permitam a aplicação deste conjunto de VTSs.

A geração do primeiro vector não necessita de levar em conta as respostas correspondentes à aplicação de um vector anterior, pelo que o procedimento correspondente pode ser especificado da forma que se segue:

```
/*  Procedimento responsável pela geração do primeiro vector a ser inserido nas cadeias BST da
    CCI. Este vector destina-se à detecção de curto-circuitos entre as ligações BST, de acordo com
    um conjunto de VTSs previamente especificados.                                          */

para (cada ligação)  {
   activar o(s) nó(s) CL correspondente(s);
   atribuir ao(s) nó(s) CL activo(s) o valor indicado pelo primeiro bit, no respectivo VTS;
}
```

Proc. 7.10: Geração do primeiro vector para a detecção de curto-circuitos entre ligações BST.

A geração de cada um dos (p-1) vectores seguintes deverá já levar em conta os valores esperados, e as respectivas máscaras, referentes ao vector anterior, pelo que o procedimento correspondente será agora especificado da seguinte maneira:

```
/*  Procedimento responsável pela geração de (p-1) vectores a serem inseridos nas cadeias BST da
    CCI. Estes vectores destinam-se à detecção de curto-circuitos entre as ligações BST, de acordo
    com um conjunto de VTSs previamente especificados.                                      */

para (cada um dos p-1 vectores a gerar)
   para (cada ligação)  {
      activar o(s) nó(s) CL correspondentes;
      atribuir ao(s) nó(s) CL activo(s) o valor indicado pelo bit correspondente, no respectivo VTS;
      atribuir aos valores esperados nos nós OL as respostas esperadas para o vector anterior;
      modificar a máscara correspondente aos nós OL para permitir a comparação;
   }
```

Proc. 7.11: Geração de vectores para a detecção de curto-circuitos entre ligações BST.

Finalmente, o último vector gerado tem por objectivo permitir o deslocamento para o exterior das respostas correspondentes ao pº vector destinado à detecção de curto-circuitos entre as ligações. Atendendo à estratégia definida para o teste de grupos de componentes não BST (que será realizado na etapa seguinte), este último vector a ser inserido na cadeia BST deverá responsabilizar-se por aplicar o valor lógico não dominante a todas as ligações BST, pelo que o respectivo procedimento poderá ser especificado da maneira que se segue:

```
/*  Procedimento responsável pela geração do (p+1)º vector a ser inserido nas cadeias BST da CCI.
    Este vector tem por objectivo forçar o deslocamento para o exterior das respostas ao pº vector, e
    aplica a todas as ligações BST o valor lógico não dominante.                              */

para (cada ligação)  {
   activar o(s) nó(s) CL correspondente(s);
   atribuir ao(s) nó(s) activo(s) o valor lógico não dominante;
   atribuir aos valores esperados nos nós OL as respostas esperadas para a aplicação do pº vector;
   modificar as máscaras correspondentes aos nós OL para permitir a comparação;
}
```

Proc. 7.12: Geração do último vector para a detecção de curto-circuitos entre ligações BST.

Também para este caso a possível existência de duas cadeias BST não implica modificações nos procedimentos apresentados, já que a geração dos vectores tem uma vez mais lugar por enumeração de todas as ligações existentes na CCI, tendo a tarefa de seleccionar o conjunto de nós do tipo CL activos (numa ou noutra cadeia BST) sido já realizada.

7.4.2.4. Geração de vectores para o teste de ligações pertencentes a grupos de componentes não BST

A geração dos vectores destinados ao teste de ligações pertencentes a grupos de componentes não BST será efectuada em duas etapas. A primeira destas etapas consiste numa pré-geração do conjunto de vectores necessários, em que se recorre à selecção de nós do tipo CL já efectuada anteriormente (secção 7.4.2.2), de modo a que a todas as ligações seja aplicado o valor lógico não dominante. A segunda etapa tem lugar sobre este conjunto,

e realiza a integração dos vectores de teste especificados para cada grupo de componentes não BST presente.

A aplicação do valor lógico não dominante ao conjunto de ligações BST, requerida para a pré-geração dos vectores, foi já referida no fim da secção anterior. Os procedimentos correspondentes à segunda etapa, que procede à integração dos vectores de teste especificados para cada grupo, consideram explicitamente cada cadeia BST, de acordo com a informação necessária a esta etapa da GAPT, descrita no quadro 7.8. Tal como sucedeu já anteriormente em relação ao teste das ligações BST, também o teste das ligações pertencentes a grupos de componentes não BST será realizado por deslocamentos sucessivos de estímulos e respostas. Isto significa que a geração de vectores deverá uma vez mais levar em conta que o deslocamento de um vector para o interior da cadeia BST ocorre em simultâneo com o deslocamento para o exterior das respostas ao vector anterior. Nestas circunstâncias, a especificação de M vectores dará origem ao deslocamento de (M+1) vectores, em que o (M+1)º vector deslocado tem por objectivo permitir a leitura das respostas ao último vector especificado.

A geração do primeiro vector é feita sem se compararem os valores que serão deslocados para o exterior da cadeia BST, pelo que o respectivo procedimento pode ser especificado da forma que se segue:

```
/*  Procedimento responsável pela geração do primeiro vector a ser inserido nas cadeias BST da
    CCI. Este vector destina-se ao teste das ligações pertencentes a grupos de componentes não BST,
    de acordo com um conjunto de vectores especificados para este fim.                    */

para (cada grupo de componentes não BST)
   para (cada cadeia BST)  {
      atribuir às células de saída os valores especificados no primeiro vector;
      atribuir às células de controlo os valores especificados no primeiro vector;
      atribuir às entradas primárias os valores especificados no primeiro vector;
      atribuir aos respectivos sinais de controlo de estado os valores especificados no primeiro vector;
   }
```

Proc. 7.13: Geração do primeiro vector para o teste de ligações pertencentes a grupos não BST.

A geração de cada um dos (M-1) vectores seguintes deverá levar já em conta a

especificação dos valores esperados, e das respectivas máscaras, relativamente ao vector anterior. O procedimento correspondente poderá ser especificado da forma que se segue:

```
/*  Procedimento responsável pela geração de (M-1) vectores a serem inseridos nas cadeias BST da
    CCI. Estes vectores destinam-se ao teste das ligações pertencentes a grupos de componentes não
    BST, de acordo com os conjuntos de vectores especificados para este fim.                    */

para (cada grupo de componentes não BST)
   para (cada um dos M-1 vectores restantes vectores especificados)
      para (cada cadeia BST)  {
         atribuir às células de saída os valores especificados;
         atribuir às células de controlo os valores especificados;
         atribuir aos nós OL que são células de entrada os valores esperados para o vector anterior;
         modificar as máscaras correspondentes a estes nós para permitir a comparação;
         atribuir às entradas primárias os valores especificados;
         atribuir aos respectivos sinais de controlo de estado os valores especificados;
         atribuir aos nós OL que são saídas primárias os valores esperados para o vector anterior;
         modificar as máscaras correspondentes a estes nós para permitir a comparação;
      }
```

Proc. 7.14: Geração de vectores para o teste de ligações pertencentes a grupos não BST.

Finalmente, a geração de um (M+1)º vector tem por objectivo deslocar para o exterior as respostas correspondentes ao último dos M vectores especificados, e compará-las com os respectivos valores esperados. Os valores que este vector desloca para o interior da cadeia BST não têm relevância, optando-se por manter a regra de aplicar a todas as ligações BST o valor lógico não dominante. A especificação do procedimento correspondente pode ser efectuada como se segue:

```
/*  Procedimento responsável pela geração do (M+1)º vector a ser inserido nas cadeias BST da CCI.
    Este vector tem por objectivo forçar o deslocamento para o exterior das respostas ao Mº vector
    aplicado, e aplica a todas as ligações BST o valor lógico não dominante.                    */
```

```
para (cada grupo de componentes não BST)
    para (cada cadeia BST)  {
        /*    mantêm-se os valores da pré-geração, para células BST de saída e de controlo    */
        atribuir aos nós OL que são células de entrada os valores esperados para o vector anterior;
        modificar as máscaras correspondentes a estes nós, para permitir a comparação;
        /*    mantêm-se os valores da pré-geração, para entradas primárias e sinais de controlo;    */
        atribuir aos nós OL que são saídas primárias os valores esperados para o vector anterior;
        modificar as máscaras correspondentes a estes nós, para permitir a comparação;
    }
```

Proc. 7.15: Geração do último vector para o teste de ligações pertencentes a grupos não BST.

7.4.3. Teste dos componentes BST

A especificação dos procedimentos principais relativos ao teste de componentes será feita em duas etapas, correspondentes à execução do auto-teste em componentes BST que suportem esta possibilidade, e ao teste dos restantes componentes BST (por reconfiguração das células BST para funcionarem em modo de teste interno).

7.4.3.1. Teste de componentes BST com auto-teste incorporado

Esta etapa só terá lugar se existirem componentes com auto-teste incorporado, e será realizada pelo deslocamento do código da instrução *Runbist* para os registos de instrução dos componentes que dispuserem desta facilidade, sendo para os restantes enviado o código da instrução *Bypass*. A execução das funções de auto-teste terá lugar durante a permanência dos controladores do TAP no estado *Run Test / Idle*, e terá uma duração não inferior ao imposto pelo maior número de impulsos requeridos no relógio de teste (TCK), por qualquer dos componentes. A execução das funções de auto-teste deverá ser seguida pela leitura dos registos de dados que contêm os resultados correspondentes, o que permite desde já identificar dois procedimentos distintos.

Adicionalmente, e se for requerida a aplicação de valores pré-especificados nos pinos dos componentes com auto-teste incorporado, durante a respectiva execução, esta etapa dará origem a mais dois procedimentos, responsáveis pela geração do vector que envia os códigos da instrução *Sample/Preload* (ou *Bypass*), e do vector que insere na cadeia BST os

valores pretendidos.

A especificação do procedimento responsável pela geração do vector que envia para cada registo de instrução o código de *Sample/Preload*, ou de *Bypass* (quando não for requerida a aplicação de valores pré-especificados nos pinos de saída), pode ser efectuada como se apresenta a seguir:

```
/*  Procedimento responsável pela geração do vector que envia para os registos de instrução os
    códigos de Sample/Preload, ou de Bypass (para aqueles em que não for requerida a aplicação
    de valores pré-especificados nos pinos, durante a execução do auto-teste).          */

para (cada componente)
   para (cada bit do registo de instrução)
      /*  valor a deslocar para o interior da cadeia BST.                              */
      se (é requerida a aplicação de valores pré-especificados nos pinos, durante o auto-teste)
         o valor a deslocar é o correspondente à instrução Sample/Preload;
      senão
         o valor a deslocar é o correspondente à instrução Bypass;
```

Proc. 7.16: Geração do vector que envia os códigos de *Sample/Preload* ou *Bypass*.

O procedimento responsável pela geração do vector que insere na cadeia BST os valores pré-especificados, de acordo com a informação que descreve a infraestrutura BST no componente (ver quadro 7.9), pode por sua vez ser especificado como se apresenta a seguir:

```
/*  Procedimento responsável pela geração do vector que insere os valores pré-especificados a
    aplicar nos pinos de saída, durante a realização do auto-teste (estes valores deverão constar da
    descrição de cada componente, nos casos em que este passo deva ter lugar).          */

para (cada componente)
   se (é requerida a aplicação de valores pré-especificados nos pinos, durante o auto-teste)
      para (cada bit do registo boundary scan)
         /*  valor a deslocar para o interior dos registos BST                         */
         o valor a deslocar é o especificado na descrição do componente;
```

```
senão
    /*    valor a deslocar (para o registo de BYPASS)                    */
    o valor a deslocar é (arbitrariamente) posto a 1;
```

Proc. 7.17: Geração do vector a aplicar nos pinos de saída dos componentes durante o auto-teste.

Repare-se que este procedimento não gera os valores esperados para os bits deslocados para o exterior da cadeia BST, durante a inserção do vector gerado, nem os valores para as respectivas máscaras, uma vez que o seu deslocamento para o interior da cadeia BST não é acompanhado pelo deslocamento para o exterior de respostas com significado.

A geração do vector que contem os códigos de *Runbist* pode ser realizada pelo procedimento que se apresenta a seguir:

```
/*  Procedimento responsável pela geração do vector que envia para cada registo de instrução o
    código de Runbist, ou de Bypass (quando não for suportada a execução de auto-teste).   */

para (cada componente)
    para (cada bit do registo de instrução)
        /*    valor a deslocar para o interior da cadeia BST                 */
        se (for suportada a execução de auto-teste)
            o valor a deslocar é o que corresponde à instrução Runbist;
        senão
            o valor a deslocar é o que corresponde à instrução Bypass;
```

Proc. 7.18: Geração do vector que envia os códigos de *Runbist* ou *Bypass*.

O vector que efectua a leitura dos resultados do auto-teste pode por sua vez ser gerado pelo procedimento seguinte:

```
/*  Procedimento responsável pela geração do vector que efectua a leitura dos resultados do auto-
    teste. Este vector tem apenas por objectivo forçar o deslocamento para o exterior do conteúdo dos
    registos seleccionados pela instrução Runbist, pelo que os valores deslocados para o interior
```

```
    destes registos são (arbitrariamente) colocados em 1.                    */

para (cada componente)
  se (for suportada a execução de auto-teste)
    para (cada bit do registo que contem o resultado)  {
      /*    valor a deslocar para o interior da cadeia BST                   */
      o valor a deslocar é (arbitrariamente) posto a 1;
      /*    valor esperado para o bit que será deslocado para o exterior     */
      o valor esperado é o especificado na descrição do componente;
      /*    máscara                                                          */
      o valor da máscara é de modo a permitir a comparação;
    }
  senão  {
    /*    valor a deslocar (para o registo de BYPASS)                        */
    o valor a deslocar é (arbitrariamente) posto em 1;
    /*    valor esperado                                                     */
    o valor esperado é 0;
    /*    máscara                                                            */
    o valor da máscara é de modo a permitir a comparação;
  }
```

Proc. 7.19: Geração do vector que permite a leitura dos resultados do auto-teste.

Também neste caso a possível existência de duas cadeias BST não implica modificações nos procedimentos descritos, devendo apenas ser repetida a sua execução relativamente a cada cadeia.

7.4.3.2. Teste de componentes BST sem auto-teste incorporado

Esta etapa só terá lugar quando existirem componentes BST para os quais tenha lugar a especificação de um conjunto de vectores destinados ao teste da sua lógica interna, usando para o efeito o seu próprio registo *boundary scan*, com as células BST configuradas em modo de teste interno (instrução *Intest*). Isto significa que a realização deste teste requer a existência de dois tipos de procedimentos, o primeiro destinado ao envio da instrução *Intest*, para os componentes que serão testados por este processo, e o segundo para permitir a aplicação dos vectores especificados na descrição de cada um destes componentes. Tal

como sucedeu já anteriormente em relação ao teste das ligações BST, e das ligações pertencentes a grupos de componentes não BST, a realização deste teste terá lugar por deslocamento sucessivo de estímulos e respostas, sendo o deslocamento de um vector para o interior da cadeia BST acompanhado pelo deslocamento para o exterior das respostas correspondentes ao vector anterior.

A geração do vector que envia os códigos correspondentes à instrução *Intest* pode ser realizada pelo procedimento seguinte:

```
/*  Procedimento responsável pela geração do vector que envia para os registos de instrução os
    códigos de Intest, ou de Bypass (para aqueles em que não for pretendida a realização de um
    teste através do registo boundary scan).                                              */

para (cada componente)
   para (cada bit do registo de instrução)
      /*   valor a deslocar para o interior da cadeia BST                                 */
      se (for requerida a realização de um teste através do registo boundary scan)
         o valor a deslocar é o que corresponde à instrução Intest;
      senão
         o valor a deslocar é o que corresponde à instrução Bypass;
```

Proc. 7.20: Geração do vector que envia os códigos de *Intest* ou *Bypass*.

A geração dos vectores que integram os valores especificados para os componentes a testar pode ser realizada por procedimentos semelhantes aos que foram já apresentados relativamente ao teste das ligações pertencentes a grupos de componentes não BST. O primeiro vector a deslocar para o interior da(s) cadeia(s) BST não requer a comparação com os valores deslocados para o exterior, pelo que a sua especificação pode ser apresentada como se indica a seguir:

```
/*  Procedimento responsável pela geração do primeiro vector destinado ao teste interno de
    componentes, através dos registos boundary scan, de acordo com um conjunto de valores
    especificados para este fim.                                                          */
```

```
para (cada cadeia BST)
   para (cada componente)
      se (for requerida a realização de um teste interno através do registo boundary scan)
         /*    valores a deslocar para a periferia da lógica interna do componente        */
         para (cada célula do registo boundary scan)
            atribuir a esta célula o valor especificado no primeiro vector;
      senão
         /*    valor a deslocar para o registo de BYPASS                                  */
         atribuir um 1 à posição correspondente;
```

Proc. 7.21: Geração do primeiro vector para o teste de componentes BST sem auto-teste.

A integração dos restantes vectores será então realizada por um procedimento que leva já em conta a especificação de valores esperados, e das respectivas máscaras, relativamente a cada vector anterior. Este procedimento poderá ser especificado da forma que se segue:

```
/*  Procedimento responsável pela geração dos restantes vectores destinados ao teste interno de
    componentes, através dos respectivos registos boundary scan, de acordo com os conjuntos de
    valores especificados para este fim.                                                  */

para (cada cadeia BST)
   para (cada componente)
      para (cada um dos restantes vectores especificados)
         se (for requerida a realização de um teste interno através do registo boundary scan)
            /*    valores a deslocar para a periferia da lógica interna do componente    */
            para (cada célula do registo boundary scan)  {
               atribuir a esta célula o valor especificado para este vector;
               atribuir a esta célula o valor esperado para o vector anterior;
               atribuir à máscara de comparação o valor especificado para o vector anterior;
            }
         senão  {
            /*    valor a deslocar para o registo de BYPASS                              */
            atribuir um 1 à posição correspondente;
            atribuir um 0 ao respectivo valor esperado;
```

```
        atribuir à máscara o valor que permite a comparação;
    }
```

Proc. 7.22: Geração de vectores para o teste de componentes BST sem auto-teste.

A geração de um vector adicional tem por objectivo permitir o deslocamento para o exterior das respostas correspondentes ao último vector especificado para este tipo de teste. Uma vez que não existe especificação para os valores a deslocar para o interior dos registos *boundary scan*, opta-se por repetir o deslocamento do último vector especificado. O procedimento correspondente pode ser apresentado como se segue:

```
/*  Procedimento responsável pela geração do vector destinado a permitir a leitura das respostas,
    relativamente ao último vector especificado para o teste interno de componentes, através dos
    respectivos registos boundary scan.                                                    */

para (cada cadeia BST)
  para (cada componente)
    se (for requerida a realização de um teste interno através do registo boundary scan)
      /*  valores a deslocar para a periferia da lógica interna do componente          */
      para (cada célula do registo boundary scan) {
        atribuir a esta célula o valor especificado no último vector;
        atribuir a esta célula o valor esperado para o último vector;
        atribuir à máscara de comparação o valor especificado para o último vector;
      }
    senão {
      /*  valor a deslocar para o registo de BYPASS                                    */
      atribuir um 1 à posição correspondente;
      atribuir um 0 ao respectivo valor esperado;
      atribuir à máscara o valor que permite a comparação;
    }
```

Proc. 7.23: Geração do último vector para o teste de componentes BST sem auto-teste.

Também para o teste interno de componentes, através dos respectivos registos

boundary scan, a possível existência de duas cadeias BST não implica modificações nos procedimentos descritos, devendo apenas ser repetida a sua execução relativamente a cada cadeia.

7.5. INTEGRAÇÃO DO CONJUNTO DE VECTORES GERADOS

Esta secção descreve como realizar a integração do conjunto de vectores gerados pelos procedimentos descritos na secção anterior, de modo a produzir um programa de teste especificado em termos do conjunto de instruções (*assembly*) suportadas pelo controlador residente.

Apesar de a conversão entre os vectores de teste gerados, e as instruções do controlador residente, se resumir essencialmente à escolha entre NSHF e NSHFCP (conforme o deslocamento seja para efectuar sem ou com comparação, respectivamente), e à determinação do respectivo argumento, a geração do programa de teste completo requer ainda a inclusão de um largo conjunto de instruções complementares. Estão nestas circunstâncias, entre outras, as instruções que implementam o protocolo de sincronismo com os recursos de teste exteriores (de acordo com a sequência de operações descritas no quadro 7.3), bem como as que promovem as necessárias transições através do diagrama de estados dos controladores do TAP.

A detecção de erros durante a execução do programa de teste poderá ou não dar origem à sua interrupção, conforme o tipo de erro detectado, e de acordo com o que foi dito quando se descreveu a estratégia global de teste. Adicionalmente, a detecção de uma falta será memorizada no registo de estados do controlador residente, sendo a sua identificação efectuada a partir das atribuições que se especificam no quadro 7.12.

O código gerado para o controlador residente deverá responsabilizar-se por efectuar a selecção da *flag* de erro adequada (através das instruções SERFLG), previamente à aplicação dos vectores para a detecção de cada um dos tipos de faltas referidas no quadro 7.12. Recorde-se que, quando existirem duas cadeias BST, a execução das instruções SELTAP dará automaticamente lugar à comutação do grupo de *flags* de erro seleccionado, passando a estar activa a *flag* correspondente à indicação dada pela última instrução SERFLG, mas agora para o novo TAP seleccionado.

Tipo de falta detectada	*Flag* de erro
Falta no percurso série (TDO-TDI)	0
Erro na leitura dos códigos de identificação	1
Erro na leitura dos códigos de identificação adicional	2
Falta de continuidade numa ligação BST	3
Ligações BST em curto-circuito	4
Falta em ligações pertencentes a grupos de componentes não BST	5
Erro no auto-teste de componentes BST	6
Erro no teste de componentes BST sem auto-teste	7

Quadro 7.12: Atribuições entre faltas e *flags* de erro.

A contribuição de cada etapa para a geração do programa de teste final, a partir dos vectores gerados por cada um dos procedimentos descritos na secção anterior, será agora descrita. Cada linha de código apresentada consiste numa marca (*label*, opcional), na mnemónica da instrução, num argumento (quando tiver lugar), e num comentário (opcional). A marca tem por objectivo melhorar a legibilidade do programa produzido, e será empregue sempre que tiver lugar o deslocamento de um dos vectores gerados. Considera-se ainda que a inclusão de comentários será delimitada à esquerda pelo carácter ";", e que se prolongará até ao fim da respectiva linha.

7.5.1. Integração dos vectores destinados ao teste da infraestrutura BST da CCI

O teste da infraestrutura BST é constituído pelas três etapas já anteriormente descritas, que consistem na inicialização da lógica BST em cada componente, no deslocamento através dos registos de instrução, e na leitura dos registos de identificação, quando disponíveis. A existência de componentes que disponham de códigos de identificação adicionais exige ainda o envio da instrução que os selecciona, e a leitura do respectivo valor.

A integração dos vectores de teste gerados pelos procedimentos descritos na secção 7.4.1 (Teste da infraestrutura BST) terá início através da seguinte sequência de código, destinada a provocar a inicialização da infraestrutura BST:

```
;
;  teste da infraestrutura BST
```

```
;
;  a cadeia BST 0, e a flag de erro 0, já estão seleccionados por omissão
        TMS1            ;  inicia a aplicação de cinco impulsos em TCK, com TMS em 1
        TMS1
        TMS1
        TMS1
        TMS1            ;  o estado actual é Test Logic Reset
```

A colocação dos controladores do TAP no estado *Shift-IR*, a que se segue o teste à operacionalidade do percurso série, por deslocamento através dos registos de instrução, será agora descrito. Este deslocamento serve simultaneamente para enviar o código que selecciona o registo de identificação, relativamente a todos os componentes que dispuserem desta facilidade. A detecção de um erro deverá interromper a realização do teste, tal como se definiu quando foi especificada a estratégia global.

```
        TMS0            ;  passa para Run Test / Idle
        TMS1            ;  passa para Select-DR-scan
        TMS1            ;  passa para Select-IR-scan
        TMS0            ;  passa para Capture-IR
        TMS0            ;  passa para Shift-IR
        LD      CNT, <conteúdo>
;  o conteúdo do contador corresponde ao tamanho do vector gerado pelo procedimento 7.2
inf... †    NSHFCP  <argumento>
;  o argumento é obtido a partir do vector gerado pelo procedimento 7.2
        JPE     fim
```

A colocação dos controladores do TAP no estado *Shift-DR* deverá agora ter lugar, de modo a permitir a leitura dos registos de dados seleccionados (identificação, ou *BYPASS*). Entretanto, a *flag* de erro seleccionada deverá passar a ser a que corresponde a um erro na leitura dos códigos de identificação dos componentes.

```
        TMS1            ;  passa para Update-IR
```

---

† A marca *inf* será seguida por um valor numérico que identifica a ordem do vector, na sequência correspondente ao teste da infraestrutura BST (inf1, inf2, ...).

```
        TMS1            ; passa para Select-DR-scan
        TMS0            ; passa para Capture-DR
        TMS0            ; passa para Shift-DR
        SERFLG1         ; selecciona a flag de erro na leitura dos códigos de identificação
        LD      CNT, <comprimento da cadeia BST>
;  o comprimento da cadeia BST seleccionada é o somatório dos tamanhos dos registos de
;  identificação (nos componentes em que existirem), mais o número de componentes sem este registo
;  (em BYPASS)
inf...  NSHFCP  <argumento>
;  o argumento é obtido a partir do vector gerado pelo procedimento 7.3
        JPE     fim
```

A próxima instrução a inserir corresponderá à que permite a leitura dos códigos de identificação adicionais (*Usercode*), nos componentes que dispuserem desta possibilidade. Para os componentes em que tal não aconteça, deverá ser enviada a instrução que selecciona o registo de *BYPASS*. Para este efeito, os controladores do TAP deverão ser colocados no estado *Shift-IR*, o que dá origem à seguinte sequência de código:

```
        TMS1            ; passa para Update-DR
        TMS1            ; passa para Select-DR-scan
        TMS1            ; passa para Select-IR-scan
        TMS0            ; passa para Capture-IR
        TMS0            ; passa para Shift-IR
```

O segmento de código descrito a seguir deverá ser incluído apenas quando existirem componentes que disponham de um código de identificação adicional. A detecção de um erro na leitura deste código de identificação adicional deverá ser memorizada na *flag* de erro correspondente, e dar origem à interrupção do programa de teste. O segmento de código correspondente deverá iniciar-se com a selecção da *flag* de erro associada a este passo (de acordo com a especificação apresentada no quadro 7.12), e poderá apresentar-se como se segue:

```
        SERFLG2         ; selecciona a flag de erro na leitura do código de ident. adicional
        LD      CNT, <comprimento da cadeia BST>
;  o comprimento da cadeia BST seleccionada é o somatório dos tamanhos dos registos de instrução
```

```
inf...      NSHF    <argumento>
;  o argumento é obtido a partir do vector gerado pelo procedimento 7.4
           TMS1            ;  passa para Update-IR
           TMS1            ;  passa para Select-DR-scan
           TMS0            ;  passa para Capture-DR
           TMS0            ;  passa para Shift-DR
           LD      CNT, <comprimento da cadeia BST>
;  o comprimento da cadeia BST seleccionada é o somatório dos tamanhos dos registos de
;  identificação (nos componentes que disponham de código de identificação adicional), mais o
;  número dos restantes componentes BST nesta cadeia
inf...      NSHFCP  <argumento>
;  o argumento é obtido a partir do vector gerado pelo procedimento 7.3, mas agora com o código de
;  identificação substituído pelo código de identificação adicional
           JPE     fim
```

O envio da instrução *Sample/Preload* deverá agora ter lugar, a que se seguirá uma sequência de transição de estados que coloca nas saídas das células BST os valores presentes nas saídas da lógica interna dos componentes. A esta sequência deve finalmente seguir-se o envio da instrução *Extest*, e a colocação dos controladores do TAP no estado *Shift-DR*, o que deixa a infraestrutura BST preparada para a realização do teste correspondente à etapa seguinte (teste de ligações).

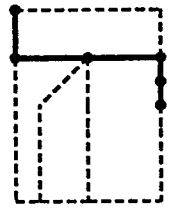
```
           TMS1            ;  passa para Update-DR
           TMS1            ;  passa para Select-DR-scan
           TMS1            ;  passa para Select-IR-scan
           TMS0            ;  passa para Capture-IR
           TMS0            ;  passa para Shift-IR
           LD      CNT, <comprimento da cadeia BST>
;  o comprimento da cadeia BST seleccionada é o somatório dos tamanhos dos registos de instrução
inf...      NSHF    <argumento>
;  o argumento é obtido a partir do vector gerado pelo procedimento 7.5, que deverá enviar o código
;  da instrução Sample/Preload
           TMS1            ;  passa para Update-IR
           TMS1            ;  passa para Select-DR-scan
           TMS0            ;  passa para Capture-DR
           TMS1            ;  passa para Exit1-DR
```

```
        TMS1            ; passa para Update-DR
        TMS1            ; passa para Select-DR-scan
        TMS1            ; passa para Select-IR-scan
        TMS0            ; passa para Capture-IR
        TMS0            ; passa para Shift-IR
        LD      CNT, <comprimento da cadeia BST>
; o comprimento da cadeia BST seleccionada é o somatório dos tamanhos dos registos de instrução
inf...  NSHF    <argumento>
; o argumento é obtido a partir do vector gerado pelo procedimento 7.5, que deverá enviar o código
; da instrução Extest
        TMS1            ; passa para Update-IR
        TMS1            ; passa para Select-DR-scan
        TMS0            ; passa para Capture-DR
        TMS0            ; passa para Shift-DR
```

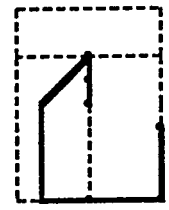
Nas CCIs em que existirem duas cadeias BST, deverá agora ter lugar a repetição dos passos descritos para a segunda cadeia, sendo esta sequência precedida pela comutação do TAP, e terminando da mesma maneira:

```
        SELTAP1         ; selecciona a cadeia BST 1
        ...
        <repete os blocos de código descritos para o teste da infraestrutura BST>
        ...
        SELTAP0         ; retorna à cadeia BST 0
```

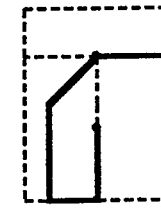
As sequências de transição de estados impostas pelas etapas descritas estão sintetizadas na figura 7.4, onde se ilustram sobre uma representação simplificada do diagrama de estados do controlador do TAP (ver figura 2.7). A existência de componentes que disponham de códigos de identificação adicional requer a repetição das transições ilustradas nas figuras 7.4.(b) e (c), para permitir a respectiva leitura.



