

1.98

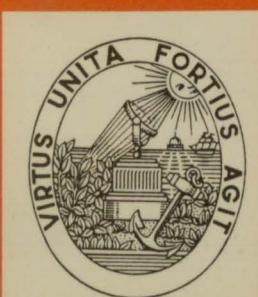
UNIVERSIDADE DO PORTO  
**FACULDADE DE ENGENHARIA**  
DEPARTAMENTO DE ENGENHARIA ELECTROTECNICA E DE COMPUTADORES

CONTROLADOR DE TESTE PARA CIRCUITOS INTEGRADOS  
COM "BST"

RELATORIO DO ESTAGIO REALIZADO NO AMBITO DO PROGRAMA  
PRODEP MEDIDA 4.3 - AREA 7.4

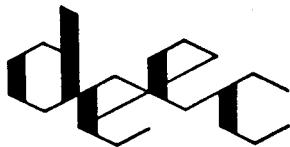
JOAO JORGE DA SILVA RESENDE MOREIRA PEIXOTO

JUNHO DE 1993



Universidade do Porto  
Faculdade de Engenharia  
Biblioteca 4

Nº  
CDU 621.3(042-3) / LEEC 1992/P-1  
Data 01/10/2009



FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Departamento de Engenharia Electrotécnica e de Computadores

Rua dos Bragas, 4099 Porto Codex, PORTUGAL

Telef. 351-2-317105/107/412/457 · Telex 27323 FEUP P · Telefax 351-2-319280

### PARECER

João Jorge da Silva Resende Moreira Peixoto licenciou-se em Engenharia Electrotécnica e de Computadores pela FEUP, em Setembro de 1992, com média final de 12 valores.

O trabalho realizado no âmbito do estágio PRODEP, medida 4.3, área 7.4, encontra-se descrito no seu relatório de conclusão de estágio, com o título "Controlador de teste para cartas de circuito impresso com BST".

O Licenciado João Jorge Peixoto levou a cabo com empenho todas as tarefas que lhe foram confiadas, tendo atingido todos os objectivos que lhe foram apresentados. Penso ser de interesse referir que o trabalho realizado se enquadrou no âmbito de um projecto patrocinado pela Junta Nacional de Investigação Científica e Tecnológica (JNICT), sob o contracto nº PMCT/C/TIT/937/90.

Na qualidade de orientador pela Faculdade de Engenharia, e tendo em consideração os aspectos referidos, posso afirmar que o trabalho realizado no âmbito deste estágio foi francamente positivo, quer em termos dos resultados alcançados, quer no que respeita à valorização profissional que proporcionou ao Licenciado João Jorge Peixoto.

José M. Martins Ferreira  
Professor Auxiliar da FEUP

*José M. Martins Ferreira*



**PARECER**

O Licenciado João Jorge da Silva Resende Moreira Peixoto realizou no INESC, no âmbito do programa PRODEP, medida 4.3, área 7.4, um estágio cujo trabalho se encontra descrito no relatório "Controlador para cartas de circuito impresso com BST".

Durante a execução do conjunto de tarefas que lhe foram atribuídas, o Licenciado João Jorge mostrou possuir um elevado sentido de responsabilidade, tendo atingido a totalidade dos objectivos que lhe foram propostos. Mostrou ter conhecimentos adequados, tanto na área técnica a que correspondeu o trabalho como na selecção e implementação das metodologias adequadas aos vários problemas que lhe se depararam.

Face à qualidade do trabalho produzido e à competência profissional demonstrada, cumpre-me informar que o estágio realizado se revestiu de grande interesse, quer para a instituição como para o Licenciado João Jorge.

José Alberto P. Machado da Silva

Investigador do INESC

# **CONTROLADOR PARA CARTAS DE CIRCUITO IMPRESSO COM "BST"**

**Subprograma 4 do PRODEP  
Relatório de Estágio**

**Junho de 1993**

**Supervisores :**

José Manuel Martins Ferreira ..... Professor Auxiliar da FEUP  
José Alberto Peixoto Machado Silva ..... Investigador do INESC

**Executou :**

João Jorge da Silva Resende Moreira Peixoto ..... Licenciado em Engenharia

# **SUMÁRIO**

Este documento descreve a implementação de um Controlador de Teste para Cartas de Circuito Impresso com "BST". O Controlador suporta directamente um conjunto de instruções que implementam as operações necessárias para o controlo da infraestrutura "BST" de uma CCI. É descrita igualmente uma aplicação prática baseada neste controlador.

Corresponde ao relatório a apresentar no final do estágio ao abrigo do subprograma 4 do PRODEP.

**INESC - Porto, Junho de 1993**

**João Jorge da Silva Resende Moreira Peixoto**

**José Manuel Martins Ferreira**

**José Alberto Peixoto Machado Silva**

# ABREVIATURAS PRINCIPAIS

**BIST:** *Built-In Self-Test*

**BS:** *Boundary Scan*

**BST:** *Boundary Scan Test*

**CAD:** *Computer Aided Design*

**CCI:** Carta de Circuito Impresso

**CI:** Circuito Integrado

**EPLD:** *Erasable Programmable Logic Device*

**E/S:** Entrada / Saída

**ESPRIT:** *European Strategic Programme for Research and Development in Information Technology*

**GAPT:** Geração Automática do Programa de Teste

**IEEE:** *Institute of Electrical and Electronics Engineers*

**PLD:** *Programmable Logic Device*

**Std:** *Standard*

**TAP:** *Test Access Port*

**TCK:** *Test Clock*

**TDI:** *Test Data Input*

**TDO:** *Test Data Output*:

**TMS:** *Test Mode Select*

**TRST:** *Test Reset*

# INDÍCE

1.INTRODUÇÃO .....	1
2.DESCRIÇÃO GERAL.....	2
2.1.Especificação do Controlador de Teste.....	2
2.1.1.Caracterização global.....	9
2.1.2.Arquitectura do controlador de teste .....	10
2.1.3.Especificação das instruções suportadas.....	12
2.1.3.1.Definição do conjunto de instruções.....	12
2.1.3.2.Caracterização individual .....	14
2.1.3.2.1. Instruções para o controlo da infraestrutura BST.....	14
2.1.3.2.2. Instruções para implementação do protocolo de sincronismo com o exterior. .	22
2.1.3.2.3. Instruções para o controlo de recursos internos, e do fluxo do programa.....	24
2.1.3.2.4. Requisitos em memória, e tempo de execução .....	27
3.EXEMPLO DE APLICAÇÃO .....	29
3.1.Descrição da CCI de demonstração .....	29
3.2.Informação requerida pelo GAPT .....	29
3.2.1.Teste da infraestrutura BST .....	32
3.2.2.Teste de ligações.....	32
3.2.2.1.Ligações BST .....	32
3.2.2.2.Grupos de componentes não BST .....	34
3.2.3.Teste de componentes.....	42
3.3.Resultados produzidos pelo GAPT .....	43
3.3.1.Teste de curto-circuitos entre ligações BST.....	43
3.3.2.Congunto de vectores de teste.....	43
3.3.3.Programa de teste.....	53
4.CONCLUSÃO .....	60
5.REFERÊNCIAS.....	61
6.ANEXO .....	62

# 1. INTRODUÇÃO

Este documento descreve uma solução desenvolvida com o objectivo de permitir o controlo de uma infraestrutura *Boundary Scan* existente numa carta de circuito impresso (CCI). Este controlador suporta directamente um conjunto de instruções que implementam as operações necessárias para o controlo da infraestrutura BST (incluindo as que se referem ao protocolo de sincronismo com um equipamento de teste exterior). O programa para este controlador de teste é gerado por uma ferramenta de geração automática de programas de teste (GAPT) descrita em (Tavares et al, 93), e que produz o código a ser executado.

O controlador de teste suporta a existência de duas cadeias BST na CCI a testar, acedidas e controladas independentemente, e uma saída do tipo Sim/Não que indica para o exterior o resultado do teste.

Todo o sistema foi implementado numa PLD (*Programmable Logic Device*) de média complexidade. Este componente, que passará no seguimento a ser designado por "Controlador de Teste", utiliza um encapsulamento PLCC com 68 pinos, cuja funcionalidade foi definida como se ilustra na figura 1.1.

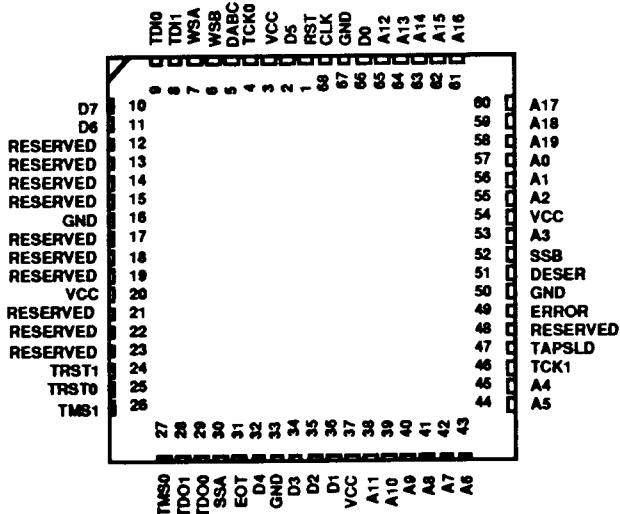


Figura 1.1: Controlador das E/S Digitais: configuração de pinos.

## 2. DESCRIÇÃO GERAL

Neste capítulo é realizada a especificação do controlador com base numa identificação das operações elementares necessárias ao controlo da infraestrutura BST, por forma a se proporcionar os mecanismos básicos que permitam implementar as etapas definidas no protocolo de teste de CCI's com BST. Numa segunda parte é apresentada a arquitectura interna do controlador de teste. A ideia de implementar este controlador de teste numa PLD revelou-se simples e eficaz. Internamente este dispositivo possui todos os blocos necessários, conforme a norma IEEE 1149.1, sendo cada um deles analisado em pormenor nos pontos seguintes.

### 2.1. Especificação do Controlador de Teste

Os objectivos a atingir consistem fundamentalmente em explorar todas as potencialidades proporcionadas pela infraestrutura BST para o teste de CCIs. A apresentação que é efectuada nesta secção segue de perto a que tem lugar em (Ferreira, 92).

As operações elementares requeridas determinam o conjunto de instruções a ser suportado pelo controlador de teste, 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 do controlador de teste.

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.

#### I) Teste da Infraestrutura BST

- Inicialização da infraestrutura BST

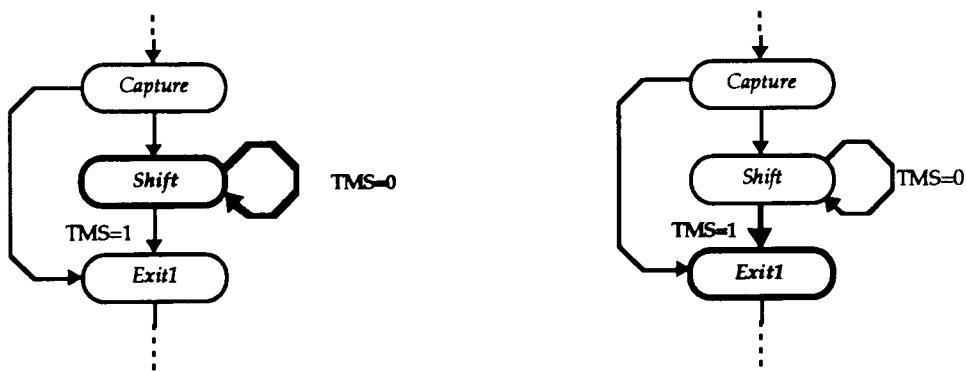
A concretização desta etapa será possível pela existência de uma operação elementar que provoque um impulso a 0 na linha /TRST associada à cadeia BS que se pretenda inicializar. Repare-se que esta etapa pode alternativamente ser garantida pela aplicação de cinco impulsos no relógio de teste (TCK) do 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 "bit's" 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 "bit's" 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 "bit's" através da cadeia BS 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 "bit's", passando a 1 para o deslocamento do último "bit", tal como se ilustra na figura 2.1. 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 "bit's" a deslocar, já que poderá não ser conhecido o valor esperado para alguns destes "bit's".

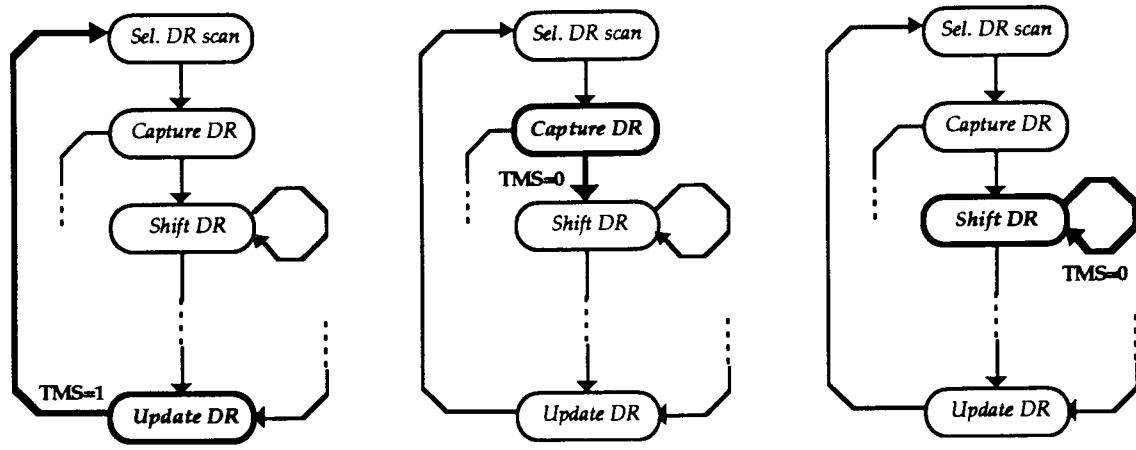


(a) Deslocamento dos primeiros (N-1) "bit's".      (b) Deslocamento do último "bit".

**Figura 2.1 : Estados do controlador do TAP no deslocamento de N "bit's" 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 "bit's" 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 2.1, e garantindo a presença na linha TDI da CCI de cada novo "bit" a deslocar, e o controlo da linha TMS.



(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.

**Figura 2.2 : Estados do controlador do TAP para o teste de ligações.**

## 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

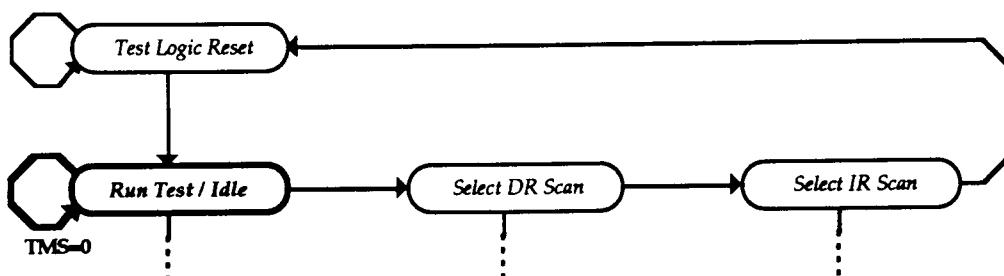
deslocamento para o exterior das respostas capturadas para o vector anterior, tal como se ilustra na figura 2.2.

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 registo 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 2.3. 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 (*Runbis*), tal como definida em (IEEE Std 1149.1, 90, p. 7-20).



**Figura 2.3 : Estado do controlador do TAP, durante a execução do auto-teste em componentes BST.**

**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 "bit's" para o interior da cadeia BST, sem comparar os "bit's" deslocados para o exterior. A linha TMS deverá ser mantida em 0 durante o deslocamento dos primeiros N-1 "bit's", e colocada em 1 para o deslocamento do último "bit".
- Deslocar N "bit's" para o interior da cadeia BST, comparando os "bit's" deslocados para o exterior com o seu valor esperado, e fazendo uso de uma máscara que indique quais os "bit's" cuja comparação é relevante. A linha TMS deverá ser mantida em 0 durante o deslocamento dos primeiros N-1 "bit's", 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 2.1 : Síntese das operações para o controlo da infraestrutura BST.**

Finalmente, deve estar também disponível uma operação elementar que permita seleccionar qual das duas cadeias BST suportadas pelo controlador de teste 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.

O quadro 2.1 apresenta uma síntese das operações elementares identificadas para permitir o controlo da infraestrutura BST.

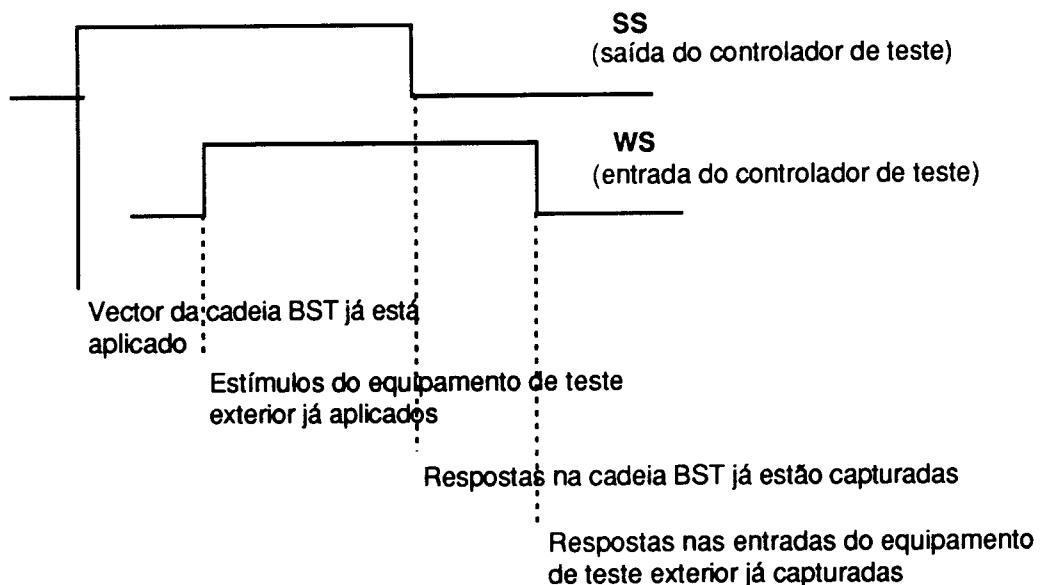
O controlador de teste suporta ainda um protocolo que possibilita a sincronização do nosso sistema de teste com um sistema de teste exterior. Este protocolo integra quatro operações principais, que se podem especificar da seguinte maneira:

- i) Aplicação do conjunto de estímulos através da infraestrutura BST.
- ii) Aplicação dos estímulos por parte do sistema de teste exterior.
- iii) Captura das respostas através da infraestrutura BST.
- iv) Captura das respostas nas entradas do sistema de teste exterior.

Este protocolo pode ser implementado através de um *handshaking* assíncrono com duas linhas (Bagge et al., 89), tal como se exemplifica na figura 2.4.

O controlador de teste compara as respostas capturadas com os seus valores esperados, e memoriza a ocorrência de um erro. Repare-se que existe interesse em que o armazenamento das respostas capturadas através da infraestrutura BST não integre o conjunto de tarefas a realizar por este controlador de teste, uma vez que este facto permitirá um acesso unidireccional à sua memória privativa. Com efeito, a restrição do acesso só a operações de leitura tornará mais fácil que toda a arquitectura interna deste bloco seja integrada num único componente, o que possibilita uma alternativa altamente compacta para a inclusão de auto-teste em CCIs que disponham de BST. A tarefa de deserialização e armazenamento das respostas capturadas através da infraestrutura BST terá deste modo de ser implementada noutro bloco.

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.



**Figura 2.4 : Protocolo de sincronismo com um equipamento de teste exterior.**

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 2.2.(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 2.2.(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 conjunto de estímulos (destinada a preceder a operação elementar que provoca a passagem para o estado *Capture-DR* (*Capture-IR*), na figura 2.2.(b)), ou quando é que este equipamento exterior já capturou as respostas a esses estímulos (destinada a preceder o deslocamento de um novo vector para o interior da cadeia BST, ilustrado na figura 2.2.(c)).

As operações elementares para a implementação deste protocolo de sincronismo estão identificadas no quadro 2.2.

#### Operações elementares para o protocolo de sincronismo com equipamentos exteriores

- Colocar na saída de sincronismo o valor pretendido.
- Reter o controlador de teste durante um número indefinido de ciclos de relógio (relógio do controlador de teste), até que a entrada de sincronismo se encontre no valor pretendido.

**Quadro 2.2 :** 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 de teste, e o fluxo do programa de teste.

As operações elementares destinadas a permitir o deslocamento de N "bit's", com ou sem comparação, ou a aplicação de N impulsos no relógio de teste (TCK), descritas no quadro 2.1, evidenciam a necessidade de existir pelo menos um contador interno ao controlador de teste, 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 existência de uma *flag* destinada 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. 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 2.3 apresenta uma síntese do último grupo de operações elementares identificadas, destinadas ao controlo dos recursos internos do controlador de teste, 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).
- Permitir um salto condicional no fluxo do programa de teste, de acordo com o estado da flag de erro.
- Concluir a execução do programa de teste.

