

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/37655827>

# Metodologias de teste para FPGAs (Field Programmable Gate Arrays) integradas em sistemas reconfiguráveis

Article · January 2003

Source: OAI

---

CITATION

1

READS

38

1 author:



Manuel Gericota

Polytechnic Institute of Porto

75 PUBLICATIONS 362 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Weebly [View project](#)



Concurrent Test Methods for Reconfigurable Hardware Systems (based on partial and dynamically reconfigurable FPGAs) [View project](#)

Manuel Gradim de Oliveira Gericota

**Metodologias de teste para FPGAs  
(*Field Programmable Gate Arrays*)  
integradas em sistemas reconfiguráveis**

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

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

Abril de 2003



Tese realizada sob supervisão do  
Prof. Doutor José Manuel Martins Ferreira  
Professor Associado com Agregação do  
Departamento de Engenharia Electrotécnica e de Computadores da  
Faculdade de Engenharia da Universidade do Porto



À Fátima,  
*companheira dos meus dias.*



*É possível falhar de muitas maneiras...  
mas triunfar só é possível de uma  
(razão também pela qual uma é fácil e outra difícil –  
falhar o objectivo é fácil, alcançá-lo, difícil)*

Aristóteles (*circa* 384 A.C.)



# RESUMO

**Palavras-chave:** dispositivos lógicos programáveis, FPGAs (*Field Programmable Gate Arrays*) com capacidade de reconfiguração parcial dinâmica, replicação activa, teste estrutural concorrente, normas IEEE 1149.1 e IEEE 1532.

Os dispositivos lógicos programáveis, nomeadamente as FPGAs (*Field Programmable Gate Arrays*), conheceram uma expansão considerável nos últimos anos, devido ao aparecimento de componentes com maior capacidade e complexidade, e à sua inerente flexibilidade. O advento de um novo tipo de FPGAs, baseadas em memória estática, com capacidade de reconfiguração parcial dinâmica, reforçou as suas vantagens, incrementando o seu uso como base para os sistemas de computação reconfigurável.

A redução para escalas submicrométricas do processo de fabrico deste novo tipo de dispositivos, com o consequente aumento da densidade e complexidade dos circuitos, agravou a probabilidade de ocorrência de defeitos, obrigando à busca de novos métodos de detecção, diagnóstico e tolerância a faltas, capazes de assegurar a sua fiabilidade a longo prazo.

Esta tese apresenta uma nova metodologia, não-intrusiva, para o teste estrutural concorrente de FPGAs com capacidade de reconfiguração parcial dinâmica, alicerçada na replicação activa e na libertação para o teste dos seus recursos (blocos lógicos configuráveis e interligações). O objectivo é a detecção de faltas permanentes, que podem eventualmente emergir durante o funcionamento da FPGA, e de faltas transitórias, como, por exemplo, as que são provocadas por radiação, que de outra forma alterariam permanentemente a funcionalidade das funções afectadas.

A abordagem subjacente ao método proposto assume que apenas uma porção relativamente pequena do componente está sob teste, sem afectar a sua operação. Se a funcionalidade de um pequeno conjunto de recursos da FPGA puder ser relocada no mesmo componente, de forma completamente transparente (isto é, sem perturbar a sua operação), então esses recursos ficam livres e podem ser testados. Se, no final do teste, nenhuma falta tiver sido detectada, esses recursos ficam novamente disponíveis para serem reutilizados. Através de uma estratégia de relocação e teste sequencial de todos os recursos, a totalidade da FPGA pode ser sistematicamente testada em busca de faltas emergentes.

A metodologia proposta implica um reduzido dispêndio de recursos ao nível do componente, uma vez que os procedimentos de reconfiguração e teste são efectuados através da infra-estrutura de teste definida na norma IEEE 1149.1 (*IEEE Standard Test Access Port and Boundary-Scan Architecture*).



## ABSTRACT

**Keywords:** *programmable logic devices, partial and dynamically reconfigurable FPGAs (Field Programmable Gate Arrays), active replication, concurrent structural test, IEEE 1149.1 and IEEE 1532 standards.*

Reconfigurable logic devices, namely Field Programmable Gate Arrays (FPGAs), experienced a considerable expansion in the last few years, due to an increase in their size and complexity. The introduction of a new type of SRAM-based FPGAs, capable of implementing fast run-time partial reconfiguration, reinforced the advantages of these devices, wide-spreading their usage as a base for reconfigurable computing systems.

However, larger dies and smaller submicron scales increase the probability of lifetime operation failures, requiring new test/fault tolerance strategies, capable of assuring long-term reliability.

This thesis presents a novel non-intrusive methodology addressing the concurrent structural test of partial and dynamically reconfigurable FPGAs, based on the active replication and release for test of their internal resources (configurable logic blocks and interconnections). The main objective consists of detecting permanent faults, which may emerge during FPGA operation, and transient faults, such as single event upsets in space environments, which would otherwise become permanent faults.

The approach underlying the proposed method assumes that only a relatively small portion of the chip is being tested off-line, while the remaining part continues its normal on-line operation. If the functionality of a small number of FPGA resources can be relocated on another portion of the same device, in a way that is completely transparent to the operation of the system (i.e. without disturbing the device functionality), then those resources can be taken off-line and tested. If no faults are detected, these resources are again made available to be reused; otherwise, they are removed from operation. This fault scanning procedure moves on to relocate and test another set of resources, sweeping through the whole FPGA, systematically testing for faults.

The proposed methodology presents a very low overhead at chip level, since all the reconfiguration and test actions are carried out through the IEEE 1149.1 (IEEE Standard Test Access Port and Boundary-Scan Architecture) infrastructure.



## AGRADECIMENTOS

Muitas pessoas e instituições contribuíram directa ou indirectamente para a efectivação deste trabalho, e, mesmo correndo o risco de esquecer alguém, não posso deixar de lhes expressar aqui os meus agradecimentos.

Abrindo com as instituições, em primeiro lugar, os agradecimentos à minha escola de origem, o Instituto Superior de Engenharia do Porto, e, em particular, à pessoa do Eng. Sousa Guimarães que sempre apoiou as minhas pretensões; ao Ministério da Educação pela bolsa concedida ao abrigo do programa PRODEP III; à Fundação para a Ciência e Tecnologia que, através da aprovação do projecto POCTI 33842 dentro do qual este trabalho se desenvolveu, forneceu os meios financeiros para a sua realização; à Faculdade de Engenharia da Universidade do Porto pelas facilidades concedidas como instituição de acolhimento, e à Fundação Calouste Gulbenkian pelo apoio económico a algumas das missões.

Em relação aos agradecimentos pessoais, não poderia deixar de começar pelo meu orientador, o Professor Doutor José Manuel Martins Ferreira, que, ao longo de mais de dez anos de colaboração, sempre me demonstrou a sua confiança no meu trabalho e me apoiou na persecução dos objectivos que me propus atingir.

Ao Eng. Miguel Silva, por quem passou parte significativa do trabalho experimental que consolidou as bases teóricas expostas nesta dissertação, cabe uma menção especial. A minha deferência também para com o Professor Doutor Gustavo Alves, pelo seu incentivo e colaboração, e o Professor Doutor José Miguel Vieira dos Santos, pela sua disponibilidade.

Os meus agradecimentos igualmente ao Professor Charles E. Stroud da *University of North Carolina at Charlotte* por ter acedido a deslocar-se a Portugal, pelo seu estímulo, pelos avisos e pelas muitas dicas que nos deixou e que nos permitiram descobrir novos caminhos para este trabalho.

Uma referência também para o apoio de secretariado das Dras. Inês Cambeiro e Alexandrina Sousa e informático do António Cardoso.

Por último, com um carinho muito especial, aos meus pais, pela confiança e encorajamento, e à minha esposa, a quem devo muito tempo, que ingratamente lhe roubei para terminar este trabalho.

A todos o meu muito obrigado.



## NOTA AO LEITOR

Ao longo da escrita de um documento técnico sobre uma área em que, quase exclusivamente, toda a literatura existente se exprime em inglês, é por vezes difícil o emprego de termos traduzidos para o português, caso existam, sem que isso afecte a compreensão da exposição, por o seu uso não estar consagrado. Por isso, nos casos em que o emprego do termo em português não suscite dúvidas é omitido o vocábulo original. Quando o termo em português não é de uso corrente, opta-se por colocar o vocábulo original subsequentemente em estilo itálico e entre parêntesis. Sempre que se revele impossível a tradução para o português, mantém-se o vocábulo original em língua inglesa e em estilo itálico.

O estilo itálico emprega-se ainda para destacar uma determinada palavra ou expressão em português.

Os acrónimos, sejam em português, sejam em estrangeiro, são sempre apresentados em maiúsculas e em estilo normal. Por vezes, apesar de o emprego do termo em português estar consagrado, não se verifica o mesmo com o seu acrónimo, pelo que se mantêm posteriores referências usando o acrónimo original em língua inglesa. Uma lista de acrónimos é incluída para facilitar a sua identificação.



# ÍNDICE

1.	INTRODUÇÃO.....	1
1.1.	ÂMBITO DO TRABALHO .....	5
1.2.	ENQUADRAMENTO PROPORCIONADO PELO PROJECTO POCTI 33842 .....	7
1.3.	CONTRIBUIÇÕES INOVADORAS.....	8
1.4.	ESTRUTURA DA DISSERTAÇÃO .....	8
2.	O PROJECTO COM DISPOSITIVOS LÓGICOS PROGRAMÁVEIS DE ELEVADA COMPLEXIDADE.....	11
2.1.	Evolução dos dispositivos lógicos programáveis.....	15
2.2.	ARQUITECTURA DAS FPGAS .....	25
2.2.1	TECNOLOGIAS DE PROGRAMAÇÃO .....	27
2.2.2	ARQUITECTURA DOS BLOCOS LÓGICOS.....	32
2.2.3	ARQUITECTURA DOS BLOCOS DE ENTRADA/SAÍDA.....	38
2.2.4	ARQUITECTURA DOS RECURSOS DE ENCaminhamento .....	40
2.3.	FERRAMENTAS DE APOIO AO PROJECTO .....	44
2.3.1	SIMULAÇÃO LÓGICA .....	45
2.3.2	SÍNTESE E MAPEAMENTO .....	46
2.3.3	ANÁLISE TEMPORAL.....	49
2.4.	FPGAS COM RECONFIGURAÇÃO PARCIAL DINÂMICA .....	52
2.4.1	A ARQUITECTURA DA FAMÍLIA VIRTEX .....	54
2.4.2	VALOR ACRESCENTADO PELA RECONFIGURAÇÃO PARCIAL DINÂMICA.....	63
2.5.	SUMÁRIO.....	68
3.	O TESTE DE FPGAS E OS DESAFIOS INTRODUZIDOS PELA RECONFIGURAÇÃO PARCIAL DINÂMICA.....	69
3.1.	ESPECTRO DE DEFEITOS E MODELAÇÃO DE FALTAS.....	73
3.1.1	MODELOS DE FALTAS .....	76
3.1.2	DEFEITOS PARAMÉTRICOS .....	79
3.1.3	DEFEITOS CARACTERÍSTICOS DAS FPGAS.....	83
3.1.4	O MODELO SEMPRE-A.....	84
3.2.	TESTE ESTRUTURAL .....	86

3.2.1 TESTE DE ESTRUTURAS REGULARES .....	88
3.2.2 MODELO DE FALTAS HÍBRIDO .....	90
3.3. ESTADO DA ARTE .....	92
3.4. SUMÁRIO .....	115
<b>4. UMA METODOLOGIA PARA O TESTE CONCORRENTE VIA RECONFIGURAÇÃO PARCIAL DINÂMICA .....</b>	<b>117</b>
4.1. SOLUÇÃO GLOBAL .....	121
4.2. DEFINIÇÃO DA METODOLOGIA .....	128
4.2.1 RELOCAÇÃO DOS BLOCOS FUNCIONAIS .....	128
4.2.2 ESTRATÉGIA DE ROTAÇÃO .....	133
4.2.3 APLICAÇÃO DO TESTE .....	136
4.3. SUMÁRIO .....	140
<b>5. REPLICAÇÃO DE CIRCUITOS ACTIVOS .....</b>	<b>143</b>
5.1. CIRCUITOS COMBINATÓRIOS .....	147
5.2. CIRCUITOS SEQUENCIAIS .....	151
5.2.1 CIRCUITOS SÍNCRONOS COM SINAL DE RELÓGIO LIVRE .....	152
5.2.2 CIRCUITOS SÍNCRONOS COM SINAL DE RELÓGIO BLOQUEÁVEL .....	154
5.2.3 CIRCUITOS ASSÍNCRONOS .....	163
5.3. INTERLIGAÇÕES .....	164
5.4. RECUPERAÇÃO DE ERROS E RESTRIÇÕES .....	168
5.4.1 DIRECCIONALIDADE DOS RECURSOS DE ENCAMINHAMENTO .....	169
5.4.2 IMPEDÂNCIA DAS INTERLIGAÇÕES .....	170
5.4.3 CORRECÇÃO DE ERROS DURANTE A AQUISIÇÃO DE ESTADO .....	180
5.4.4 CORRECÇÃO DE FALTAS TRANSITÓRIAS .....	187
5.5. SUMÁRIO .....	189
<b>6. ROTAÇÃO DOS RECURSOS .....</b>	<b>191</b>
6.1. ESTRATÉGIAS PARA A ROTAÇÃO DE RECURSOS .....	195
6.1.1 FACTORES DE CUSTO .....	196
6.1.2 ROTAÇÃO ALEATÓRIA .....	197
6.1.3 ROTAÇÃO HORIZONTAL .....	198
6.1.4 ROTAÇÃO VERTICAL .....	199
6.2. APLICAÇÃO AO CONJUNTO DE CIRCUITOS-PADRÃO .....	201
6.2.1 IMPACTO SOBRE O TEMPO DE PROPAGAÇÃO .....	203

6.2.2 VECTORES DE PROXIMIDADE COMO MÉTRICA DE CUSTO .....	209
6.2.3 INFLUÊNCIA DO NÚMERO DE DERIVAÇÕES E DA DISPOSIÇÃO SOBRE O CUSTO.....	220
6.2.4 DETERMINAÇÃO DA ESTRATÉGIA DE MAIS BAIXO CUSTO.....	224
6.3.ROTAÇÃO DAS INTERLIGAÇÕES .....	226
6.4.SUMÁRIO.....	229
<b>7. APLICAÇÃO DO TESTE .....</b>	<b>231</b>
7.1.DESCRIÇÃO DA ESTRATÉGIA DE TESTE.....	235
7.1.1 APLICAÇÃO DE ESTÍMULOS.....	236
7.1.2 CAPTURA DAS RESPOSTAS .....	241
7.2.TESTE DOS BLOCOS LÓGICOS CONFIGURÁVEIS.....	241
7.2.1 ESTRUTURA DOS BLOCOS LÓGICOS CONFIGURÁVEIS.....	243
7.2.2 TESTE DOS BLOCOS ELEMENTARES .....	244
7.2.3 TESTE DE UM BLOCO LÓGICO.....	248
7.2.3.1 DETERMINAÇÃO DO NÚMERO DE CONFIGURAÇÕES DE TESTE .....	248
7.2.3.2 TESTE DAS LINHAS DE TRANSPORTE E DA LÓGICA ASSOCIADA .....	255
7.2.3.3 GERAÇÃO DOS CONJUNTOS DE VECTORES DE TESTE.....	257
7.2.4 RESOLUÇÃO DO TESTE E TOLERÂNCIA A FALTAS .....	262
7.2.5 TESTE DAS TABELAS DE CONSULTA EM MODO DE MEMÓRIA.....	263
7.2.6 LATÊNCIA DO TESTE .....	263
7.3.TESTE DAS INTERLIGAÇÕES .....	264
7.4.TESTE DA MEMÓRIA DE CONFIGURAÇÃO .....	270
7.5.SUMÁRIO.....	271
<b>8. IMPLEMENTAÇÃO E VALIDAÇÃO .....</b>	<b>273</b>
8.1.IMPLEMENTAÇÃO .....	277
8.2.VALIDAÇÃO .....	287
8.3.RESULTADOS TEMPORAIS .....	292
8.4.SUMÁRIO.....	297
<b>9. CONCLUSÕES E PERSPECTIVAS DE TRABALHO FUTURO .....</b>	<b>299</b>
9.1.CONCLUSÕES.....	301
9.2.PERSPECTIVAS DE TRABALHO FUTURO .....	302
<b>10. REFERÊNCIAS BIBLIOGRÁFICAS.....</b>	<b>305</b>
<b>11. ANEXOS .....</b>	<b>329</b>



# ÍNDICE DE FIGURAS

Figura 1.1: Representação esquemática de uma FPGA.....	6
Figura 2.1: Principais marcos na evolução dos dispositivos lógicos programáveis .....	15
Figura 2.2: Arquitectura interna de uma PROM .....	17
Figura 2.3: Arquitectura interna de uma PLA .....	18
Figura 2.4: Arquitectura interna de uma PAL .....	19
Figura 2.5: Arquitectura interna de uma PAL 22V10.....	20
Figura 2.6: Arquitectura interna de uma MAX 9000.....	22
Figura 2.7: Arquitectura interna de uma FPGA [Rose et al., 93].....	23
Figura 2.8: Árvore de decisão FPGA <i>versus</i> CPLD .....	25
Figura 2.9: Classificação das FPGAs em função das possibilidades de configuração.....	26
Figura 2.10: Blocos de interligação a) ponto-a-ponto e b) usando um multiplexador .....	27
Figura 2.11: Implementação de uma tabela de consulta .....	28
Figura 2.12: Célula de configuração SRAM da Xilinx com cinco transístores a) implementação e b) equivalente lógico.....	28
Figura 2.13: Constituição de um anti-fusível de silício amorfo .....	30
Figura 2.14: Constituição de um anti-fusível de dieléctrico a) secção e b) esquema simplificado....	31
Figura 2.15: Constituição interna de uma EEPROM .....	32
Figura 2.16: Arquitectura de um bloco de granulosidade fina baseado em multiplexadores, o ACT1 da Actel .....	34
Figura 2.17: Exemplo da implementação de uma função Booleana no ACT1 .....	34
Figura 2.18: Arquitectura de um bloco de granulosidade fina baseado em multiplexadores e contendo um <i>flip-flop</i> , o AT6000 da Atmel.....	35
Figura 2.19: Arquitectura de um bloco de granulosidade grossa baseado em tabelas de consulta (CLB da família XC3000 da Xilinx) .....	36
Figura 2.20: Representação esquemática e possibilidades de configuração dos blocos de SRAM da família Virtex-E da Xilinx .....	37
Figura 2.21: Arquitectura genérica dos blocos de E/S.....	39
Figura 2.22: Arquitectura de encaminhamento assimétrica .....	41
Figura 2.23: Arquitectura de encaminhamento simétrica.....	42
Figura 2.24: Fluxograma das ferramentas de apoio ao projecto com FPGAs .....	44
Figura 2.25: Fluxo do processo de síntese .....	47

Figura 2.26: Circuitos equivalentes para os recursos de encaminhamento da FPGA.....	50
Figura 2.27: Exemplo de um gráfico temporal para um pequeno circuito (adapt. de [Betz et al., 99]) .....	51
Figura 2.28: Gráfico temporal completo com indicação do caminho crítico .....	52
Figura 2.29: Representação esquemática da arquitectura das Virtex.....	55
Figura 2.30: Arquitectura de um CLB .....	56
Figura 2.31: Esquema detalhado de uma <i>slice</i> .....	57
Figura 2.32: Constituição interna de um bloco de E/S .....	58
Figura 2.33: Arquitectura de registos da infra-estrutura de teste .....	59
Figura 2.34: Diagrama de estados do controlador do TAP .....	60
Figura 2.35: Recursos de encaminhamento associados a cada bloco lógico.....	61
Figura 2.36: Organização da memória de configuração.....	62
Figura 2.37: Visão tridimensional do espaço de configuração.....	65
Figura 2.38: Escalonamento temporal das funções no espaço de configuração .....	66
Figura 3.1: Modelação de uma falta a nível físico e a nível lógico numa porta ‘Não-E’ .....	75
Figura 3.2: Evolução previsível das características de fabrico dos semicondutores .....	80
Figura 3.3: Variação percentual das falhas devidas a defeitos ténues com a redução de escala em relação à totalidade das falhas [Needham et al., 98].....	80
Figura 3.4: Visualização de um fenómeno de electromigração.....	82
Figura 3.5: Visualização de uma arquitectura de interligação em cobre .....	82
Figura 3.6: Detecção de um defeito por teste múltiplo de uma falta sempre-a.....	85
Figura 3.7: Arquitectura genérica para implementação de auto-teste interno numa FPGA [Stroud, 02] .....	92
Figura 3.8: Arquitectura de uma estratégia de teste baseada em matrizes unidimensionais [Huang et al., 96] .....	92
Figura 3.9: PLA com incorporação de um esquema de testabilidade [Fujiwara et al., 81].....	95
Figura 3.10: Arquitectura genérica de auto-teste para uma PLA [Agrawal et al., 93a] .....	96
Figura 3.11: Formas de teste [Abramovici et al., 90] .....	96
Figura 3.12: Matriz de blocos sob teste .....	98
Figura 3.13: Teste de uma FPGA com compactação de respostas por intermédio de portas ‘E’ e ‘OU’ .....	99
Figura 3.14: Teste do modo memória das tabelas de consulta .....	99
Figura 3.15: Estrutura de auto-teste baseada em matrizes C-testáveis.....	103
Figura 3.16: Estrutura de auto-teste dos recursos de interligação e encaminhamento.....	104
Figura 3.17: Sequência de teste dos nós.....	104

Figura 3.18: Estratégia de auto-teste em anel.....	105
Figura 3.19: Metodologia de implementação de auto-teste <i>on-line</i> .....	110
Figura 3.20: Unidade autónoma de teste (BISTER) .....	112
Figura 3.21: Posição e deslocamento das áreas de auto-teste dentro da FPGA.....	112
Figura 3.22: Configurações de teste das unidades autónomas .....	113
Figura 4.1: Metodologia de teste proposta.....	129
Figura 4.2: As duas fases do processo de replicação .....	131
Figura 4.3: Estratégias de rotação .....	134
Figura 4.4: Varrimento adoptado para os recursos de interligação .....	135
Figura 4.5: Modelo de teste .....	137
Figura 4.6: Recolha de estímulos do teste da linha de transporte.....	137
Figura 4.7: Procedimento de teste de um bloco lógico .....	139
Figura 5.1: Alocação dos vectores aos recursos configuráveis.....	148
Figura 5.2: Colocação em paralelo das entradas dos dois CLBs envolvidos na replicação.....	148
Figura 5.3: Fluxograma do processo de replicação de um bloco de lógica combinatória .....	150
Figura 5.4: Fluxograma do processo de replicação de um bloco de lógica síncrona com relógio livre.....	153
Figura 5.5: Replicação de uma célula contendo um registo simples com sinal de habilitação de relógio e com sinal de inicialização síncrono ou assíncrono .....	156
Figura 5.6: Fluxograma do processo de replicação de uma célula contendo um registo simples com sinal de habilitação de relógio e com sinal de inicialização síncrono ou assíncrono.....	158
Figura 5.7: Implementação numa <i>slice</i> do bloco auxiliar de replicação.....	159
Figura 5.8: Replicação de uma célula contendo um registo simples com sinal de habilitação de relógio e com sinal de inicialização síncrono ou assíncrono e um bloco combinatório independente .....	160
Figura 5.9: Replicação de uma célula contendo lógica combinatória registada com sinal de habilitação de relógio e com sinal de inicialização síncrono ou assíncrono.....	161
Figura 5.10: Fluxograma do processo de replicação de uma célula contendo lógica combinatória registada com sinal de habilitação de relógio e com sinal de inicialização síncrono ou assíncrono.....	162
Figura 5.11: Simulação de uma operação de transferência e actualização durante o processo de replicação .....	163
Figura 5.12: Contador binário assíncrono ( <i>ripple counter</i> ) .....	164

Figura 5.13: Replicação de uma célula contendo uma <i>latch</i> simples com sinal de habilitação de entrada e com sinal de inicialização assíncrono .....	165
Figura 5.14: Replicação de uma célula contendo lógica combinatória e uma <i>latch</i> com sinal de habilitação de entrada e com sinal de inicialização assíncrono.....	165
Figura 5.15: Replicação de uma interligação .....	167
Figura 5.16: Recursos de interligação nas matrizes de encaminhamento global.....	167
Figura 5.17: Atraso de propagação durante a replicação de recursos de interligação.....	168
Figura 5.18: Recursos de encaminhamento associados a um CLB .....	169
Figura 5.19: Colocação em paralelo das ligações aos CLBs durante a replicação .....	170
Figura 5.20: Topologia da interligação entre as saídas de dois CLBs .....	171
Figura 5.21: Circuito eléctrico equivalente que decorre da interligação entre as saídas de dois CLBs .....	171
Figura 5.22: Esquema eléctrico de um multiplexador de 8 entradas controlado pela memória de configuração .....	172
Figura 5.23: Representação esquemática das ligações e dos pontos de medida antes da interligação das saídas .....	173
Figura 5.24: Diagrama de tensões nos pontos referenciados na Figura 5.23 .....	173
Figura 5.25: Representação esquemática das ligações e dos pontos de medida .....	174
Figura 5.26: Paralelo unidireccional dos CLBs envolvidos na replicação .....	175
Figura 5.27: Diagrama de tensões nos pontos referenciados na Figura 5.25 .....	175
Figura 5.28: Representação esquemática das ligações e dos pontos de medida .....	176
Figura 5.29: Diagrama de tensões nos pontos referenciados na Figura 5.28 .....	177
Figura 5.30: Representação esquemática das ligações e dos pontos de medida .....	177
Figura 5.31: Diagrama de tensões nos pontos referenciados na Figura 5.30 .....	178
Figura 5.32: Tensão de transição entre níveis lógicos.....	178
Figura 5.33: Variação da forma de onda da tensão ao longo da interligação.....	179
Figura 5.34: Representação esquemática das ligações e dos pontos de medida .....	179
Figura 5.35: Diagrama de tensões nos pontos referenciados na Figura 5.34 .....	180
Figura 5.36: Paralelo das entradas.....	180
Figura 5.37: Comportamento do paralelo das entradas em caso de defeito na entrada do CLB replicado .....	181
Figura 5.38: Comportamento do paralelo das saídas em caso de defeito na saída do CLB replicado .....	182
Figura 5.39: Derivação imediata do sinal após a chegada à matriz de encaminhamento global ....	182

Figura 5.40: Derivação do sinal de saída na matriz de encaminhamento global após várias interligações internas .....	183
Figura 5.41: Pontos de derivação possíveis em interligações unidireccionais ou mistas .....	183
Figura 5.42: Representação esquemática das ligações e dos pontos de medida.....	184
Figura 5.43: Diagrama de tensões nos pontos referenciados na Figura 5.42.....	184
Figura 5.44: Representação esquemática das ligações e dos pontos de medida.....	185
Figura 5.45: Diagrama de tensões nos pontos referenciados na Figura 5.44.....	185
Figura 5.46: Representação esquemática das ligações e dos pontos de medida.....	186
Figura 5.47: Diagrama de tensões nos pontos referenciados na Figura 5.46.....	186
Figura 6.1: Estratégia de rotação horizontal .....	198
Figura 6.2: Estratégia de rotação vertical.....	199
Figura 6.3: Dispersão dos componentes de uma mesma função ao longo de sucessivas relocações sempre no mesmo sentido .....	200
Figura 6.4: Modelo comportamental do contador binário de 24 bits.....	206
Figura 6.5: Implementação física do contador e sua representação simplificada.....	206
Figura 6.6: Parte da implementação física do contador e sua representação simplificada com um dos CLBs libertado para ser testado empregando uma estratégia de rotação vertical.....	208
Figura 6.7: Parte do ficheiro de simulação temporal do contador de 24 bits após uma das rotações .....	208
Figura 6.8: Variação da frequência máxima de funcionamento do contador .....	209
Figura 6.9: Implementação física do circuito B01.....	210
Figura 6.10: Variação da frequência com a aplicação das duas estratégias de rotação ao circuito B01.....	211
Figura 6.11: Implementação física, com restrições à distribuição, do circuito B01 .....	211
Figura 6.12: Disposição predominantemente horizontal com linha com elevado número de derivações.....	213
Figura 6.13: Disposição predominantemente vertical com linha com elevado número de derivações.....	215
Figura 6.14: Aplicação da estratégia de rotação vertical a uma disposição predominantemente horizontal com linha com elevado número de derivações .....	216
Figura 6.15: Aplicação da estratégia de rotação horizontal a uma disposição predominantemente horizontal com linha com elevado número de derivações .....	216
Figura 6.16: Variação da frequência máxima com a aplicação das duas estratégias de rotação à implementação sem e com restrições à distribuição do circuito B01.....	218

Figura 6.17: Implementação física do circuito B06 .....	219
Figura 6.18: Implementação física, com restrições à distribuição, do circuito B06.....	219
Figura 6.19: Variação da frequência máxima com a aplicação das duas estratégias de rotação à implementação sem e com restrições à distribuição do circuito B06.....	220
Figura 6.20: Circuito com 25 blocos lógicos, onde cada um apresenta uma linha com derivação para todos os restantes .....	221
Figura 6.21: Valor do factor de proximidade para uma linha saindo de um CLB com derivação para todos os outros.....	222
Figura 6.22: Diferença percentual entre os factores de proximidade para cada CLB, resultante da aplicação da estratégia vertical relativamente à horizontal (valores negativos indicam vantagem da aplicação da estratégia vertical) .....	222
Figura 6.23: Circuito com 10 blocos lógicos, onde cada um apresenta uma linha com derivação para todos os restantes, com uma disposição predominantemente horizontal .....	223
Figura 6.24: Comparação entre os valores do factor de proximidade para uma linha saindo de um CLB com derivação para todos os outros, numa disposição predominantemente horizontal.....	223
Figura 6.25: Circuito com 10 blocos lógicos, onde cada um apresenta uma linha com derivação para todos os restantes, com uma disposição predominantemente vertical.....	223
Figura 6.26: Comparação entre os valores do factor de proximidade para uma linha saindo de um CLB com derivação para todos os outros, numa disposição predominantemente vertical.....	224
Figura 6.27: Tamanho médio dos ficheiros de configuração por CLB para os circuitos B01 a B14, aplicando ambas as estratégias de rotação .....	225
Figura 6.28: Comparação entre o tamanho médio dos ficheiros de reconfiguração por CLB e o tamanho do circuito .....	226
Figura 6.29: Modelo simplificado de uma célula da matriz de interligação global.....	226
Figura 6.30: Rotação dos canais de interligação sob teste .....	227
Figura 6.31: Exemplo da rotação dos canais em teste.....	228
Figura 7.1: Infra-estrutura para o teste de um bloco lógico .....	236
Figura 7.2: Representação simbólica da macro BSCAN_VIRTEX.....	237
Figura 7.3: Representação esquemática da célula-base do Registo do Utilizador .....	238
Figura 7.4: Descrição em VHDL do Registo do Utilizador.....	239
Figura 7.5: Implementação física do Registo do Utilizador .....	240
Figura 7.6: Modelo de teste de uma slice do CLB .....	243
Figura 7.7: Estrutura interna de uma tabela de consulta.....	244

Figura 7.8: Sequência de teste para um multiplexador.....	245
Figura 7.9: Sequência de teste para um multiplexador programável .....	246
Figura 7.10: Optimização das configurações de teste para uma tabela de consulta.....	247
Figura 7.11: Estrutura dos elementos de retenção presentes na <i>slice</i> .....	248
Figura 7.12: Exemplo de uma possível interligação de primitivas num bloco lógico .....	248
Figura 7.13: Exemplo da segmentação do circuito por cones de influência.....	251
Figura 7.14: Configurações de teste dentro do cone de influência .....	252
Figura 7.15: Cone de influência da saída YB .....	253
Figura 7.16: Recolha de estímulos do teste da linha de transporte.....	256
Figura 7.17: Condicionantes ao teste da linha de transporte .....	256
Figura 7.18: Exemplo da compactação de vectores de teste .....	259
Figura 7.19: Exemplo do teste das matrizes locais de encaminhamento.....	261
Figura 7.20: Sequência de teste das matrizes locais.....	262
Figura 7.21: Pontos possíveis de aplicação e captura de teste numa linha sob teste .....	265
Figura 7.22: Diagnóstico de um PIP com uma falta do tipo sempre-aberto.....	266
Figura 7.23: Diagnóstico de um segmento aberto.....	267
Figura 7.24: Teste de PIPs contra faltas do tipo sempre-fechado.....	268
Figura 8.1: Janela de gestão de projecto no <i>Foundation</i> .....	278
Figura 8.2: Janela de interface com o <i>FPGA Editor</i> .....	279
Figura 8.3: Janela de interface com o <i>BoardScope</i> .....	280
Figura 8.4: Janela de interface com o <i>Partial Reconfiguration Creator</i> .....	282
Figura 8.5: Janela de interface com o <i>Virtex PART</i> .....	282
Figura 8.6: Placa de experimentação .....	283
Figura 8.7: Formas de onda recolhidas durante uma das fases experimentais .....	284
Figura 8.8: Resultado da aplicação de ambas as estratégias de rotação à distribuição inicial do circuito B08 .....	286
Figura 8.9: Janela de interface com a ferramenta de aplicação e recolha dos vectores de teste....	287
Figura 8.10: Linhas activas num bloco de E/S configurado como entrada e usado para capturar uma das saídas do CLB sob teste .....	288
Figura 8.11: Ficheiro de restrições para implementação do Registo do Utilizador .....	289
Figura 8.12: Colocação do Registo do Utilizador dentro do espaço físico da FPGA .....	289
Figura 8.13: Verificação do correcto funcionamento do contador durante a relocação .....	290
Figura 8.14: Exemplo de relocação de um dos CLBs do contador .....	291
Figura 8.15: CLB sob teste .....	291
Figura 8.16: Slice com configuração de teste.....	292



# ÍNDICE DE TABELAS

Tabela 2.1: Nomenclatura das PAL [Sandige, 90].....	21
Tabela 6.1: Principais características dos circuitos usados no teste da estratégia de rotação .....	202
Tabela 6.2: Variação da frequência e relação entre o tamanho global dos ficheiros de reconfiguração parcial com a aplicação das duas estratégias de rotação .....	204
Tabela 6.3: Tamanho médio dos ficheiros de reconfiguração parcial por CLB ocupado com a aplicação das duas estratégias de rotação .....	205
Tabela 6.4: Valores da frequência máxima encontrada para o circuito B01 com restrições.....	212
Tabela 6.5: Quadro resumo da aplicação dos factores de proximidade ao circuito B01 .....	217
Tabela 6.6: Quadro resumo da aplicação dos factores de proximidade ao circuito B06 .....	220
Tabela 6.7: Valores máximos da frequência de funcionamento de B06 em ambos os casos após a aplicação das estratégias de rotação.....	220
Tabela 7.1: Resumo do espaço ocupado pela implementação do Registo do Utilizador numa XCV200.....	238
Tabela 7.2: Tabela de cobertura para determinação do número mínimo de configurações de teste .....	254
Tabela 7.3: Conjunto de vectores de teste .....	260
Tabela 7.4: Conjunto de vectores para o teste das interligações .....	268
Tabela 7.5: Número mínimo de configurações de teste .....	269
Tabela 8.1: Tamanho dos vectores e tempo de reconfiguração para cada etapa, na replicação de circuitos sequenciais síncronos com sinal de habilitação de relógio .....	293
Tabela 8.2: Tamanho dos vectores e tempo de reconfiguração para cada etapa, na replicação de circuitos sequenciais síncronos com sinal de relógio livre e de circuitos puramente combinatórios .....	294
Tabela 8.3: Tamanho dos vectores e tempo de reconfiguração das configurações de teste.....	295
Tabela 8.4: Tempo de deslocamento dos vectores de teste a aplicar.....	295
Tabela 8.5: Tempo de deslocamento das respostas aos vectores aplicados ao CLB sob teste .....	295
Tabela 8.6: Valores médios para o tempo total de teste de um CLB .....	296
Tabela 8.7: Valores para o tempo médio de teste da matriz lógica de uma FPGA.....	296



## ACRÓNIMOS

ABEL	<i>Advanced Boolean Expression Language</i>
API	<i>Application Programming Interface</i>
ASIC	<i>Application Specific Integrated Circuit</i>
ATE	<i>Automatic Test Equipment</i>
bps	bits por segundo
BS	<i>Boundary Scan</i>
BST	<i>Boundary Scan Test</i>
CCI	Carta de Circuito Impresso
CI	Circuito Integrado
CLB	<i>Configurable Logic Block</i>
CMOS	<i>Complementary Metal-Oxide-Semiconductor</i>
COTS	<i>Components Off-The-Shelf</i>
CPLD	<i>Complex Programmable Logic Device</i>
CPU	<i>Central Processing Unit</i>
CRC	<i>Cyclic Redundancy Check</i>
CUPL	<i>Universal Compiler for Programmable Language</i>
EEPROM	<i>Electrically Erasable Programmable Read Only Memory</i>
EPROM	<i>Erasable Programmable Read Only Memory</i>
ERA	<i>Electrically Reconfigurable Array</i>
E/S	Entrada/Saída
FPGA	<i>Field Programmable Gate Array</i>
HDL	<i>Hardware Description Language</i>
IEEE	<i>Institute of Electrical and Electronics Engineers</i>
IFA	<i>Inductive Fault Analysis</i>
ISP	<i>In-System Programming</i>
ITC	<i>International Test Conference</i>
LCA	<i>Logic Cell Array</i>

LFSR	<i>Linear-Feedback Shift Register</i>
LUT	<i>Look-Up Table</i>
MOS	<i>Metal-Oxide-Semiconductor</i>
MPGA	<i>Mask Programmable Gate Array</i>
MTTF	<i>Mean Time To Failures</i>
ORA	<i>Output Response Analyser</i>
ORCA	<i>Optimized Reconfigurable Cell Array</i>
OTP	<i>One-Time Programmable</i>
PAL	<i>Programmable Array Logic</i>
PALASM	<i>PAL Assembler</i>
PCI	<i>Peripheral Component Interconnect</i>
PIP	<i>Programmable Interconnect Point</i>
PLA	<i>Programmable Logic Array</i>
pmos	<i>positive metal-oxide-semiconductor</i>
PLD	<i>Programmable Logic Device</i>
PROM	<i>Programmable Read Only Memory</i>
RAM	<i>Random Access Memory</i>
RISC	<i>Reduced Instruction Set Computer</i>
ROM	<i>Read Only Memory</i>
RTL	<i>Register Transfer Level</i>
SET	<i>Single Event Transient</i>
SEU	<i>Single Event Upset</i>
SoC	<i>System-on-a-Chip</i>
SPLD	<i>Simple Programmable Logic Device</i>
SRAM	<i>Static Random Access Memory</i>
STAR	<i>Self-Testing ARea</i>
TAP	<i>Test Access Port</i>
TCK	<i>Test Clock</i>
TDI	<i>Test Data Input</i>
TDO	<i>Test Data Output</i>

TMS	<i>Test Mode Select</i>
TMR	<i>Triple Modular Redundancy</i>
TPG	<i>Test Pattern Generator</i>
TTL	<i>Transistor-Transistor Logic</i>
UV-EPROM	<i>Ultra-Violet Erasable Programmable Read Only Memory</i>
VHDL	<i>VHSIC Hardware Description Language</i>
VHSIC	<i>Very High Speed Integrated Circuit</i>
VLSI	<i>Very Large Scale Integration</i>
XHWIF	<i>Xilinx HardWare InterFace</i>



# 1. INTRODUÇÃO



O uso de dispositivos lógicos programáveis, nomeadamente de FPGAs (*Field Programmable Gate Arrays*), experimentou uma expansão considerável nos últimos anos, devido, em parte, a um aumento da sua densidade e complexidade. Este género de componentes é actualmente utilizado nos mais diversos produtos electrónicos, desde simples aplicações de electrónica de consumo, telefones celulares ou consolas de jogos, até sistemas mais avançados de telecomunicações e sistemas de satélite para fins militares e civis. Estes autênticos “mini-computadores” configuráveis pelos utilizadores apresentam menores requisitos em termos de espaço, proporcionam menores prazos de introdução no mercado e oferecem maior flexibilidade relativamente aos componentes tradicionais com funcionalidade pré-definida, tornando-se cada vez mais competitivos em termos de preço. O advento de FPGAs parcialmente reconfiguráveis, baseadas em tecnologia de memória estática (*SRAM(Static Random Access Memory)-based*), abriu, por sua vez, um novo mundo de possibilidades, ao permitir o desenvolvimento de circuitos cuja funcionalidade pode ser alterada total ou parcialmente, adaptando-se a novos requisitos aplicacionais de forma dinâmica e praticamente instantânea. Esta nova potencialidade generalizou o seu uso como base para plataformas de computação dinamicamente reconfiguráveis, onde múltiplas funções partilham o espaço de configuração, sendo implementadas quando necessário e substituídas por outras, quando dispensáveis.

As novas áreas de aplicação viabilizadas por estes desenvolvimentos criaram a necessidade de abordagens inovadoras no domínio do teste, capazes de lidar com a arquitectura interna das novas gerações de dispositivos e com a sua utilização em aplicações reconfiguráveis dinamicamente. Tornava-se, pois, premente o estudo de novos procedimentos e metodologias de teste, baseados em modelos de faltas que garantissem níveis aceitáveis de cobertura de defeitos, quando a redefinição de funcionalidade ocorresse com o sistema (a que o circuito pertence) em funcionamento e por reutilização constante dos mesmos recursos reprogramáveis, para implementar as funções lógicas necessárias a cada nova aplicação. A mesma característica que deu origem a este novo desafio (a reconfiguração dinâmica e parcial) constituiu também o ponto de partida para o desenvolvimento de procedimentos eficientes de teste da estrutura interna, optimizados para a especificidade arquitectónica desses dispositivos, que, sem perturbar a operação do circuito, garantem uma elevada fiabilidade do sistema.