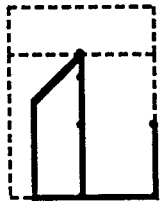
(a) Desde *Test Logic Reset* até *Shift-IR*, para permitir o envio da instrução *Idcode*.



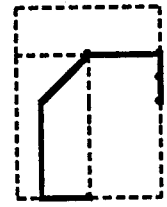
(b) Desde *Exit1-IR* até *Shift-DR*, para permitir a leitura dos códigos de identificação.



(c) Desde *Exit1-DR* até *Shift-IR*, para permitir o envio da instrução *Sample/Preload*.



(d) Desde *Exit1-IR* até *Update-DR*, para transferir as saídas da lógica interna para o interior das células BST.



(e) Desde *Update-DR* até *Shift-IR*, para permitir o envio da instrução *Extest*.

(f) Desde *Exit1-IR* até *Shift-DR*, para preparar a infraestrutura BST para a etapa seguinte.

Fig.7.4: Teste da infraestrutura BST: Sequências de transição de estados.

7.5.2. Integração dos vectores destinados ao teste das ligações

Tanto os vectores gerados para o teste das ligações BST, como para o teste das ligações pertencentes a grupos de componentes não BST, serão considerados nesta secção. Esta etapa tem início com a infraestrutura BST já preparada pela etapa anterior, que enviou a instrução *Extest* para todos os componentes, e com a garantia de que os valores presentes nos pinos de saída não provocarão conflitos no acesso a barramentos. A integração do conjunto de vectores de teste gerados pelos procedimentos apresentados na secção 7.4.2 (Teste das ligações) far-se-á então de acordo com um dos dois ciclos principais descritos no quadro 7.3, conforme existir apenas uma, ou duas cadeias BST. Para o caso de existirem duas cadeias, o segmento de código correspondente pode apresentar-se como se segue:

```
;
;  teste das ligações
;
```

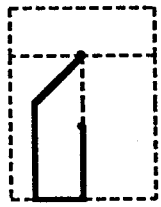
```
...

        LD      CNT, <comprimento da cadeia BST>
;  o compr. da cadeia seleccionada é o somatório dos tamanhos dos registos BST na cadeia BST 0
lig... †    NSHFCP  <argumento>
;  o argumento é obtido a partir de um vector gerado pelos procedimentos 7.6, ou 7.10 a 7.15
        JPE     fim     ;  interrompe o teste, se for detectada uma falta
        TMS1            ;  passa para Update-DR, na cadeia BST 0
        TMS1            ;  passa para Select-DR-scan, na cadeia BST 0
        SELTAP1         ;  selecciona a cadeia BST 1
        LD      CNT, <comprimento da cadeia BST>
;  o compr. da cadeia seleccionada é o somatório dos tamanhos dos registos BST na cadeia BST 1
lig...      NSHFCP  <argumento>
;  o argumento é obtido a partir de um vector gerado pelos procedimentos 7.6, ou 7.10 a 7.15
        JPE     fim     ;  interrompe o teste, se for detectada uma falta
        TMS1            ;  passa para Update-DR, na cadeia BST 1
        TMS1            ;  passa para Select-DR-scan, na cadeia BST 1
        SSA1            ;  sobe o sinal SSA
        WSA1            ;  espera que o sinal WSA suba
        TMS0            ;  passa para Capture-DR, na cadeia BST 1
        TMS0            ;  passa para Shift-DR, na cadeia BST 1
        SELTAP0         ;  selecciona a cadeia BST 0
        TMS0            ;  passa para Capture-DR, na cadeia BST 0
        TMS0            ;  passa para Shift-DR, na cadeia BST 0
        SSA0            ;  desce o sinal SSA
        WSA0            ;  espera que o sinal WSA desça
```

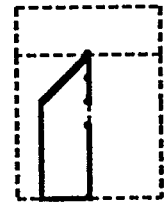
A sequência de transição de estados imposta pela realização do ciclo descrito está ilustrada na figura 7.5, onde se apresentam os percursos correspondentes às cadeias BST existentes.

---

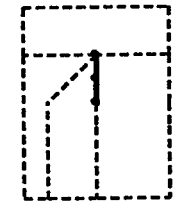
† As marcas *lig* identificam os vectores envolvidos no teste de ligações, e serão seguidas por um valor numérico que indica a ordem do vector (lig1, lig2, ...).



(a) Desde *Exit1-DR* até *Select-DR-scan* (cadeia BST 0), para aplicar o vector na cadeia BST 0.



(b) Desde *Exit1-DR* até *Shift-DR* (cadeia BST 1), para aplicar o vector e capturar as respostas, na cadeia BST 1.



(c) Desde *Select-DR-scan* até *Shift-DR* (cadeia BST 0), para capturar as respostas, na cadeia BST 0.

Fig.7.5: Teste de ligações: Sequência de transição de estados.

Para o caso de existir apenas uma cadeia BST, e ainda de acordo com a sequência de operações elementares descrita no quadro 7.3, o ciclo de código correspondente pode ser apresentado como se segue:

```
        LD      CNT, <comprimento da cadeia BST>
;  o compr. da cadeia seleccionada é o somatório dos tamanhos dos registos BST
lig...  NSHFCP  <argumento>
;  o argumento é obtido a partir de um vector gerado pelos procedimentos 7.6, ou 7.10 a 7.15
        JPE     fim
;       interrompe o teste, se for detectada uma falta
        TMS1            ;  passa para Update-DR
        TMS1            ;  passa para Select-DR-scan
        SSA1            ;  sobe o sinal SSA
        WSA1            ;  espera que o sinal WSA suba
        TMS0            ;  passa para Capture-DR
        TMS0            ;  passa para Shift-DR
        SSA0            ;  desce o sinal SSA
        WSA0            ;  espera que o sinal WSA desça
```

A sequência de transição de estados imposta pelo ciclo descrito é a mesma que se ilustra na figura 7.5.(b). Adicionalmente, e para ambos os casos (uma ou duas cadeias BST), deverão ainda ser observadas as seguintes directivas:

- O primeiro ciclo deve empregar a instrução NSHF (em vez de NSHFCP), uma vez que os resultados deslocados para o exterior, com a inserção do primeiro vector,

não constituem respostas com significado. Nestas circunstâncias, não terá lugar a inclusão da instrução JPE.

- Ainda para o primeiro ciclo, e posteriormente para os dois outros ciclos em que se passam a aplicar vectores de teste destinados à detecção de faltas de outro tipo (curto-circuitos entre ligações BST, faltas em ligações pertencentes a grupos de componentes não BST), deverá incluir-se a correspondente instrução SERFLG no início, de acordo com as atribuições apresentadas anteriormente no quadro 7.12: SERFLG3 para as faltas de continuidade em ligações BST, SERFLG4 para os curto-circuitos em ligações BST, e SERFLG5 para as faltas em ligações pertencentes a grupos de componentes não BST.
- O último ciclo não necessita de incluir a parte do código correspondente à aplicação de valores e captura de respostas, uma vez que se destina apenas a permitir o deslocamento para o exterior das respostas ao último vector aplicado para o teste de ligações. Neste sentido, a sequência de código para este ciclo deverá terminar com a instrução JPE, em cada cadeia BST.

7.5.3. Integração dos vectores destinados ao teste dos componentes BST

A integração dos vectores destinados ao teste de componentes terá lugar em duas etapas, a primeira correspondente aos componentes BST que dispõem de auto-teste incorporado, e a segunda aos restantes componentes BST.

Quando existirem componentes que disponham de auto-teste incorporado, terá lugar a inclusão de um segmento de código que tem início pela selecção da *flag* de erro correspondente:

```
;
;  teste de componentes
;
        SERFLG6          ;  selecciona a flag de erro no auto-teste de componentes
```

Para o caso de existirem componentes que requeiram a aplicação de valores pré-especificados nos seus pinos de saída, durante a realização das funções de auto-teste, terá lugar a inclusão da sequência de código descrita a seguir. Esta sequência assume que os controladores do TAP se encontram previamente no estado *Exit1-DR*, e envia os códigos das

instruções *Sample/Preload* ou *Bypass*.

```
        TMS1            ; passa para Update-DR
        TMS1            ; passa para Select-DR-scan
        TMS1            ; passa para Select-IR-scan
        TMS0            ; passa para Capture-IR
        TMS0            ; passa para Shift-IR
        LD      CNT, <comprimento da cadeia BST>
; o compr. da cadeia seleccionada é o somatório dos tamanhos dos registos de instrução
com... †    NSHF    <argumento>
; o argumento é obtido a partir do vector gerado pelo procedimento 7.16
        TMS1            ; passa para Update-IR
        TMS1            ; passa para Select-DR-scan
        TMS0            ; passa para Capture-DR
        TMS0            ; passa para Shift-DR
        LD      CNT, <conteúdo>
; o conteúdo do contador corresponde ao comprimento do vector gerado pelo procedimento 7.17
com...      NSHF    <argumento>
; o argumento é obtido a partir do vector gerado pelo procedimento 7.17
```

Deve agora ter lugar a inserção do vector que envia o código da instrução *Runbist* para os componentes que suportam a realização do auto-teste, seguida pela aplicação do número de impulsos necessários no relógio de teste (com os controladores do TAP no estado *Run Test / Idle*), de acordo com a seguinte sequência:

```
        TMS1            ; passa para Update-DR
        TMS1            ; passa para Select-DR-scan
        TMS1            ; passa para Select-IR-scan
        TMS0            ; passa para Capture-IR
        TMS0            ; passa para Shift-IR
        LD      CNT, <comprimento da cadeia BST>
; o compr. da cadeia seleccionada é o somatório dos tamanhos dos registos de instrução
```

---

† A marca *com* será seguida por um valor numérico que indica a ordem do vector, na sequência correspondente ao teste de componentes (com1, com2, ...).

```
com...      NSHF    <argumento>
;  o argumento é obtido a partir do vector gerado pelo procedimento 7.18
            TMS1            ;  passa para Update-IR
            TMS0            ;  passa para Run Test / Idle
            LD      CNT, <número de impulsos em TCK>
;  o número de impulsos em TCK é o maior número necessário, entre todos os CIs com auto-teste
            NTCK            ;  aplica o número de impulsos pretendidos
```

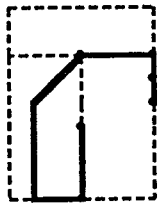
A leitura dos resultados do auto-teste deverá agora ter lugar, de acordo com a seguinte sequência de código:

```
            TMS1            ;  passa para Select-DR-scan
            TMS0            ;  passa para Capture-DR
            TMS0            ;  passa para Shift-DR
            LD      CNT, <comprimento da cadeia BST>
;  o compr. da cadeia seleccionada é o somatório dos tamanhos dos registos que contêm os resultados
;  do auto-teste, mais o número de componentes sem auto-teste (em BYPASS)
com...      NSHFCP  <argumento>
;  o argumento é obtido a partir do vector gerado pelo procedimento 7.19
```

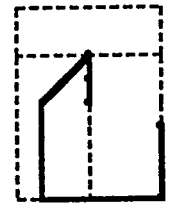
As sequências de transição de estados impostas pela realização do auto-teste em componentes BST estão ilustradas na figura 7.6, onde as correspondentes às alíneas (a) e (b) são opcionais.

Após a realização do auto-teste, deve ter lugar a preparação da infraestrutura BST para se proceder à última etapa, que diz respeito aos componentes BST sem auto-teste. A instrução *Intest* será enviada para os registos de instrução dos componentes a testar nesta etapa, sendo para os restantes enviada a instrução *Bypass*. A correspondente sequência de código pode apresentar-se como se segue:

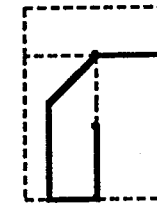
```
            SERFLG7         ;  selecciona a flag de erro em componentes BST sem auto-teste
            TMS1            ;  passa para Update-DR
            TMS1            ;  passa para Select-DR-scan
            TMS1            ;  passa para Select-IR-scan
            TMS0            ;  passa para Capture-IR
            TMS0            ;  passa para Shift-IR
```



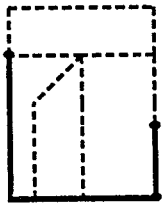
(a) Desde *Exit1-DR* até *Shift-IR*, para permitir o envio da instrução *Sample/Preload*.



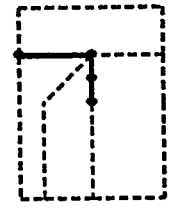
(b) Desde *Exit1-IR* até *Shift-DR*, para permitir o envio dos valores a aplicar aos pinos.



(c) Desde *Exit1-DR* até *Shift-IR*, para permitir o envio da instrução *Runbist*.



(d) Desde *Exit1-IR* até *Run Test / Idle*, para permitir a execução do auto-teste.



(e) Desde *Run Test / Idle* até *Shift-DR*, para permitir a leitura dos resultados.

Fig.7.6: Auto-teste de componentes: Sequências de transição de estados.

```
        LD      CNT, <comprimento da cadeia BST>
;  o compr. da cadeia seleccionada é o somatório dos tamanhos dos registos de instrução
com...  NSHF    <argumento>
;  o argumento é obtido a partir do vector gerado pelo procedimento 7.20
        TMS1            ;  passa para Update-IR
        TMS1            ;  passa para Select-DR-scan
```

A aplicação do conjunto de vectores especificados para os componentes a testar terá então lugar por execução de um conjunto de segmentos de código com a composição que se apresenta a seguir:

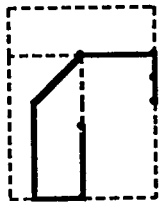
```
        LD      CNT, <comprimento da cadeia BST>
;  o comprimento da cadeia seleccionada é o somatório dos tamanhos dos registos boundary scan dos
;  componentes a testar por este processo, mais o número dos restantes componentes BST (em BYPASS)
com...  NSHFCP  <argumento>
;  o argumento é obtido a partir de um vector gerado pelos procedimentos 7.21, 7.22, ou 7.23
```

TMS1	;	passa para *Update-DR*
TMS1	;	passa para *Select-DR-scan*
TMS0	;	passa para *Capture-DR*
TMS0	;	passa para *Shift-DR*

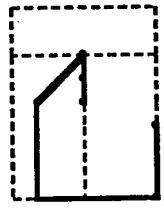
A geração do programa de teste correspondente a esta etapa deverá ainda observar as seguintes directivas:

- O primeiro ciclo deve empregar a instrução NSHF (em vez de NSHFCP), uma vez que os resultados deslocados para o exterior, com a inserção do primeiro vector, não correspondem a respostas com significado.
- O último ciclo não necessita de incluir a parte de código correspondente à aplicação do vector deslocado para o interior da cadeia BST, uma vez que se destina apenas a permitir o deslocamento para o exterior das respostas ao último vector especificado.

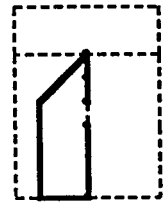
As sequências de transição de estados correspondentes à aplicação de um conjunto de vectores para o teste da lógica interna de componentes BST, através dos seus próprios registos *boundary scan*, estão ilustradas na figura 7.7.



(a) Desde *Exit1-DR* até *Shift-IR*, para permitir o envio da instrução *Intest*.



(b) Desde *Exit1-IR* até *Shift-DR*, para permitir o envio do primeiro vector.



(c) Desde *Exit1-DR* até *Shift-DR*, para permitir a aplicação dos restantes vectores.

Fig.7.7: Teste de componentes BST em modo interno: Sequência de transição de estados.

A detecção de uma avaria num componente não provoca a interrupção do programa de teste, de acordo com a estratégia global que foi apresentada (ver secção 7.2.3). Esta afirmação tanto é válida relativamente aos componentes BST que dispõem de auto-teste, como aos que são testados através do respectivo registo *boundary scan*, pelo que a instrução JPE não é empregue em nenhum dos casos.

Para o caso de existirem duas cadeias BST, deverá agora ter lugar a repetição dos passos descritos relativamente aos componentes da segunda cadeia (quer para o auto-teste,

quer para o teste interno através dos registos *boundary scan*), sendo esta sequência precedida pela comutação do TAP, e terminando da mesma maneira:

```
SELTAP1          ; selecciona a cadeia BST 1
...
<repete os blocos de código descritos para o auto-teste de componentes>
...
SELTAP0          ; retorna à cadeia BST 0
```

7.5.4. Conclusão do programa de teste

Após a integração dos vectores gerados pelos procedimentos descritos na secção 7.4 (Procedimentos principais), deverá ter lugar a inclusão do segmento que termina o programa de teste, e cuja apresentação pode ser feita da seguinte forma, quando existirem duas cadeias BST:

```
;
;  fim do programa de teste
;
fim      SELTAP1          ; selecciona a cadeia BST 1
         TMS1             ; inicia a aplicação de cinco impulsos em TCK, com TMS em 1
         TMS1
         TMS1
         TMS1
         TMS1             ; todos os componentes em Test Logic Reset, na cadeia BST 1
         SELTAP0          ; selecciona a cadeia BST 0
         TMS1             ; inicia a aplicação de cinco impulsos em TCK, com TMS em 1
         TMS1
         TMS1
         TMS1
         TMS1             ; todos os componentes em Test Logic Reset
         HALT             ; termina a execução
```

Se existir apenas uma cadeia BST, este segmento de código restringir-se-à às seis últimas instruções, passando a marca *fim* para a primeira instrução deste grupo.

Capítulo 8

Conclusão

Resultados

Perspectivas de futuro

8. CONCLUSÃO

O trabalho descrito nesta tese teve por componentes principais a especificação de uma arquitectura para um controlador residente destinado ao teste de CCIs com BST, e a especificação dos procedimentos que permitem a geração automática do programa de teste (GAPT), a partir da informação que descreve a CCI a testar.

A arquitectura especificada para o controlador residente é essencialmente constituída por um núcleo que constitui um processador dedicado para o teste da CCI, e por um bloco que permite a extensão do BST como barramento de teste a nível hierárquico superior. Também o barramento MTM (IEEE P1149.5 Std, 92) constitui uma alternativa a este nível, embora decorra ainda o seu processo de normalização. A implementação de modificações na arquitectura apresentada é no entanto facilitada pelo carácter modular que presidiu ao seu desenvolvimento, com o objectivo de maximizar a independência entre os seus blocos constituintes, e de definir de forma clara a respectiva interacção (Ferreira et al., 92b). A selecção do BST como barramento de teste a nível hierárquico superior apresenta porém a vantagem adicional de permitir uma estratégia de teste hierárquica em que o mesmo controlador pode ser usado a diferentes níveis.

O controlador proposto é particularmente importante para CCIs que empreguem circuitos integrados para aplicações específicas (ASICs), e componentes VLSI, que disponham de uma infraestrutura BST com acesso a funções de auto-teste, e em que os grupos de componentes não BST presentes sejam de dimensão e complexidade reduzidas. Nestas circunstâncias, e conforme foi discutido na secção 5.1 (Benefícios e requisitos de um controlador residente), um programa de teste com tamanho mínimo poderá atingir uma elevada cobertura de faltas estruturais.

A geração do programa de teste tem lugar de forma automática, a partir da informação que descreve as ligações na CCI (*netlist*), a implementação BST em cada componente, e os grupos de componentes não BST (incluindo a especificação dos respectivos vectores de teste). Estes três tipos de informação permitem gerar o programa de teste correspondente às três etapas principais do protocolo, que dizem respeito à infraestrutura BST, às ligações (incluindo as ligações pertencentes a grupos de componentes não BST), e aos componentes.

A geração automática do programa de teste (GAPT) é realizada com base no pressuposto de os modos de operação disponíveis para a infraestrutura BST não ultrapassarem as três instruções obrigatórias (*Bypass, Sample/Preload, Extest*) e as quatro instruções opcionais (*Idcode, Usercode, Intest, Runbist*) descritas na norma BST (IEEE Std

1149.1, 90, cap. 7). Com base nestes pressupostos, foi identificada a informação mínima necessária aos procedimentos que permitem a geração dos vectores de teste. Estes procedimentos foram descritos numa pseudo-linguagem de alto nível, e foi apresentada a integração dos vectores gerados, no sentido de produzir um programa de teste especificado em termos das instruções suportadas pelo controlador residente.

8.1. RESULTADOS

A arquitectura descrita para o controlador residente foi implementada num componente LCC com 68 pinos, em tecnologia CMOS (1.5μ), cuja folha de características se apresenta no anexo A. Este componente foi fabricado pela ES2 (*European Silicon Structures*), no âmbito da participação da Universidade do Porto no programa EUROCHIP (em Setembro de 1991), e ocupa uma área total com cerca de 24 mm^2. O projecto foi realizado em SOLO 1400, e teve por objectivo permitir uma frequência de funcionamento de 25 MHz.

Uma primeira versão deste processador fora anteriormente fabricada também no âmbito do programa EUROCHIP (em Dezembro de 1990), em encapsulamento DIP com 40 pinos. Esta primeira versão dispõe apenas de um subconjunto dos recursos disponíveis na segunda, e não inclui os blocos que permitem implementar uma estratégia de teste hierárquica. Ambas as versões se encontram a funcionar, incluídas em sistemas de demonstração. Esta primeira versão foi precedida por uma implementação em que um sistema baseado num microprocessador de 8 bits (Z80) foi usado para emular o conjunto de instruções pretendido.

As especificações apresentadas no capítulo 7 serviram de base à realização de uma primeira versão para um conjunto de programas que realiza a GAPT. Esta implementação baseia-se num conjunto de estruturas de dados que contêm a informação de entrada necessária, e produz os segmentos de programa de teste para todas as etapas do protocolo.

O anexo B descreve um exemplo de demonstração em que se ilustra a aplicação do controlador residente, e o programa produzido pela GAPT, para uma CCI que contem duas cadeias BST, e dois grupos de componentes não BST. Um destes grupos está rodeado por células de uma só cadeia, e o outro por células de ambas as cadeias.

8.2. PERSPECTIVAS DE FUTURO

A estratégia adoptada neste trabalho relativamente à geração de vectores para o teste de ligações torna difícil formular uma estimativa para a cobertura de faltas resultante, uma vez que envolve a reunião de conjuntos de vectores gerados para etapas supostamente

independentes: teste da infraestrutura BST na CCI, teste das ligações BST, e teste das ligações pertencentes a grupos de componentes não BST. Apesar de constituir uma estratégia comum para o teste de CCIs com implementações parciais de BST, como se pode constatar em diversos trabalhos publicados (Robinson et al., 90), (Jong, 90a, 90b), (Hansen, 91), (Lefebvre, 91), é também patente nestes trabalhos a dificuldade em se formular uma estimativa fiável para a qualidade do teste assim produzido. Esta dificuldade deve-se essencialmente à utilização de diferentes ferramentas para a geração dos vectores correspondentes às etapas referidas, mas está também relacionada com as diferentes condições de acesso (C&O) que caracterizam as ligações envolvidas, e os diferentes modelos de faltas, e algoritmos, que são empregues em cada caso.

A obtenção de uma estimativa fiável para a cobertura de faltas estruturais nas ligações de CCIs com implementações parciais de BST constitui deste modo um problema que precisa de ser resolvido, e que constituirá um tópico de trabalho futuro. Uma primeira aproximação a este problema poderá consistir na criação de um modelo de simulação para a CCI, no qual a funcionalidade de cada componente BST seja definida apenas em termos de uma infraestrutura BST mínima (omitindo a sua lógica interna). A simulação de faltas poderá então fornecer uma estimativa para a qualidade do teste proporcionado pelo conjunto total de vectores disponíveis, o que pode por sua vez conduzir a conclusões que influenciem a própria estratégia subjacente à respectiva geração.

Para além da questão referida nos parágrafos anteriores, as perspectivas de continuidade sugeridas por este trabalho identificam-se sobretudo em quatro domínios:

- Modificações na arquitectura do controlador de teste.
- Aplicações que tenham por base o controlador de teste desenvolvido.
- Refinamento do módulo de GAPT, de modo a suportar a geração automática de programas de teste que explorem modos de funcionamento mais poderosos do que os que foram assumidos.
- Novos módulos que complementem a GAPT, nomeadamente para permitir o apoio a aplicações com base no controlador desenvolvido.

As modificações na arquitectura do controlador pretenderão sobretudo suportar novos desenvolvimentos que tenham lugar em normas internacionais no domínio do teste, tais como a aprovação (esperada para um futuro próximo) do barramento MTM (IEEE P1149.5 Std, 92) como barramento de teste à escala do sistema. Esta aprovação não significa que a arquitectura proposta neste trabalho perca o seu interesse, até porque a adopção do MTM impede que se use o mesmo controlador em diferentes níveis hierárquicos. As características modulares da arquitectura proposta facilitam no entanto o desenvolvimento de um controlador apto a suportar outra estratégia de testabilidade

hierárquica, mas mantendo o núcleo descrito para o processador dedicado.

A inclusão de auto-teste no controlador revela-se igualmente uma característica importante, que deverá necessariamente estar presente num componente com estas características. Esta inclusão não teve lugar no protótipo fabricado, quer por razões de tempo, quer pelo facto de o objectivo principal no seu fabrico consistir em validar a arquitectura proposta. Por outro lado, os níveis de C&O no interior do componente permitem que um programa de teste reduzido, comandado a partir da sua infraestrutura BST, se revele suficiente para comutar o nível lógico em 99,93% dos nós, pelo que o teste de produção não revelou dificuldades (a especificação imposta pelo sistema de projecto usado requeria apenas que esta percentagem fosse não inferior a 95%).

O controlador desenvolvido pode entretanto ser empregue em aplicações alternativas àquelas para que foi originalmente concebido, estando actualmente em curso projectos que empregam este componente com dois fins distintos.

Um primeiro projecto consiste num sistema de demonstração para apoio ao ensino e divulgação da tecnologia BST, e que proporciona facilidades para o desenvolvimento e depuração de programas de teste. Este sistema possibilita a execução passo-a-passo de um programa de teste, a especificação de pontos de paragem (*breakpoints*) em função do conteúdo dos barramentos de dados, ou de endereços, a execução de segmentos do programa (por especificação do número de ciclos de relógio pretendidos), e ainda a execução em modo contínuo. O sistema suporta um canal de comunicação série para um computador pessoal, onde os programas de teste podem ser editados em linguagem simbólica (*assembly*), convertidos para código objecto do controlador, e enviados para execução no sistema de demonstração.

Um outro projecto actualmente em curso consiste no desenvolvimento de um testador baseado em computador pessoal, que é subsidiado pela Junta Nacional de Investigação Científica e Tecnológica (JNICT), através do contrato PMCT/C/TIT/937/90 (Ferreira et al., 91c, 91d). Este sistema emprega uma arquitectura que recorre ao processador desenvolvido para efectuar o teste de CCIs com BST, que não disponham de controlador residente, e inclui igualmente os blocos que permitem a aplicação de estímulos e captura de respostas através dos pinos de E/S primária da CCI a testar. Estão igualmente previstos neste sistema os blocos necessários para efectuar a deserialização das respostas deslocadas para o exterior da(s) cadeia(s) BST, por cada vector inserido, e que executam esta operação sob comando do controlador de teste (através da saída *DeserEn*, que indica para o exterior quais os ciclos de relógio em que a operação de deserialização deve ter lugar). Este sistema fará uso de uma versão expandida do conjunto de programas responsáveis pela GAPT, capaz de suportar operações de diagnóstico sobre o conjunto de respostas capturadas.

As perspectivas de evolução relativamente à GAPT dizem respeito à possibilidade de se gerarem segmentos de programa de teste que permitam explorar modos de operação mais poderosos para a infraestrutura BST, e também ao desenvolvimento de pré- e pós-processadores que permitam a leitura e escrita de informação em formatos normalizados. Esta segunda questão depende naturalmente dos progressos verificados para os esforços de normalização actualmente em curso, sobretudo no que respeita à aprovação de um formato que permita representar toda a informação relacionada com a implementação BST em cada componente (Maunder, 91).

O reconhecimento de modos de operação mais poderosos para a infraestrutura BST, para além dos correspondentes às três instruções obrigatórias, e quatro instruções opcionais, tem sobretudo interesse no que respeita à utilização de componentes BST que permitam efectuar a geração local de estímulos, e a compactação de respostas, para o teste de grupos de componentes não BST. Começam actualmente a estar disponíveis no mercado componentes BST com esta capacidade, que se traduz essencialmente na geração pseudo-aleatória de estímulos de teste e análise de assinatura (Whetsel, 91), ou na geração de vectores de acordo com algoritmos pré-definidos. Enquanto o primeiro caso pode facilitar substancialmente o teste de blocos de lógica combinacional, o segundo apresenta sobretudo interesse para o teste de blocos de memória (Kritter et al., 91), (Raghavachari, 91). Em qualquer dos casos, identificam-se imediatamente duas vantagens principais:

- Um novo vector é aplicado por cada impulso no relógio de teste (TCK), o que se revela muito mais eficiente do que a serialização necessária para a instrução *Extest* .
- O facto de os vectores não estarem armazenados na memória com o programa de teste constitui uma redução importante, atendendo ao elevado espaço de armazenamento que estes vectores podem requerer.

A geração do segmento de programa de teste necessário para explorar convenientemente os recursos locais destinados à geração e aplicação de vectores de teste apresenta no entanto requisitos em termos de informação de entrada cuja caracterização não foi ainda efectuada, e que terá que ser estudada. O facto de este processo ser necessariamente dependente da funcionalidade e características dos grupos de componentes não BST pode inclusivamente significar que seja requerido um nível de intervenção manual importante, restando ainda avaliar até que ponto será possível automatizar este processo.

Finalmente, o desenvolvimento de um módulo de diagnóstico permitirá complementar a GAPT, no sentido de desenvolver um suporte lógico para a arquitectura de um testador baseado no controlador descrito. Este módulo deverá permitir que a análise de respostas a um conjunto de vectores se traduza numa identificação das faltas presentes, e eventualmente na identificação de situações de ambiguidade que possam ser resolvidas por

posterior geração de novos segmentos de programa de teste. Este módulo está directamente relacionado com o desenvolvimento do testador que foi referido anteriormente, e pretende dotar o sistema resultante com a possibilidade de produzir a informação que oriente operações de manutenção.

Referências bibliográficas

Avra, L. 1987. A VHSIC ETM-Bus-Compatible Test and Maintenance Interface. *IEEE International Test Conference Proceedings*, 1987, pp. 964-971.