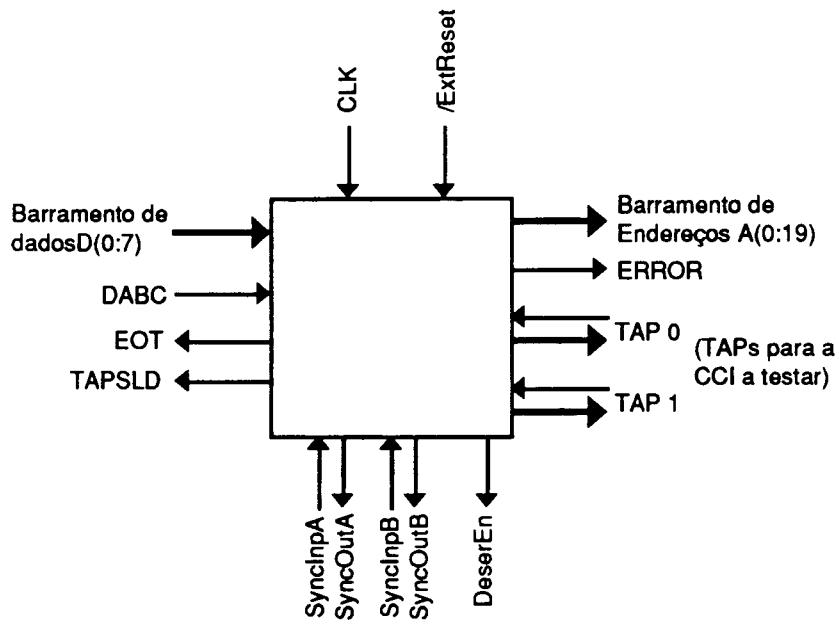
**Quadro 2.3 : Síntese das operações para o controlo de recursos internos, e do fluxo do programa de teste.**

A especificação do controlador de teste 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.

#### **2.1.1. Caracterização global**

A definição da largura para o barramento de dados do controlador de teste teve em conta vários factores que devidamente ponderados levaram à opção por um barramento de dados com 8 "bit's".

A determinação da largura recomendável para o barramento de endereços foi mais problemática, sobretudo pela falta de uma experiência de projecto neste domínio. A opção recaiu neste caso num barramento de 20 "bit's" o que permite endereçar directamente 1 "MByte" de memória de programa de teste externa.



**Figura 2.5 : O conjunto de pinos do controlador de teste.**

A figura 2.5 mostra o conjunto de pinos que fazem parte do controlador de teste, e cuja descrição se encontra sintetizada no quadro 2.4.

### 2.1.2. Arquitectura do controlador de teste

Esta parte apresenta uma arquitectura para o controlador de teste, 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 2.1 a 2.3, tal como se apresenta a seguir:

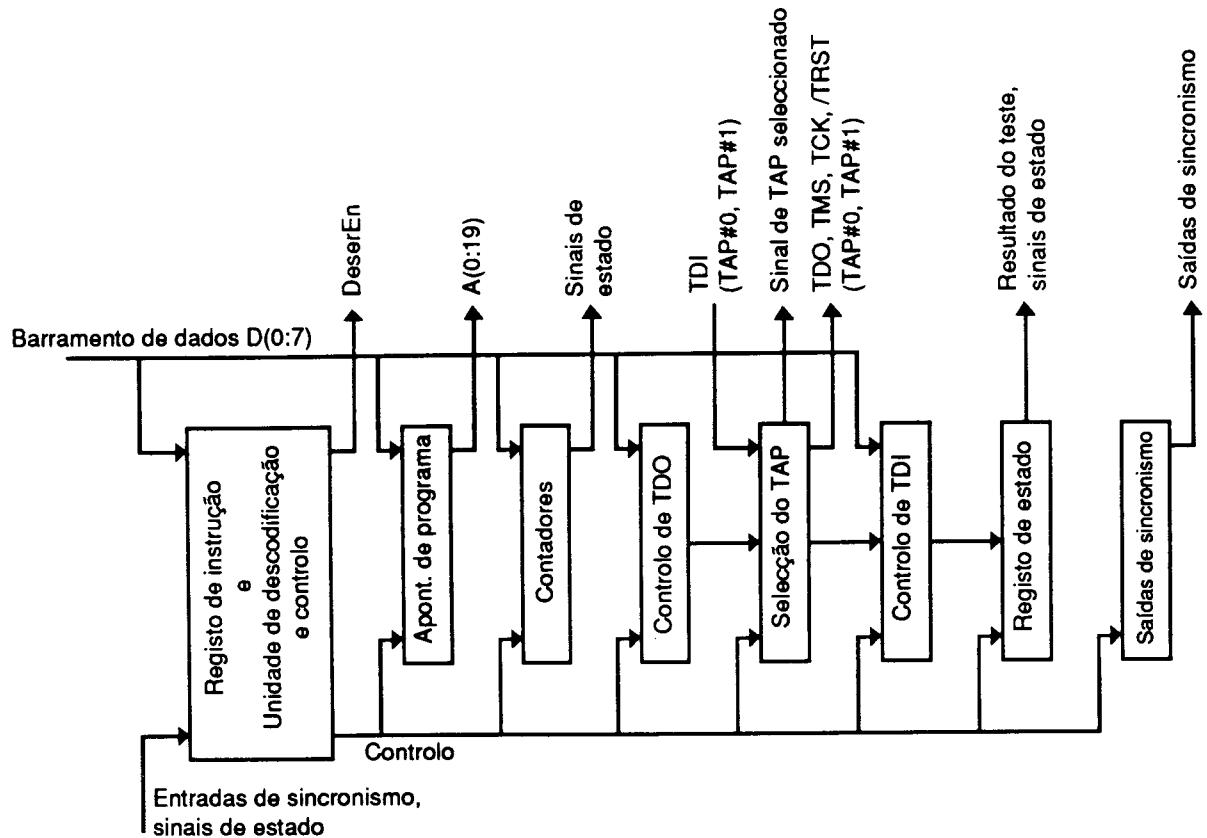
- i) As operações elementares para o controlo da infraestrutura BST (quadro 2.1) requerem a existência de blocos que se responsabilizem pelo deslocamento de vectores para a cadeia BST (controlo da linha TDO para os dois TAPs comandados pelo controlador de teste), pela recepção das respostas aos vectores aplicados anteriormente (controlo da linha TDI proveniente dos dois TAPs comandados pelo controlador de teste), 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 2.2) 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.

<b>Funcionalidade definida para o conjunto de pinos do controlador de teste BST</b>		
Designação	Núm. de pinos	Funcionalidade
A(0:19)	20	Barramento de endereços
D(0:7)	8	Barramento de dados
TAP 0	5	Controlo de uma cadeia BST da CCI a testar
TAP 1	5	Controlo da outra cadeia BST da CCI a testar
Sync...	4	Entradas e saídas de sincronismo
CLK	1	Entrada de relógio do controlador de teste
/ExtReset	1	Entrada de inicialização para o núcleo que constitui o controlador de teste 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.
EOT	1	Saída que indica quando termina o teste.
TAPS LD	1	Saída que indica qual das cadeias está seleccionada.

**Quadro 2.4 : Síntese da funcionalidade definida para os pinos do controlador de teste.**

- iii) Finalmente, as operações elementares para o controlo dos recursos internos, e controlo do fluxo do programa de teste (quadro 2.3), identificam a necessidade da existência de contadores internos, e de um registo de estado, que incluirá a *flag* de erro destinada a memorizar a ocorrência de uma falta.

A figura 2.6 ilustra a arquitectura definida para este controlador de teste, onde se identificam todos os blocos anteriormente referidos.



**Figura 2.6 : Arquitectura do controlador de teste dedicado ao controlo da infraestrutura BST da CCI a testar.**

### 2.1.3. Especificação das instruções suportadas

Esta parte efectuará a apresentação do conjunto de instruções suportadas pelo controlador de teste, de acordo com as operações elementares que foram identificadas nos quadros 2.1 a 2.3.

#### 2.1.3.1. Definição do conjunto de instruções

O conjunto de instruções suportado tem uma correspondência directa com os três tipos de operações elementares identificadas e que estão traduzidas nos quadros 2.5 a 2.7.

<b>Instruções para o controlo da Infraestrutura BST</b>	
<b>Operação elementar</b>	<b>Instrução</b>
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 2.5 : Instruções para o controlo da infraestrutura BST.

<b>Instruções para implementação do protocolo de sincronismo com o exterior</b>	
<b>Operação elementar</b>	<b>Instrução</b>
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 2.6 : Instruções relativas ao protocolo de sincronismo com o exterior.

<b>Instruções para o controlo dos recursos internos, e do fluxo do programa de teste</b>	
<b>Operação elementar</b>	<b>Instrução</b>
Carregar o contador interno com o número de impulsos pretendidos no relógio de teste.	LD C16, N LD C24, N
Implementar um salto condicional, de acordo com o estado da <i>flag</i> de erro seleccionada.	JPE Endereço, JPNE Endereço
Concluir a execução do programa de teste.	HALT

**Quadro 2.7 : Instruções para o controlo dos recursos internos, e do fluxo do programa de teste.**

### 2.1.3.2. Caracterização Individual

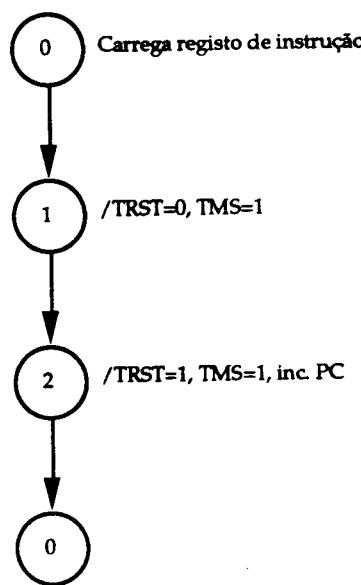
A caracterização individual de cada instrução inclui uma descrição sucinta do seu efeito, e a apresentação do respectivo diagrama de transição de estados. A caracterização de algumas instruções poderá ainda incluir elementos adicionais, tais como a descrição do armazenamento de dados em memória, ou a 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.

#### 2.1.3.2.1. 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 2.5.

##### 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 2.7.



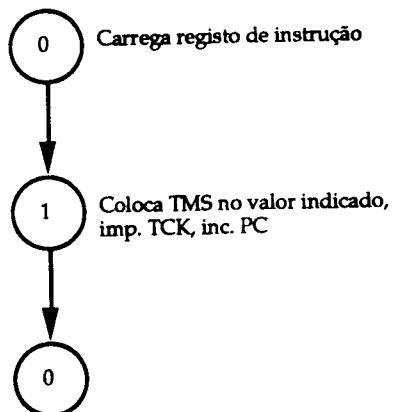
**Figura 2.7 : Diagrama de transição de estados para a instrução TRST.**

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.

O nível lógico 0 é temporariamente aplicado nas linhas /TRST dos dois TAPs comandados pelo controlador de teste 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 2.8.



**Figura 2.8 : 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).

#### NSHF

Esta instrução faz com que a sequência de N "bit's" 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 (de 16 ou 24 "bit's"), pelo que esta instrução deverá ser precedida por uma outra que carregue um ou outro contador, consoante o valor pretendido seja inferior a  $2^{16}$  ou entre  $2^{16}$  e  $2^{24}$  (LD C16, N ou LD C24, N respectivamente). O diagrama de estados para a execução desta instrução está ilustrado na figura 2.9, no qual "CNT" se refere indiferentemente à utilização de um ou outro contador.

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).

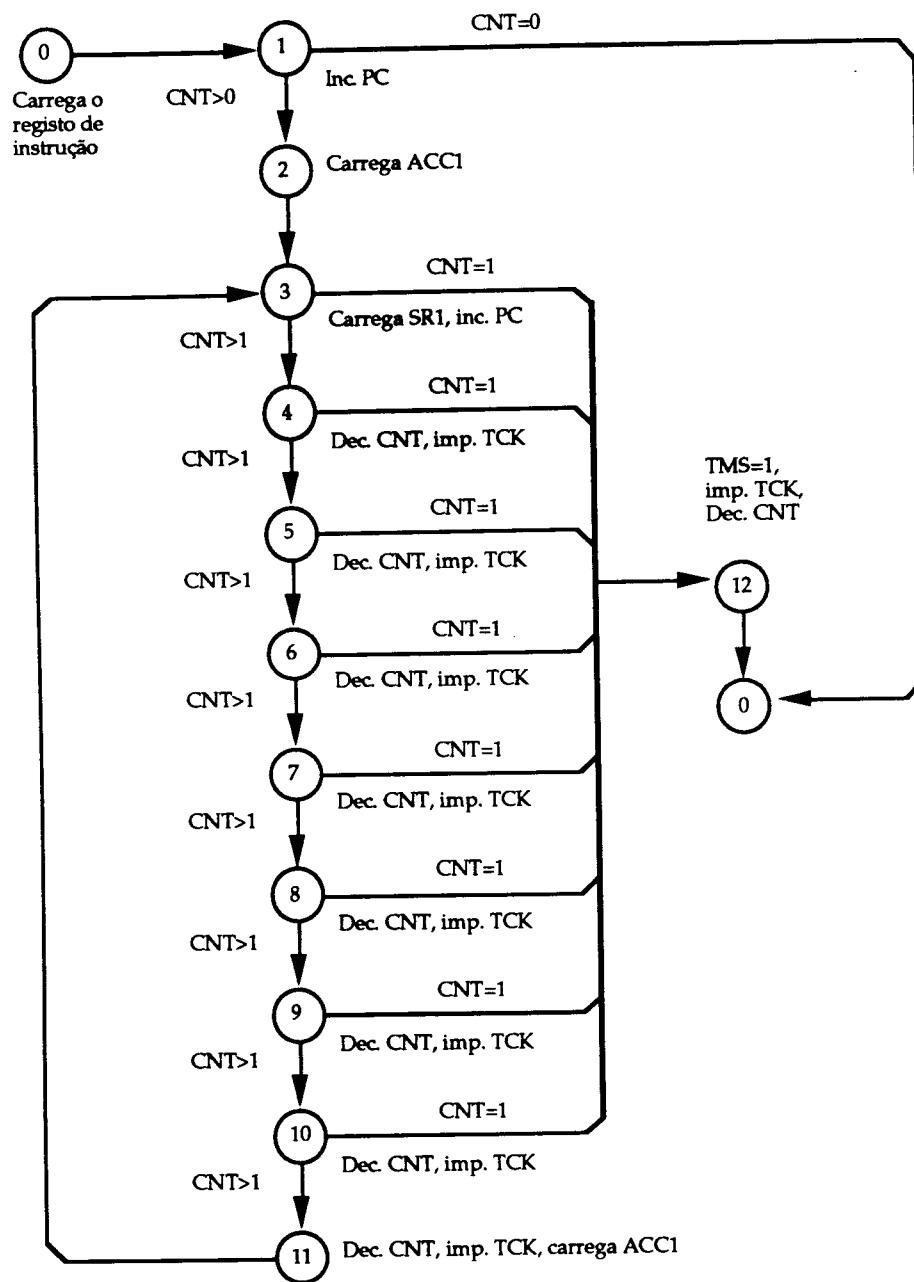


Figura 2.9 : Diagrama de transição de estados para a instrução NSHF.

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).

Os dados a deslocar para o interior da cadeia BST deverão estar armazenados como se ilustra na figura 2.10. 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
Código de NSHF								
(PC)								
(PC)+1	0	1	0	1	0	1	0	1
(PC)+2	1	1	0	0	1	0	1	0
...								

55  
CA

Figura 2.10 : Armazenamento dos dados na instrução NSHF \$55,\$CA<sup>†</sup>.

## NSHFCP

Esta instrução faz com que uma sequência de N "bit's", 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 (de 16 ou 24 "bit's"), pelo que esta instrução deverá ser precedida por uma outra que carregue um ou outro contador, consoante o valor pretendido seja inferior a  $2^{16}$  ou entre  $2^{16}$  e  $2^{24}$  (LD C16, N ou LD C24, N respectivamente). A primeira comparação de um "bit" que difira do seu valor esperado, e cuja máscara permita a comparação, actuará a *flag* de erro, e fará com que a saída do controlador de teste 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 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

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

estados para a execução desta instrução está ilustrado na figura 2.11, no qual "CNT" se refere indiferentemente à utilização de um ou outro contador.

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).

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 "bit's" a deslocar (serão necessários 9 ciclos de relógio (CLK) por cada 8 impulsos no relógio de teste (TCK)).

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 2.12. 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.

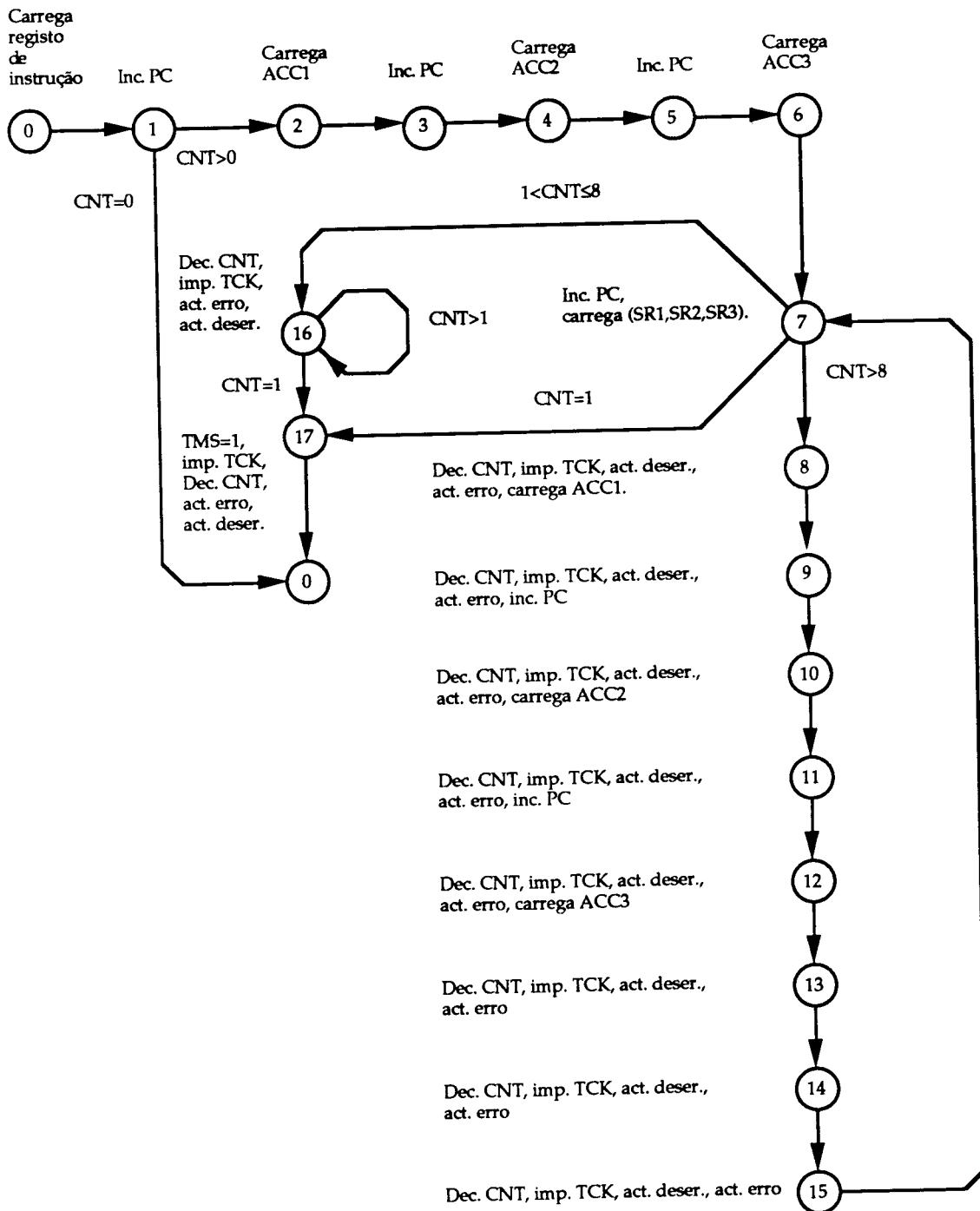


Figura 2.11 : Diagrama de transição de estados para a instrução NSHFCP.

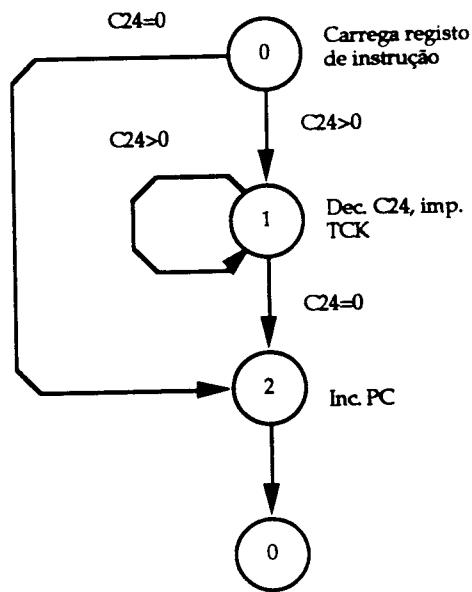
Endereço:	D7	D6	D5	D4	D3	D2	D1	D0	
Código de NSHFCP									
(PC)	0	0	0	0	1	1	0	0	0C (a deslocar)
(PC)+1	1	1	0	0	0	0	0	0	C0 (valores esperados)
(PC)+2	1	1	1	0	0	1	1	1	E7 (máscara de comparação)
(PC)+3	0	0	0	0	1	1	0	0	0C (a deslocar)
(PC)+4	1	0	0	0	0	0	0	0	80 (valores esperados)
(PC)+5	0	0	0	0	0	0	1	1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

Figura 2.12 : 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 24 "bit's"), pelo que esta instrução deverá ser precedida por uma outra que carregue este contador com o valor pretendido (LD C24, 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 2.13.