## 1.1. ÂMBITO DO TRABALHO

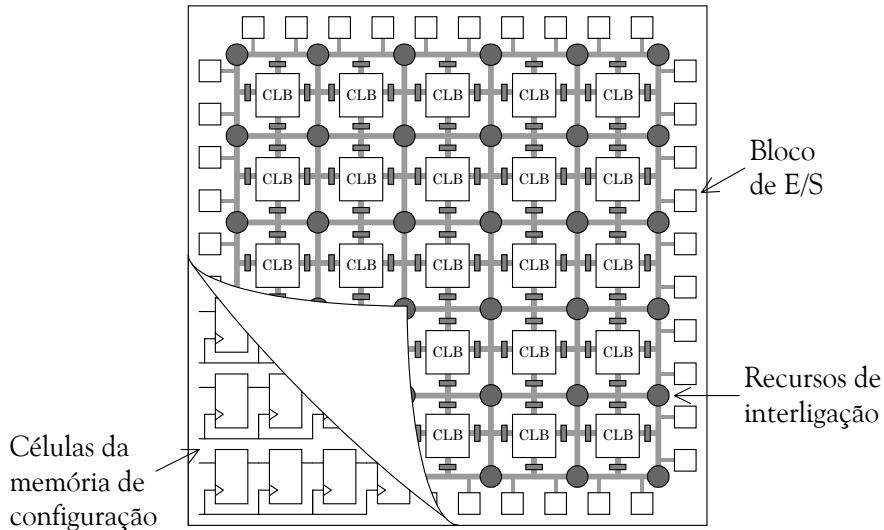
A exemplo do que se passa no resto da indústria de semicondutores, os avanços tecnológicos que tornaram as novas FPGAs mais apelativas e baratas também as tornaram menos fiáveis. A evolução do processo de fabrico para escalas submicrométricas alargou o espectro de defeitos e aumentou a possibilidade da sua ocorrência. O aumento da densidade de corrente nas interligações de metal agravou a possibilidade de ocorrência de fenómenos de electromigração, enquanto a diminuição das tensões de alimentação e, por arrastamento, das tensões de limiar, tornou os componentes mais susceptíveis à radiação cósmica, mesmo ao nível do mar. Este último factor afecta sobretudo os componentes que, como as FPGAs, se baseiam em grandes quantidades de memória, e conduz à ocorrência de faltas transitórias em funcionamento.

A redução para escalas submicrométricas permite também que alguns dos defeitos estruturais, relacionados com imperfeições durante o processo de fabrico, não sejam suficientemente importantes para se manifestarem durante os testes efectuados antes de o componente ser enviado para o cliente final, mas apenas depois de um período de operação mais ou menos longo, dependente das condições de funcionamento, em que emergem como faltas permanentes.

Estes aspectos tornaram evidente a necessidade de conceção de uma metodologia de teste que fosse simultaneamente capaz de lidar com os problemas emergentes e de não limitar as novas potencialidades. A nova metodologia pretendida deveria ser capaz de detectar a ocorrência de faltas permanentes, derivadas de defeitos estruturais emergentes, e de faltas transitórias, ocorridas durante a operação do componente. Este objectivo deveria, no entanto, ser atingido sem se perturbar o funcionamento do sistema, sempre que fosse possível o recurso à capacidade de reconfiguração parcial dinâmica. Daqui nasceu a motivação para o trabalho descrito nesta dissertação, que tem por objectivo principal a conceção de uma metodologia vocacionada para o teste estrutural concorrente de dispositivos lógicos programáveis, do tipo FPGA, com capacidade de reconfiguração parcial dinâmica, integrados em sistemas reconfiguráveis.

A nova metodologia de teste proposta pretende cobrir faltas permanentes e transitórias. No caso de faltas permanentes, após a detecção e diagnóstico da falta, a FPGA pode ser reconfigurada por forma a excluir o uso do recurso afectado, utilizando, para a sua substituição, recursos livres. No caso de faltas transitórias, a reconfiguração parcial dinâmica permite a recuperação de erros nas células da memória de configuração do componente, que, a manterem-se, modificariam a funcionalidade da lógica implementada.

Conceptualmente, uma FPGA pode ser vista como uma matriz de blocos lógicos configuráveis (CLB – *Configurable Logic Block*), independentes, rodeada na sua periferia por blocos de E/S, interligáveis por meio de um conjunto de recursos de encaminhamento, cuja configuração é controlada por um conjunto de células de memória que se encontram por baixo, conforme ilustrado na Figura 1.1.



**Figura 1.1:** Representação esquemática de uma FPGA

Na maioria dos casos, apenas uma porção desses recursos é efectivamente ocupada pela implementação de uma determinada especificação funcional, restando sempre alguns recursos livres. Mesmo no caso em que, visando um melhor aproveitamento dos recursos e uma redução da área da carta de circuito impresso, são implantadas várias aplicações na mesma FPGA, é altamente improvável que todos os recursos lógicos e de encaminhamento sejam ocupados. É partindo deste pressuposto que se desenvolve uma metodologia de teste onde os blocos lógicos não ocupados vão sendo testados sem se comprometer a operação da FPGA. Por sua vez, o teste dos blocos ocupados faz-se com recurso à replicação, nos blocos lógicos já testados, da sua configuração, relocando a sua lógica e libertando-os para o teste após todas as suas interligações terem sido restabelecidas. Através de uma técnica de varrimento, por rotação do bloco lógico sob teste, e de replicações e relocações consecutivas, é possível testar toda a matriz. Os blocos lógicos já testados e não usados ficam disponíveis como blocos sobresselentes, prontos a substituir outros onde seja detectada uma falta. O mesmo princípio é aplicado na replicação, relocação e teste dos recursos de encaminhamento.

Este método apenas é possível com este tipo de dispositivos, que suportam reconfiguração parcial dinâmica. Neles, a replicação de um bloco lógico, incluindo a sua informação funcional,

eventualmente o seu estado actual e o restabelecimento das suas interligações, é possível sem interromper a operação do componente.

A implementação desses novos procedimentos de teste não deve, no entanto, consumir demasiados recursos. Daí que seja preconizada a utilização de uma infra-estrutura normalizada, definida na norma IEEE 1149.1 [IEEE 1149.1, 01], já amplamente em uso na indústria. Através dessa infra-estrutura, pretende-se controlar não só a aplicação dos testes e a recolha dos resultados, mas também a necessária reconfiguração parcial dinâmica da própria FPGA.

A solução proposta permite, sem perturbar o funcionamento do sistema, efectuar o teste estrutural da FPGA, de forma dinâmica e totalmente transparente para o utilizador, possibilitando a detecção e a identificação de faltas permanentes e transitórias, e conduzindo a uma operação tolerante a faltas.

## 1.2. ENQUADRAMENTO PROPORCIONADO PELO PROJECTO POCTI 33842

A realização do presente trabalho enquadra-se num projecto designado *Teste Concorrente para Sistemas Electrónicos Reconfiguráveis (baseados em FPGAs com capacidade de reconfiguração parcial dinâmica)*, financiado pela Fundação para a Ciência e Tecnologia, sob o contrato POCTI/33842/ESE/2000, sendo esta tese um dos indicadores de realização previstos inicialmente. Este projecto tem como finalidade o desenvolvimento de uma metodologia para o teste de FPGAs dinâmica e parcialmente reconfiguráveis, podendo ser decomposto em três etapas principais:

- a análise do espectro de defeitos passível de caracterizar as FPGAs baseadas em memória estática, usadas em aplicações reconfiguráveis, e a avaliação da sua cobertura pelos modelos de faltas tradicionais, ao nível dos blocos lógicos, interligações e elementos da memória estática de configuração;
- o desenvolvimento de um procedimento de teste que, explorando a inerente regularidade das FPGAs e os mecanismos de acesso proporcionados pela infra-estrutura *Boundary Scan* (IEEE 1149.1), implemente uma detecção concorrente de faltas estruturais e permita a manutenção de uma lista de recursos defeituosos;
- o desenvolvimento de algoritmos que suportem a reconfiguração parcial dinâmica da FPGA, escalonando o uso dos recursos internos e libertando áreas para serem testadas sem interrupção da operação do sistema.

O enquadramento proporcionado pelo projecto permitiu, por um lado, suportar a participação em cursos, seminários e conferências, e alcançar uma maior divulgação dos resultados obtidos, sendo,

por outro lado, um sustentáculo indispensável para a realização de toda a parte experimental e para o desenvolvimento de uma ferramenta destinada a automatizar a implementação da metodologia proposta.

### 1.3. CONTRIBUIÇÕES INOVADORAS

De entre as contribuições inovadoras deste trabalho destaca-se o facto de ser a primeira proposta para uma metodologia de teste estrutural concorrente totalmente não intrusiva para FPGAs, baseada num novo mecanismo para a replicação e relocação dinâmica de funções em uso pelo sistema sem perturbação da sua funcionalidade. Esta nova metodologia permite a detecção e o diagnóstico de qualquer falta estrutural permanente ao nível dos recursos internos da FPGA, representada no modelo de faltas considerado, durante a operação do sistema, e a correcção da funcionalidade da função implementada nesses recursos. Possibilita igualmente a correcção de faltas transitórias que afectem a memória de configuração, alterando a funcionalidade da função afectada. Para a implementação e o controlo da totalidade da metodologia recorre-se à reutilização de uma infra-estrutura de teste previamente existente, à qual é acrescentada um registo de dados que ocupa sete dos blocos lógicos configuráveis da FPGA, que podem ser libertados no final do teste, não implicando, por isso, uma ocupação permanente de recursos.

A determinação do número mínimo de configurações de teste para o bloco lógico baseia-se no princípio da segmentação do circuito em cones de influência, numa nova aplicação da técnica apresentada em [McCluskey, 84], complementada pela utilização do método tabular apresentado em [Renovell et al., 99] e [Renovell et al., 99a].

Alguns destes factores inovadores extravasam o âmbito desta tese, tendo aplicação em outras áreas relacionadas com a utilização de plataformas de computação reconfiguráveis, baseadas em FPGAs com capacidade de reconfiguração parcial dinâmica [Gericota et al., 02] e [Gericota et al., 03a], como se dá conta na secção 2.4.2 e se retoma na secção 9.2.

### 1.4. ESTRUTURA DA DISSERTAÇÃO

O segundo capítulo inicia-se com uma síntese histórica da evolução dos dispositivos lógicos programáveis e uma comparação entre as suas diferentes características e técnicas de implementação. Mais pormenoradamente, são descritas a arquitectura interna e as ferramentas de apoio ao projecto com FPGAs. Por último, são analisadas as características específicas de uma família particular destes componentes, que suporta reconfiguração parcial dinâmica e sobre a qual incide o desenvolvimento e a validação da metodologia proposta.

O terceiro capítulo começa com uma abordagem ao espectro de defeitos que caracteriza os novos processos de fabrico em escalas submicrométricas e os modelos de faltas mais adequados para os representar, com particular ênfase para os defeitos característicos das FPGAs e para a proposta de um modelo de faltas concordante. Este capítulo finaliza com uma síntese do estado da arte na área do teste das FPGAs, em que, para além da apresentação do trabalho de outros autores, se identificam, na perspectiva do teste de FPGAs com reconfiguração parcial dinâmica, as suas lacunas.

A partir dessa análise, dos defeitos característicos, dos modelos de faltas mais convenientes para a sua representação e do regime de funcionamento associado aos sistemas que empregam este tipo de componentes, estabelece-se, no quarto capítulo, o tipo de teste que melhor se lhes adequa e identificam-se os requisitos a que a metodologia de teste deve obedecer. Seguidamente, é proposta uma metodologia global de abordagem ao teste estrutural concorrente, que, para efeitos de descrição, é repartida em três partes: replicação, rotação e teste, constituindo-se cada uma delas, respectivamente, como objecto de cada um dos três capítulos seguintes.

No capítulo cinco, é efectuada a descrição pormenorizada dos problemas que se levantam à relocação dinâmica das funções, sem interrupção da sua operação e consoante o tipo de circuito a replicar, bem como à das interligações, e são propostas soluções para a sua resolução. Em complemento, apresenta-se uma descrição dos mecanismos de recuperação de erros e as restrições associadas ao mecanismo proposto.

O capítulo seis analisa os aspectos relacionados com a forma como o procedimento de teste deve varrer toda a FPGA, sem afectar o funcionamento das funções nela implementadas. São analisadas basicamente duas alternativas, estudadas sob o ponto de vista do custo sobre o funcionamento das funções afectadas e sobre o tempo de latência das faltas.

No capítulo sete, é proposta uma estratégia para o teste dos diferentes tipos de recursos presentes numa FPGA. Para os recursos lógicos e de interligação, é determinado o número de configurações de teste necessárias, analisada a forma como é efectuada a geração dos vectores e descrita a arquitectura global para a sua aplicação e para a recolha das respostas, através da infra-estrutura de teste IEEE 1149.1. Para a memória de configuração, é recomendada uma estratégia de leitura, comparação e rescrita, que permite a correcção de faltas transitórias.

No capítulo 8, apresenta-se um exemplo prático de aplicação e os resultados experimentais recolhidos, que permitem avaliar a eficácia da solução proposta.

A concluir, o capítulo 9 apresenta uma súmula da proposta desenvolvida e traça possíveis direcções para a continuação da investigação, tanto na área do teste, como noutras domínios em que as FPGAs consideradas são usadas.

No capítulo 10, são listadas as referências bibliográficas indicadas ao longo do trabalho.

O anexo, em formato electrónico, é constituído por um CD-ROM, fixado no verso da contracapa, que contém vários elementos considerados relevantes. Estes elementos incluem, para além deste documento, uma descrição aprofundada da arquitectura e do modo de programação do componente usado para efeitos de implementação e validação, os artigos aceites em várias conferências e seminários, os ficheiros relativos à implementação da proposta, os ficheiros com resultados das experimentações efectuadas, o esquema da fonte de tensão comutada projectada e construída pelo autor para alimentação da FPGA usada na validação deste trabalho, uma cópia do relatório intercalar do projecto mencionado na secção 1.2, uma cópia do *Curriculum Vitae* do autor e uma base de dados contendo as referências de toda a bibliografia reunida pelo autor durante o trabalho de Doutoramento.

## **2. O PROJECTO COM DISPOSITIVOS LÓGICOS PROGRAMÁVEIS DE ELEVADA COMPLEXIDADE**



Os dispositivos lógicos programáveis (PLDs – *Programmable Logic Devices*), termo genérico usado para aludir a qualquer tipo de circuitos integrados que podem ser configurados para uma aplicação particular pelo utilizador final [Sharma, 98], bem como as ferramentas de apoio ao desenvolvimento de aplicações neles baseados, conheceram, a exemplo de toda a fileira tecnológica em que se enquadram, considerável evolução nos últimos anos. Desde o surgimento das primeiras matrizes lógicas programáveis ‘à medida’, usadas no início dos anos 70 na concepção de sistemas lógicos digitais, até aos dispositivos lógicos com mais de oito milhões de portas lógicas equivalentes e com possibilidade de reconfiguração parcial dinâmica no circuito (ISP – *In-System Programming*)<sup>1</sup>, permitindo a integração de sistemas num único integrado (SoC – *System-on-a-Chip*), um longo caminho foi percorrido, quer em termos da tecnologia de implementação, quer das ferramentas que dessa evolução permitem tirar o maior partido. A título de enquadramento, dar-se-á conta desta evolução, que se estende ao longo de mais de três décadas, dos principais aspectos construtivos e das ferramentas de apoio ao projecto desenvolvidas.

A exemplo da evolução das espécies relatada por Charles Darwin no seu livro *The Origin of Species by Means of Natural Selection or the Preservation of Favored Races in Struggle for Life* [Darwin, 1859], em que, por selecção natural, várias espécies foram evoluindo em diferentes direcções, e em que a resposta às condições envolventes ditava a sua sobrevivência ou extinção<sup>2</sup>, também na evolução dos dispositivos lógicos programáveis existiu um desenvolvimento em várias direcções, como resposta às exigências do mercado. Surgiram, desta forma, diferentes arquitecturas, com tecnologias de programação diversas, propondo recursos lógicos e disponibilidades de encaminhamento distintas, cada qual procurando dar resposta a problemas específicos sem perda de generalidade, afinal um dos esteios do seu desenvolvimento. A par dessa evolução, surgiu igualmente um conjunto de ferramentas de apoio ao projecto com estes dispositivos, que eram compostas por aplicações, na maior parte dos casos provenientes de diferentes fontes, e que, constituindo um pacote coerente, permitiam a automatização, ainda que com total controlo do projectista, de todas as etapas do desenvolvimento, desde o esquema físico ou descrição conceptual da funcionalidade pretendida até à configuração do dispositivo, passando pela síntese – mapeamento lógico, distribuição e encaminhamento das interligações -, pela simulação e pela análise temporal do circuito implementado.

---

<sup>1</sup> Virtex-II XC2V8000 da Xilinx [Xilinx, 02a]

<sup>2</sup> “... can we doubt (remembering that many more individuals are born than can possibly survive) that individuals having any advantage, however slight, over others, would have the best chance of surviving and of procreating their kind? On the other hand, we may feel sure that any variation in the least degree injurious would be rigidly destroyed. This preservation of favourable variations and the rejection of injurious variations, I call Natural Selection.”, [Darwin, 1859 – cap. IV].

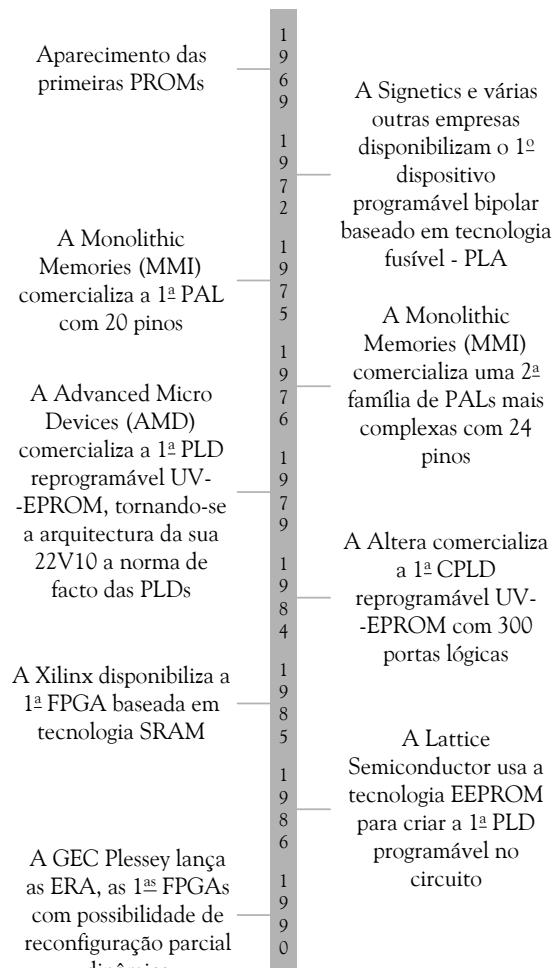
Um dos ramos da história evolutiva dos dispositivos lógicos programáveis desembocou no aparecimento de uma família de componentes com possibilidade de reconfiguração parcial dinâmica. Significa isto que deixou de ser necessário parar o sistema e configurar a totalidade do dispositivo quando se pretende apenas alterar ou mudar parte da sua funcionalidade, não se perturbando inclusive o normal funcionamento das funções que, embora ocupando o mesmo dispositivo, não são objecto de alteração ou mudança. O estudo da arquitectura de uma família de dispositivos lógicos programáveis dinâmica e parcialmente reconfiguráveis, sobre a qual incide a metodologia de teste desenvolvida nesta tese, a referência a um conjunto de aplicações que fazem uso específico desta particularidade e dos problemas por ela levantados, constituem a última parte deste capítulo.

## 2.1. EVOLUÇÃO DOS DISPOSITIVOS LÓGICOS PROGRAMÁVEIS

Anteriormente ao desenvolvimento da lógica programável, apenas se prefiguravam duas alternativas para a concepção e desenvolvimento de sistemas digitais: o uso de circuitos lógicos comerciais de catálogo (COTS – *Components Off-The-Shelf*), disponíveis no mercado a baixo custo, ou a concepção de um ou mais componentes dedicados (ASIC – *Application Specific Integrated Circuit*), realizados ‘à medida’, mas implicando um custo muito alto. Estes últimos conjugam uma elevada integração, da qual decorre uma redução do espaço ocupado nas cartas de circuito impresso (CCIs), com um elevado desempenho, mas apresentam a desvantagem do seu alto custo inicial, não só em termos monetários, mas também, e não menos importante, em termos do longo tempo de desenvolvimento que requerem. Por outro lado, a sua falta de flexibilidade elimina a hipótese de adaptação a posteriores mudanças. Se o volume de produção for elevado, os custos iniciais serão diluídos durante o tempo de vida útil, resultando num baixo valor por unidade. Contudo, nos últimos anos, a tendência em toda a fileira tecnológica tem evoluído no sentido da diminuição dos tempos de vida útil dos produtos (uma rápida rotação no mercado), tendo como consequência uma diminuição nos volumes de produção e um aumento da pressão para redução do denominado *time-to-market*. Os ASICs deixam, por isso, de ser, na grande maioria das aplicações, uma opção a considerar.

Aparecimento das primeiras PROMs 1969  
A Monolithic Memories (MMI) comercializa a 1<sup>a</sup> PAL com 20 pinos 1972  
A Advanced Micro Devices (AMD) comercializa a 1<sup>a</sup> PLD reprogramável UV-EPROM 1976

O desenvolvimento da lógica programável teve como objectivo proporcionar todas as vantagens da lógica ‘à medida’, sem as penalidades a ela associadas [Carter, 91]. A lógica programável alia um reduzido tempo de desenvolvimento à manutenção de uma elevada integração proporcionada pela lógica ‘à medida’, acrescentando ainda a possibilidade de permitir alterações em fases avançadas do desenvolvimento do produto, bem como posteriores



**Figura 2.1:** Principais marcos na evolução dos dispositivos lógicos programáveis

A Actel emprega a tecnologia anti-fusível em FPGAs OTP	1991	A QuickLogic apresenta as FPGAs anti-fusível de alta velocidade	1992
A Xilinx e a Altera comercializam as 1 <sup>as</sup> FPGAs com blocos de memória RAM dedicados	1993	A AT&T Microelectronics integra uma interface PCI nas suas FPGAs	1995
Processadores dedicados embutidos são integrados nas FPGAs	1996	A Xilinx lança a família XC6200	1996
A Xilinx e a Altera comercializam as 1 <sup>as</sup> FPGAs com um milhão de portas lógicas; a Actel disponibiliza um RISC CPU de 8 bits e memória SRAM	1998	Início da comercialização da série VIRTEX da Xilinx	1998
A Atmel disponibiliza uma combinação de FPGA, RISC CPU de 8 bits e SRAM	1999	A Atmel embute pela 1 <sup>a</sup> vez blocos de FPGA no projecto de ASICs	2000
20 milhões de portas lógicas disponíveis (prev.) permitindo a implementação de soluções do tipo SoC numa única FPGA	2002	A Xilinx lança as Plataformas FPGA com 4 processadores embutidos, lógica dedicada para processamento de sinal e interfaces de comunicação de alto débito (3,125 Gbps)	2005

**Figura 2.1:** Principais marcos na evolução dos dispositivos lógicos programáveis (cont.)

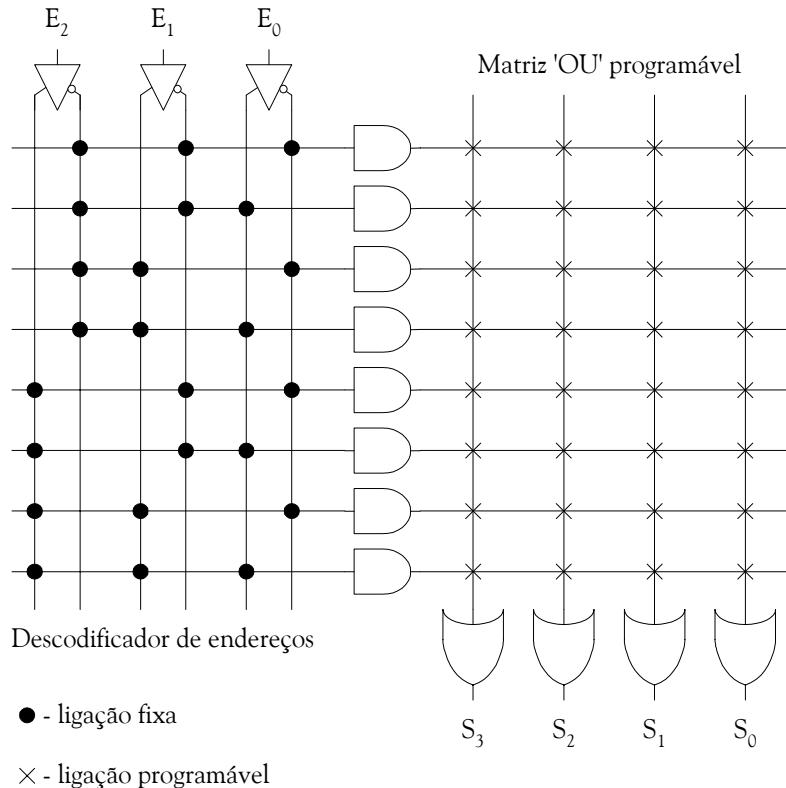
actualizações, sem a necessidade de se proceder ao redesenho da CCI. Estes benefícios pressupõem alguns custos, tais como um menor desempenho e dispositivos com custo unitário mais elevado, mesmo para grandes volumes de produção.

As memórias PROM (*Programmable Read Only Memory*), em tecnologia bipolar, só de leitura e programáveis uma única vez (daí a sua designação de dispositivos OTP – *One-Time Programmable*), foram, de facto, os primeiros dispositivos nos quais era permitido ao utilizador a implementação de uma função Booleana ‘à medida’ [Sachdev, 98]. O seu aparecimento deu-se por volta de 1969, marcando o surgimento da lógica programável, conforme ilustrado na Figura 2.1 (barra cronológica adaptada de [Bursky, 02]). As linhas de endereços eram usadas como linhas de entrada do circuito lógico, enquanto as linhas de dados funcionavam como saídas desse mesmo circuito. Na Figura 2.2 encontra-se representada esquematicamente a arquitectura interna de uma PROM. As PROMs são constituídas por uma matriz ‘E’ fixa e ‘OU’ programável. Os buffers/inversores e a matriz ‘E’ fixa

funcionam como circuito de descodificação. Para  $n$  entradas, o descodificador proporciona todos os  $2^n$  termos de produto possíveis à saída das portas ‘E’<sup>3</sup>. Uma vez que estas saídas são fornecidas às portas ‘OU’ através da matriz ‘OU’ programável, as PROMs podem ser usadas para implementar equações Booleanas por transposição directa das tabelas de verdade respectivas. Daí o terem sido

<sup>3</sup> Por razões de simplicidade, as  $n$  entradas de cada porta ‘E’ são representadas por uma única linha, onde se sobrepõe a indicação do respectivo termo de produto. De igual forma se representam as oito entradas de cada porta ‘OU’.

tipicamente usadas para implementação de conversores de código, geradores de caracteres e tabelas de armazenamento de dados, aplicações onde a funcionalidade é usualmente descrita através de tabelas de verdade, e não por equações Booleanas. A definição dessa funcionalidade na PROM é realizada pela fusão selectiva de fusíveis existentes em cada intersecção das linhas da matriz ‘OU’ programável (ligação programável) durante o processo de programação.

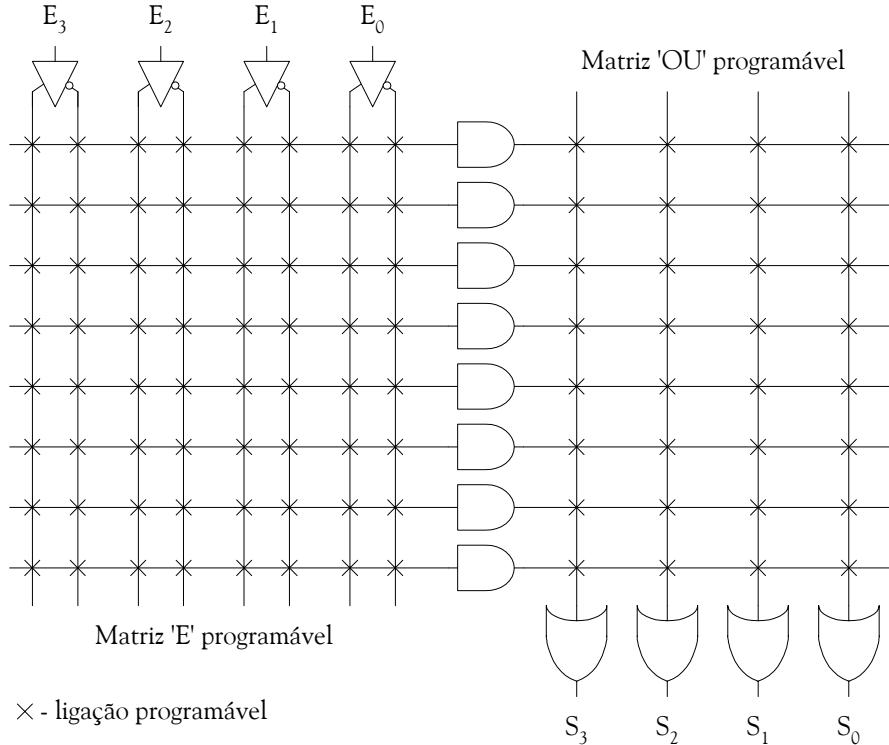


**Figura 2.2:** Arquitectura interna de uma PROM

Como as funções lógicas raramente requerem mais do que alguns termos de produto, a disponibilização de todos os  $2^n$  termos torna as PROM ineficientes na realização de circuitos lógicos [Brown et al., 96a]. Além disso, a simples adição de uma entrada provoca a duplicação das portas ‘E’ da matriz fixa e do número de fusíveis necessários na matriz ‘OU’ programável.

O primeiro dispositivo concebido especificamente para a implementação de circuitos lógicos foi a PLA (*Programmable Logic Array*), que teve a sua estreia no mercado em 1972. A sua estrutura, ilustrada na Figura 2.3, disponibiliza um número limitado de termos de produto definidos na matriz ‘E’, igualmente programável. A PLA possibilita a geração de uma qualquer função, desde que a sua implementação não exceda o número de termos de produto disponível. Uma vez que a matriz ‘E’ é programável, a adição de uma entrada efectua-se sem que ocorra a duplicação do tamanho de qualquer uma das matrizes. No entanto, estes dispositivos tiveram um sucesso limitado devido ao facto de serem lentos, visto que apresentavam duas matrizes programáveis em série, e difíceis de

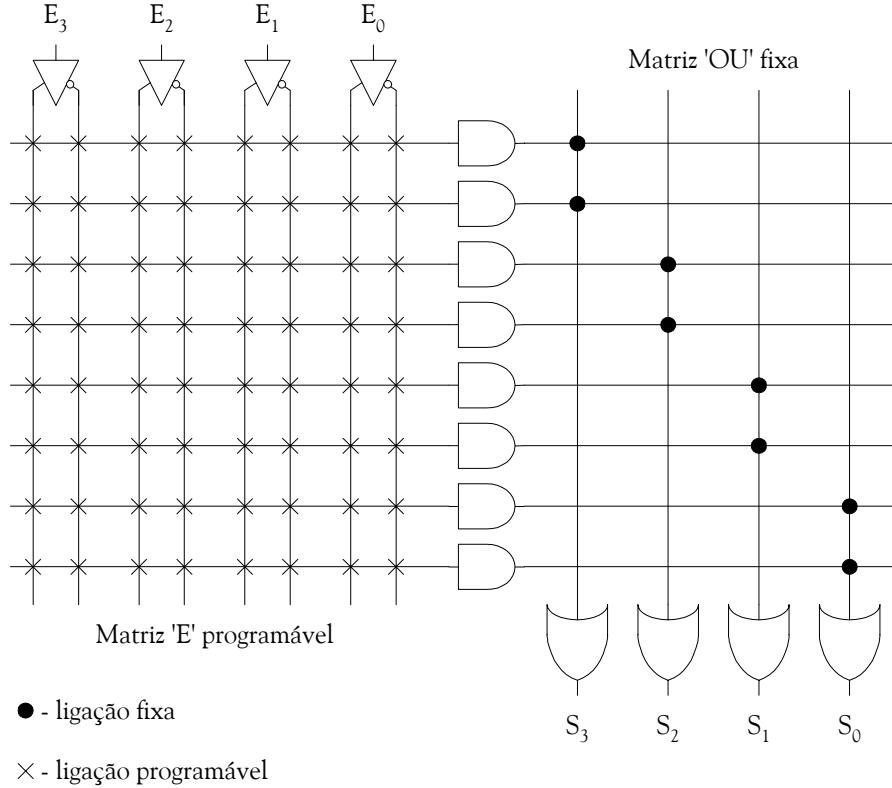
usar, pois a informação de programação era um mapa com indicação dos fusíveis a fundir, sem qualquer semelhança com os esquemáticos ou com as funções Booleanas tipicamente usadas no projecto de circuitos lógicos [Carter, 91].



**Figura 2.3:** Arquitectura interna de uma PLA

O grande salto tecnológico na lógica programável residiu na invenção da PAL (*Programmable Array Logic*), cuja arquitectura foi desenvolvida e patenteada por John Birkner da Monolithic Memories, Inc. (mais tarde absorvida pela Advanced Micro Devices – AMD), em 1975. A estrutura da PAL, ilustrada na Figura 2.4, consiste numa matriz de portas lógicas ‘E’ programável, seguida por uma matriz de portas ‘OU’ fixa, onde cada saída resulta da soma de um conjunto pré-determinado de termos de produto. A redução para uma única matriz programável contribuiu para a diminuição da complexidade do dispositivo e para um aumento da velocidade, sem comprometer a flexibilidade. Simultaneamente, foram criadas ferramentas de desenvolvimento baseadas em linguagens de descrição de hardware, HDLs (*Hardware Description Languages*), de que são exemplo a PAL Assembler (PALASM), da MMI, a Advanced Boolean Expression Language (ABEL), da Data I/O, e a Universal Compiler for Programmable Language (CUPL), da Logical Devices. Todas elas foram desenvolvidas na segunda metade da década de 70 e permitiram ao projectista a fácil conversão de equações lógicas Booleanas e de descrições de máquinas de estado em ficheiros de configuração, simplificando a fase de projecto.

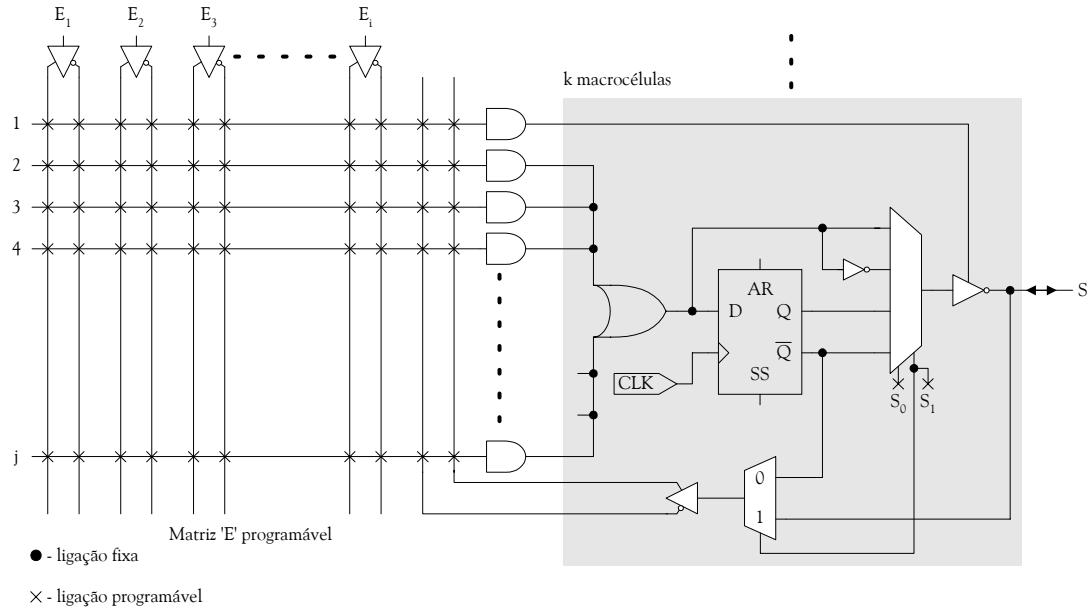
Para obstar à falta de generalidade devida ao facto de a matriz de portas lógicas ‘OU’ ser fixa, as PALs disponibilizavam uma grande variedade de recursos, quer em termos do número de portas lógicas ‘E’ e ‘OU’, quer no que respeita à composição da matriz de ligações programáveis. Para permitir a implementação de circuitos sequenciais, a generalidade das PALs possui *flip-flops* nas saídas das portas lógicas ‘OU’, os quais podem ou não ser intercalados entre essas portas e as saídas.



**Figura 2.4:** Arquitectura interna de uma PAL

As ligações programáveis usadas nos primeiros dispositivos, baseadas em tecnologia fusível, não eram reversíveis, não podendo efectuar-se, por conseguinte, a sua reprogramação. Essa condicionante foi ultrapassada em 1979, com o aparecimento do primeiro dispositivo lógico programável apagável por incidência de radiação ultra-violeta, UV-EPROM (*Ultra-Violet Erasable Programmable Read Only Memory*), lançado pela AMD, que popularizou o uso da lógica programável. A arquitectura da sua PAL 22V10, ilustrada na Figura 2.5, viria a tornar-se numa norma de facto para o desenvolvimento posterior destes dispositivos. A principal diferença entre a arquitectura das 22V10 e das PAL, suas antecessoras, reside no conjunto de recursos disponíveis em cada uma das saídas, que constituem um bloco designado pelo termo *macrocélula*. A macrocélula permite a programação da polaridade do sinal de saída, a colocação da saída em estado de alta impedância (terceiro estado) e a inclusão de um *flip-flop* na saída, a qual pode igualmente ser realimentada, reaparecendo como entrada da matriz ‘E’. As PAL deste tipo, cuja nomenclatura se

detalha na Tabela 2.1, estão disponíveis numa grande variedade, com diferente número de entradas e número e tipos de saídas (ex: 16L8, 20R6, 20RP8, ...).



**Figura 2.5:** Arquitectura interna de uma PAL 22V10

As PLAs e as PALs são caracterizadas por uma estrutura em bloco, única, em encapsulamentos de 20 a 44 pinos, densidades médias entre as 100 e as poucas centenas de portas lógicas, elevadas velocidades de propagação e baixo custo. Estes dispositivos são normalmente designados por SPLDs (*Simple Programmable Logic Devices*), sendo, pela sua estrutura básica, a melhor escolha para a implementação de equações lógicas Booleanas sobre a forma de soma de termos de produto. Recentemente, foram introduzidas outras possibilidades, tais como pinos bidireccionais programáveis e configuração flexível dos registos e dos sinais de relógio. Outras vantagens do seu uso incluem um consumo reduzido, tempo de desenvolvimento muito curto e elevada fiabilidade [Sharma, 98].

Por causa da sua arquitectura interna, o nível de complexidade das SPLDs não pode ser meramente estendido para permitir a implementação de funções lógicas mais complexas. O simples aumento do número de portas lógicas disponibilizado nos primeiros dispositivos esbarrou no facto de a estrutura da matriz lógica programável crescer muito rapidamente com o aumento do número de entradas. A única forma exequível de proporcionar dispositivos com maior capacidade, baseados na mesma arquitectura, é a interligação através de uma estrutura programável de múltiplos SPLDs. Nascem, desta forma, as primeiras CPLDs (*Complex Programmable Logic Devices*), com o lançamento, pela Altera, em 1984, da família *Classic*, a que se juntaram posteriormente as famílias MAX 5000, 7000 e 9000, cuja arquitectura interna está ilustrada na Figura 2.6 [Altera, 98]. A sua crescente procura levou outros fabricantes a desenvolverem este tipo de dispositivos, que se encontram actualmente

disponíveis num grande número de opções, acrescentando-lhes novas possibilidades e características, embora sem alterar a estrutura base. As CPLDs proporcionam uma capacidade equivalente a cerca de 50 SPLDs, mas a extensão dessas arquitecturas a densidades superiores mostra-se inefficiente. O desenvolvimento de PLDs de muito elevada capacidade requer uma abordagem diferente.