Bagge, C. and Driessen, W. 1989. *Tester Architecture.* Report R6.1, ESPRIT Project 2478 (Research into Boundary Scan Test Implementation), July 1989.

Ballew, W. and Streb, L. 1989. Board-Level Boundary Scan: Regaining Observability with an Additional IC. *IEEE International Test Conference Proceedings*, 1989, pp. 182-189.

Ballew, W. and Streb, L. 1992. Board-Level Boundary Scan: Regaining Observability with an Additional IC. *IEEE Transactions on Computer-Aided Design*, January 1992, Vol. 11, Nº 1, pp. 68-75.

Bassett, R., Gillis, P. and Shushereba, J. 1991. High-Density CMOS Multichip-Module Testing and Diagnosis. *IEEE International Test Conference Proceedings*, 1991, pp. 530-539.

Bateson, J. 1985. *In-Circuit Testing.* Van Nostrand Reinhold, 1985.

Beenker, F. 1985. Systematic and Structured Methods for Digital Board Testing. *IEEE International Test Conference Proceedings*, 1985, pp. 380-385.

Beenker, F., Eerdewijk, K. J., Gerritsen, R. B., Peacock, F. N. and Star, M. 1986. Macro Testing: Unifying IC and Board Test. *IEEE Design and Test of Computers*, Dec. 1986, pp. 26-32.

Bennetts, R. G. 1982. *Introduction to Digital Board Testing.* Crane Russak & Company. ISBN 0-8449-1385-0.

Bennetts, R. G. 1984. *Design of Testable Logic Circuits.* Addison-Wesley Publishers. ISBN 0-201-14403-4.

Bennetts, R. G. and Osseyran, A. 1991. IEEE Standard 1149.1-1990 on Boundary-Scan: History, Literature Survey, and Current Status. *Journal of Electronic Testing: Theory and Applications.* March 1991, Vol.2, Nº 1, pp. 11-25.

Bhavsar, D. 1990. Cell Designs that Help Test Interconnect Shorts. In Maunder, C. et al. *The Test Access Port and Boundary Scan Architecture.* The IEEE Computer Society Press. ISBN 0-8186-9070-4, pp. 183-189.

Bhavsar, D. 1991a. Testing Interconnections to Static RAMs. *IEEE Design and Test of Computers*, June 1991, pp. 63-71.

Bhavsar, D. 1991b. An Architecture for Extending the IEEE Standard 1149.1 Test Access Port to System Backplanes. *IEEE International Test Conference Proceedings*, 1991, pp. 768-776.

Bishop, P. E., Giles, G. L., Iyengar, S. N., Glover, C. T. and Law, W. 1990. Testability Considerations in the Design of the MC68340 Integrated Processor Unit. *IEEE International Test Conference Proceedings*, 1990, pp. 337-346.

Blahut, R. 1983. *Theory and Practice of Error Control Codes.* Addison-Wesley Publishing Company. ISBN 0-201-10102-5.

Breuer, M. A. and Lien, J.-C. 1988. A Test and Maintenance Controller for a Module Containing Testable Chips. *IEEE International Test Conference Proceedings*, 1988, pp. 502-513.

Brown, J. 1991. Implementing a Serial/Parallel Converter for Board Scan Test (poster). *European Test Conference Proceedings*, 1991, p. 530.

Bruce, W., Gallup, M., Giles, G. and Munns, T. 1991. Implementing 1149.1 on CMOS Microprocessors. *IEEE International Test Conference Proceedings*, 1991, pp. 879-886.

Budde, W. O. 1988. Modular Testprocessor for VLSI Chips and High-Density PC Boards. *IEEE Transactions on Computer Aided Design*, October 1988, Vol.7, Nº 10, pp. 1118-1124.

Bullock, M. 1987. Designing SMT Boards for In-Circuit Testability. *IEEE International Test Conference Proceedings*, 1987, pp. 606-613.

Cheng, W. T., Lewandowski, J. L. and Wu, E. 1990. Diagnosis for Wiring Interconnects. *IEEE International Test Conference Proceedings*, 1990, pp. 565-571.

Chiles, D. and DeJaco, J. 1991. Using Boundary Scan Description Language in Design. *IEEE International Test Conference Proceedings*, 1991, pp. 865-868.

Christofides, N. 1975. *Graph Theory - An Algorithmic Approach.* Academic Press. ISBN 0-12-174350-0.

Cortner, J. M. 1987. *Digital Test Engineering*. John Wiley & Sons. ISBN 0-471-85135-3.

DasGupta, S., Graf, M. C., Rasmussen, R. A., Walther, R. G. and Williams, T. W. 1984. Chip Partitioning Aid: A Design Technique for Partitionability and Testability in VLSI. *Design Automation Conference Proceedings*, 1984, pp. 203-208.

Dervisoglu, B. I. 1990. Towards a Standard Approach for Controlling Board-Level Test Functions. *IEEE International Test Conference Proceedings*, 1990, pp. 582-590.

Dervisoglu, B. I. 1991. Features of a Scan and Clock Resource Chip for Providing Access to Board-Level Test Functions. *Journal of Electronic Testing: Theory and Applications*. March 1991, Vol.2, Nº 1, pp. 107-115.

Dislis, C., Dear, I., Miles, J., Lau, S. and Ambler, A. 1989. Cost Analysis of Test Method Environments. *IEEE International Test Conference Proceedings*, 1989, pp. 875-883.

Driessen, W. and Bagge, C. 1990. *ATE: Boundary Scan Pin Characteristics*: Deliverable D6.1, ESPRIT Project 2478 (Research into Boundary Scan Test Implementation), March 1990.

Eerenstein, L. 1989. *TIM: Testability IMprover*. Philips Technical Note 013/89EN, 1989.

EIA EDIF 1987. *Electronic Design Interchange Format Version 2.0.0*. EDIF Steering Committee, Electronic Industries Association, 1987.

Eichelberger, E. and Williams, T. 1977. A Logic Design Structure for LSI Testability. *Design Automation Conference Proceedings*, 1977, pp. 462-468.

Etherington, A. 1987. Interfacing Design to Test Using the Electronic Design Interchange Format (EDIF). *IEEE International Test Conference Proceedings*, 1987, pp. 378-383.

Fasang, P. 1989. Boundary Scan and Its Application to Analog-Digital ASIC Testing in a Board/System Environment. *IEEE Custom Integrated Circuits Conference Proceedings*, 1989, pp. 22.4.1-22.4.4.

Ferreira, J. M., Matos, J. S. and Jong, F. 1989. *Test Pattern Generation Methodology for Full Boundary Scan Test*. Deliverable D4.1, ESPRIT Project 2478 (Research into Boundary Scan Test Implementation), December 1989.

Ferreira, J. M., Matos, J. S. and Jong, F. 1990. *High Level Specification of TPG Procedures for F-BST Boards*. Deliverable D4.2, ESPRIT Project 2478 (Research into Boundary Scan Test Implementation), August 1990.

Ferreira, J. M., Matos, J. S., Jong, F. and Bennetts, R. G. 1991a. A New Algorithm for Diagnosing Interconnect Faults on Boundary Scan Boards (poster). *European Test Conference Proceedings*, 1991, p. 502.

Ferreira, J. M., Matos, J. S. and Jong, F. 1991b. *Verification of TPG Procedures for F-BST Boards*. Report R4.3, ESPRIT Project 2478 (Research into Boundary Scan Test Implementation), June 1991.

Ferreira, J. M., Matos, J. S. e Pinto, F. S. 1991c. *Definição de Requisitos*. Relatório nº 1, Projecto JNICT PMCT/C/TIT/937/90 (Sistema de Validação e Teste de Cartas de Circuito Impresso com BST), Julho 1991.

Ferreira, J. M., Araújo. A. J., Pinto, F. S., Tavares, J. A., Alves, G. R. e Matos, J. S. 1991d. *Especificação da Arquitectura Global*. Relatório nº 2, Projecto JNICT PMCT/C/TIT/937/90 (Sistema de Validação e Teste de Cartas de Circuito Impresso com BST), Novembro 1991.

Ferreira, J. M., Matos, J. S. and Pinto, F. S. 1992a. Automatic Generation of a Single-Chip Solution for Board-Level BIST of Boundary Scan Boards. *European Design Automation Conference Proceedings*, March 1992, pp. 154-158.

Ferreira, J. M., Pinto, F. S. and Matos, J. S. 1992b. A Modular Architecture for Board-Level BIST of Boundary Scan Boards. *EuroASIC Conference Proceedings*, June 1992.

Ferreira, J. M., Matos, J. S. and Jong, F. 1992c. *Verification of TPG Procedures for BST Boards with Additional ST-Clusters*. Report R4.4, ESPRIT Project 2478 (Research into Boundary Scan Test Implementation), March 1992.

Fleming, P. 1990. Applications of the IEEE Std 1149.1: A Overview. In Maunder, C. et al. *The Test Access Port and Boundary Scan Architecture*. The IEEE Computer Society Press. ISBN 0-8186-9070-4, pp. 129-140.

Gallup, M. G., Ledbetter, W., McGarity, R., McMahan, S., Scheuer, K. C., Shepard, C. G. and Sood, L. 1990. Testability Features of the 68040. *IEEE International Test Conference Proceedings*, 1990, pp. 749-757.

Garey, M. R. and Johnson, D. S. 1979. *Computers and Intractability.* W. H. Freeman and Co. ISBN 0-7167-1044-7.

Gelsinger, P. 1987. Design and Test of the 80386 CPU. *IEEE Design and Test of Computers,* June 1987, pp. 42-50.

Goel, P. and McMahon, M. T. 1982. Electronic Chip-In-Place Test. *IEEE International Test Conference Proceedings,* 1982, pp. 83-90.

Goor, A. J. and Tetering, J. A. 1991. A Low-Cost Tester for Boundary Scan. *Microprocessors and Microsystems,* March 1991, Vol. 15, Nº 2, pp. 82-90.

Halliday, A., Young, G. and Crouch, A. 1989. Prototype Testing Simplified by Scannable Buffers and Latches. *IEEE International Test Conference Proceedings,* 1989, pp. 174-181.

Hansen, P. 1983. New Techniques for Manufacturing Test and Diagnosis of LSSD Boards. *IEEE International Test Conference Proceedings,* 1983, pp. 40-45.

Hansen, P. 1989. Testing Conventional Logic and Memory Clusters Using Boundary Scan Devices as Virtual ATE Channels. *IEEE International Test Conference Proceedings,* 1989, pp. 166-173.

Hansen, P. 1990. Taking Advantage of Boundary-Scan in Loaded-Board Testing. In Maunder, C. et al. *The Test Access Port and Boundary Scan Architecture.* The IEEE Computer Society Press. ISBN 0-8186-9070-4, pp. 81-96.

Hansen, P. 1991. Assessing Fault Coverage in Virtual In-Circuit Testing of Partial Boundary-Scan Boards. *European Test Conference Proceedings,* 1991, pp. 393-396.

Hassan, A., Rajski, J. and Agarwal, V. K. 1988. Testing and Diagnosis of Interconnects Using Boundary Scan Architecture. *IEEE International Test Conference Proceedings,* 1988, pp. 126-137.

Heyden, F. 1990. *Automatic Test Pattern Generation for Testing of the Boundary Scan Infrastructure.* Deliverable D4.1a, ESPRIT Project 2478 (Research into Boundary Scan Test Implementation), December 1990.

Hines, J. 1987. Where VHDL Fits Within the CAD Environment. *Design Automation Conference Proceedings,* 1987, pp. 491-494.

Hirzer, J. 1990. Testing Mixed Analog/Digital ICs. In Maunder, C. et al. *The Test Access Port and Boundary Scan Architecture.* The IEEE Computer Society Press. ISBN 0-8186-9070-4, pp. 199-204.

IEEE P1149.5 Std 1992. *Standard Backplane Module Test and Maintenance (MTM) Bus Protocol.* Test Technology Technical Committee of the IEEE Computer Society, Draft 0.9.4, March, 1992.

IEEE Std 1076 1987. *IEEE Standard VHDL Language Reference.* IEEE Standards Board, March 1988.

IEEE Std 1149.1 1990. *IEEE Standard Test Access Port and Boundary Scan Architecture.* IEEE Standards Board, May 1990.

Jarwala, N., and Yau, C. W. 1989. A New Framework for Analyzing Test Generation and Diagnosis Algorithms for Wiring Interconnects. *IEEE International Test Conference Proceedings,* 1989, pp. 63-70.

Jarwala, N., and Yau, C. W. 1991a. The Boundary-Scan Master: Architecture and Implementation. *European Test Conference Proceedings,* 1991, pp. 1-10.

Jarwala, N. and Yau, C. W. 1991b. Achieving Board-Level BIST Using the Boundary-Scan Master. *IEEE International Test Conference Proceedings,* 1991, pp. 649-658.

JEDEC 1986. Standard Manufacturer's Identification Code. Publication 106-A, July 1986.

Jong, F. and Ferreira, J. M. 1989. *Test Methodology for Full Boundary Scan Test.* Philips Report 65/89EN, August 1989.

Jong, F. 1990a. Boundary Scan Test Used at Board Level. *IEEE International Test Conference Proceedings,* 1990, pp. 235-242.

Jong, F. 1990b. *Test Methodology for Partial Boundary Scan Test.* Report R4.2, ESPRIT Project 2478 (Research into Boundary Scan Test Implementation), December 1990.

Jong, F., Matos, J. S. and Ferreira, J. M. 1991a. Boundary Scan Test, Test Methodology, and Fault Modeling. *Journal of Electronic Testing: Theory and Applications,* March 1991, Vol. 2, Nº 1, pp. 77-88.

Jong, F. and Heyden, F. 1991b. Testing the Integrity of the Boundary Scan Test Infrastructure. *IEEE International Test Conference Proceedings*, 1991, pp. 106-112.

Kakani, T. 1987. Testing of Surface Mount Technology Boards. *IEEE International Test Conference Proceedings*, 1987, pp. 614-620.

Kautz, W. H. 1974. Testing for Faults in Wiring Networks. *IEEE Transactions on Computers*, April 1974, Vol. C-23, Nº 4, pp. 358-363.

Kritter, S. and Rahaga, T. 1991. Boundary Scan and BIST Compatible IEEE 1149.1: VHDL & Autosynthesis of a SRAM Tester Macrocell and Chip. *European Test Conference Proceedings*, 1991, pp. 17-25.

Landis, D. 1989. A Self-Test System Architecture for Reconfigurable WSI. *IEEE International Test Conference Proceedings*, 1989, pp. 275-282.

Landis, D. 1992. A Test Methodology for Wafer Scale Systems. *IEEE Transactions on Computer-Aided Design*, January 1992, Vol. 11, Nº 1, pp. 76-82.

Laurent, D. 1985. An Example of Test Strategy for Computer Implemented with VLSI Circuits. *Proceedings of the IEEE International Conference on Circuits and Computers*, 1985, pp. 679-682.

LeBlanc, J. J. 1984. LOCST: A Built-In Self-Test Technique. *IEEE Design and Test of Computers*, Nov. 1984, pp. 45-52.

Lefebvre, M. F. 1990. Functional Test and Diagnosis: A Proposed JTAG Sample Mode Scan Tester. *IEEE International Test Conference Proceedings*, 1990, pp. 294-303.

Lefebvre, M. F. 1991. Test Generation: A Boundary Scan Implementation for Module Interconnect Testing. *IEEE International Test Conference Proceedings*, 1991, pp. 88-95.

Lien, J.-C. and Breuer, M. A. 1989. A Universal Test and Maintenance Controller for Modules and Boards. *IEEE Transactions on Industrial Electronics*, May 1989, Vol. 36, Nº 2, pp. 231-240.

Lien, J.-C. and Breuer, M. A. 1991a. Maximal Diagnosis for Wiring Networks. *IEEE International Test Conference Proceedings*, 1991, pp. 96-105.

Lien, J.-C. and Breuer, M. A. 1991b. An Optimal Scheduling Algorithm for Testing Interconnect Using Boundary Scan. *Journal of Electronic Testing: Theory and Applications*, March 1991, Vol. 2, Nº 1, pp. 117-130.

Lyon, J. A., Gladden, M., Hartung, E., Hoang, E., and Raghunathan, K. 1991. Testability Features of the 68HC16Z1. *IEEE International Test Conference Proceedings*, 1991, pp. 122-130.

Maierhofer, J. 1990. Hierarchical Self-Test Concept Based on the JTAG Standard. *IEEE International Test Conference Proceedings*, 1990, pp. 127-134.

Maunder, C. and Beenker, F. P. 1987. Boundary-Scan: A Framework for Structured Design-for-Test. *IEEE International Test Conference Proceedings*, 1987, pp. 714-723.

Maunder, C. and Tulloss, R. E. 1990. *The Test Access Port and Boundary Scan Architecture.* The IEEE Computer Society Press. ISBN 0-8186-9070-4.

Maunder, C. 1991. Languages to Support Boundary-Scan Test (panel summary). *IEEE International Test Conference Proceedings*, 1991, p. 1104.

Maunder, C. and Tulloss, R. E. 1991. An Introduction to the Boundary Scan Standard: ANSI/IEEE Std 1149.1. *Journal of Electronic Testing: Theory and Applications*. March 1991, Vol.2, Nº 1, pp. 27-42.

Miles, J., Bondt, R. and Daemen, L. 1991. A Test Economics Model & Cost-Benefit Analysis of Boundary Scan. *European Test conference Proceedings*, 1991, pp. 375-384.

Moore, T. J. 1991. A Workstation Environment for Boundary Scan Interconnect Testing. *IEEE International Test Conference Proceedings*, 1991, pp. 1096-1103.

Moran, L., Hillman, R., Burlison, P. and Gurda, T. 1990. The Waveform and Vector Exchange Specification (WAVES). *IEEE International Test Conference Proceedings*, 1990, pp. 978-987.

Mortensen, C. and Sejthen, N. 1991. *EBST Toolkit Reference Manual.* Version 1.2, October 1991.

Muris, M. 1989. *A Synchronous Boundary Scan Test Library.* Philips Technical Note 011/89EN, 1989.

Muris, M. 1990. Integrating Boundary Scan Test Into an ASIC Design Flow. *IEEE International Test Conference Proceedings*, 1990, pp. 472-477.

Parker, K. 1986. Testability: Barriers to Acceptance. *IEEE Design and Test of Computers*, Vol. 3, Nº 5, October 1986, pp. 11-15.

Parker, K. 1987. *Integrating Design and Test: Using CAE Tools for ATE Programming.* Computer Society Press of the IEEE. ISBN 0-8186-8788-6.

Parker, K. 1989. The Impact of Boundary Scan on Board Test. *IEEE Design and Test of Computers*, Aug. 1989, pp. 18-30.

Parker, K. and Oresjo, S. 1990. A Language for Describing Boundary Scan Devices. *IEEE International Test Conference Proceedings*, 1990, pp. 222-234.

Parker, K. and Oresjo, S. 1991. A Language for Describing Boundary-Scan Devices. *Journal of Electronic Testing: Theory and Applications.* March 1991, Vol.2, Nº 1, pp. 43-75.

Plassart, A. and Mortensen, C. 1990. *Test Interface Requirements and Choice of Format.* Report R5.2, ESPRIT Project 2478 (Research into Boundary Scan Test Implementation), April 1990.

Plassart, A. and Mortensen, C. 1991. *Specification of Test Interface Layer.* Deliverable D5.1, ESPRIT Project 2478 (Research into Boundary Scan Test Implementation), January 1991.

Plessey 1990. *Semi-Custom BIST Applications.* Publication Nº 2422, Plessey Semiconductors Ltd., January 1990.

Posse, K. E. 1991. A Design-for-Testability Architecture for Multichip Modules. *IEEE International Test Conference Proceedings*, 1991, pp. 113-121.

Pradhan, M. M., Tulloss, R. E., Bleeker, H. and Beenker, F. P. 1987. Developing a Standard for Boundary Scan Implementation. *IEEE International Conference on Computer Design: VLSI in Computers and Processors*, 1987, pp. 462-466.

Raghavachari, P. 1991. Circuit Pack BIST from System to Factory - The MCERT Chip. *IEEE International Test Conference Proceedings*, 1991, pp. 641-648.

Raymond, D. 1980. In-Circuit and Functional ATE: What's the Real Difference? *IEEE International Test Conference Proceedings*, 1980, pp. 315-317.

Riessen, R., Kerkhoff, H. G. and Kloppenburg, A. 1989. Design and Implementation of a Hierarchical Testable Architecture Using the Boundary Scan Standard. *European Test Conference Proceedings*, 1989, pp. 112-118.

Riessen, R. and Kerkhoff, H. G. 1991. Automatic Test-Specification Generation for Macro-Level BIST Based on the Boundary Scan Standard. *European Test Conference Proceedings*, 1991, pp. 447-453.

Rijk, R. H., Dekker, R. W. and Kerkhoff, H. G. 1991. Self-Test of a 254Kx4 bit Stand-Alone Static Ram. *European Test Conference Proceedings*, 1991, pp. 11-15.

Robinson, G. D. and Deshayes, J. G. 1990. Interconnect Testing of Boards with Partial Boundary Scan. *IEEE International Test Conference Proceedings*, 1990, pp. 572-581.

Robinson, J. P. and Cohn, M. 1981. Counting Sequences. *IEEE Transactions on Computers*, Vol. C-30, Nº 1, January 1981, pp. 17-23.

Rohde & Schwarz 1989. *Tester Compendium*. Info. Nº PD 0756.7565.21. 1989.

Russell, B. 1990. In-Circuit Testing. In Maunder, C. et al. *The Test Access Port and Boundary Scan Architecture*. The IEEE Computer Society Press. ISBN 0-8186-9070-4, pp. 123-126.

Schlotzhauer, E. O. and Balzer, R. J. 1987. Real-World Board Test Effectiveness: What Does It Mean When the Board Passes? *IEEE International Test Conference Proceedings*, 1987, pp. 792-797.

Schwederski, T., Büchner, T., Leenstra, J., Roos, G. and Spaanenburg, L. 1991. Built-In Pad Test with Boundary Scan. *European Test Conference Proceedings*, 1991, pp. 385-392.

Sebesta, W., Verhelst, B. and Wahl, M. G. 1990. Development of a New Standard for Test. *IEEE International Test Conference Proceedings*, 1990, pp. 988-993.

Sedmak, R. and Maunder, C. 1990. Benefits and Penalties of Boundary-Scan. In Maunder, C. et al. *The Test Access Port and Boundary Scan Architecture*. The IEEE Computer Society Press. ISBN 0-8186-9070-4, pp. 141-149.

Siemens 1990a. *Specification and Logical Design of Test Chips with BST.* Deliverable D2.4, ESPRIT Project 2478 (Research into Boundary Scan Test Implementation), August 1990.

Siemens 1990b. *TAP Interface Cell Design (Logic Level).* Report R1.2, ESPRIT Project 2478 (Research into Boundary Scan Test Implementation), January 1990.

Sterba, D., Halliday, A. and McClean, D. 1990. ATPG Issues for Board Designs Implementing Boundary Scan. *IEEE International Test Conference Proceedings,* 1990, pp. 243-251.

Sterba, D., Halliday, A. and McClean, D. 1991. ATPG and Diagnosis for Boards Implementing Boundary Scan. *Journal of Electronic Testing: Theory and Applications,* March 1991, Vol. 2, Nº 1, pp. 89-98.

Sweeney, J. 1990. Testability Implemented in the VAX 6000 Model 400. *IEEE International Test Conference Proceedings,* 1990, pp. 109-114.

TI ASSET 1990. *ASSET Scan-Based Diagnostics: Technical Overview.* Texas Instruments (Design Automation — Semiconductor Group), 1990.

TI BCT8244 1990. *SN74BCT8244: Scan Test Device with Octal Buffers.* Texas Instruments SCOPE Family of Testability Octals Data Sheet, 1990.

TI DBM 1990. *SN54ACT8994 / SN74ACT8994 Digital Bus Monitors.* Texas Instruments Data Sheet (Product Preview), 1990.

TI SCL162 1990. *SCOPE - An Extended JTAG Architecture for TI's Customers.* Texas Instruments Application Note SCL162, May 1990.

Tulloss, R. E. and Yau, C. W. 1989. BIST & Boundary-Scan for Board Level Test: Test Program Pseudocode. *European Test Conference Proceedings,* 1989, pp. 106-111.

Tulloss, R. E. and Yau, C. W. 1990a. A Test Program Pseudocode. In Maunder, C. et al. *The Test Access Port and Boundary Scan Architecture.* The IEEE Computer Society Press. ISBN 0-8186-9070-4, pp. 97-113.

Tulloss, R. E., Yau, C. W. and Whetsel, L. 1990b. Diagnosing Faults in the Serial Test Data Path. In Maunder, C. et al. *The Test Access Port and Boundary Scan Architecture.* The IEEE Computer Society Press. ISBN 0-8186-9070-4, pp. 115-121.

Turino, J. 1984. A Totally Universal Reset, Initialization (and) Nodal Observation Circuit. *IEEE International Test Conference Proceedings*, 1984, pp. 878-883.

Turino, J. 1989. Standard Testability Bus - An Applications Example (poster). *IEEE International Test Conference Proceedings*, 1989, p. 943.

Vefling, H. and Viktil, M. 1990. *Test of Mixed Analog/Digital Components by Boundary Scan Methods*. Report R8E.5, ESPRIT Project 2478 (Research into Boundary Scan Test Implementation), November 1990.

Verhelst, B. 1989. The Use of a Test Specification Format in Automatic Test Program Generation. *European Test Conference Proceedings*, 1989, pp. 362-368.

Vining, S. 1989. Tradeoff Decisions Made for a P1149.1 Controller Design. *IEEE International Test Conference Proceedings*, 1989, pp. 47-54.

Wagner, K. D. and Williams, T. W. 1991. Enhancing Board Functional Self-Test by Concurrent Sampling. *IEEE International Test Conference Proceedings*, 1991, pp. 633-640.

Wagner, P. 1987. Interconnect Testing with Boundary Scan. *IEEE International Test Conference Proceedings*, 1987, pp. 52-57.

Wang, L. T., Marhoefer, M. and McCluskey, E. J. 1989. A Self-Test and Self-Diagnosis Architecture for Boards Using Boundary Scans. *European Test Conference Proceedings*, 1989, pp. 119-126.

Whetsel, L. 1988. Standard Test Port and Cells Provide an ASIC Testability Toolkit. *Texas Instruments Technical Journal*; July-August 1988, pp. 35-43.

Whetsel, L. 1990. Event Qualification: A Gateway to At-Speed System Testing. *IEEE International Test Conference Proceedings*, 1990, pp. 135-141.

Whetsel, L. 1991. An IEEE 1149.1 Based Logic / Signature Analyzer in a Chip. *IEEE International Test Conference Proceedings*, 1991, pp. 869-878.

Wijngaarden, A. 1990. *The BST Datasheet. A Formal Description of the Boundary Scan Test logic of Components*. Philips CFT Automation, June 1990.

Williams, T. W. and Parker, K. P. 1983. Design for Testability - A Survey. *Proceedings of the IEEE*, Vol. 7, Nº 1, January 1983, pp. 98-112.

Winter, T., Muris, M. and Randgaard, H. 1990. *Fault Diagnosis Algorithms for PCBs with Boundary Scan Test*. Report R7.1a, ESPRIT Project 2478 (Research into Boundary Scan Test Implementation), October 1990.

Yau, C. W. and Jarwala, N. 1989. A Unified Theory for Designing Optimal Test Generation and Diagnosis Algorithms for Board Interconnects. *IEEE International Test Conference Proceedings*, 1989, pp. 71-77.

Yau, C. W. and Jarwala, N. 1990. The Boundary-Scan Master: Target Applications and Functional Requirements. *IEEE International Test Conference Proceedings*, 1990, pp. 311-315.

Anexo A

Folha de características do controlador de teste

CONTROLADOR DE TESTE PARA CARTAS DE CIRCUITO IMPRESSO COM BST

- A realização do teste requer apenas dois componentes (o controlador, e uma memória com o programa de teste).
- Oscilador interno permite frequências de funcionamento até 25 MHz.
- Disponível em encapsulamento LCC com 68 pinos.
- Suporta um conjunto de instruções que traduzem directamente as operações elementares necessárias para controlar a infraestrutura BST da CCI a testar.
- Suporta uma estratégia de teste hierárquica, em que o mesmo controlador pode ser usado a diferentes níveis.

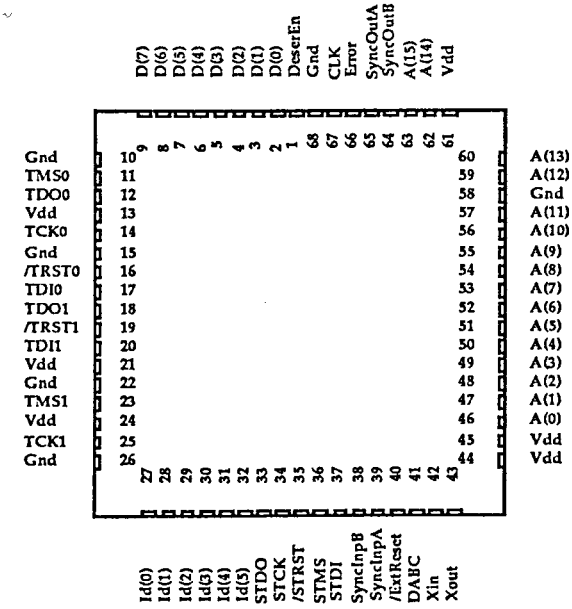


Fig.1: Configuração de pinos para o controlador de teste.

1. INTRODUÇÃO

Este componente constitui um controlador dedicado para o teste de cartas de circuito impresso (CCIs) com uma ou duas cadeias BST, e suporta um conjunto de instruções que traduzem directamente as operações elementares necessárias para uma exploração eficiente da infraestrutura BST disponível. O teste da CCI realiza-se por execução de um programa armazenado em memória exterior. Este componente apresenta-se em encapsulamento LCC com 68 pinos, agrupados funcionalmente da forma que se ilustra na figura 2.