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 controlador de teste (CLK).



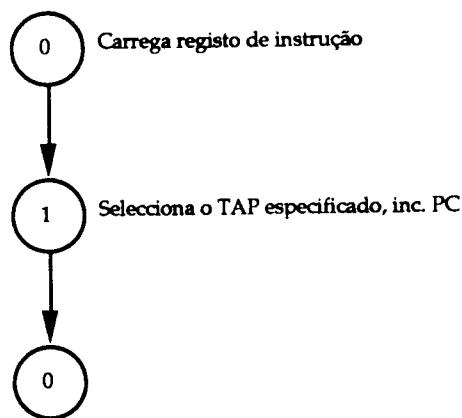
**Figura 2.13 : Diagrama de transição de estados para a instrução NTCK.**

### **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 controlador de teste, ou por outra instrução SELTAP posterior. O diagrama de transição de estados para esta instrução está ilustrado na figura 2.14.

#### **2.1.3.2.2. Instruções para implementação do protocolo de sincronismo com o exterior.**

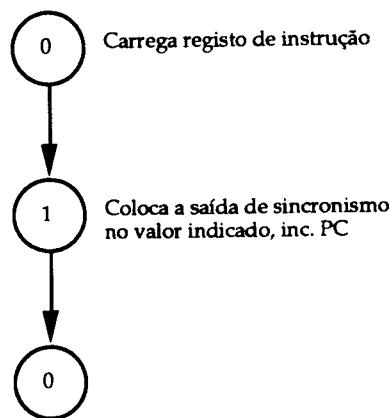
Apresenta-se nesta secção a caracterização individual das instruções destinadas a implementar o protocolo de sincronismo com o exterior, que foram descritas no quadro 2.6.



**Figura 2.14 :** Diagrama de transição de estados para a instrução SELTAP.

#### 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).

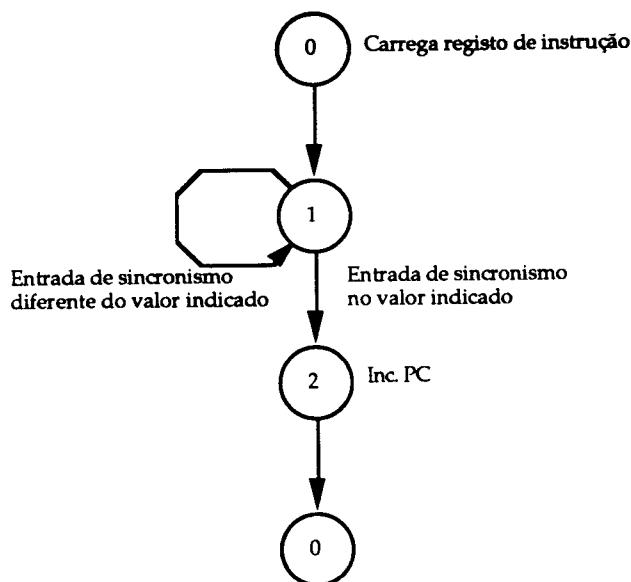


**Figura 2.15 :** Diagrama de transição de estados para as instruções SS.

O valor aplicado na saída de sincronismo poderá apenas ser modificado por uma reinicialização do controlador de teste, 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 2.15.

**WSA0, WSA1, WSB0, WSB1**

Estas instruções retêm o controlador de teste 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 2.16.



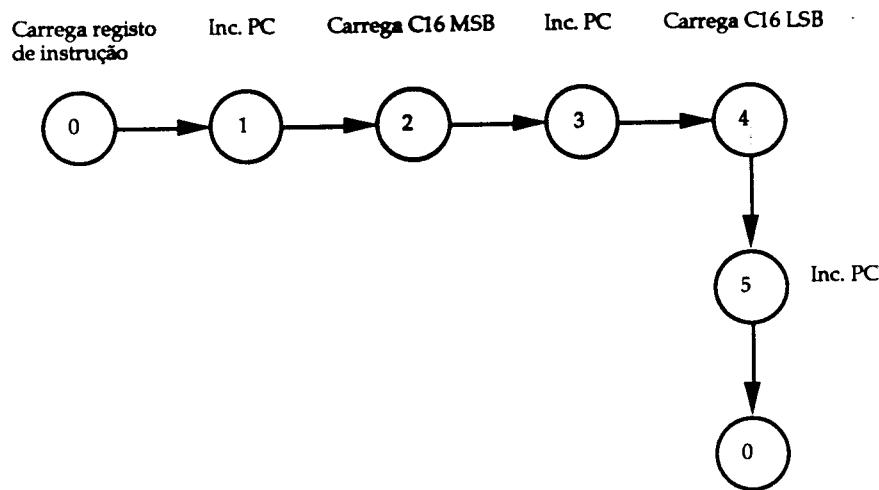
**Figura 2.16 :** Diagrama de transição de estados para as instruções WS.

#### 2.1.3.2.3. 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 controlador de teste, e do fluxo do programa de teste, que foram descritas no quadro 2.7.

#### LD C16, Conteúdo

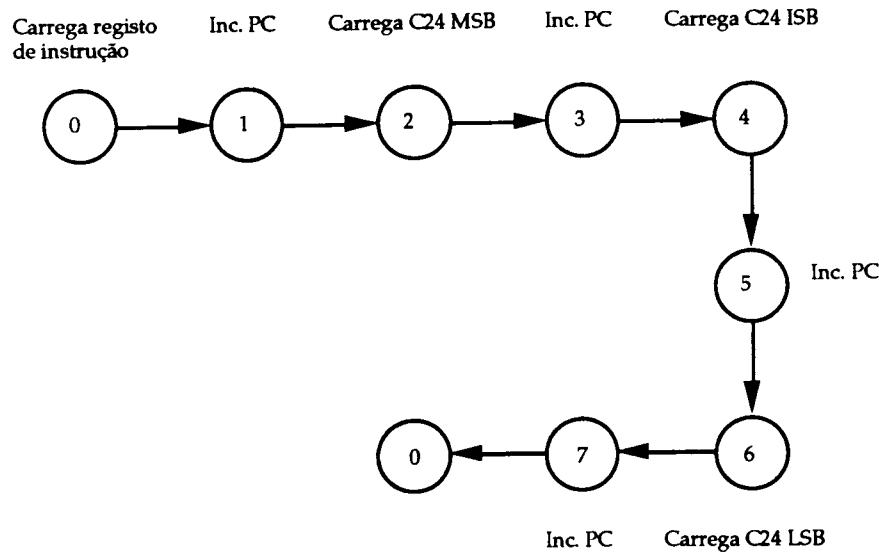
Esta instrução transfere o conteúdo das primeira e segunda palavras de memória, após o código de instrução, para o contador interno. A primeira palavra deverá conter os 8 "bit's" mais significativos do valor a carregar, e a segunda palavra os 8 "bit's" menos significativos. O diagrama de transição de estados para a execução desta instrução está ilustrado na figura 2.17.



**Figura 2.17 :** Diagrama de transição de estados para a instrução LD C16, Conteúdo.

### LD C24, Conteúdo

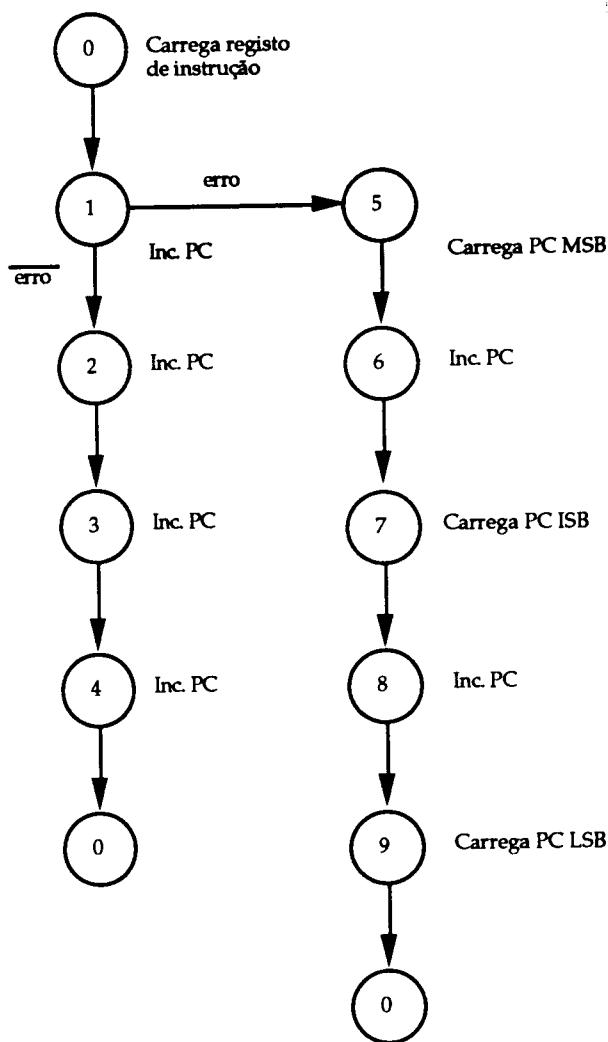
Esta instrução transfere o conteúdo das primeira, segunda e terceira palavras de memória, após o código de instrução, para o contador interno. A primeira palavra deverá conter os 8 "bit's" mais significativos do valor a carregar, e a terceira palavra os 8 "bit's" menos significativos. O diagrama de transição de estados para a execução desta instrução está ilustrado na figura 2.18.



**Figura 2.18 :** Diagrama de transição de estados para a instrução LD C24, Conteúdo.

### JPE Endereço

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

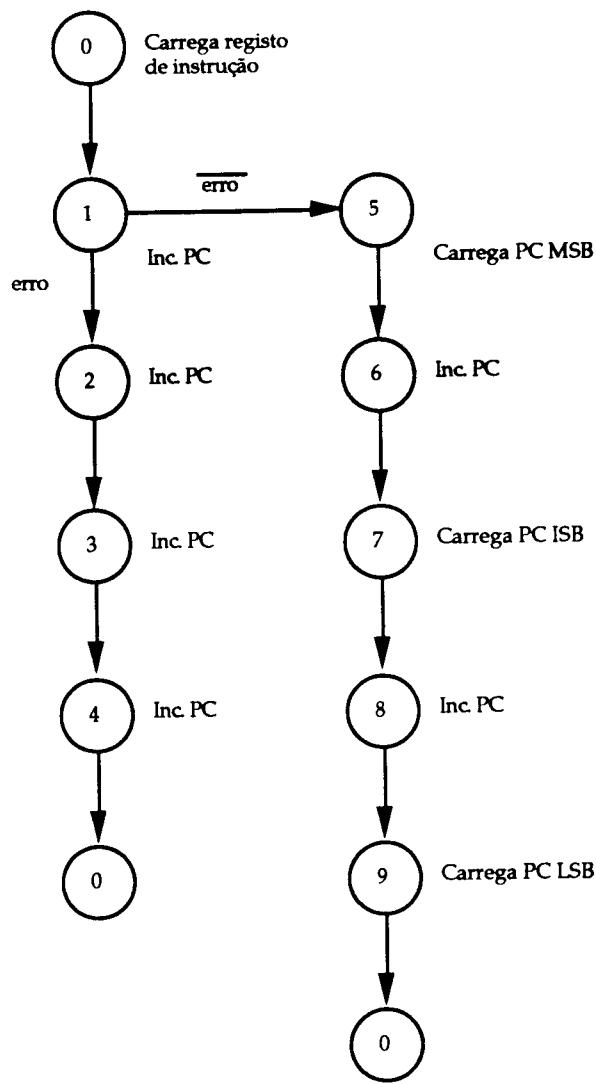


**Figura 2.19 : Diagrama de transição de estados para a instrução JPE Endereço.**

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 2.19.

### JPNE Endereço

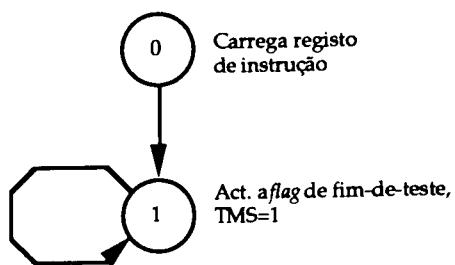
Esta instrução provoca um salto para o endereço especificado, se a *flag* de erro não tiver sido actuada anteriormente. Caso contrário, o próximo código de instrução será lido na primeira posição que se segue ao operando desta instrução. O correspondente diagrama de transição de estados está ilustrado na figura 2.20.



**Figura 2.20 :** 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 na figura 2.21.



**Figura 2.21 :** Diagrama de transição de estados para a instrução HALT.

#### 2.1.3.2.4. 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 2.8.

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	$1 + \lceil (N/8) \rceil \uparrow\uparrow$	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 C16, Conteúdo	3	6
LD C24, Conteúdo	4	8
JPE Endereço, JPNE Endereço	4	5, se o salto não ocorrer 6, se o salto ocorrer
HALT	1	Indefinido

Quadro 2.8 : Requisitos em memória, e tempo de execução, para cada instrução (N: valor do contador).

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

## 3. EXEMPLO DE APLICAÇÃO

### 3.1. Descrição da CCI de demonstração

A CCI descrita neste anexo permite demonstrar o funcionamento do controlador de teste. 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 subprojeto 4 do projecto ESPRIT 2478, tal como se descreve em (Ferreira et al., 92c).

### 3.2. Informação requerida pelo 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).

A figura 2 identifica a sequência de células nas duas cadeias BST, e os grupos de componentes não BST, presentes nesta CCI.

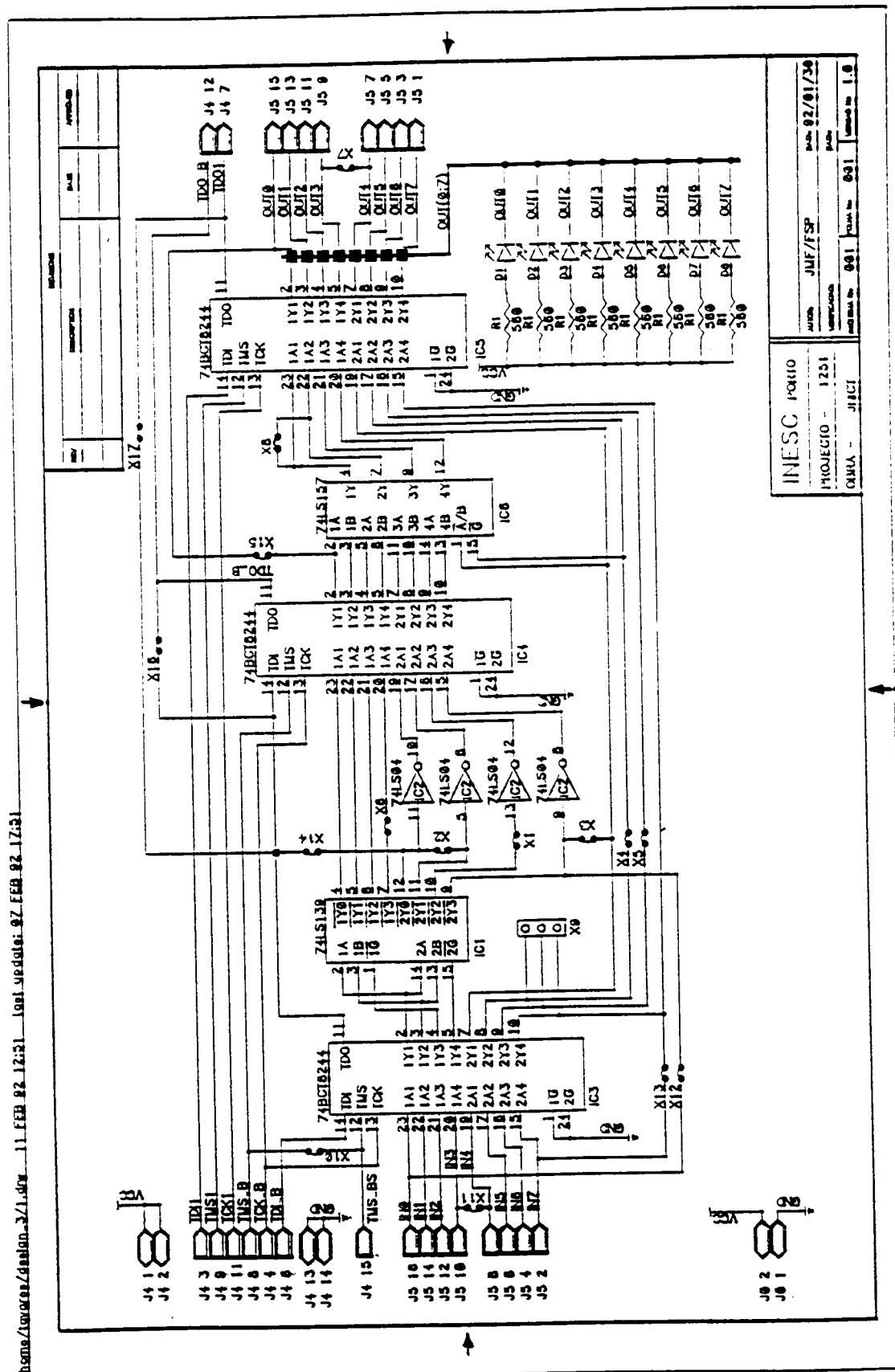
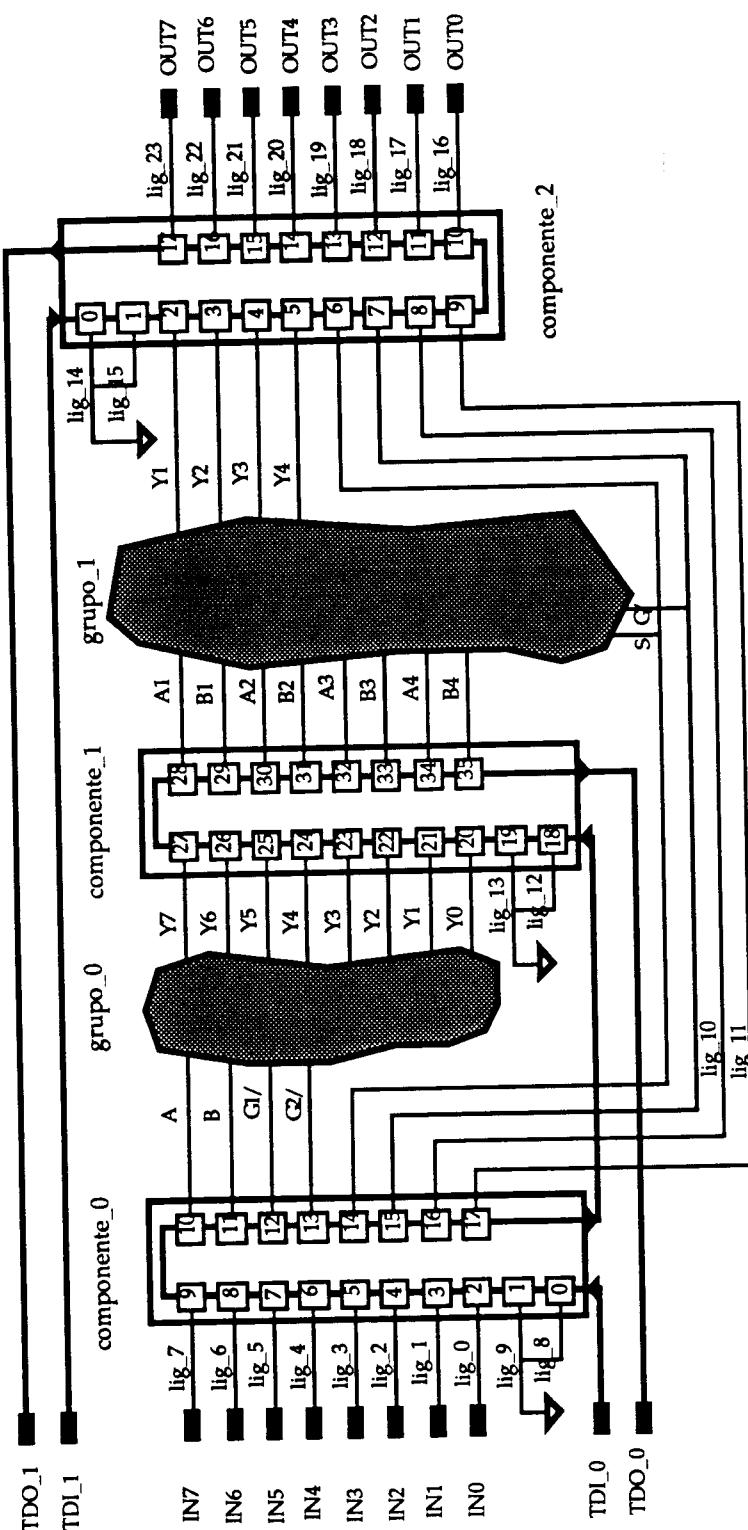


Figura 3.1: Esquema eléctrico da CCI de demonstração.



**Figura 3.2:** CCI de demonstração: ligações e componentes presentes nas duas cadeias BST.

### 3.2.1. 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.

	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 3.1: Informação necessária para o teste da infraestrutura BST da CCI.

### 3.2.2. Teste de ligações

Esta etapa inclui o teste das ligações BST, e também do conjunto de ligações pertencentes a grupos de componentes não BST.