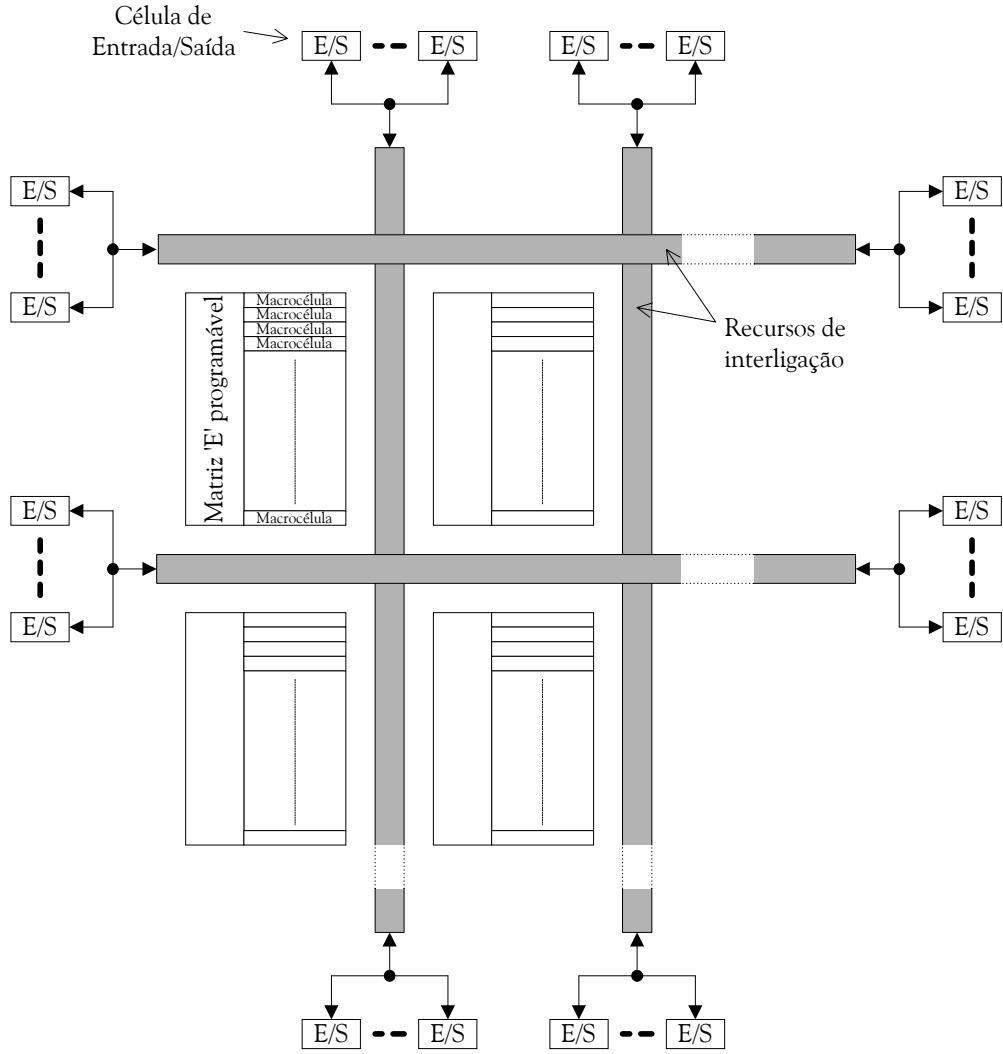
**Tabela 2.1:** Nomenclatura das PAL [Sandige, 90]

PAL $i^j t oo$
$i^j$ representa o número máximo de entradas na matriz 'E' (directas, a partir dos pinos, ou provenientes da realimentação das saídas)
$t$ representa o tipo de saída
combinatória: H – activa a nível lógico alto L – activa a nível lógico baixo P – polaridade programável C – invertida
registada: R – registada RP – registada com polaridade programável V – versátil, isto é, com macrocélulas de saída programáveis
$oo$ representa o número máximo de saídas dedicadas (combinatórias ou registadas) ou programáveis (combinatórias e registadas)

Em 1985, a Xilinx anunciou uma nova arquitectura, rompendo radicalmente com as arquitecturas baseadas nas PLAs. O objectivo que inspirou o desenvolvimento desta nova arquitectura foi o de alcançar a densidade lógica e a flexibilidade das matrizes de portas lógicas (MPGAs – *Mask Programmable Gate Arrays*)<sup>4</sup> retendo as melhores características da lógica programável tradicional. Pretendia-se, assim, um dispositivo lógico cuja funcionalidade fosse programável e facilmente modificável pelo utilizador, que proporcionasse uma redução nos tempos de desenvolvimento e que contornasse os riscos associados a soluções específicas do tipo ASIC. Esta nova arquitectura, denominada *Logic Cell Array* (LCA), constituiu o protótipo do que desde então se convencionou designar por *Field Programmable Gate Array* (FPGA). Conforme se ilustra na Figura 2.7, a nova

<sup>4</sup> Circuitos semi-específicos baseados em estruturas regulares de portas lógicas pré-definidas e pré-implementadas, interligadas por uma ou mais camadas de metal, cujas máscaras conduzem à definição da funcionalidade requerida pelo utilizador final.

arquitectura era composta por uma matriz de blocos lógicos<sup>5</sup> independentes, rodeada por um perímetro de blocos de Entrada/Saída (E/S) igualmente independentes e por um conjunto de recursos de interligação programáveis, que permitiam a interligação arbitrária dos blocos uns aos outros. Cada bloco lógico e de E/S podia ser programado individualmente para desempenhar uma função específica.

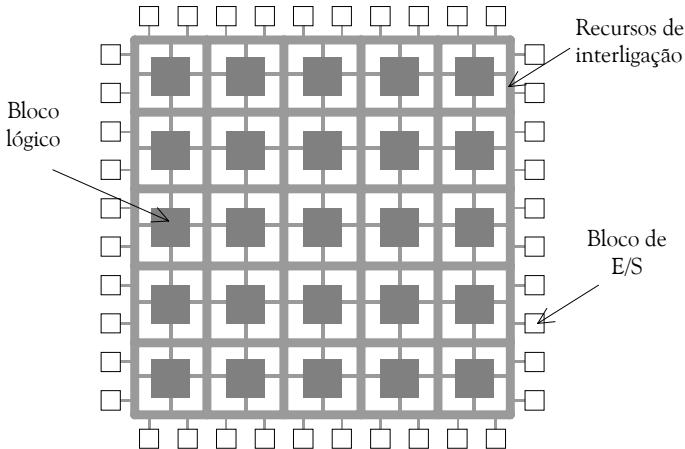


**Figura 2.6:** Arquitectura interna de uma MAX 9000

No caso da primeira LCA, cada bloco lógico era composto por um gerador de funções combinatórias e por um *flip-flop*. O gerador de funções permitia a implementação de duas funções com até três variáveis ou de uma função com quatro variáveis. Cada bloco de E/S podia ser programado para ser uma entrada (directa ou registada), uma saída ou um pino bidireccional. Os recursos de interligação programáveis permitiam a distribuição do sinal de relógio e o encaminhamento local ou global de sinais [Carter, 91].

---

<sup>5</sup> Vários autores usam, em alternativa, a designação “células lógicas” para se referirem ao mesmo ente.



**Figura 2.7:** Arquitectura interna de uma FPGA [Rose et al., 93]

Novas arquitecturas para FPGAs têm sido desenvolvidas desde 1985, com variações significativas na composição dos blocos e das tecnologias de implementação, mas sempre mantendo a mesma estrutura básica da LCA primordial – blocos lógicos, blocos de E/S e recursos de interligação programáveis. Alguns desses avanços são apresentados em [El Gamal et al., 88], [Hsieh et al., 88], [Hastie et al., 90], [Hsieh et al., 90], [Greene et al., 93], [Rose et al., 93], [Trimberger, 93], [DeHon, 94], [Leeser et al., 98], [Campenhout et al., 99], [Chiricescu et al., 01] e [Renovell et al., 01]. As novas arquitecturas oferecem igualmente um conjunto alargado de outras características, como sejam a inclusão de blocos de memória, localizados e distribuídos, barramentos internos com terceiro estado, capacidades de auto-teste e a possibilidade de serem reprogramáveis total ou parcialmente, sem necessidade de se retirar o componente do circuito.

Ao contrário do que se passava com as PALs, a descrição dos circuitos a implementar era, no início, efectuada sob a forma de esquemático. Ferramentas de apoio, desenvolvidas especificamente por cada fabricante de FPGAs, mapeavam posteriormente essa informação nos recursos dos blocos lógicos e de E/S únicos da arquitectura escolhida. Essa lógica era então distribuída pelos blocos da FPGA e a rede de sinais de interligação, definida.

As FPGAs permitem alcançar níveis de integração muito superiores aos das CPLDs, embora à custa de uma maior complexidade das arquitecturas de encaminhamento e de implementação lógica. As arquitecturas de encaminhamento das CPLDs são tipicamente muito simples, já que cada saída é directamente ligável a cada entrada através de um comutador, mas altamente ineficientes. Por sua vez, nas FPGAs, as arquitecturas de encaminhamento são mais eficientes, passando, no entanto, cada ligação normalmente por vários comutadores, no que se assemelha às MPGAs. Por outro lado, enquanto nas CPLDs a lógica é implementada usando predominantemente dois planos lógicos ‘E’–‘OU’, com portas ‘E’ de elevado número de entradas, nas FPGAs, a lógica é implementada empregando múltiplos planos e portas com reduzido número de entradas, o que frequentemente

conduz a implementações mais compactas [Rose et al., 93]. As vantagens enunciadas, fruto sobretudo da extrema flexibilidade das arquitecturas de encaminhamento, presumem um custo: um aumento dos tempos de propagação dos sinais dentro da FPGA. Por cada passagem por um comutador, é adicionado ao sinal um atraso, o que afecta adversamente o desempenho do circuito e torna difícil a sua previsão até o projecto estar completamente implementado. Os atrasos nas interligações não são conhecidos até que as funções a implementar sejam mapeadas nos blocos lógicos, os sinais sejam encaminhados através das interligações disponíveis e o seu atraso de propagação posteriormente calculado, em função dos caminhos usados. Outro problema prende-se com a implementação de funções combinatórias com um número elevado de entradas. Uma vez que os blocos lógicos apresentam tipicamente poucas entradas, a implementação de funções com um grande número de entradas requer a utilização de vários níveis lógicos. Como cada nível adiciona um atraso na propagação dos sinais, o desempenho do sistema é substancialmente degradado. As novas arquitecturas têm incluído recursos dedicados para a resolução destes problemas, nomeadamente estruturas para descodificação rápida de endereços, tipicamente entre 16 e 32 bits, e recursos para a rápida propagação de sinais de transporte<sup>6</sup>, permitindo aumentar a velocidade nas operações de aritmética e a implementação de contadores mais extensos. Estas características, aliadas à utilização de blocos de hardware com mapeamento pré-definido (conhecidos pela designação de *macros*) e apresentando, como tal, um desempenho conhecido *a priori*, têm contribuído para eliminar as limitações causadas pela dependência entre o desempenho e o encaminhamento dos sinais.

Muitos factores determinarão a escolha do tipo de dispositivo (CPLD ou FPGA) que melhor se adapta a uma determinada aplicação. A árvore de decisão que se ilustra na Figura 2.8 inclui alguns dos critérios básicos a considerar quando se pretenda optar por um dos dois tipos. Esta árvore pode fazer parecer que a decisão é evidente, mas o que efectivamente se verifica é que a maioria das aplicações mais complexas envolve compromissos entre os pontos fortes de cada um dos tipos. Muitos outros critérios, externos ao próprio dispositivo, poderão igualmente pesar nessa decisão. Incluem-se entre esses critérios:

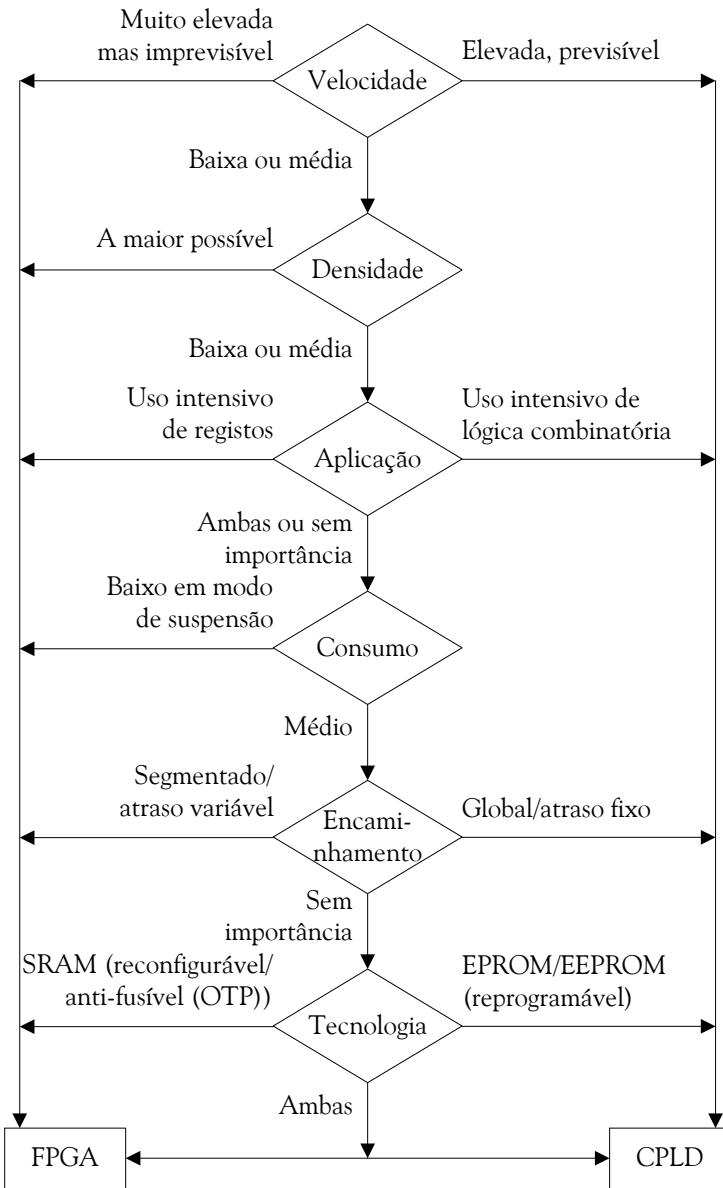
- o custo do dispositivo e das ferramentas de apoio ao projecto;
- a qualidade das ferramentas de apoio ao projecto e a familiaridade do projectista com elas;
- as *macros* disponibilizadas pelo fabricante;
- o encapsulamento;

---

<sup>6</sup> Recursos lógicos do tipo série, dedicados ao deslocamento de valores de transporte resultantes de operações aritméticas, conduzindo a uma maior rapidez na execução dessas operações.

- a disponibilidade de infra-estruturas de teste e programação no circuito.

Da consideração de todos estes critérios e do seu peso relativo dependerá a solução final a adoptar [Swager, 92].



**Figura 2.8:** Árvore de decisão FPGA versus CPLD

## 2.2. ARQUITECTURA DAS FPGAs

A arquitectura das FPGAs divide-se primariamente em duas grandes classes, consoante o tamanho dos blocos lógicos constituintes do dispositivo:

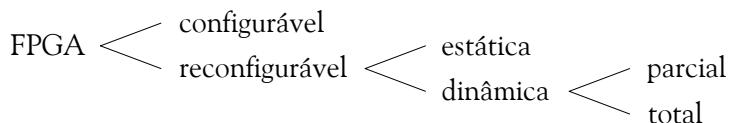
1. arquitecturas de granulosidade grossa;
2. arquitecturas de granulosidade fina.

As arquitecturas de granulosidade grossa baseiam-se em blocos lógicos razoavelmente grandes, contendo tipicamente uma ou mais tabelas de consulta (LUTs - *Look-Up Tables*) e dois ou mais *flip-flops*. As tabelas de consulta, que apresentam, na grande maioria das arquitecturas, quatro entradas, assemelham-se a memórias ROM (*Read Only Memory*) de 16 por 1, permitindo a implementação de lógica combinatória, a exemplo dos primeiros dispositivos lógicos focados na secção anterior. As arquitecturas de granulosidade fina são caracterizadas por blocos lógicos mais simples, com uma ou mais portas lógicas de duas entradas, um ou mais multiplexadores usados na implementação das funções lógicas definidas pelo utilizador, e, em alguns casos, um *flip-flop*.

O uso de tabelas de consulta ou de multiplexadores na implementação de funções lógicas conduz a que alguns autores dividam as FPGAs em dois grupos: FPGAs baseadas em multiplexadores e FPGAs baseadas em tabelas de consulta.

Outra diferença importante entre as arquitecturas refere-se à tecnologia de programação do dispositivo. Actualmente, as FPGAs de maior densidade de integração têm por base a tecnologia das memórias estáticas (SRAM(*Static Random Access Memory*)-based), embora a tecnologia anti-fusível tenha também larga expressão no mercado, dependendo a escolha do tipo de utilização. Existem igualmente FPGAs baseadas em tecnologia EEPROM (*Electrically Erasable Programmable Read Only Memory*) e Flash.

As FPGAs podem ainda ser classificadas quanto às possibilidades de configuração que oferecem, distinguindo-se, neste caso, quatro classes, conforme se representa na Figura 2.9. As FPGAs do tipo OTP designam-se por configuráveis, enquanto as FPGAs reconfiguráveis se subdividem em dois grupos: aquelas cuja configuração é efectuada de forma estática, isto é, em que a FPGA é configurada durante a fase de arranque do sistema, sendo a sua funcionalidade mantida a partir daí sem alterações, e aquelas cuja funcionalidade pode ser modificada a qualquer instante, de forma total ou parcial [Marchal, 99].



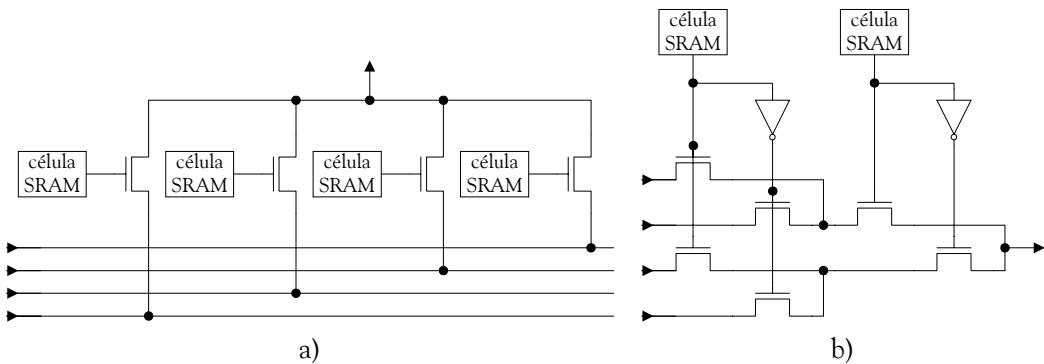
**Figura 2.9:** Classificação das FPGAs em função das possibilidades de configuração

As várias famílias de FPGAs apresentam também diferenças entre si no que diz respeito à arquitectura dos recursos de encaminhamento e ao conteúdo dos blocos de E/S, aspectos que, juntamente com os já elencados, serão alvo de análise mais detalhada nas secções seguintes.

## 2.2.1 TECNOLOGIAS DE PROGRAMAÇÃO

A funcionalidade de uma FPGA é definida através de comutadores eléctricos programáveis cujas propriedades, tamanho, resistência em condução e capacidade parasita, definem a eficiência e o desempenho do dispositivo. Outros atributos a ter em conta são a volatilidade da sua programação, a possibilidade de reprogramação e a complexidade do processo de fabrico. As tecnologias mais usadas na implementação desses comutadores são as tecnologias de programação baseadas em memória estática (SRAM-based), anti-fusível e porta flutuante (*floating gate*).

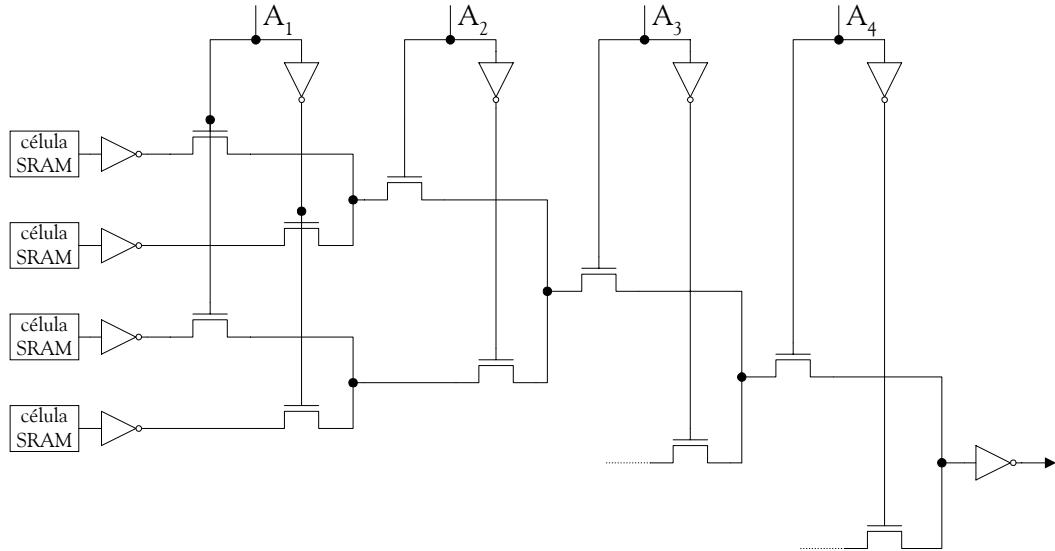
Numa FPGA com tecnologia de programação baseada em memória estática, é o estado das células da memória de configuração que configura a lógica para desempenhar a funcionalidade exigida pelo utilizador. Essas células encontram-se distribuídas entre os elementos lógicos que controlam, para implementar a função lógica e os recursos de encaminhamento que interligam os vários blocos lógicos. Presentemente, esta é a tecnologia de programação utilizada nas FPGAs de maior densidade de integração. Os comutadores que possibilitam o estabelecimento das interligações, denominados por alguns autores *Programmable Interconnect Points* (PIPs), são transístores de passagem controlados pelo estado de uma célula de configuração. Esses transístores estabelecem ou não a conexão entre os segmentos a eles ligados, dependendo do valor presente nas respectivas células. Os transístores de passagem comportam-se como um elemento resistivo-capacitivo (RC) e, como tal, implicam a introdução de um atraso na propagação dos sinais que os atravessam. Esses transístores controlam ligações ponto-a-ponto (unidireccionais ou bidireccionais) ou constituem multiplexadores, conforme se ilustra na Figura 2.10.



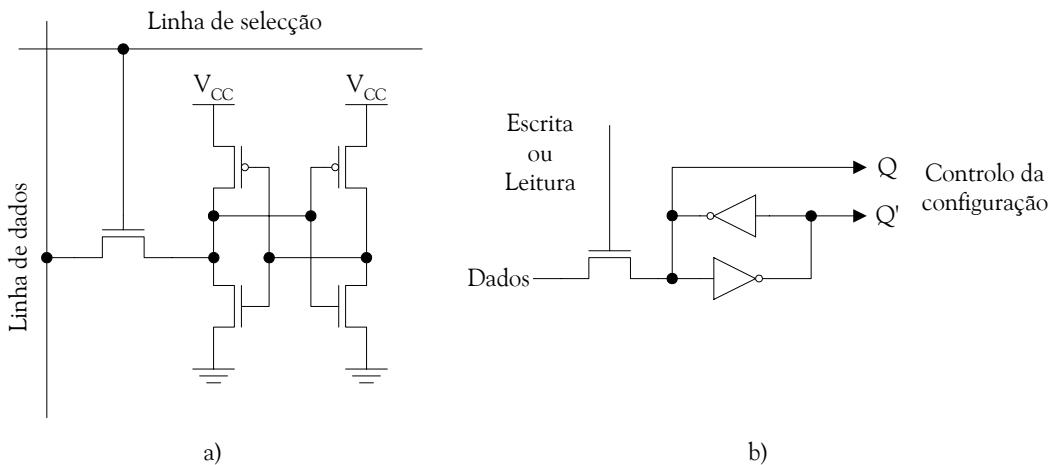
**Figura 2.10:** Blocos de interligação a) ponto-a-ponto e b) usando um multiplexador

A implementação das tabelas de consulta processa-se de forma idêntica, como se ilustra na Figura 2.11 [Chow et al., 99a]. Na Figura 2.12, apresenta-se o exemplo de uma célula de configuração SRAM da Xilinx. Esta célula é construída com base em dois inversores ligados em anti-paralelo, cujas saídas controlam o estado de transístores de passagem, definem valores à

entrada ou dos sinais de controlo de elementos lógicos (portas, multiplexadores ou *flip-flops*), ou guardam o estado de cada posição das tabelas de consulta.



**Figura 2.11:** Implementação de uma tabela de consulta



**Figura 2.12:** Célula de configuração SRAM da Xilinx com cinco transístores

a) implementação e b) equivalente lógico

As vantagens da utilização da tecnologia de programação baseada em memória estática prendem-se com a possibilidade de reutilização do dispositivo durante a fase de desenvolvimento e com o facto de o sistema poder ser produzido e só posteriormente os seus componentes de lógica programável configurados, usando as facilidades oferecidas pela programação no circuito (ISP). Esta tecnologia é igualmente útil no caso de actualizações posteriores, pois, em vez de ser enviado ao cliente um novo circuito integrado (CI), é-lhe enviado um novo ficheiro de configuração. No caso de o sistema se encontrar ligado à Internet, pode ser actualizado directamente pelo fabricante sem intervenção do cliente, inclusive *on-the-fly*, isto é, mantendo-se sem interrupção de funcionamento as partes do

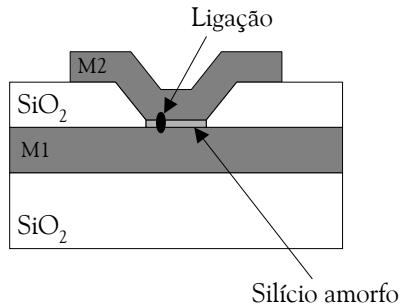
sistema não afectadas pela reconfiguração, mesmo que utilizem recursos do próprio dispositivo em reconfiguração [Maxfield, 96].

A tecnologia de programação baseada em SRAM apresenta uma desvantagem óbvia, a da sua volatilidade. Quando a alimentação é desligada, a FPGA perde a sua programação, pelo que tem de ser reprogramada a cada arranque do sistema. Para sistemas que necessitem de lógica activa durante esta fase, uma FPGA baseada nesta tecnologia de programação não é uma opção viável. No entanto, as FPGAs deste tipo incluem sensores capazes de detectar a aplicação da alimentação, inicializando-se automaticamente, desde que a aplicação possa esperar algumas dezenas de milisegundos, requeridos para a sua programação. Tal implica a existência de uma memória externa, tipicamente do tipo PROM, onde se encontre armazenado o ficheiro de configuração da FPGA (o que implica a anulação da sua flexibilidade de reconfiguração, pelo menos no arranque do sistema), ou de um processador que se responsabilize por fornecer, a partir de uma fonte externa, o ficheiro de configuração. O espaço total ocupado pelas células de configuração e pelos transístores de passagem é superior ao dos dispositivos cuja tecnologia de programação se baseia em anti-fusíveis [Smith, 97a].

Embora a tecnologia de programação anti-fusível tenha a vantagem de ser uma tecnologia não volátil, apresenta o inconveniente de ser programável uma única vez (OTP). Um anti-fusível é um elemento programável de dois terminais, que, ao ser-lhe aplicada uma tensão de programação (tipicamente superior aos níveis de tensão habituais nos CIs), vê irreversivelmente alterada a sua impedância de condução, que passa de um valor elevado para um valor muito baixo. A baixa resistência em condução apresentada pelos anti-fusíveis e o seu pequeno tamanho tornam notavelmente adequado o seu emprego como tecnologia de programação das FPGAs. Uma vez que os anti-fusíveis se encontram inicialmente em estado de não condução, a definição da funcionalidade do circuito a implementar na FPGA implicará, em média, a passagem ao estado de condução de apenas 2% da sua totalidade, pelo que o tempo de programação será muito curto em comparação com o necessário para a programação da totalidade da memória de configuração de uma FPGA baseada em tecnologia SRAM [Greene et al., 93].

As primeiras experiências com anti-fusíveis, que datam de 1957, tinham em vista o seu uso no fabrico de memórias. A sua evolução processou-se em duas direcções distintas, desembocando em dois tipos construtivos: os anti-fusíveis de silício amorfo e os de dieléctrico. Os primeiros são constituídos por uma camada de silício amorfo, colocada entre duas camadas de metal, que se torna condutora quando atravessada por uma corrente. A Figura 2.13 ilustra em corte a constituição de um anti-fusível de silício amorfo [Smith, 97a]. Em virtude de a ligação se formar entre duas

camadas de metal, alguns autores designam alternativamente este tipo de anti-fusíveis por metal-metal.



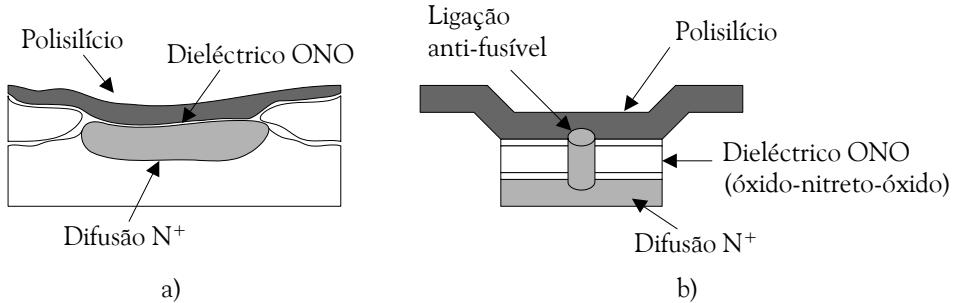
**Figura 2.13:** Constituição de um anti-fusível de silício amorfo

Dois problemas afectam os anti-fusíveis deste género: o primeiro tem a ver com a possibilidade de reversibilidade da programação por aplicação de uma corrente inversa, transformando o anti-fusível de condutor em não condutor; o segundo prende-se com a elevada corrente de fugas que flui através dos elementos não programados, o que, numa FPGA, constitui um grave problema, já que a tensão de alimentação se encontra aplicada aos terminais de, em média, metade dos anti-fusíveis. As suas vantagens residem no facto de a ligação entre as camadas de metal do anti-fusível e as camadas de metal das interligações ser directa, não conduzindo ao aparecimento de capacidades parasitas, e de a resistência em condução se situar em redor dos  $80\Omega$ , para uma corrente de programação de 15mA.

Os anti-fusíveis de dieléctrico consistem numa camada de material dieléctrico colocada entre uma camada tipo N<sup>+</sup> e uma de polisilício, conforme se ilustra na Figura 2.14. O dieléctrico é constituído por duas finas camadas de óxido de silício (SiO<sub>2</sub>) intercaladas por uma camada de nitreto de silício (Si<sub>3</sub>N<sub>4</sub>). Após a aplicação de uma tensão suficientemente elevada (usualmente 16V), o dieléctrico é perfurado, fundindo e criando uma ligação condutora de silício policristalino entre os eléctrodos. Tipicamente, apenas uma ligação é formada através do dieléctrico, com um raio que depende da corrente de programação. Correntes mais elevadas conduzem a raios maiores e, consequentemente, a menor resistência em condução.

Os anti-fusíveis de dieléctrico apresentam uma resistência em condução de cerca de  $500\Omega$ , para uma corrente de 5mA, valor mais elevado do que o apresentado pelos seus congêneres de silício amorfado. No entanto, a sua corrente de fugas, quando não programado, é de cerca de 1fA, valor negligenciável mesmo em FPGAs de grande capacidade. A capacidade parasita de cada anti-fusível programado ronda os 10fF com tecnologia CMOS (*Complementary Metal-Oxide-Semiconductor*) de 1,2  $\mu\text{m}$ . Este valor inclui a contribuição dos eléctrodos de polisilício e de difusão e a sua ligação ao

metal das camadas de interligação. A programação é irreversível e insensível à radiação [Smith, 97a].



**Figura 2.14:** Constituição de um anti-fusível de dieléctrico a) secção e b) esquema simplificado

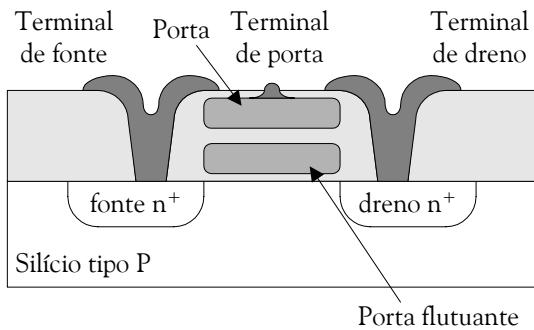
O tamanho do anti-fusível é limitado pelo tamanho do equipamento litográfico usado para fabricar os CIs. Os anti-fusíveis de dieléctrico requerem, devido à natureza demasiado resistiva do polisilício e da difusão, o uso de contactos para ligação ao metal das camadas de interligação que ocupam mais espaço do que o próprio anti-fusível, sobrepondo-se à vantagem da sua pequenez. Mesmo assim, os anti-fusíveis são geralmente tão pequenos que são as regras de espaçamento entre segmentos de metal e contactos que limitam a proximidade entre eles, e não o seu próprio tamanho.

Uma das suas desvantagens mais notórias é a impossibilidade de programação da FPGA no circuito, necessitando, por isso, de passar por um dispositivo próprio para o efeito, antes de ser montada na CCI. De igual forma, actualizações posteriores implicam a substituição do componente, situação impraticável na maioria dos casos.

A tecnologia de programação EEPROM (ou E<sup>2</sup>PROM) e Flash tem sido preferencialmente usada nas CPLDs, embora esporadicamente tenha aparecido também aplicada às FPGAs. Uma célula de uma EEPROM é um transístor MOS (Metal-Oxide-Semiconductor) que armazena carga numa placa condutora electricamente isolada – ou porta flutuante – a qual, localizada acima do canal do transístor, cria, quando carregada, um campo eléctrico que modifica a condutividade desse canal. A estrutura desta célula encontra-se ilustrada na Figura 2.15 [Hauck, 98].

A carga na porta flutuante, composta de electrões, é aumentada ou diminuída por acção de um mecanismo de condução de corrente denominado efeito de tunelamento de Fowler-Nordheim [Constancias, 98]. A camada de óxido de silício, que é um excelente isolante sob níveis baixos de tensão de polarização, torna-se suficientemente condutora quando sujeita a tensões de polarização elevadas (tipicamente 12 a 14V), permitindo a passagem dos electrões, e carregando ou descarregando a porta flutuante. Esta corrente de tunelamento varia exponencialmente com a tensão aplicada, cuja polaridade determina a carga ou descarga da porta. O potencial da porta

flutuante e os potenciais através do óxido são manipulados por acoplamento capacitivo electrostático, por meio da porta de controlo e pelos outros terminais do transístor. Quando os valores de polarização são reduzidos, o óxido volta a comportar-se como um excelente isolante. Sob esta condição, a quantidade de carga na porta flutuante fica retida, mantendo-se neste estado por muitos anos, mesmo se sujeita a temperaturas elevadas. A tensão na porta flutuante pode agora ser “lida” sem perturbar a carga, e, como tal, sem alterar a tensão armazenada. As EEPROMs comportam-se, pois, como memórias não voláteis, que podem ser apagadas e reprogramadas repetidamente, embora esta possibilidade de reprogramação esteja limitada a poucas centenas de vezes, devido à degradação que o efeito de tunelamento provoca na camada de óxido [Altera, 98].



**Figura 2.15:** Constituição interna de uma EEPROM

A tecnologia *Flash* é uma variação da tecnologia EEPROM caracterizando-se por possuir células que ocupam cerca de metade da área ocupada por uma célula EEPROM normal. O mecanismo de apagamento das células é mais simples e rápido que o das EEPROMs tradicionais, mas, ao contrário destas, em que cada posição de memória pode ser reprogramada individualmente, nas *Flash*, a reprogramação implica o apagamento em bloco de um grande número de células.

À vantagem da não volatilidade e da reprogramabilidade destas duas tecnologias de programação em relação às FPGAs baseadas em SRAM, contrapõem-se o maior tempo de reprogramação e a limitação do número de vezes em que tal pode ocorrer. Em ambos os casos, a geração das tensões de programação é efectuada internamente nos dispositivos, pelo que é possível tirar todo o partido da programação no circuito.

Actualmente, existem disponíveis FPGAs baseadas em tecnologia *Flash*. O único fabricante que possuía FPGAs baseadas em tecnologia EEPROM desapareceu recentemente do mercado.

## 2.2.2 ARQUITECTURA DOS BLOCOS LÓGICOS

A natureza dos blocos lógicos tem uma influência particular sobre o desempenho e a densidade lógica de uma FPGA. A quantidade de lógica a incluir em cada bloco lógico, o tamanho das tabelas

de consulta e o seu número, têm sido alvo de inúmeros estudos que visam maximizar o desempenho das FPGAs, destacando-se entre eles os de [Rose et al., 89], [Rose et al., 93], [Betz et al., 98], [Betz et al., 99], [Chow et al., 99], [Marquardt et al., 99], [Ahmed et al., 00] e [Marquardt et al., 00]. Como regra básica, as FPGAs devem ter uma granulosidade tão fina quanto possível, mas mantendo uma elevada disponibilidade no encaminhamento dos sinais e um baixo atraso na sua propagação. Os módulos devem permitir a implementação eficiente de uma grande variedade de funções, com atrasos pequenos e ocupação de uma área diminuta [Greene et al., 93].

A fronteira entre granulosidade fina e granulosidade grossa aparece estabelecida diferentemente consoante os autores, tendo-se deslocado no sentido do aumento da complexidade dos blocos lógicos à medida que a evolução das arquitecturas se foi afastando da ideia inicial da *sea-of-gates* (grande número de pequenas células dispostas de forma regular, algumas vezes consistindo apenas num par de transístores), herdada das MPGAs. Actualmente, as arquitecturas de granulosidade fina englobam os blocos lógicos constituídos por um pequeno número de portas lógicas, um ou mais multiplexadores e, em alguns deles, um *flip-flop*. A Figura 2.16 ilustra um destes casos, o ACT1 da Actel [Actel, 99], um bloco constituído apenas por três multiplexadores e uma porta lógica de duas entradas. O ACT1 permite a implementação de um vasto conjunto de lógica combinatória, de duas, três e quatro entradas, e de lógica sequencial, usando um bloco para estabelecer um elemento de retenção (*latch*) transparente ou dois para um *flip-flop*. Neste tipo de blocos, as funções lógicas definidas pelo utilizador são implementadas com base em multiplexadores. Por exemplo, a função lógica Booleana

Equação 2.1

$$F = A \cdot B + \bar{B} \cdot C + D$$

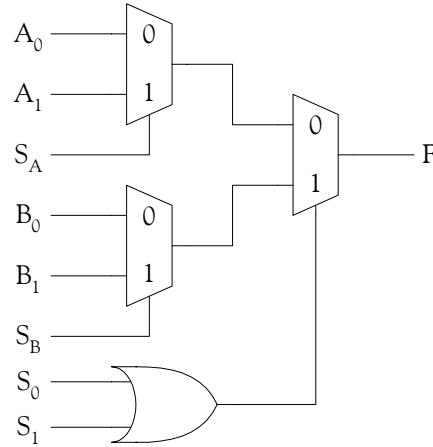
resulta na implementação ilustrada na Figura 2.17<sup>7</sup>.

A Figura 2.18 ilustra a arquitectura de um outro bloco pertencente a uma FPGA da ATMEL, a AT6000 [Atmel, 99]. A exemplo do anterior, este bloco é constituído por multiplexadores e portas lógicas de duas entradas, embora em maior número, e incorpora já um *flip-flop*. As entradas  $A_x$  e  $B_x$  e as saídas  $A$  e  $B$  permitem a ligação aos quatro blocos adjacentes lateralmente, enquanto as entradas e saídas  $L_x$  possibilitam o encaminhamento dos sinais para blocos mais afastados.

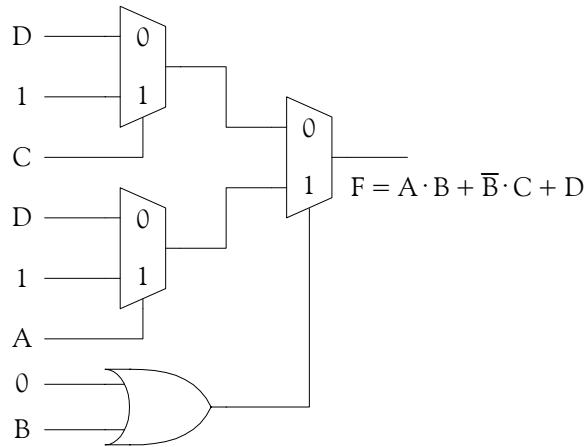
A grande vantagem da utilização de blocos lógicos de granulosidade fina é o elevado índice de ocupação que se atinge em cada bloco, ainda que a ele se contrapõha a necessidade de um maior

<sup>7</sup> O estudo da implementação de funções lógicas usando multiplexadores sai fora do âmbito deste trabalho. Uma explanação da teoria de suporte a este método pode ser encontrada em [Sandige, 90] e [Smith, 97a].

número de recursos de encaminhamento e de comutadores programáveis, de que resulta um aumento dos custos, quer em termos de área, quer em termos do atraso de propagação.



**Figura 2.16:** Arquitectura de um bloco de granulosidade fina baseado em multiplexadores, o ACT1 da Actel

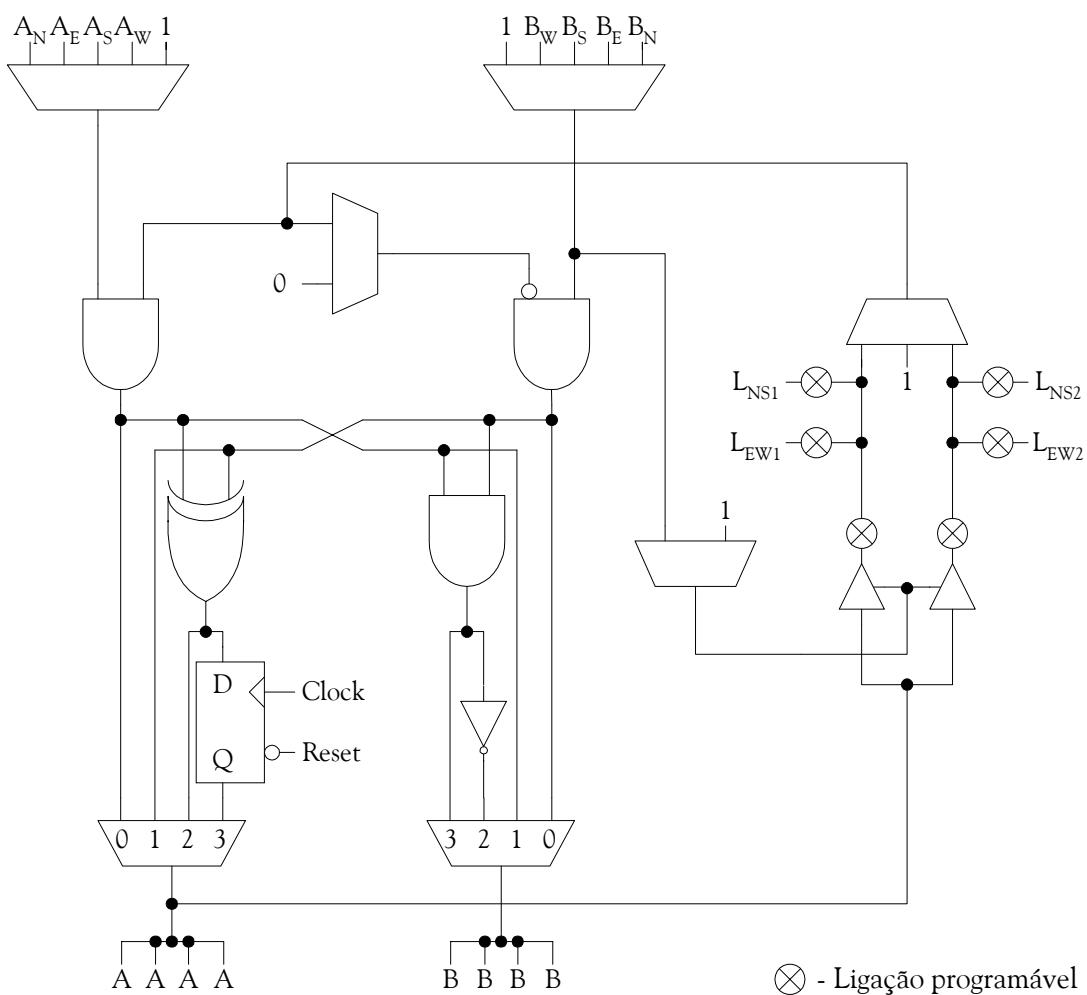


**Figura 2.17:** Exemplo da implementação de uma função Booleana no ACT1

As arquitecturas de granulosidade grossa caracterizam-se por uma maior complexidade da lógica presente nos blocos e pela inclusão de tabelas de consulta, elementos onde as funções lógicas combinatórias definidas pelo utilizador são implementadas. O exemplo apresentado na Figura 2.19 é um bloco lógico XC3000 CLB (Configurable Logic Block) da Xilinx [Xilinx, 02a].