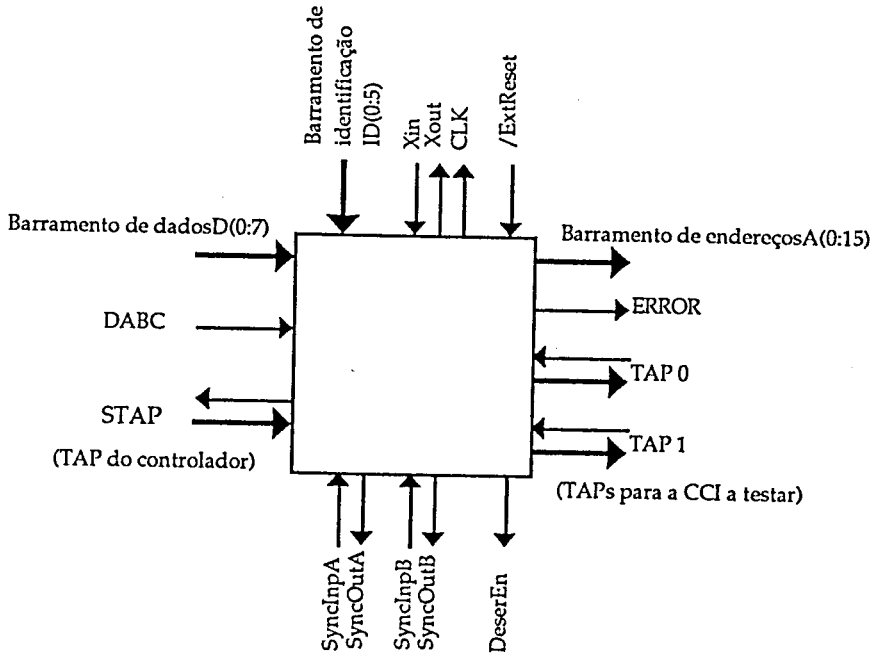


Fig.2: Agrupamento de pinos por afinidade funcional, para o controlador de teste.

O acesso à memória com o programa de teste é efectuado através de um barramento de endereços com 16 bits, e de um barramento de dados com 8 bits. Este acesso é unidireccional, sendo apenas efectuadas operações de leitura.

O próprio controlador é um componente BST, o que permite a implementação de uma estratégia de teste hierárquica, em que o mesmo componente pode ser empregue em diferentes níveis. A execução do programa de teste em qualquer CCI pode nestas circunstâncias ser comandada por um controlador residente ao nível do sistema ou subsistema. A identificação do tipo de faltas detectadas (grupos de vectores que

permitiram a detecção) fica memorizada num registo de estado com 2x8 *flags* (8 *flags* por cada cadeia BST da CCI a testar), acessível através da infraestrutura BST do próprio controlador. A disjunção deste conjunto de *flags* está presente na saída *Error* (qualquer *flag* de erro actuada força a saída *Error* a 1).

A infraestrutura BST do próprio controlador pode ser acedida através do conjunto de sinais referidos por STAP na figura 2 (STDI, STDO, STMS, STCK, /STRST), destinando-se os conjuntos designados por TAP0 e TAP1 a permitir o controlo da infraestrutura BST da CCI a testar. A infraestrutura BST do controlador permite ainda a leitura de um conjunto de 6 pinos destinados a codificar um endereço para a CCI (barramento de identificação, Id(0:5)), o que permite verificar o posicionamento das CCIs presentes.

O quadro 1 apresenta a caracterização funcional de pinos para este componente. Todos os pinos não representados pertencem à alimentação ou massa.

Designação	Entrada	Saída	Funcionalidade
DeserEn		•	Indica para o exterior os ciclos de relógio de teste (TCK0, TCK1) em que têm lugar operações de comparação sobre os valores deslocados para o exterior da(s) cadeia(s) BST da CCI a testar.
D(0:7)	•		Barramento de dados.
TMS0, TDO0, TCK0, /TRST0		•	TAP0, destinado ao controlo da cadeia BST da CCI a testar.
TDI0	•		
TMS1, TDO1, TCK1, /TRST1		•	TAP1, destinado ao controlo de uma segunda cadeia BST na CCI a testar.
TDI1	•		
Id(0:5)	•		Barramento de identificação. A leitura do endereço contido nestes 6 bits pode ser efectuada através do registo BST do controlador.
STMS, STDI, STCK, /STRST	•		TAP do controlador (STAP), para acesso à sua própria infraestrutura BST.
STDO		•	
SyncInpA, SyncInpB	•		Entradas de sincronismo.
SyncOutA, SyncOutB		•	Saídas de sincronismo.

Quadro 1: Caracterização funcional dos pinos do controlador de teste (continua).

Designação	Entrada	Saída	Funcionalidade
/ExtReset	•		Inicialização do controlador. Em situação de *reset*, os dois TAPs, e o barramento de endereços, passam ao estado de alta impedância.
DABC	•		Permite o acesso directo às cadeias BST da CCI a testar, colocando as saídas dos respectivos TAPs em estado de alta impedância.
X_{in}	•		Relógio do controlador. Um cristal externo pode ser ligado entre X_{in} e X_{out}, e o sinal de relógio gerado internamente está disponível no pino CLK.
X_{out}, CLK		•	
A(0:15)		•	Barramento de endereços
Error		•	Apresenta a disjunção das 16 *flags* de erro internas (Error em 1 significa a anterior detecção de uma ou mais faltas).

Quadro 1: Caracterização funcional dos pinos do controlador de teste (conclusão).

2. DESCRIÇÃO FUNCIONAL

A arquitectura deste componente é composta pelos blocos principais ilustrados na figura 3, e cujo núcleo consiste num processador dedicado para o controlo da infraestrutura BST de uma CCI. Os blocos designados por *ClockMux* e *ResetGen* permitem que o controlo (inicialização, sinal de relógio) deste núcleo tenha lugar localmente (na CCI onde o componente está colocado), ou através da sua infraestrutura BST (a partir de um nível hierárquico superior).

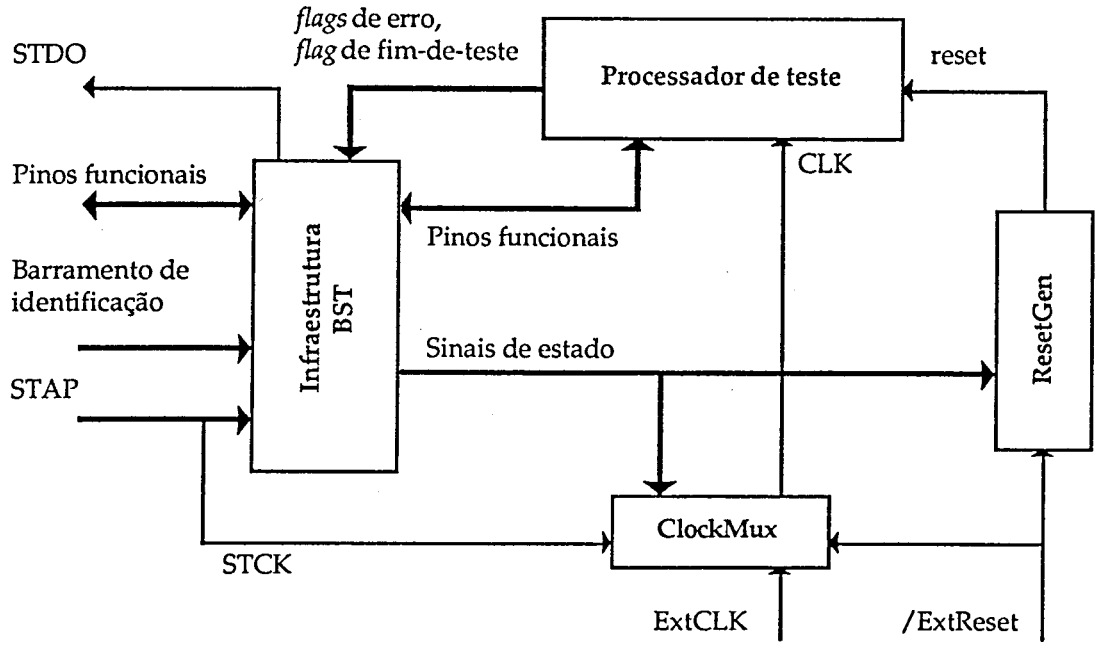


Fig.3: Arquitectura geral do controlador de teste.

2.1. O PROCESSADOR DE TESTE

O processador de teste que constitui o núcleo principal deste componente está representado na figura 4. O programa a executar está armazenado numa memória exterior, cujo acesso é comandado pelo apontador de programa. Cada código de operação extraído da memória é fornecido à unidade de descodificação e controlo, que coordena a execução da

instrução correspondente.

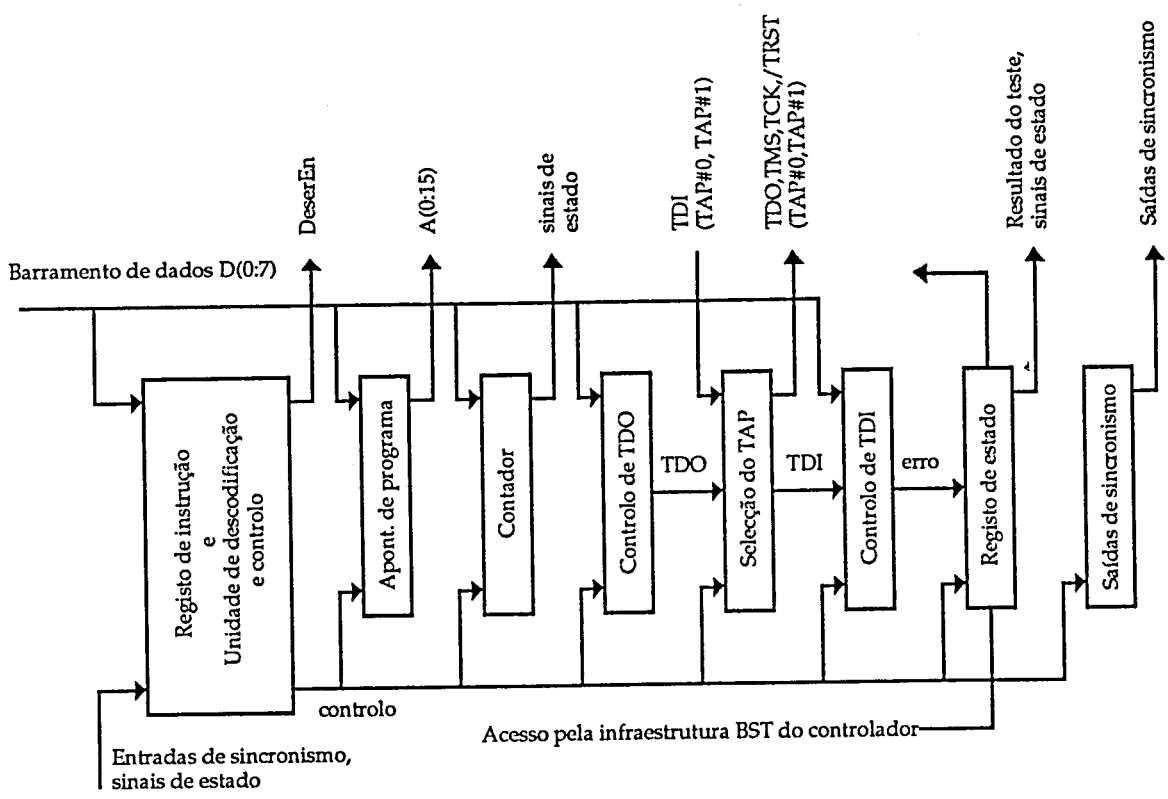


Fig.4: O processador de teste.

Um contador com 16 bits permite a especificação do número de impulsos a aplicar no relógio de teste (TCK) da cadeia BST seleccionada, seja para permitir o deslocamento de estímulos (e respostas) através do percurso série, ou para a execução de funções de auto-teste disponíveis em componentes presentes.

Os blocos *Controlo de TDI* e *Controlo de TDO* responsabilizam-se pela interacção com o percurso série, na cadeia BST seleccionada pelo bloco *Selecção do TAP*. A comparação entre os valores deslocados para o exterior desta cadeia BST, e os correspondentes valores esperados, é feita no bloco *Controlo de TDI*. De acordo com o valor da respectiva máscara, o resultado desta comparação pode ser memorizado na *flag* de erro seleccionada, ou desprezado.

O *Registo de estado* contem dois grupos de 8 *flags* de erro, e uma *flag* de fim de teste. Cada grupo de 8 *flags* de erro está atribuído a uma cadeia BST, tendo lugar automaticamente a comutação do grupo activo, de cada vez que se comuta o TAP seleccionado (a cadeia BST seleccionada). Esta comutação não tem porém efeito sobre a identificação da *flag* de erro seleccionada (0 a 7), que é controlada pela instrução

correspondente.

A saída DeserEn estará activa (em 1) sempre que tiver lugar um impulso de relógio em que se efectue a comparação entre o valor deslocado para o exterior da cadeia BST seleccionada, e o correspondente valor esperado (independentemente do valor da respectiva máscara). Este sinal pode ser empregue para controlar um circuito exterior que efectue a deserialização e armazenamento das respostas deslocadas para o exterior da cadeia BST, nas aplicações em que esta operação tenha interesse.

A sincronização com recursos de teste exteriores pode ser efectuada através dos pinos *SyncInp* (para a unidade de descodificação e controlo) e *SyncOut* (do bloco *saídas de sincronismo*). Estes pinos são directamente acessíveis a partir de instruções que possibilitam o estabelecimento de um protocolo simples para o sincronismo com equipamentos exteriores.

2.2. A INFRAESTRUTURA BST DO CONTROLADOR DE TESTE

A infraestrutura BST referida na figura 3 está ilustrada na figura 5, e permite a implementação de uma estratégia hierárquica, em que a execução do programa de teste é controlada a partir de um controlador residente ao nível do sistema, ou subsistema.

Esta infraestrutura BST é acessível através do STAP, que integra os sinais STDI, STDO, STMS, STCK, e /STRST. O controlador do STAP coordena o respectivo funcionamento, de acordo com o diagrama de estados ilustrado na figura 6.

A inicialização desta infraestrutura é provocada internamente sempre que se liga a alimentação, sem que seja necessária a presença de uma malha RC exterior, no pino /STRST. Este pino está no entanto disponível para permitir a inicialização noutras circunstâncias, tais como quando vários controladores forem usados numa configuração hierárquica. O sinal de inicialização gerado quando se liga a alimentação é também aplicado nas saídas /TRST0 e /TRST1.

O acesso exterior às cadeias BST da CCI é possível por colocação das saídas correspondentes do controlador em estado de alta impedância (TDO, TMS, TCK, e /TRST, para cada TAP), através da aplicação de um 1 na entrada DABC.

Um registo de instrução com 2 bits permite as quatro instruções BST descritas no quadro 2. A execução do programa de teste pode ser controlada através da infraestrutura BST do próprio componente, o que terá lugar quando o controlador do STAP se encontrar no estado *Run Test / Idle*, tendo previamente sido carregado o registo de instrução com o código de *Run Board Test*. O pino STCK deverá nestas circunstâncias proporcionar o sinal de relógio necessário para a execução do programa de teste.

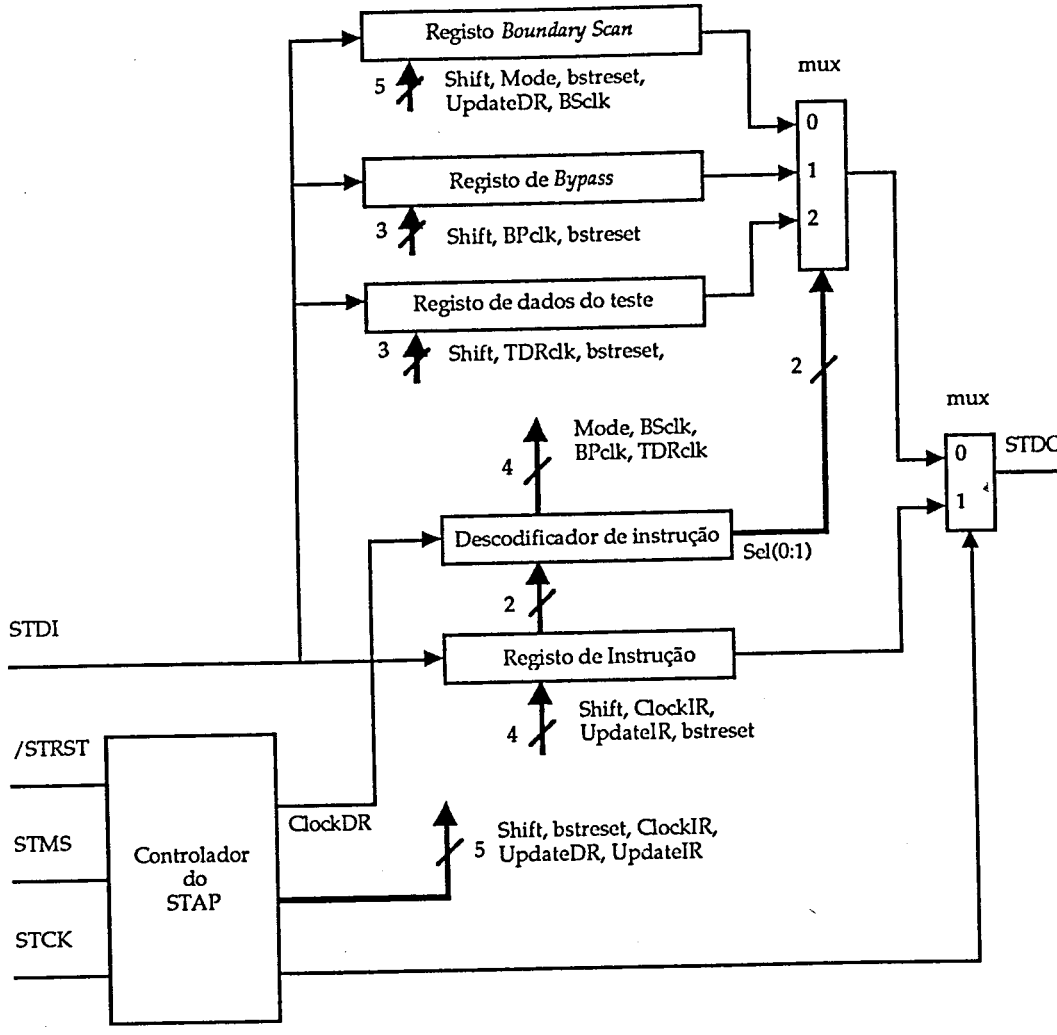


Fig.5: Infraestrutura BST do controlador de teste.

Bit 1	Bit 0†	Instrução
0	0	*Extest*
0	1	*Run Board Test*
1	0	*Sample / Preload*
1	1	*Bypass*

Quadro 2: Instruções suportadas pela infraestrutura BST do controlador de teste.

† O bit 0 é o bit menos significativo (mais próximo de STDO).

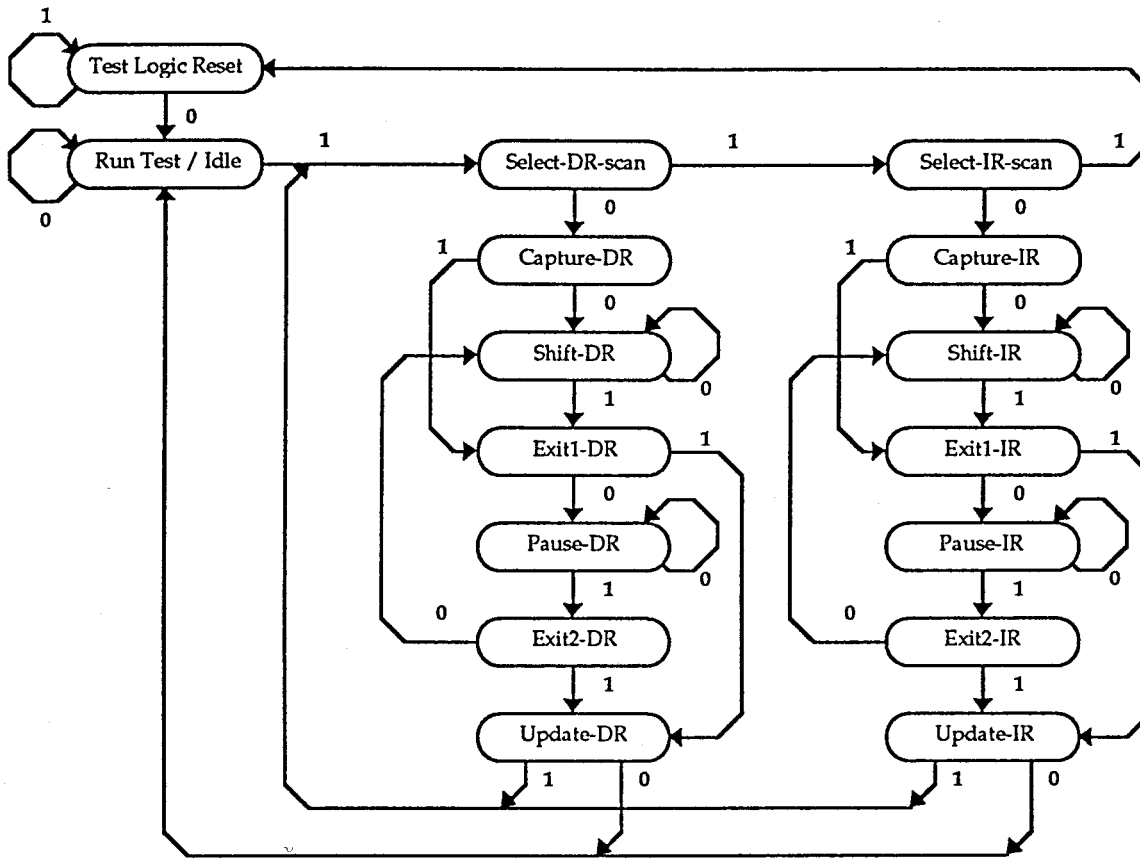


Fig.6: Diagrama de estados para o controlador do STAP.

A execução do programa de teste pode ser interrompida se se suspender a aplicação de impulsos em STCK, ou por deslocamento do controlador do STAP para fora do estado *Run Test / Idle*. A instrução *Run Board Test* selecciona o registo de dados do teste (figura 5), que captura o conteúdo do registo de estado do processador (figura 4), pelo que a leitura do resultado parcial ou total da execução do teste pode ser feita por colocação do controlador do STAP no estado *Shift-DR*. A posterior colocação no estado *Run Test / Idle* permite que se retome a execução do programa de teste, caso esta ainda não tenha sido concluída. O registo de dados seleccionado pela instrução *Run Board Test* é composto por 17 bits, como se descreve no quadro 3.

Bit	Significado
0...7	*flag* de erro(0:7), TAP0
8...15	*flag* de erro(0:7), TAP1
16	*flag* de fim de teste

Quadro 3: Composição do registo de dados do teste, seleccionado pela instrução *Run Board Test*.

Cada CCI que contenha um controlador de teste pode ter atribuído um endereço específico, codificado através do barramento de identificação disponível. Este barramento é composto por 6 bits, e pode ser lido através do registo BST, cuja composição está por sua vez descrita no quadro 4.

Célula	Pino	Comentários
0	-	Saída interna do oscilador a cristal (ExtCLK)
1	Error	Indica a ocorrência de uma ou mais faltas
2	ExtReset	Inicializa o processador de teste
3...10	D(0:7)	Barramento de dados
11...26	A(0:15)	Barramento de endereços
27, 28	SyncInpA, SyncOutA	Entrada e saída de sincronismo, canal A
29, 30	SyncInpB, SyncOutB	Entrada e saída de sincronismo, canal B
31...35	TDI0, TDO0, TCK0, TMS0, /TRST0	Sinais que controlam a cadeia BST ligada ao TAP0
36...40	TDI1, TDO1, TCK1, TMS1, /TRST1	Sinais que controlam a cadeia BST ligada ao TAP1
41	CLK	Saída com o relógio do sistema
42	DABC	Permite o acesso exterior às cadeias BST da CCI
43	DeserEn	Activa um deserializador externo
44...49	Id(0:5)	Barramento de identificação

Quadro 4: Composição do registo BST.

2.3. CONJUNTO DE INSTRUÇÕES DO PROCESSADOR DE TESTE

O conjunto de instruções suportadas pelo processador permite a implementação directa das operações elementares necessárias para o teste de uma CCI com BST, e está descrito nos quadros 5 a 7 (N representa o conteúdo do contador interno).

Qualquer código de operação ilegal é tratado como um código de operação nula (NOP). O único efeito deste código consiste em incrementar o apontador de programa.

INSTRUÇÕES PARA O CONTROLO DAS CADEIAS BST DA CCI A TESTAR	
Mnemónica	Descrição
TRST	Aplica um impulso a 0 na linha /TRST da cadeia BST seleccionada.
TMS0, TMS1	Aplica um impulso no relógio de teste (TCK) da cadeia BST seleccionada, com a linha TMS no valor pretendido.
NSHF	Efectua o deslocamento para a cadeia BST seleccionada, sem comparação, de um vector com N bits.
NSHFCP	Efectua o deslocamento para a cadeia BST seleccionada, com comparação, de um vector com N bits.
NTCK	Aplica N impulsos no relógio de teste (TCK) da cadeia BST seleccionada, mantendo a linha TMS em 0.
SELTAP0, SELTAP1	Selecciona qual a cadeia BST a controlar.

Quadro 5: Instruções para o controlo das cadeias BST da CCI a testar.

INSTRUÇÕES PARA O SINCRONISMO COM RECURSOS DE TESTE EXTERIORES	
Mnemónica	Descrição
SSA0, SSA1, SSB0, SSB1	Coloca na saída de sincronismo o valor pretendido.
WSA0, WSA1, WSB0, WSB1	Espera até que a entrada de sincronismo se encontre no valor pretendido.

Quadro 6: Instruções para o sincronismo com recursos de teste exteriores.

INSTRUÇÕES PARA O CONTROLO DOS RECURSOS INTERNOS	
Mnemónica	Descrição
LD CNT, N	Carrega o contador interno com o número de impulsos pretendidos no relógio de teste.
SERFLG0,..., SERFLG7	Selecciona a *flag* de erro a ser actuada pela posterior detecção de uma falta.
JPE Endereço, JPNE Endereço	Implementa um salto condicional, de acordo com o estado da *flag* de erro seleccionada.
HALT	Conclui a execução do programa de teste.

Quadro 7: Instruções para o controlo dos recursos internos.

3. CARACTERIZAÇÃO DO CONJUNTO DE INSTRUÇÕES

A caracterização individual apresentada para cada instrução inclui uma descrição sucinta do seu efeito, e é em alguns casos complementada por um diagrama temporal que ilustra a respectiva execução. Todos os códigos de operação são apresentados em formato hexadecimal.

TRST

Código de operação: 11.

Número de posições de memória: 1.

Número de ciclos de relógio: 3.

Esta instrução permite a inicialização da lógica BST em todos os componentes com um pino /TRST ligado a esta saída, na cadeia BST seleccionada. A inicialização de toda a infraestrutura BST só será possível por este processo se todos os componentes BST dispuserem de um pino /TRST ligado à correspondente saída do controlador de teste. A linha TMS é mantida em 1 durante a transição ascendente da linha /TRST, pelo que o diagrama temporal correspondente à execução desta instrução é o que se ilustra na figura 7.

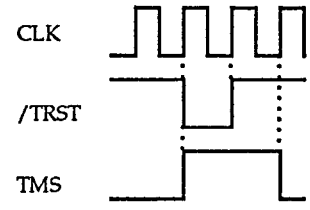


Fig.7: Diagrama temporal para a execução da instrução TRST.

Um impulso em 0 é aplicado nas linhas /TRST dos dois TAPs comandados pelo controlador sempre que a alimentação é ligada, com o objectivo de provocar a inicialização da lógica BST (*power-up reset*), nos componentes em que esta não ocorra automaticamente.

TMS0, TMS1

Códigos de operação: 00, 01.

Número de posições de memória: 1.

Número de ciclos de relógio: 2.

Esta instrução permite a aplicação de um impulso no relógio de teste (TCK), enquanto a linha TMS é mantida em **0** (TMS0), ou em **1** (TMS1). O diagrama temporal que caracteriza a execução da instrução TMS1 está ilustrado na figura 8.

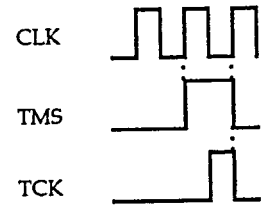


Fig.8: Diagrama temporal para a execução da instrução TMS1.

NSHF

Código de operação: 04.

Número de posições de memória: $1 + \lceil (CNT)/8 \rceil$ †

Número de ciclos de relógio: 2, se (CNT)=0;

5, se (CNT)=1;

$3 + (CNT) + \lceil (CNT)/8 \rceil$, nos restantes casos.

A execução desta instrução fará com que a sequência de N bits que se segue ao respectivo código de instrução seja deslocada para o interior da cadeia BST seleccionada. N representa o conteúdo do contador interno (16 bits), pelo que esta instrução deverá ser precedida por uma outra que carregue este contador com o valor pretendido (LD CNT, N).

A linha TMS é mantida em 0 durante os primeiros (N-1) impulsos no relógio de teste (TCK), de modo a que o controlador do TAP, na lógica BST de cada componente, se conserve no estado *Shift-DR* (*Shift-IR*). O último impulso aplicado no relógio de teste tem a linha TMS em 1, de modo a que ocorra a transição para o estado *Exit1-DR* (*Exit1-IR*). A transição ascendente na linha TMS tem lugar com a transição descendente do (N-1)º impulso no relógio de teste. O controlador do TAP, na lógica BST de cada componente, deve encontrar-se já no estado *Shift-DR* (*Shift-IR*), anteriormente à execução desta instrução.

As transições na linha TDO (actualização do valor a deslocar para a cadeia BST) ocorrem com as transições descendentes no relógio de teste, tal como se ilustra na figura 9, que apresenta o diagrama temporal para a execução da instrução NSHF \$55,\$CA†† (assume-se que o contador tinha sido anteriormente carregado através da instrução LD CNT, 10).

A execução da operação de deslocamento, e a leitura de novos valores a deslocar, é feita de forma concorrente, o que permite ter um atraso de apenas um único impulso no relógio de teste (TCK), por cada 8 bits a deslocar. Devido a este facto, são necessários 9 ciclos de relógio (CLK) por cada 8 impulsos no relógio de teste (TCK), tal como se ilustra na figura 9 (aparte este atraso, a frequência do relógio de teste é igual à frequência de relógio

---

† (CNT) representa o conteúdo do contador interno, e $\lceil x \rceil$ o menor inteiro não inferior a x (tecto de x).

†† O símbolo \$ é aqui usado para indicar a representação hexadecimal.

do processador).