#### 3.2.2.1. Ligações BST

A CCI de demonstração inclui as 24 ligações BST representadas na figura 3.2. A informação necessária para a geração do respectivo segmento do programa de teste está representada no quadro 3.2. Com a excepção das ligações a valores lógicos fixos, todas as restantes ligações descritas no quadro 3.2 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.

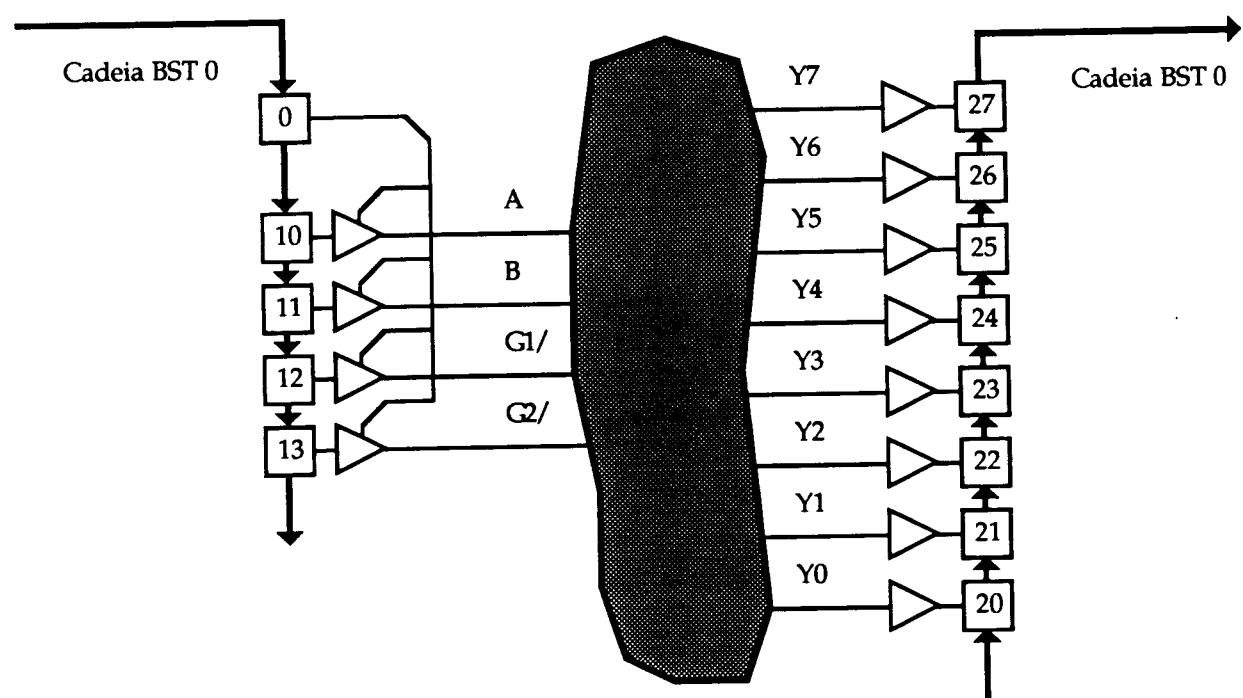
	N.º cad. BST	Ident. cad. BST	N.º pin. saída said.	Cél. saída	Alta contr.	N.º pinos entr.	Cél. entr.	N.º pinos entr. da bid.	N.º entr. prim.	Alta imp.	N.º saída prim.	Saída prim.	Pot. fixo?	Nível lógico
lig_0	1	0	0	-	-	1	2	0	1	0	0	0	-	não
lig_1	1	0	0	-	-	1	3	0	1	1	0	0	-	não
lig_2	1	0	0	-	-	1	4	0	1	2	0	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	0	0	-	sim 0
lig_9	1	0	0	-	-	1	1	0	0	0	0	0	-	sim 0
lig_10	2	0	1	16	1	1	0	-	0	0	0	0	-	não -
lig_11	2	0	1	17	1	1	0	-	1	8	0	0	-	sim 0
lig_12	1	0	0	-	-	1	9	0	0	0	0	0	-	não -
lig_13	1	0	0	-	-	1	18	0	0	0	0	0	-	sim 0
lig_14	1	1	0	-	-	1	19	0	0	0	0	0	-	sim 0
lig_15	1	1	0	-	-	1	0	0	0	0	0	0	-	sim 0
lig_16	1	1	1	10	0	1	0	-	0	0	0	0	-	sim 0
lig_17	1	1	1	11	0	1	0	-	0	0	0	0	-	sim 0
lig_18	1	1	1	12	0	1	0	-	0	0	0	0	-	não -
lig_19	1	1	1	13	0	1	0	-	0	0	0	0	-	sim 0
lig_20	1	1	1	14	1	1	0	-	0	0	0	0	-	não -
lig_21	1	1	1	15	1	1	0	-	0	0	0	0	-	sim 0
lig_22	1	1	1	16	1	1	0	-	0	0	0	0	-	não -
lig_23	1	1	1	17	1	1	0	-	0	0	0	0	-	não -

Quadro 3.2: Informação necessária para o teste das ligações BST.

### 3.2.2.2. 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 3.2.

O grupo 0 é constituído pelos componentes designados como IC1 e IC2 (SN74LS139 e SN74LS04) na figura 3.1, e está completamente envolvido por células pertencentes à cadeia BST 0, tal como se apresenta na figura 3.3.



**Figura 3.3:** 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 3.3 e 3.4.

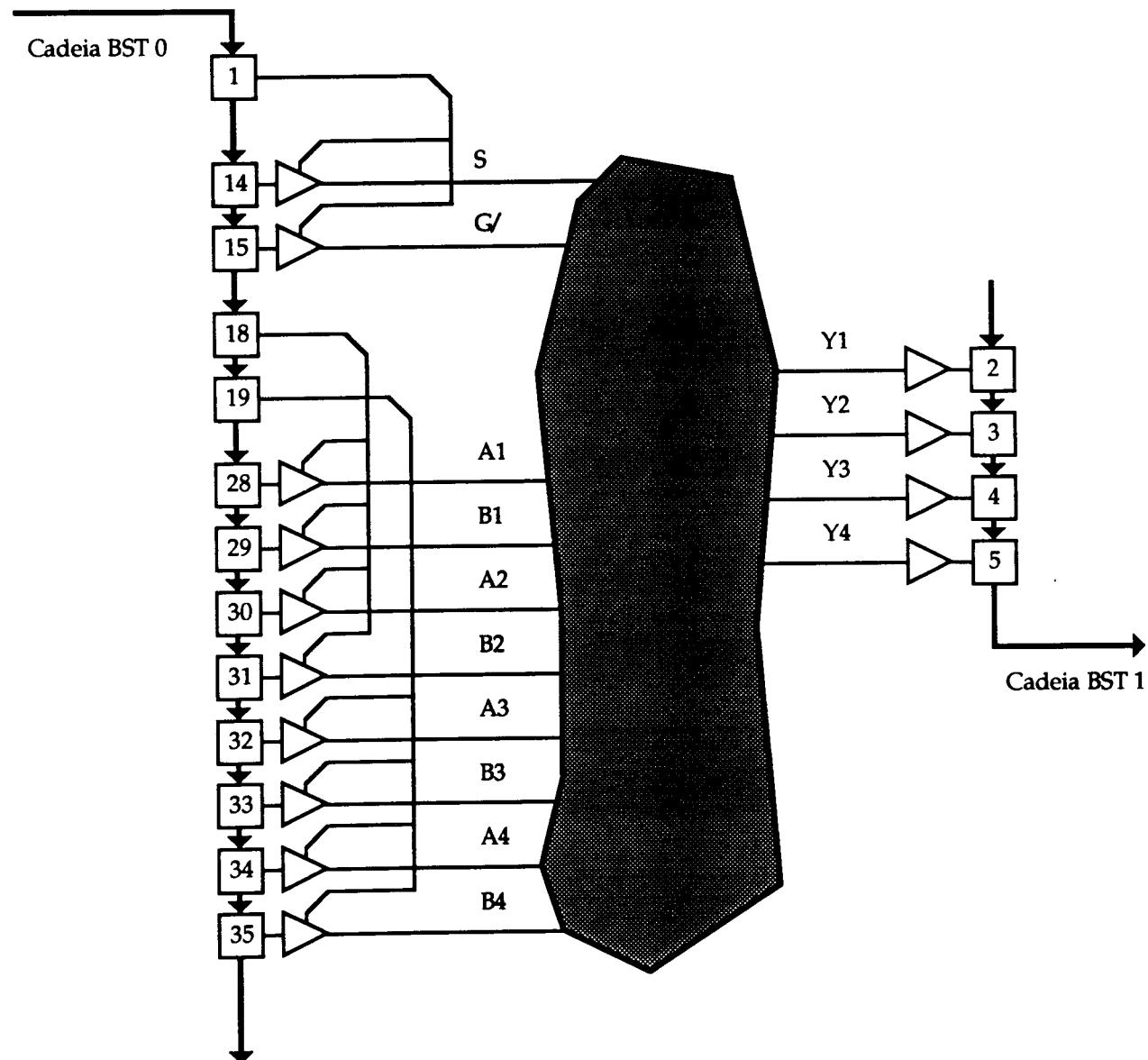
N.º cadeias BST	Ident. cadeia BST	N.º células	Identificação das células BST	N.º entr. prim.	N.º saídas prim.	Aplicar valores durante o teste às ligações BST?
1	0	13	0,10,11,12,13,20,21,22,23,24,25,26, 27	0	0	não

Quadro 3.3: 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	011101111111 011011111111 000001111111 001001111111 010001111111	111111110001 1111111100000 1111101111000 1111111010010 1111110110100	000001111111 000001111111 000001111111 000001111111 000001111111

Quadro 3.4: 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 3.4.



**Figura 3.4:** 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 3.5 e 3.6. A informação contida nestes quadros corresponde à informação necessária para que se processe a geração automática dos vectores de teste pelo programa descrito em (Tavares et al, 93).

N.º cadeias BST	Ident. cadeia BST	N.º células	Identificação das células BST	N.º entr. prim.	N.º 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 3.5: 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	000000000000000
		0000000010001	1111111111111	000000000000000
		0110000110111	1111111111111	000000000000000
		0100011010101	1111111111111	000000000000000
		0100000100010	1111111111111	000000000000000
	1	1111	1111	1111
		1111	0000	1111
		1111	0000	1111
		1111	1111	1111
		1111	0000	1111

Quadro 3.6: Descrição dos vectores de teste para o grupo 1.

### 3.2.3. 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. registro BST	Supora 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 3.7: 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 3.8: Descrição dos vectores para o teste interno dos componentes.

### 3.3. Resultados produzidos pelo 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.

#### 3.3.1. 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 3.9. 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).

ptv[0]:	010101010001000001010101
ptv[1]:	010101010010000010101010
ptv[2]:	011010100001000001011010
ptv[3]:	100110100001000010100101
ptv[4]:	10100110001000001100110
ptv[5]:	101010010010000010011001

Quadro 3.9: Resultado produzido pelo algoritmo para teste de curto-circuitos entre ligações BST.

#### 3.3.2. 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.

```

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 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 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 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 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 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 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):
```











```
*****
Boundary scan chain number 0:
Test pattern:
TP[0][9]:
    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 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 0 0 0 0 0 1 1 1 1 1 1 1
Mask bits (for this expected result):
MB[0][9]:
    0 0 1 1 1 1 1 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

Boundary scan chain number 1:
Test pattern:
TP[1][9]:
    0 0 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][9]:
    0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1
Mask bits (for this expected result):
MB[1][9]:
    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
CS[9]:
    1 1 1 1 1 1 1

Expected values and masks at primary outputs:
PO[9]:
    1 1 1 1 1 1 1
POM[9]:
    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 0 1 1 1 1 0 0 1 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 0 1 1 1 1 0 0 0 1 1 1 1 1 1 1  
Mask bits (for this expected result):  
MB[0][10]:  
0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 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  
Expected result - shifted out while the next TP is shifted in:  
ER[1][10]:  
0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1  
Mask bits (for this expected result):  
MB[1][10]:  
1 1 1 1 1 0 0 1 1 0 0 0 0 0 0 0

Primary inputs and corresponding control signals:

PI[10]:  
1 1 1 1 1 1 1  
CS[10]:  
1 1 1 1 1 1 1

Expected values and masks at primary outputs:

PO[10]:  
1 1 1 1 1 1 1  
POM[10]:  
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 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
Mask bits (for this expected result):
MB[0][11]:
  0 0 1 1 1 1 1 1 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 0 0

Boundary scan chain number 1:

Test pattern:
TP[1][11]:
  0 0 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][11]:
  0 0 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 0

Primary inputs and corresponding control signals:

PI[11]:
  1 1 1 1 1 1 1
CS[11]:
  1 1 1 1 1 1 1

Expected values and masks at primary outputs:

PO[11]:
  1 1 1 1 1 1 1
POM[11]:
  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 1 0 0 0 1 0 1 1 0 0 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 1 0 1 1 0 1 0 0 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 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0

Boundary scan chain number 1:

Test pattern:
TP[1][12]:
  0 0 1 1 1 0 0 0 0 1 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 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 0

Primary inputs and corresponding control signals:

PI[12]:
  1 1 1 1 1 1 1
CS[12]:
  1 1 1 1 1 1 1

Expected values and masks at primary outputs:

PO[12]:
  1 1 1 1 1 1 1
POM[12]:
  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

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 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 0 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 0 1 0 1 0 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 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

Boundary scan chain number 1:

Test pattern:

TP[1][0]:  
1 1 0 1 0 1 0 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 0 1 0 1 0 1  
Mask bits (for this expected result):  
MB[1][0]:  
0 0 0 0 0 0 0 0 0 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 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 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 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 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 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 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 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

### 3.3.3. Programa de teste

O programa de teste que permite a aplicação dos vectores descritos foi igualmente produzido pelo GAPT, em linguagem simbólica (*assembly*). O programa de teste gerado produziu a listagem apresentada a seguir e foi posteriormente convertido para código objecto com o auxílio do *cross-assembler* TASM †.

```
; ****
; Assembly source code for the demophd circuit.
; ****

start .org      0h

; Sequence to reset all BST logic.

    seltap0      ; (switch to) TAP 0
    trst          ; TRST0 output is pulsed low
    tmsl
    tmsl
    tmsl
    tmsl
    tmsl      ; soft reset of BST chain
    seltap1      ; switch to TAP 1
    trst          ; TRST1 output is pulsed low
    tmsl
    tmsl
    tmsl
    tmsl
    tmsl      ; soft reset of BST chain 1
    seltap0      ; switch to TAP 0

; Sequence to test the BST infrastructure

    tms0          ; BST chain 0, >> Run Test / Idle
    tms1          ; BST chain 0, >> Select DR scan
    tms1          ; BST chain 0, >> Select IR scan
    tms0          ; BST chain 0, >> Capture IR
    tms0          ; BST chain 0, >> Shift IR
    ld            cl16,26 ; length of IR's + 10, BST chain 0
    nshfcop
    inf0 .db      ; shift in the ID or BP opcodes
    $01,$fd,$03,$fc,$fd,$03,$ff,$01,$ff,$03,$00,$03
    theend        ; stop the test, if a BST infrastructure fault is found
    ; BST chain 0, >> Update IR
    ; BST chain 0, >> Select DR scan
    ; BST chain 0, >> Capture DR
    tms1          ; BST chain 0, >> Shift DR
    tms0          cl16,2 ; length of ID (BP, if no ID) registers, BST chain 0
    nshfcop
    inf1 .db      ; shift out the contents of the ID (BP, if no ID) registers
    $03,$00,$03
    theend        ; stop the test, if a BST infrastructure fault is found
    ; BST chain 0, >> Update DR
    ; BST chain 0, >> Select DR scan
    ; BST chain 0, >> Select IR scan
    ; BST chain 0, >> Capture IR
    tms1          ; BST chain 0, >> Shift IR
    ld            cl16,16 ; length of IR's, BST chain 0
    nshfcop
    inf2 .db      ; shift in the sample/preload opcodes
    $02,$fd,$03,$02,$fd,$03
    theend        ; stop the test, if a BST infrastructure fault is found
    ; BST chain 0, >> Update IR
    ; BST chain 0, >> Select DR scan
    ; BST chain 0, >> Capture DR
    tms1          ; BST chain 0, >> Exit-1 DR
    tms1          ; BST chain 0, >> Update DR
    tms1          ; BST chain 0, >> Select DR scan
    tms1          ; BST chain 0, >> Select IR scan
    tms0          ; BST chain 0, >> Capture IR
```

† © Speech Technology Incorporated.

```

tms0 ; BST chain 0, >> Shift IR
ld c16,16 ; length of IR's, BST chain 0
nshfcop ; shift in the extest opcodes
.inf3 .db $00,$fd,$03,$00,$fd,$03
jpe tms1 ; BST chain 0, >> Update IR
tms1 ; BST chain 0, >> Select DR scan
tms0 ; BST chain 0, >> Capture DR
tms0 ; BST chain 0, >> Shift DR
seltap1 ; switch to TAP 1
tms0 ; BST chain 1, >> Run Test / Idle
tms1 ; BST chain 1, >> Select DR scan
tms1 ; BST chain 1, >> Select IR scan
tms0 ; BST chain 1, >> Capture IR
tms0 ; BST chain 1, >> Shift IR
ld c16,18 ; length of IR's + 10, BST chain 1
nshfcop ; shift in the ID or BP opcodes
.inf4 .db $01,$fd,$03,$fc,$01,$ff,$03,$00,$03
jpe tms1 ; stop the test, if a BST infrastructure fault is found
tms1 ; BST chain 1, >> Update DR
tms1 ; BST chain 1, >> Select DR scan
tms0 ; BST chain 1, >> Capture DR
tms0 ; BST chain 1, >> Shift DR
ld c16,1 ; length of ID (BP, if no ID) registers, BST chain 1
nshfcop ; shift out the contents of the ID (BP, if no ID) registers
.inf5 .db $01,$00,$01
jpe tms1 ; stop the test, if a BST infrastructure fault is found
tms1 ; BST chain 1, >> Update DR
tms1 ; BST chain 1, >> Select DR scan
tms1 ; BST chain 1, >> Select IR scan
tms0 ; BST chain 1, >> Capture IR
tms0 ; BST chain 1, >> Shift IR
ld c16,8 ; length of IR's, BST chain 1
nshfcop ; shift in the sample/preload opcodes
.inf6 .db $02,$fd,$03
jpe tms1 ; stop the test, if a BST infrastructure fault is found
tms1 ; BST chain 1, >> Update IR
tms1 ; BST chain 1, >> Select DR scan
tms0 ; BST chain 1, >> Capture DR
tms1 ; BST chain 1, >> Exit-1 DR
tms1 ; BST chain 1, >> Update DR
tms1 ; BST chain 1, >> Select DR scan
tms1 ; BST chain 1, >> Select IR scan
tms0 ; BST chain 1, >> Capture IR
tms0 ; BST chain 1, >> Shift IR
ld c16,8 ; length of IR's, BST chain 1
nshfcop ; shift in the extest opcodes
.inf7 .db $00,$fd,$03
jpe tms1 ; stop the test, if a BST infrastructure fault is found
tms1 ; BST chain 1, >> Update DR
tms1 ; BST chain 1, >> Select DR scan
tms0 ; BST chain 1, >> Capture DR
tms0 ; BST chain 1, >> Shift DR
seltap0 ; switch to TAP 0

; Sequence for testing the full BST interconnects

serflg3 ; select error flag 3 (for full BST interconnect open faults)
ld c16,36 ; length of BST chain 0
nshf ; shift 1st TP (for open testing), no compare
.int0 .db $00,$00,$00,$fc,$03
tms1 ; BST chain 0, >> Update DR
tms1 ; BST chain 0, >> Sel. DR scan
seltap1 ; switch to TAP 1
ld c16,18 ; length of BST chain 1
nshf ; shift the 1st TP (for open faults) on BST chain 1
.int1 .db $00,$03,$00
tms1 ; BST chain 1, >> Update DR
tms1 ; BST chain 1, >> Sel. DR scan
ssal ; SyncOutA goes high (BST test data applied)
wsal ; wait for SyncInpA to go high (external test data applied)
tms0 ; BST chain 1, >> Capture DR
tms0 ; BST chain 1, >> Shift DR
seltap0 ; switch to TAP 0
tms0 ; BST chain 0, >> Capture DR
tms0 ; BST chain 0, >> Shift DR
ssal ; SyncOutA goes low (BST test results captured)
wsal ; wait for SyncInpA to go low (external test results captured)
ld c16,36 ; length of BST chain 0
nshfcop ; test pattern for open faults
.int2 .db $00,$00,$00,$00,$00,$0c,$00,$03,$fc,$00,$fc,$03,$00,$0f
jpe tms1 ; stop the test, if a net is found open, or s-a-1
tms1 ; BST chain 0, >> Update DR
tms1 ; BST chain 0, >> Sel. DR scan
seltap1 ; switch to TAP 1
ld c16,18 ; length of BST chain 1
nshfcop ; TP for open faults, BST chain 1
.int3 .db $ff,$00,$00,$03,$00,$03,$00,$00,$03
jpe tms1 ; stop the test, if a net is found open, or s-a-1
tms1 ; BST chain 1, >> Update DR
tms1 ; BST chain 1, >> Sel. DR scan
ssal ; SyncOutA goes high (BST test data applied)
wsal ; wait for SyncInpA to go high (external test data applied)
tms0 ; BST chain 1, >> Capture DR
tms0 ; BST chain 1, >> Shift DR
seltap0 ; switch to TAP 0

```