A função combinatória presente no bloco XC3000 é constituída por uma tabela de consulta de 32 entradas, formada por 32 células SRAM, que permite a implementação de uma qualquer expressão Booleana de cinco variáveis. Esta função possui sete entradas: as cinco entradas do bloco (A, ..., E) e as saídas de ambos os *flip-flops*, QX e QY, e duas saídas, F e G. Uma vez que a tabela de consulta requer apenas cinco variáveis para formar um único endereço ( $32=2^5$ ), várias configurações são possíveis:

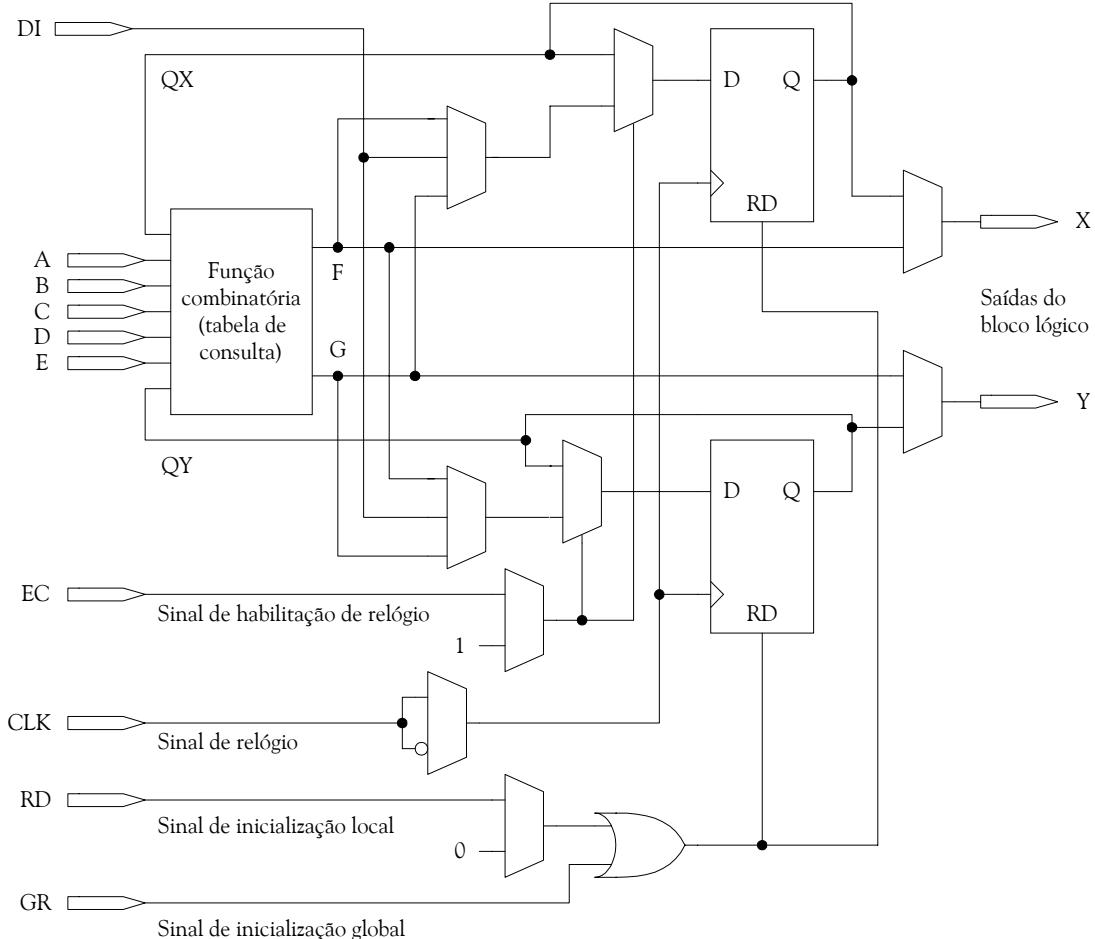
- o uso de cinco das sete entradas possíveis ( $A, \dots, E, QX, QY$ ) como variáveis de entrada, situação em que a tabela é usada em toda a sua extensão, sendo as saídas  $F$  e  $G$  iguais;
- a divisão da tabela ao meio e a implementação de duas funções de quatro variáveis cada, escolhidas das sete de entrada; em cada metade da tabela, duas pertencem obrigatoriamente às cinco de entrada no bloco ( $A, \dots, E$ ), pelo que uma das funções ligará à saída  $F$  e a outra, à  $G$ ;
- a divisão da tabela ao meio, usando uma das sete variáveis de entrada como linha de seleção entre as saídas  $F$  e  $G$ , o que permite a implementação de algumas funções de seis ou sete variáveis.



**Figura 2.18:** Arquitectura de um bloco de granulosidade fina baseado em multiplexadores e contendo um *flip-flop*, o AT6000 da Atmel

A simetria das funções  $F$  e  $G$  e dos *flip-flops* permite a troca das saídas do bloco, facilitando a optimização da utilização dos recursos de encaminhamento entre blocos lógicos e entre estes e os blocos de E/S. Os multiplexadores sem linhas de seleção presentes no bloco lógico são

multiplexadores programáveis, sendo a selecção da entrada efectuada pelo valor presente nas células respectivas da memória de configuração.

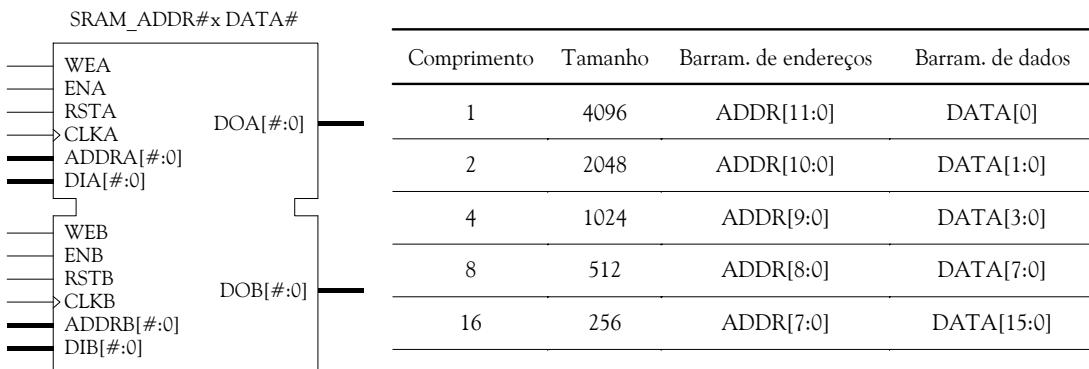


**Figura 2.19:** Arquitectura de um bloco de granulosidade grossa baseado em tabelas de consulta (CLB da família XC3000 da Xilinx)

A grande vantagem das tabelas de consulta é a sua elevada versatilidade, já que uma tabela com  $n$  entradas permite a implementação de uma qualquer função com esse número de entradas. A desvantagem, por sua vez, reside no facto de, para valores de  $n$  superiores a cinco, as tabelas de consulta se tornarem inaceitavelmente extensas, uma vez que o número de células de memória requeridas cresce exponencialmente com o número de entradas ( $2^n$ ). Em vez de se aumentar o número de entradas, criando blocos lógicos maiores, é preferível o agrupamento de várias tabelas de consulta num único bloco e a sua interligação usando recursos de encaminhamento locais. Visto que estes são mais rápidos que os recursos de encaminhamento genérico entre blocos, blocos com várias tabelas de consulta contribuem para um aumento da velocidade da FPGA. Adicionalmente, se um bloco contém várias tabelas, a implementação de um circuito requererá menos blocos do que os que seriam necessários no caso de cada bloco conter apenas uma tabela, reduzindo-se, em consequência, as necessidades de encaminhamento [Betz et al., 98].

À medida que a granulosidade dos blocos lógicos aumenta, o número de blocos necessário para implementar um dado circuito decresce, decrescendo igualmente os atrasos de propagação, fruto da maior utilização dos recursos de encaminhamento locais. Por outro lado, blocos maiores possuem mais lógica e, como tal, ocupam uma maior área. Este facto sugere a existência de um compromisso óptimo entre a granulosidade do bloco, a área necessária para a sua implementação e os atrasos de propagação. Vários estudos apontam que o valor óptimo das variáveis envolvidas se situa em quatro tabelas de consulta por bloco lógico, com quatro entradas cada [Rose et al., 93], [Klein, 94], [Marquardt et al., 99] e [Ahmed et al., 00], sendo essa a solução adoptada por alguns fabricantes nas suas famílias de FPGAs mais recentes.

Os circuitos implementados nas FPGAs requerem frequentemente blocos de memória cujo tamanho varia consoante a aplicação [Brown, 96] e [Bursky, 96]. Blocos de memória SRAM, normalmente de duplo porto e com sinais de controlo independentes, e com o comprimento da palavra configurável para 1, 2, 4, 8 ou 16 bits, com tamanho variável entre 256 e alguns milhares de bits cada e com o seu número a ir desde os quatro blocos por FPGA até algumas centenas, têm sido adicionados às arquitecturas mais recentes. Estes blocos, dispersos uniformemente pela FPGA ou reunidos em colunas intercaladas com os blocos lógicos, permitem a implementação eficiente de aplicações que requerem funções de manipulação e transformação intensiva de dados. Na Figura 2.20, encontram-se ilustradas a representação esquemática e as várias possibilidades de configuração de um bloco de memória da família Virtex-E da Xilinx [Xilinx, 02a]. Algumas arquitecturas apresentam também tabelas de consulta passíveis de serem configuradas para funcionar como blocos de memória, quer de forma separada, quer em conjunto com outras tabelas do mesmo bloco, proporcionando maior capacidade ou permitindo dispor de duplo porto.



**Figura 2.20:** Representação esquemática e possibilidades de configuração dos blocos de SRAM da família Virtex-E da Xilinx

A implementação de lógica aritmética (acumuladores, subtractores, multiplicadores, comparadores, contadores, registos de deslocamento) beneficia igualmente da existência, em várias arquitecturas

de recursos de encaminhamento dedicados para propagação dos sinais de transporte. Isso possibilita a criação de registo extensos sem incorrer nas penalizações inerentes ao atraso de propagação associado ao uso de recursos genéricos de encaminhamento, aumentando, consequentemente, a velocidade nas operações de aritmética. Uma proposta de modificação da estrutura dos blocos lógicos por forma a permitir uma mais eficiente implementação de multiplicadores é apresentada em [Kaviani et al., 98]. Em certos casos, os recursos de transporte dedicados podem também ser usados para interligar em cascata geradores de funções, permitindo, assim, uma eficiente implementação de funções com grande número de variáveis.

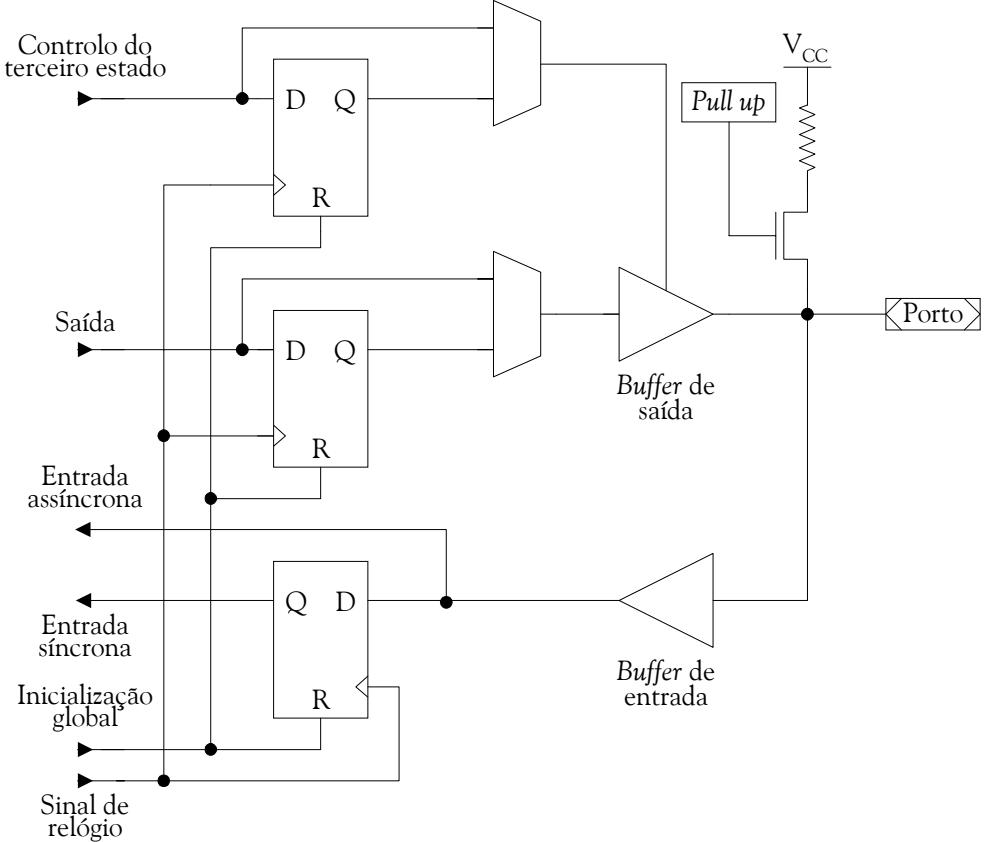
Nas arquitecturas actuais, os blocos lógicos primam pela homogeneidade dentro de cada FPGA. No entanto, alguns estudos sugerem que um aproveitamento mais eficiente da área de implementação seria conseguido se se optasse por matrizes de blocos não homogéneas. Em [He et al., 93], é apresentado um estudo em que tabelas de consulta com diferente número de entradas (2, 3, 4, 5 e 6) são combinadas aos pares e em número diverso num mesmo bloco lógico, demonstrando que certas combinações resultam na obtenção de ganhos significativos em termos de área e de velocidade. Soluções híbridas, misturando blocos lógicos baseados em tabelas de consulta, característicos das FPGAs, com blocos do tipo PAL, característicos das CPLDs, são igualmente propostas em [Kaviani et al., 96] e [Kaviani et al., 99], atestando ganhos significativos em termos de área e de desempenho das funções implementadas relativamente a uma solução que use apenas blocos lógicos com tabelas de consulta de quatro entradas. O uso de blocos do tipo PAL permite reduzir o número de planos lógicos na implementação de funções com maior número de entradas, conduzindo a uma diminuição da ocupação dos recursos de encaminhamento. Daí resulta uma redução dos tempos de propagação por diminuição do número de PIPs atravessados pelos sinais, o que contribui para um aumento da velocidade e para uma diminuição da área ocupada. Por outro lado, a manutenção de blocos baseados em tabelas de consulta permite conservar as vantagens associadas ao uso das FPGAs.

### 2.2.3 ARQUITECTURA DOS BLOCOS DE ENTRADA/SAÍDA

A arquitectura básica dos blocos de E/S pouco difere nas várias famílias de FPGAs, tendo-se a sua funcionalidade mantido praticamente estável ao longo dos anos. A Figura 2.21 ilustra a arquitectura genérica de um bloco de E/S. Cada bloco de E/S controla um pino, que pode ser configurado como entrada, saída ou bidireccional. A entrada pode ser directa ou registada, o mesmo acontecendo com a saída e com o sinal de controlo do terceiro estado.

Outras características comuns são a possibilidade de configuração do *flip-flop* como elemento de retenção transparente, a inclusão de inversores programáveis, que permitem a alteração da

polaridade do sinal de saída e de controlo do terceiro estado, e a possibilidade de controlar o tempo de subida e descida do sinal de saída (*slew rate*). Circuitos de *pull-up* e *pull-down* programáveis permitem a determinação do nível lógico em pinos não usados e durante o intervalo de configuração.



**Figura 2.21:** Arquitectura genérica dos blocos de E/S

Em arquitecturas mais recentes surgiu o conceito de *bloco de E/S enterrado*, que se caracteriza por o porto não ligar a um pino exterior, sendo antes o sinal que chega ao bloco realimentado para a matriz lógica através do *buffer* de entrada. Este conceito permite um aumento do número de *flip-flops* disponíveis para implementação de aplicações baseadas no uso intensivo de registos. O aumento do número de pinos, em resposta a aplicações mais exigentes, tem-se deparado com limitações, em termos de espaço na periferia, para a implementação de mais blocos de E/S. [Marck et al., 98] apresentam uma proposta de solução para este problema sugerindo a disseminação regular por toda a área da matriz lógica da FPGA de blocos de E/S, a qual demonstra vantagens em termos da diminuição do comprimento das interligações, com consequente redução dos tempos de propagação.

Em termos de estrutura dos blocos, a alteração mais significativa prendeu-se com a introdução de células pertencentes ao registo de varrimento pela periferia (*Boundary-Scan - BS*), componente da

infra-estrutura de teste definida na norma IEEE 1149.1, conhecida pela designação *Boundary-Scan Test* (BST) [IEEE 1149.1, 01]. Esta norma descreve a lógica de teste embutida num circuito integrado para permitir o teste das interligações entre componentes após a sua incorporação numa carta de circuito impresso, o teste dos próprios componentes e a observação e modificação da actividade do circuito durante a sua operação normal.

As mudanças mais profundas verificaram-se ao nível das características eléctricas, em resposta ao surgimento no mercado de um grande número de normas de transmissão diferentes, adaptadas às novas exigências de velocidade. Cada novo barramento, criado para suporte específico de uma nova aplicação no sector da electrónica digital, tornou-se numa norma de facto, que possuía, na maior parte dos casos, as suas próprias especificações em termos de valores de corrente e tensão, velocidade, tempos de subida e descida dos sinais e técnicas de terminação. As especificações eram fornecidas a outros fabricantes que desenvolviam componentes para interface com essas aplicações. Entre eles incluem-se os fabricantes de FPGAs, que, tentando manter a sua generalidade, se viam obrigados a suportar o maior número possível dessas normas. Enquanto nas primeiras FPGAs os buffers de entrada eram apenas compatíveis com a tensão de limiar das tecnologias TTL (*Transistor-Transistor Logic*) ou CMOS, nas mais recentes, os blocos de E/S suportam mais de uma dezena de normas diferentes.

#### 2.2.4 ARQUITECTURA DOS RECURSOS DE ENCAMINHAMENTO

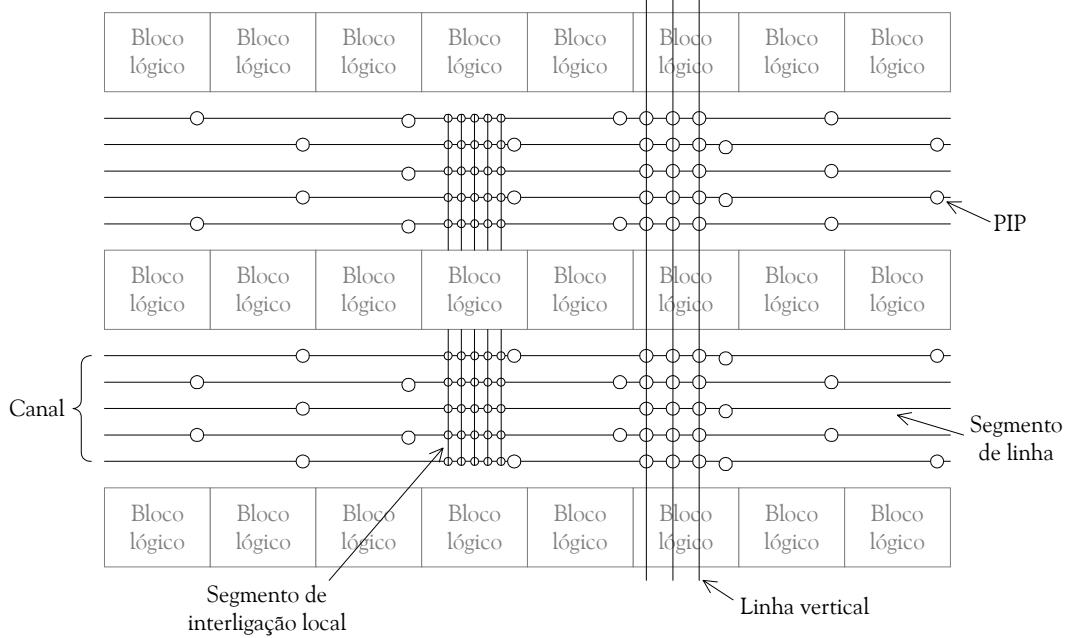
A arquitectura dos recursos de encaminhamento numa FPGA descreve a forma como os PIPs e os segmentos de linha são distribuídos e usados para permitir a interligação programável dos blocos lógicos. Os recursos de interligação incluem os blocos de interligação locais, que permitem a ligação dos segmentos de linha às entradas e saídas dos blocos lógicos, e os blocos de interligação global, ou de comutação, que permitem o encaminhamento dos sinais ao longo dos segmentos verticais e/ou horizontais que se estendem por dois ou mais blocos lógicos. Enquanto, em algumas arquitecturas, os dois blocos constituem entidades distintas, noutras, encontram-se associados num mesmo ente.

Uma arquitectura de encaminhamento típica engloba segmentos de comprimento diferente, interligáveis através de PIPs, cujo número e relação afectam a densidade e o desempenho da FPGA. Dois critérios devem ser satisfeitos na concepção de uma estrutura deste género: velocidade e capacidade de encaminhamento. Por capacidade de encaminhamento entende-se a possibilidade de uma FPGA acomodar toda a rede de interligações de uma aplicação típica, apesar de os segmentos de linha estarem definidos *a priori*. Só os PIPs que permitem a interligação entre os segmentos podem ser programados para acomodar especificamente a aplicação, não o número, comprimento e localização desses segmentos [Greene et al., 93]. Um número demasiado pequeno

de segmentos pode levar a que apenas seja possível utilizar uma pequena fração dos blocos lógicos, enquanto um número excessivo levará a um desperdício da área de silício. Outro factor importante, com influência na eficiência da FPGA, é o atraso de propagação, que dependerá do número de PIPs introduzidos no caminho de um determinado sinal, já que o atraso máximo de um sinal definirá a frequência máxima de funcionamento da aplicação. O encaminhamento é responsável por uma parte significativa do atraso total de um sinal (em média 40 a 60%) [Sharma, 98].

As arquitecturas de encaminhamento dividem-se basicamente em três tipos: assimétricas ou em linha (*row-based*), simétricas ou em ilha (*island-style*) e hierárquicas. De salientar que esta divisão não é consensual nem estanque, variando consoante os autores e os fabricantes.

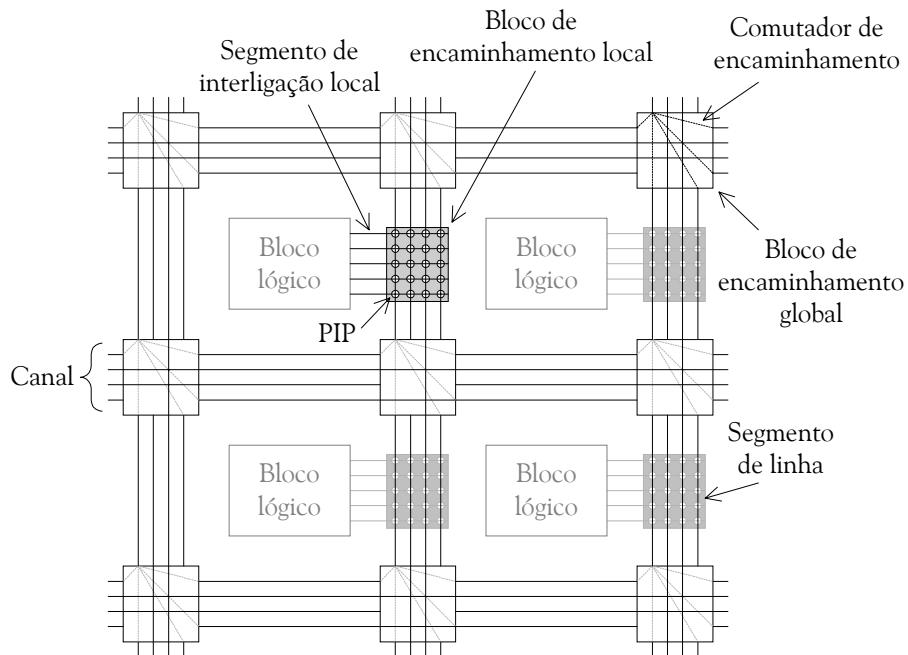
A arquitectura assimétrica caracteriza-se por possuir diversos canais horizontais, intercalados com as linhas de blocos lógicos, com um grande número de segmentos horizontais de comprimento variável e interligáveis por PIPs, e dois tipos básicos de segmentos verticais, os que estabelecem a ligação entre as entradas e saídas dos blocos lógicos e os segmentos horizontais, e as linhas que normalmente abrangem toda a altura da matriz lógica e que permitem a interligação entre segmentos horizontais de diferentes canais. As linhas verticais são em menor número que as horizontais, razão pela qual se designa de arquitectura assimétrica. A Figura 2.22 ilustra esquematicamente esta arquitectura.



**Figura 2.22:** Arquitectura de encaminhamento assimétrica

Na arquitectura simétrica, os blocos lógicos estão rodeados nos seus quatro lados por idêntico número de segmentos de diferentes comprimentos, conforme se representa na Figura 2.23. As entradas ou saídas de cada bloco podem interligar-se a um ou mais dos segmentos do canal

adjacente através do bloco de encaminhamento local. Um conjunto de comutadores de encaminhamento, agrupados nos blocos de encaminhamento global situados em cada uma das intersecções entre os canais horizontais e verticais, asseguram o encaminhamento de sinais entre segmentos do mesmo canal ou do canal transversal. Como os segmentos apresentam comprimentos diferentes, alguns atravessam os blocos de comutação sem serem interrompidos, embora possam ter uma ou mais derivações intermédias. Segmentos mais pequenos podem ser interligados entre si para formarem linhas mais longas, à custa de um aumento do tempo de propagação, devido ao atraso introduzido por cada comutador atravessado.



**Figura 2.23:** Arquitectura de encaminhamento simétrica

A arquitectura hierárquica caracteriza-se pela divisão da matriz de blocos lógicos em secções, às quais correspondem recursos de encaminhamento próprios e que, por sua vez, se interligam a outras secções através de um outro conjunto de recursos de nível superior. Este tipo de arquitectura é característica das CPLDs, tendo o seu uso sido abandonado nas FPGAs.

Dada a importância que a arquitectura dos recursos de encaminhamento tem no aproveitamento eficiente da área e no desempenho de uma FPGA, alguns estudos têm sido conduzidos no sentido de determinar qual a melhor arquitectura e qual o número óptimo de segmentos e o seu comprimento, bem como a relação numérica que entre eles deve existir em cada canal, e o número e organização dos comutadores, quer nos blocos de encaminhamento local, quer nos blocos de encaminhamento global.

Numa FPGA, o atraso de propagação introduzido num sinal é devido mais ao número de PIPs que ele atravessa e à carga introduzida pelo próprio segmento do que à distância que ele percorre, podendo esse atraso ser diminuído pela introdução de segmentos mais longos e pela utilização de *buffers* que colmatem o aumento da carga. A redução do número de PIPs na interligação reduz a flexibilidade de encaminhamento enquanto o seu aumento a melhora, à custa, porém, de uma maior carga e de um maior dispêndio de área [Trimberger, 93].

Num estudo apresentado em [Brown et al, 96], efectuado no sentido de determinar quais as características óptimas de uma arquitectura de encaminhamento simétrica relativamente à velocidade dos circuitos implementados e ao uso efectivo dos recursos de interligação disponíveis, chegou-se às seguintes conclusões:

- o atraso de propagação médio num circuito diminui em cerca de 40% à medida que o comprimento dos segmentos aumenta de um para quatro blocos lógicos; no entanto, o uso de segmentos com comprimento superior a quatro provoca uma ligeira degradação do desempenho; o atraso de propagação diminui inicialmente à medida que o número de PIPs atravessados por um sinal diminui, mas, se os segmentos se tornarem mais extensos que o necessário para interligar o sinal entre bloco de origem e o de destino, o excesso de carga introduzido pelo maior comprimento do segmento resulta num ligeiro aumento do atraso;
- a maioria dos segmentos num canal deve ter comprimento superior a um bloco lógico;
- o aumento do número de segmentos por canal não só desperdiça área, mas também degrada o desempenho em termos de velocidade, pois segmentos adicionais implicam um aumento da capacidade entre segmentos e, consequentemente, uma maior carga para as saídas dos blocos lógicos;
- tendo em conta apenas a velocidade para o encaminhamento dos sinais, os melhores esquemas de segmentação (balanço óptimo entre segmentos de vários comprimentos) exigem um número extra de cinco a seis segmentos por canal; para os mesmos esquemas, se se tiver em conta o não desperdício de área, apenas um segmento extra por canal será necessário.

Em conclusão, as ferramentas de apoio ao encaminhamento devem centrar a sua atenção inicial no encaminhamento de sinais cujo atraso de propagação seja crítico. Deixando os restantes para serem posteriormente encaminhados através dos recursos ainda disponíveis, optimiza-se a velocidade da aplicação sem desperdiçar recursos.

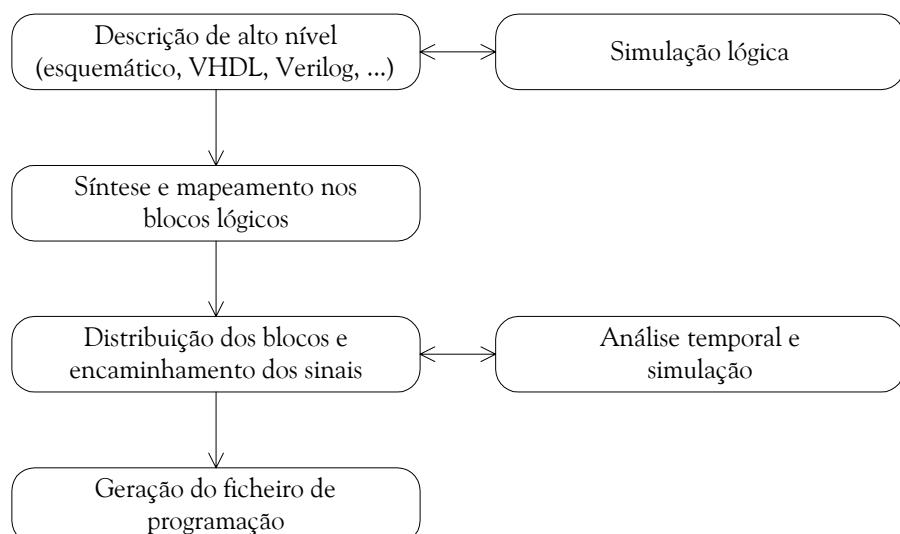
O número de comutadores a colocar nos blocos de encaminhamento locais e globais e a sua estrutura de ligações, o número de segmentos adjacentes a que as entradas e saídas de um bloco lógico podem ser interligadas e o número e comprimento dos segmentos a incluir em cada canal nas

arquitecturas simétricas foram alvo de estudos similares em [Betz et al., 99a] e em [Chow et al., 99]. O tamanho dos transístores de passagem, a sua corrente de fugas e os diferentes esquemas de implementação dos blocos de encaminhamento, são apresentados em [Betz et al., 99b]. Para as arquitecturas assimétricas, estudo idêntico é exposto em [Greene et al., 93].

Embora as características arquitectónicas das FPGAs, nas várias vertentes que foram expostas, sejam factor determinante na eficiência e desempenho dos circuitos nelas implementados, as ferramentas de apoio jogam igualmente um papel essencial na optimização desses parâmetros.

### 2.3. FERRAMENTAS DE APOIO AO PROJECTO

A implementação de um circuito nos actuais dispositivos lógicos programáveis envolve a configuração de entre algumas centenas de milhar a mais de 30 milhões de bits de configuração, tarefa só possível com recurso a sofisticadas ferramentas de apoio ao projecto. O circuito é descrito pelo projectista através de uma linguagem de descrição de alto nível, por exemplo, VHDL (*Very High Speed Integrated Circuit Hardware Description Language*) [IEEE 1076, 02] ou Verilog [IEEE 1364, 01], por meio de um esquemático ou, o mais comum, através de uma mistura de ambos. Posteriormente, as ferramentas de apoio ao projecto convertem, através de uma série de processos intermédios, essas descrições de alto nível em ficheiros de programação que especificam o estado de cada um dos bits de configuração. A Figura 2.24 esquematiza a sequencialização das várias fases do desenvolvimento de um projecto com uma FPGA. Cada uma destas fases será descrita em pormenor nas secções seguintes.



**Figura 2.24:** Fluxograma das ferramentas de apoio ao projecto com FPGAs

As ferramentas de apoio ao projecto podem ser fornecidas pelo fabricante das FPGAs especificamente para uso nos seus próprios dispositivos, ou ser independentes destes, situação com vantagem para o projectista, que apenas tem de decidir, numa fase já avançada do projecto, qual a FPGA que melhor se adequa aos seus objectivos. Essa liberdade só é possível nas fases iniciais de descrição, simulação e síntese do circuito, pois as etapas finais de implementação física e de validação são, obviamente, dependentes do dispositivo escolhido. Tal liberdade exige alguns sacrifícios como contrapartida, por exemplo a disponibilidade de um conjunto de ferramentas de implementação, o que implica um elevado tempo de aprendizagem. Por outro lado, a familiaridade do projectista com uma arquitectura permite a formação de uma bolsa de conhecimento que possibilita a obtenção de melhores resultados a nível de desempenho do produto final e obsta naturalmente à mudança [Maliniak, 95].

### 2.3.1 SIMULAÇÃO LÓGICA

A simulação lógica de um circuito, uma das etapas chave para o sucesso do resultado final, é um processo interativo que permite aferir se a operação do circuito descrito em alto nível vai ao encontro dos objectivos pretendidos ou se são necessárias modificações nessa descrição. A aferição é baseada na aplicação de estímulos às entradas do circuito e na observação das suas saídas verificando-se se este responde da forma esperada, cumprindo as especificações iniciais em termos de funcionalidade. De notar que a análise não tem em conta quaisquer considerações em termos temporais, uma vez que o suporte físico do circuito ainda não se encontra, nesta etapa, definido.

A crescente complexidade dos circuitos actuais e a inexistência de um modelo de validação torna esta fase numa das mais penosas para o projectista. A geração do conjunto de estímulos de teste a aplicar depende em grande parte da sua sensibilidade e experiência e, embora possam ser aplicadas várias métricas para validação dos resultados da simulação, nada garante o total cumprimento da especificação inicial. Alguns trabalhos têm sido desenvolvidos no sentido da criação de modelos de erros que permitam aliviar a complexidade e morosidade desta etapa e torná-la menos falível, nomeadamente criando modelos de erro para descrições em VHDL e ao nível de transferência entre registos (*Register Transfer Level*, RTL) [López et al., 00], embora com grandes limitações e sem grande sucesso prático. Os modelos limitam-se a cobrir erros de interpretação da linguagem e não propriamente a validar a descrição do circuito. Além disso, erros não cobertos pelo modelo não são detectados. Métodos de diagnóstico têm sido igualmente propostos com base numa pré-análise do circuito e no diagnóstico da zona em que ocorre o erro, seu posterior redesenho e verificação, mas sempre sem se apoiarem em qualquer modelo de erros específico [Ubar et al., 99].

A simulação lógica faz uma verificação funcional do circuito recorrendo a modelos para os seus componentes básicos, ou primitivas, tais como inversores e portas lógicas ‘E’ ou ‘OU’ de duas entradas, modeladas através da sua tabela de verdade ou usando as instruções que implementam as funções lógicas *Booleanas* do próprio processador no qual a ferramenta de apoio ao projecto está a correr. Em termos temporais, é considerado um atraso unitário, igual para todos os elementos. Dado que as primitivas são constituídas por portas lógicas de duas entradas, o seu número, mesmo em circuitos de baixa complexidade, tende a ser muito elevado, o que, associado a um extenso conjunto de estímulos de entrada e à natureza computacional do teste, torna o processo de simulação bastante moroso [Murgai et al., 99].

O tipo mais comum de simulador é o simulador guiado por eventos (*event-driven*). Quando o valor presente num nó do circuito é alterado, o instante em que essa alteração ocorre, o nó onde ocorre e o novo valor recebem colectivamente a designação de *evento*, que fica registado numa fila ou lista de eventos. À medida que decorre a simulação, os diferentes nós vão vendo os seus valores lógicos alterados. Como essa alteração afecta todas as primitivas que têm esse nó como entrada, todas elas têm de ser avaliadas, o que, por sua vez, acrescenta mais eventos à lista. O simulador mantém registo do instante de tempo, do espaçamento temporal e da lista de eventos.

Os simuladores de código interpretado (*interpreted-code simulators*) e os simuladores de código compilado (*compiled-code simulators*) usam a descrição em HDL como entrada. No primeiro, os dados são compilados num modelo executável pelo próprio simulador, enquanto, no segundo, a descrição é convertida numa linguagem intermédia (usualmente C) por um compilador separado do simulador, criando um modelo em código binário executável pelo processador. O tempo de compilação é, neste último caso, mais longo, mas o tempo de execução é mais curto, o que se torna uma vantagem quando a complexidade dos circuitos exige intervalos longos de simulação. Os simuladores de código nativo (*native-code simulator*) convertem directamente as descrições HDL num executável, apresentando os mais baixos tempos de execução.

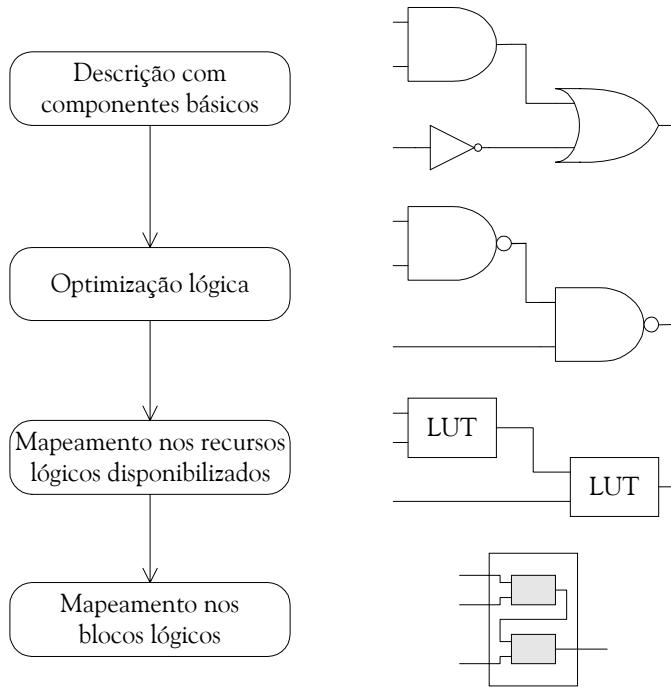
Uma vez que não é tida em conta qualquer informação temporal, o resultado da simulação apenas é fiável no caso de o projecto não alcançar os resultados esperados, mas não prova que o modelo produzido pelo projectista verifica integralmente a especificação.

### 2.3.2 SÍNTSE E MAPEAMENTO

As ferramentas de síntese realizam de forma automática o mapeamento da especificação de um circuito, em HDL ou esquemático, nos recursos lógicos disponíveis numa dada FPGA. A abordagem do problema da síntese divide-se habitualmente em duas partes: uma independente da

plataforma física de implementação e a outra dependente dela. Na primeira, é gerada uma representação abstracta optimizada do circuito lógico, enquanto, na segunda, o resultado é o mapeamento físico do circuito nos recursos lógicos da FPGA escolhida.

A especificação do circuito, em HDL ou esquemático, é convertida numa descrição de implementação empregando componentes básicos e na rede de interligações entre eles (*netlist*). O circuito obtido é posteriormente optimizado, eliminando-se toda a lógica redundante e, sempre que possível, simplificando-o. Após a optimização, inicia-se a fase de mapeamento do circuito nas primitivas disponibilizadas pela FPGA onde o circuito irá ser implementado. Posteriormente, são-lhe atribuídas posições físicas dentro dela, tendo-se então já em conta os recursos de encaminhamento disponibilizados. Na Figura 2.25, encontram-se ilustradas as várias fases deste processo.