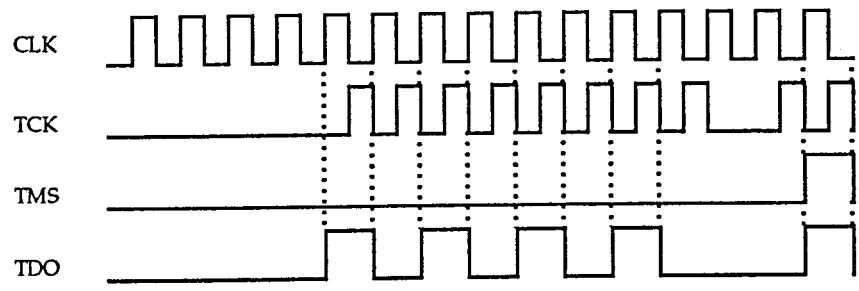


Fig.9: Diagrama temporal para a execução da instrução NSHF $55,$CA (N=10).

Os dados a deslocar para o interior da cadeia BST deverão estar armazenados em memória da forma que se ilustra na figura 10. Estes dados serão deslocados da esquerda para a direita (do bit mais significativo para o menos significativo), pelo que o primeiro bit a ser deslocado será o bit menos significativo (D0) da palavra que se segue ao código desta instrução.

Endereço:	D7	D6	D5	D4	D3	D2	D1	D0	
(PC)	Código de NSHF								
(PC)+1	0	1	0	1	0	1	0	1	55
(PC)+2	1	1	0	0	1	0	1	0	CA
...									

Fig.10: Armazenamento dos dados na instrução NSHF $55,$CA.

NSHFCP

Código de operação: 05.

Número de posições de memória: $1 + 3 \cdot \lceil (CNT)/8 \rceil$

Número de ciclos de relógio: 2, se (CNT)=0;

9, se (CNT)=1;

$3 + (CNT) + 3 \cdot \lceil (CNT)/8 \rceil$, nos restantes casos.

A execução desta instrução fará com que uma sequência de N bits, posterior ao respectivo código de instrução, seja deslocada para o interior da cadeia BST seleccionada. O armazenamento em memória desta sequência deve no entanto ser alternado com o de outras duas sequências, palavra a palavra, destinadas respectivamente à indicação dos valores esperados, e das máscaras de comparação a utilizar. N representa o conteúdo do contador interno (16 bits), pelo que esta instrução deverá ser precedida por uma outra que carregue este contador com o valor pretendido (LD CNT, N). A primeira comparação de um bit que difira do seu valor esperado, e cuja máscara valide a comparação, actuará a *flag* de erro seleccionada, e fará com que a saída do processador que indica o resultado do teste (*Error*) passe ao estado activo. A instrução NSHFCP terminará a sua execução em qualquer dos casos, mas o estado da *flag* de erro seleccionada poderá ser posteriormente testado por uma operação de salto condicional. Os impulsos no relógio de teste (TCK), aplicados durante a execução da instrução NSHFCP, serão acompanhados pela aplicação de um 1 na saída DeserEn, com o objectivo de indicar para o exterior que decorre uma operação de deslocamento com comparação.

A linha TMS é mantida em 0 durante os primeiros (N-1) impulsos no relógio de teste, de modo a que o controlador do TAP, na lógica BST de cada componente, se conserve no estado *Shift-DR* (*Shift-IR*). O último impulso aplicado no relógio de teste tem a linha TMS em 1, de modo a que ocorra a transição para o estado *Exit1-DR* (*Exit1-IR*). A transição ascendente na linha TMS tem lugar com a transição descendente do $(N-1)^{º}$ impulso no relógio de teste. O controlador do TAP, na lógica BST de cada componente, deve encontrar-se já no estado *Shift-DR* (*Shift-IR*), anteriormente à execução desta instrução.

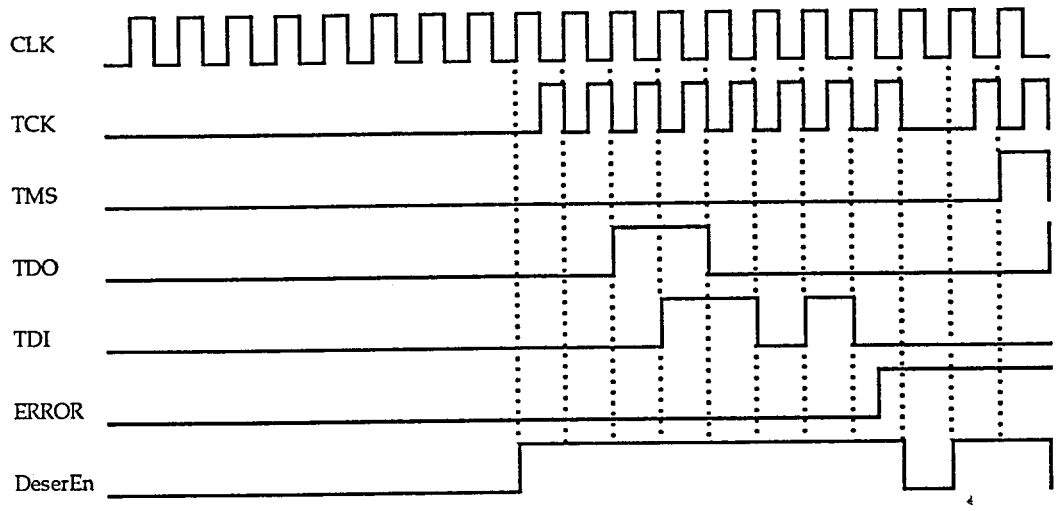


Fig.11: Diagrama temporal na execução da instrução NSHFCP $0C,$C0,$E7,$0C,$80,$03 (N=10).

As transições na linha TDO (actualização do valor a deslocar para a cadeia BST) ocorrem com as transições descendentes no relógio de teste, tal como se ilustra na figura 11, que apresenta o diagrama temporal para a execução da instrução NSHFCP $0C,$C0,$E7,$0C,$80,$03 (assume-se que o contador interno tinha sido anteriormente carregado através da instrução LD CNT, 10).

Também na instrução NSHFCP, e a exemplo do que sucedia já na instrução NSHF, a execução concorrente da operação de deslocamento, e a leitura dos novos valores a extrair da memória, permite ter apenas um atraso de um impulso no relógio de teste, por cada 8 bits a deslocar. Isto significa que são necessários 9 ciclos de relógio (CLK) por cada 8 impulsos no relógio de teste (TCK), tal como se verifica na figura 11 (aparte este atraso, a frequência do relógio de teste é igual à frequência de relógio do processador).

Endereço:	D7	D6	D5	D4	D3	D2	D1	D0		
(PC)	Código de NSHFCP									
(PC)+1	0	0	0	0	1	1	0	0	0C	(a deslocar)
(PC)+2	1	1	0	0	0	0	0	0	C0	(valores esperados)
(PC)+3	1	1	1	0	0	1	1	1	E7	(máscara de comparação)
(PC)+4	0	0	0	0	1	1	0	0	0C	(a deslocar)
(PC)+5	1	0	0	0	0	0	0	0	80	(valores esperados)
(PC)+6	0	0	0	0	0	0	1	1	03	(máscara de comparação)

Fig.12: Armazenamento dos dados na instrução NSHFCP $0C,$C0,$E7,$0C,$80,$03 (N=10).

Os dados correspondentes a esta instrução deverão estar armazenados em memória de modo a que surja alternadamente uma palavra de cada um dos três tipos (valores a deslocar, valores esperados, e máscaras de comparação), tal como se ilustra na figura 12. O deslocamento destes valores procede da esquerda para a direita (do bit mais significativo para o menos significativo), pelo que o primeiro bit a ser deslocado será o bit menos significativo (D0) da palavra que se segue ao código desta instrução. A primeira palavra que se segue ao código da instrução pertence ao conjunto dos valores a deslocar, a segunda ao conjunto dos valores esperados, e a terceira às máscaras de comparação, mantendo-se daqui por diante esta mesma sequência.

NTCK

Código de operação: 03.

Número de posições de memória: 1.

Número de ciclos de relógio: 2 + (CNT).

Esta instrução destina-se a permitir a execução das funções de auto-teste em componentes BST que disponham desta possibilidade, e terá por resultado a aplicação de N impulsos no relógio de teste (TCK), enquanto a linha TMS é mantida em **0** (de modo a que o controlador do TAP, na lógica BST de cada componente, se mantenha no estado *Run Test / Idle*). N representa o conteúdo do contador interno (de 16 bits), pelo que esta instrução deverá ser precedida por uma outra que carregue este contador com o valor pretendido (LD CNT, N). Repare-se que o controlador do TAP, em cada componente com BST, se deverá encontrar já no estado *Run Test / Idle*. A frequência dos impulsos gerados no relógio de teste (TCK) será a mesma do relógio do processador (CLK).

SELTAP0, SELTAP1

Códigos de operação: 1A, 1B.

Número de posições de memória: 1.

Número de ciclos de relógio: 2.

Estas instruções permitem seleccionar qual dos dois TAPs será controlado pelas instruções seguintes, e destinam-se a permitir o controlo de CCIs com duas cadeias BST. O TAP seleccionado será apenas modificado pela reinicialização do processador, ou por outra instrução SELTAP posterior.

SSA0, SSA1, SSB0, SSB1

Códigos de operação: 12, 13, 16, 17.
Número de posições de memória: 1.
Número de ciclos de relógio: 2.

Estas instruções destinam-se a actuar sobre as saídas de sincronismo, e aplicam o valor especificado (0, 1) à saída de sincronismo indicada (A, B). O valor aplicado na saída de sincronismo poderá apenas ser modificado por uma reinicialização do processador, ou por outra instrução SS posterior. Quando usadas em conjunto com as instruções WS (a descrever a seguir), permitem implementar um protocolo de sincronismo assíncrono com equipamentos exteriores.

WSA0, WSA1, WSB0, WSB1

Códigos de operação: 14, 15, 18, 19.
Número de posições de memória: 1.
Número de ciclos de relógio: Indefinido. O processador monitorizará o valor da entrada especificada, até que esta assuma o valor indicado.

Estas instruções retêm o processador no mesmo estado por um número indefinido de ciclos de relógio, até que a entrada de sincronismo (A, B) se encontre no valor especificado (0, 1). Quando usadas em conjunto com as instruções SS, permitem implementar um protocolo de sincronismo assíncrono com equipamentos exteriores.

LD CNT, N

Código de operação: 02.

Número de posições de memória: 3.

Número de ciclos de relógio: 6.

Esta instrução transfere o conteúdo das segunda e terceira palavras de memória, após o código de instrução, para o contador interno. A segunda palavra deverá conter os 8 bits mais significativos do valor a carregar, e a seguinte os 8 bits menos significativos.

SERFLG0, ..., SERFLG7

Códigos de operação: 08, ..., 0F.

Número de posições de memória: 1.

Número de ciclos de relógio: 2.

Este conjunto de 8 instruções permite seleccionar uma de entre as 8 *flags* de erro disponíveis para cada TAP. A ocorrência de um erro, na execução posterior de uma instrução NSHFCP, actuará sobre a *flag* de erro seleccionada. A selecção de diferentes *flags* de erro, a preceder instruções NSHFCP para diferentes tipos de vectores, permite discriminar entre diferentes tipos de faltas. A execução de uma instrução que selecciona a cadeia BST a controlar tem automaticamente por efeito que passa a estar seleccionado o respectivo grupo de *flags* de erro, embora se mantenha (dentro do novo grupo) a indicação da *flag* de erro seleccionada (0 a 7). A inicialização do processador selecciona a *flag* 0. O quadro 8 apresenta os códigos de operação atribuídos a cada uma das instruções SERFLG.

SERFLG0	08	SERFLG4	0C
SERFLG1	09	SERFLG5	0D
SERFLG2	0A	SERFLG6	0E
SERFLG3	0B	SERFLG7	0F

Quadro 8: Códigos de operação para as instruções SERFLG.

JPE Endereço

Código de operação: 06.

Número de posições de memória: 3.

Número de ciclos de relógio: 4, se não ocorrer o salto, ou 5, se ocorrer.

Esta instrução provoca um salto para o endereço especificado, se a *flag* de erro seleccionada tiver sido actuada anteriormente. Caso contrário, o próximo código de instrução será lido na primeira posição que se segue ao operando desta instrução.

JPNE Endereço

Código de operação: 07.

Número de posições de memória: 3.

Número de ciclos de relógio: 4, se não ocorrer o salto, ou 5, se ocorrer.

Esta instrução provoca um salto para o endereço especificado, se a *flag* de erro seleccionada não tiver sido actuada anteriormente. Caso contrário, o próximo código de instrução será lido na primeira posição que se segue ao operando desta instrução.

HALT

Código de operação: 10.

Número de posições de memória: 1.

Número de ciclos de relógio: ∞.

Esta instrução destina-se a terminar um programa de teste. A sua execução actua a *flag* de fim-de-teste, e faz com que o processador não incremente mais o apontador de programa.

4. PARÂMETROS DE FUNCIONAMENTO

4.1. VALORES MÁXIMOS ABSOLUTOS

Símbolo	Descrição	Min.	Max.	Unidades
V_{DD}	Tensão de alimentação	-0,5	7,0	V
V_{in}	Tensão de entrada	-0,5	V_{DD} + 0,5	V
T_{St}	Temperatura de armazenamento	-65	+150	°C
∂_{cc}	Duração máxima para um curto circuito entre saídas	-	5	s

Quadro 9: Valores máximos absolutos para o controlador de teste.

4.2. CARACTERÍSTICAS ELÉCTRICAS

As entradas */ExtReset* (pino 40) e X_{in} (pino 42) apresentam características especiais, e dizem respeito respectivamente a uma entrada de *Schmitt* para a malha RC de inicialização exterior (para o processador de teste), e à entrada do oscilador a cristal. Todas as restantes entradas dispõem de uma resistência de *pull-up* de 9 KΩ, e apresentam as características eléctricas descritas no quadro 10.

Parâmetro	Valor típico	Unidades
$\|I_{IL}\|$	10	µA
I_{IH}	10	µA
V_{IL}	$0,3 \cdot V_{DD}$	V
V_{IH}	$0,7 \cdot V_{DD}$	V

Quadro 10: Características eléctricas do controlador de teste (1).

As características eléctricas relativamente às saídas *DeserEn*, *SyncOutA*, *SyncOutB*,

Error, *TDO0*, *TDO1*, *STDO*, e *A(0:15)*, estão descritas no quadro 11.

Parâmetro	Min.	Típ.	Max.	Unidades
I_{OL} ($V_O \leq 0,5$ V)	-	4	10	mA
$\lvert I_{OH} \rvert$ ($V_O \geq V_{DD} - 0,5$ V)	-	4	10	mA
V_{OH} ($\lvert I_O \rvert < 1$ µA)	$V_{DD} - 0,05$	-	-	V
V_{OL} ($I_O < 1$ µA)	-	-	0,05	V

Quadro 11: Características eléctricas do controlador de teste (2).

As características eléctricas relativamente às saídas */TRST0*, */TRST1*, e *CLK*, estão descritas no quadro 12.

Parâmetro	Min.	Típ.	Max.	Unidades
I_{OL} ($V_O \leq 0,5$ V)	-	16	40	mA
$\lvert I_{OH} \rvert$ ($V_O \geq V_{DD} - 0,5$ V)	-	16	40	mA
V_{OH} ($\lvert I_O \rvert < 1$ µA)	$V_{DD} - 0,05$	-	-	V
V_{OL} ($I_O < 1$ µA)	-	-	0,05	V

Quadro 12: Características eléctricas do controlador de teste (3).

As características eléctricas relativamente às saídas TMS0, TMS1, TCK0, e TCK1, estão descritas no quadro 13.

Parâmetro	Min.	Típ.	Max.	Unidades
I_{OL} ($V_O \leq 0,5$ V)	-	24	60	mA
$\lvert I_{OH} \rvert$ ($V_O \geq V_{DD} - 0,5$ V)	-	24	60	mA
V_{OH} ($\lvert I_O \rvert < 1$ µA)	$V_{DD} - 0,05$	-	-	V
V_{OL} ($I_O < 1$ µA)	-	-	0,05	V

Quadro 13: Características eléctricas do controlador de teste (4).

A saída X_{out} tem características especiais, e corresponde à saída do oscilador a cristal. Esta saída não deverá ser usada para alimentar quaisquer entradas.

4.3. GERAÇÃO DO SINAL DE RELÓGIO

O oscilador a cristal gera internamente um sinal de relógio a partir de um cristal colocado entre os pinos X_{in} (pino 42) e X_{out} (pino 43), tal como se ilustra na figura 13[†] . O sinal presente na saída X_{out} não deve ser usado para alimentar quaisquer entradas, devendo para este fim ser usado o sinal presente na saída *CLK* (pino 67).

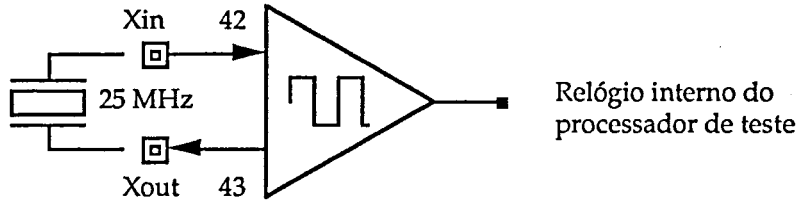


Fig.13: Geração do sinal de relógio por recurso ao oscilador interno.

O circuito ilustrado na figura 13 usa um cristal com uma frequência fundamental de 25 MHz. As características do oscilador interno permitem no entanto a utilização de cristais com uma frequência entre 5 MHz e 30 MHz. O processador de teste poderá em alternativa receber um sinal de relógio gerado exteriormente, aplicado na entrada X_{in}, e cuja frequência poderá neste caso estender-se de 0 até 25 MHz.

---

† ECPD15 Library Data Book, SOLO1400 Release 3.0, January 1990, pp. 3-82 a 3-84

5. EXEMPLOS DE APLICAÇÃO

As secções seguintes ilustram alguns exemplos de aplicação para o controlador de teste.

5.1. CONFIGURAÇÃO EM MODO AUTÓNOMO

A figura 14 ilustra uma aplicação em que o controlador de teste é empregue em modo autónomo (sem um controlador a nível hierárquico superior).

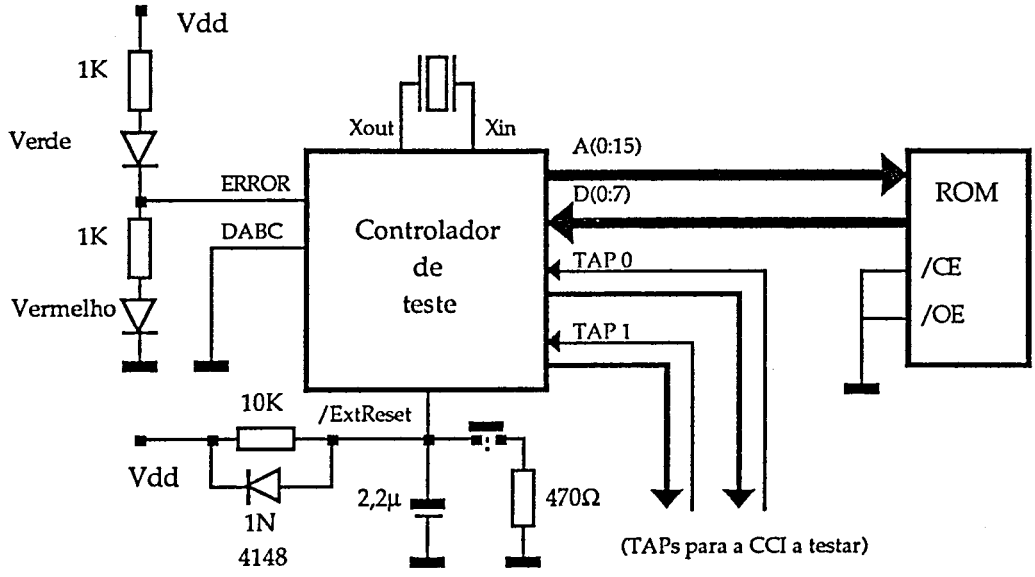


Fig.14: Aplicação em configuração autónoma.

5.2. CONFIGURAÇÃO EM TESTE HIERÁRQUICO

A figura 15 ilustra uma aplicação em o mesmo controlador de teste é empregue em diferentes níveis hierárquicos. Todos os controladores representados devem dispor de recursos próprios (locais) para a inicialização, e geração de relógio.

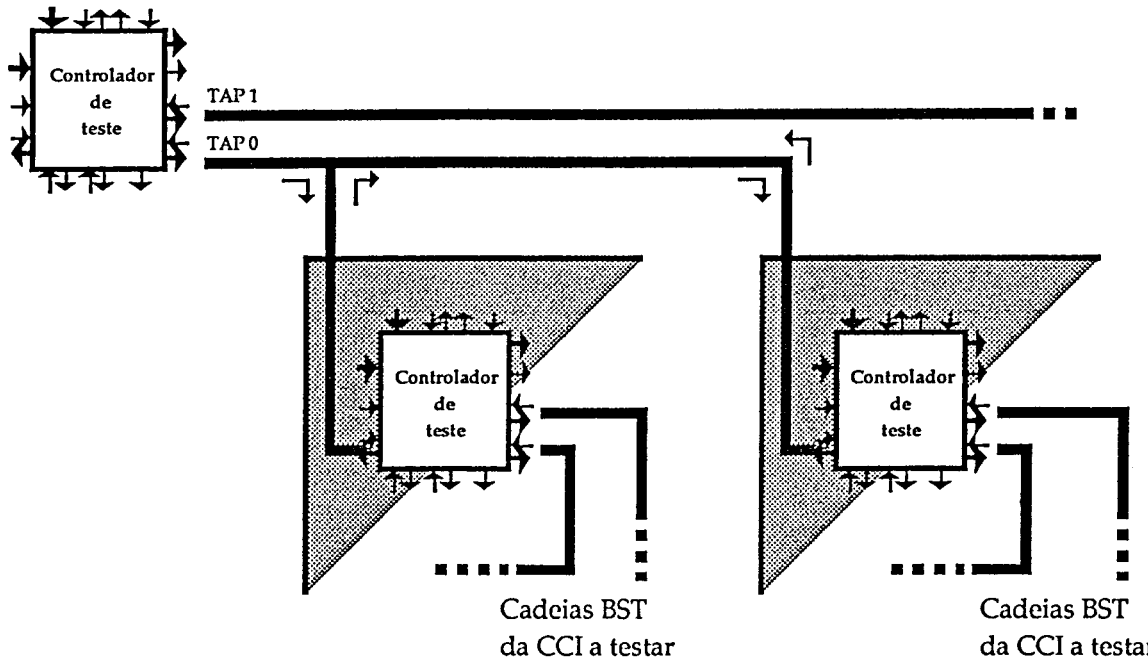


Fig.15: Aplicação em configuração hierárquica.

Cada controlador disporá de uma memória local onde estará armazenado o correspondente programa de teste, que poderá ser executado por ordem do controlador de teste residente a nível hierárquico superior. O segmento de programa apresentado a seguir permitiria a este controlador ordenar a execução do programa de teste na primeira (à esquerda) das duas CCIs.

```
inicio    .org    0h
                              ; está seleccionado o TAP0 (por omissão)
                              ; está seleccionada a flag de erro 0 (por omissão)
          tms1
          tms1
          tms1
          tms1
          tms1                ; todos os controladores de CCI estão em Test Logic Reset
          tms0                ; >> Run Test / Idle
          tms1                ; >> Select-DR-scan
          tms1                ; >> Select-IR-scan
          tms0                ; >> Capture-IR
          tms0                ; >> Shift-IR
          ld      cnt, 4      ; carrega contador com 4
```

```
inf     nshfcp                  ; desloca 0111 (Run Board Test, Bypass), espera 0101
        .db     $07, $05, $0F
        jpe     fim             ; interrompe, se detectar uma falta na infraestrutura
        serflg1                 ; selecciona a flag de erro correspondente à CCI a testar
        tms1                    ; >> Update-IR
        tms0                    ; >> Run Test / Idle
        ld      cnt, 50000      ; carrega contador com 50.000
teste   ntck                    ; executa o teste na CCI pretendida
        tms1                    ; >> Select-DR-scan
        tms0                    ; >> Capture-DR
        tms0                    ; >> Shift-DR
        ld      cnt, 18         ; >> carrega o contador com 18 (compr. do resultado + bypass)
result  nshfcp                  ; desloca 0s, espera tudo 0s, excepto a flag de fim-de-teste
        .db     $00, $02, $FF, $00, $00, $FF, $00, $00, $03
fim     tms1                    ; >> Update-DR
        tms1                    ; >> Select-DR-scan
        tms1                    ; >> Select-IR-scan
        tms1                    ; >> Test Logic Reset
        halt                    ; termina a execução do programa de teste
                                ; o resultado do teste está presente na saída Error
        .end
```

Anexo B

Exemplo de demonstração

1. DESCRIÇÃO DA CCI DE DEMONSTRAÇÃO

A CCI descrita neste anexo permite demonstrar o funcionamento do controlador de teste, e os resultados produzidos pelo conjunto de programas responsáveis pela GAPT. O esquema eléctrico desta CCI está apresentado na figura 1, sendo de realçar os seguintes aspectos:

- Existem duas cadeias BST. A primeira (cadeia BST 0) inclui os componentes BST IC3 e IC4 (SN74BCT8244), e a segunda (cadeia BST 1) o componente BST IC5 (também um SN74BCT8244).
- Existem dois grupos de componentes não BST. O primeiro destes grupos é constituído pelos componentes IC1 e IC2 (74LS139 e 74LS04), e encontra-se completamente envolvido pela cadeia BST 0. O segundo (grupo 1) é constituído pelo componente IC6 (74LS157), e tem ligações para ambas as cadeias BST.
- Um conjunto de 17 *jumpers* permite provocar faltas de continuidade, ou curto-circuitos, em diversos pontos do circuito.
- O funcionamento do controlador de teste tem que estar sincronizado com um equipamento de teste exterior, responsável pela aplicação de estímulos nas oito entradas primárias (IN0 a IN7), e pela captura de respostas nas oito saídas primárias (OUT0 a OUT7).

Esta mesma CCI permitiu igualmente a demonstração de resultados obtidos no âmbito do subprojecto 4 do projecto ESPRIT 2478, tal como se descreve em (Ferreira et al., 92c).

2. INFORMAÇÃO REQUERIDA PELA GAPT

Esta secção descreve a informação necessária para permitir a geração automática dos segmentos do programa de teste correspondentes às três etapas principais (teste da infraestrutura BST, teste das ligações, e teste de componentes), de acordo com os requisitos apresentados no capítulo 7 (A geração automática do programa de teste).

A figura 2 identifica a sequência de células nas duas cadeias BST, e os grupos de componentes não BST, presentes nesta CCI.

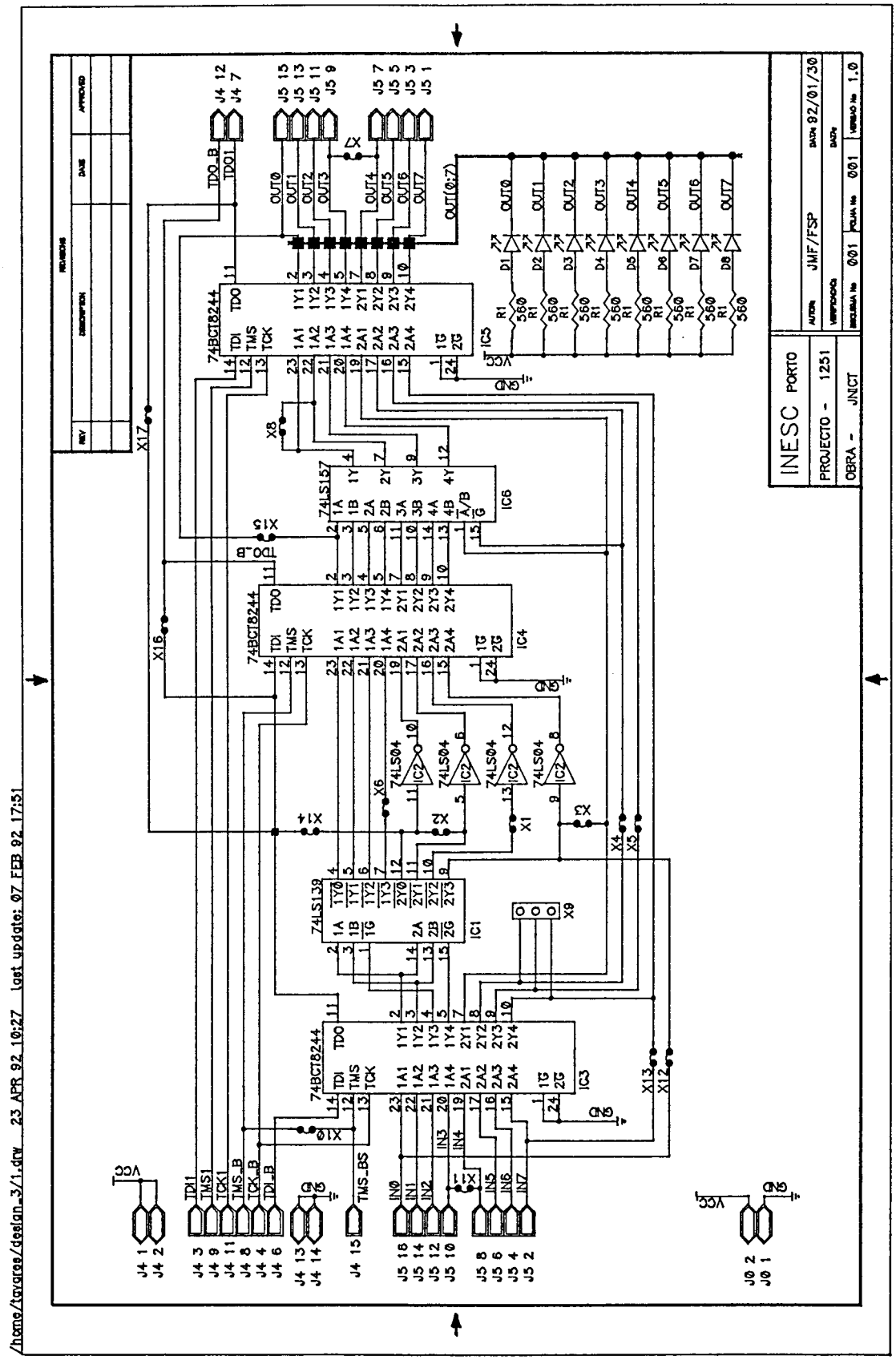


Fig.1: Esquema eléctrico da CCI de demonstração.