```

tms0          ; BST chain 0, >> Capture DR
tms0          ; BST chain 0, >> Shift DR
ssa0          ; SyncOutA goes low (BST test results captured)
wsa0          ; wait for SyncInpA to go low (external test results captured)
serflg4      ; select error flag 4 (for full BST interconnect short faults)
ld            ; c16,36 : length of BST chain 0
nshfcop      ; test pattern for short faults
int4         $00,$00,$00,$00,$00,$04,$0c,$fc,$fc,$03,$03,$0f
.db          theend ; stop the test, if a short fault is found
jpe          ; BST chain 0, >> Update DR
tms1          ; BST chain 0, >> Sel. DR scan
tms1          ; switch to TAP 1
seltap1      c16,18 : length of BST chain 1
ld            ; TP for short faults, BST chain 1
nshfcop      $55,$ff,$00,$03,$03,$03,$00,$00,$03
int5         theend ; stop the test, if a short fault is found
.db          ; BST chain 1, >> Update DR
jpe          ; BST chain 1, >> Sel. DR scan
tms1          ; SyncOutA goes high (BST test data applied)
ssal          ; wait for SyncInpA to go high (external test data applied)
wsal          ; BST chain 1, >> Capture DR
tms0          ; BST chain 1, >> Shift DR
tms0          ; switch to TAP 0
seltap0      ; BST chain 0, >> Capture DR
tms0          ; BST chain 0, >> Shift DR
ssa0          ; SyncOutA goes low (BST test results captured)
wsa0          ; wait for SyncInpA to go low (external test results captured)
ld            ; c16,36 : length of BST chain 0
nshfcop      ; test pattern for short faults
int6         $00,$00,$00,$00,$00,$00,$05,$04,$54,$fc,$03,$01,$0f
.db          theend ; stop the test, if a short fault is found
jpe          ; BST chain 0, >> Update DR
tms1          ; BST chain 0, >> Sel. DR scan
tms1          ; switch to TAP 1
seltap1      c16,18 : length of BST chain 1
ld            ; TP for short faults, BST chain 1
nshfcop      $aa,$55,$00,$03,$01,$03,$00,$00,$03
int7         theend ; stop the test, if a short fault is found
.db          ; BST chain 1, >> Update DR
jpe          ; BST chain 1, >> Sel. DR scan
tms1          ; SyncOutA goes high (BST test data applied)
wsal          ; wait for SyncInpA to go high (external test data applied)
tms0          ; BST chain 1, >> Capture DR
tms0          ; BST chain 1, >> Shift DR
seltap0      ; switch to TAP 0
tms0          ; BST chain 0, >> Capture DR
tms0          ; BST chain 0, >> Shift DR
ssa0          ; SyncOutA goes low (BST test results captured)
wsa0          ; wait for SyncInpA to go low (external test results captured)
ld            ; c16,36 : length of BST chain 0
nshfcop      ; test pattern for short faults
int8         $00,$00,$00,$00,$00,$00,$04,$08,$03,$fc,$54,$fc,$03,$01,$0f
.db          theend ; stop the test, if a short fault is found
jpe          ; BST chain 0, >> Update DR
tms1          ; BST chain 0, >> Sel. DR scan
tms1          ; switch to TAP 1
seltap1      c16,18 : length of BST chain 1
ld            ; TP for short faults, BST chain 1
nshfcop      $5a,$aa,$00,$03,$02,$03,$00,$00,$03
int9         theend ; stop the test, if a short fault is found
.db          ; BST chain 1, >> Update DR
jpe          ; BST chain 1, >> Sel. DR scan
tms1          ; SyncOutA goes high (BST test data applied)
ssal          ; wait for SyncInpA to go high (external test data applied)
wsal          ; BST chain 1, >> Capture DR
tms0          ; BST chain 1, >> Shift DR
seltap0      ; switch to TAP 0
tms0          ; BST chain 0, >> Capture DR
tms0          ; BST chain 0, >> Shift DR
ssa0          ; SyncOutA goes low (BST test results captured)
wsa0          ; wait for SyncInpA to go low (external test results captured)
ld            ; c16,36 : length of BST chain 0
nshfcop      ; test pattern for short faults
int10        $00,$00,$00,$00,$00,$00,$04,$04,$03,$fc,$a8,$fc,$03,$01,$0f
.db          theend ; stop the test, if a short fault is found
jpe          ; BST chain 0, >> Update DR
tms1          ; BST chain 0, >> Sel. DR scan
tms1          ; switch to TAP 1
seltap1      c16,18 : length of BST chain 1
ld            ; TP for short faults, BST chain 1
nshfcop      $a5,$5a,$00,$03,$01,$03,$00,$00,$03
int11        theend ; stop the test, if a short fault is found
.db          ; BST chain 1, >> Update DR
jpe          ; BST chain 1, >> Sel. DR scan
tms1          ; SyncOutA goes high (BST test data applied)
ssal          ; wait for SyncInpA to go high (external test data applied)
wsal          ; BST chain 1, >> Capture DR
tms0          ; BST chain 1, >> Shift DR
seltap0      ; switch to TAP 0
tms0          ; BST chain 0, >> Capture DR
tms0          ; BST chain 0, >> Shift DR
ssa0          ; SyncOutA goes low (BST test results captured)
wsa0          ; wait for SyncInpA to go low (external test results captured)
ld            ; c16,36 : length of BST chain 0
nshfcop      ; test pattern for short faults
int12        $00,$00,$00,$00,$00,$00,$08,$04,$03,$fc,$68,$fc,$03,$02,$0f

```

```

jpe      ; stop the test, if a short fault is found
tms1    ; BST chain 0, >> Update DR
tms1    ; BST chain 0, >> Sel. DR scan
; switch to TAP 1
c16,18  ; length of BST chain 1
; TP for short faults, BST chain 1
; SyncOutA goes high (BST test data applied)
; wait for SyncInpA to go high (external test data applied)
$66,$a5,$00,$03,$01,$03,$00,$00,$03
theend  ; stop the test, if a short fault is found
tms1    ; BST chain 1, >> Update DR
tms1    ; BST chain 1, >> Sel. DR scan
; SyncOutA goes high (BST test data applied)
; wait for SyncInpA to go high (external test data applied)
$66,$a5,$00,$03,$01,$03,$00,$00,$03
theend  ; stop the test, if a short fault is found
tms1    ; BST chain 1, >> Capture DR
tms1    ; BST chain 1, >> Shift DR
; switch to TAP 0
tms0    ; BST chain 0, >> Capture DR
tms0    ; BST chain 0, >> Shift DR
; SyncOutA goes low (BST test results captured)
; wait for SyncInpA to go low (external test results captured)
c16,36  ; length of BST chain 0
; test pattern for short faults
$00,$00,$00,$00,$00,$08,$08,$03,$fc,$98,$fc,$03,$02,$0f
theend  ; stop the test, if a short fault is found
tms1    ; BST chain 0, >> Update DR
tms1    ; BST chain 0, >> Sel. DR scan
; switch to TAP 1
c16,18  ; length of BST chain 1
; TP for short faults, BST chain 1
; SyncOutA goes high (BST test data applied)
; wait for SyncInpA to go high (external test data applied)
$99,$66,$00,$03,$02,$03,$00,$00,$03
theend  ; stop the test, if a short fault is found
tms1    ; BST chain 1, >> Update DR
tms1    ; BST chain 1, >> Sel. DR scan
; SyncOutA goes high (BST test data applied)
; wait for SyncInpA to go high (external test data applied)
$99,$66,$00,$03,$02,$03,$00,$00,$03
theend  ; stop the test, if a short fault is found
tms1    ; BST chain 1, >> Capture DR
tms1    ; BST chain 1, >> Shift DR
; switch to TAP 0
tms0    ; BST chain 0, >> Capture DR
tms0    ; BST chain 0, >> Shift DR
; SyncOutA goes low (BST test results captured)
; wait for SyncInpA to go low (external test results captured)
c16,36  ; length of BST chain 0
; last test pattern for short faults, BS chain 0
$00,$00,$00,$00,$00,$08,$08,$03,$fc,$a4,$fc,$03,$02,$0f
theend  ; stop the test, if a short fault is found
; switch to TAP 1
c16,18  ; length of BST chain 1
; last TP for short faults, BST chain 1
; SyncOutA goes high (BST test data applied)
$99,$99,$00,$03,$02,$03,$00,$00,$03
theend  ; stop the test, if a short fault is found
; switch to TAP 0

; Sequence for testing the cluster interconnects

serfig5 ; select error flag 5 (for cluster interconnect faults)
tms1    ; >> Update DR, BS chain 0
tms1    ; >> Select DR scan, BS chain 0
tms0    ; >> Capture DR, BS chain 0
tms0    ; >> Shift DR, BS chain 0
; switch to TAP 1
tms1    ; >> Update DR, BS chain 1
tms1    ; >> Select DR scan, BS chain 1
tms0    ; >> Capture DR, BS chain 1
tms0    ; >> Shift DR, BS chain 1
; switch to TAP 0
c16,36  ; length of BST chain 0
; test pattern for cluster interconnect testing
$fa,$ff,$8c,$03,$00
theend  ; stop the test, if a short fault is found
tms1    ; BST chain 0, >> Update DR
tms1    ; BST chain 0, >> Sel. DR scan
; switch to TAP 1
c16,18  ; length of BST chain 1
; test pattern for cluster interconnect testing, BS chain 1
$ff,$f0,$00
theend  ; stop the test, if a short fault is found
tms1    ; BST chain 1, >> Update DR
tms1    ; BST chain 1, >> Sel. DR scan
; SyncOutA goes high (BST test data applied)
; wait for SyncInpA to go high (external test data applied)
$ff,$ff,$8c,$03,$00
theend  ; stop the test, if a short fault is found
tms1    ; BST chain 1, >> Capture DR
tms1    ; BST chain 1, >> Shift DR
; switch to TAP 0
tms0    ; BST chain 0, >> Capture DR
tms0    ; BST chain 0, >> Shift DR
; SyncOutA goes low (BST test results captured)
; wait for SyncInpA to go low (external test results captured)
c16,36  ; length of BST chain 0
; test pattern for cluster interconnect testing
$11,$ff,$00,$ff,$f1,$ff,$4c,$ff,$00,$03,$ff,$fc,$00,$0f,$03
theend  ; stop the test, if a cluster fault is detected
tms1    ; BST chain 0, >> Update DR
tms1    ; BST chain 0, >> Sel. DR scan
; switch to TAP 1
c16,18  ; length of BST chain 1
; test pattern for cluster interconnect testing, BS chain 1
$ff,$ff,$00,$f0,$f3,$f3,$00,$00,$03
theend  ; stop the test, if a cluster fault is detected

```

```

tms1          ; BST chain 1, >> Update DR
tms1          ; BST chain 1, >> Sel. DR scan
ssal          ; SyncOutA goes high (BST test data applied)
wsal          ; wait for SyncInpA to go high (external test data applied)
tms0          ; BST chain 1, >> Capture DR
tms0          ; BST chain 1, >> Shift DR
seltap0       ; switch to TAP 0
tms0          ; BST chain 0, >> Capture DR
tms0          ; BST chain 0, >> Shift DR
ssal0         ; SyncOutA goes low (BST test results captured)
wsal0         ; wait for SyncInpA to go low (external test results captured)
ld            ; length of BST chain 0
c16,36        ; test pattern for cluster interconnect testing
$37,$ff,$00,$ff,$00,$ff,$3c,$ff,$00,$00,$ff,$fc,$00,$0f,$03
theend        ; stop the test, if a cluster fault is detected
                ; BST chain 0, >> Update DR
                ; BST chain 0, >> Sel. DR scan
clu4          .db
jpe           tms1
tms1          ; BST chain 0, >> Capture DR
tms1          ; BST chain 0, >> Shift DR
seltap1       ; switch to TAP 1
ld            ; length of BST chain 1
c16,18        ; test pattern for cluster interconnect testing, BS chain 1
$ff,$ff,$00,$f0,$03,$f3,$00,$00,$03
theend        ; stop the test, if a cluster fault is detected
                ; BST chain 1, >> Update DR
                ; BST chain 1, >> Sel. DR scan
ssal          ; SyncOutA goes high (BST test data applied)
wsal          ; wait for SyncInpA to go high (external test data applied)
tms0          ; BST chain 1, >> Capture DR
tms0          ; BST chain 1, >> Shift DR
seltap0       ; switch to TAP 0
tms0          ; BST chain 0, >> Capture DR
tms0          ; BST chain 0, >> Shift DR
ssal0         ; SyncOutA goes low (BST test results captured)
wsal0         ; wait for SyncInpA to go low (external test results captured)
ld            ; length of BST chain 0
c16,36        ; test pattern for cluster interconnect testing
$45,$ff,$00,$ff,$78,$ff,$2c,$ff,$00,$01,$ff,$fc,$00,$0f,$03
theend        ; stop the test, if a cluster fault is detected
                ; BST chain 0, >> Update DR
                ; BST chain 0, >> Sel. DR scan
clu6          .db
jpe           tms1
tms1          ; BST chain 1, >> Capture DR
tms1          ; BST chain 1, >> Shift DR
seltap1       ; switch to TAP 1
ld            ; length of BST chain 1
c16,18        ; test pattern for cluster interconnect testing, BS chain 1
$ff,$ff,$00,$f0,$03,$f3,$00,$00,$03
theend        ; stop the test, if a cluster fault is detected
                ; BST chain 1, >> Update DR
                ; BST chain 1, >> Sel. DR scan
clu7          .db
jpe           tms1
tms1          ; BST chain 1, >> Capture DR
tms1          ; BST chain 1, >> Shift DR
ssal          ; SyncOutA goes high (BST test data applied)
wsal          ; wait for SyncInpA to go high (external test data applied)
tms0          ; BST chain 1, >> Capture DR
tms0          ; BST chain 1, >> Shift DR
seltap0       ; switch to TAP 0
tms0          ; BST chain 0, >> Capture DR
tms0          ; BST chain 0, >> Shift DR
ssal0         ; SyncOutA goes low (BST test results captured)
wsal0         ; wait for SyncInpA to go low (external test results captured)
ld            ; length of BST chain 0
c16,36        ; test pattern for cluster interconnect testing
$22,$ff,$00,$ff,$d2,$ff,$2c,$ff,$00,$02,$ff,$fc,$00,$0f,$03
theend        ; stop the test, if a cluster fault is detected
                ; BST chain 0, >> Update DR
                ; BST chain 0, >> Sel. DR scan
clu8          .db
jpe           tms1
tms1          ; BST chain 1, >> Capture DR
tms1          ; BST chain 1, >> Shift DR
seltap1       ; switch to TAP 1
ld            ; length of BST chain 1
c16,18        ; test pattern for cluster interconnect testing, BS chain 1
$ff,$ff,$00,$f0,$f3,$f3,$00,$00,$03
theend        ; stop the test, if a cluster fault is detected
                ; BST chain 1, >> Update DR
                ; BST chain 1, >> Sel. DR scan
clu9          .db
jpe           tms1
tms1          ; BST chain 1, >> Capture DR
tms1          ; BST chain 1, >> Shift DR
ssal          ; SyncOutA goes high (BST test data applied)
wsal          ; wait for SyncInpA to go high (external test data applied)
tms0          ; BST chain 1, >> Capture DR
tms0          ; BST chain 1, >> Shift DR
seltap0       ; switch to TAP 0
tms0          ; BST chain 0, >> Capture DR
tms0          ; BST chain 0, >> Shift DR
ssal0         ; SyncOutA goes low (BST test results captured)
wsal0         ; wait for SyncInpA to go low (external test results captured)
ld            ; length of BST chain 0
c16,36        ; last test pattern for cluster testing, BS chain 0
$22,$ff,$00,$ff,$b4,$ff,$2c,$ff,$00,$02,$ff,$fc,$00,$0f,$03
theend        ; stop the test, if a cluster fault is detected
                ; switch to TAP 1
                ; length of BST chain 1
clu10         .db
jpe           seltap1
ld            ; last TAP for cluster interconnect testing, BST chain 1
c16,18        ; last T for cluster interconnect testing, BST chain 1
theend        ; stop the test, if a cluster fault is detected
                ; switch to TAP 0

; Sequence for internal test of components

serflg7       ; select error flag 7 (for faults detected during internal test)
seltap0       ; switch to BS chain 0
tms1          ; >> Update DR, BS chain 0
tms1          ; >> Select DR scan, BS chain 0
tms1          ; >> Select IR scan, BS chain 0
tms0          ; >> Capture IR, BS chain 0

```

```

tms0      ; >> Shift IR, BS chain 0
ld        c16,16 ; length of the IRs in scan chain 0
nshf      ; shift in the inttest/bypass opcodes
.com0    .db      $00,$00
          ; >> Update IR, BS chain 0
tms1      ; >> Select DR scan, BS chain 0
tms1      ; >> Capture DR, BS chain 0
tms0      ; >> Shift DR, BS chain 0
tms0      ; length of scan chain 0
ld        c16,36 ; shift the 1st TP for internal test (no compare)
nshf      ; test pattern for internal test of components
.com1    .db      $ff,$55,$ff,$57,$0d
          ; >> Update DR, BS chain 0
tms1      ; >> Select DR scan, BS chain 0
tms1      ; >> Capture DR, BS chain 0
tms0      ; >> Shift DR, BS chain 0
tms0      ; length of scan chain 0
ld        c16,36 ; length of scan chain 0
nshfcop   ; test pattern for internal test of components
.com2    .db      $00,$55,$ff,$ff,$00,$03,$57,$fc,$48,$fd,$03,$0e,$0f,$00
          ; >> Update DR, BS chain 0
tms1      ; >> Select DR scan, BS chain 0
tms1      ; >> Capture DR, BS chain 0
tms0      ; >> Shift DR, BS chain 0
tms0      ; length of scan chain 0
ld        c16,36 ; length of scan chain 0
nshfcop   ; last test pattern for internal test of components
.com3    .db      $00,$aa,$ff,$aa,$00,$00,$03,$48,$fc,$48,$02,$03,$0e,$00,$00
          ; switch to BS chain 1
          ; >> Update DR, BS chain 1
tms1      ; >> Select DR scan, BS chain 1
tms1      ; >> Capture DR, BS chain 1
tms0      ; >> Shift DR, BS chain 1
tms0      ; length of scan chain 1
ld        c16,8  ; length of the IRs in scan chain 1
nshf      ; shift in the inttest/bypass opcodes
.com4    .db      $00
          ; >> Update IR, BS chain 1
tms1      ; >> Select DR scan, BS chain 1
tms0      ; >> Capture DR, BS chain 1
tms0      ; >> Shift DR, BS chain 1
c16,18   ; length of scan chain 1
ld        c16,18 ; shift the 1st TP for internal test (no compare)
nshf      ; test pattern for internal test of components
.com5    .db      $ff,$55,$03
          ; >> Update DR, BS chain 1
tms1      ; >> Select DR scan, BS chain 1
tms1      ; >> Capture DR, BS chain 1
tms0      ; >> Shift DR, BS chain 1
tms0      ; length of scan chain 1
ld        c16,18 ; length of scan chain 1
nshfcop   ; test pattern for internal test of components
.com6    .db      $00,$55,$ff,$aa,$ff,$00,$03,$03,$00
          ; >> Update DR, BS chain 1
tms1      ; >> Select DR scan, BS chain 1
tms1      ; >> Capture DR, BS chain 1
tms0      ; >> Shift DR, BS chain 1
tms0      ; length of scan chain 1
ld        c16,18 ; length of scan chain 1
nshfcop   ; last test pattern for internal test of components
.com7    .db      $00,$aa,$ff,$aa,$00,$00,$03,$00,$00
          ; End of test program.