**Figura 2.25:** Fluxo do processo de síntese

A distribuição da lógica pelas primitivas e a atribuição dos recursos físicos efectuam-se segundo critérios de minimização de área (número de blocos empregues) e/ou maximização da velocidade de funcionamento do circuito, dependendo da opção do projectista. Uma abordagem aprofundada dos métodos de síntese para FPGAs pode ser encontrada em [Sangiovanni-Vincentelli et al., 93]. Um sem número de algoritmos, alguns dos quais são apresentados em [Breuer et al., 00], foi desenvolvido no sentido de aperfeiçoar o desempenho das ferramentas de síntese, abrangendo as várias fases desse processo. Em [Krupnova et al., 00], é exposta uma visão global comparativa sobre as mais recentes famílias de FPGAs e as possibilidades oferecidas pelas ferramentas disponíveis de

apoio ao projecto, nomeadamente a sua capacidade de lidar e tirar partido das mais recentes inovações arquitectónicas. Alguns aspectos dessas ferramentas têm sido aperfeiçoados com o objectivo de conseguir melhores desempenhos, quer em termos genéricos [Brown et al., 96], quer em termos de uma maior eficiência na implementação de determinadas funções, como os registo de deslocamento com realimentação linear (*Linear-Feedback Shift Registers* – LFSRs) [Dufaza, 98], ou de operações aritméticas, como a multiplicação [Kaviani et al., 98]. Outro aspecto que tem merecido atenção é o da síntese para a testabilidade, visto que a própria ferramenta tem em conta, para além doutros parâmetros, aspectos como a observabilidade e controlabilidade dos nós internos dos circuitos aquando da implementação física [Ghosh et al., 98], permitindo eventualmente a incorporação automática de estruturas de teste [Burress et al., 97].

O trabalho da ferramenta de síntese e o sucesso da implementação final, em termos de maximização do desempenho do circuito e de minimização do seu custo, este último caracterizado fundamentalmente pela área ocupada, dependem igualmente do desenvolvimento de descrições bem estruturadas e que tenham em conta, quer as possibilidades da ferramenta, quer os recursos disponíveis na própria FPGA [Singh, 95]. Para cada função lógica existe uma forma optimizada de implementação, que deve ter a sua representação traduzida na linguagem de descrição de hardware usada [Smith, 97].

O desenvolvimento de sistemas multi-FPGA, que permitem a repartição de aplicações que requerem um espaço de implementação superior ao disponível num único componente por vários dispositivos, actuando como um só, alargaram o rol de desafios que se colocam às ferramentas de síntese. Um dos aspectos críticos nestes sistemas é o mapeamento automático dos circuitos, especialmente a partição do circuito em termos espaciais, tendo em conta a estrutura de interligação entre as várias FPGAs, sem perda de eficiência e deterioração do desempenho [Hauck et al., 95], [Duncan et al., 01], [Srinivasan et al., 01] e [Compton et al., 02]. A estrutura de interligação em sistemas multi-FPGA constitui aliás um problema *per si* que tem vindo a ser analisado por vários autores [Hauck, 98], [Brunfaut et al., 99] e [Campenhout et al., 99].

Mais recentemente, uma terceira dimensão foi adicionada a este problema. Depois da possibilidade de partição espacial entre várias FPGAs, surgiu, com o desenvolvimento de dispositivos de elevada capacidade reconfiguráveis parcial e dinamicamente, a possibilidade de partição do espaço de configuração da própria FPGA e de partição temporal. Várias aplicações partilham num dado instante o mesmo componente, acontecendo que algumas, quando a sua disponibilidade deixa de ser necessária, cedem o espaço que ocupam a outras cuja execução será entretanto requerida pelo sistema. A análise deste tipo de FPGAs e dos problemas levantados às ferramentas de apoio ao projecto serão alvo de estudo na última parte deste capítulo.

### 2.3.3 ANÁLISE TEMPORAL

A simulação lógica efectuada antes da implementação física do circuito permite aferir se este cumpre os requisitos fixados pelo projectista, mas, ao não incluir qualquer informação temporal, não garante que tal continue a ocorrer após a sua implementação sobre os recursos da FPGA. Somente nessa altura é possível analisar o verdadeiro desempenho do circuito e verificar se este responde às exigências iniciais do projecto, ou se é necessário introduzir alterações.

Este processo de síntese e análise temporal não é um procedimento estático mas interactivo, em que, após a primeira implementação, são avaliados os resultados de desempenho e eficiência do circuito, impondo-se, se conveniente, restrições em termos de atrasos máximos de propagação de alguns sinais, ocupação de determinados recursos, de área ocupada parcial ou global, ou outros, e procedendo-se então, de novo, à síntese de todo o projecto. Esta sequência poderá ser repetida tantas vezes quantas as necessárias até que o circuito corresponda aos requisitos impostos pelo sistema ou que todas as possibilidades estejam esgotadas, caso em que será forçoso reformular todo o problema e, eventualmente, optar por outras soluções. Nesta fase, a experiência do projectista é fundamental para guiar o processo de síntese ao encontro dos resultados pretendidos.

A simulação temporal permite a determinação da velocidade do circuito, depois de este ter sido completamente implementado na FPGA, e a estimação de quais os sinais com maior atraso de propagação, que devem ser encaminhados por recursos mais rápidos, de modo a evitar uma escusada lentidão do circuito. O atraso de propagação de um sinal tem origem no comportamento resistivo-capacitivo (RC) dos transístores de passagem e das linhas, modelado por intermédio de um modelo  $\pi$ -equivalente, e dos *buffers*, modelado por intermédio de uma constante de atraso e de uma resistência. A constante de atraso representa o atraso intrínseco do *buffer*, enquanto a resistência representa a dependência do atraso relativamente à carga. A Figura 2.26 ilustra o modelo RC para cada um dos três elementos que constituem os recursos de encaminhamento da FPGA [Cong et al., 96].

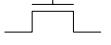
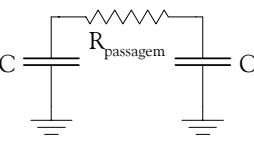
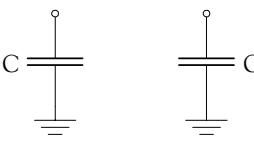
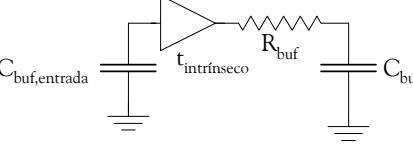
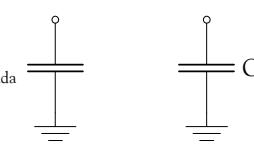
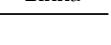
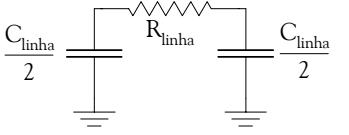
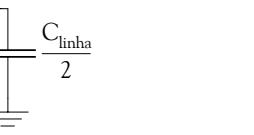
A análise temporal é efectuada directamente sobre um gráfico representando a estrutura do circuito, como aquele que se exemplifica na Figura 2.27. Cada nó do gráfico representa as entradas e saídas das primitivas do circuito, tais como tabelas de consulta e *flip-flops*, enquanto cada segmento une as entradas e saídas das primitivas, representando igualmente as interligações entre as primitivas especificadas na *netlist*. A seta indica o sentido de propagação do sinal. Cada segmento é anotado com o atraso introduzido no sinal, pela passagem através da primitiva ou da interligação a que diz respeito. O período mínimo do sinal de relógio é determinado pelo caminho que apresenta o atraso mais longo entre as entradas primárias do circuito ou as saídas dos *flip-flops*, e as saídas

primárias do circuito ou as entradas dos *flip-flops*. A determinação desse valor é efectuada a partir da determinação do tempo mais cedo de cada nó,  $t_{cedo}$ , ou seja, do instante a partir do qual todos os sinais que chegam a esse nó se encontram estáveis (aos nós correspondentes às entradas primárias e às saídas dos *flip-flops* é atribuído um  $t_{cedo}=0$ ). O cálculo do  $t_{cedo}$  de cada nó ‘i’ é efectuado percorrendo o gráfico no sentido da propagação dos sinais e de acordo com a Equação 2.2, onde o atraso ( $j,i$ ) corresponde ao valor marcado no segmento, unindo o nó anterior ‘j’ a esse nó ‘i’. Este procedimento repete-se até o parâmetro  $t_{cedo}$  ter sido calculado para todos os nós.

**Equação 2.2**

$$t_{cedo}(i) = \text{MÁX}_{\forall j \in \text{entradas}(i)} \{t_{cedo}(j) + \text{atraso}(j, i)\}$$

O nó que apresenta o valor maior de  $t_{cedo}$ , e que é sempre uma saída primária ou a entrada de um *flip-flop*, define o atraso máximo do circuito, o que equivale ao período mínimo do sinal de relógio aplicável a esse circuito. No caso exemplificado na Figura 2.27, o atraso máximo é de 12 ns no nó correspondente à saída primária, a que corresponde uma frequência máxima de funcionamento de 83,3MHz (=1/12ns).

	Se ligado	Se desligado
Transístor de passagem 		
Buffer com 3º estado 		
Linha 		

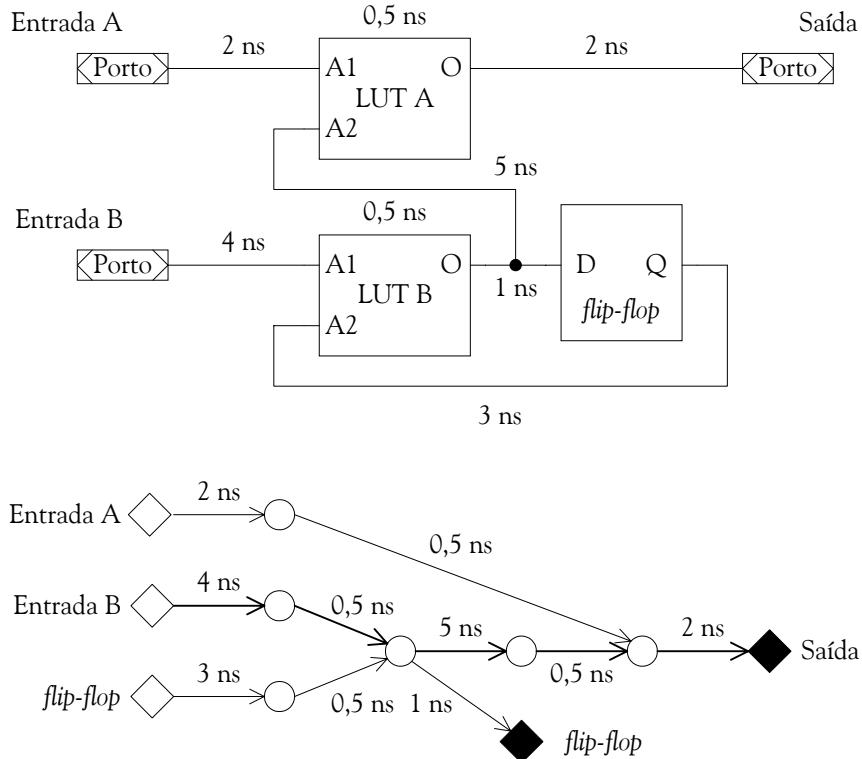
**Figura 2.26:** Circuitos equivalentes para os recursos de encaminhamento da FPGA

Se a distribuição da lógica pelas primitivas e a atribuição dos recursos físicos se efectuarem, por opção do projectista, segundo o critério de maximização da velocidade de funcionamento do circuito, torna-se necessário o estabelecimento de uma métrica que indique à ferramenta qual a importância de cada interligação, em termos do seu efeito no estabelecimento do atraso máximo do circuito, guiando-a na sua tarefa de determinar quais os melhores caminhos para o encaminhamento dos vários sinais. O tempo de folga de uma interligação define a quantidade de atraso que lhe pode ser adicionada sem aumentar o atraso máximo do circuito, supondo que todos os atrasos nas restantes interligações se mantêm constantes. Para o cálculo desse tempo de folga, é

necessário determinar o tempo mais tarde para cada nó,  $t_{\text{tarde}}$ , ou seja, o instante em que um sinal que chega a um nó tem obrigatoriamente de se encontrar estável, sob pena de aumentar o atraso máximo do circuito. A determinação de  $t_{\text{tarde}}$  para cada nó efectua-se partindo dos nós das saídas primárias ou das entradas dos *flip-flops*, aos quais é atribuído um valor  $t_{\text{tarde}}$  igual ao valor do atraso máximo do circuito, seguindo o sentido inverso ao da propagação dos sinais. A Equação 2.3 permite calcular o valor de  $t_{\text{tarde}}$  para cada nó.

**Equação 2.3**

$$t_{\text{tarde}}(i) = \text{Min}_{\forall j \in \text{saídas}(i)} \{ t_{\text{tarde}}(j) - \text{atraso}(i, j) \}$$



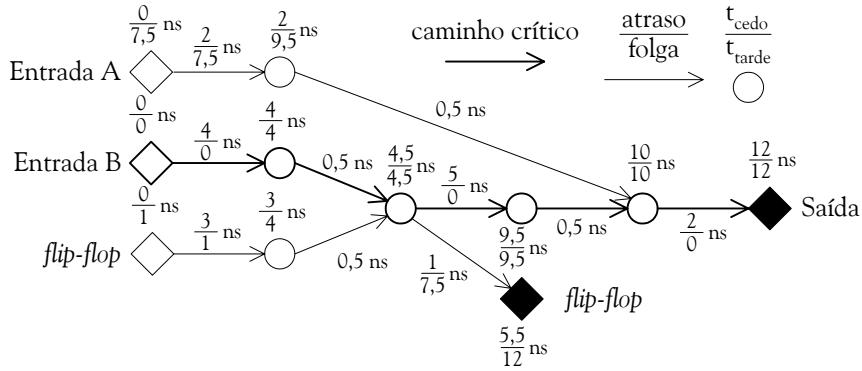
**Figura 2.27:** Exemplo de um gráfico temporal para um pequeno circuito (adapt. de [Betz et al., 99])

A folga da interligação entre o nó ‘i’ e o nó ‘j’ é, por sua vez, obtida através da Equação 2.4.

**Equação 2.4**

$$\text{folga}(i, j) = t_{\text{tarde}}(j) - t_{\text{cedo}}(i) - \text{atraso}(i, j)$$

As interligações com folga nula pertencem ao denominado *caminho crítico*, onde qualquer aumento no atraso conduz a um aumento correspondente no atraso máximo do circuito. Por outro lado, interligações com valores de folga elevados podem ser encaminhadas por caminhos mais lentos, sem que isso afecte a velocidade do circuito. A Figura 2.28 mostra os valores calculados dos tempos mais cedo e mais tarde, para cada nó, e as folgas nas interligações, para o circuito exemplificado na Figura 2.27.



**Figura 2.28:** Gráfico temporal completo com indicação do caminho crítico

As folgas são usadas pelas ferramentas de síntese para determinar os limites superiores dos atrasos nas interligações, de forma a garantir que a velocidade do circuito não decresce desde que cada interligação seja encaminhada com um atraso inferior ao seu limite máximo.

No caso de circuitos reconfiguráveis dinamicamente, as ferramentas de síntese são limitadas pelos recursos disponíveis aquando da configuração da aplicação no dispositivo. Esses recursos, variáveis no tempo, conduzem a valores diferentes para o atraso máximo de propagação, sendo eventualmente necessário proceder-se à prévia optimização dos recursos ocupados, de modo a garantir o valor mínimo de velocidade pretendido.

## 2.4. FPGAS COM RECONFIGURAÇÃO PARCIAL DINÂMICA

De acordo com a classificação das FPGAs em função das possibilidades de configuração, apresentada na Figura 2.9, existe um grupo cuja reconfiguração se pode efectuar de forma dinâmica e parcial. Este grupo de FPGAs, baseado em tecnologia SRAM, é passível de ser reconfigurado um número infinito de vezes, podendo essa reconfiguração incluir ou não a totalidade dos seus recursos, de forma dinâmica, sem perturbar a funcionalidade e a operação das funções configuradas em recursos não abrangidos pela reconfiguração.

O primeiro dispositivo a apresentar tais possibilidades foi o ERA60100 (*Electrically Reconfigurable Array*) da GEC Plessey Semiconductors, lançado em 1989 [Rose et al., 93]. Da mesma forma que as sub-rotinas de software num programa de computador são carregadas e executadas em resposta a um conjunto de eventos, tornava-se possível, com esta nova tecnologia, a programação de um circuito, em hardware, apenas quando este se tornasse necessário [Hastie et al., 90]. Nascia desta forma o conceito de *hardware virtual*, domínio no qual os recursos configuráveis são supostamente ilimitados e a implementação estática de um circuito que excede a área de reconfiguração é fraccionada em duas ou mais partes independentes, que podem ser trocadas dinamicamente, sendo

o escalonamento do recurso configurável efectuado por multiplexagem temporal [Cardoso et al., 99].

As aplicações que então podiam beneficiar destas novas possibilidades eram escassas, pelo que o dispositivo não teve o sucesso esperado, tendo desaparecido do mercado pouco tempo depois. A experiência seguinte surgiu em 1995, com o lançamento da série XC6200 pela Xilinx, que tinha como objectivo testar algumas inovações, mais tarde introduzidas numa nova família, e aferir do acolhimento do mercado [XCell, 95]. A aplicação-alvo desta nova série era a computação reconfigurável, em que FPGAs com capacidade de programação no circuito eram usadas como co-processadores reconfiguráveis, combinando a versatilidade de uma solução programável com o desempenho do hardware dedicado. O programa principal era executado por um processador genérico, com a atribuição de certas tarefas de computação intensiva ao co-processador, para acelerar a sua execução. Esta abordagem explora o facto de a maioria do tempo de processamento ser gasto com tarefas deste tipo, que correspondem a pequenas porções de código. Por isso, a sua implementação em hardware, ao acelerar a sua execução, contribui para o aumento do desempenho total do sistema. A atribuição dos recursos a cada tarefa é efectuada conforme as necessidades de processamento, sendo as funções trocadas quando necessário pela sequência de processamento.

Os problemas das arquitecturas reconfiguráveis de uso genérico, utilizadas na altura para este fim, eram principalmente dois: uma parte significativa dos recursos da própria FPGA era consumida na implementação da interface com o barramento do processador; e a reconfiguração de novas tarefas era um processo lento, quando comparado com a velocidade de processamento. A arquitectura das XC6200 foi idealizada no sentido de obstar a estes problemas, possuindo uma interface dedicada, projectada para interligar a FPGA directamente ao barramento do processador principal, uma granulosidade fina, recursos de encaminhamento abundantes e possibilidade de reconfiguração dinâmica total ou parcial. A interface permitia que as células de configuração e os registo dos blocos lógicos aparecessem como memória convencional mapeada no espaço de endereçamento do processador principal. Desta forma, o acesso aos registo e a alteração da funcionalidade eram possíveis de uma forma extremamente rápida, com a vantagem adicional de esta alteração poder ser efectuada apenas de forma parcial, sem afectar a operação da parte restante, o que contribuía ainda mais para a diminuição dos tempos de reconfiguração [Xilinx, 95]. No entanto, esta família seria descontinuada cerca de quatro anos depois, tendo sobretudo sido utilizada no meio académico como suporte à investigação de novas aplicações, onde o recurso à reconfiguração parcial dinâmica podia ser usado com vantagem.

No início de 1998, a Xilinx lançou a nova família Virtex, que incorpora algumas das características presentes na XC6200, entre as quais se destaca a possibilidade de reconfiguração parcial dinâmica,

embora funcione em moldes diferentes daquela [Bursky, 97] e [XCell, 98]. Desaparece a possibilidade de endereçamento individual das células de configuração e dos registos dos blocos lógicos, que é substituída pelo acesso a um grupo onde se misturam células de configuração de diferentes recursos configuráveis com os registos dos blocos lógicos. Esta família foi posteriormente alargada incorporando as séries Virtex-E e Virtex-II, que retendo as mesmas possibilidades que caracterizavam a série inicial, incorporam interfaces mais rápidas, maior capacidade de memória interna, um maior número de recursos lógicos e, sobretudo, tensões de funcionamento mais baixas. A última série a ser lançada, a série Virtex-II Pro, cuja comercialização se iniciou no começo de 2002, incorpora, dependendo do dispositivo, até quatro microprocessadores PowerPC 405, desenvolvidos pela IBM, com velocidades de funcionamento superiores a 300MHz, totalmente integrados na lógica reconfigurável [Xilinx, 02]. Esta série é compatível com a norma IEEE 1532 [IEEE 1532, 01], que descreve uma série de instruções de programação e registos de dados associados, definidores de uma metodologia para o acesso e configuração de dispositivos programáveis, como extensão das capacidades dos dispositivos conformes à norma IEEE 1149.1. De notar que as séries anteriores já permitiam a configuração através da interface de acesso à infra-estrutura de teste, mas, dado terem sido lançadas anteriormente à publicação da norma IEEE 1532, não apresentam compatibilidade com esta.

Paralelamente, a Xilinx comercializa, desde do início do ano 2000, a família Spartan-II. Com uma arquitectura idêntica à das Virtex, suporta igualmente reconfiguração parcial dinâmica, mostrando capacidades que, nos dispositivos maiores, se sobrepõem às das Virtex mais pequenas, mas com um custo bastante inferior, o que lhe permite, em grandes volumes de produção, ser considerada como uma alternativa aos ASICs [XCell, 00] e [Xilinx, 02a].

Outros fabricantes que oferecem actualmente a possibilidade de reconfiguração parcial dinâmica nas suas FPGAs são a Atmel, com as famílias AT6000 e AT40K [Atmel, 99], e a Lattice, com a família ORCA [Lattice, 02].

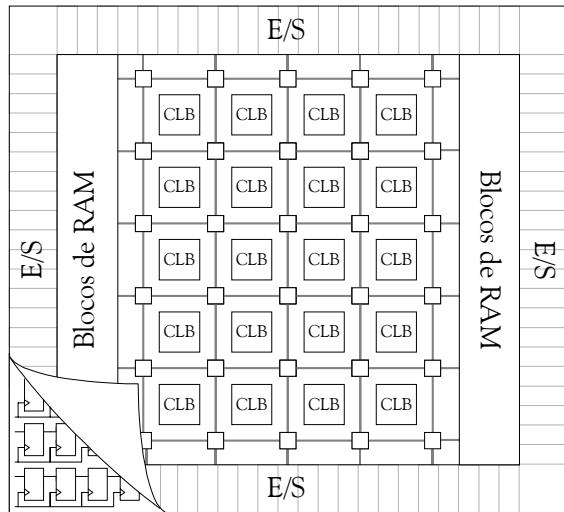
#### **2.4.1 A ARQUITECTURA DA FAMÍLIA VIRTEX**

A metodologia de teste para dispositivos lógicos programáveis com possibilidade de reconfiguração parcial dinâmica, desenvolvida nesta tese, teve como alvo e plataforma de validação os dispositivos da família Virtex, pelo que se justifica um estudo mais aprofundado da sua arquitectura, de forma a compreender todos os aspectos relacionados com a implementação prática da metodologia.

As FPGAs da família Virtex são baseadas numa arquitectura simétrica, regular, que, para efeitos de uma análise mais profunda, pode ser dividida em várias partes:

- uma matriz de blocos lógicos configuráveis, designados por CLBs;
- um conjunto de blocos de E/S que serve de interface entre os CLBs e os pinos do componente;
- blocos de memória RAM dedicados de 4096 bits cada;
- um conjunto de recursos de interligação local e outro de recursos de interligação global;
- uma memória estática de configuração.

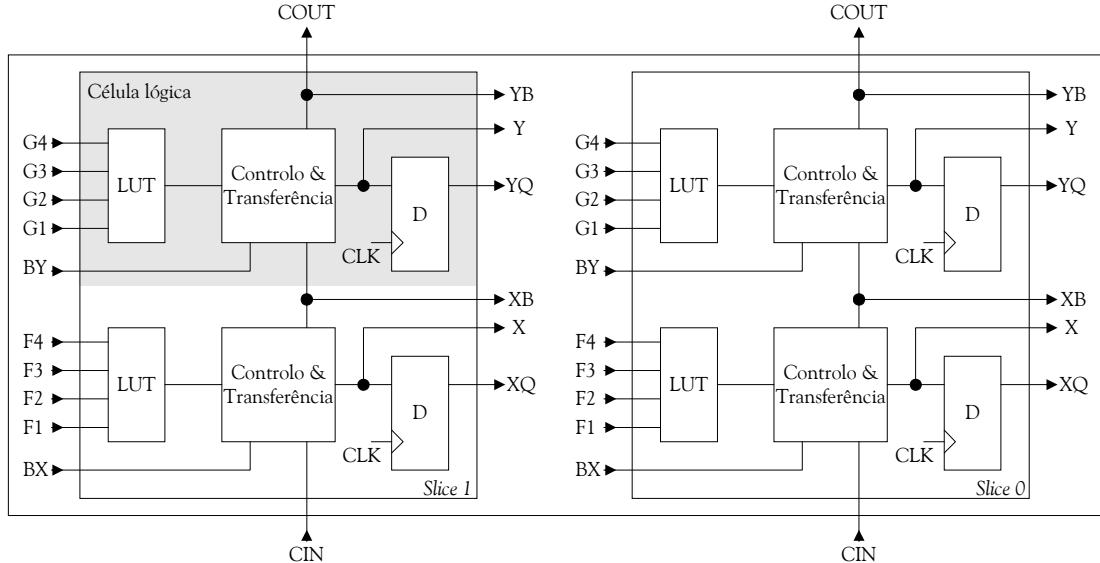
Para além destes blocos principais, a FPGA possui ainda recursos dedicados para controlo e distribuição de sinais de relógio, bem como uma infra-estrutura de teste compatível com a norma IEEE 1149.1 [IEEE 1149.1, 01]. A Figura 2.29 apresenta uma representação esquemática da arquitectura das Virtex.



**Figura 2.29:** Representação esquemática da arquitectura das Virtex

A constituição dos blocos lógicos configuráveis, CLBs, tem por base as células lógicas, que incluem, cada uma, um gerador de funções de 4 entradas, implementado sob a forma de uma tabela de consulta, lógica de controlo e transporte, e um elemento de retenção. A saída do gerador de funções de cada célula lógica opera uma saída do CLB, directamente ou através do elemento de retenção, de forma independente. Cada CLB, cuja organização interna se encontra esquematizada na Figura 2.30, contém quatro destas células lógicas, organizadas em duas partes similares, designadas *slice*. Os vários geradores de funções de um CLB podem ser combinados de forma a proporcionar funções de cinco ou seis entradas.

A tabela de consulta é um elemento bastante versátil, sendo possível o seu uso não só como gerador de funções, mas igualmente como memória ROM ou memória RAM síncrona local (LUT RAM), ou como registo de deslocamento de 16 bits.



**Figura 2.30:** Arquitectura de um CLB

Os elementos de retenção são configuráveis, quer como *flip-flops* do tipo D, quer como *latches*, podendo a sua entrada ser operada a partir da saída do gerador de funções ou directamente a partir da entrada do CLB.

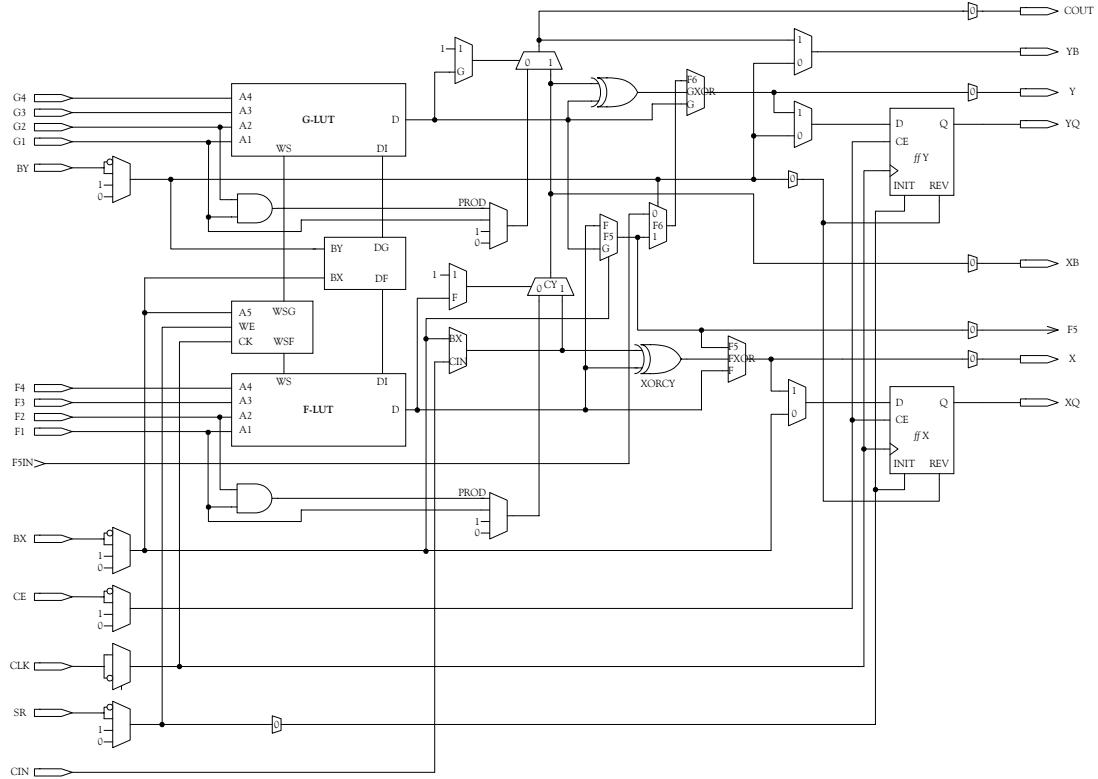
Cada *slice*, para além dos sinais de relógio (CLK) e habilitação de relógio (CE), que figuram no seu esquema detalhado apresentado na Figura 2.31, possui sinais para Set e Reset síncronos ou Preset e Clear assíncronos, dependendo da configuração (SR e BY). SR força o elemento de retenção ao estado de inicialização especificado no ficheiro de configuração (entrada INIT do elemento de retenção), enquanto BY o força ao estado inverso (entrada REV do elemento de retenção).

A existência de lógica aritmética dedicada proporciona os recursos necessários para a implementação de funções aritméticas com elevadas velocidades de execução. Cada CLB suporta duas cadeias de transporte, uma por *slice* (CIN->COUT), que se podem estender por várias *slices*, permitindo a implementação de funções aritméticas de elevada complexidade e de funções lógicas com grande número de variáveis.

Os blocos de E/S são constituídos basicamente por três elementos distintos: elementos de retenção configuráveis, multiplexadores e *buffers*. A estrutura interna de cada bloco está representada na Figura 2.32, onde se inclui igualmente toda a lógica associada à implementação das respectivas células do registo de varrimento pela periferia (BS), definidas pela norma IEEE 1149.1 [IEEE 1149.1, 01]. A família Virtex é totalmente conforme a esta norma, pelo que a sua arquitectura inclui os seguintes elementos:

- TAP (Test Access Port);

- controlador do TAP;
- registo de instrução;
- registo de varrimento pela periferia;
- registo de bypass.



**Figura 2.31:** Esquema detalhado de uma slice

As Virtex suportam ainda alguns elementos adicionais, que a norma IEEE 1149.1 indica como opcionais:

- um registo de identificação com 32 bits;
- um registo de identificação programável pelo utilizador;
- um registo de configuração.

Ao utilizador, é permitida a configuração de dois registos por si definidos, que podem ser acedidos, depois de criados, através dos pinos do TAP. A Figura 2.33 ilustra a implementação interna dos registos presentes na infra-estrutura de teste das Virtex. Alguns dos blocos de E/S são enterrados, o que significa que não têm pino de ligação ao exterior, funcionando unicamente com realimentação interna, embora todos eles sejam percorridos pela cadeia de varrimento pela periferia.

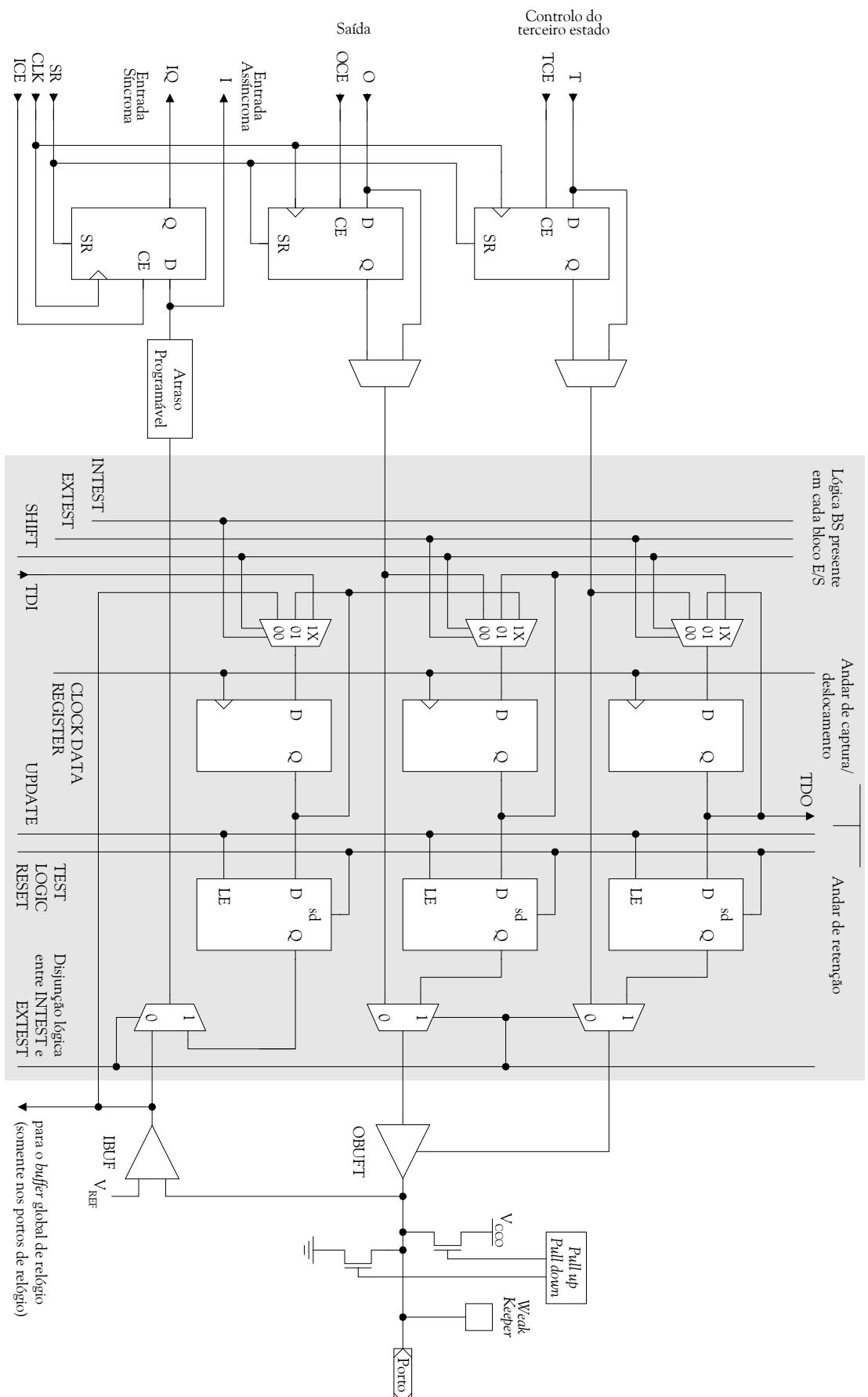
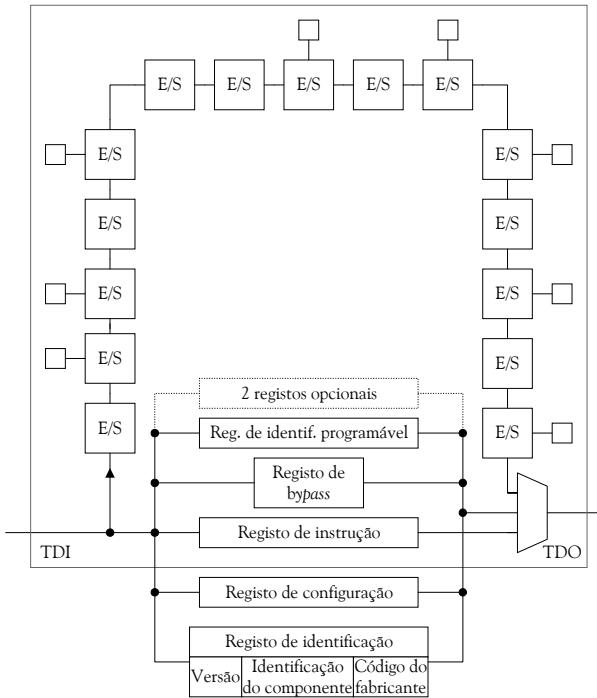


Figura 2.32: Constituição interna de um bloco de E/S



**Figura 2.33:** Arquitectura de registos da infra-estrutura de teste

O TAP é, nas Virtex, constituído por quatro pinos dedicados, três de entrada e um de saída, que controlam o funcionamento desta infra-estrutura de teste, que é gerida por uma máquina de estados cujo diagrama de transição está ilustrado na Figura 2.34. Os quatro pinos dedicados são:

- TMS – Test Mode Select A sequência de estados percorrida pelo controlador do TAP é determinada pelo estado do pino TMS quando ocorre a transição ascendente do relógio de teste (Test Clock – TCK). O pino TMS possui uma resistência de *pull-up* interna, que assegura um valor lógico alto, no caso de o pino não ser usado.
- TCK – Test Clock Relógio de teste, sequencia o controlador do TAP e os registos associados a esta infra-estrutura de teste. O valor máximo da frequência do relógio de teste na família Virtex é 33 MHz, independentemente da velocidade dos circuitos implementados no componente.
- TDI – Test Data In Entrada série de todas as instruções e dados para a infra-estrutura de teste. O estado do controlador do TAP e a instrução guardada no registo de instrução, que indica a operação a executar, determinam o registo que será atravessado pela sequência de bits injectada neste pino. O deslocamento dos bits ocorre quando se verifica a transição ascendente do relógio de teste (TCK). O pino TDI possui uma resistência de *pull-up* interna, que assegura um valor lógico alto, no caso de o pino não ser usado.

- TDO – Test Data Out Saída série de todos os registos internos, instrução e dados. O estado do controlador do TAP e a instrução guardada no registo de instrução, que indica a operação a executar, determinam qual o registo que alimenta esta saída. O valor de TDO é alterado na transição descendente do relógio de teste, encontrando-se activo somente durante o deslocamento de instruções ou de dados através do componente e em alta impedância durante o resto do tempo.

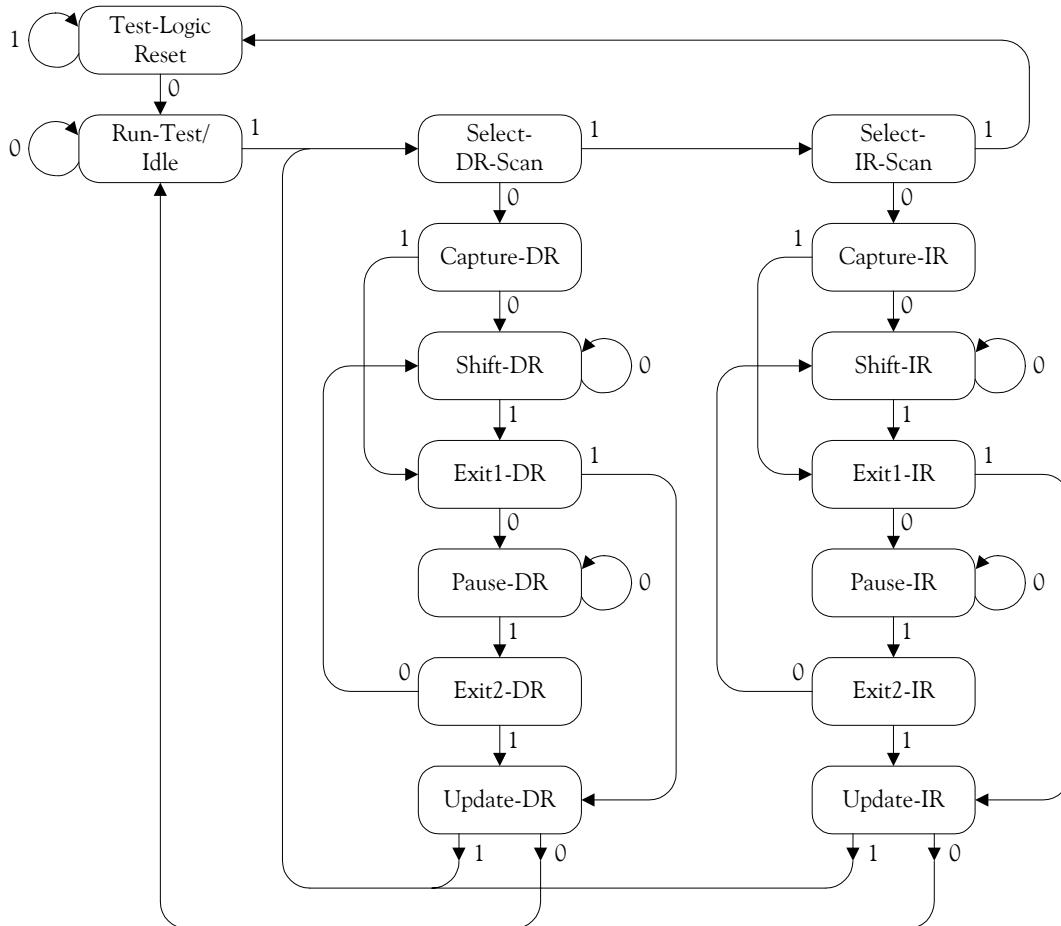


Figura 2.34: Diagrama de estados do controlador do TAP<sup>8</sup>

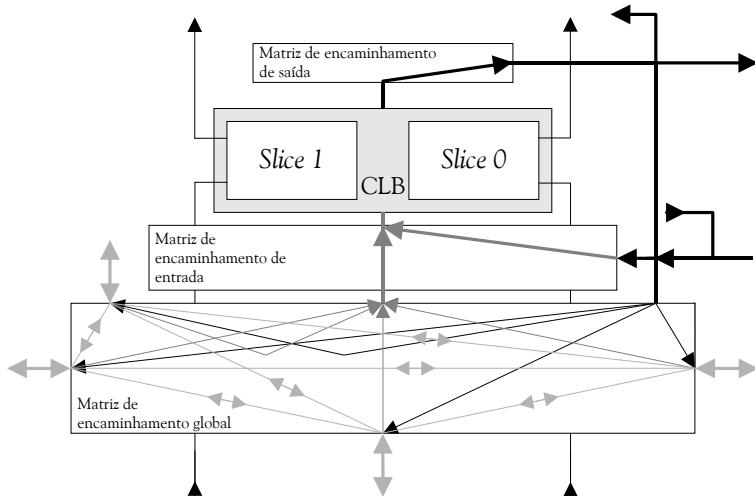
Tendo em conta a extensa bibliografia existente sobre a norma IEEE 1149.1, não se aprofundará mais o seu estudo, remetendo-se antes o leitor para algumas dessas obras e para o texto da norma [Maunder et al., 90], [Maunder et al., 91], [Parker, 1992], [Bleeker, 1993], [Wondolowski et al., 99], [Bushnell et al., 00] e [IEEE 1149.1, 01]. Em anexo a esta tese, encontra-se igualmente uma descrição mais aprofundada da arquitectura das Virtex, na qual o funcionamento da infra-estrutura de teste é objecto de um estudo mais pormenorizado.