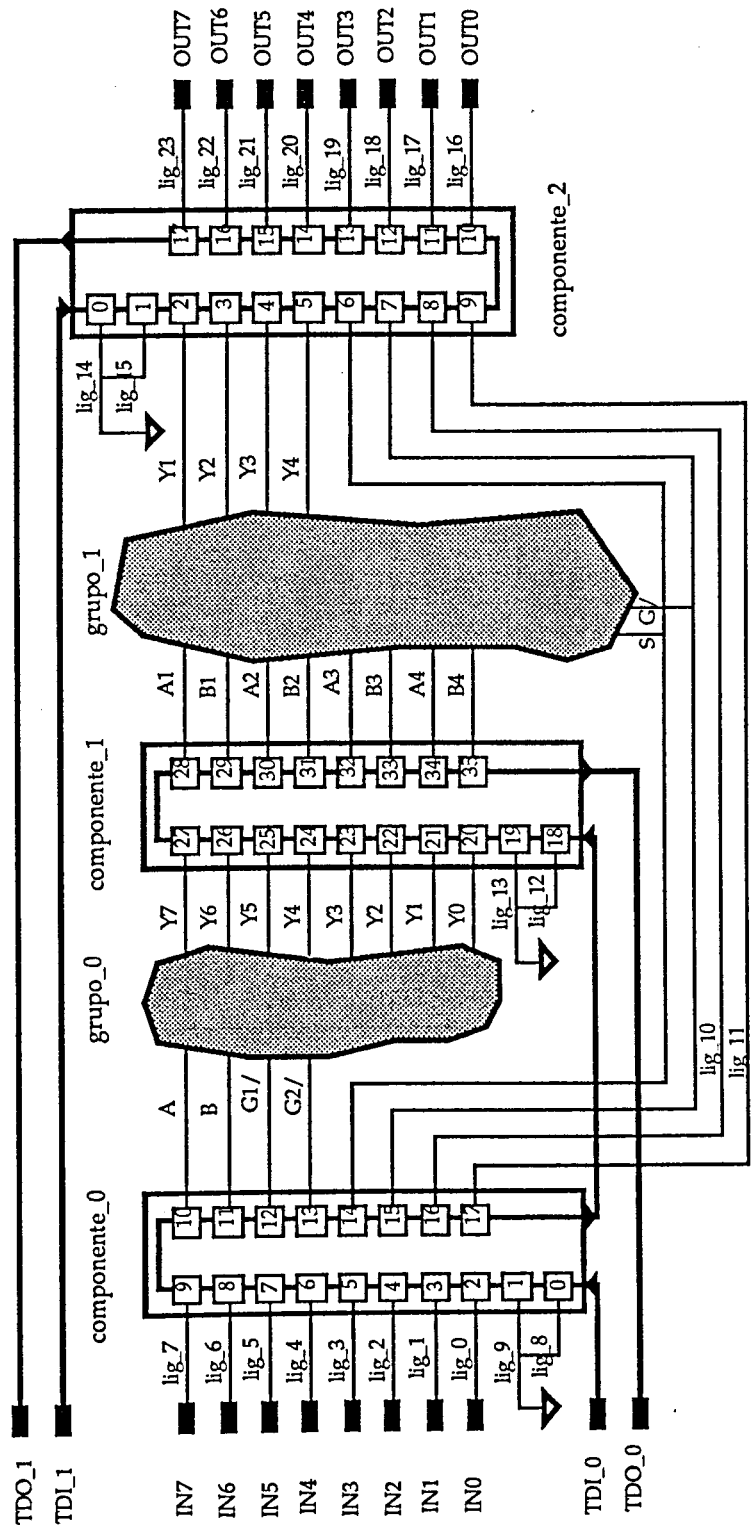


Fig.2: CCI de demonstração: ligações e componentes presentes nas duas cadeias BST.

2.1. Teste da infraestrutura BST

A informação necessária à geração automática do segmento do programa de teste para a infraestrutura BST da CCI está representada no quadro 1. A informação contida neste quadro corresponde aos requisitos apresentados na secção 7.2 (A informação necessária à GAPT), quadro 7.5.

	Identif. da cadeia BST	Ordem do compo-nente	Compr. reg. instr. (RI)	Conteúdo do RI após captura	Existe reg. identif.?	Código de *Sample/ Preload*
componente_0	0	0	8	XXXXXX01	não	00000010
componente_1	0	1	8	XXXXXX01	não	00000010
componente_2	1	0	8	XXXXXX01	não	00000010

Quadro 1: Informação necessária para o teste da infraestrutura BST da CCI.

2.2. Teste de ligações

Esta etapa inclui o teste das ligações BST, e também do conjunto de ligações pertencentes a grupos de componentes não BST.

2.2.1. Ligações BST

A CCI de demonstração inclui as 24 ligações BST representadas na figura 2. A informação necessária para a geração do respectivo segmento do programa de teste está representada no quadro 2. A informação contida neste quadro corresponde aos requisitos apresentados na secção 7.2 (A informação necessária à GAPT), quadro 7.7. Com a excepção das ligações a valores lógicos fixos, todas as restantes ligações descritas no quadro 2 são comandadas por andares de saída com terceiro estado. Repare-se ainda que só as ligações 10 e 11 incluem nós pertencentes a ambas as cadeias BST.

	Núm. cad. BST	Ident. cad. BST	Núm. pin. saíd.	Cél. saída	Cél. contr.	Alta imp.	Núm. pinos entr.	Cél. entrada	Núm. pinos bid.	Núm. entr. prim.	Entr. prim.	Alta imp.	Núm. saída prim.	Saída prim.	Pot. fixo?	Nível lógico
lig_0	1	0	0	-	-	-	1	2	0	1	0	0	0	-	não	-
lig_1	1	0	0	-	-	-	1	3	0	1	1	0	0	-	não	-
lig_2	1	0	0	-	-	-	1	4	0	1	2	0	0	-	não	-
lig_3	1	0	0	-	-	-	1	5	0	1	3	0	0	-	não	-
lig_4	1	0	0	-	-	-	1	6	0	1	4	0	0	-	não	-
lig_5	1	0	0	-	-	-	1	7	0	1	5	0	0	-	não	-
lig_6	1	0	0	-	-	-	1	8	0	1	6	0	0	-	não	-
lig_7	1	0	0	-	-	-	1	9	0	1	7	0	0	-	não	-
lig_8	1	0	0	-	-	-	1	0	0	0	-	-	0	-	sim	0
lig_9	1	0	0	-	-	-	1	1	0	0	-	-	0	-	sim	0
lig_10	2	0	1	16	1	1	0	-	0	0	-	-	0	-	não	-
		1	0	-	-	-	1	8	0							
lig_11	2	0	1	17	1	1	0	-	0	0	-	-	0	-	não	-
		1	0	-	-	-	1	9	0							
lig_12	1	0	0	-	-	-	1	18	0	0	-	-	0	-	sim	0
lig_13	1	0	0	-	-	-	1	19	0	0	-	-	0	-	sim	0
lig_14	1	1	0	-	-	-	1	0	0	0	-	-	0	-	sim	0
lig_15	1	1	0	-	-	-	1	1	0	0	-	-	0	-	sim	0
lig_16	1	1	1	10	0	1	0	-	0	0	-	-	1	0	não	-
lig_17	1	1	1	11	0	1	0	-	0	0	-	-	1	1	não	-
lig_18	1	1	1	12	0	1	0	-	0	0	-	-	1	2	não	-
lig_19	1	1	1	13	0	1	0	-	0	0	-	-	1	3	não	-
lig_20	1	1	1	14	1	1	0	-	0	0	-	-	1	4	não	-
lig_21	1	1	1	15	1	1	0	-	0	0	-	-	1	5	não	-
lig_22	1	1	1	16	1	1	0	-	0	0	-	-	1	6	não	-
lig_23	1	1	1	17	1	1	0	-	0	0	-	-	1	7	não	-

Quadro 2: Informação necessária para o teste das ligações BST.

2.2.2. Grupos de componentes não BST

A CCI de demonstração inclui dois grupos de componentes não BST (grupo 0, e grupo 1), como se ilustra na figura 2.

O grupo 0 é constituído pelos componentes designados como IC1 e IC2 (SN74LS139 e SN74LS04) na figura 1, e está completamente envolvido por células pertencentes à cadeia BST 0, tal como se apresenta na figura 3.

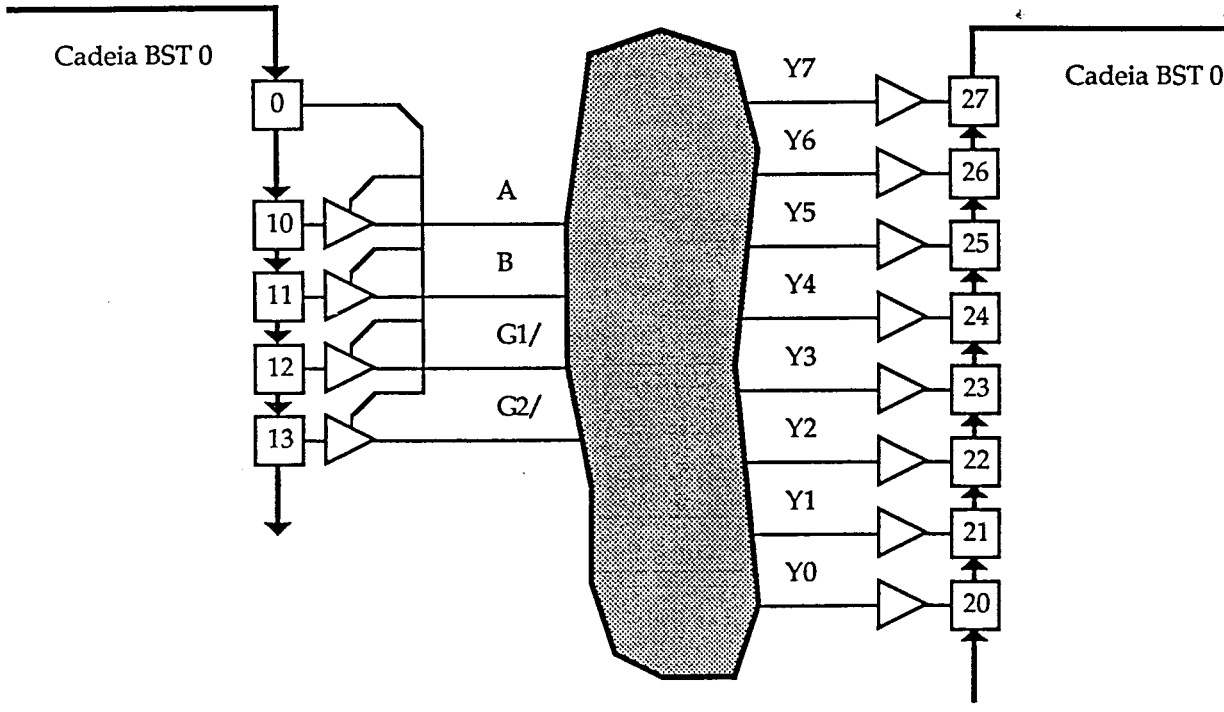


Fig.3: Identificação das células BST que permitem o teste do grupo 0.

A geração de vectores de teste para o grupo 0 de componentes não BST foi realizada no sistema HILO (GenRad), tendo produzido o seguinte resultado:

```
Waveform JMF3 ;
    Input A B GBARRA1 GBARRA2 ;
    Output Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7 ;
    Strobeoffset 45  ;
    Begin
        ** TG ==========================================================
        ** TG Task SPOT
        ** TG ==========================================================
        ** TG   Problem SPOT_FAULTS
        ** TG     Working in Combinational Area (Default)
        ** TG     Trying to catch stuck-at-0 on Y3
        ** TG     Trying to catch stuck-at-0 on Y7
        ** TG     Trying to catch 2 faults
        ** TG     Applying RAPS to combinational area (Default)
        B := Bin 1
            A := Bin 1
            GBARRA2 := Bin 0
            GBARRA1 := Bin 1
            Strobe ( TG001 )
            Y0 = Bin 1
            Y1 = Bin 1
            Y2 = Bin 1
            Y3 = Bin 1
            Y4 = Bin 0
            Y5 = Bin 0
            Y6 = Bin 0
            Y7 = Bin 1
            ** TG   End_problem SPOT_FAULTS
            ;
            ** TG ==========================================================
            ** TG Task SPOT repeated
            ** TG ==========================================================
            ** TG   Problem SPOT_FAULTS
            ** TG     Working in Combinational Area (Default)
            ** TG     Trying to catch stuck-at-1 on Y3
            ** TG     Trying to catch stuck-at-0 on X7
            ** TG     Trying to catch 2 faults
            ** TG     Applying RAPS to combinational area (Default)
            GBARRA2 := Bin 1
            GBARRA1 := Bin 0
            Strobe ( TG002 )
            Y3 = Bin 0
            Y7 = Bin 0
            ** TG   End_problem SPOT_FAULTS
            ;
            ** TG ==========================================================
            ** TG Task SPOT repeated
            ** TG ==========================================================
            ** TG   Problem SPOT_FAULTS
            ** TG     Working in Combinational Area (Default)
            ** TG     Trying to catch stuck-at-1 on B
            ** TG     Trying to catch stuck-at-0 on Y4
            ** TG     Trying to catch 2 faults
            ** TG     Applying RAPS to combinational area (Default)
            B := Bin 0
            A := Bin 0
            GBARRA2 := Bin 0
            Strobe ( TG003 )
            Y0 = Bin 0
            Y3 = Bin 1
            Y4 = Bin 1
            ** TG   End_problem SPOT_FAULTS
```

```
                ;
                ** TG ═══════════════════════════════════════════════════════
                ** TG Task SPOT repeated
                ** TG ═══════════════════════════════════════════════════════
                ** TG   Problem SPOT_FAULTS
                ** TG     Working in Combinational Area (Default)
                ** TG     Trying to catch stuck-at-1 on Y2
                ** TG     Trying to catch stuck-at-0 on Y6
                ** TG     Trying to catch 2 faults
                ** TG     Applying RAPS to combinational area (Default)
                B := Bin 1
                Strobe ( TG004 )
                Y0 = Bin 1
                Y2 = Bin 0
                Y4 = Bin 0
                Y6 = Bin 1
                ** TG   End_problem SPOT_FAULTS
                ;
                ** TG ═══════════════════════════════════════════════════════
                ** TG Task SPOT repeated
                ** TG ═══════════════════════════════════════════════════════
                ** TG   Problem SPOT_FAULTS
                ** TG     Working in Combinational Area (Default)
                ** TG     Trying to catch stuck-at-1 on Y1
                ** TG     Trying to catch stuck-at-0 on Y5
                ** TG     Trying to catch 2 faults
                ** TG     Applying RAPS to combinational area (Default)
                B := Bin 0
                A := Bin 1
                Strobe ( TG005 )
                Y1 = Bin 0
                Y2 = Bin 1
                Y5 = Bin 1
                Y6 = Bin 0
                ** TG   End_problem SPOT_FAULTS
                ** TG ═══════════════════════════════════════════════════════
                ** TG End_Task SPOT
                ** TG ═══════════════════════════════════════════════════════
                ;
    End
Endwaveform JMF3
```

Os cinco vectores de teste gerados proporcionam uma cobertura de faltas de 100%, relativamente a faltas do tipo *stuck-at* em cada pino funcional dos componentes IC1 e IC2. Estes vectores foram integrados na informação necessária à geração do segmento do programa de teste relativo ao grupo 0 de componentes não BST, que está descrita nos quadros 3 e 4. A informação contida nestes quadros corresponde aos requisitos apresentados na secção 7.2 (A informação necessária à GAPT), quadro 7.8.

Núm. cadeias BST	Ident. cadeia BST	Núm. células	Identificação das células BST	Núm. entr. prim.	Núm. saídas prim.	Aplicar valores durante o teste às ligações BST?
1	0	13	0,10,11,12,13,20,21,22,23,24,25,26,27	0	0	não

Quadro 3: Informação necessária para o teste do grupo 0 (excepto os vectores de teste).

Número de vectores	Cadeia BST	Vectores		
		Valores a deslocar	Valores esperados	Máscaras
5	0	0111011111111	1111111110001	0000011111111
		0110111111111	1111111100000	0000011111111
		0000011111111	1111101111000	0000011111111
		0010011111111	1111111010010	0000011111111
		0100011111111	1111110110100	0000011111111

Quadro 4: Descrição dos vectores de teste para o grupo 0.

O grupo 1 é constituído pelo componente IC3 (SN74LS157) da figura 1, e apresenta ligações para células BST em ambas as cadeias BST, tal como se ilustra na figura 4.

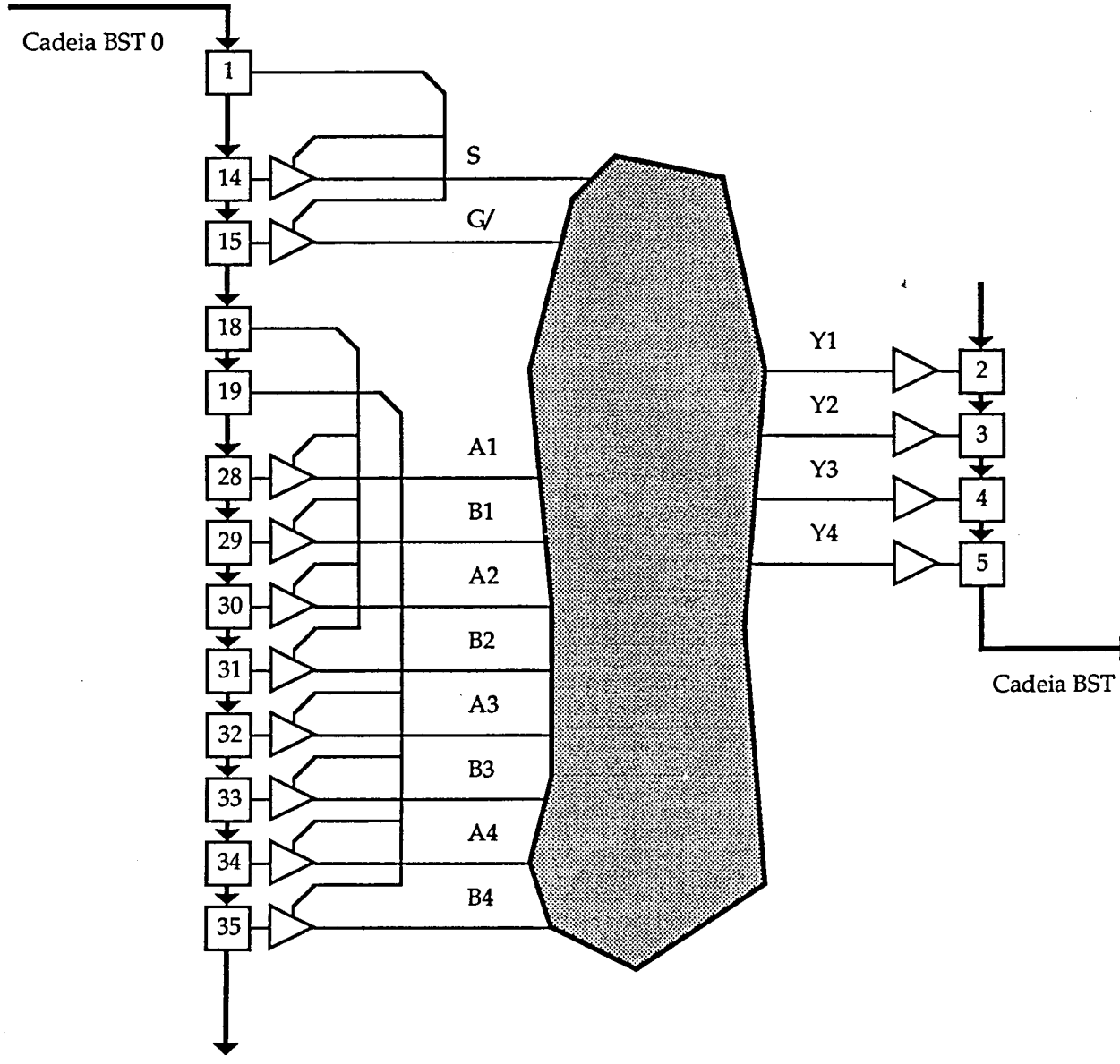


Fig.4: Identificação das células BST que permitem o teste do grupo 1.

A geração dos vectores de teste para este grupo de componentes não BST foi igualmente realizada no sistema HILO (GenRad), tendo produzido o seguinte resultado:

```
Waveform JMF2 ;
    Input DSELECT STRB A[1] A[2] A[3] A[4] B[1] B[2] B[3] B[4] ;
    Output Y[1] Y[2] Y[3] Y[4] ;
    Strobeoffset 45  ;
    Begin
        ** TG ================================================================
        ** TG Task SPOT
        ** TG ================================================================
        ** TG   Problem SPOT_FAULTS
        ** TG     Working in Combinational Area (Default)
        ** TG     Trying to catch stuck-at-0 on Y[4]
        ** TG     Trying to catch stuck-at-0 on Y[3]
        ** TG     Trying to catch stuck-at-0 on Y[2]
        ** TG     Trying to catch stuck-at-0 on Y[1]
        ** TG     Trying to catch 4 faults
        ** TG     Applying RAPS to combinational area (Default)
        STRB := Bin 0
            DSELECT := Bin 0
            B[4] := Bin 0
            B[3] := Bin 0
            B[2] := Bin 1
            B[1] := Bin 1
            A[4] := Bin 1
            A[3] := Bin 1
            A[2] := Bin 1
            A[1] := Bin 1
            Strobe ( TG001 )
            Y[1] = Bin 1
            Y[2] = Bin 1
            Y[3] = Bin 1
            Y[4] = Bin 1
            ** TG   End_problem SPOT_FAULTS
            ;
            ** TG ================================================================
            ** TG Task SPOT repeated
            ** TG ================================================================
            ** TG   Problem SPOT_FAULTS
            ** TG     Working in Combinational Area (Default)
            ** TG     Trying to catch stuck-at-1 on Y[4]
            ** TG     Trying to catch stuck-at-1 on Y[3]
            ** TG     Trying to catch stuck-at-1 on Y[2]
            ** TG     Trying to catch stuck-at-1 on Y[1]
            ** TG     Trying to catch 4 faults
            ** TG     Applying RAPS to combinational area (Default)
            B[4] := Bin 1
            B[1] := Bin 0
            A[4] := Bin 0
            A[3] := Bin 0
            A[2] := Bin 0
            A[1] := Bin 0
            Strobe ( TG002 )
            Y[1] = Bin 0
            Y[2] = Bin 0
            Y[3] = Bin 0
            Y[4] = Bin 0
            ** TG   End_problem SPOT_FAULTS
            ;
            ** TG ================================================================
            ** TG Task SPOT repeated
            ** TG ================================================================
            ** TG   Problem SPOT_FAULTS
```

```
                ** TG     Working in Combinational Area (Default)
                ** TG     Trying to catch stuck-at-0 on STRBWIRE
                ** TG     Trying to catch 1 fault
                ** TG     Applying RAPS to combinational area (Default)
                STRB := Bin 1
                DSELECT := Bin 1
                B[3] := Bin 1
                A[4] := Bin 1
                A[2] := Bin 1
                Strobe ( TG003 )
                ** TG   End_problem SPOT_FAULTS
                ;
                ** TG ================================================================
                ** TG Task SPOT repeated
                ** TG ================================================================
                ** TG   Problem SPOT_FAULTS
                ** TG     Working in Combinational Area (Default)
                ** TG     Trying to catch stuck-at-0 on SELWIRE
                ** TG     Trying to catch stuck-at-0 on B[1]
                ** TG     Trying to catch stuck-at-0 on B[2]
                ** TG     Trying to catch stuck-at-0 on B[3]
                ** TG     Trying to catch 4 faults
                ** TG     Applying RAPS to combinational area (Default)
                STRB := Bin 0
                B[1] := Bin 1
                A[4] := Bin 0
                A[2] := Bin 0
                A[1] := Bin 1
                Strobe ( TG004 )
                Y[1] = Bin 1
                Y[2] = Bin 1
                Y[3] = Bin 1
                Y[4] = Bin 1
                ** TG   End_problem SPOT_FAULTS
                ;
                ** TG ================================================================
                ** TG Task SPOT repeated
                ** TG ================================================================
                ** TG   Problem SPOT_FAULTS
                ** TG     Working in Combinational Area (Default)
                ** TG     Trying to catch stuck-at-1 on B[1]
                ** TG     Trying to catch stuck-at-1 on B[2]
                ** TG     Trying to catch stuck-at-1 on B[3]
                ** TG     Trying to catch stuck-at-1 on B[4]
                ** TG     Trying to catch 4 faults
                ** TG     Applying RAPS to combinational area (Default)
                B[4] := Bin 0
                B[3] := Bin 0
                B[2] := Bin 0
                B[1] := Bin 0
                A[4] := Bin 1
                A[2] := Bin 1
                A[1] := Bin 0
                Strobe ( TG005 )
                Y[1] = Bin 0
                Y[2] = Bin 0
                Y[3] = Bin 0
                Y[4] = Bin 0
                ** TG   End_problem SPOT_FAULTS
                ** TG ================================================================
                ** TG End_Task SPOT
                ** TG ================================================================
                ;
    End
Endwaveform JMF2
```

Os cinco vectores de teste gerados proporcionam uma cobertura de faltas de 100%, relativamente a faltas do tipo *stuck-at* em cada pino funcional do componente IC3. Estes vectores foram integrados na informação necessária à geração do segmento do programa de teste relativo ao grupo 1 de componentes não BST, que está descrita nos quadros 5 e 6. A informação contida nestes quadros corresponde aos requisitos apresentados na secção 7.2 (A informação necessária à GAPT), quadro 7.8.

Núm. cadeias BST	Ident. cadeia BST	Núm. células	Identificação das células BST	Núm. entr. prim.	Núm. saídas prim.	Aplicar valores durante o teste às ligações BST?
2	0	13	1,14,15,18,19,28,29,30,31,32,33,34,35	0	0	não
	1	4	2,3,4,5			

Quadro 5: Informação necessária para o teste do grupo 1 (excepto os vectores de teste).

Número de vectores	Cadeia BST	Vectores		
		Valores a deslocar	Valores esperados	Máscaras
5	0	0000011111010	1111111111111	0000000000000
		0000000010001	1111111111111	0000000000000
		0110000110111	1111111111111	0000000000000
		0100011010101	1111111111111	0000000000000
		0100000100010	1111111111111	0000000000000
	1	1111	1111	1111
		1111	0000	1111
		1111	0000	1111
		1111	1111	1111
		1111	0000	1111

Quadro 6: Descrição dos vectores de teste para o grupo 1.

2.3. Teste de componentes

A simplicidade da lógica interna dos componentes BST presentes justifica a inexistência de funções de auto-teste, pelo que se optou por efectuar um teste simples, que consiste em comutar o nível lógico em todos os nós de entrada e saída. A informação necessária à geração do segmento do programa para o teste interno dos componentes BST está descrita nos quadros 7 e 8. A informação contida nestes quadros corresponde aos requisitos apresentados na secção 7.2 (A informação necessária à GAPT), quadro 7.9.

	Ident. cadeia BST	Ordem do compon.	Compr. registo BST	Suporta BIST?	Teste interno?	Número de vectores	Código de intest
Componente_0	0	0	18	não	sim	2	00000000
Componente_1	0	1	18	não	sim	2	00000000
Componente_2	1	0	18	não	sim	2	00000000

Quadro 7: Informação necessária ao teste interno dos componentes (excepto os vectores de teste).

	Valores a deslocar	Valores esperados	Máscaras
componente_0	110101010111111111	111111111101010101	000000000011111111
	111010101000000000	000000000010101010	000000000011111111
componente_1	110101010111111111	111111111101010101	000000000011111111
	111010101000000000	000000000010101010	000000000011111111
componente_2	110101010111111111	111111111101010101	000000000011111111
	111010101000000000	000000000010101010	000000000011111111

Quadro 8: Descrição dos vectores para o teste interno dos componentes.

3. RESULTADOS PRODUZIDOS PELA GAPT

Esta secção apresenta os resultados produzidos pelos programas responsáveis pela GAPT. Estes resultados incluem a sequência de valores lógicos atribuídos a cada ligação BST pelo algoritmo para a detecção de curto-circuitos, o conjunto de vectores que serão deslocados para o interior das cadeias BST, durante a realização das três etapas principais, e o programa gerado (em código simbólico — *assembly*) para o controlador de teste.

3.1. Teste de curto-circuitos entre ligações BST

A sequência de valores lógicos atribuídos a cada ligação BST, para a detecção de curto-circuitos, está apresentada no quadro 9. O conjunto de valores aplicados em simultâneo a todas as ligações BST está representado horizontalmente. Cada coluna representa a sequência de valores atribuídos a cada ligação (VTS). Este conjunto de VTSs foi gerado pelo algoritmo destinado à detecção de curto-circuitos entre ligações BST, apresentado no capítulo 7 (procedimento 7.8).

```
ptv[0]:  010101010001000001010101
ptv[1]:  010101010010000010101010
ptv[2]:  011010100001000001011010
ptv[3]:  100110100001000010100101
ptv[4]:  101001100010000001100110
ptv[5]:  101010010010000010011001
```

Quadro 9: Resultado produzido pelo algoritmo para teste de curto-circuitos entre ligações BST.