.theend    seltap0 ; (switch to) TAP 0
          ; reset sequence for TAP 0
          ; switch to TAP 1
          ; reset sequence for TAP 1
          ; this is the end
          .end

```

primary	.org	8000h
; pattern number 0		
p10 .db	\$00	; primary inputs
c50 .db	\$ff	; control signal
p00 .db	\$00	; expected values at primary outputs
pom0 .db	\$ff	; primary outputs mask
; pattern number 1		
p11 .db	\$ff	; primary inputs
c51 .db	\$ff	; control signal
p01 .db	\$ff	; expected values at primary outputs
pom1 .db	\$ff	; primary outputs mask
; pattern number 2		
p12 .db	\$aa	; primary inputs
c52 .db	\$ff	; control signal
p02 .db	\$aa	; expected values at primary outputs
pom2 .db	\$ff	; primary outputs mask
; pattern number 3		
p13 .db	\$aa	; primary inputs
c53 .db	\$ff	; control signal

```

po3 .db      $55 ; expected values at primary outputs
pos3 .db      $ff ; primary outputs mask
; pattern number 4
pi4 .db      $56 ; primary inputs
cs4 .db      $ff ; control signal
po4 .db      $5a ; expected values at primary outputs
pos4 .db      $ff ; primary outputs mask
; pattern number 5
pi5 .db      $59 ; primary inputs
cs5 .db      $ff ; control signal
po5 .db      $a5 ; expected values at primary outputs
pos5 .db      $ff ; primary outputs mask
; pattern number 6
pi6 .db      $65 ; primary inputs
cs6 .db      $ff ; control signal
po6 .db      $66 ; expected values at primary outputs
pos6 .db      $ff ; primary outputs mask
; pattern number 7
pi7 .db      $95 ; primary inputs
cs7 .db      $ff ; control signal
po7 .db      $99 ; expected values at primary outputs
pos7 .db      $ff ; primary outputs mask
; pattern number 8
pi8 .db      $ff ; primary inputs
cs8 .db      $ff ; control signal
po8 .db      $ff ; expected values at primary outputs
pos8 .db      $ff ; primary outputs mask
; pattern number 9
pi9 .db      $ff ; primary inputs
cs9 .db      $ff ; control signal
po9 .db      $ff ; expected values at primary outputs
pos9 .db      $ff ; primary outputs mask
; pattern number 10
pi10 .db     $ff ; primary inputs
cs10 .db     $ff ; control signal
po10 .db     $ff ; expected values at primary outputs
pos10 .db    $ff ; primary outputs mask
; pattern number 11
pi11 .db     $ff ; primary inputs
cs11 .db     $ff ; control signal
po11 .db     $ff ; expected values at primary outputs
pos11 .db    $ff ; primary outputs mask
; pattern number 12
pi12 .db     $ff ; primary inputs
cs12 .db     $ff ; control signal
po12 .db     $ff ; expected values at primary outputs
pos12 .db    $ff ; primary outputs mask
.end

```

tasm: Number of errors = 0

---

## **4. CONCLUSÃO**

A integração deste controlador de teste num sistema do tipo descrito em (Alves et al, 93), que incorpora um controlador de E/S Digitais e um controlador de E/S Analógicas, aliado a um pacote de "software" que gera automaticamente os vectores necessários ao teste de uma carta, como o apresentado em (Tavares et al, 93), conduz a um alargamento do campo de aplicação da norma IEEE 1149.1 a uma maior diversidade de circuitos.

Outro ponto a salientar é o facto de o projecto de componentes em PLD's levar a uma flexibilidade de características não possível noutro tipo de componentes.

## 5. REFERÊNCIAS

- Alves, G. R., Tavares, J. A., Gericota, M. G., Ramalho, J. L., Pinto, F. S. e Ferreira, J. M. 1993. *Hardware: Implementação e Verificação*. Relatório nº3, Projecto JNICT PMCT/C/TIT/937/90 (Sistema de Validação e Teste de Cartas de Circuito Impresso com BST), Janeiro 1993.
- Bagge, C. and Driessen, W. 1989. *Tester Architecture*. Report R6.1, ESPRIT Project 2478 (Research into Boundary Scan Test Implementation), July 1989.
- 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.
- Ferreira, J. M. 1992. *O Teste de Cartas de Circuito Impresso com BST: Arquitectura de um Controlador Residente, e Geração Automática do Programa de Teste*. Dissertação de Doutoramento, Universidade do Porto, Abril 1992
- Ferreira, J. M., Matos, J. S. and Jong, F. 1992. *Verification of TPG Procedures for BST Boards with Additional ST-Clusters*. Report R4.4, ESPRIT Project 2478 (Research into Boundary Scan Test Implementation), Março 1992.
- IEEE Std 1149.1 1990. *IEEE Standard Test Access Port and Boundary Scan Architecture*. IEEE Standards Board, May 1990.
- Tavares, J. A., Alves, G. R., Gericota, M. G., Ramalho, J. L., Pinto, F. S. e Ferreira, J. M. 1993. *Software: Implementação e Verificação*. Relatório nº4, Projecto JNICT PMCT/C/TIT/937/90 (Sistema de Validação e Teste de Cartas de Circuito Impresso com BST), Janeiro 1993.

## **6. ANEXO**

**Ficheiros de Descrição da PLD Desenvolvida**

MAX+plus II Compiler Report File  
Version 2.13 06/02/92

\*\*\*\*\* Project compiled without errors

Title: Procv2--epld  
Company: FEUP / INESC  
Rev: A  
Date: 11:43a 1-12-1993

\*\* DEVICE SUMMARY \*\*

Chip/ POF	Device	Input Pins	Output Pins	Bidir Pins	MCs	Shareable EXPs	% Utilized
procepld	EPM5128JC-1	15	34	0	127	158	99 %

\*\* PIN/MC/CHIP ASSIGNMENTS \*\*

User Assignments	Actual Assignments (if different)	Node Name
procepld@MC80		accum:77 acc_bits0
procepld@MC79		accum:77 acc_bits1
procepld@MC78		accum:77 acc_bits2
procepld@MC77		accum:77 acc_bits3
procepld@MC128		accum:77 acc_bits4
procepld@MC127		accum:77 acc_bits5
procepld@MC76		accum:77 acc_bits6
procepld@MC75		accum:77 acc_bits7
procepld@57		A0
procepld@56		A1
procepld@55		A2
procepld@53		A3
procepld@45		A4
procepld@44		A5
procepld@43		A6
procepld@42		A7
procepld@41		A8
procepld@40		A9
procepld@39		A10
procepld@38		A11
procepld@65		A12
procepld@64		A13
procepld@63		A14
procepld@62		A15
procepld@61		A16
procepld@60		A17
procepld@59		A18
procepld@58		A19
procepld@68		clock
procepld@MC16		cntclken
procepld@MC48		cnt24eq0
procepld@MC96		count:63 cnt_bits0
procepld@MC95		count:63 cnt_bits1

```

procepld@MC94           |count:63|cnt_bits2
procepld@MC93           |count:63|cnt_bits3
procepld@MC92           |count:63|cnt_bits4
procepld@MC91           |count:63|cnt_bits5
procepld@MC90           |count:63|cnt_bits6
procepld@MC89           |count:63|cnt_bits7
procepld@MC88           |count:63|cnt_bits8
procepld@MC87           |count:63|cnt_bits9
procepld@MC47           |count:63|cnt_bits10
procepld@MC46           |count:63|cnt_bits11
procepld@MC45           |count:63|cnt_bits12
procepld@MC44           |count:63|cnt_bits13
procepld@MC43           |count:63|cnt_bits14
procepld@MC42           |count:63|cnt_bits15
procepld@MC41           |count:63|cnt_bits16
procepld@MC40           |count:63|cnt_bits17
procepld@MC39           |count:63|cnt_bits18
procepld@MC38           |count:63|cnt_bits19
procepld@MC37           |count:63|cnt_bits20
procepld@MC36           |count:63|cnt_bits21
procepld@MC35           |count:63|cnt_bits22
procepld@MC34           |count:63|cnt_bits23
procepld@MC33           |count:63|:70
procepld@51             DESER
procepld@66             D0
procepld@36             D1
procepld@35             D2
procepld@34             D3
procepld@32             D4
procepld@2              D5
procepld@11             D6
procepld@10             D7
procepld@31             EOT
procepld@49             ERROR
procepld@MC64           |insreg:50|opcode0
procepld@MC126          |insreg:50|opcode1
procepld@MC112          |insreg:50|opcode2
procepld@MC63           |insreg:50|opcode3
procepld@MC62           |insreg:50|opcode4
procepld@MC125          pcclken
procepld@MC124          |prgcnt:78|latch_bits0
procepld@MC123          |prgcnt:78|latch_bits1
procepld@MC122          |prgcnt:78|latch_bits2
procepld@MC121          |prgcnt:78|latch_bits3
procepld@MC111          |proc fsm:79|Q0
procepld@MC61           |proc fsm:79|Q1
procepld@MC110          |proc fsm:79|Q2
procepld@MC74            |proc fsm:79|Q3
procepld@MC32           |proc fsm:79|Q4
procepld@MC31           |proc fsm:79|Q5
procepld@1               reset
procepld@MC109          |scanin:76|acc3_bits0
procepld@MC108          |scanin:76|acc3_bits1
procepld@MC107          |scanin:76|acc3_bits2
procepld@MC106          |scanin:76|acc3_bits3
procepld@MC115          |scanin:76|acc3_bits4
procepld@MC30            |scanin:76|acc3_bits5
procepld@MC73            |scanin:76|acc3_bits6
procepld@MC114          |scanin:76|acc3_bits7
procepld@MC113          |scanin:76|sr2_bits0
procepld@MC112          |scanin:76|sr2_bits1

```

```

procepld@MC11           | scanin:76|sr2_bits2
procepld@MC10           | scanin:76|sr2_bits3
procepld@MC9            | scanin:76|sr2_bits4
procepld@MC8            | scanin:76|sr2_bits5
procepld@MC7            | scanin:76|sr2_bits6
procepld@MC6            | scanin:76|sr2_bits7
procepld@MC86           | scanin:76|sr3_bits0
procepld@MC83           | scanin:76|sr3_bits1
procepld@MC29           | scanin:76|sr3_bits2
procepld@MC28           | scanin:76|sr3_bits3
procepld@MC27           | scanin:76|sr3_bits4
procepld@MC26           | scanin:76|sr3_bits5
procepld@MC5            | scanin:76|sr3_bits6
procepld@MC4            | scanin:76|sr3_bits7
procepld@MC105          | scanout:66|acc1_bits0
procepld@MC104          | scanout:66|acc1_bits1
procepld@MC103          | scanout:66|acc1_bits2
procepld@MC25           | scanout:66|acc1_bits3
procepld@MC24           | scanout:66|acc1_bits4
procepld@MC23           | scanout:66|acc1_bits5
procepld@MC22           | scanout:66|acc1_bits6
procepld@MC3             | scanout:66|acc1_bits7
procepld@MC60            | scanout:66|sr1_bits0
procepld@MC59            | scanout:66|sr1_bits1
procepld@MC58            | scanout:66|sr1_bits2
procepld@MC21            | scanout:66|sr1_bits3
procepld@MC20            | scanout:66|sr1_bits4
procepld@MC19            | scanout:66|sr1_bits5
procepld@MC18            | scanout:66|sr1_bits6
procepld@MC2             | scanout:66|sr1_bits7
procepld@MC17            | sr1cken
procepld@MC57            | sr1clken
procepld@30              | SSA
procepld@52              | SSB
procepld@47              | TAPSLD
procepld@4               | TCK0
procepld@46              | TCK1
procepld@9               | TDI0
procepld@8               | TDI1
procepld@29              | TDO0
procepld@28              | TDO1
procepld@27              | TMS0
procepld@26              | TMS1
procepld@25              | TRST0
procepld@24              | TRST1
procepld@7               | WSA
procepld@6               | WSB

```

\*\* STATE MACHINE ASSIGNMENTS \*\*

```

|proc fsm:79|proc fsm: MACHINE
  OF BITS
  (|proc fsm:79|Q5, |proc fsm:79|Q4, |proc fsm:79|Q3, |proc fsm:79|Q2, |proc fsm:79|Q1,
  |proc fsm:79|Q0 )
    WITH STATES (
      S0 = B"000000",
      S1 = B"000001",
      S2 = B"000011",
      S3 = B"001110",

```

```
S4 = B"000110",
S5 = B"000010",
S6 = B"001010",
S7 = B"010010",
S8 = B"101010",
S9 = B"100110",
S10 = B"100100",
S11 = B"000100",
S12 = B"001100",
S13 = B"001000",
S14 = B"011000",
S15 = B"010100",
S16 = B"100000",
S17 = B"010000"
};
```

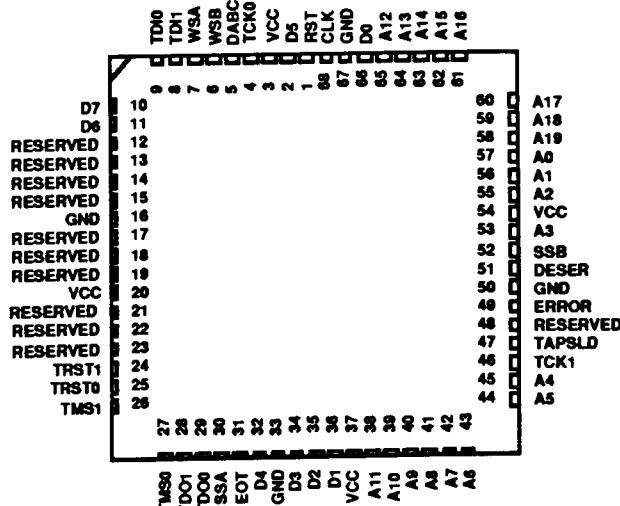
\*\* FILE HIERARCHY \*\*

```
insreg:50|  
tapsel:83|  
proc fsm:79|  
prgcnt:78|  
accum:77|  
scanin:76|  
scanout:66|  
count:63|  
status:60|  
syncout:54|
```

\*\*\*\*\* Logic for device 'procepld' compiled without errors

Device: EPM5128JC-1

Security: OFF



## \*\* RESOURCE USAGE \*\*

Logic Array Block	Macrocells	I/O Pins	Shareable Expanders	External Interconnect
A: MC1 - MC16	16/16(100%)	8/ 8(100%)	32/32(100%)	24/24(100%)
B: MC17 - MC32	16/16(100%)	0/ 5( 0%)	32/32(100%)	21/24( 87%)
C: MC33 - MC48	16/16(100%)	0/ 5( 0%)	6/32( 18%)	24/24(100%)
D: MC49 - MC64	16/16(100%)	8/ 8(100%)	30/32( 93%)	22/24( 91%)
E: MC65 - MC80	16/16(100%)	8/ 8(100%)	26/32( 81%)	22/24( 91%)
F: MC81 - MC96	16/16(100%)	4/ 5( 80%)	30/32( 93%)	22/24( 91%)
G: MC97 - MC112	15/16( 93%)	5/ 5(100%)	32/32(100%)	15/24( 62%)
H: MC113 - MC128	16/16(100%)	8/ 8(100%)	23/32( 71%)	24/24(100%)

Total dedicated input pins used:	8/ 8 (100%)
Total I/O pins used:	41/ 52 ( 78%)
Total macrocells used:	127/128 ( 99%)
Total shareable expanders used:	211/256 ( 82%)
Total input pins required:	15
Total output pins required:	34
Total bidirectional pins required:	0
Total macrocells required:	127
Total shareable expanders in database:	158
Synthesized macrocells:	0/128 ( 0%)

## \*\* INPUTS \*\*

Pin	MCell	LAB	Primitive	Shareable Expanders		Fan-In		
				Total	Shared	INP	FBK	Name
68	-	-	INPUT	0	0	0	0	clock
5	(2)	(A)	INPUT	0	0	0	0	DABC
66	-	-	INPUT	0	0	0	0	D0
36	-	-	INPUT	0	0	0	0	D1
35	-	-	INPUT	0	0	0	0	D2
34	-	-	INPUT	0	0	0	0	D3
32	-	-	INPUT	0	0	0	0	D4
2	-	-	INPUT	0	0	0	0	D5
11	(8)	(A)	INPUT	0	0	0	0	D6
10	(7)	(A)	INPUT	0	0	0	0	D7
1	-	-	INPUT	0	0	0	0	reset
9	(6)	(A)	INPUT	0	0	0	0	TDI0
8	(5)	(A)	INPUT	0	0	0	0	TDI1
7	(4)	(A)	INPUT	0	0	0	0	WSA
6	(3)	(A)	INPUT	0	0	0	0	WSB

## \*\* OUTPUTS \*\*

Pin	MCell	LAB	Primitive	Shareable Expanders		Fan-In		
				Total	Shared	INP	FBK	Name
57	101	G	DFF	1	1	3	11	A0
56	100	G	DFF	1	1	3	12	A1

55	99	G	DFF	1	1	3	13	A2
53	98	G	DFF	1	1	3	14	A3
45	72	E	DFF	1	1	3	15	A4
44	71	E	DFF	1	1	3	16	A5
43	70	E	DFF	1	1	3	17	A6
42	69	E	DFF	1	1	3	18	A7
41	68	E	DFF	1	1	2	20	A8
40	67	E	DFF	1	1	2	21	A9
39	66	E	DFF	1	1	2	22	A10
38	65	E	DFF	1	1	2	23	A11
65	120	H	DFF	1	1	2	24	A12
64	119	H	DFF	1	1	2	25	A13
63	118	H	DFF	1	1	2	26	A14
62	117	H	DFF	1	1	2	27	A15
61	116	H	DFF	1	1	2	28	A16
60	115	H	DFF	1	1	2	29	A17
59	114	H	DFF	1	1	2	30	A18
58	113	H	DFF	1	1	2	31	A19
51	85	F	OUTPUT	4	4	0	11	DESER
31	56	D	DFF	0	0	2	11	EOT
49	84	F	DFF	3	0	4	4	ERROR
30	55	D	DFF	1	0	2	11	SSA
52	97	G	DFF	1	0	2	11	SSB
47	82	F	DFF	1	0	2	11	TAPSLD
4	1	A	WIRE	12	0	2	12	TCK0
46	81	F	WIRE	12	0	2	12	TCK1
29	54	D	WIRE	0	0	1	2	TDO0
28	53	D	WIRE	0	0	1	2	TDO1
27	52	D	WIRE	0	0	1	12	TMS0
26	51	D	WIRE	0	0	1	12	TMS1
25	50	D	DFF	0	0	3	12	TRST0
24	49	D	DFF	0	0	3	12	TRST1

## \*\* BURIED LOGIC \*\*

Pin	MCell	LAB	Primitive	Shareable		Fan-In	Name
				Total	Shared		
-	80	E	DFF	4	4	3	accum:77 acc_bits0
-	79	E	DFF	4	4	3	accum:77 acc_bits1
-	78	E	DFF	4	4	3	accum:77 acc_bits2
-	77	E	DFF	4	4	3	accum:77 acc_bits3
-	128	H	DFF	4	4	3	accum:77 acc_bits4
-	127	H	DFF	4	4	3	accum:77 acc_bits5
-	76	E	DFF	4	4	3	accum:77 acc_bits6
-	75	E	DFF	4	4	3	accum:77 acc_bits7
-	16	A	MCELL	11	2	0	cntclken
-	48	C	MCELL	0	0	8	cnt24eq0
-	96	F	DFF	6	6	3	count:63 cnt_bits0
-	95	F	DFF	6	6	3	count:63 cnt_bits1
-	94	F	DFF	6	6	3	count:63 cnt_bits2
-	93	F	DFF	6	6	3	count:63 cnt_bits3
-	92	F	DFF	6	6	3	count:63 cnt_bits4
-	91	F	DFF	6	6	3	count:63 cnt_bits5
-	90	F	DFF	6	6	3	count:63 cnt_bits6
-	89	F	DFF	6	6	3	count:63 cnt_bits7
-	88	F	DFF	6	6	3	count:63 cnt_bits8
-	87	F	DFF	6	6	3	count:63 cnt_bits9
-	47	C	DFF	6	6	3	count:63 cnt_bits10

-	46	C	DFF	6	6	3	23	count:63 cnt_bits11
-	45	C	DFF	6	6	3	24	count:63 cnt_bits12
-	44	C	DFF	6	6	3	25	count:63 cnt_bits13
-	43	C	DFF	6	6	3	26	count:63 cnt_bits14
-	42	C	DFF	6	6	3	27	count:63 cnt_bits15
-	41	C	DFF	5	5	3	28	count:63 cnt_bits16
-	40	C	DFF	5	5	3	29	count:63 cnt_bits17
-	39	C	DFF	5	5	3	30	count:63 cnt_bits18
-	38	C	DFF	5	5	3	31	count:63 cnt_bits19
(23)	37	C	DFF	5	5	3	32	count:63 cnt_bits20
(22)	36	C	DFF	5	5	3	33	count:63 cnt_bits21
(21)	35	C	DFF	5	5	3	34	count:63 cnt_bits22
(19)	34	C	DFF	5	5	3	35	count:63 cnt_bits23
(18)	33	C	MCELL	0	0	0	16	count:63 :70
-	64	D	DFF	0	0	3	6	insreg:50 opcode0
-	126	H	DFF	0	0	3	6	insreg:50 opcode1
-	112	G	DFF	0	0	3	6	insreg:50 opcode2
-	63	D	DFF	0	0	3	6	insreg:50 opcode3
-	62	D	DFF	0	0	3	6	insreg:50 opcode4
-	125	H	MCELL	17	3	0	11	pcclken
-	124	H	DFF	1	1	3	10	
prgcnt:78 latch_bits0								
-	123	H	DFF	1	1	3	10	
prgcnt:78 latch_bits1								
-	122	H	DFF	1	1	3	10	
prgcnt:78 latch_bits2								
-	121	H	DFF	1	1	3	10	
prgcnt:78 latch_bits3								
-	111	G	DFF	14	3	2	16	proc fsm:79 Q0
-	61	D	DFF	26	4	4	16	proc fsm:79 Q1
-	110	G	DFF	11	6	2	15	proc fsm:79 Q2
-	74	E	DFF	19	5	2	15	proc fsm:79 Q3
-	32	B	DFF	16	1	2	15	proc fsm:79 Q4
-	31	B	DFF	3	0	2	15	proc fsm:79 Q5
-	109	G	DFF	3	3	3	11	
scanin:76 acc3_bits0								
-	108	G	DFF	3	3	3	11	
scanin:76 acc3_bits1								
-	107	G	DFF	3	3	3	11	
scanin:76 acc3_bits2								
-	106	G	DFF	3	3	3	11	
scanin:76 acc3_bits3								
-	15	A	DFF	3	3	3	11	
scanin:76 acc3_bits4								
-	30	B	DFF	3	3	3	11	
scanin:76 acc3_bits5								
-	73	E	DFF	3	3	3	11	
scanin:76 acc3_bits6								
-	14	A	DFF	3	3	3	11	
scanin:76 acc3_bits7								
-	13	A	DFF	3	3	2	14	scanin:76 sr2_bits0
-	12	A	DFF	3	3	2	14	scanin:76 sr2_bits1
-	11	A	DFF	3	3	2	14	scanin:76 sr2_bits2
-	10	A	DFF	3	3	2	14	scanin:76 sr2_bits3
-	9	A	DFF	3	3	2	14	scanin:76 sr2_bits4
(11)	8	A	DFF	3	3	2	14	scanin:76 sr2_bits5
(10)	7	A	DFF	3	3	2	14	scanin:76 sr2_bits6
(9)	6	A	DFF	3	3	2	12	scanin:76 sr2_bits7
-	86	F	DFF	3	3	2	14	scanin:76 sr3_bits0
(48)	83	F	DFF	3	3	2	14	scanin:76 sr3_bits1
-	29	B	DFF	3	3	2	14	scanin:76 sr3_bits2

-	28	B	DFF	3	3	2	14	scanin:76 sr3_bits3
-	27	B	DFF	3	3	2	14	scanin:76 sr3_bits4
-	26	B	DFF	3	3	2	14	scanin:76 sr3_bits5
(8)	5	A	DFF	3	3	2	14	scanin:76 sr3_bits6
(7)	4	A	DFF	3	3	2	12	scanin:76 sr3_bits7
-	105	G	DFF	4	4	3	11	
scanout:66 acc1_bits0	-	104	G	DFF	4	4	3	11
scanout:66 acc1_bits1	-	103	G	DFF	4	4	3	11
scanout:66 acc1_bits2	-	25	B	DFF	4	4	3	11
scanout:66 acc1_bits3	-	24	B	DFF	4	4	3	11
scanout:66 acc1_bits4	-	23	B	DFF	4	4	3	11
scanout:66 acc1_bits5	-	22	B	DFF	4	4	3	11
scanout:66 acc1_bits6	(6)	3	A	DFF	4	4	3	11
scanout:66 acc1_bits7	-	60	D	DFF	3	3	2	15
scanout:66 sr1_bits0	-	59	D	DFF	3	3	2	15
scanout:66 sr1_bits1	-	58	D	DFF	3	3	2	15
scanout:66 sr1_bits2	(17)	21	B	DFF	3	3	2	15
scanout:66 sr1_bits3	(15)	20	B	DFF	3	3	2	15
scanout:66 sr1_bits4	(14)	19	B	DFF	3	3	2	15
scanout:66 sr1_bits5	(13)	18	B	DFF	3	3	2	15
scanout:66 sr1_bits6	(5)	2	A	DFF	3	3	2	12
scanout:66 sr1_bits7	(12)	17	B	MCELL	4	4	0	11 sr1cken
-	57	D	MCELL		0	0	0	11 sr1cken

**DESIGN IS "procepld"**