---

<sup>8</sup> O valor adjacente a cada transição de estado representa o sinal presente em TMS aquando da transição ascendente de TCK.

Cada Virtex apresenta vários blocos de memória do tipo RAM, complementares da RAM distribuída passível de ser implementada em cada CLB. Estes blocos estão organizados em colunas, contendo cada Virtex duas dessas colunas, uma de cada lado da matriz de blocos lógicos. Cada bloco de memória tem uma altura de 4 CLBs, o que implica que, num componente com 64 CLBs de altura, existam 16 blocos de memória por coluna, num total de 32 blocos. Cada bloco de memória de dupla porta, totalmente síncrona, tem um tamanho de 4096 bits, com sinais de controlo independentes para cada porta. O comprimento da palavra de dados das duas portas é configurável independentemente, proporcionando conversão interna entre barramentos de dados de diferentes comprimentos.

Um conjunto de recursos de interligação local proporciona o encaminhamento dos sinais entre as saídas do CLB e as matrizes de encaminhamento global, linhas internas para realimentação dos CLBs (permitindo encadear sem atraso de propagação tabelas de consulta de um mesmo CLB) e linhas directas, que proporcionam ligações com elevada velocidade de propagação entre CLBs adjacentes horizontalmente, eliminando o atraso introduzido pela matriz de encaminhamento global. A interligação das entradas e saídas do CLB aos recursos de encaminhamento global é executada através de matrizes de encaminhamento local. A Figura 2.35 mostra uma representação esquemática dos recursos de encaminhamento associados a cada CLB, com indicação da sua direccionalidade.



**Figura 2.35:** Recursos de encaminhamento associados a cada bloco lógico

Como grande parte dos sinais dentro da FPGA é encaminhada através de linhas genéricas, a maioria dos recursos de interligação está associada a essas linhas, localizadas em canais horizontais e verticais, que se distribuem paralelamente às linhas e colunas da matriz de CLBs. As Virtex possuem igualmente um conjunto de recursos de interligação periféricos dispostos entre a matriz de

CLBs e os blocos de E/S, o que facilita a troca de pinos e a manutenção da atribuição de sinais aos pinos aquando de alterações na lógica interna, e evita a necessidade de redesenho da carta de circuito impresso. Adicionalmente, possuem um conjunto de recursos de interligação dedicados a várias funções específicas, como sejam a distribuição global e local de sinais de relógio, duas linhas dedicadas em cada CLB para a propagação de sinais de transporte verticalmente entre CLBs adjacentes e ainda recursos de encaminhamento horizontais, que permitem a implementação de barramentos com terceiro estado.

Embora a memória de configuração das Virtex, do tipo SRAM, permita um número infinito de reconfigurações do componente, a sua volatilidade obriga à existência de uma memória ou processador externo dedicado a essa tarefa. A memória de configuração pode ser visualizada como uma matriz rectangular de bits agrupados em vectores verticais com um bit de largura, que se estendem do topo ao fundo da matriz. O vector é a unidade atómica de configuração, a mais pequena porção da memória de configuração endereçável individualmente que pode ser escrita e/ou lida. Os vectores agrupam-se paralelamente em unidades maiores, designadas colunas de configuração e constituídas por um número de vectores variável consoante o seu tipo. Na Figura 2.36, é possível visualizar a organização da memória de configuração, salientando-se a coluna central, que inclui a configuração dos quatro pinos de relógio global (GCLK), rodeada pelo conjunto formado pelas colunas de configuração de cada uma das colunas de CLBs, e respectivos blocos de E/S (dois no topo e dois em baixo). Nos extremos, de ambos os lados, mais três colunas de configuração, duas referentes à configuração dos blocos de RAM (uma para a configuração das interligações e a outra relativa ao conteúdo de cada célula de memória) e uma, à configuração das colunas laterais de blocos de E/S.

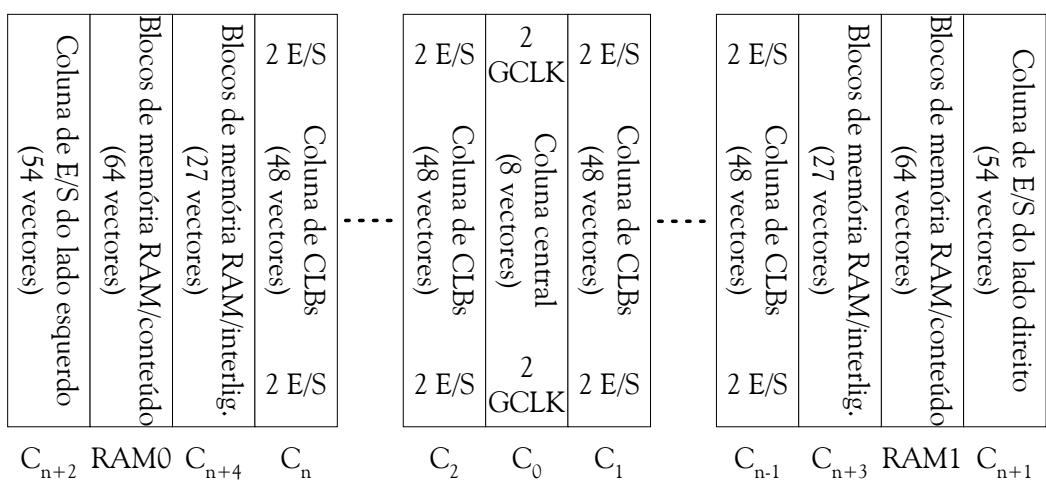


Figura 2.36: Organização do módulo de configuração.

A divisão da totalidade da memória de configuração da FPGA em vectores endereçáveis individualmente permite a sua configuração parcial dinâmica. O processo de configuração é um mecanismo sequencial que varre algumas, ou eventualmente a totalidade, das colunas de configuração.

As Virtex suportam quatro modos de configuração diferentes, três série e um paralelo. O modo de configuração usado na implementação das metodologias propostas neste trabalho é o *Boundary-Scan*, escolhido pela sua relação com o teste e por, recentemente, o uso desta interface como porto de acesso para a configuração ter sido normalizado pela norma IEEE 1532, com a vantagem de não requerer a ocupação de pinos suplementares. Este modo sobreponde-se a todos os outros e pode ser usado mesmo que outro modo esteja seleccionado através dos pinos de selecção de modo. Uma vez que toda a configuração é efectuada usando apenas os pinos do TAP, este modo não requer a existência de pinos dedicados para esse fim. A configuração através do TAP usa a instrução CFG\_IN, que converte os dados entrados por TDI em pacotes para o barramento interno de configuração.

Para um estudo aprofundado da arquitectura da família Virtex, remete-se o leitor para a descrição mais pormenorizada em anexo a esta tese e para a literatura fornecida pelo fabricante [Xilinx, 02a].

#### **2.4.2 VALOR ACRESCENTADO PELA RECONFIGURAÇÃO PARCIAL DINÂMICA**

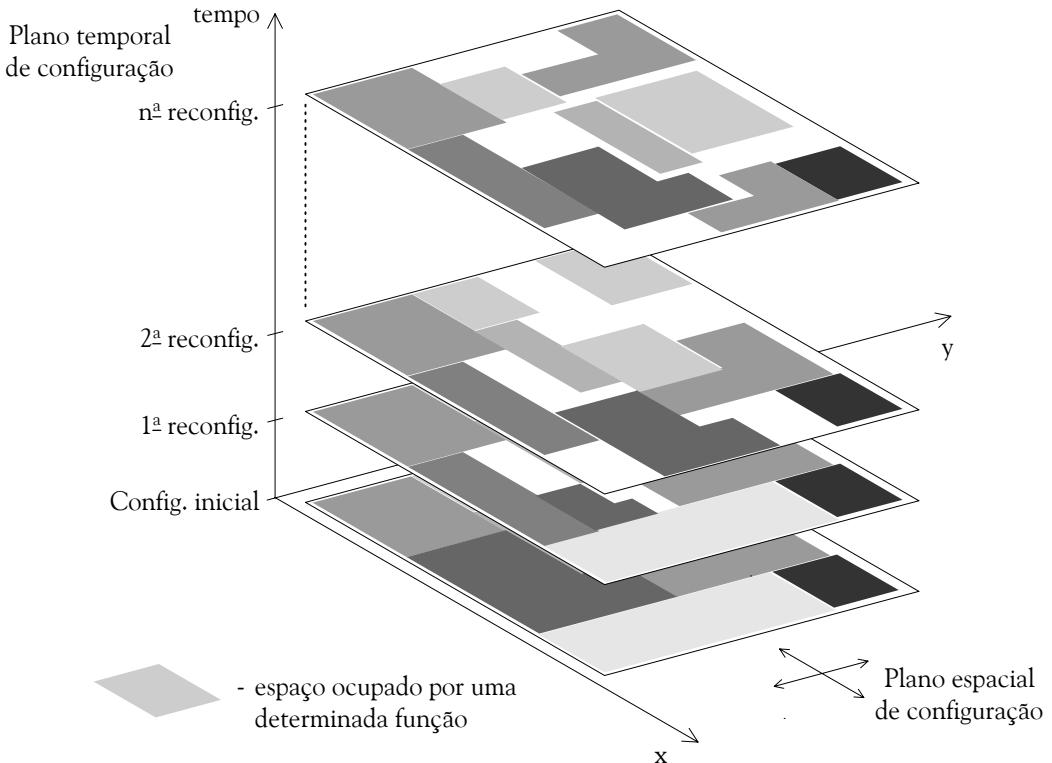
Uma das interrogações surgidas com o aparecimento das FPGAs com reconfiguração parcial dinâmica foi saber que aplicações podiam beneficiar do seu uso. Durante algum tempo, pensou-se que o avanço tecnológico estava um passo à frente da necessidade, o que terá levado a que as ERA60100, lançadas em 1989, desaparecessem do mercado pouco tempo depois, delas restando apenas algumas referências na literatura da época. Nova tentativa surgiu com o aparecimento da família XC6200, que, mercê da sua revolucionária arquitectura, encontrou um forte acolhimento nos meios académicos. No entanto, o aviso difundido pelo fabricante de que esta família se destinava apenas a experimentar novas possibilidades arquitectónicas e de que a mesma seria descontinuada no futuro, o que de facto veio a acontecer, obstou a que o seu uso se estendesse também à indústria. Apesar disso, e por via desse forte acolhimento académico, um grande esforço de investigação foi desenvolvido, tendo-se estudado que novas áreas e aplicações poderiam tirar vantagem do uso da reconfiguração parcial dinâmica. Uma das áreas fortemente beneficiadas foi a da computação reconfigurável [Compton et al., 02]. As arquitecturas reconfiguráveis permitem ao projectista criar novas funções e executar em hardware dedicado, dinamicamente configurado numa FPGA, operações que num processador de uso genérico levariam muitos ciclos de relógio a completar [Sima et al., 02]. Dado que apenas as unidades funcionais necessárias para executar uma

determinada função estão presentes no hardware, ao contrário do que se passa num processador genérico, consegue-se alcançar níveis inigualáveis de eficiência em termos de consumo e de espaço ocupado [Adário et al., 99]. Os princípios preliminares desta nova área, que se mostraram muito avançados em relação à tecnologia existente, foram enunciados no início dos anos 60 por Gerald Estrin, que propôs a concepção de um sistema “variável” composto por três elementos: um computador de uso genérico, uma plataforma de hardware “variável”, que permitiria a implementação de uma função digital arbitrária, e uma unidade de controlo e supervisão [Estrin et al., 63].

A computação reconfigurável oferece igualmente outras vantagens, como a rápida adaptação a novas exigências do mercado, a redução do tempo de colocação no mercado de um novo produto e a possibilidade de alteração das suas características, mesmo em fases avançadas do desenvolvimento, sem uma excessiva penalização dos custos. Torna-se também possível a introdução de alterações e correcções, ou a satisfação de novos pedidos do cliente, mesmo após a instalação do sistema (neste caso de forma remota), reduzindo os custos de manutenção e substituição. Algumas das áreas que podem beneficiar desta nova abordagem, e que se caracterizam por serem fortemente baseadas em algoritmos de computação intensiva, são a encriptação e a desencriptação, a compressão de dados, a comparação de sequências, a ordenação de dados, a análise e simulação de sistemas físicos, o processamento de imagem e vídeo e o processamento digital de sinal. A elevada eficiência em termos de consumo, bem como a adaptabilidade às mudanças requeridas pelo sistema, torna-as, por sua vez, particularmente vocacionadas para uso em sistemas móveis [David et al., 02], [Lauwereins, 02] e [Smit et al., 02]. A sua utilização é ainda referida na implementação de algoritmos de processamento de imagem [Burns et al., 97], [Woods et al., 98], [Rizzo et al., 02] e [Sassatelli et al., 02], de algoritmos de descodificação [Haase et al., 02], de circuitos de controlo definidos com base em descrições hierárquicas conduzindo a uma estrutura modular [Lau et al., 99], de redes neurais [Porrmann et al., 02], na prototipagem rápida de circuitos [Vasilko et al., 98] e, mais recente, na implementação de hardware auto-reconfigurável [Köster et al., 02] e no desenvolvimento de hardware evolucionário. Este último constitui um campo onde se procura transportar para o mundo da electrónica os princípios da diferenciação celular e evolução embrionária que se verificam nos seres vivos, [Higuchi et al., 99], [Levi et al., 99], [Sipper et al., 99], [Thompson et al., 99], [Yao, 99] e [Sánchez, 01].

A reconfiguração parcial dinâmica, aliada à elevada capacidade dos dispositivos actuais e a uma redução do tempo de configuração, permitiu avançar no sentido da flexibilização e incremento do uso da computação reconfigurável, ao possibilitar que várias funções concorrentes ocupem um mesmo espaço de configuração e que a alteração da funcionalidade de uma ou mais delas possa ser

efectuada sem que isso afecte o funcionamento das restantes. O espaço de configuração pode agora ser visualizado como um espaço tridimensional, contendo duas dimensões espaciais, que albergam o plano de configuração correspondente aos recursos físicos da FPGA e onde diferentes funções se encontram configuradas num dado instante, e uma dimensão temporal, correspondente ao escalonamento das funções a configurar (funções essas que ocupam recursos deixados livres por funções cujo intervalo de execução já terminou), conforme se ilustra na Figura 2.37.

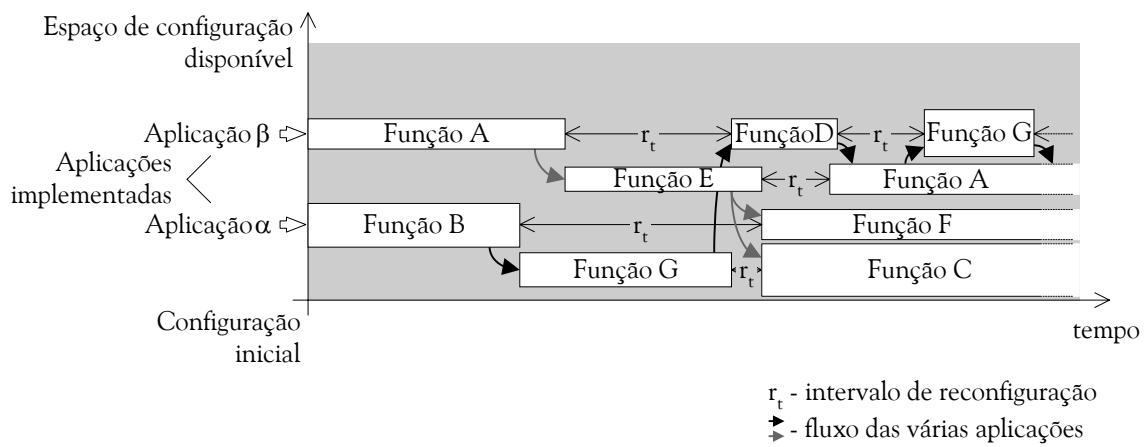


**Figura 2.37:** Visão tridimensional do espaço de configuração

Sendo as aplicações normalmente constituídas por conjuntos de funções que, com frequência, são executadas sequencialmente, ou com um baixo grau de paralelismo, a sua disponibilidade simultânea não é necessária. Por outro lado, os muito baixos tempos de configuração possibilitam a troca de funções em tempo real, isto é, sem que a execução da aplicação seja retardada. Se se conseguir um escalonamento apropriado das funções, torna-se possível que, com base na reconfiguração parcial dinâmica, um único dispositivo execute um conjunto de aplicações que, no seu conjunto, requerem mais do que a totalidade dos recursos disponíveis, por troca das várias funções conforme necessário.

O tempo de reconfiguração parcial de uma função situa-se na ordem dos microsegundos, dependendo da interface de configuração e da complexidade (e, como tal, do tamanho) da função a ser implementada. Esse tempo pode, no entanto, ser virtualmente reduzido a zero se as novas

funções forem implementadas antes de terminada a execução da função que a precede no fluxo da aplicação. Esta possibilidade será viável se houver espaço disponível suficiente, quer livre, quer ocupado por funções cujo tempo de execução já terminou e que, por conseguinte, estão disponíveis para serem substituídas, conforme se ilustra na Figura 2.38. Note-se que o *intervalo de reconfiguração*,  $r_t$ , se refere ao intervalo de tempo durante o qual a reconfiguração de uma nova função pode ser efectuada de forma a estar operacional quando requerido pelo fluxo da aplicação e não deve, por isso, ser confundido com o *tempo de reconfiguração*. Um aumento do grau de paralelismo das funções pode retardar a configuração de novas funções, devido a falta de espaço na FPGA, pelo que podem ser introduzidos atrasos na execução das aplicações, de forma sistemática ou não, dependendo do seu fluxo. Um aumento do espaço de configuração disponível pode obstar a esta situação, à custa de uma perda de eficiência na ocupação do espaço [Gericota et al., 02].



**Figura 2.38:** Escalonamento temporal das funções no espaço de configuração

A implementação desta filosofia encontra obstáculo na falta de ferramentas de apoio ao projecto que a suportem. As ferramentas de síntese deveriam ser capazes de, na eventualidade de os recursos requeridos pelo circuito ou circuitos a implementar excederem os recursos configuráveis disponíveis, fraccionar o circuito e escalar espacial e temporalmente o uso do espaço configurável, de forma eficiente, sem afectar a execução das aplicações executadas pelo sistema, e gerar os ficheiros de configuração parciais [Zhang et al., 00]. Alguns trabalhos pioneiros nesta área abordaram a questão da partição e do escalonamento automático, embora sem alcançarem resultados completamente satisfatórios [Vasilko et al., 95], [Vasilko et al., 96], [Gibson et al., 98] e [Vasilko, 99], pelo que surgiram propostas alternativas que possibilitavam ao projectista um maior controlo e intervenção sobre todo o processo de síntese [Gehring et al., 98], ou que propunham simplesmente a troca de contextos, entendida como a reconfiguração da totalidade da FPGA. Neste caso, para evitar a paragem do sistema durante a reconfiguração, tenta-se minimizar o número de configurações que se sobreponem à utilização do próprio hardware dedicado, de tal forma

que o intervalo de configuração ocorra quando a sequência de processamento não requerer a sua existência para prosseguir [Cantó et al., 01] e [Sanchez-Elez et al., 02].

Deve ainda referir-se que a definição de reconfiguração dinâmica não é pacífica visto que, ao contrário de alguns autores que entendem que a troca de contextos se insere nesta definição, outros defendem que ela apenas faz sentido se ligada à reconfiguração parcial.

Outro problema que se coloca é a questão da simulação funcional, uma vez que a lógica implementada na FPGA se altera com o tempo. A operação de cada combinação diferente de circuitos, que possa estar simultaneamente activa no dispositivo, deve ser verificada funcionalmente, assim como as transições dinâmicas entre as diferentes combinações. Após o sucesso de cada verificação, deve ser criada uma implantação única para cada combinação e efectuada a respectiva análise temporal, devendo os resultados ser usados para a reverificação funcional de cada uma dessas combinações [Robinson et al., 00] e [Robertson et al., 02].

A geração dos ficheiros de reconfiguração parcial é outro problema que se levanta à implantação da reconfiguração parcial dinâmica, dado que as ferramentas de apoio actualmente disponibilizadas pelos fabricantes não estão preparadas para a suportar. Por isso, foi necessário desenvolver aplicações que, a partir de ficheiros de configuração completos, por comparação, geram ficheiros diferença [Horta et al., 02].

Trabalhos mais recentes apresentam já ambientes de trabalho mais consistentes, em que todas as diferentes fases do projecto se encontram integradas num mesmo ambiente de trabalho [Vasilko et al., 99], [Vasilko, 00] e [Vasilko et al., 00].

Uma outra questão que se põe é a partição entre software e hardware. Se, no passado, com a utilização dos microprocessadores de uso genérico, essa partição era clara, o aparecimento da lógica reconfigurável implicou que o momento em que é necessário definir que funções serão implementadas em hardware e quais as que o serão em software possa agora deslizar dentro de uma janela temporal. O objectivo é atrasar o mais possível essa opção, tendo-se evoluído para linguagens de especificação e programação indistintas, por forma a permitir retardar essa escolha, sem o risco de incorrer em custos por redefinição do projecto. Existe já algum trabalho conducente à integração desta decisão como parte integrante das ferramentas de apoio ao projecto [Harkin et al., 01].

A partilha do mesmo recurso de configuração por múltiplas funções independentes, cada uma delas com os seus requisitos próprios em termos espaciais e temporais, levanta outro problema, o da fragmentação do espaço de configuração. Uma função ocupa durante um certo tempo um determinado número de recursos findo o qual esses recursos são libertados, ficando disponíveis para

serem alocados a outra função. Contudo, será de esperar que a nova função não requeira exactamente os mesmos recursos, pelo que pequenas áreas, incapazes de satisfazer os requisitos de novas funções, vão ficando por ocupar, diminuindo, dessa forma, a eficiência da utilização do espaço configurável. Se os requisitos das novas funções e a sua sequência são conhecidos *a priori*, um conveniente rearranjo das funções em execução pode ser implementado, para que recursos contíguos suficientes sejam garantidos, quando necessário, para novas implementações [Teich et al., 99]. De notar que a dispersão dos componentes de uma função pelo espaço livre disponível conduziria a uma degradação do seu desempenho, atrasando a execução das tarefas e reduzindo a utilização da FPGA. Se a sequência de processamento não é conhecida, a alocação do espaço em tempo real para a implementação de uma nova função pode não ser possível por falta de área livre contígua suficiente (mesmo que a área livre total disponível seja suficiente). Três métodos são propostos em [Diessel et al., 00] com os objectivos de determinar os rearranjos mais adequados e diminuir o tempo de espera para alocar novas funções. Uma implementação prática desses rearranjos, sem interferir com as funções actualmente em execução, é proposta em [Gericota et al., 02] e [Gericota et al., 03a]. Uma análise dos problemas levantados pela fragmentação do espaço de configuração e a análise de um conjunto de propostas de solução desenvolvidas, com base nas características oferecidas pela família XC6200 da Xilinx, são apresentadas em [Compton, 99].

Os dispositivos lógicos programáveis, e em especial os que permitem a reconfiguração parcial dinâmica, ao abrirem um novo leque rico de possibilidades, despertam um conjunto de novos desafios aos quais se torna necessário dar resposta. Esta assunção é igualmente válida para a área do teste, sobre a qual se centrará o próximo capítulo.

## 2.5. SUMÁRIO

Ao longo deste capítulo apresentou-se um sumário do percurso histórico-evolutivo dos dispositivos lógicos programáveis, dando-se especial ênfase a aspectos arquitectónicos, cujo conhecimento é essencial para se compreender e justificar este trabalho.

A área do teste não escapa às novas questões colocadas pela reconfiguração parcial dinâmica. As alterações constantes da funcionalidade e o seu carácter imprevisível, bem como a exigência de uma permanente disponibilidade do recurso, implicam a busca de novas soluções também na área do teste. É no encalço dessas soluções que esta tese é desenvolvida.

O próximo capítulo abordará especificamente os desafios colocados ao teste pela reconfiguração parcial dinâmica e apresentará os estudos mais recentes efectuados neste domínio.

### **3. O TESTE DE FPGAS E OS DESAFIOS INTRODUZIDOS PELA RECONFIGURAÇÃO PARCIAL DINÂMICA**



A tecnologia actual permite a concepção de componentes com elevada densidade de integração (VLSI – *Very Large Scale Integration*) e de grande complexidade, cuja correcta funcionalidade não pode ser verificada exaustivamente com recurso a ferramentas de apoio ao projecto. Se a fase de fabrico fosse executada correctamente e nenhum defeito ocorresse, não seria necessário testar o componente final. Infelizmente, tal perfeição não foi (e certamente nunca será) alcançada, pelo que uma subsequente fase de teste é sempre obrigatória para se garantir um produto fiável.

A redução para escalas submicrométricas do processo de fabrico, com o consequente aumento da densidade e complexidade dos circuitos, agrava ainda mais a probabilidade de ocorrência de defeitos. Sendo o seu espectro praticamente infinito, é necessário mapeá-los num modelo de faltas finito, para que seja viável o teste dos circuitos. Embora a correspondência entre o espectro real de defeitos e o modelo de faltas considerado deva ser o mais fiel possível, não se pode esquecer que um aumento da complexidade do modelo se traduz num agravamento do custo do teste e que esse agravamento na relação custo/cobertura de defeitos pode ser incomportável de um ponto de vista prático. Uma análise dos vários modelos de faltas e da cobertura de defeitos por eles proporcionada ocupa a primeira parte deste capítulo.

Aos primeiros circuitos lógicos comerciais de catálogo, com funcionalidade pré-definida, baixa complexidade e um reduzido número de entradas, era possível aplicar testes funcionais exaustivos. Porém, o aumento da complexidade e do número de entradas tornou os testes funcionais inviáveis, conduzindo à inclusão de estruturas de auto-teste interno e de técnicas de concepção orientadas para a testabilidade que facilitassem a realização do teste e permitissem manter o seu custo dentro de valores aceitáveis. O aparecimento de dispositivos lógicos programáveis veio colocar novos desafios às técnicas existentes. Na realidade, deixámos de ter um componente cuja funcionalidade estava perfeitamente definida para passarmos a ter um conjunto de recursos que podem assumir “qualquer” função. O alvo do teste deixa, neste caso, de ser funcional e passa a ser estrutural, embora ambos, na realidade, se confundam. O desconhecimento da estrutura de implementação dos blocos que constituem a FPGA obriga à adopção de um modelo de faltas híbrido para o seu teste estrutural. A necessidade de cobrir todos os modos de operação, bem como muitas faltas “não-clássicas”, implica o uso de várias configurações de teste. A possibilidade de reconfiguração parcial dinâmica veio acrescentar novos desafios ao problema do teste das FPGAs, mas possibilitou igualmente a concepção de novas soluções para o resolver.

O capítulo encerra com uma síntese de várias propostas apresentadas por um grupo de autores cujo trabalho se tem centrado no teste das FPGAs, procurando aferir das soluções encontradas e identificar a lacuna que se pretende cobrir com a proposta apresentada nesta tese.



### 3.1. ESPECTRO DE DEFEITOS E MODELAÇÃO DE FALTAS

As imperfeições do processo de fabrico impõem a necessidade do teste dos componentes, que tem como objectivo fundamental distinguir entre circuitos integrados bons e defeituosos. Se a complexidade dos circuitos é relativamente baixa, este objectivo é facilmente alcançado com um teste funcional, intimamente relacionado com a função executada pelo circuito, sendo a sua implementação simples e directa. Por exemplo, um CI 74HCT08 [Texas, 97], constituído por quatro portas lógicas ‘E’ de duas entradas, pode ser exaustivamente testado por quatro vectores de teste, aplicados a uma frequência de 1 MHz, em cerca de 4  $\mu$ s. Contudo, um CI com 32 entradas não disjuntas necessita, para o mesmo tipo de teste, de, pelo menos,  $2^{32}=4\,294\,967\,296$  vectores, o que, à mesma frequência de aplicação, implica um tempo de teste de cerca de 72 minutos. Este número cresce exponencialmente se o componente contiver lógica sequencial, o que torna a situação manifestamente incomportável. De qualquer forma, o teste funcional permite apenas aferir do correcto funcionamento do circuito, não traduzindo o tipo ou a origem de um eventual defeito, nem eliminando a possibilidade da sua existência, a qual, embora não se manifestando durante o curto espaço do teste, pode emergir a médio prazo. Tal conhecimento é, todavia, fundamental para que o fabricante introduza alterações correctivas e melhore o rendimento do seu processo de fabrico (*yield*<sup>1</sup>).

Dá-se a designação de *defeito* a um problema físico que afecta a implantação do circuito no suporte de silício. Diferentes processos e escalas de integração apresentam habitualmente tipos diversos de defeitos. A sua quantidade e variedade torna impossível uma enumeração exaustiva, mesmo no caso de circuitos relativamente simples, pelo que, desde cedo, se identificou a necessidade de lidar com representações alternativas, designadas por *faltas*.

Uma *falta* é a manifestação a nível lógico de um defeito, embora não exista tradicionalmente uma relação directa entre faltas e defeitos. A caracterização de um universo de faltas possíveis e das condições em que a sua manifestação pode ter lugar dá origem à definição de um *modelo de faltas*. Atendendo a que a adopção de um modelo de faltas tem por objectivo simplificar a tarefa de detecção dos defeitos reais que podem afectar o funcionamento de um circuito, torna-se claro que um bom modelo deve possuir dois atributos principais:

---

<sup>1</sup> Relação percentual entre o número de componentes testados com sucesso e a totalidade dos componentes produzidos. Um *yield* elevado é um dos parâmetros mais importantes na determinação da relação custo/eficiência do processo de fabrico.

- simplicidade – em particular, é desejável que o número de faltas cresça linearmente com a dimensão do circuito;
- abrangência (cobertura de defeitos) – para um teste que detecte uma elevada percentagem de faltas, o número de defeitos que escapam à detecção deve ser negligenciável.

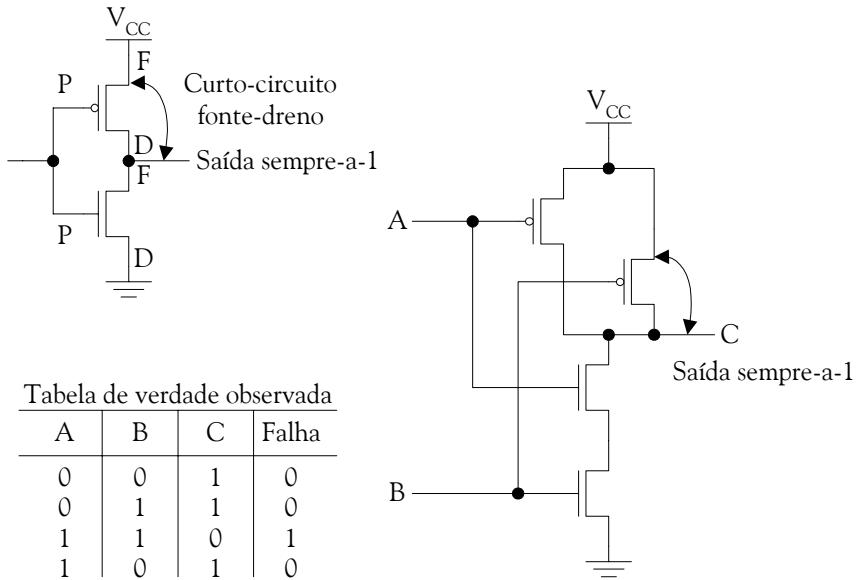
Para ser possível aferir da qualidade de um teste, deve existir um critério de medida que permita o estabelecimento de termos comparativos. O desconhecimento do espectro completo de defeitos de um circuito dificulta a introdução desse tipo de critérios. No entanto, é possível utilizar modelos de faltas finitos, que representem a nível lógico um espectro infinito de defeitos [Hayes, 85] e [Sengupta et al., 99]. Por exemplo, um defeito de isolamento na camada de óxido da porta de um transístor tem como resultado prático a existência de uma ligação de baixa impedância entre a fonte e o dreno desse transístor. Em termos de um modelo de faltas do transístor, teremos um curto-circuito fonte-dreno. Estas faltas podem ser modelizadas pela colocação da saída do transístor num valor lógico fixo (0 ou 1). Se esse transístor for um elemento preponderante na determinação do valor na saída de uma porta lógica, então essa porta apresentará um valor lógico fixo (0 ou 1). Por conseguinte, o modelo de faltas de uma porta lógica do tipo ‘Não-E’, representada na Figura 3.1, pode ser relacionado com o defeito existente no transístor, se se usar um modelo de faltas ao nível do transístor, ou com o comportamento verificado na saída da porta lógica, se se optar por um modelo ao nível da porta. Neste último caso, o modelo mais usado para permitir uma medição da qualidade do teste considera que todos os defeitos levam o circuito a comportar-se como se existisse um valor lógico fixo à entrada ou à saída de uma porta lógica (faltas do tipo *sempre-a-1* e *sempre-a-0*), situação que se manifesta num único nó de cada vez – modelo de faltas individual *sempre-a* (*single stuck-at - ss@*) [Ferreira, 92a]<sup>2</sup>.

Outro exemplo do mapeamento de um defeito num modelo de faltas é a interligação de metal em aberto, devido a uma pequena interrupção na pista. Neste caso, o valor lógico pode continuar a ser propagado através da interrupção, mas por intermédio de uma corrente de tunelamento, atrasando a propagação do sinal [Constancias, 98]. O modelo de faltas ao nível do circuito representa o defeito como uma unidade de atraso do sinal ao longo de todo o comprimento da linha, sendo geralmente denominado modelo de faltas de atraso de propagação (*delay fault model*).

Repare-se que um circuito pode continuar a funcionar correctamente na presença de defeitos, directa ou indirectamente representáveis dentro do modelo de faltas considerado. Passa-se o mesmo

<sup>2</sup> Por razões de simplicidade de escrita, o modelo de faltas individual *sempre-a* será doravante referido apenas como modelo *sempre-a*.

ao nível das faltas, cuja presença pode ou não manifestar-se nas saídas, de acordo com a situação de funcionamento imposta pelos estímulos aplicados nas entradas.



**Figura 3.1:** Modelação de uma falta a nível físico e a nível lógico numa porta ‘Não-E’

Se, devido à presença de defeitos mapeados pelo modelo de faltas considerado, um circuito deixar de funcionar correctamente, diz-se que este revela uma falha. A manifestação de uma falha exige o cumprimento de duas condições:

1. o modelo de faltas deve ser exercitável e observável;
2. deve ser estabelecido um critério para medir a falha.

No exemplo apresentado na Figura 3.1, um defeito no isolamento da porta do transístor (curto entre fonte e dreno) faz com que a saída da porta lógica ‘Não-E’ apresente sempre um valor lógico 1. Para que este defeito possa ser classificado como falha, a porta ‘Não-E’ deve ser exercitada de tal forma que o defeito seja activado e a sua presença observável. Neste caso, se às entradas A e B da porta lógica puder ser aplicado o valor lógico 1 (forçando a saída a 0) e se se puder observar que o valor lógico apresentado pela saída não é 0, então esta falta será classificada como falha [Crouch, 99]. Contudo, se o circuito for construído de tal modo que as entradas da porta não possam ser colocados no valor lógico 1, ou que a sua saída não possa ser observada, então existe a falta mas não a falha, uma vez que aquela não é susceptível de impedir o funcionamento correcto do circuito.

A possibilidade de controlar e observar o valor presente nos nós internos do circuito, permitindo revelar a existência das faltas, depende dos seus níveis de *controlabilidade* e *observabilidade*, definidos como [Santos, 99]:

- *Controlabilidade*: quantifica a dificuldade de forçar um determinado valor num nó a partir das entradas primárias;
- *Observabilidade*: quantifica a dificuldade de propagar o valor lógico presente num nó até uma saída primária.

As faltas que não afectam o funcionamento correcto do circuito podem ser classificadas como não detectáveis, redundantes ou não-testáveis, e podem ou não ser contabilizadas na métrica de detecção de faltas. Outros exemplos de faltas nestas circunstâncias são:

- pequenos atrasos em interligações com folgas grandes;
- faltas do tipo sempre-a em circuitos com lógica redundante ou reconvergente [Abramovici et al., 90];
- correntes de fuga com valores inferiores a um determinado limiar de rejeição.

Em todos estes casos, o componente possui realmente um defeito, que, não resultando num funcionamento incorrecto no imediato, deve ser detectado, por uma questão de fiabilidade, durante os testes de fabrico.

Uma métrica comum para aferir da qualidade de um teste em circuitos digitais é a sua *cobertura de faltas*, calculada relativamente a um dado conjunto de vectores. Dentro da totalidade das faltas que podem ocorrer no circuito, um grupo de vectores exercita e detecta um subconjunto dessas faltas, ao qual corresponde uma cobertura de faltas dada pela Equação 3.1.

**Equação 3.1**

$$\frac{\text{faltas detectadas}}{\text{número total de faltas}} \times 100 (\%)$$

Diferentes ferramentas podem indicar valores diferentes para a cobertura de faltas, dependendo da consideração ou não das faltas não detectáveis, ou da forma como estas são identificadas.

### 3.1.1 MODELOS DE FALTAS

Os modelos de faltas mais comuns actualmente são o sempre-a estático, já referido, os de atraso de transição e de propagação dinâmicos, e os baseados na medida da corrente, inversor (*toggle*) e pseudo-sempre-a. De todos estes, o modelo de faltas mais popular, mais usado e mais estudado é o sempre-a, referido como “clássico” por ter sido o primeiro modelo de faltas a ser enunciado. Embora não tenha validade universal, a sua popularidade resulta dos seguintes atributos [Abramovici et al., 90]:

- representa uma larga fatia dos defeitos físicos;

- é independente da tecnologia, já que o conceito de uma linha sempre-a pode ser aplicado a qualquer modelo estrutural;
- a experiência tem demonstrado que os testes gerados para o modelo sempre-a também detectam muitas faltas “não-clássicas”, entendidas como faltas não cobertas pelo modelo;
- comparado com outros modelos, o número de faltas sempre-a num circuito é pequeno, podendo ainda ser reduzido através de técnicas de colapso de faltas (*fault-collapsing*)<sup>3</sup>;
- o modelo sempre-a pode ser usado para modelar outro tipo de faltas.

A limitação imposta pelo modelo da não ocorrência de mais do que uma falta simultaneamente visa reduzir a complexidade da análise, que, de outro modo, cresceria exponencialmente para valores inexequíveis. Este modelo não tem em consideração características temporais ou de frequência. Um problema que advém do crescimento da complexidade dos circuitos é o aumento do número de planos lógicos, que dificulta a controlabilidade e observabilidade dos seus nós internos, resultando na necessidade de um maior número de vectores de teste.

O modelo de atraso de transição é um modelo de faltas temporal baseado no modelo sempre-a com uma limitação adicional que permite o seu uso na avaliação do domínio temporal. O passo extra consiste em forçar o valor esperado da falta na saída de uma porta num dado instante anterior à observação dessa saída, repetindo-se o processo para a transição oposta. O intervalo entre o forçar da transição e a sua observação não deve exceder um determinado intervalo pré-determinado, normalmente um ciclo de relógio. O modelo de atraso de propagação estende este conceito da consideração do atraso de transição, introduzido por uma única porta lógica, à soma de todos os atrasos introduzidos na propagação de um determinado sinal.

Alguns tipos de defeitos não produzem alterações na função *Booleana* definida para o circuito, pelo que não são mapeáveis nos modelos de faltas do tipo sempre-a, ainda que possam causar um aumento da corrente de fugas, indicando, assim, a possibilidade de existência de problemas de fiabilidade. Uma corrente de fugas elevada pode indicar a existência de uma condição que, embora não afecte actualmente o funcionamento do circuito, o venha a fazer mais tarde, devido, por exemplo, a fenómenos como a electromigração, cada vez mais actuais por causa da utilização de escalas submicrométricas e nanométricas no fabrico de CI's.

O modelo pseudo-sempre-a baseia-se no modelo sempre-a estático, mas, ao invés de procurar conduzir o valor lógico resultante da exercitação da falta até um ponto onde o mesmo possa ser observado, intenta exercitar a falta e medir a corrente de alimentação consumida pelo circuito

<sup>3</sup> Técnica de redução do número de faltas do tipo sempre-a com base na sua equivalência funcional.

nesse momento. Em circuitos baseados em tecnologia CMOS, o valor da corrente quiescente ( $I_{DDq}$ ) é muito baixo, resultando apenas do somatório das correntes inversas de fuga dos transístores que os constituem. Qualquer defeito do tipo ponte entre interligações, ou entre uma interligação e a alimentação (curto), e alguns defeitos do tipo interligações em aberto, resultam numa corrente quiescente uma ordem de grandeza acima da normal.

Frequentemente usado para a medida de corrente, o modelo de faltas inversor, que se baseia na capacidade de exercitar cada nó do circuito, garantindo a possibilidade de definir e inverter o seu valor lógico, constitui uma poderosa ferramenta de simulação. Este modelo é naturalmente usado nos testes baseados na medida da corrente quiescente ( $I_{DDq}$  testing), dado o seu objectivo consistir em exercitar o máximo de pontes e curtos possíveis. É ainda um dos modelos de mais simples aplicação, uma vez que requer apenas que cada nó seja controlável, mas não exige a sua observabilidade.