3.2. Conjunto de vectores de teste

O conjunto completo de vectores de teste cobre as três etapas principais que constituem o teste da CCI (teste da infraestrutura BST, teste das ligações, e teste de componentes), e especifica a sequência de valores lógicos que deverão ser deslocados para o interior das cadeias BST, bem como os respectivos valores esperados (quando existentes), e máscaras de comparação. Os resultados apresentados a seguir correspondem ao conjunto de procedimentos que foram apresentados na secção 7.4 (Procedimentos principais).

```
TOTAL NUMBER OF INFRASTRUCTURE TEST PATTERNS: 4

******************************************************************************
Infrastructure test pattern number 0 (ID or BP register opcodes for IR's):
******************************************************************************

Boundary scan chain number 0:

Test pattern:
TP[0][0]:
          1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1
Expected result (to be shifted out while this test pattern is shifted in):
ER[0][0]:
          0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1
Mask bits (for this expected result):
MB[0][0]:
          1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1

Boundary scan chain number 1:

Test pattern:
TP[1][0]:
          1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1
Expected result (to be shifted out while this test pattern is shifted in):
ER[1][0]:
          0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 1
Mask bits (for this expected result):
MB[1][0]:
          1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1

******************************************************************************
Infrastructure test pattern number 1 (for the selected DRs - ID or BP):
******************************************************************************

Boundary scan chain number 0:

Test pattern:
TP[0][1]:
          1 1
Expected result (to be shifted out while this test pattern is shifted in):
ER[0][1]:
          0 0
Mask bits (for this expected result):
MB[0][1]:
          1 1

Boundary scan chain number 1:

Test pattern:
TP[1][1]:
          1
Expected result (to be shifted out while this test pattern is shifted in):
ER[1][1]:
          0
Mask bits (for this expected result):
MB[1][1]:
          1

******************************************************************************
Infrastructure test pattern number 2 (Sample/Preload opcode for the IR's):
******************************************************************************
```

```
Boundary scan chain number 0:

Test pattern:
TP[0][2]:
          0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0
Expected result (to be shifted out while this test pattern is shifted in):
ER[0][2]:
          1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1
Mask bits (for this expected result):
MB[0][2]:
          0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1

Boundary scan chain number 1:

Test pattern:
TP[1][2]:
          0 0 0 0 0 0 1 0
Expected result (to be shifted out while this test pattern is shifted in):
ER[1][2]:
          1 1 1 1 1 1 0 1
Mask bits (for this expected result):
MB[1][2]:
          0 0 0 0 0 0 1 1

*****************************************************************************
Infrastructure test pattern number 3 (Extest opcode for the IR's):
*****************************************************************************

Boundary scan chain number 0:

Test pattern:
TP[0][3]:
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Expected result (to be shifted out while this test pattern is shifted in):
ER[0][3]:
          1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1
Mask bits (for this expected result):
MB[0][3]:
          0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1

Boundary scan chain number 1:

Test pattern:
TP[1][3]:
          0 0 0 0 0 0 0 0
Expected result (to be shifted out while this test pattern is shifted in):
ER[1][3]:
          1 1 1 1 1 1 0 1
Mask bits (for this expected result):
MB[1][3]:
          0 0 0 0 0 0 1 1


TOTAL NUMBER OF INTERCONNECT TEST PATTERNS: 8

*****************************************************************************
Interconnect test pattern number 0 (for open fault detection):
*****************************************************************************

Boundary scan chain number 0:

Test pattern:
TP[0][0]:
          0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Expected result - shifted out while the next TP is shifted in:
ER[0][0]:
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
Mask bits (for this expected result):
MB[0][0]:
          1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Boundary scan chain number 1:

Test pattern:
TP[1][0]:
          0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0
Expected result - shifted out while the next TP is shifted in:
ER[1][0]:
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Mask bits (for this expected result):
MB[1][0]:
          1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0

Primary inputs and corresponding control signals:

PI[0]:
          0 0 0 0 0 0 0 0
CS[0]:
          1 1 1 1 1 1 1 1

Expected values and masks at primary outputs:

PO[0]:
          0 0 0 0 0 0 0 0
POM[0]:
          1 1 1 1 1 1 1 1

*************************************************************************
Interconnect test pattern number 1 (for open fault detection):
*************************************************************************

Boundary scan chain number 0:

Test pattern:
TP[0][1]:
          0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Expected result - shifted out while the next TP is shifted in:
ER[0][1]:
          0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Mask bits (for this expected result):
MB[0][1]:
          1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Boundary scan chain number 1:

Test pattern:
TP[1][1]:
          0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1
Expected result - shifted out while the next TP is shifted in:
ER[1][1]:
          0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1
Mask bits (for this expected result):
MB[1][1]:
          1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0

Primary inputs and corresponding control signals:

PI[1]:
          1 1 1 1 1 1 1 1
CS[1]:
          1 1 1 1 1 1 1 1

Expected values and masks at primary outputs:
```

```
PO[1]:
          1 1 1 1 1 1 1 1
POM[1]:
          1 1 1 1 1 1 1 1

*****************************************************************************
Interconnect test pattern number 2 (for short fault detection):
*****************************************************************************

Boundary scan chain number 0:

Test pattern:
TP[0][2]:
          0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Expected result - shifted out while the next TP is shifted in:
ER[0][2]:
          0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Mask bits (for this expected result):
MB[0][2]:
          1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0

Boundary scan chain number 1:

Test pattern:
TP[1][2]:
          0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 1 0 1
Expected result - shifted out while the next TP is shifted in:
ER[1][2]:
          0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1
Mask bits (for this expected result):
MB[1][2]:
          1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0

Primary inputs and corresponding control signals:

PI[2]:
          0 1 0 1 0 1 0 1
CS[2]:
          1 1 1 1 1 1 1 1

Expected values and masks at primary outputs:

PO[2]:
          0 1 0 1 0 1 0 1
POM[2]:
          1 1 1 1 1 1 1 1

*****************************************************************************
Interconnect test pattern number 3 (for short fault detection):
*****************************************************************************

Boundary scan chain number 0:

Test pattern:
TP[0][3]:
          0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Expected result - shifted out while the next TP is shifted in:
ER[0][3]:
          0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Mask bits (for this expected result):
MB[0][3]:
          1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0

Boundary scan chain number 1:

Test pattern:
TP[1][3]:
```

```
          0 0 0 0 0 0 0 0 1 1 1 0 1 0 1 0 1 0
Expected result - shifted out while the next TP is shifted in:
ER[1][3]:
          0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 0
Mask bits (for this expected result):
MB[1][3]:
          1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0

Primary inputs and corresponding control signals:

PI[3]:
          0 1 0 1 0 1 0 1
CS[3]:
          1 1 1 1 1 1 1 1

Expected values and masks at primary outputs:

PO[3]:
          1 0 1 0 1 0 1 0
POM[3]:
          1 1 1 1 1 1 1 1

*****************************************************************************
Interconnect test pattern number 4 (for short fault detection):
*****************************************************************************

Boundary scan chain number 0:

Test pattern:
TP[0][4]:
          0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Expected result - shifted out while the next TP is shifted in:
ER[0][4]:
          0 0 0 1 1 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Mask bits (for this expected result):
MB[0][4]:
          1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Boundary scan chain number 1:

Test pattern:
TP[1][4]:
          0 0 0 0 0 0 0 0 1 1 0 1 0 1 1 0 1 0
Expected result - shifted out while the next TP is shifted in:
ER[1][4]:
          0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 0 1 0
Mask bits (for this expected result):
MB[1][4]:
          1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0

Primary inputs and corresponding control signals:

PI[4]:
          0 1 1 0 1 0 1 0
CS[4]:
          1 1 1 1 1 1 1 1

Expected values and masks at primary outputs:

PO[4]:
          0 1 0 1 1 0 1 0
POM[4]:
          1 1 1 1 1 1 1 1

*****************************************************************************
Interconnect test pattern number 5 (for short fault detection):
*****************************************************************************
```

```
Boundary scan chain number 0:

Test pattern:
TP[0][5]:
          0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Expected result - shifted out while the next TP is shifted in:
ER[0][5]:
          0 0 1 0 0 1 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Mask bits (for this expected result):
MB[0][5]:
          1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0

Boundary scan chain number 1:

Test pattern:
TP[1][5]:
          0 0 0 0 0 0 0 0 1 1 1 0 1 0 0 1 0 1
Expected result - shifted out while the next TP is shifted in:
ER[1][5]:
          0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 1 0 1
Mask bits (for this expected result):
MB[1][5]:
          1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0

Primary inputs and corresponding control signals:

PI[5]:
          1 0 0 1 1 0 1 0
CS[5]:
          1 1 1 1 1 1 1 1

Expected values and masks at primary outputs:

PO[5]:
          1 0 1 0 0 1 0 1
POM[5]:
          1 1 1 1 1 1 1 1

*****************************************************************************
Interconnect test pattern number 6 (for short fault detection):
*****************************************************************************

Boundary scan chain number 0:

Test pattern:
TP[0][6]:
          0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Expected result - shifted out while the next TP is shifted in:
ER[0][6]:
          0 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Mask bits (for this expected result):
MB[0][6]:
          1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0

Boundary scan chain number 1:

Test pattern:
TP[1][6]:
          0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 1 1 0
Expected result - shifted out while the next TP is shifted in:
ER[1][6]:
          0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 1 1 0
Mask bits (for this expected result):
MB[1][6]:
          1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0
```

```
Primary inputs and corresponding control signals:

PI[6]:
          1 0 1 0 0 1 1 0
CS[6]:
          1 1 1 1 1 1 1 1

Expected values and masks at primary outputs:

PO[6]:
          0 1 1 0 0 1 1 0
POM[6]:
          1 1 1 1 1 1 1 1

*************************************************************************
Interconnect test pattern number 7 (last TP for short fault detection):
*************************************************************************

Boundary scan chain number 0:

Test pattern:
TP[0][7]:
          0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Expected result - shifted out while a following "dummy" TP is shifted in:
ER[0][7]:
          0 0 1 0 1 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Mask bits (for this expected result):
MB[0][7]:
          1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Boundary scan chain number 1:

Test pattern:
TP[1][7]:
          0 0 0 0 0 0 0 0 1 1 1 0 0 1 1 0 0 1
Expected result - shifted out while a following "dummy" TP is shifted in:
ER[1][7]:
          0 0 0 0 0 0 0 0 1 0 1 0 0 1 1 0 0 1
Mask bits (for this expected result):
MB[1][7]:
          1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0

Primary inputs and corresponding control signals:

PI[7]:
          1 0 1 0 1 0 0 1
CS[7]:
          1 1 1 1 1 1 1 1

Expected values and masks at primary outputs:

PO[7]:
          1 0 0 1 1 0 0 1
POM[7]:
          1 1 1 1 1 1 1 1


TOTAL NUMBER OF CLUSTER TEST PATTERNS: 5

*************************************************************************
Cluster test pattern number 0
*************************************************************************

Boundary scan chain number 0:

Test pattern:
TP[0][8]:
```

```
          0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0
Expected result - shifted out while the next TP is shifted in:
ER[0][8]:
          1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1
Mask bits (for this expected result):
MB[0][8]:
          0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0

Boundary scan chain number 1:

Test pattern:
TP[1][8]:
          0 0 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1
Expected result - shifted out while the next TP is shifted in:
ER[1][8]:
          0 0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1
Mask bits (for this expected result):
MB[1][8]:
          1 1 1 1 1 1 1 0 0 1 1 0 0 0 0 0 0 0 0

Primary inputs and corresponding control signals:

PI[8]:
          1 1 1 1 1 1 1 1
CS[8]:
          1 1 1 1 1 1 1 1

Expected values and masks at primary outputs:

PO[8]:
          1 1 1 1 1 1 1 1
POM[8]:
          1 1 1 1 1 1 1 1

*******************************************************************************
Cluster test pattern number 1
*******************************************************************************

Boundary scan chain number 0:

Test pattern:
TP[0][9]:
          0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 1 1 0 0 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0 1
Expected result - shifted out while the next TP is shifted in:
ER[0][9]:
          1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1
Mask bits (for this expected result):
MB[0][9]:
          0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0

Boundary scan chain number 1:

Test pattern:
TP[1][9]:
          0 0 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1
Expected result - shifted out while the next TP is shifted in:
ER[1][9]:
          0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1
Mask bits (for this expected result):
MB[1][9]:
          1 1 1 1 1 1 1 0 0 1 1 0 0 0 0 0 0 0 0

Primary inputs and corresponding control signals:

PI[9]:
          1 1 1 1 1 1 1 1
CS[9]:
```

```
          1 1 1 1 1 1 1 1

Expected values and masks at primary outputs:

PO[9]:
          1 1 1 1 1 1 1 1
POM[9]:
          1 1 1 1 1 1 1 1

*************************************************************************
Cluster test pattern number 2
*************************************************************************

Boundary scan chain number 0:

Test pattern:
TP[0][10]:
          0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 1 1 1 1 1 1 0 0 1 1 0 1 1 1
Expected result - shifted out while the next TP is shifted in:
ER[0][10]:
          1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1
Mask bits (for this expected result):
MB[0][10]:
          0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0

Boundary scan chain number 1:

Test pattern:
TP[1][10]:
          0 0 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1
Expected result - shifted out while the next TP is shifted in:
ER[1][10]:
          0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1
Mask bits (for this expected result):
MB[1][10]:
          1 1 1 1 1 1 0 0 1 1 0 0 0 0 0 0 0 0

Primary inputs and corresponding control signals:

PI[10]:
          1 1 1 1 1 1 1 1
CS[10]:
          1 1 1 1 1 1 1 1

Expected values and masks at primary outputs:

PO[10]:
          1 1 1 1 1 1 1 1
POM[10]:
          1 1 1 1 1 1 1 1

*************************************************************************
Cluster test pattern number 3
*************************************************************************

Boundary scan chain number 0:

Test pattern:
TP[0][11]:
          0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1
Expected result - shifted out while the next TP is shifted in:
ER[0][11]:
          1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 1 1 1
Mask bits (for this expected result):
MB[0][11]:
          0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
```

```
Boundary scan chain number 1:

Test pattern:
TP[1][11]:
          0 0 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1
Expected result - shifted out while the next TP is shifted in:
ER[1][11]:
          0 0 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1
Mask bits (for this expected result):
MB[1][11]:
          1 1 1 1 1 1 0 0 1 1 0 0 0 0 0 0 0

Primary inputs and corresponding control signals:

PI[11]:
          1 1 1 1 1 1 1 1
CS[11]:
          1 1 1 1 1 1 1 1

Expected values and masks at primary outputs:

PO[11]:
          1 1 1 1 1 1 1 1
POM[11]:
          1 1 1 1 1 1 1 1

*******************************************************************************
Cluster test pattern number 4
*******************************************************************************

Boundary scan chain number 0:

Test pattern:
TP[0][12]:
          0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 0 1 1 1 1 1 1 1 1 0 0 1 0 0 0 1 0
Expected result - shifted out while a following "dummy" TP is shifted in:
ER[0][12]:
          1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1
Mask bits (for this expected result):
MB[0][12]:
          0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0

Boundary scan chain number 1:

Test pattern:
TP[1][12]:
          0 0 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1
Expected result - shifted out while a following "dummy" TP is shifted in:
ER[1][12]:
          0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1
Mask bits (for this expected result):
MB[1][12]:
          1 1 1 1 1 1 0 0 1 1 0 0 0 0 0 0 0

Primary inputs and corresponding control signals:

PI[12]:
          1 1 1 1 1 1 1 1
CS[12]:
          1 1 1 1 1 1 1 1

Expected values and masks at primary outputs:

PO[12]:
          1 1 1 1 1 1 1 1
POM[12]:
          1 1 1 1 1 1 1 1
```

```
TOTAL NUMBER OF COMPONENT BIST TEST PATTERNS: 0


TOTAL NUMBER OF TEST PATTERNS FOR INTERNAL TEST OF COMPONENTS: 3

*****************************************************************************
Internal test of components: Test pattern for shifting the intest/bypass opcodes:
*****************************************************************************

Boundary scan chain number 0:

Test pattern:
TP[0]:
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Boundary scan chain number 1:

Test pattern:
TP[1]:
          0 0 0 0 0 0 0 0

*****************************************************************************
Internal test of components: Test pattern number 0.
*****************************************************************************

Boundary scan chain number 0:

Test pattern:
TP[0][0]:
          1 1 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1
Expected result - shifted out while the next TP is shifted in:
ER[0][0]:
          1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 0 1
Mask bits (for this expected result):
MB[0][0]:
          0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1

Boundary scan chain number 1:

Test pattern:
TP[1][0]:
          1 1 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1
Expected result - shifted out while the next TP is shifted in:
ER[1][0]:
          1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 0 1
Mask bits (for this expected result):
MB[1][0]:
          0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1

*****************************************************************************
Internal test of components: Test pattern number 1.
*****************************************************************************

Boundary scan chain number 0:

Test pattern:
TP[0][1]:
          1 1 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0
Expected result - shifted out while the next TP is shifted in:
ER[0][1]:
          0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0
Mask bits (for this expected result):
MB[0][1]:
```

```
          0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
Boundary scan chain number 1:

Test pattern:
TP[1][1]:
          1 1 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0
Expected result - shifted out while the next TP is shifted in:
ER[1][1]:
          0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1
Mask bits (for this expected result):
MB[1][1]:
          0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
```

3.3. Programa de teste

O programa de teste que permite a aplicação dos vectores descritos foi igualmente produzido pela GAPT, em linguagem simbólica (*assembly*). Este programa de teste obedece às directivas apresentadas na secção 7.5 (Integração do conjunto de vectores gerados). O programa de teste gerado foi posteriormente convertido para código objecto com o auxílio do *cross-assembler* TASM †, que produziu a listagem apresentada a seguir.

```
0001   0000
0002   0000                ;*************************************************************
0003   0000                ;       Assembly source code for the demophd circuit.
0004   0000                ;*************************************************************
0005   0000
0006   0000                start   .org  0h
0007   0000
0008   0000
0009   0000                ; Sequence to reset all BST logic.
0010   0000
0011   0000 1A                     seltap0       ; (switch to) TAP 0
0012   0001 11                     trst          ; TRST0 output is pulsed low
0013   0002 01                     tms1
0014   0003 01                     tms1
0015   0004 01                     tms1
0016   0005 01                     tms1
0017   0006 01                     tms1          ; soft reset of BST chain
0018   0007 1B                     seltap1       ; switch to TAP 1
0019   0008 11                     trst          ; TRST1 output is pulsed low
0020   0009 01                     tms1
0021   000A 01                     tms1
0022   000B 01                     tms1
0023   000C 01                     tms1
0024   000D 01                     tms1          ; soft reset of BST chain 1
0025   000E 1A                     seltap0       ; switch to TAP 0
0026   000F
0027   000F                ; Sequence to test the BST infrastructure
0028   000F
0029   000F 00                     tms0          ; BST chain 0, >> Run Test / Idle
0030   0010 01                     tms1          ; BST chain 0, >> Select DR scan
0031   0011 01                     tms1          ; BST chain 0, >> Select IR scan
0032   0012 00                     tms0          ; BST chain 0, >> Capture IR
0033   0013 00                     tms0          ; BST chain 0, >> Shift IR
0034   0014 02 00 1A               ld    cnt,26  ; length of IR's + 10, BST chain 0
0035   0017 05                     nshfcp        ; shift in the ID or BP opcodes
0036   0018 01FD03FCFD03 inf0      .db   $01,$fd,$03,$fc,$fd,$03,$ff,$01,$ff,$03,$00,$03
0037   0024 06 04 01               jpe   theend  ; stop the test, if a BST infrastructure fault is found
0038   0027 01                     tms1          ; BST chain 0, >> Update IR
0039   0028 01                     tms1          ; BST chain 0, >> Select DR scan
0040   0029 00                     tms0          ; BST chain 0, >> Capture DR
0041   002A 00                     tms0          ; BST chain 0, >> Shift DR
0042   002B 02 00 02               ld    cnt,2   ; length of ID (BP, if no ID) registers, BST chain 0
0043   002E 05                     nshfcp        ; shift out the contents of the ID (BP, if no ID) registers
0044   002F 03 00 03        inf1  .db   $03,$00,$03
0045   0032 06 04 01               jpe   theend  ; stop the test, if a BST infrastructure fault is found
0046   0035 01                     tms1          ; BST chain 0, >> Update DR
0047   0036 01                     tms1          ; BST chain 0, >> Select DR scan
0048   0037 01                     tms1          ; BST chain 0, >> Select IR scan
0049   0038 00                     tms0          ; BST chain 0, >> Capture IR
0050   0039 00                     tms0          ; BST chain 0, >> Shift IR
0051   003A 02 00 10               ld    cnt,16  ; length of IR's, BST chain 0
0052   003D 05                     nshfcp        ; shift in the sample/preload opcodes
0053   003E 02FD0302FD03   inf2    .db   $02,$fd,$03,$02,$fd,$03
0054   0044 06 04 01               jpe   theend  ; stop the test, if a BST infrastructure fault is found
0055   0047 01                     tms1          ; BST chain 0, >> Update IR
0056   0048 01                     tms1          ; BST chain 0, >> Select DR scan
0057   0049 00                     tms0          ; BST chain 0, >> Capture DR
0058   004A 01                     tms1          ; BST chain 0, >> Exit-1 DR
0059   004B 01                     tms1          ; BST chain 0, >> Update DR
```