```

BEGIN
  DEVICE "procepld" IS "EPM5128JC-1"
  BEGIN
    clock          @ 68 : INPUT ;
    DABC          @ 5  : INPUT ; %MC2%
    D0            @ 66 : INPUT ;
    D1            @ 36 : INPUT ;
    D2            @ 35 : INPUT ;
    D3            @ 34 : INPUT ;
    D4            @ 32 : INPUT ;
    D5            @ 2  : INPUT ;
    D6            @ 11 : INPUT ; %MC8%
    D7            @ 10 : INPUT ; %MC7%
    reset          @ 1  : INPUT ;
    TDI0          @ 9  : INPUT ; %MC6%
    TDI1          @ 8  : INPUT ; %MC5%
  
```

WSA	@ 7	:	INPUT	;	%MC4%
WSB	@ 6	:	INPUT	;	%MC3%
A0	@ 57	:	OUTPUT	;	%MC101%
A1	@ 56	:	OUTPUT	;	%MC100%
A2	@ 55	:	OUTPUT	;	%MC99%
A3	@ 53	:	OUTPUT	;	%MC98%
A4	@ 45	:	OUTPUT	;	%MC72%
A5	@ 44	:	OUTPUT	;	%MC71%
A6	@ 43	:	OUTPUT	;	%MC70%
A7	@ 42	:	OUTPUT	;	%MC69%
A8	@ 41	:	OUTPUT	;	%MC68%
A9	@ 40	:	OUTPUT	;	%MC67%
A10	@ 39	:	OUTPUT	;	%MC66%
A11	@ 38	:	OUTPUT	;	%MC65%
A12	@ 65	:	OUTPUT	;	%MC120%
A13	@ 64	:	OUTPUT	;	%MC119%
A14	@ 63	:	OUTPUT	;	%MC118%
A15	@ 62	:	OUTPUT	;	%MC117%
A16	@ 61	:	OUTPUT	;	%MC116%
A17	@ 60	:	OUTPUT	;	%MC115%
A18	@ 59	:	OUTPUT	;	%MC114%
A19	@ 58	:	OUTPUT	;	%MC113%
DESER	@ 51	:	OUTPUT	;	%MC85%
EOT	@ 31	:	OUTPUT	;	%MC56%
ERROR	@ 49	:	OUTPUT	;	%MC84%
SSA	@ 30	:	OUTPUT	;	%MC55%
SSB	@ 52	:	OUTPUT	;	%MC97%
TAPSLD	@ 47	:	OUTPUT	;	%MC82%
TCK0	@ 4	:	OUTPUT	;	%MC1%
TCK1	@ 46	:	OUTPUT	;	%MC81%
TDO0	@ 29	:	OUTPUT	;	%MC54%
TDO1	@ 28	:	OUTPUT	;	%MC53%
TMS0	@ 27	:	OUTPUT	;	%MC52%
TMS1	@ 26	:	OUTPUT	;	%MC51%
TRST0	@ 25	:	OUTPUT	;	%MC50%
TRST1	@ 24	:	OUTPUT	;	%MC49%
accum:77 acc_bits0	@ MC80	:	BURIED	;	
accum:77 acc_bits1	@ MC79	:	BURIED	;	
accum:77 acc_bits2	@ MC78	:	BURIED	;	
accum:77 acc_bits3	@ MC77	:	BURIED	;	
accum:77 acc_bits4	@ MC128	:	BURIED	;	
accum:77 acc_bits5	@ MC127	:	BURIED	;	
accum:77 acc_bits6	@ MC76	:	BURIED	;	
accum:77 acc_bits7	@ MC75	:	BURIED	;	
cntclken	@ MC16	:	BURIED	;	
cnt24eq0	@ MC48	:	BURIED	;	
count:63 cnt_bits0	@ MC96	:	BURIED	;	
count:63 cnt_bits1	@ MC95	:	BURIED	;	
count:63 cnt_bits2	@ MC94	:	BURIED	;	
count:63 cnt_bits3	@ MC93	:	BURIED	;	
count:63 cnt_bits4	@ MC92	:	BURIED	;	
count:63 cnt_bits5	@ MC91	:	BURIED	;	
count:63 cnt_bits6	@ MC90	:	BURIED	;	
count:63 cnt_bits7	@ MC89	:	BURIED	;	
count:63 cnt_bits8	@ MC88	:	BURIED	;	
count:63 cnt_bits9	@ MC87	:	BURIED	;	
count:63 cnt_bits10	@ MC47	:	BURIED	;	
count:63 cnt_bits11	@ MC46	:	BURIED	;	
count:63 cnt_bits12	@ MC45	:	BURIED	;	
count:63 cnt_bits13	@ MC44	:	BURIED	;	
count:63 cnt_bits14	@ MC43	:	BURIED	;	

```

|count:63|cnt_bits15          @ MC42   : BURIED ;
|count:63|cnt_bits16          @ MC41   : BURIED ;
|count:63|cnt_bits17          @ MC40   : BURIED ;
|count:63|cnt_bits18          @ MC39   : BURIED ;
|count:63|cnt_bits19          @ MC38   : BURIED ;
|count:63|cnt_bits20          @ MC37   : BURIED ; %PIN 23%
|count:63|cnt_bits21          @ MC36   : BURIED ; %PIN 22%
|count:63|cnt_bits22          @ MC35   : BURIED ; %PIN 21%
|count:63|cnt_bits23          @ MC34   : BURIED ; %PIN 19%
|count:63|cnt_bits24          @ MC33   : BURIED ; %PIN 18%
|count:63|:70                @ MC64   : BURIED ;
|insreg:50|opcode0           @ MC126  : BURIED ;
|insreg:50|opcode1           @ MC112  : BURIED ;
|insreg:50|opcode2           @ MC63   : BURIED ;
|insreg:50|opcode3           @ MC62   : BURIED ;
|insreg:50|opcode4           @ MC125  : BURIED ;
pcclken
|prgcnt:78|latch_bits0       @ MC124  : BURIED ;
|prgcnt:78|latch_bits1       @ MC123  : BURIED ;
|prgcnt:78|latch_bits2       @ MC122  : BURIED ;
|prgcnt:78|latch_bits3       @ MC121  : BURIED ;
|proc fsm:79|Q0              @ MC111  : BURIED ;
|proc fsm:79|Q1              @ MC61   : BURIED ;
|proc fsm:79|Q2              @ MC110  : BURIED ;
|proc fsm:79|Q3              @ MC74   : BURIED ;
|proc fsm:79|Q4              @ MC32   : BURIED ;
|proc fsm:79|Q5              @ MC31   : BURIED ;
|scanin:76|acc3_bits0        @ MC109  : BURIED ;
|scanin:76|acc3_bits1        @ MC108  : BURIED ;
|scanin:76|acc3_bits2        @ MC107  : BURIED ;
|scanin:76|acc3_bits3        @ MC106  : BURIED ;
|scanin:76|acc3_bits4        @ MC15   : BURIED ;
|scanin:76|acc3_bits5        @ MC30   : BURIED ;
|scanin:76|acc3_bits6        @ MC73   : BURIED ;
|scanin:76|acc3_bits7        @ MC14   : BURIED ;
|scanin:76|sr2_bits0         @ MC13   : BURIED ;
|scanin:76|sr2_bits1         @ MC12   : BURIED ;
|scanin:76|sr2_bits2         @ MC11   : BURIED ;
|scanin:76|sr2_bits3         @ MC10   : BURIED ;
|scanin:76|sr2_bits4         @ MC9    : BURIED ;
|scanin:76|sr2_bits5         @ MC8    : BURIED ; %PIN 11%
|scanin:76|sr2_bits6         @ MC7    : BURIED ; %PIN 10%
|scanin:76|sr2_bits7         @ MC6    : BURIED ; %PIN 9%
|scanin:76|sr3_bits0         @ MC86   : BURIED ;
|scanin:76|sr3_bits1         @ MC83   : BURIED ; %PIN 48%
|scanin:76|sr3_bits2         @ MC29   : BURIED ;
|scanin:76|sr3_bits3         @ MC28   : BURIED ;
|scanin:76|sr3_bits4         @ MC27   : BURIED ;
|scanin:76|sr3_bits5         @ MC26   : BURIED ;
|scanin:76|sr3_bits6         @ MC5    : BURIED ; %PIN 8%
|scanin:76|sr3_bits7         @ MC4    : BURIED ; %PIN 7%
|scanout:66|acc1_bits0        @ MC105  : BURIED ;
|scanout:66|acc1_bits1        @ MC104  : BURIED ;
|scanout:66|acc1_bits2        @ MC103  : BURIED ;
|scanout:66|acc1_bits3        @ MC25   : BURIED ;
|scanout:66|acc1_bits4        @ MC24   : BURIED ;
|scanout:66|acc1_bits5        @ MC23   : BURIED ;
|scanout:66|acc1_bits6        @ MC22   : BURIED ;
|scanout:66|acc1_bits7        @ MC3    : BURIED ; %PIN 6%
|scanout:66|sr1_bits0         @ MC60   : BURIED ;
|scanout:66|sr1_bits1         @ MC59   : BURIED ;
|scanout:66|sr1_bits2         @ MC58   : BURIED ;

```

```

| scanout:66|sr1_bits3      @ MC21 : BURIED ; %PIN 17%
| scanout:66|sr1_bits4      @ MC20 : BURIED ; %PIN 15%
| scanout:66|sr1_bits5      @ MC19 : BURIED ; %PIN 14%
| scanout:66|sr1_bits6      @ MC18 : BURIED ; %PIN 13%
| scanout:66|sr1_bits7      @ MC2  : BURIED ; %PIN 5%
sr1cken          @ MC17 : BURIED ; %PIN 12%
sr1cken          @ MC57 : BURIED ;

END;
END;

%***** Instruction register *****
**           Instruction register
** *****%


TITLE "instruction register";

SUBDESIGN insreg
(
    reset      : INPUT;      % system reset %
    clock      : INPUT;      % system clock %
    ircken     : INPUT;      % load the instruction register %
    Data[4..0]   : INPUT;      % data bus bits D4 to D0 %

    op_code[4..0] : OUTPUT;    % instruction register outputs %

)

VARIABLE
opcode[4..0]      : DFF;      % FF outputs %

BEGIN

opcode[].prn = reset;
opcode[].clk = !clock;

IF (ircken) THEN
    opcode[4..0] = Data[4..0];
ELSE
    opcode[4..0] = opcode[4..0];
END IF;

op_code[4..0] = opcode[4..0];

END;

%***** Test Access Port selection *****
**           Test Access Port selection
** *****%


TITLE "TAP selection";

SUBDESIGN tapsel
(
    reset      : INPUT;      % system reset %

```

```

clock      : INPUT;      % system clock %
DABC       : INPUT;      % direct access to boundary chain %
tsffclken : INPUT;      % test selection ff clock enable %
op_code0   : INPUT;      % instruction register bit 0 %

TDO        : INPUT;      % Test Data Output from the ScanOut block %
TMS        : INPUT;      % Test Mode Select %
TCK        : INPUT;      % Test Clock %
TRST       : INPUT;      % Test Reset %
TDI0,TDI1  : INPUT;      % Teste Data Input, TAP 0 and TAP 1 %

TDO0       : OUTPUT;     % TAP0 Test Data Output %
TMS0       : OUTPUT;     % TAP0 Test Mode Select %
TCK0       : OUTPUT;     % TAP0 Test Clock %
TRST0      : OUTPUT;     % TAP0 Test Reset %

TDO1       : OUTPUT;     % TAP1 Test Data Output %
TMS1       : OUTPUT;     % TAP1 Test Mode Select %
TCK1       : OUTPUT;     % TAP1 Test Clock %
TRST1      : OUTPUT;     % TAP1 Test Reset %

TDI        : OUTPUT;     % Test Data Input to the ScanIn block %

TAPSID    : OUTPUT;     % Test Access Port Selected %

)

```

## VARIABLE

```

ts          : DFF;
trstout[1..0] : DFF;

tdoout1,tdoout0,tmsout1,tmsout0,tckout1,tckout0 : NODE;

```

## BEGIN

```

ts.clrn = reset;
ts.clk = !clock;
trstout[1..0].prn = reset;
trstout[1..0].clk = clock;

IF (tsffclken) THEN
  ts = op_code0;
ELSE
  ts = ts;
END IF;

IF (ts) THEN
  trstout1 = TRST;
  tdoout1 = TDO;
  tmsout1 = TMS;
  tckout1 = TCK;
  TDI = TDII;
  trstout0 = VCC;
  tmsout0 = GND;
  tdoout0 = GND;
  tckout0 = GND;
ELSE
  trstout0 = TRST;
  tdoout0 = TDO;
  tmsout0 = TMS;
  tckout0 = TCK;

```

```

    TDI = TDI0;
    trstout1 = VCC;
    tmsout1 = GND;
    tdoout1 = GND;
    tckout1 = GND;
END IF;

TAPSLD = ts;
TRST1 = TRI(trstout1.q,!DABC);
TRST0 = TRI(trstout0.q,!DABC);
TCK1 = TRI(tckout1,!DABC);
TCK0 = TRI(tckout0,!DABC);
TMS1 = TRI(tmsout1,!DABC);
TMS0 = TRI(tmsout0,!DABC);
TDO1 = TRI(tdoout1,!DABC);
TDO0 = TRI(tdoout0,!DABC);

END;

*****
**           Instruction decode and execution control unit      **
*****
```

TITLE "control unit";

% opcodes 0000X correspond to TMS output control %

% opcodes 0001X correspond to counters operations %

% opcodes 0010X correspond to shift operations %

% opcodes 0011X correspond to jump operations %

% opcode 10000 is reserved for the HALT operation %

% opcode 10001 is reserved for the TRST operation %

% opcodes 1001X to 101XX and 1100X correspond to synchronism operations %

```

CONSTANT WSA      = B"1010";
CONSTANT WSA0     = B"10100";
CONSTANT WSA1     = B"10101";
CONSTANT WSB      = B"1011";
CONSTANT WSB0     = B"10110";
CONSTANT WSB1     = B"10111";

% opcode 1101X correspond to select TAP operations %
CONSTANT SELTAP   = B"1101";
CONSTANT SELTAP0  = B"11010";
CONSTANT SELTAP1  = B"11011";

% opcode 11100 is reserved for the NTCK operation %
CONSTANT NTCK     = B"11100";

SUBDESIGN proc fsm
(
    reset          : INPUT;           % system reset %
    clock          : INPUT;           % system clock %
    error          : INPUT;           % matching error %
    cnt16eq0       : INPUT;           % true when the counter16 contents are zero %
    cnt16eq1       : INPUT;           % true when the counter16 contents are one %
    cnt16leq8      : INPUT;           % true when the counter16 contents are less
                                    or equal to eight %
    cnt24eq0       : INPUT;           % true when the counter24 contents are zero %
    syncinpA        : INPUT;           % external synchronization input channel A %
    syncinpB        : INPUT;           % external synchronization input channel B %
    op_code[4..0]    : INPUT;           % contents of the instruction register %

    irclken        : OUTPUT;          % load the instruction register %
    ldpc            : OUTPUT;          % load the program counter %
    pcclken        : OUTPUT;          % increment the program counter %
    pclt2clken    : OUTPUT;          % load the PC most significant byte %
    ldcnt16lsb     : OUTPUT;          % load the counter16 least significant byte %
    ldcnt16msb     : OUTPUT;          % load the counter16 most significant byte %
    ldcnt24         : OUTPUT;          % load the counter24 most significant byte %
    cntclken       : OUTPUT;          % decrement counter %
    acc1clken      : OUTPUT;          % load the accumulator1 byte %
    acc3clken      : OUTPUT;          % load the accumulator3 byte %
    gpaccclken    : OUTPUT;          % load the general purpose accumulator byte %
    ldsr            : OUTPUT;          % load the shift register %
    srlclken       : OUTPUT;          % one shift-register1 bit to the right %
    srclken         : OUTPUT;          % one shift-register2and3 bit to the right %
    tsffclken      : OUTPUT;          % change the selected TAP %
    eotffclken    : OUTPUT;          % flags the end of test %
    syncaclken     : OUTPUT;          % update the identified output bit %
    syncbclken     : OUTPUT;          % update the identified output bit %
    deseren         : OUTPUT;          % update the deserializer enable output %
    tmsout          : OUTPUT;          % update the tms output %
    trstout         : OUTPUT;          % update the trst output %
    tck              : OUTPUT;          % update the tck output %
    saidas[5..0]    : OUTPUT;          % output from the state machine %

)

```

## VARIABLE

```
% State machine declaration %
```

```

proc fsm: MACHINE OF BITS (Q[5..0])
    WITH STATES (
        S0,
        S1,
        S2,
        S3,
        S4,
        S5,
        S6,
        S7,
        S8,
        S9,
        S10,
        S11,
        S12,
        S13,
        S14,
        S15,
        S16,
        S17);

sinal1,sinal2,sinal3,sinal4,sinal5,sinal6,sinal7 : NODE ;
sinal8,sinal9,sinal10,sinal11,sinal12,sinal13,sinal14 : NODE ;
syncaclken_n, syncbclken_n, tsffclken_n : NODE ;
pclt2clken_n, ldpc_n, ldcnt24_n : NODE ;

BEGIN

    DEFAULTS
        irclken      = GND;
        pcclken      = GND;
        cntclken     = GND;
        srlclken     = GND;
        srclken      = GND;
        deseren      = GND;
        tmsout       = GND;
        tck          = GND;

    END DEFAULTS;

    proc fsm.reset = !reset;
    proc fsm.clk = clock;

    % State transition and outputs definition %

CASE (proc fsm) IS

    WHEN S0 =>
        irclken = VCC;
        IF (op_code[] == NTCK & cnt24eq0 & cnt16eq0) THEN
            proc fsm = S2;
        ELSE
            proc fsm = S1;
        END IF;

    WHEN S1 =>
        IF (op_code[4..1] == TMS) THEN
            pcclken = VCC;
            !tck = clock;
            tmsout = op_code[0];
            proc fsm = S0;

```

```

ELSIF (op_code[4..1] == LDC) THEN
    pcclken = VCC;
    proc fsm = S2;
ELSIF (op_code[] == NTCK) THEN
    cntclken = VCC;
    !tck = clock;
    IF (cnt24eq0 & cnt16eq0) THEN
        proc fsm = S2;
    ELSE
        proc fsm = S1;
    END IF;
ELSIF (op_code[] == NSHF) THEN
    pcclken = VCC;
    IF (cnt16eq0) THEN
        proc fsm = S0;
    ELSE
        proc fsm = S2;
    END IF;
ELSIF (op_code[] == NSHFCP) THEN
    pcclken = VCC;
    IF (cnt16eq0) THEN
        proc fsm = S0;
    ELSE
        proc fsm = S2;
    END IF;
ELSIF (op_code[4..1] == SSA) THEN
    pcclken = VCC;
    proc fsm = S0;
ELSIF (op_code[4..1] == WSA) THEN
    IF (syncinpa == op_code[0]) THEN
        proc fsm = S2;
    ELSE
        proc fsm = S1;
    END IF;
ELSIF (op_code[4..1] == SSB) THEN
    pcclken = VCC;
    proc fsm = S0;
ELSIF (op_code[4..1] == WSB) THEN
    IF (syncinpb == op_code[0]) THEN
        proc fsm = S2;
    ELSE
        proc fsm = S1;
    END IF;
ELSIF (op_code[] == JPE) THEN
    pcclken = VCC;
    IF (error) THEN
        proc fsm = S5;
    ELSE
        proc fsm = S2;
    END IF;
ELSIF (op_code[] == JPNE) THEN
    pcclken = VCC;
    IF (error) THEN
        proc fsm = S2;
    ELSE
        proc fsm = S5;
    END IF;
ELSIF (op_code[] == HALT) THEN
    eotffclken = VCC;
    proc fsm = S1;
ELSIF (op_code[] == TRST) THEN

```