A principal limitação imposta aos métodos baseados na medida de corrente é a exigência de que os circuitos estejam num estado estático quando ocorre a leitura, visto que a corrente quiescente é muito baixa (da ordem dos  $\mu A$ ) e qualquer actividade no circuito conduziria a valores de corrente, no mínimo, uma ordem de grandeza acima, mascarando qualquer falta.

Acreditava-se que os testes baseados na medida da corrente quiescente se constituiriam como a técnica mais eficaz na detecção de faltas não-sempre-a, que têm aumentado de importância com a continuada diminuição da escala empregue nos processos de fabrico. No entanto, a diminuição do valor da relação entre as correntes de fuga dos transístores nas escalas nanométricas e as suas correntes de defeito, não esquecendo que se trata de valores estatísticos e, como tal, sujeitos à consideração de uma distribuição e de um desvio-padrão em relação à média, tornam o valor da corrente quiescente de componentes com defeito difícil de diferenciar face ao valor exibido pelos componentes bons. Esta afirmação é tanto mais verdadeira quanto se considerarem as variações desse valor em função das variabilidades paramétricas verificadas no processo de fabrico, caso em que o valor do desvio-padrão será maior [Lu et al., 02].

O desenho estrutural do circuito tem uma influência considerável no impacto que um defeito impõe ao seu comportamento. No entanto, esta informação tem sido ignorada no desenvolvimento dos modelos de faltas ao nível do transistor ou da porta lógica. Em [Shen et al., 85], é proposta uma metodologia para o mapeamento dos defeitos físicos nas alterações que ocorrem no comportamento do circuito como consequência desses defeitos. Desta forma, faltas especificamente derivadas da tecnologia e do desenho do circuito são registadas e definidas de acordo com a sua probabilidade de ocorrência. Uma das conclusões desta análise é que os defeitos de fabrico originam uma muito

maior gama de faltas do que as modeladas pelo modelo sempre-a. De forma similar, [Dekker et al., 88] prova que muita informação com relevância para a análise do comportamento do circuito pode, em caso de defeito, ser retirada do seu desenho estrutural. Estas abordagens deram origem ao que se veio a designar por Análise de Faltas por Indução (IFA - *Inductive Fault Analysis*). Os três principais passos desta abordagem são:

1. a geração de uma lista de defeitos físicos a partir de dados estatísticos retirados do processo de fabrico;
2. a análise das faltas provocadas por esses defeitos;
3. a classificação dos tipos de faltas e ordenação baseada na sua probabilidade de ocorrência.

Os modelos de faltas extraídos por este processo são geralmente referidos como modelos realísticos de faltas, derivando o termo realístico do facto de cada falta ter como base um defeito físico.

### 3.1.2 DEFEITOS PARAMÉTRICOS

A evolução tecnológica verificada no sentido da utilização de escalas submicrométricas e nanométricas no processo de fabrico, aliada ao consequente uso de tensões de alimentação cada vez mais baixas, tem contribuído para a crescente importância atribuída aos defeitos paramétricos, originados nas variações verificadas nos parâmetros individuais associados a cada transístor ou interligação de metal devido a imperfeições introduzidas durante a manufatura das pastilhas de silício. As previsões expressas no *International Technology Roadmap for Semiconductors*<sup>4</sup> para os próximos 15 anos acentuam ainda mais esta tendência, como se infere do gráfico apresentado na Figura 3.2 [Allan et al., 02], na tentativa de responder aos desafios implícitos na Lei de Moore [Moore, 65].

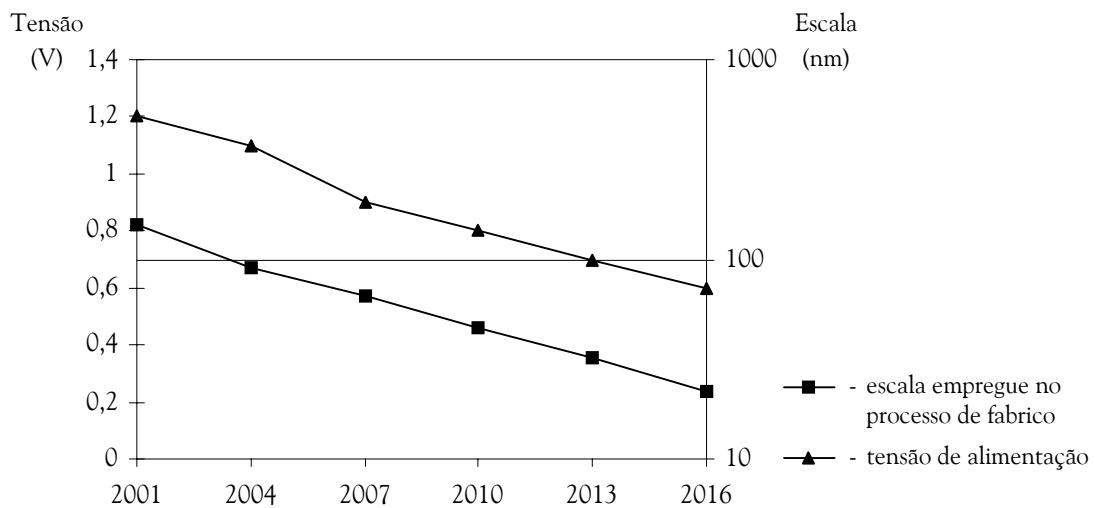
Existem milhares de variáveis no processo de fabrico de circuitos baseados em tecnologia CMOS que podem provocar variações nos parâmetros desses circuitos e resultar no aparecimento de defeitos ténues (*soft defects*)<sup>5</sup> que vão afectar o desempenho do CI [Segura et al., 02]. No gráfico da Figura 3.3, construído a partir de dados fornecidos pela Intel<sup>6</sup>, é visível o aumento da importância dos defeitos ténues com a diminuição da escala, em relação à totalidade dos defeitos.

---

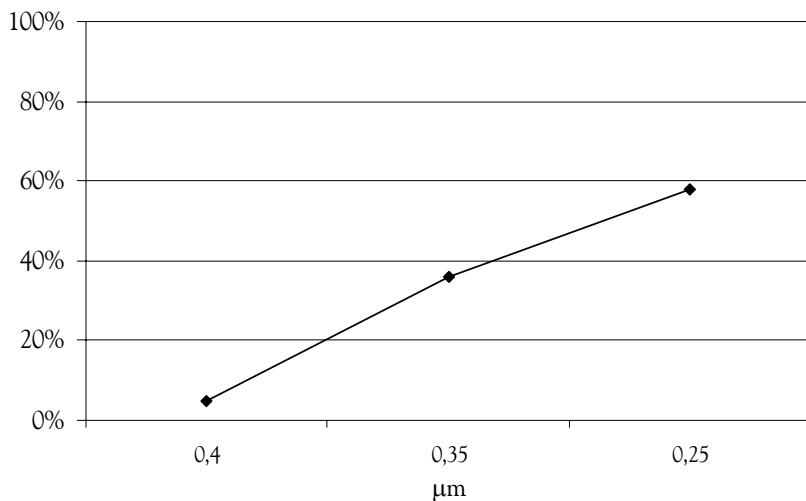
<sup>4</sup> O *International Technology Roadmap for Semiconductors* resulta de um consenso entre as indústrias de semicondutores quanto à estimativa das suas necessidades de investigação e desenvolvimento num horizonte de 15 anos.

<sup>5</sup> Por oposição aos denominados defeitos francos (*hard defects*).

<sup>6</sup> Intel Corporation



**Figura 3.2:** Evolução previsível das características de fabrico dos semicondutores



**Figura 3.3:** Variação percentual das falhas devidas a defeitos ténues com a redução de escala em relação à totalidade das falhas [Needham et al., 98]

Concorrem para alterar significativamente a velocidade do circuito as variações que se verificam nos parâmetros dos transístores e das interligações, de que são exemplo [Segura et al., 02] e [Sengupta et al., 99]:

- as variações no comprimento e na largura do canal dos transístores;
- as variações na espessura da camada de óxido da porta dos transístores;
- as variações na dopagem, que conduzem a alterações na tensão de limiar e na resistência em condução dos transístores;
- as variações na resistência de contacto;

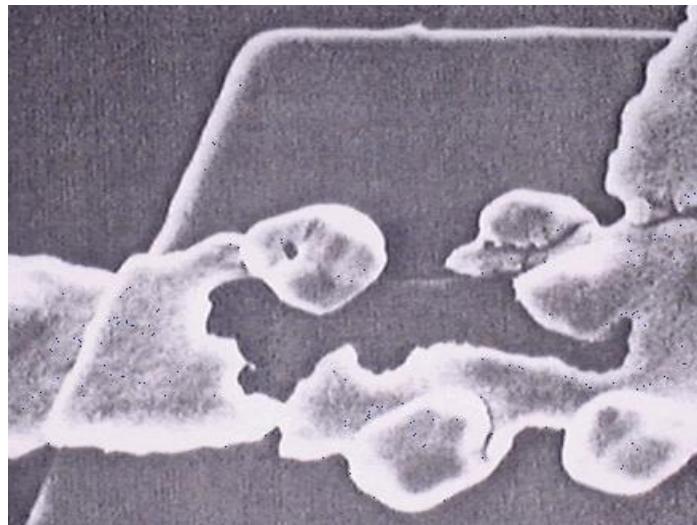
- a não uniformidade das interligações de metal – variações da largura, espessura, espaçamento, granulosidade e densidade de corrente;
- as variações na espessura do dieléctrico entre diferentes camadas de metal.

As diminuições de escala e as alterações no processo de fabrico têm contribuído igualmente para um aumento do número de defeitos nas interligações de metal, que se traduzem em regra por uma diminuição acentuada da secção da linha num determinado ponto, implicando um aumento da resistência e, consequentemente, do atraso de propagação. O emprego, em técnicas de fabrico mais recentes, do cobre nas camadas de interligação, para além de influir no aumento da incidência de circuitos abertos, tem conduzido a um maior aparecimento deste tipo de defeitos resistivos<sup>7</sup>. Contudo, embora a resistência cresça exponencialmente com a diminuição da secção, o seu valor só se torna significativo quando essa diminuição atinge 95% do valor normal, pelo que estes defeitos são, na prática, indetectáveis. No entanto, o aumento do valor da densidade de corrente nesse ponto da interligação resulta numa maior ameaça de electromigração e, por conseguinte, no aparecimento em operação de circuitos abertos, cuja propensão escapa aos testes de fabrico [Lach et al., 98]. Este tipo de defeitos é igualmente conhecido pela designação de defeitos de fiabilidade, uma vez que, não se manifestando de início, afectam a fiabilidade dos circuitos a médio ou longo prazo, diminuindo o seu tempo médio de funcionamento até à falha (*Mean Time To Failures - MTTF*).

O fenómeno de electromigração deriva da interacção entre os electrões e as imperfeições resultantes da distribuição não uniforme do metal, cobre ou alumínio, que formam a linha e que, na maior parte dos CIs, é policristalino, isto é, constituído por aglomerações compostas por cristais orientados aleatoriamente em resultado de uma rápida solidificação. A corrente eléctrica pode ser visualizada como uma “brisa de electrões” que lentamente move os átomos numa determinada direcção, transportando-os entre as periferias desses aglomerados. Na direcção do fluxo de electrões, os átomos de metal serão depositados, com o tempo, em pequenos montículos, enquanto, na direcção oposta, se vão criando espaços vazios. Os espaços irão reduzindo a condutividade da linha, podendo eventualmente conduzir à cessação total da condução. Os montículos podem introduzir tensões mecânicas ou criar pontes com as linhas vizinhas [Otten et al., 02]. Na Figura 3.4, encontra-se ilustrado um processo de electromigração, podendo observar-se a formação dos pequenos montículos de metal e o acentuar do espaço livre, que converge para o rápido aparecimento de um circuito aberto.

---

<sup>7</sup> Notas retiradas da palestra “Silicon Technology Advances and Implications on Test”, proferida por Greg Spirakis, Director of Design Technology da Intel, no decorrer do 7<sup>th</sup> IEEE European Test Workshop, 2002.



**Figura 3.4:** Visualização de um fenómeno de electromigração<sup>8</sup>

Outra consequência da diminuição da escala é a redução do espaçamento entre linhas. Na Figura 3.5, ilustra-se uma arquitectura de interligação em cobre em que é visível a grande proximidade entre linhas, quer da mesma camada, quer de diferentes camadas. Imperfeições em linhas de metal muito próximas, que inicialmente não se encontram em contacto, e que, por isso, não provocam qualquer falha durante o teste, podem, devido ao aumento de temperatura e da consequente dilatação do metal durante a sua operação normal, causar uma falha. O dielétrico de óxido de metal é perfurado e os electrões fluem livremente, originando que os átomos de metal se interliguem, estabelecendo uma ponte permanente.



**Figura 3.5:** Visualização de uma arquitectura de interligação em cobre<sup>8</sup>

---

<sup>8</sup> Fotografia cedida pela Intel e retirada da apresentação da palestra “Defect Based Test Challenges in a Nanometric World”, proferida por Joan Figueras no decorrer da 2<sup>nd</sup> IEEE Latin-American Test Workshop, 2001.

Outro tipo de defeito ténue é observado no isolamento na camada de óxido da porta dos transístores. Embora não se apresente como uma perfuração no isolamento representando um curto-circuito entre a porta e o substrato, uma série de lacunas cuja distribuição física é transversal à camada de óxido propicia o aparecimento de uma corrente. Apresentando uma resistência óhmica elevada, este tipo de defeito escapa à detecção por testes baseados na medida da corrente quiescente, uma vez que o aumento verificado é muito pequeno. As quebras de isolamento entre porta e fonte, ou dreno, apresentam resistências óhmicas significativamente mais baixas, afectando o funcionamento do circuito, e, como tal, gerando correntes quiescentes muito mais elevadas e facilmente detectáveis.

Não sendo na generalidade, conforme exposto, suficientemente importante para se denunciar durante os testes iniciais, esta classe de defeitos manifesta-se após um longo período de operação. Emergindo como faltas do tipo sempre-a ou transitórias, colocam em evidência a importância que os testes realizados ao longo da vida útil do componente têm para um aumento da fiabilidade dos sistemas [Shnidman et al, 98].

### 3.1.3 DEFEITOS CARACTERÍSTICOS DAS FPGAs

Um outro problema que afecta sobretudo os dispositivos que, como as FPGAs, têm por base funcional uma matriz de memória é a sua susceptibilidade à radiação cósmica. Quando uma partícula ionizada de elevada energia atinge um circuito integrado, provoca a libertação de uma série de electrões à medida que, penetrando o substrato do semicondutor, vai perdendo energia. Se a zona de impacto se situar perto de ou num nó do circuito, o excesso de carga induz um impulso transitório (SET – *Single Event Transient*), com algumas centenas de picosegundos de duração (o valor exacto depende das características do circuito e da energia da partícula). Se o comprimento do impulso for superior ao tempo de transição da lógica presente no circuito, o impulso não é atenuado, mesmo que a partícula possua uma energia baixa. Se o nó de impacto é um nó de dados de um circuito de retenção, a tensão induzida força momentaneamente no circuito um estado instável. Se a lógica falhar o restabelecimento do estado anterior, o circuito assumirá eventualmente um novo valor lógico. [Rockett, 01] e [Alexandrescu et al., 02]. Dada a diminuição do tamanho dos transístores e da sua tensão de alimentação, a quantidade de carga necessária para alterar o seu estado também se reduziu, pelo que a sua susceptibilidade ao impacto de partículas deste género aumentou. No passado, este problema era uma preocupação somente em ambientes hostis como o espaço, mas os CIs baseados em tecnologias nanométricas tornaram-se tão sensíveis que mesmo a radiação ao nível do mar origina níveis inaceitáveis de faltas provocadas pela indução de carga (SEUs – *Single Event Upsets*) [Nicolaidis et al., 99]. Sendo responsáveis pela alteração dos

valores armazenados em elementos de memória, a sua sensibilidade aos SEUs encontra-se profundamente correlacionada com a quantidade de elementos de memória presentes no circuito, tais como *flip-flops* e células de memória [Lima et al., 02]. Em sentido contrário ao da diminuição de escala, a concepção de FPGAs de tamanho cada vez maior, com o objectivo de disponibilizar um crescente número de portas lógicas equivalentes, implica que o alvo das partículas aumente de tamanho, crescendo a probabilidade de falha [Lach et al., 98] e [Shnidman et al, 98]. Duas técnicas para a recuperação de erros provocados por SEUs na memória de configuração das FPGAs, embora com algumas limitações, são propostas em [Xilinx, 00e] e em [Huang et al., 01]. Essas limitações prendem-se com a possibilidade de as tabelas de consulta poderem ser usadas como memórias distribuídas, situação em que, em caso de detecção de um erro, se o vector para a sua correcção abrange uma dessas tabelas, a sua correcção obrigar à paragem do sistema como única forma de garantir a coerência dos dados nela constantes.

### 3.1.4 O MODELO SEMPRE-A

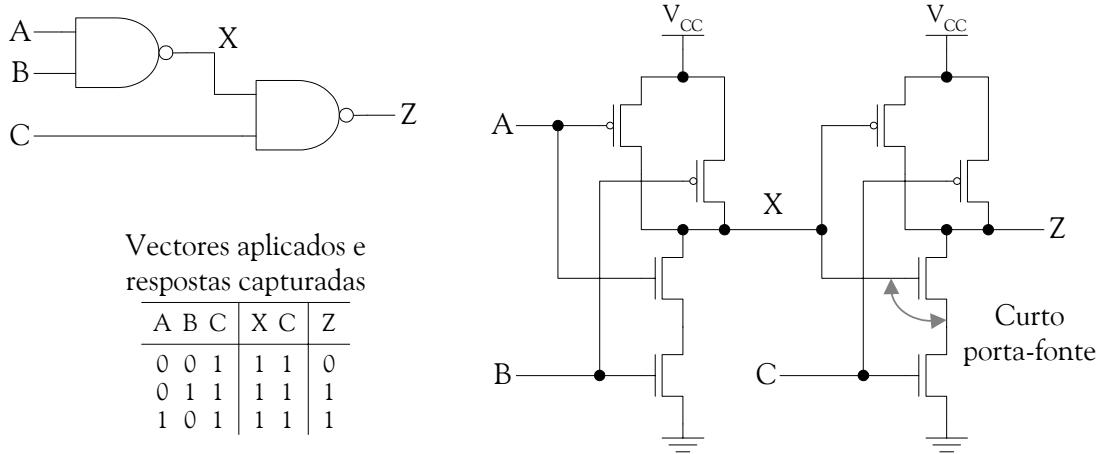
Embora continue a ser um dos modelos mais usados na indústria para o teste final de CIs, a fiabilidade do modelo sempre-a tem sido posta em causa por alguns autores no que diz respeito, por exemplo, à sua eficácia na detecção de defeitos paramétricos com um comportamento do tipo analógico, que, como exposto, não sendo suficientemente severos para causar uma falha lógica, se comportam como pontes resistivas. Outra razão apontada consiste na dificuldade de prever o nível de cobertura de defeitos a partir da cobertura de faltas proporcionada pelo modelo [Park et al., 94], [Powell et al., 94], [Thompson, 96], [Renovell et al., 98], [Sar-Dessai et al., 99] e [McCluskey, 00].

Um estudo levado a cabo com o objectivo de conhecer na prática a eficácia relativa de diferentes técnicas de teste como o teste sempre-a, testes temporais e testes baseados na medição da corrente quiescente, concluiu que, embora o modelo sempre-a não proporcione uma representação muito perfeita dos actuais defeitos de produção, é uma poderosa ajuda no desenvolvimento de bons testes de fabrico [McCluskey et al., 00]. Provou-se nesse estudo que conjuntos de vectores de teste que detectam cada uma das faltas sempre-a mais do que uma vez possuem uma maior eficácia na detecção de defeitos, como se mostra no exemplo ilustrado na Figura 3.6. Existem três vectores capazes de detectar a falta Z sempre-a-1: ABC= 001, 011, 101<sup>9</sup>. Para a geração de um conjunto de vectores que detectasse cada falta sempre-a pelo menos uma vez, seria suficiente incluir apenas um destes três vectores. No entanto, a aplicação de cada um dos três vectores pode conduzir a resultados diferentes se existir uma ponte resistiva entre a porta e a fonte de um dos transístores,

---

<sup>9</sup> Estes vectores detectam igualmente as faltas X e C sempre-a-0, mas esse facto é irrelevante para o exemplo presente.

conforme indicado na Figura 3.6. Assume-se, assim, que, se apenas um dos transístores *pmos* na porta AB conduz, o curto consegue fazer baixar suficientemente a tensão em X para que o transístor *pmos* conduza e a tensão em Z se mantenha num nível lógico 1. Contudo, se ambos os transístores *pmos* da porta AB conduzirem, então o transístor *pmos* em X não conduz e a saída Z apresenta um valor lógico 0, indicando a presença do defeito.



**Figura 3.6:** Detecção de um defeito por teste múltiplo de uma falta sempre-a

Apenas o vector  $ABC = 001$  permite a detecção do defeito, mas, no caso de um conjunto de vectores que cubra 100% de faltas sempre-a, este vector poderia não estar presente, falhando a detecção. Trata-se de um defeito interessante, pois a sua detecção não depende só das entradas da porta lógica onde se situa, mas também das entradas das portas lógicas a montante. A prática actual usa os vectores gerados a partir da aplicação do modelo sempre-a na geração de testes de fiabilidade [McCluskey et al., 00].

A eficácia do modelo sempre-a na detecção de defeitos arbitrários, não explicitamente especificados pelo modelo, que ocorrem num determinado ponto do circuito, é reafirmada em [Lee et al., 02]. Segundo os autores, existe uma forte correlação entre o número de vezes que um determinado ponto do circuito é observado e o número de defeitos detectados. Conclui-se que a geração de um número de vectores superior ao necessário para cobrir 100% de faltas sempre-a aumenta a probabilidade de excitação de um defeito não explícito nesse modelo, resultando na sua eventual detecção.

De igual forma, embora a geração de vectores de teste para o modelo sempre-a se efectue com o pressuposto da não existência de mais do que uma falta deste tipo simultaneamente, a eficácia dos vectores na detecção de múltiplas faltas sempre-a foi comprovada por [Hughes et al., 84] e [Hughes, 88]. Aliás, no decorrer deste estudo, conjuntos maiores de testes mostraram-se mais eficazes do que conjuntos mais pequenos, demonstrando que a geração de testes segundo este

modelo, para a detecção múltipla de faltas sempre-a, aumenta a sua eficácia, ao detectar defeitos não explícitos no modelo.

### 3.2. TESTE ESTRUTURAL

Partindo do pressuposto que todos os defeitos fracos foram detectados durante os testes de fabrico, as faltas posteriores possíveis numa FPGA podem derivar de defeitos ténues que, em consequência do seu uso, se manifestem como faltas, ou ser provocadas por SEUs após a configuração. Os primeiros podem ocorrer em qualquer das partes constituintes da FPGA, enquanto os segundos se restringem às células de memória presentes nos blocos deste tipo, na memória de configuração, nas tabelas de consulta e nos eventuais registo configurados na parte lógica. Esta conclusão, apoiada na análise previamente efectuada, contraria a ideia defendida em [Shnidman et al., 98], segundo a qual, após a configuração da FPGA, as únicas faltas relevantes seriam as sempre-a, cuja ocorrência se deveria à incidência de partículas de radiação. Aliás, em [Metra et al., 98], é colocada em evidência a susceptibilidade das FPGAs ao aparecimento de defeitos físicos, quando operadas em ambientes adversos e sujeitas a reconfigurações constantes. Segundo aqueles autores, estes defeitos podem ser realisticamente descritos como faltas sempre-a, transístores sempre-fechados ou sempre-abertos e tipo ponte.

A implementação de um teste concorrente com o funcionamento da FPGA invalida logo à partida o uso de determinadas estratégias, nomeadamente as que são baseadas na corrente quiescente, uma vez que não se pretende retirar de serviço o componente durante a realização do teste, nem se consegue isolar uma parte para este fim. De igual forma, é eliminada a hipótese de testes funcionais. Outro motivo prende-se com o facto de que uma determinada função ocupa somente uma porção dos recursos disponibilizados, pelo que um teste funcional garantiria apenas o teste dos recursos presentemente ocupados, deixando por testar recursos que poderiam vir a ser usados mais tarde por outra função.

A implementação de testes concorrentes desenvolvendo lógica do tipo auto-verificável (*self-checking*) evitaria a retirada de operação das funções a testar, mas implicaria um dispêndio extra de recursos e obrigaria ao desenvolvimento de estruturas para cada um dos circuitos a implementar.

Uma vez que a funcionalidade que a FPGA poderá assumir, quer no todo, quer em parte, é desconhecida à partida, podendo inclusive uma mesma função ser alocada a diferentes recursos ao longo do tempo, a opção por um teste estrutural que regularmente garanta o bom estado da plataforma de implementação apresenta-se como a mais viável para garantir uma elevada fiabilidade de funcionamento do sistema. Obviamente que um teste estrutural implica que as áreas a testar

estejam livres, isto é, nenhuma funcionalidade correntemente a ser usada deve estar implementada nessas áreas, e que este se processe de forma faseada e regular.

Embora as FPGAs sempre tivessem oferecido a possibilidade de reprogramação, a sua configuração tendia a ser estática, no sentido em que após a fase de desenvolvimento, em que era definida a configuração da lógica a implementar, permanecia fixa ao longo da operação do sistema. Na generalidade dos casos, o uso destes componentes resumia-se à substituição de um punhado de componentes com funções lógicas pré-definidas, tirando partido da redução nas dimensões da carta de circuito impresso e, consequentemente, dos custos que tal opção implicava. Nesses casos, o desenvolvimento de um teste funcional ou de lógica do tipo auto-verificável era suficiente para garantir o bom funcionamento do sistema. No entanto, com o advento da possibilidade de reconfiguração parcial dinâmica, tal presunção deixou de ser verdadeira e novas questões surgiram, implicando a necessidade de uma nova abordagem ao teste destes dispositivos. O espaço de configuração passou a ser encarado como um espaço tridimensional em que às duas dimensões espaciais se juntou uma terceira, a dimensão temporal. Adicionalmente, o aumento da capacidade e das possibilidades lógicas das FPGAs alargou o seu emprego à implementação de funções mais complexas, que antes eram alvo dos ASICs. A associação destes dois pontos conduziu à implementação de sistemas em que o recurso de configuração é partilhado por múltiplas funções independentes que são configuradas à medida que vão sendo necessárias e de acordo com a disponibilidade de recursos na própria FPGA.

Esta situação acarreta que uma mesma função possa ser invocada pelo sistema múltiplas vezes, espaçadas temporalmente, ocupando em cada uma das suas implementações recursos diferentes, conforme a sua disponibilidade imediata. As FPGAs tornaram realidade o conceito de *hardware evolutivo*, que, no entanto, levanta uma outra dificuldade, a de à partida não se saber sequer qual a função e quais os recursos que ela ocupará como resultado da geração evolutiva do circuito. De referir que, como mencionado em [Thompson et al., 99], as técnicas de hardware evolutivo conduzem, por vezes, a circuitos “bizarros” que, embora cumprindo os requisitos definidos inicialmente, possuem uma arquitectura cujo funcionamento escapa à compreensão dos projectistas. A inexistência de técnicas de “evolução para a testabilidade”, no sentido de que o circuito resultante seja testável, dificulta ainda mais o estabelecimento de uma estratégia de teste. Aliás, o estudo de técnicas de condicionamento evolutivo, que garantam uma “evolução para a testabilidade”, é um campo de exploração ainda em aberto.

Por estas razões, deixa de fazer sentido a geração de testes orientados à aplicação, conforme proposto em [Renovell et al., 00], mesmo que a geração dos vectores de teste tenha em conta, como é o caso, a estrutura da FPGA onde o circuito é implementado. Qualquer falta que afecte recursos

da FPGA não utilizados é considerada redundante, por não ser detectável através dos testes gerados com base naquela aplicação específica. A reconfiguração da FPGA e a possível ocupação desses recursos conduziriam eventualmente ao mau funcionamento do sistema. Outro problema do emprego de testes orientados à aplicação é a diminuição da disponibilidade do sistema, uma vez que a aplicação do teste implica uma paragem no seu funcionamento normal. Pelo contrário, a aplicação de um teste estrutural, ao testar a totalidade dos recursos da FPGA, permite a detecção de todas as faltas possíveis, preavendo falhas do sistema após sucessivas reconfigurações.

O modelo estrutural descreve um circuito como uma coleção de pequenas “caixas”, denominadas componentes ou elementos. Este modelo é frequentemente hierárquico, no sentido em que um componente é, por sua vez, descrito por um conjunto interligado de componentes de mais baixo nível. O nível mais baixo é constituído por elementos denominados *primitivas*, cujo modelo funcional é suposto ser conhecido. O teste estrutural, aplicado com base no modelo estrutural do circuito [Eldred, 59], é usado com o objectivo de verificar a topologia do circuito fabricado, ou seja, de garantir que todas as ligações estão intactas e que o funcionamento das primitivas (portas lógicas, multiplexadores, *flip-flops*) é o esperado. Os testes são gerados de modo a que o estado dos sinais internos possa ser observado nas saídas primárias do circuito. Numa FPGA, a aplicação de um teste estrutural implica o uso de várias configurações de teste, de maneira a exercitar todos os modos de operação, uma vez que a própria estrutura é programável. A aparente complexidade desta abordagem é, no entanto, enganadora, pois tratando-se de uma estrutura regular, o desenvolvimento, quer das configurações de teste, quer dos vectores a usar para atingir o nível desejado de cobertura de faltas, é grandemente simplificado.

### 3.2.1 TESTE DE ESTRUTURAS REGULARES

Embora os modelos de faltas associados a estruturas regulares se orientem sobretudo para o teste de memórias e a sua exploração se encontre fora do âmbito desta tese<sup>10</sup>, o facto de as FPGAs serem estruturas regulares e terem como base de definição da sua funcionalidade uma memória de configuração, torna importante a abordagem destes modelos neste ponto. Aliás, como referido em [Renovell et al., 98a], o teste de uma FPGA apresenta-se como uma boa coleção de todos os problemas encontrados no teste.

Existem dois tipos de modelos de faltas associados a estruturas regulares: faltas clássicas e não-clássicas [Stroud, 02]. Os modelos clássicos incluem:

---

<sup>10</sup> Para um aprofundamento da modelação de faltas e teste de memórias, aconselha-se a consulta de [Mazumder et al., 96] e [Bushnell et al., 00].

- *faltas numa matriz de células de memória (cell array faults)*: o conteúdo da célula encontra-se sempre-a-1 ou sempre-a-0; a detecção destas faltas implica a escrita em cada célula dos valores lógicos 0 e 1;
- *faltas de endereçamento (addressing faults)*: selecção de um endereço errado devido a uma falta na lógica de descodificação de endereços; a detecção destas faltas implica o teste de todos os endereços;
- *faltas nas linhas de dados (data line faults)*: faltas nas entradas ou nas saídas dos registos de dados que impedem a sua correcta escrita ou leitura; a detecção destas faltas implica que em cada linha de dados sejam colocados os valores lógicos 0 e 1;
- *faltas de leitura/escrita (read/write faults)*: resultam de faltas do tipo sempre-a nas linhas de controlo ou na lógica de controlo da escrita/leitura, que impedem as operações de escrita e leitura na matriz de células; a detecção destas faltas implica a realização de uma operação de escrita e leitura em todas as células.

Os modelos de faltas não clássicos, para estruturas regulares como as memórias, incluem:

- *faltas de transição (transition faults)*: o valor presente na célula é incapaz de transitar entre 0→1 ou 1→0; para detectar esta falta é necessário testar ambas as transições;
- *faltas de retenção de dados (data retention faults)*: a célula perde o seu conteúdo após um certo período de tempo; a detecção destas faltas implica a leitura do valor contido na célula após um dado período de inactividade;
- *faltas por leitura destrutiva (destructive read faults)*: a leitura do conteúdo da célula implica uma mudança no valor armazenado; a detecção destas faltas implica a leitura múltipla do conteúdo de uma célula, para ambos os valores lógicos, após uma operação de escrita;
- *faltas sensíveis ao conjunto de valores (pattern sensitivity faults)*: o conteúdo de uma célula é sensível ao conteúdo de outras células; uma falta em ponte entre duas células é um exemplo de uma falta deste género; a detecção destas faltas implica a alteração dos valores nas células adjacentes àquela que está sob teste e a verificação do seu conteúdo;
- *faltas de acoplamento (coupling faults)*: o conteúdo de uma célula é afectado por operações realizadas noutras células; são diferentes das anteriores, no sentido em que estas se manifestam durante as transições de valores que ocorrem nas células adjacentes, devido a operações de leitura ou de escrita, e não pelo seu conteúdo estático.

Os valores lógicos requeridos para a detecção da sensibilidade de uma célula, face aos valores presentes em células adjacentes, dependem da colocação das células de memória na matriz. Como

se viu no capítulo anterior, numa FPGA com tecnologia de programação baseada em memória estática, as células da memória de configuração encontram-se distribuídas por entre os elementos lógicos que controlam, para permitir a definição da funcionalidade lógica requerida. Sendo assim, a geração de testes para este tipo de faltas torna-se inviável pelo desconhecimento da colocação espacial das células.

Outros casos, como as faltas de endereçamento e de dados, ou as faltas nas próprias células de memória, têm aplicação directa na geração dos testes para as tabelas de consulta de uma FPGA. O seu pequeno tamanho (apenas 16 células) e a sua repetição em todos os blocos lógicos simplificam grandemente a geração desses testes.

Em virtude da necessidade do emprego de mais que uma configuração para permitir o teste estrutural do bloco lógico, são indirectamente testadas faltas do tipo leitura/escrita e transição por configuração e leitura de diferentes funções lógicas implementadas na tabela de consulta, e faltas nas células de configuração dos multiplexadores programáveis que complementam o seu conteúdo.

### 3.2.2 MODELO DE FALTAS HÍBRIDO

Com base nos enunciados antecedentes, vários autores propuseram um modelo de faltas específico para o teste dos blocos lógicos das FPGAs baseadas em memória estática [Fawcett, 94], [Huang et al., 96], [Huang et al., 96a], [Wang et al., 97], [Huang et al., 98], [Metra et al., 98], [Mitra et al., 98], [Renovell, 98] e [Renovell et al., 98a]. Trata-se de um modelo de faltas híbrido baseado no modelo sempre-a, modelado quer física, quer funcionalmente. Os blocos lógicos são usualmente constituídos por três tipos diferentes de primitivas funcionais<sup>11</sup>: tabelas de consulta, *flip-flops* do tipo D e multiplexadores.

No modelo proposto, uma falta pode ocorrer somente num bloco lógico e afectar apenas uma primitiva. Numa tabela de consulta, uma falta pode ocorrer numa das células da matriz de memória, no descodificador ou nas linhas de entrada e de saída. Uma falta na matriz de memória tem como consequência que as células sejam incapazes de guardar o valor lógico correcto, 0 ou 1, podendo um qualquer número de células da matriz ser afectada. Uma falta no descodificador resulta num endereçamento errado, podendo as linhas de entrada ou de saída da tabela apresentar faltas do tipo sempre-a. Por aplicação do conceito de falta equivalente, a falta num dos bits do descodificador pode ser reduzida por colapso de faltas a uma falta sempre-a na linha de entrada da tabela. As faltas sempre-a na linha de saída são detectadas quando a matriz de memória é testada.

---

<sup>11</sup> Ver capítulo 2.

Para os multiplexadores, é usado um modelo funcional, assumindo-se faltas do tipo sempre-a nas entradas e nas saídas da primitiva, incluindo as entradas configuráveis nos multiplexadores programáveis, uma vez que a estrutura da sua lógica interna é desconhecida. A validação desta abordagem, necessária por esta aproximação não garantir *a priori* a detecção de faltas no interior da primitiva por aplicação de um conjunto de vectores de teste gerados com base no modelo, é apresentada em [Portal, 99]. Um conjunto de vectores de teste gerados para detectar 100% das faltas sempre-a-0 e sempre-a-1 nas entradas e na saída do multiplexador foi aplicado a três tipos de implementação diferentes (implementação lógica em dois planos, implementação lógica em árvore e implementação eléctrica em árvore), tendo-se concluído pela eficácia do modelo.

Os *flip-flops* possuem também, a exemplo das tabelas de consulta e dos multiplexadores, uma entrada programável, que define qual o seu valor de inicialização, mais duas entradas: uma que coloca o *flip-flop* no seu estado inicial programado (INIT); e outra que o coloca no estado complementar esse (REV). Um modelo funcional é igualmente aplicado na geração dos vectores para o teste desta primitiva, sendo testada a escrita de um 0 e de um 1, e as entradas de inicialização ao valor configurado (INIT) e ao seu complementar (REV) para as duas configurações possíveis do valor inicial, 0 ou 1. Uma análise mais profunda da estrutura e da geração dos testes para cada uma destas primitivas, tendo em conta estes modelos de faltas, é efectuada no capítulo 7, dedicado ao teste da FPGA.

No teste estrutural dos recursos lógicos da FPGA, têm-se distinguido duas abordagens [Huang et al., 96]:

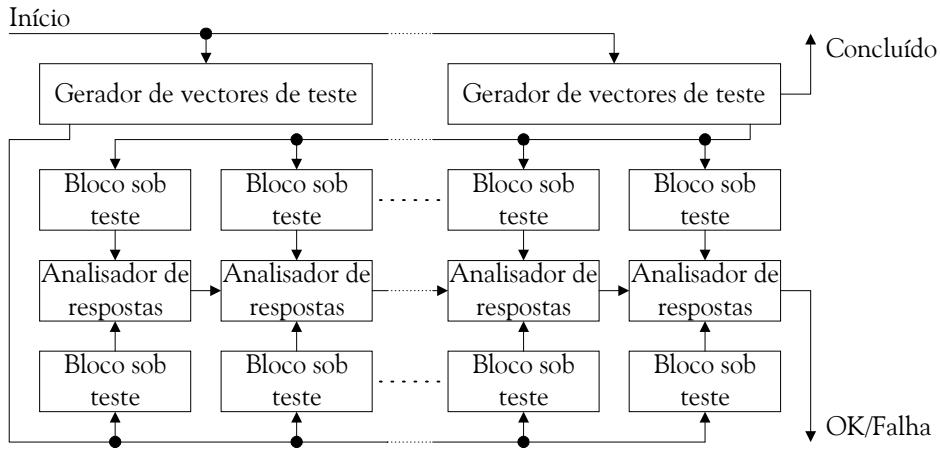
- as baseadas no uso de estruturas de auto-teste interno<sup>12</sup>;
- as baseadas no aproveitamento da disposição em matriz dos recursos.

No primeiro caso, um grupo de blocos lógicos programáveis é configurado como gerador de vectores de teste e analisador de respostas, enquanto os restantes são testados, conforme se ilustra na Figura 3.7. Os blocos sob teste são repetidamente reconfigurados, de forma a que todos os seus modos de operação sejam testados [Stroud et al., 96a]. Esta abordagem requer, no mínimo, uma segunda sessão de teste, em que os papéis atribuídos aos dois grupos são invertidos.

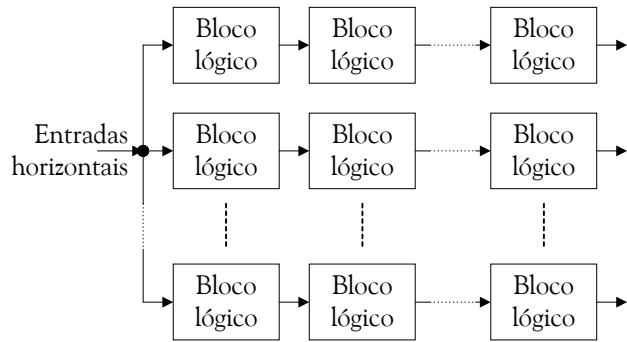
No segundo caso, os blocos lógicos da FPGA são interligados em vários conjuntos de matrizes unidimensionais disjuntas, com ligações unilaterais ao longo de uma linha, isto é, matrizes onde as

<sup>12</sup> Uma análise aprofundada (teórica e prática) do auto-teste pode ser encontrada em [McCluskey, 85], [McCluskey, 85a], [Agrawal et al., 93], [Agrawal et al., 93a], [Dufaza, 98], [Zorian, 99] e [Stroud, 02].

ligações entre os blocos existem numa única direcção. A Figura 3.8 ilustra a implementação desta segunda abordagem.



**Figura 3.7:** Arquitectura genérica para implementação de auto-teste interno numa FPGA [Stroud, 02]



**Figura 3.8:** Arquitectura de uma estratégia de teste baseada em matrizes unidimensionais [Huang et al., 96]

O número de sessões de teste pode ser significativamente reduzido se forem estabelecidas as necessárias condições de testabilidade, nomeadamente a construção de matrizes unidimensionais C-testáveis<sup>13</sup> [Huang et al., 96].

É com base nestas duas abordagens que se tem desenvolvido a maioria dos trabalhos que versam este tema, de cujos resultados e insuficiências se dá conta na próxima secção.

### 3.3. ESTADO DA ARTE

Desde o aparecimento dos primeiros dispositivos lógicos programáveis, colocou-se a questão do seu teste, tendo em atenção as suas características específicas. Embora, em termos conceptuais, uma

<sup>13</sup> Propriedade das estruturas lógicas em matriz, constituídas por células idênticas interligadas segundo um padrão regular, que se traduz no facto de o número de vectores a aplicar para o seu teste pseudo-exhaustivo ser independente do número de células da matriz [Abramovici et al., 90].