† © Speech Tecnology Incorporated.

```
0060   004C 01                     tms1          ; BST chain 0, >> Select DR scan
0061   004D 01                     tms1          ; BST chain 0, >> Select IR scan
0062   004E 00                     tms0          ; BST chain 0, >> Capture IR
0063   004F 00                     tms0          ; BST chain 0, >> Shift IR
0064   0050 02 00 10               ld    cnt,16  ; length of IR's, BST chain 0
0065   0053 05                     nshfcp        ; shift in the extest opcodes
0066   0054 00FD0300FD03    inf3   .db   $00,$fd,$03,$00,$fd,$03
0067   005A 06 04 01               jpe   theend  ; stop the test, if a BST infrastructure fault is found
0068   005D 01                     tms1          ; BST chain 0, >> Update IR
0069   005E 01                     tms1          ; BST chain 0, >> Select DR scan
0070   005F 00                     tms0          ; BST chain 0, >> Capture DR
0071   0060 00                     tms0          ; BST chain 0, >> Shift DR
0072   0061 1B                     seltap1       ; switch to TAP 1
0073   0062 00                     tms0          ; BST chain 1, >> Run Test / Idle
0074   0063 01                     tms1          ; BST chain 1, >> Select DR scan
0075   0064 01                     tms1          ; BST chain 1, >> Select IR scan
0076   0065 00                     tms0          ; BST chain 1, >> Capture IR
0077   0066 00                     tms0          ; BST chain 1, >> Shift IR
0078   0067 02 00 12               ld    cnt,18  ; length of IR's + 10, BST chain 1
0079   006A 05                     nshfcp        ; shift in the ID or BP opcodes
0080   006B 01FD03FC01FF    inf4   .db   $01,$fd,$03,$fc,$01,$ff,$03,$00,$03
0081   0074 06 04 01               jpe   theend  ; stop the test, if a BST infrastructure fault is found
0082   0077 01                     tms1          ; BST chain 1, >> Update DR
0083   0078 01                     tms1          ; BST chain 1, >> Select DR scan
0084   0079 00                     tms0          ; BST chain 1, >> Capture DR
0085   007A 00                     tms0          ; BST chain 1, >> Shift DR
0086   007B 02 00 01               ld    cnt,1   ; length of ID (BP, if no ID) registers, BST chain 1
0087   007E 05                     nshfcp        ; shift out the contents of the ID (BP, if no ID) registers
0088   007F 01 00 01        inf5   .db   $01,$00,$01
0089   0082 06 04 01               jpe   theend  ; stop the test, if a BST infrastructure fault is found
0090   0085 01                     tms1          ; BST chain 1, >> Update DR
0091   0086 01                     tms1          ; BST chain 1, >> Select DR scan
0092   0087 01                     tms1          ; BST chain 1, >> Select IR scan
0093   0088 00                     tms0          ; BST chain 1, >> Capture IR
0094   0089 00                     tms0          ; BST chain 1, >> Shift IR
0095   008A 02 00 08               ld    cnt,8   ; length of IR's, BST chain 1
0096   008D 05                     nshfcp        ; shift in the sample/preload opcodes
0097   008E 02 FD 03        inf6   .db   $02,$fd,$03
0098   0091 06 04 01               jpe   theend  ; stop the test, if a BST infrastructure fault is found
0099   0094 01                     tms1          ; BST chain 1, >> Update IR
0100   0095 01                     tms1          ; BST chain 1, >> Select DR scan
0101   0096 00                     tms0          ; BST chain 1, >> Capture DR
0102   0097 01                     tms1          ; BST chain 1, >> Exit-1 DR
0103   0098 01                     tms1          ; BST chain 1, >> Update DR
0104   0099 01                     tms1          ; BST chain 1, >> Select DR scan
0105   009A 01                     tms1          ; BST chain 1, >> Select IR scan
0106   009B 00                     tms0          ; BST chain 1, >> Capture IR
0107   009C 00                     tms0          ; BST chain 1, >> Shift IR
0108   009D 02 00 08               ld    cnt,8   ; length of IR's, BST chain 1
0109   00A0 05                     nshfcp        ; shift in the extest opcodes
0110   00A1 00 FD 03        inf7   .db   $00,$fd,$03
0111   00A4 06 04 01               jpe   theend  ; stop the test, if a BST infrastructure fault is found
0112   00A7 01                     tms1          ; BST chain 1, >> Update DR
0113   00A8 01                     tms1          ; BST chain 1, >> Select DR scan
0114   00A9 00                     tms0          ; BST chain 1, >> Capture DR
0115   00AA 00                     tms0          ; BST chain 1, >> Shift DR
0116   00AB 1A                     seltap0       ; switch to TAP 0
0117   00AC
0118   00AC                 ; Sequence for testing the full BST interconnects
0119   00AC
0120   00AC 0B                     serflg3       ; select error flag 3 (for full BST interconnect open faults)
0121   00AD 02 00 24               ld    cnt,36  ; length of BST chain 0
0122   00B0 04                     nshf          ; shift 1st TP (for open testing), no compare
0123   00B1 000000FC03      int0   .db   $00,$00,$00,$fc,$03
0124   00B6 01                     tms1          ; BST chain 0, >> Update DR
0125   00B7 01                     tms1          ; BST chain 0, >> Sel. DR scan
0126   00B8 1B                     seltap1       ; switch to TAP 1
0127   00B9 02 00 12               ld    cnt,18  ; length of BST chain 1
0128   00BC 04                     nshf          ; shift the 1st TP (for open faults) on BST chain 1
0129   00BD 00 03 00        int1   .db   $00,$03,$00
0130   00C0 01                     tms1          ; BST chain 1, >> Update DR
0131   00C1 01                     tms1          ; BST chain 1, >> Sel. DR scan
0132   00C2 13                     ssa1          ; SyncOutA goes high (BST test data applied)
0133   00C3 15                     wsa1          ; wait for SyncInpA to go high (external test data applied)
0134   00C4 00                     tms0          ; BST chain 1, >> Capture DR
0135   00C5 00                     tms0          ; BST chain 1, >> Shift DR
0136   00C6 1A                     seltap0       ; switch to TAP 0
0137   00C7 00                     tms0          ; BST chain 0, >> Capture DR
0138   00C8 00                     tms0          ; BST chain 0, >> Shift DR
0139   00C9 12                     ssa0          ; SyncOutA goes low (BST test results captured)
0140   00CA 14                     wsa0          ; wait for SyncInpA to go low (external test results captured)
0141   00CB 02 00 24               ld    cnt,36  ; length of BST chain 0
0142   00CE 05                     nshfcp        ; test pattern for open faults
0143   00CF 000000000000    int2   .db   $00,$00,$00,$00,$00,$0c,$00,$03,$fc,$00,$fc,$03,$00,$0f
0144   00DE 06 04 01               jpe   theend  ; stop the test, if a net is found open, or s-a-1
0145   00E1 01                     tms1          ; BST chain 0, >> Update DR
0146   00E2 01                     tms1          ; BST chain 0, >> Sel. DR scan
0147   00E3 1B                     seltap1       ; switch to TAP 1
0148   00E4 02 00 12               ld    cnt,18  ; length of BST chain 1
0149   00E7 05                     nshfcp        ; TP for open faults, BST chain 1
0150   00E8 FF0000030003    int3   .db   $ff,$00,$00,$03,$00,$03,$00,$00,$03
0151   00F1 06 04 01               jpe   theend  ; stop the test, if a net is found open, or s-a-1
0152   00F4 01                     tms1          ; BST chain 1, >> Update DR
0153   00F5 01                     tms1          ; BST chain 1, >> Sel. DR scan
0154   00F6 13                     ssa1          ; SyncOutA goes high (BST test data applied)
```

```
0155   00F7 15                     wsa1          ; wait for SyncInpA to go high (external test data applied)
0156   00F8 00                     tms0          ; BST chain 1, >> Capture DR
0157   00F9 00                     tms0          ; BST chain 1, >> Shift DR
0158   00FA 1A                     seltap0       ; switch to TAP 0
0159   00FB 00                     tms0          ; BST chain 0, >> Capture DR
0160   00FC 00                     tms0          ; BST chain 0, >> Shift DR
0161   00FD 12                     ssa0          ; SyncOutA goes low (BST test results captured)
0162   00FE 14                     wsa0          ; wait for SyncInpA to go low (external test results captured)
0163   00FF 0C                     serflg4       ; select error flag 4 (for full BST interconnect short faults)
0164   0100 02 00 24               ld    cnt,36  ; length of BST chain 0
0165   0103 05                     nshfcp        ; test pattern for short faults
0166   0104 000000000000    int4  .db   $00,$00,$00,$00,$00,$00,$04,$0c,$03,$fc,$fc,$fc,$03,$03,$0f
0167   0113 06 04 01               jpe   theend  ; stop the test, if a short fault is found
0168   0116 01                     tms1          ; BST chain 0, >> Update DR
0169   0117 01                     tms1          ; BST chain 0, >> Sel. DR scan
0170   0118 1B                     seltap1       ; switch to TAP 1
0171   0119 02 00 12               ld    cnt,18  ; length of BST chain 1
0172   011C 05                     nshfcp        ; TP for short faults, BST chain 1
0173   011D 55FF00030303    int5  .db   $55,$ff,$00,$03,$03,$03,$00,$00,$03
0174   0126 06 04 01               jpe   theend  ; stop the test, if a short fault is found
0175   0129 01                     tms1          ; BST chain 1, >> Update DR
0176   012A 01                     tms1          ; BST chain 1, >> Sel. DR scan
0177   012B 13                     ssa1          ; SyncOutA goes high (BST test data applied)
0178   012C 15                     wsa1          ; wait for SyncInpA to go high (external test data applied)
0179   012D 00                     tms0          ; BST chain 1, >> Capture DR
0180   012E 00                     tms0          ; BST chain 1, >> Shift DR
0181   012F 1A                     seltap0       ; switch to TAP 0
0182   0130 00                     tms0          ; BST chain 0, >> Capture DR
0183   0131 00                     tms0          ; BST chain 0, >> Shift DR
0184   0132 12                     ssa0          ; SyncOutA goes low (BST test results captured)
0185   0133 14                     wsa0          ; wait for SyncInpA to go low (external test results captured)
0186   0134 02 00 24               ld    cnt,36  ; length of BST chain 0
0187   0137 05                     nshfcp        ; test pattern for short faults
0188   0138 000000000000    int6  .db   $00,$00,$00,$00,$00,$00,$08,$04,$03,$fc,$54,$fc,$03,$01,$0f
0189   0147 06 04 01               jpe   theend  ; stop the test, if a short fault is found
0190   014A 01                     tms1          ; BST chain 0, >> Update DR
0191   014B 01                     tms1          ; BST chain 0, >> Sel. DR scan
0192   014C 1B                     seltap1       ; switch to TAP 1
0193   014D 02 00 12               ld    cnt,18  ; length of BST chain 1
0194   0150 05                     nshfcp        ; TP for short faults, BST chain 1
0195   0151 AA5500030103    int7  .db   $aa,$55,$00,$03,$01,$03,$00,$00,$03
0196   015A 06 04 01               jpe   theend  ; stop the test, if a short fault is found
0197   015D 01                     tms1          ; BST chain 1, >> Update DR
0198   015E 01                     tms1          ; BST chain 1, >> Sel. DR scan
0199   015F 13                     ssa1          ; SyncOutA goes high (BST test data applied)
0200   0160 15                     wsa1          ; wait for SyncInpA to go high (external test data applied)
0201   0161 00                     tms0          ; BST chain 1, >> Capture DR
0202   0162 00                     tms0          ; BST chain 1, >> Shift DR
0203   0163 1A                     seltap0       ; switch to TAP 0
0204   0164 00                     tms0          ; BST chain 0, >> Capture DR
0205   0165 00                     tms0          ; BST chain 0, >> Shift DR
0206   0166 12                     ssa0          ; SyncOutA goes low (BST test results captured)
0207   0167 14                     wsa0          ; wait for SyncInpA to go low (external test results captured)
0208   0168 02 00 24               ld    cnt,36  ; length of BST chain 0
0209   016B 05                     nshfcp        ; test pattern for short faults
0210   016C 000000000000    int8  .db   $00,$00,$00,$00,$00,$00,$04,$08,$03,$fc,$54,$fc,$03,$01,$0f
0211   017B 06 04 01               jpe   theend  ; stop the test, if a short fault is found
0212   017E 01                     tms1          ; BST chain 0, >> Update DR
0213   017F 01                     tms1          ; BST chain 0, >> Sel. DR scan
0214   0180 1B                     seltap1       ; switch to TAP 1
0215   0181 02 00 12               ld    cnt,18  ; length of BST chain 1
0216   0184 05                     nshfcp        ; TP for short faults, BST chain 1
0217   0185 5AAA00030203    int9  .db   $5a,$aa,$00,$03,$02,$03,$00,$00,$03
0218   018E 06 04 01               jpe   theend  ; stop the test, if a short fault is found
0219   0191 01                     tms1          ; BST chain 1, >> Update DR
0220   0192 01                     tms1          ; BST chain 1, >> Sel. DR scan
0221   0193 13                     ssa1          ; SyncOutA goes high (BST test data applied)
0222   0194 15                     wsa1          ; wait for SyncInpA to go high (external test data applied)
0223   0195 00                     tms0          ; BST chain 1, >> Capture DR
0224   0196 00                     tms0          ; BST chain 1, >> Shift DR
0225   0197 1A                     seltap0       ; switch to TAP 0
0226   0198 00                     tms0          ; BST chain 0, >> Capture DR
0227   0199 00                     tms0          ; BST chain 0, >> Shift DR
0228   019A 12                     ssa0          ; SyncOutA goes low (BST test results captured)
0229   019B 14                     wsa0          ; wait for SyncInpA to go low (external test results captured)
0230   019C 02 00 24               ld    cnt,36  ; length of BST chain 0
0231   019F 05                     nshfcp        ; test pattern for short faults
0232   01A0 000000000000    int10 .db   $00,$00,$00,$00,$00,$00,$04,$04,$03,$fc,$a8,$fc,$03,$01,$0f
0233   01AF 06 04 01               jpe   theend  ; stop the test, if a short fault is found
0234   01B2 01                     tms1          ; BST chain 0, >> Update DR
0235   01B3 01                     tms1          ; BST chain 0, >> Sel. DR scan
0236   01B4 1B                     seltap1       ; switch to TAP 1
0237   01B5 02 00 12               ld    cnt,18  ; length of BST chain 1
0238   01B8 05                     nshfcp        ; TP for short faults, BST chain 1
0239   01B9 A55A00030103    int11 .db   $a5,$5a,$00,$03,$01,$03,$00,$00,$03
0240   01C2 06 04 01               jpe   theend  ; stop the test, if a short fault is found
0241   01C5 01                     tms1          ; BST chain 1, >> Update DR
0242   01C6 01                     tms1          ; BST chain 1, >> Sel. DR scan
0243   01C7 13                     ssa1          ; SyncOutA goes high (BST test data applied)
0244   01C8 15                     wsa1          ; wait for SyncInpA to go high (external test data applied)
0245   01C9 00                     tms0          ; BST chain 1, >> Capture DR
0246   01CA 00                     tms0          ; BST chain 1, >> Shift DR
0247   01CB 1A                     seltap0       ; switch to TAP 0
0248   01CC 00                     tms0          ; BST chain 0, >> Capture DR
0249   01CD 00                     tms0          ; BST chain 0, >> Shift DR
```

```
0250  01CE 12                        ssa0          ; SyncOutA goes low (BST test results captured)
0251  01CF 14                        wsa0          ; wait for SyncInpA to go low (external test results captured)
0252  01D0 02 00 24                  ld    cnt,36  ; length of BST chain 0
0253  01D3 05                        nshfcp        ; test pattern for short faults
0254  01D4 000000000000     int12    .db   $00,$00,$00,$00,$00,$00,$08,$04,$03,$fc,$68,$fc,$03,$02,$0f
0255  01E3 06 04 01                  jpe   theend  ; stop the test, if a short fault is found
0256  01E6 01                        tms1          ; BST chain 0, >> Update DR
0257  01E7 01                        tms1          ; BST chain 0, >> Sel. DR scan
0258  01E8 1B                        seltap1       ; switch to TAP 1
0259  01E9 02 00 12                  ld    cnt,18  ; length of BST chain 1
0260  01EC 05                        nshfcp        ; TP for short faults, BST chain 1
0261  01ED 66A500030103     int13    .db   $66,$a5,$00,$03,$01,$03,$00,$00,$03
0262  01F6 06 04 01                  jpe   theend  ; stop the test, if a short fault is found
0263  01F9 01                        tms1          ; BST chain 1, >> Update DR
0264  01FA 01                        tms1          ; BST chain 1, >> Sel. DR scan
0265  01FB 13                        ssa1          ; SyncOutA goes high (BST test data applied)
0266  01FC 15                        wsa1          ; wait for SyncInpA to go high (external test data applied)
0267  01FD 00                        tms0          ; BST chain 1, >> Capture DR
0268  01FE 00                        tms0          ; BST chain 1, >> Shift DR
0269  01FF 1A                        seltap0       ; switch to TAP 0
0270  0200 00                        tms0          ; BST chain 0, >> Capture DR
0271  0201 00                        tms0          ; BST chain 0, >> Shift DR
0272  0202 12                        ssa0          ; SyncOutA goes low (BST test results captured)
0273  0203 14                        wsa0          ; wait for SyncInpA to go low (external test results captured)
0274  0204 02 00 24                  ld    cnt,36  ; length of BST chain 0
0275  0207 05                        nshfcp        ; test pattern for short faults
0276  0208 000000000000     int14    .db   $00,$00,$00,$00,$00,$00,$08,$08,$03,$fc,$98,$fc,$03,$02,$0f
0277  0217 06 04 01                  jpe   theend  ; stop the test, if a short fault is found
0278  021A 01                        tms1          ; BST chain 0, >> Update DR
0279  021B 01                        tms1          ; BST chain 0, >> Sel. DR scan
0280  021C 1B                        seltap1       ; switch to TAP 1
0281  021D 02 00 12                  ld    cnt,18  ; length of BST chain 1
0282  0220 05                        nshfcp        ; TP for short faults, BST chain 1
0283  0221 996600030203     int15    .db   $99,$66,$00,$03,$02,$03,$00,$00,$03
0284  022A 06 04 01                  jpe   theend  ; stop the test, if a short fault is found
0285  022D 01                        tms1          ; BST chain 1, >> Update DR
0286  022E 01                        tms1          ; BST chain 1, >> Sel. DR scan
0287  022F 13                        ssa1          ; SyncOutA goes high (BST test data applied)
0288  0230 15                        wsa1          ; wait for SyncInpA to go high (external test data applied)
0289  0231 00                        tms0          ; BST chain 1, >> Capture DR
0290  0232 00                        tms0          ; BST chain 1, >> Shift DR
0291  0233 1A                        seltap0       ; switch to TAP 0
0292  0234 00                        tms0          ; BST chain 0, >> Capture DR
0293  0235 00                        tms0          ; BST chain 0, >> Shift DR
0294  0236 12                        ssa0          ; SyncOutA goes low (BST test results captured)
0295  0237 14                        wsa0          ; wait for SyncInpA to go low (external test results captured)
0296  0238 02 00 24                  ld    cnt,36  ; length of BST chain 0
0297  023B 05                        nshfcp        ; last test pattern for short faults, BS chain 0
0298  023C 000000000000     int16    .db   $00,$00,$00,$00,$00,$00,$08,$08,$03,$fc,$a4,$fc,$03,$02,$0f
0299  024B 06 04 01                  jpe   theend  ; stop the test, if a short fault is found
0300  024E 1B                        seltap1       ; switch to TAP 1
0301  024F 02 00 12                  ld    cnt,18  ; length of BST chain 1
0302  0252 05                        nshfcp        ; last TP for short faults, BST chain 1
0303  0253 999900030203     int17    .db   $99,$99,$00,$03,$02,$03,$00,$00,$03
0304  025C 06 04 01                  jpe   theend  ; stop the test, if a short fault is found
0305  025F 1A                        seltap0       ; switch to TAP 0
0306  0260
0307  0260                   ; Sequence for testing the cluster interconnects
0308  0260
0309  0260 0D                        serflg5       ; select error flag 5 (for cluster interconnect faults)
0310  0261 01                        tms1          ; >> Update DR, BS chain 0
0311  0262 01                        tms1          ; >> Select DR scan, BS chain 0
0312  0263 00                        tms0          ; >> Capture DR, BS chain 0
0313  0264 00                        tms0          ; >> Shift DR, BS chain 0
0314  0265 1B                        seltap1       ; switch to TAP 1
0315  0266 01                        tms1          ; >> Update DR, BS chain 1
0316  0267 01                        tms1          ; >> Select DR scan, BS chain 1
0317  0268 00                        tms0          ; >> Capture DR, BS chain 1
0318  0269 00                        tms0          ; >> Shift DR, BS chain 1
0319  026A 1A                        seltap0       ; switch to TAP 0
0320  026B 02 00 24                  ld    cnt,36  ; length of BST chain 0
0321  026E 04                        nshf          ; test pattern for cluster interconnect testing
0322  026F FAFF8C0300        clu0    .db   $fa,$ff,$8c,$03,$00
0323  0274 01                        tms1          ; BST chain 0, >> Update DR
0324  0275 01                        tms1          ; BST chain 0, >> Sel. DR scan
0325  0276 1B                        seltap1       ; switch to TAP 1
0326  0277 02 00 12                  ld    cnt,18  ; length of BST chain 1
0327  027A 04                        nshf          ; test pattern for cluster interconnect testing, BS chain 1
0328  027B FF F0 00          clu1    .db   $ff,$f0,$00
0329  027E 01                        tms1          ; BST chain 1, >> Update DR
0330  027F 01                        tms1          ; BST chain 1, >> Sel. DR scan
0331  0280 13                        ssa1          ; SyncOutA goes high (BST test data applied)
0332  0281 15                        wsa1          ; wait for SyncInpA to go high (external test data applied)
0333  0282 00                        tms0          ; BST chain 1, >> Capture DR
0334  0283 00                        tms0          ; BST chain 1, >> Shift DR
0335  0284 1A                        seltap0       ; switch to TAP 0
0336  0285 00                        tms0          ; BST chain 0, >> Capture DR
0337  0286 00                        tms0          ; BST chain 0, >> Shift DR
0338  0287 12                        ssa0          ; SyncOutA goes low (BST test results captured)
0339  0288 14                        wsa0          ; wait for SyncInpA to go low (external test results captured)
0340  0289 02 00 24                  ld    cnt,36  ; length of BST chain 0
0341  028C 05                        nshfcp        ; test pattern for cluster interconnect testing
0342  028D 11FF00FFF1FF      clu2    .db   $11,$ff,$00,$ff,$f1,$ff,$4c,$ff,$00,$03,$ff,$fc,$00,$0f,$03
0343  029C 06 04 01                  jpe   theend  ; stop the test, if a cluster fault is detected
0344  029F 01                        tms1          ; BST chain 0, >> Update DR
```

```
0345   02A0 01                        tms1          ; BST chain 0, >> Sel. DR scan
0346   02A1 1B                        seltap1       ; switch to TAP 1
0347   02A2 02 00 12                  ld    cnt,18  ; length of BST chain 1
0348   02A5 05                        nshfcp        ; test pattern for cluster interconnect testing, BS chain 1
0349   02A6 FFFF00F0F3F3    clu3      .db   $ff,$ff,$00,$f0,$f3,$f3,$00,$00,$03
0350   02AF 06 04 01                  jpe   theend  ; stop the test, if a cluster fault is detected
0351   02B2 01                        tms1          ; BST chain 1, >> Update DR
0352   02B3 01                        tms1          ; BST chain 1, >> Sel. DR scan
0353   02B4 13                        ssa1          ; SyncOutA goes high (BST test data applied)
0354   02B5 15                        wsa1          ; wait for SyncInpA to go high (external test data applied)
0355   02B6 00                        tms0          ; BST chain 1, >> Capture DR
0356   02B7 00                        tms0          ; BST chain 1, >> Shift DR
0357   02B8 1A                        seltap0       ; switch to TAP 0
0358   02B9 00                        tms0          ; BST chain 0, >> Capture DR
0359   02BA 00                        tms0          ; BST chain 0, >> Shift DR
0360   02BB 12                        ssa0          ; SyncOutA goes low (BST test results captured)
0361   02BC 14                        wsa0          ; wait for SyncInpA to go low (external test results captured)
0362   02BD 02 00 24                  ld    cnt,36  ; length of BST chain 0
0363   02C0 05                        nshfcp        ; test pattern for cluster interconnect testing
0364   02C1 37FF00FFE0FF    clu4      .db   $37,$ff,$00,$ff,$e0,$ff,$3c,$ff,$00,$00,$ff,$fc,$00,$0f,$03
0365   02D0 06 04 01                  jpe   theend  ; stop the test, if a cluster fault is detected
0366   02D3 01                        tms1          ; BST chain 0, >> Update DR
0367   02D4 01                        tms1          ; BST chain 0, >> Sel. DR scan
0368   02D5 1B                        seltap1       ; switch to TAP 1
0369   02D6 02 00 12                  ld    cnt,18  ; length of BST chain 1
0370   02D9 05                        nshfcp        ; test pattern for cluster interconnect testing, BS chain 1
0371   02DA FFFF00F003F3    clu5      .db   $ff,$ff,$00,$f0,$03,$f3,$00,$00,$03
0372   02E3 06 04 01                  jpe   theend  ; stop the test, if a cluster fault is detected
0373   02E6 01                        tms1          ; BST chain 1, >> Update DR
0374   02E7 01                        tms1          ; BST chain 1, >> Sel. DR scan
0375   02E8 13                        ssa1          ; SyncOutA goes high (BST test data applied)
0376   02E9 15                        wsa1          ; wait for SyncInpA to go high (external test data applied)
0377   02EA 00                        tms0          ; BST chain 1, >> Capture DR
0378   02EB 00                        tms0          ; BST chain 1, >> Shift DR
0379   02EC 1A                        seltap0       ; switch to TAP 0
0380   02ED 00                        tms0          ; BST chain 0, >> Capture DR
0381   02EE 00                        tms0          ; BST chain 0, >> Shift DR
0382   02EF 12                        ssa0          ; SyncOutA goes low (BST test results captured)
0383   02F0 14                        wsa0          ; wait for SyncInpA to go low (external test results captured)
0384   02F1 02 00 24                  ld    cnt,36  ; length of BST chain 0
0385   02F4 05                        nshfcp        ; test pattern for cluster interconnect testing
0386   02F5 D5FF00FF78FF    clu6      .db   $d5,$ff,$00,$ff,$78,$ff,$2c,$ff,$00,$01,$ff,$fc,$00,$0f,$03
0387   0304 06 04 01                  jpe   theend  ; stop the test, if a cluster fault is detected
0388   0307 01                        tms1          ; BST chain 0, >> Update DR
0389   0308 01                        tms1          ; BST chain 0, >> Sel. DR scan
0390   0309 1B                        seltap1       ; switch to TAP 1
0391   030A 02 00 12                  ld    cnt,18  ; length of BST chain 1
0392   030D 05                        nshfcp        ; test pattern for cluster interconnect testing, BS chain 1
0393   030E FFFF00F003F3    clu7      .db   $ff,$ff,$00,$f0,$03,$f3,$00,$00,$03
0394   0317 06 04 01                  jpe   theend  ; stop the test, if a cluster fault is detected
0395   031A 01                        tms1          ; BST chain 1, >> Update DR
0396   031B 01                        tms1          ; BST chain 1, >> Sel. DR scan
0397   031C 13                        ssa1          ; SyncOutA goes high (BST test data applied)
0398   031D 15                        wsa1          ; wait for SyncInpA to go high (external test data applied)
0399   031E 00                        tms0          ; BST chain 1, >> Capture DR
0400   031F 00                        tms0          ; BST chain 1, >> Shift DR
0401   0320 1A                        seltap0       ; switch to TAP 0
0402   0321 00                        tms0          ; BST chain 0, >> Capture DR
0403   0322 00                        tms0          ; BST chain 0, >> Shift DR
0404   0323 12                        ssa0          ; SyncOutA goes low (BST test results captured)
0405   0324 14                        wsa0          ; wait for SyncInpA to go low (external test results captured)
0406   0325 02 00 24                  ld    cnt,36  ; length of BST chain 0
0407   0328 05                        nshfcp        ; test pattern for cluster interconnect testing
0408   0329 22FF00FFD2FF    clu8      .db   $22,$ff,$00,$ff,$d2,$ff,$2c,$ff,$00,$02,$ff,$fc,$00,$0f,$03
0409   0338 06 04 01                  jpe   theend  ; stop the test, if a cluster fault is detected
0410   033B 01                        tms1          ; BST chain 0, >> Update DR
0411   033C 01                        tms1          ; BST chain 0, >> Sel. DR scan
0412   033D 1B                        seltap1       ; switch to TAP 1
0413   033E 02 00 12                  ld    cnt,18  ; length of BST chain 1
0414   0341 05                        nshfcp        ; test pattern for cluster interconnect testing, BS chain 1
0415   0342 FFFF00F0F3F3    clu9      .db   $ff,$ff,$00,$f0,$f3,$f3,$00,$00,$03
0416   034B 06 04 01                  jpe   theend  ; stop the test, if a cluster fault is detected
0417   034E 01                        tms1          ; BST chain 1, >> Update DR
0418   034F 01                        tms1          ; BST chain 1, >> Sel. DR scan
0419   0350 13                        ssa1          ; SyncOutA goes high (BST test data applied)
0420   0351 15                        wsa1          ; wait for SyncInpA to go high (external test data applied)
0421   0352 00                        tms0          ; BST chain 1, >> Capture DR
0422   0353 00                        tms0          ; BST chain 1, >> Shift DR
0423   0354 1A                        seltap0       ; switch to TAP 0
0424   0355 00                        tms0          ; BST chain 0, >> Capture DR
0425   0356 00                        tms0          ; BST chain 0, >> Shift DR
0426   0357 12                        ssa0          ; SyncOutA goes low (BST test results captured)
0427   0358 14                        wsa0          ; wait for SyncInpA to go low (external test results captured)
0428   0359 02 00 24                  ld    cnt,36  ; length of BST chain 0
0429   035C 05                        nshfcp        ; last test pattern for cluster testing, BS chain 0
0430   035D 22FF00FFB4FF    clu10     .db   $22,$ff,$00,$ff,$b4,$ff,$2c,$ff,$00,$02,$ff,$fc,$00,$0f,$03
0431   036C 06 04 01                  jpe   theend  ; stop the test, if a cluster fault is detected
0432   036F 1B                        seltap1       ; switch to TAP 1
0433   0370 02 00 12                  ld    cnt,18  ; length of BST chain 1
0434   0373 05                        nshfcp        ; last TP for cluster interconnect testing, BST chain 1
0435   0374 FFFF00F003F3    clu11     .db   $ff,$ff,$00,$f0,$03,$f3,$00,$00,$03
0436   037D 06 04 01                  jpe   theend  ; stop the test, if a cluster fault is detected
0437   0380 1A                        seltap0       ; switch to TAP 0
0438   0381
0439   0381                 ; Sequence for internal test of components
```

```
0440   0381
0441   0381 0F                        serflg7       ; select error flag 7 (for faults detected during internal test)
0442   0382 1A                        seltap0       ; switch to BS chain 0
0443   0383 01                        tms1          ; >> Update DR, BS chain 0
0444   0384 01                        tms1          ; >> Select DR scan, BS chain 0
0445   0385 01                        tms1          ; >> Select IR scan, BS chain 0
0446   0386 00                        tms0          ; >> Capture IR, BS chain 0
0447   0387 00                        tms0          ; >> Shift IR, BS chain 0
0448   0388 02 00 10                  ld    cnt,16  ; length of the IRs in scan chain 0
0449   038B 04                        nshf          ; shift in the intest/bypass opcodes
0450   038C 00 00             com0    .db   $00,$00
0451   038E 01                        tms1          ; >> Update IR, BS chain 0
0452   038F 01                        tms1          ; >> Select DR scan, BS chain 0
0453   0390 00                        tms0          ; >> Capture DR, BS chain 0
0454   0391 00                        tms0          ; >> Shift DR, BS chain 0
0455   0392 02 00 24                  ld    cnt,36  ; length of scan chain 0
0456   0395 04                        nshf          ; shift the 1st TP for internal test (no compare)
0457   0396 FF55FF570D        com1    .db   $ff,$55,$ff,$57,$0d
0458   039B 01                        tms1          ; >> Update DR, BS chain 0
0459   039C 01                        tms1          ; >> Select DR scan, BS chain 0
0460   039D 00                        tms0          ; >> Capture DR, BS chain 0
0461   039E 00                        tms0          ; >> Shift DR, BS chain 0
0462   039F 02 00 24                  ld    cnt,36  ; length of scan chain 0
0463   03A2 05                        nshfcp        ; test pattern for internal test of components
0464   03A3 0055FFAAFF00      com2    .db   $00,$55,$ff,$aa,$ff,$00,$03,$57,$fc,$a8,$fd,$03,$0e,$0f,$00
0465   03B2 01                        tms1          ; >> Update DR, BS chain 0
0466   03B3 01                        tms1          ; >> Select DR scan, BS chain 0
0467   03B4 00                        tms0          ; >> Capture DR, BS chain 0
0468   03B5 00                        tms0          ; >> Shift DR, BS chain 0
0469   03B6 02 00 24                  ld    cnt,36  ; length of scan chain 0
0470   03B9 05                        nshfcp        ; last test pattern for internal test of components
0471   03BA 00AAFFAA0000      com3    .db   $00,$aa,$ff,$aa,$00,$00,$03,$a8,$fc,$a8,$02,$03,$0e,$00,$00
0472   03C9 1B                        seltap1       ; switch to BS chain 1
0473   03CA 01                        tms1          ; >> Update DR, BS chain 1
0474   03CB 01                        tms1          ; >> Select DR scan, BS chain 1
0475   03CC 01                        tms1          ; >> Select IR scan, BS chain 1
0476   03CD 00                        tms0          ; >> Capture IR, BS chain 1
0477   03CE 00                        tms0          ; >> Shift IR, BS chain 1
0478   03CF 02 00 08                  ld    cnt,8   ; length of the IRs in scan chain 1
0479   03D2 04                        nshf          ; shift in the intest/bypass opcodes
0480   03D3 00                com4    .db   $00
0481   03D4 01                        tms1          ; >> Update IR, BS chain 1
0482   03D5 01                        tms1          ; >> Select DR scan, BS chain 1
0483   03D6 00                        tms0          ; >> Capture DR, BS chain 1
0484   03D7 00                        tms0          ; >> Shift DR, BS chain 1
0485   03D8 02 00 12                  ld    cnt,18  ; length of scan chain 1
0486   03DB 04                        nshf          ; shift the 1st TP for internal test (no compare)
0487   03DC FF 55 03          com5    .db   $ff,$55,$03
0488   03DF 01                        tms1          ; >> Update DR, BS chain 1
0489   03E0 01                        tms1          ; >> Select DR scan, BS chain 1
0490   03E1 00                        tms0          ; >> Capture DR, BS chain 1
0491   03E2 00                        tms0          ; >> Shift DR, BS chain 1
0492   03E3 02 00 12                  ld    cnt,18  ; length of scan chain 1
0493   03E6 05                        nshfcp        ; test pattern for internal test of components
0494   03E7 0055FFAAFF00      com6    .db   $00,$55,$ff,$aa,$ff,$00,$03,$03,$00
0495   03F0 01                        tms1          ; >> Update DR, BS chain 1
0496   03F1 01                        tms1          ; >> Select DR scan, BS chain 1
0497   03F2 00                        tms0          ; >> Capture DR, BS chain 1
0498   03F3 00                        tms0          ; >> Shift DR, BS chain 1
0499   03F4 02 00 12                  ld    cnt,18  ; length of scan chain 1
0500   03F7 05                        nshfcp        ; last test pattern for internal test of components
0501   03F8 00AAFFAA0000      com7    .db   $00,$aa,$ff,$aa,$00,$00,$03,$00,$00
0502   0401
0503   0401                   ;       End of test program.
0504   0401
0505   0401 1A                theend  seltap0       ; (switch to) TAP 0
0506   0402 01                        tms1
0507   0403 01                        tms1
0508   0404 01                        tms1
0509   0405 01                        tms1
0510   0406 01                        tms1          ; reset sequence for TAP 0
0511   0407 1B                        seltap1       ; switch to TAP 1
0512   0408 01                        tms1
0513   0409 01                        tms1
0514   040A 01                        tms1
0515   040B 01                        tms1
0516   040C 01                        tms1          ; reset sequence for TAP 1
0517   040D 10                        halt          ; this is the end
0518   040E
0519   040E                           .end
0520   040E
0521   040E
0522   040E
0523   8000                   primary .org  8000h
0524   8000
0525   8000                   ; pattern number 0
0526   8000 00                pi0     .db   $00     ; primary inputs
0527   8001 FF                cs0     .db   $ff     ; control signal
0528   8002 00                po0     .db   $00     ; expected values at primary outputs
0529   8003 FF                pom0    .db   $ff     ; primary outputs mask
0530   8004                   ; pattern number 1
0531   8004 FF                pi1     .db   $ff     ; primary inputs
0532   8005 FF                cs1     .db   $ff     ; control signal
0533   8006 FF                po1     .db   $ff     ; expected values at primary outputs
0534   8007 FF                pom1    .db   $ff     ; primary outputs mask
```

```
0535   8008                ; pattern number 2
0536   8008 AA             pi2      .db   $aa      ; primary inputs
0537   8009 FF             cs2      .db   $ff      ; control signal
0538   800A AA             po2      .db   $aa      ; expected values at primary outputs
0539   800B FF             pom2     .db   $ff      ; primary outputs mask
0540   800C                ; pattern number 3
0541   800C AA             pi3      .db   $aa      ; primary inputs
0542   800D FF             cs3      .db   $ff      ; control signal
0543   800E 55             po3      .db   $55      ; expected values at primary outputs
0544   800F FF             pom3     .db   $ff      ; primary outputs mask
0545   8010                ; pattern number 4
0546   8010 56             pi4      .db   $56      ; primary inputs
0547   8011 FF             cs4      .db   $ff      ; control signal
0548   8012 5A             po4      .db   $5a      ; expected values at primary outputs
0549   8013 FF             pom4     .db   $ff      ; primary outputs mask
0550   8014                ; pattern number 5
0551   8014 59             pi5      .db   $59      ; primary inputs
0552   8015 FF             cs5      .db   $ff      ; control signal
0553   8016 A5             po5      .db   $a5      ; expected values at primary outputs
0554   8017 FF             pom5     .db   $ff      ; primary outputs mask
0555   8018                ; pattern number 6
0556   8018 65             pi6      .db   $65      ; primary inputs
0557   8019 FF             cs6      .db   $ff      ; control signal
0558   801A 66             po6      .db   $66      ; expected values at primary outputs
0559   801B FF             pom6     .db   $ff      ; primary outputs mask
0560   801C                ; pattern number 7
0561   801C 95             pi7      .db   $95      ; primary inputs
0562   801D FF             cs7      .db   $ff      ; control signal
0563   801E 99             po7      .db   $99      ; expected values at primary outputs
0564   801F FF             pom7     .db   $ff      ; primary outputs mask
0565   8020                ; pattern number 8
0566   8020 FF             pi8      .db   $ff      ; primary inputs
0567   8021 FF             cs8      .db   $ff      ; control signal
0568   8022 FF             po8      .db   $ff      ; expected values at primary outputs
0569   8023 FF             pom8     .db   $ff      ; primary outputs mask
0570   8024                ; pattern number 9
0571   8024 FF             pi9      .db   $ff      ; primary inputs
0572   8025 FF             cs9      .db   $ff      ; control signal
0573   8026 FF             po9      .db   $ff      ; expected values at primary outputs
0574   8027 FF             pom9     .db   $ff      ; primary outputs mask
0575   8028                ; pattern number 10
0576   8028 FF             pi10     .db   $ff      ; primary inputs
0577   8029 FF             cs10     .db   $ff      ; control signal
0578   802A FF             po10     .db   $ff      ; expected values at primary outputs
0579   802B FF             pom10    .db   $ff      ; primary outputs mask
0580   802C                ; pattern number 11
0581   802C FF             pi11     .db   $ff      ; primary inputs
0582   802D FF             cs11     .db   $ff      ; control signal
0583   802E FF             po11     .db   $ff      ; expected values at primary outputs
0584   802F FF             pom11    .db   $ff      ; primary outputs mask
0585   8030                ; pattern number 12
0586   8030 FF             pi12     .db   $ff      ; primary inputs
0587   8031 FF             cs12     .db   $ff      ; control signal
0588   8032 FF             po12     .db   $ff      ; expected values at primary outputs
0589   8033 FF             pom12    .db   $ff      ; primary outputs mask
0590   8034                .end
0591   8034

tasm: Number of errors = 0
```