```

tmsout = VCC;
proc fsm = S2;
ELSIF (op_code[4..1] == SELTAP) THEN
  pcclken = VCC;
  proc fsm = S0;
ELSE
  pcclken = VCC;                                % treat any other opcodes %
  proc fsm = S0;                                % as illegals                %
END IF;

WHEN S2 =>
  IF (op_code[] == LDC16) THEN
    proc fsm = S3;
  ELSIF (op_code[] == LDC24) THEN
    proc fsm = S3;
  ELSIF (op_code[] == NTCK) THEN
    pcclken = VCC;
    proc fsm = S0;
  ELSIF (op_code[4..1] == SHIFT) THEN
    proc fsm = S3;
  ELSIF (op_code[4..2] == WS) THEN
    pcclken = VCC;
    proc fsm = S0;
  ELSIF (op_code[4..1] == JP) THEN
    pcclken = VCC;
    proc fsm = S3;
  ELSIF (op_code[] == TRST) THEN
    pcclken = VCC;
    tmsout = VCC;
    proc fsm = S0;
  END IF;

WHEN S3 =>
  IF (op_code[4..1] == LDC) THEN
    pcclken = VCC;
    proc fsm = S4;
  ELSIF (op_code[] == NSHF) THEN
    pcclken = VCC;
    IF (cnt16eq1) THEN
      proc fsm = S13;
    ELSIF (!cnt16leq8) THEN
      proc fsm = S4;
    ELSE
      proc fsm = S12;
    END IF;
  ELSIF (op_code[] == NSHFCP) THEN
    pcclken = VCC;
    proc fsm = S4;
  ELSIF (op_code[4..1] == JP) THEN
    pcclken = VCC;
    proc fsm = S4;
  END IF;

WHEN S4 =>
  IF (op_code[] == LDC16) THEN
    proc fsm = S5;
  ELSIF (op_code[] == LDC24) THEN
    proc fsm = S5;
  ELSIF (op_code[] == NSHF) THEN
    cntclken = VCC;
    !tck = clock;

```

```

    srlclken = VCC;
    proc fsm = S5;
ELSIF (op_code[] == NSHFCP) THEN
    proc fsm = S5;
ELSIF (op_code[4..1] == JP) THEN
    pcclken = VCC;
    proc fsm = S0;
END IF;

WHEN S5 =>
    IF (op_code[] == LDC16) THEN
        pcclken = VCC;
        proc fsm = S0;
    ELSIF (op_code[] == LDC24) THEN
        pcclken = VCC;
        proc fsm = S6;
    ELSIF (op_code[] == NSHF) THEN
        cntclken = VCC;
        !tck = clock;
        srlclken = VCC;
        proc fsm = S6;
    ELSIF (op_code[] == NSHFCP) THEN
        pcclken = VCC;
        proc fsm = S6;
    ELSIF (op_code[4..1] == JP) THEN
        proc fsm = S6;
    END IF;

WHEN S6 =>
    IF (op_code[] == LDC24) THEN
        proc fsm = S7;
    ELSIF (op_code[] == NSHF) THEN
        cntclken = VCC;
        !tck = clock;
        srlclken = VCC;
        IF (cnt16eq1) THEN
            proc fsm = S12;
        ELSE
            proc fsm = S7;
        END IF;
    ELSIF (op_code[] == NSHFCP) THEN
        proc fsm = S7;
    ELSIF (op_code[4..1] == JP) THEN
        pcclken = VCC;
        proc fsm = S7;
    END IF;

WHEN S7 =>
    IF (op_code[] == LDC24) THEN
        pcclken = VCC;
        proc fsm = S0;
    ELSIF (op_code[] == NSHF) THEN
        cntclken = VCC;
        !tck = clock;
        srlclken = VCC;
        proc fsm = S8;
    ELSIF (op_code[] == NSHFCP) THEN
        pcclken = VCC;
        IF (cnt16eq8 & !cnt16eq1 & !cnt16eq0) THEN
            proc fsm = S16;
        ELSIF (cnt16eq1) THEN

```

```

        proc fsm = S17;
    ELSE
        proc fsm = S8;
    END IF;
ELSIF (op_code[4..1] == JP) THEN
    proc fsm = S8;
END IF;

WHEN S8 =>
    IF (op_code[] == NSHF) THEN
        cntclken = VCC;
        !tck = clock;
        sr1clken = VCC;
        proc fsm = S9;
    ELSIF (op_code[] == NSHFCP) THEN
        !tck = clock;
        sr1clken = VCC;
        cntclken = VCC;
        deseren = VCC;
        proc fsm = S9;
    ELSIF (op_code[4..1] == JP) THEN
        pcclken = VCC;
        proc fsm = S9;
    END IF;

WHEN S9 =>
    IF (op_code[] == NSHF) THEN
        cntclken = VCC;
        !tck = clock;
        sr1clken = VCC;
        proc fsm = S10;
    ELSIF (op_code[] == NSHFCP) THEN
        pcclken = VCC;
        !tck = clock;
        sr1clken = VCC;
        cntclken = VCC;
        deseren = VCC;
        proc fsm = S10;
    ELSIF (op_code[4..1] == JP) THEN
        proc fsm = S0;
    END IF;

WHEN S10 =>
    IF (op_code[] == NSHF) THEN
        cntclken = VCC;
        !tck = clock;
        sr1clken = VCC;
        proc fsm = S11;
    ELSIF (op_code[] == NSHFCP) THEN
        !tck = clock;
        sr1clken = VCC;
        cntclken = VCC;
        deseren = VCC;
        proc fsm = S11;
    END IF;

WHEN S11 =>
    IF (op_code[] == NSHF) THEN
        cntclken = VCC;
        !tck = clock;
        sr1clken = VCC;

```

```

proc fsm = S3;
ELSIF (op_code[] == NSHFCP) THEN
  pcclken = VCC;
  !tck = clock;
  srclken = VCC;
  cntclken = VCC;
  deseren = VCC;
  proc fsm = S12;
END IF;

WHEN S12 =>
  IF (op_code[] == NSHF) THEN
    cntclken = VCC;
    !tck = clock;
    srclken = VCC;
    IF (cnt16eq1) THEN
      proc fsm = S13;
    ELSE
      proc fsm = S12;
    END IF;
  ELSIF (op_code[] == NSHFCP) THEN
    !tck = clock;
    srclken = VCC;
    cntclken = VCC;
    deseren = VCC;
    proc fsm = S13;
  END IF;

WHEN S13 =>
  IF (op_code[] == NSHF) THEN
    tmsout = VCC;
    cntclken = VCC;
    !tck = clock;
    srclken = VCC;
    proc fsm = S0;
  ELSIF (op_code[] == NSHFCP) THEN
    !tck = clock;
    srclken = VCC;
    cntclken = VCC;
    deseren = VCC;
    proc fsm = S14;
  END IF;

WHEN S14 =>
  IF (op_code[] == NSHFCP) THEN
    !tck = clock;
    srclken = VCC;
    cntclken = VCC;
    deseren = VCC;
    proc fsm = S15;
  END IF;

WHEN S15 =>
  IF (op_code[] == NSHFCP) THEN
    !tck = clock;
    srclken = VCC;
    cntclken = VCC;
    deseren = VCC;
    proc fsm = S7;
  END IF;

```

```

WHEN S16 =>
    IF (op_code[] == NSHFCP) THEN
        !tck = clock;
        srclken = VCC;
        cntclken = VCC;
        deseren = VCC;
        IF (cnt16eq1) THEN
            proc fsm = S17;
        ELSE
            proc fsm = S16;
        END IF;
    END IF;

WHEN S17 =>
    IF (op_code[] == NSHFCP) THEN
        !tck = clock;
        srclken = VCC;
        cntclken = VCC;
        deseren = VCC;
        tmsout = VCC;
        proc fsm = S0;
    END IF;

END CASE;

tsffclken_n = EXP((op_code[4..1] == SELTAP) & S1);
syncaclken_n = EXP((op_code[4..1] == SSA) & S1);
syncbclken_n = EXP((op_code[4..1] == SSB) & S1);
pclt2clken_n = EXP((op_code[4..1] == JP) & S5);
ldpc_n = EXP((op_code[4..1] == JP) & S9);
tsffclken = !tsffclken_n;
syncaclken = !syncaclken_n;
syncbclken = !syncbclken_n;
pclt2clken = !pclt2clken_n;
ldpc = !ldpc_n;

sinal1 = EXP((op_code[4..1] == SHIFT) & S2);
sinal2 = EXP((op_code[] == NSHFCP) & S8);
sinal3 = EXP((op_code[] == NSHF) & S11);
acc1clken = EXP(sinal1 & sinal2 & sinal3);

sinal6 = EXP((op_code[] == NSHFCP) & S6);
sinal7 = EXP((op_code[] == NSHFCP) & S12);
acc3clken = EXP(sinal6 & sinal7);

sinal4 = EXP((op_code[] == NSHFCP) & S4);
sinal5 = EXP((op_code[] == NSHFCP) & S10);
sinal14 = EXP((op_code[4..1] == JP) & S7);
gpaccclken = EXP(sinal4 & sinal5 & sinal14);

sinal8 = EXP((op_code[] == NSHFCP) & S7);
sinal9 = EXP((op_code[] == NSHF) & S3);
ldsr = EXP(sinal8 & sinal9);

ldcnt24_n = EXP((op_code[] == LDC24) & S2);
ldcnt24 = !ldcnt24_n;

sinal10 = EXP((op_code[] == LDC16) & S2);
sinal11 = EXP((op_code[] == LDC24) & S4);
ldcnt16msb = EXP(sinal10 & sinal11);

```

```

sinal12      = EXP((op_code[] == LDC16) & S4);
sinal13      = EXP((op_code[] == LDC24) & S6);
ldcnt16lsb  = EXP(sinal12 & sinal13);

saidas[5]    = Q[5];
saidas[4]    = Q[4];
saidas[3]    = Q[3];
saidas[2]    = Q[2];
saidas[1]    = Q[1];
saidas[0]    = Q[0];

trstout      = !((op_code[] == TRST) & S0);

END;

%***** Program counter ****%
**          Program counter          **
%***** ****%


TITLE "program counter";

SUBDESIGN prgcnt
(
  reset       : INPUT;      % system reset %
  clock        : INPUT;      % system clock %
  ldpc         : INPUT;      % load program counter %
  pclt2clken   : INPUT;      % load latch of MSB %
  pcclken     : INPUT;      % increment program counter %
  Data[7..0]    : INPUT;      % data bus %
  ACC[7..0]    : INPUT;      % accumulator data bus %

  A[19..0]     : OUTPUT;     % address bus %

)

VARIABLE
pc_bits[19..0]  : DFF;      % FF outputs %
latch_bits[3..0] : DFF;      % latch %

BEGIN

pc_bits[].clr = reset;
pc_bits[].clk = !clock;
latch_bits[].clr = reset;
latch_bits[].clk = !clock;

IF (ldpc) THEN
  pc_bits[7..0] = Data[7..0];
  pc_bits[15..8] = ACC[7..0];
  pc_bits[19..16] = latch_bits[3..0];
ELSIF (pcclken) THEN
  pc_bits[] = pc_bits[] + 1;
ELSE
  pc_bits[] = pc_bits[];
END IF;

IF (pclt2clken) THEN
  latch_bits[3..0] = Data[3..0];

```

```

ELSE
    latch_bits[] = latch_bits[];
END IF;

A[19] = tri(pc_bits[19],reset);
A[18] = tri(pc_bits[18],reset);
A[17] = tri(pc_bits[17],reset);
A[16] = tri(pc_bits[16],reset);
A[15] = tri(pc_bits[15],reset);
A[14] = tri(pc_bits[14],reset);
A[13] = tri(pc_bits[13],reset);
A[12] = tri(pc_bits[12],reset);
A[11] = tri(pc_bits[11],reset);
A[10] = tri(pc_bits[10],reset);
A[9] = tri(pc_bits[9],reset);
A[8] = tri(pc_bits[8],reset);
A[7] = tri(pc_bits[7],reset);
A[6] = tri(pc_bits[6],reset);
A[5] = tri(pc_bits[5],reset);
A[4] = tri(pc_bits[4],reset);
A[3] = tri(pc_bits[3],reset);
A[2] = tri(pc_bits[2],reset);
A[1] = tri(pc_bits[1],reset);
A[0] = tri(pc_bits[0],reset);

END;

%*****%
** General Purpose Accumulator **
%*****%

TITLE "Accumulator";

SUBDESIGN accum
(
    reset      : INPUT;      % system reset %
    clock      : INPUT;      % system clock %
    Data[7..0]  : INPUT;      % Data bus %
    accken     : INPUT;      % load the accumulator with data bits %

    ACC[7..0]   : OUTPUT;     % accumulator data bus %

)

VARIABLE
acc_bits[7..0] : DFF;      % FF outputs %

BEGIN

    acc_bits[].clr = reset;
    acc_bits[].clk = !clock;

    IF ( accken ) THEN
        acc_bits[7..0] = Data[7..0];
    ELSE
        acc_bits[] = acc_bits[];

```

```

END IF;

ACC[7..0] = acc_bits[7..0];

END;

*****Comparing the TDI with the expected value*****
TITLE "Comparing TDI ";

SUBDESIGN scanin
(
    reset      : INPUT;      % system reset %
    clock      : INPUT;      % system clock %
    Data[7..0]  : INPUT;      % Data bus %
    ACC[7..0]   : INPUT;      % accumulator Data bus %
    acc3clken  : INPUT;      % load the accumulator with data bits %
    srclken    : INPUT;      % shift one bit to the right %
    ldsr       : INPUT;      % load the serializer with data bits %
    TDI        : INPUT;      % Test data input %

    rescomp    : OUTPUT;     % comparation result %

)

VARIABLE
acc3_bits[7..0]  : DFF;      % FF outputs %
sr2_bits[7..0]   : DFF;      % FF outputs %
sr3_bits[7..0]   : DFF;      % FF outputs %

BEGIN

sr2_bits[].clr = reset;
sr2_bits[].clk = !clock;
acc3_bits[].clr = reset;
acc3_bits[].clk = !clock;
sr3_bits[].clr = reset;
sr3_bits[].clk = !clock;

IF (ldsr) THEN
    sr2_bits[7..0] = ACC[7..0];
ELSIF ( srclken ) THEN
    sr2_bits[6..0] = sr2_bits[7..1];
    sr2_bits[7] = sr2_bits[7];
ELSE
    sr2_bits[] = sr2_bits[];
END IF;

IF (ldsr) THEN
    sr3_bits[7..0] = acc3_bits[7..0];
ELSIF ( srclken ) THEN
    sr3_bits[6..0] = sr3_bits[7..1];
    sr3_bits[7] = sr3_bits[7];
ELSE
    sr3_bits[] = sr3_bits[];
END IF;

```

```

IF (acc3clken) THEN
    acc3_bits[7..0] = Data[7..0];
ELSE
    acc3_bits[] = acc3_bits[];
END IF;

IF ((TDI != sr2_bits[0]) & sr3_bits[0]) THEN
    rescomp = VCC;
ELSE
    rescomp = GND;
END IF;

END;

%*****.Serializer to generate the TDO output ****%
TITLE "generating TDO";

SUBDESIGN scanout
(
    reset      : INPUT;      % system reset %
    clock      : INPUT;      % system clock %
    Data[7..0]  : INPUT;      % Data bus %
    acc1clken  : INPUT;      % load the accumulator with data bits
    srlclken   : INPUT;      % shift one bit to the right %
    srclken    : INPUT;      % shift one bit to the right %
    ldsr       : INPUT;      % load the serializer with data bits %

    TDO        : OUTPUT;     % Test data output %

)

VARIABLE
acc1_bits[7..0]  : DFF;      % FF outputs %
srl_bits[7..0]   : DFF;      % FF outputs %

BEGIN

acc1_bits[].clr = reset;
acc1_bits[].clk = !clock;
srl_bits[].clr = reset;
srl_bits[].clk = clock;

IF (ldsr) THEN
    srl_bits[7..0] = acc1_bits[7..0];
ELSIF (srlclken # srclken) THEN
    srl_bits[6..0] = srl_bits[7..1];
    srl_bits[7] = srl_bits[7];
ELSE
    srl_bits[] = srl_bits[];
END IF;

IF (acc1clken) THEN
    acc1_bits[7..0] = Data[7..0];
ELSE
    acc1_bits[] = acc1_bits[];

```

```

END IF;

TDO = srl1_bits[0];

END;

***** Counter ****
*****%TITLE "counter";
*****%
SUBDESIGN count
(
    reset      : INPUT;      % system reset %
    clock      : INPUT;      % system clock %
    ldcnt16msb : INPUT;      % load counter 16 most significant byte %
    ldcnt24    : INPUT;      % load counter 24 byte %
    ldcnt16lsb : INPUT;      % load counter least significant byte %
    cntclken   : INPUT;      % decrement counter %
    Data[7..0]  : INPUT;      % data bus %

    cnt24eq0   : OUTPUT;     % true when the counter 24 reaches zero %
    cnt16eq0   : OUTPUT;     % true when the counter 16 reaches zero %
    cnt16eq1   : OUTPUT;     % true when the counter 16 reaches one %
    cnt16leq8  : OUTPUT;     % true when the counter 16 is equal/greater
                            % than 8 %

)
VARIABLE
cnt_bits[23..0]  : DFF;      % FF outputs %
leq8_N           : NODE;     % node to feed the macrocell %

BEGIN
    cnt_bits[].clr_n = reset;
    cnt_bits[].clk = !clock;

    IF (ldcnt16msb) THEN
        cnt_bits[15..8] = Data[7..0];
        cnt_bits[23..16] = cnt_bits [23..16];
        cnt_bits[7..0] = cnt_bits [7..0];
    ELSIF (ldcnt16lsb) THEN
        cnt_bits[7..0] = Data[7..0];
        cnt_bits[23..8] = cnt_bits[23..8];
    ELSIF (ldcnt24) THEN
        cnt_bits[23..16] = Data[7..0];
        cnt_bits[15..0] = cnt_bits[15..0];
    ELSIF (cntclken) THEN
        cnt_bits[] = cnt_bits[] - 1;
    ELSE
        cnt_bits[] = cnt_bits[];
    END IF;

    IF (cnt_bits[15..0] <= 8) THEN
        leq8_N = VCC;
    ELSE

```

```

leq8_N = GND;
END IF;

cnt16leq8 = MCELL(leq8_N);

IF (cnt16leq8 & cnt_bits[3..0] == 0 ) THEN
  cnt16eq0 = VCC;
ELSE
  cnt16eq0 = GND;
END IF;

IF (cnt16leq8 & cnt_bits[3..0] == 1 ) THEN
  cnt16eq1 = VCC;
ELSE
  cnt16eq1 = GND;
END IF;

IF (cnt_bits[23..16] == 0 ) THEN
  cnt24eq0 = VCC;
ELSE
  cnt24eq0 = GND;
END IF;

END;

%***** Status block *****
TITLE "Test processor status";

SUBDESIGN status
(
  reset      : INPUT;      % system reset %
  clock      : INPUT;      % system clock %
  rescomp    : INPUT;      % result of the comparation %
  errffclken : INPUT;      % error status ff clock enable %
  eotffclken : INPUT;      % end of test ff clock enable %

  ERROR      : OUTPUT;     % error flag %
  EOT        : OUTPUT;     % end of test flag %

)
VARIABLE
errff      : DFF;          % FF output %
eotff      : DFF;          % FF output %

BEGIN
errff.clrn = reset;
errff.clk = !clock;
eotff.clrn = reset;
eotff.clk = !clock;

IF (eotffclken) THEN
  eotff = VCC;

```

```

ELSE
  eotff = eotff;
END IF;

IF (errffclken & !errff) THEN
  errff = rescomp;
ELSE
  errff = errff;
END IF;

ERROR = errff;
EOT = eotff;

END;

%***** Syncronism Output block *****
TITLE "generating syncout";

SUBDESIGN syncout
(
  reset      : INPUT;      % system reset %
  clock       : INPUT;      % system clock %
  op_code0    : INPUT;      % instruction register bit 0 %
  syncaclken  : INPUT;      % load the new syncronism output A value %
  syncbclken  : INPUT;      % load the new syncronism output B value %

  syncouta    : OUTPUT;     % syncronism output A %
  syncoutb    : OUTPUT;     % syncronism output B %

)
VARIABLE
  ffA          : DFF;        % FF output %
  ffB          : DFF;        % FF output %

BEGIN
  ffA.clrn = reset;
  ffA.clk = !clock;
  ffB.clrn = reset;
  ffB.clk = !clock;

  IF (syncaclken) THEN
    ffA = op_code0;
  ELSE
    ffA = ffA;
  END IF;

  IF (syncbclken) THEN
    ffB = op_code0;
  ELSE
    ffB = ffB;
  END IF;

```

```
syncouta = ffA;  
syncoutb = ffB;  
  
END;
```