PLA seja simplesmente uma colecção de múltiplas redes de portas lógicas ‘E’-‘OU’, a sua estrutura conduz não só a faltas do tipo sempre-a ou ponte, encontradas nos circuitos combinatórios, mas também a um outro tipo de faltas, denominadas faltas de contacto, entendidas como a presença ou ausência espúria de um contacto (isto é, de um fusível entre uma linha e uma coluna da PLA). A eficácia da utilização de testes desenvolvidos para a detecção de faltas do tipo sempre-a numa PLA é demonstrada em [Agarwal, 80]. A maioria das faltas de contacto múltiplas, bem como faltas sempre-a múltiplas, faltas do tipo curto e ponte<sup>14</sup>, são detectadas por esses testes [Yamada et al., 84]. Apesar disso, a utilização de modelos de faltas criados inicialmente para o teste de circuitos não programáveis revelava algumas fraquezas, por apresentarem uma deficiente cobertura dos seus defeitos característicos, já que não tinham sido desenvolvidos tendo por base a estrutura própria dos dispositivos programáveis onde se aplicavam. Partindo de uma abordagem baseada nos princípios da análise de faltas por indução apresentados por [Shen et al., 85], [Lighthart et al., 91] apresentam um modelo de faltas que incorpora quatro classes:

1. faltas sempre-a múltiplas;
2. faltas do tipo ponte múltiplas;
3. faltas de contacto múltiplas;
4. faltas do tipo circuito aberto múltiplas.

Este trabalho demonstra que faltas sempre-a múltiplas são equivalentes<sup>15</sup> a faltas de contacto múltiplas e que faltas em ponte múltiplas são sub-equivalentes a faltas de contacto múltiplas. Consequentemente, o conjunto de vectores gerado para a detecção de faltas de contacto múltiplas é um sub-conjunto do conjunto de vectores gerados para a detecção de faltas em ponte múltiplas. Logo, o conjunto de vectores gerado para a detecção de faltas de contacto múltiplas detecta igualmente faltas sempre-a e em ponte múltiplas. Desta forma, o modelo de faltas para uma PLA fica reduzido a duas classes:

1. faltas de contacto múltiplas;
2. faltas do tipo circuito aberto múltiplas.

A aparente simplificação do problema é enganadora, uma vez que as faltas múltiplas são difíceis de testar com base nos algoritmos convencionais de geração de vectores, em virtude da sua elevada

<sup>14</sup> Entende-se por faltas do tipo curto as faltas originadas por uma ligação indevida entre uma linha lógica e uma linha de alimentação, e por faltas do tipo ponte as faltas originadas por uma ligação indevida entre duas linhas lógicas.

<sup>15</sup> Duas faltas  $f$  e  $g$  são consideradas equivalentes sse o conjunto dos testes gerados para detectar  $f$  também detecta  $g$  e vice-versa [Abramovici et al., 90].

complexidade e exigência em termos computacionais. Para além dessa razão, acresce ainda um conjunto de particularidades que tornam a lógica programável mais difícil de testar face à lógica convencional:

- as PLAs contêm lógica redundante que pode produzir faltas não testáveis;
- as PLAs contêm um número significativo de caminhos reconvergentes<sup>16</sup>, o que dificulta a tarefa de geração dos vectores.

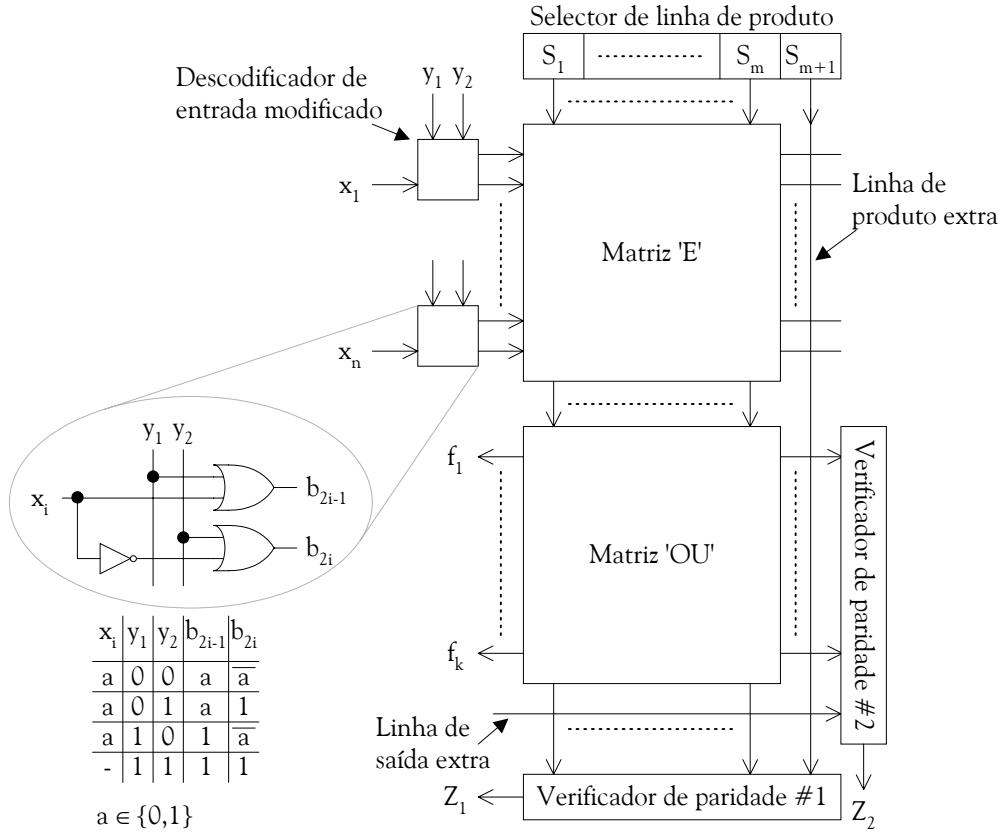
Contudo, a estrutura regular da PLA tende a facilitar a geração do teste, desde que seja incorporado algum esquema de testabilidade. A possibilidade de controlo independente das linhas de entrada e de produto apresenta-se como a chave para a solução do problema. A alteração dos vectores de teste nas entradas implica a mudança dos valores nas linhas de entrada, directa e complementar, e não a sua alteração individual. De forma similar, tal pode ocasionar a mudança de mais que uma linha de produto simultaneamente, acarretando a possibilidade de algumas faltas serem mascaradas<sup>17</sup>, ou que faltas de contacto se tornem indetectáveis [Sachdev, 98].

Uma das primeiras propostas a tornar o problema do teste independente da arquitectura própria da PLA foi apresentada por [Fujiwara et al., 81]. Uma coluna e uma linha são acrescentadas à matriz ‘E’ e à matriz ‘OU’, respectivamente, de forma a que cada linha de entrada na matriz ‘E’ apresente um número ímpar de ligações e que cada linha de produto na matriz ‘OU’ apresente um número par de ligações. Dois verificadores de paridade, baseados em cascatas de portas lógicas ‘OU Exclusivo’, são usados para testar a paridade das duas matrizes durante o teste. Um circuito de selecção é empregue para activar as colunas e as linhas da PLA, de modo a que cada ligação programável seja seleccionada e testada individualmente. Cada linha de produto é seleccionada pela adição de um registo de deslocamento  $S$ , conforme ilustrado na Figura 3.9. Cada bit  $S_i$  de  $S$  é ligado à linha de produto  $P_i$  para criar uma nova linha de produto  $P'_i = P_i S_i$ . Logo, se  $S_i = 1$  e  $S_j = 0$  para todos os  $j \neq i$ , a linha de produto  $P'_i$  é seleccionada e activada até à saída, uma vez que todas as outras linhas de produto são 0. As linhas de entrada são seleccionadas através da modificação do descodificador de entrada, conforme também se ilustra na Figura 3.9. Dependendo dos valores de  $y_1$ ,  $y_2$  e dos  $x_i$ s, todas as linhas menos uma são colocadas a 0, testando-se essa linha. Em operação normal,  $y_1 = y_2 = 0$ .

---

<sup>16</sup> Diferentes caminhos de um mesmo sinal que convergem para uma mesma primitiva.

<sup>17</sup> A falta  $f$  mascara a falta  $g$  se a falta múltipla  $\{f, g\}$  não é detectável pelo vector de teste da falta  $g$  [Abramovici et al., 90].



**Figura 3.9:** PLA com incorporação de um esquema de testabilidade [Fujiwara et al., 81]

Com base na estrutura alterada desta PLA, é possível a geração de um conjunto de vectores de teste universal, caracterizado por:

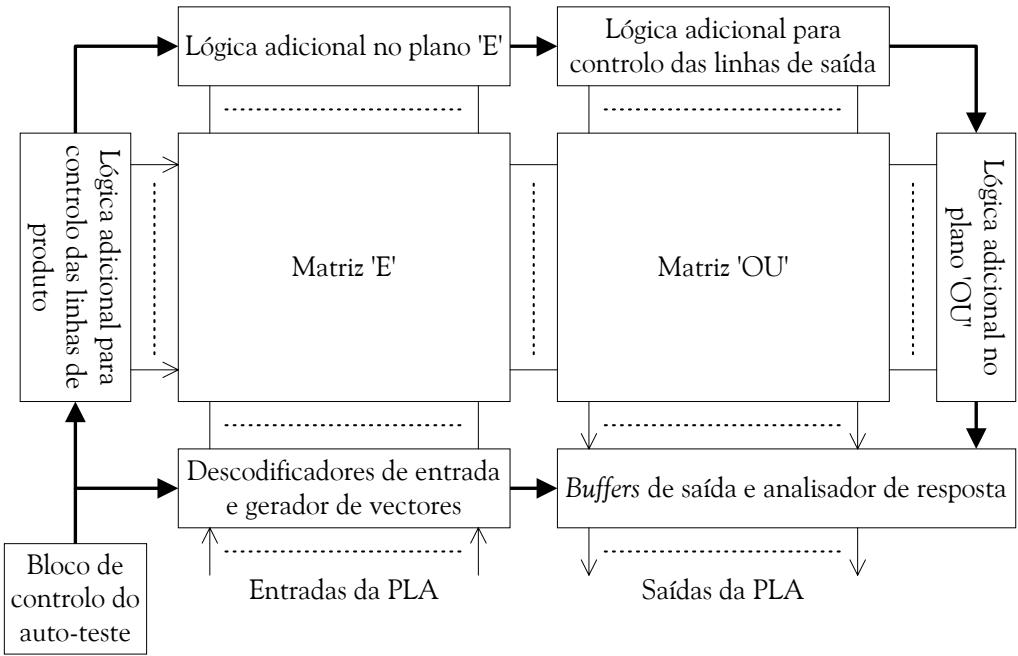
1. os vectores de teste e as respostas não dependerem da função implementada na PLA mas somente do número de entradas  $n$  e do número de colunas  $m$  (termos de produto);
2. o número de vectores de teste ser igual a  $n+m$ .

Durante o teste, cada linha de produto e de entrada é activada individualmente, sendo detectadas pelos verificadores de paridade todas as faltas simples sempre-a e de contacto. Os vectores a aplicar são guardados ou gerados externamente e os resultados são observados durante o processo de teste.

A incorporação de um esquema de testabilidade como o referido, ao tornar o teste independente da função implementada na PLA, possibilitou a inclusão de estruturas de auto-teste como a representada na Figura 3.10 [Fujiwara, 84], [Treuer et al., 85], e [Agrawal et al., 93a]. O aparecimento de dispositivos baseados em memória estática levou à consideração adicional de um modelo de faltas ao nível do transístor para lidar com faltas de transição [Liu et al., 88]<sup>18</sup>.

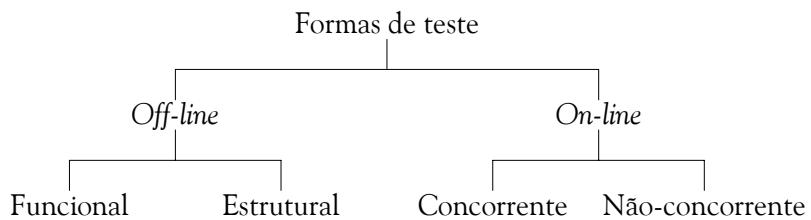
---

<sup>18</sup> Para um estudo mais aprofundado sobre a evolução do teste nas PLAs, sugere-se a consulta de [Abramovici et al., 90] e [Sachdev, 98].



**Figura 3.10:** Arquitectura genérica de auto-teste para uma PLA [Agrawal et al., 93a]

O estudo de estratégias de teste adaptadas às características próprias das FPGAs reconfiguráveis assentou inicialmente sobre estes trabalhos. Esse estudo seguiu diversas direcções, que podem ser classificadas segundo o esquema ilustrado na Figura 3.11, embora, como se verá mais adiante, a categorização de algumas dessas estratégias se apresente dúbia e dependente do ponto de vista da análise, ou consista em estratégias mistas para detecção e diagnóstico.



**Figura 3.11:** Formas de teste [Abramovici et al., 90]

Devido à sua complexidade, a totalidade das abordagens ao problema do teste das FPGAs segue uma estratégia de “dividir para reinar”. Para efeitos de teste, a FPGA é dividida nas suas componentes básicas: blocos lógicos, blocos de E/S, recursos de interligação e memória de configuração.

Com base num modelo de faltas híbrido (funcional/sempre-a), é apresentado em [Huang et al., 96] e [Huang et al., 96a] um teste off-line estrutural, restrito aos blocos lógicos baseados em tabelas de consulta, consistindo na aplicação externa de vectores segundo o esquema ilustrado na Figura 3.8.

Tirando partido da regularidade da matriz de blocos lógicos, esta abordagem segue uma estratégia de teste dividida em duas partes:

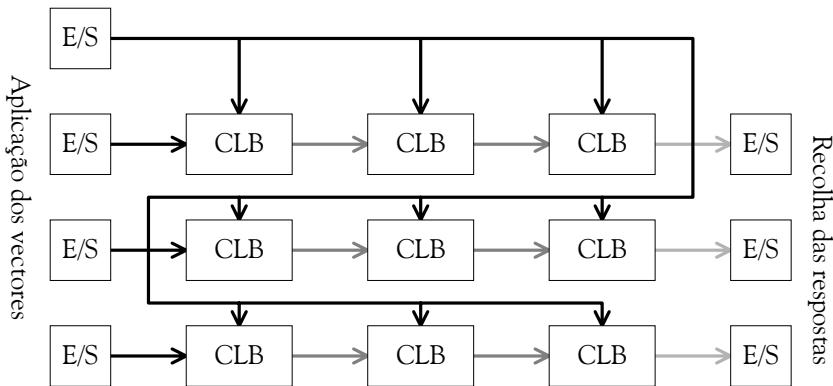
1. para o teste da parte combinatória de cada bloco lógico, situado ao longo de uma linha da FPGA, é configurada uma matriz C-testável unidimensional;
2. para o teste da parte sequencial, é igualmente configurada uma matriz unidimensional, colocando os *flip-flops* em cascata.

Sendo todas as linhas da FPGA iguais, o seu teste disjunto simultâneo conduz ao teste da totalidade do componente. De notar que o tempo de teste é independente do tamanho das matrizes configuradas, no caso da parte combinatória, e dependente do número de blocos numa linha, no caso da parte sequencial, onde o tempo de deslocamento dos vectores depende do comprimento da cadeia de *flip-flops*. Dada a estrutura configurável do bloco lógico, para se alcançar uma cobertura de faltas de 100% sob o modelo considerado são necessárias oito configurações diferentes para o teste (normalmente referidas como *configurações de teste*) da parte combinatória e treze para o da parte sequencial. Este conceito é mais tarde desenvolvido em [Huang et al., 98] e aplicado ao teste de dispositivos lógicos programáveis uma única vez, em [Liu et al., 00]. Abordagem idêntica, com ênfase no problema do teste das tabelas de consulta e da parte combinatória dos blocos lógicos, é apresentada em [Inoue et al., 95], [Inoue et al., 97] e [Inoue et al., 98].

Uma proposta similar, mas que anula a dependência do tempo de teste face ao comprimento da cadeia, aplicando as saídas dependentes do sinal de relógio do CLB precedente às entradas combinatórias do seguinte e vice-versa, é apresentada em [Metra et al., 98].

Com base nos mesmos conceitos, mas diferindo na forma como os blocos são interligados, é proposta em [Renovell et al., 97] e [Renovell et al., 98a] uma solução baseada no emprego de uma matriz unidimensional de blocos lógicos em cascata, como se ilustra na Figura 3.12, cujo comprimento está apenas dependente do número de blocos de E/S do componente (note-se que a interligação dos blocos lógicos em linhas, exposta nas abordagens anteriores, pode inviabilizar o teste simultâneo de todos eles, por insuficiência do número de blocos de E/S disponíveis). O número de configurações proposto para o teste do bloco lógico é oito: quatro configurações implementam na parte combinatória uma cascata ‘OU-Exclusivo’ ou ‘Não-OU-Exclusivo’, com um registo de deslocamento, enquanto as outras quatro, adstritas ao teste da parte sequencial, implementam dois registos de deslocamento paralelos. Esse número foi posteriormente reduzido para cinco, sendo a nova solução validada numa FPGA da família 4000 da Xilinx [Renovell et al., 99]. Mais tarde, os mesmos autores demonstraram, com base nos mesmos princípios, que

bastam quatro configurações para testar um CLB da família Spartan, também da Xilinx [Renovell et al., 99a].



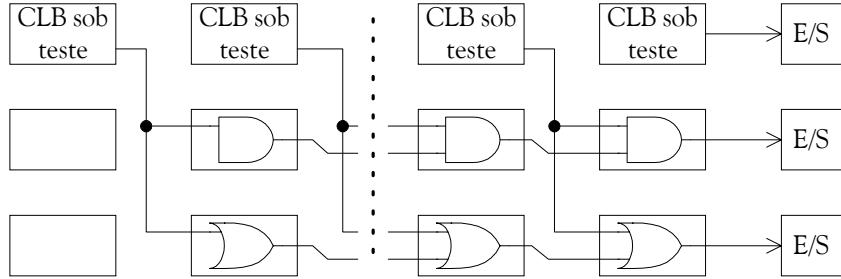
**Figura 3.12:** Matriz de blocos sob teste

Tendo as FPGAs como base uma estrutura configurável, o teste da totalidade dos seus recursos implica a programação de várias configurações. O tempo que demora a programação de uma nova configuração é, em regra, superior ao tempo de aplicação do teste, quer se trate da aplicação externa de vectores, como são os casos anteriormente expostos, quer de auto-teste interno. Daí a preocupação em reduzir ao mínimo indispensável o número de reconfigurações.

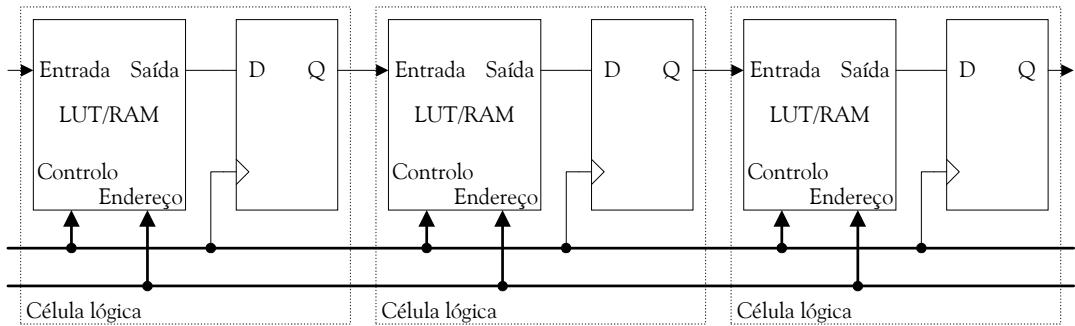
Uma metodologia baseada na utilização de portas ‘E’ e ‘OU’ para compactar as respostas provenientes das saídas dos CLBs, tornando o seu número independente do número de blocos de E/S disponíveis, com a vantagem de detectar múltiplos blocos lógicos faltosos, é apresentada em [Huang et al., 00]. Os vectores são aplicados a partir do exterior, através dos blocos de E/S, em paralelo a um conjunto de CLBs, cujas linhas de saída alimentam simultaneamente as entradas de uma porta lógica ‘E’ e de uma porta lógica ‘OU’, ambas configuradas em blocos lógicos adjacentes, conforme ilustrado na Figura 3.13. A complexidade da geração dos vectores de teste é reduzida, pois os testes são gerados para apenas um CLB, sendo possível o uso de qualquer modelo de faltas. A estrutura de teste é simples e regular e o número de reconfigurações e de vectores de teste é constante, não dependendo do tamanho da FPGA.

As tabelas de consulta dos blocos lógicos, além de permitirem a implementação de uma qualquer função combinatória, podem, em algumas FPGAs, ser configuradas como pequenos blocos de memória distribuída. O teste dos blocos em modo de memória foi objecto de estudo por vários autores. Em [Renovell et al., 98b], é apresentada uma solução envolvendo uma única configuração de teste, independente do tamanho da matriz de blocos lógicos e da tabela de consulta, que implementa um pseudo registo de deslocamento, conforme se ilustra na Figura 3.14. Os testes

aplicados são baseados no clássico zero caminhante e suas variantes, usado no teste de memórias estáticas, e cobrem faltas do tipo sempre-a, de endereçamento e de transição [Mazumder et al., 96].



**Figura 3.13:** Teste de uma FPGA com compactação de respostas por intermédio de portas ‘E’ e ‘OU’



**Figura 3.14:** Teste do modo memória das tabelas de consulta

O teste das interligações tem igualmente merecido atenção especial. A detecção de faltas individuais e o seu diagnóstico são abordados em [Yu et al., 99]. O modelo de faltas adoptado cobre faltas do tipo:

- segmento sempre-a (0 ou 1);
- segmento sempre-aberto;
- comutador sempre-desligado;
- comutador sempre-ligado;
- ponte (entre dois segmentos adjacentes).

São necessárias seis configurações de teste para cobrir a totalidade das interligações, incluindo segmentos e blocos de interligação, e permitir o correcto diagnóstico da falta. De notar que o procedimento de teste é constante, no sentido em que não é necessário o recurso a configurações de teste ou vectores extra para um correcto diagnóstico, no caso da detecção de uma falta, de qual o segmento ou comutador onde esta se verifica. Com base num modelo de faltas idêntico, uma solução empregando apenas três configurações de teste para o teste de interligações e de recursos de encaminhamento globais é proposta em [Renovell et al., 98c]. No entanto, esta solução requer um

elevado número de blocos de E/S, problema ultrapassado pelo encadeamento de vários segmentos sob teste, mas à custa da criação de faltas em ponte redundantes. O teste dos recursos locais de encaminhamento e de interligação aos blocos lógicos é abordado em [Renovell et al., 99b], demonstrando estes autores que o número de configurações de teste depende do número de segmentos que é possível interligar a uma entrada da célula lógica e independente do número possível de ligações que uma saída pode estabelecer.

Todas as estratégias anteriores enquadraram-se, segundo a Figura 3.11, no ramo *off-line* estrutural, embora a consideração de um modelo híbrido funcional/sempre-a no teste dos blocos lógicos possa colocar algumas reservas a esta classificação. Em todas elas, os vectores de teste são gerados tendo em conta a configuração específica de teste implementada e aplicados a partir do exterior, através dos blocos de E/S. Para além da consideração do custo da sua geração, esse facto implica a existência de recursos externos que permitam armazenar esses vectores e analisar as respostas, e limita o seu uso aos testes de fabrico, dada a impossibilidade da sua aplicação ao nível do sistema sem a retirada física do componente e a sua colocação num ambiente de teste adequado. Para obstar a estas questões, vários autores propuseram a utilização de estratégias de auto-teste, a exemplo do ocorrido com o teste das PLAs. A ideia básica em todos eles é a divisão dos recursos configuráveis em três partes:

- geradores de vectores de teste;
- blocos sob teste;
- analisadores de resposta.

O teste da totalidade da FPGA é realizado com base num esquema de rotação da funcionalidade pela totalidade dos recursos lógicos, dividindo-se o processo de teste em cinco fases:

1. reconfiguração do circuito;
2. início do teste;
3. geração dos vectores de teste;
4. análise das respostas para produção de uma indicação do tipo OK/Falha;
5. leitura dos resultados.

Estas cinco fases são repetidas, no mínimo duas vezes, acontecendo que os recursos sob teste durante a primeira etapa trocam, na segunda, de funções com os geradores de vectores e os analisadores de resposta. Uma vez que todos os circuitos sob teste são testados em paralelo, o tempo de teste não depende do tamanho da FPGA. As fases dois e cinco implicam a interacção

com um controlador de teste exterior, sendo viável o recurso ao uso da infra-estrutura de teste BS [IEEE 1149.1, 01] para estabelecer essa comunicação. Evita-se, assim, a ocupação de recursos de E/S e, sobretudo, aproveita-se o facto de ser também possível a reconfiguração da FPGA através do mesmo porto de acesso, para possibilitar o teste no circuito [Gibson et al., 97]. Desta forma, esta metodologia de teste pode ser usada quer nos testes de fabrico, antes ou depois do encapsulamento, quer posteriormente, após a sua montagem numa carta de circuito impresso e incorporação num sistema, possibilitando a detecção e localização de componentes defeituosos. O registo de varrimento pela periferia permite isolar electricamente o componente do resto do circuito, pelo que o teste se processa sem qualquer interferência para o exterior, implicando apenas que o funcionamento normal seja interrompido durante a sua realização. Contudo, os pinos de E/S da FPGA não são testados durante a aplicação do auto-teste, requerendo a realização de testes adicionais durante o teste de fabrico [Gericota et al., 03] ou posteriormente à sua montagem, conjuntamente com o teste das interligações ao nível da carta de circuito impresso, através da infra-estrutura BS [Ferreira, 92].

Dado que a lógica de teste é implementada e existe apenas durante o tempo em que decorre o teste do componente, não há nenhum acréscimo em termos de dispêndio de recursos, como no caso da implementação de estruturas de auto-teste tradicionais, fixas, que implicam um aumento da área de silício entre 10 e 30%. Do mesmo modo, não ocorre degradação no desempenho do circuito, devido à introdução de atrasos de propagação com origem na lógica de teste adicional [Stroud et al., 96a] e [Stroud, 02]. Por outro lado, os recursos exteriores necessários resumem-se ao próprio controlador de reconfiguração e teste, que pode ser um equipamento de teste automático (*Automatic Test Equipment - ATE*) ou um microprocessador do próprio sistema, o qual, em operação normal, controla a reconfiguração da FPGA e, durante o período de teste, fornece as configurações de teste, dá ordem para o seu início e analisa o resultado. Após a conclusão deste processo, o controlador de reconfiguração e teste reconfigura o componente para a sua operação normal.

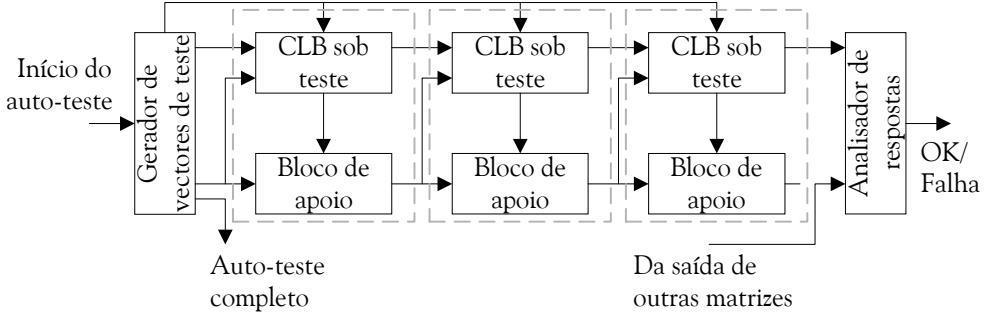
Para suportar a realização das tarefas referidas nas fases três e quatro, alguma da lógica é configurada como geradores de vectores de teste (*Test Pattern Generators – TPGs*) e analisadores de resposta (*Output Response Analysers – ORAs*). A estratégia de teste assenta num teste pseudo-exaustivo [McCluskey, 84], no qual cada subcírcuito do componente é testado exaustivamente sob várias configurações de teste. Isto resulta na maximização da cobertura de faltas sem assunção de um modelo de faltas explícito e sem simulação de faltas, com a vantagem adicional de se efectuar à frequência normal de funcionamento do sistema.

Uma análise das várias configurações propostas para o teste de um bloco lógico e a sua cobertura de faltas, encontrada por comparação com o resultado obtido por simulação de faltas num modelo do bloco ao nível da porta lógica, é apresentada em [Stroud et al., 96b]. O método mais simples de análise da resposta dos circuitos sob teste nas implementações de auto-teste é a comparação da resposta obtida com a resposta esperada, o que implica, contudo, o dispêndio de recursos no armazenamento desta resposta. No entanto, nas FPGAs, os circuitos sob teste são idênticos, sendo apenas necessário que os analisadores de resposta comparem as suas saídas. Ao contrário do que se passa nos analisadores baseados na compressão da resposta do circuito, de uso comum nas aplicações de auto-teste, os analisadores baseados em comparação não sofrem de problemas de *aliasing*<sup>19</sup>, desde que os circuitos sob teste, cujas saídas são comparadas no mesmo analisador, não manifestem a mesma falha ao mesmo tempo. Não obstante, se o gerador de vectores de teste que alimenta esses circuitos exibir uma falha, os testes são incompletos e qualquer circuito com falhas pode escapar à detecção. Para evitar essa situação, os circuitos sob teste cuja saída seja comparada por um mesmo analisador de resposta devem ser alimentados por geradores de vectores de teste diferentes, sincronizados entre si [Stroud et al., 96] e [Stroud et al., 96a]. Como esta solução implica uma sobrecarga adicional sobre os recursos de encaminhamento e originou inicialmente problemas de implementação em FPGAs maiores, [Stroud et al., 96c] propõem o uso de matrizes C-testáveis unidimensionais, a exemplo de [Huang et al., 96a], mas com os vectores de teste a serem obtidos a partir dos TPGs e não externamente. O facto de, numa estrutura regular como esta, a comunicação entre blocos lógicos ser maioritariamente local, entre blocos adjacentes, com as respostas de um bloco lógico sob teste a serem usadas como estímulos de entrada no bloco ou blocos adjacentes, diminui significativamente a pressão sobre os recursos de encaminhamento. Alguns problemas, como o número de sinais à saída de um bloco lógico ser inferior ao número requerido para atacar todas as entradas do bloco lógico seguinte e haver diminuição da observabilidade desses sinais, são resolvidos pela inclusão de um bloco local de apoio, implementado num CLB adjacente, resultando numa estrutura de auto-teste que se ilustra na Figura 3.15.

Um problema potencial desta abordagem é o facto de o atraso total de propagação na matriz poder ser superior ao período do relógio do sistema, o que obriga à divisão da frequência, efectuando-se o teste a uma frequência inferior à da sua operação normal. Outra questão é a necessidade de três fases de teste, em vez das duas da abordagem anterior, o que provoca um aumento de 50% no tempo de teste.

---

<sup>19</sup> Fenómeno que ocorre quando o resultado da compressão da resposta de um circuito faltoso é igual ao do circuito sem faltas [McCluskey, 85].



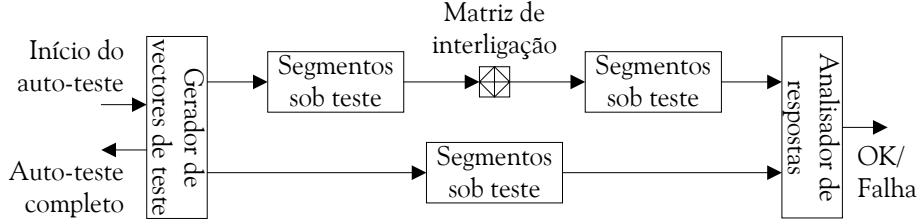
**Figura 3.15:** Estrutura de auto-teste baseada em matrizes C-testáveis

Uma solução híbrida, resultante das duas anteriores, em que à custa de uma menor utilização dos recursos globais de encaminhamento se poupa uma sessão de teste, é apresentada em [Abramovici et al., 97], [Abramovici et al., 97a] e [Stroud et al., 97]. Esta nova arquitectura, ilustrada na Figura 3.7, tornou-se efectivamente na arquitectura padrão para o auto-teste interno das FPGAs, possibilitando, para além da detecção, o diagnóstico das faltas.

A resolução num procedimento de detecção e diagnóstico de faltas depende dos objectivos que se pretendem alcançar com o teste. Num teste ao nível do sistema, o objectivo é localizar o componente defeituoso e substituí-lo. No entanto, num ambiente onde a sua reparação por substituição não é exequível, pode tirar-se partido da estrutura regular e da reconfigurabilidade da FPGA para se implementar um modo de tolerância a faltas, reparando-o no circuito [Hanchek et al., 98]. Este processo requer a identificação do CLB faltoso e a relocação da sua funcionalidade num CLB livre. Esta identificação obriga a que, após a primeira fase do teste, em que se determina a linha de CLBs em que a falta ocorre, se empregue uma segunda fase, em que, em vez da implementação em linha apresentada na Figura 3.7, se passa a uma implementação em coluna, rodando toda a infra-estrutura de teste 90°. Determina-se, assim, pela intersecção linha-coluna faltosas qual o bloco ou blocos lógicos que apresentam faltas [Abramovici et al., 00] e [Abramovici et al., 01]. A reconfiguração da FPGA, o controlo da aplicação do auto-teste e a leitura dos resultados são efectuados nestas propostas através da infra-estrutura BS, implicando um dispêndio nulo de recursos físicos ao nível do sistema.

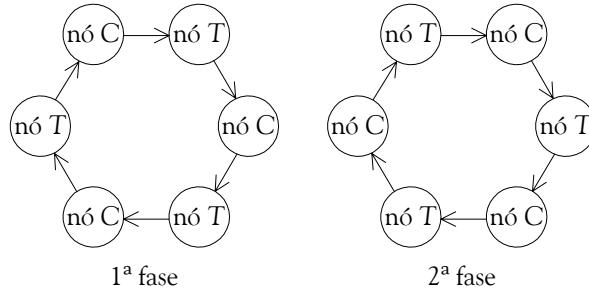
O mesmo conceito é reutilizado para o teste dos recursos de interligação e encaminhamento da FPGA. Após o teste dos blocos lógicos, alguns dos blocos são configurados como geradores de vectores e outros, como analisadores de resposta. Um subconjunto dos recursos de interligação e encaminhamento é configurado em dois grupos de segmentos sob teste, que recebem os mesmos vectores de teste do TPG, sendo os seus resultados comparados num ORA, conforme ilustrado na Figura 3.16. O auto-teste consiste numa sequência de fases que passa pela reconfiguração da FPGA e pela geração dos vectores de teste e análise das respostas, que é repetida até todos os recursos de

interligação e encaminhamento estarem testados. Como exemplo, o teste de uma FPGA da família ORCA (Optimized Reconfigurable Cell Array) 2C da Lattice [Lattice, 02] obriga ao uso de 19 configurações de teste [Abramovici et al., 98] e [Stroud et al., 98].



**Figura 3.16:** Estrutura de auto-teste dos recursos de interligação e encaminhamento

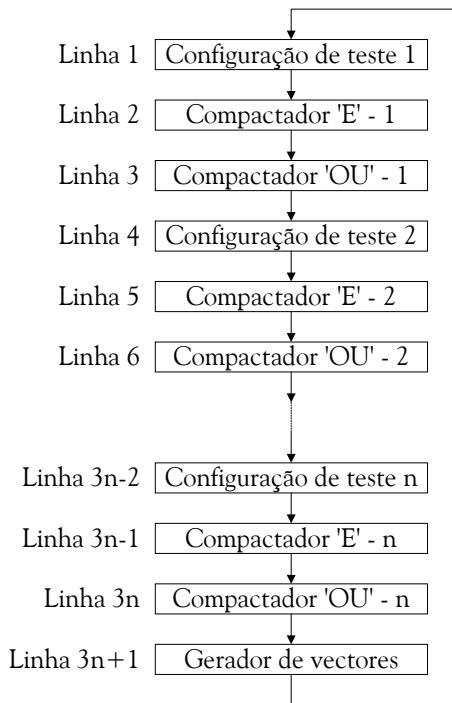
Uma outra solução, tendo por base a proposta apresentada em [Stroud et al., 96a], é exposta em [Wang et al., 99]. No sentido de evitar os problemas decorrentes de uma sobrecarga nos recursos de encaminhamento globais, o espaço lógico de configuração da FPGA é dividido em áreas iguais, designadas por nós, as quais acomodam um gerador de teste e um analisador de resposta (nós T), ou vários CLBs sob teste (nós C). O processo de teste decorre em duas fases, na primeira das quais metade dos nós é configurada como T e outra metade como C, trocando, na segunda, de funções. O procedimento de teste pode ser representado por uma figura regular, cujos vértices são constituídos pelos nós e as arestas pelas acções de teste, conforme ilustrado na Figura 3.17. Um nó testa na primeira fase o imediatamente subsequente e, na segunda, é testado pelo seu precedente. No final, é possível identificar quais os nós faltosos. Numa segunda etapa, um dos nós bons actua como T, enquanto outros dois, um bom e um faltoso, actuam como C. Da comparação das saídas dos respectivos CLBs de cada nó, é possível determinar quais os CLBs faltosos. Este método, tal como o método anterior, é um método adaptativo, no sentido em que, no caso da detecção de uma falta, é requerida uma 2<sup>a</sup> fase de testes para o seu diagnóstico.



**Figura 3.17:** Sequênciа de teste dos nós

Partindo do princípio de que a parte da memória de configuração da FPGA que define a funcionalidade das linhas de CLBs e as suas interligações é homogénea, é proposta em [Doumar et al., 99] e [Doumar et al., 99a] a implementação de uma estratégia de auto-teste que cobre

automaticamente todos os blocos lógicos. Inicialmente, os blocos de uma linha são configurados como geradores de vectores de teste, os da linha seguinte, como blocos sob teste, com uma determinada configuração, e os das duas linhas subsequentes, como compactadores de resposta ‘E’-‘OU’ (ver Figura 3.13). A sequência linha de blocos sob teste e compactadores de resposta repete-se, com uma configuração de teste diferente, até todas as linhas estarem ocupadas e todas as configurações necessárias para se alcançar a desejada cobertura de faltas terem sido implementadas. No final da primeira fase do teste, a configuração da FPGA é deslocada uma linha, repetindo-se este deslocamento até todas as linhas terem sido configuradas com todas as diferentes configurações de teste. Deste modo, apenas é necessário armazenar uma configuração de teste que é deslocada, em série, ao longo da memória de configuração, durante o decorrer do próprio teste. A estrutura básica desta abordagem é ilustrada na Figura 3.18. Porém, esta solução implica alterações a nível da arquitectura da memória de configuração, para que a parte respeitante à configuração dos blocos lógicos e suas interligações seja totalmente homogénea e para que seja possível a colocação em série, durante o teste, de todas as colunas de configuração, criando um anel de deslocamento.



**Figura 3.18:** Estratégia de auto-teste em anel

A maioria das FPGAs possui blocos de memória estática (SRAM) cujo teste exaustivo, utilizando uma estratégia de auto-teste, é impraticável, devendo ser usadas sequências de vectores de teste exaustivos para os modelos de faltas específicos das memórias [Mazumder et al., 96]. O teste da memória de configuração apresenta-se bastante mais problemático, uma vez que a escrita e a leitura de dados nas suas células não pode ser isolada da configuração da própria FPGA. Embora o uso de

várias configurações de teste no teste dos blocos de memória e dos recursos de interligação e encaminhamento garanta, logo à partida, o teste conjunto de um elevado número de bits dessa memória, bem como da sua interface, a maioria dos autores sugere que, após a configuração total ou parcial de uma FPGA, se proceda à sua leitura completa e à comparação do resultado com o ficheiro de configuração, como forma de garantir a não existência de faltas que afectem essa configuração.

Todas as estratégias apresentadas, baseadas na aplicação de vectores a partir do exterior ou em técnicas de auto-teste, têm como alvo a totalidade dos recursos da FPGA, sendo para tal, dada a estrutura configurável destes componentes, empregues várias configurações de teste, que cobrem todas as diferentes possibilidades de configuração. Embora tal se revele interessante como garante do correcto funcionamento da FPGA, nos casos em que a sua reconfiguração ocorre várias vezes ao longo da sua operação normal, naquelas aplicações em que a configuração é fixa, é mais económico testar apenas os recursos que são efectivamente usados. Neste sentido, vários autores propuseram estratégias de teste orientadas à aplicação, baseadas igualmente em estruturas de auto-teste ou na aplicação externa de vectores.

A geração de vectores de teste para uma aplicação implementada num dispositivo lógico programável tem obrigatoriamente que ter em atenção a estrutura do próprio dispositivo. Vectores de teste gerados tendo como base apenas a descrição do circuito e os modelos de faltas a ela adequados apresentam uma baixa cobertura de faltas, por não terem em conta nenhuma informação estrutural da sua real implementação física. Só após o mapeamento do circuito é possível criar a sua descrição lógica, incluindo tabelas de consulta, multiplexadores e *flip-flops*, para a qual devem ser gerados os vectores de teste. Contudo, muita da lógica associada à descrição não é efectivamente usada pela aplicação, como é o caso dos multiplexadores programáveis, em que apenas uma das entradas é usada, traduzindo-se efectivamente numa ligação fixa. No entanto, como o multiplexador faz parte da descrição, uma ferramenta automática de geração de vectores produzirá vectores para o teste das suas entradas e saída, tendo como resultado o teste de um número elevado de faltas redundantes para a aplicação e um grande dispêndio de tempo de processamento. Se a lógica redundante for identificada e ignorada, em termos de cada aplicação concreta, obtém-se uma versão reduzida da implementação, que, sem deixar de ter em conta a estrutura do dispositivo, garante uma elevada cobertura de faltas [Renovell et al., 00] e [Renovell et al., 00a].

O uso de procedimentos de auto-teste como parte de uma estratégia orientada à aplicação esbarra na necessidade da existência de recursos, da FPGA ou externos, para a sua implementação. No sentido de obstar a esse facto, é proposta em [Merino et al., 00] uma estratégia que se baseia na divisão da FPGA em duas áreas, para fins de implementação das aplicações. Aproveitando a

possibilidade de reconfiguração oferecida por estes componentes, uma das áreas é, numa primeira etapa, configurada com a lógica de auto-teste (um gerador de vectores de teste e um analisador de resposta), que testa as aplicações implementadas na outra metade. Numa segunda etapa, os papéis invertem-se. Este método obriga à partição optimizada dos recursos pelas várias aplicações implementadas na FPGA, o que, no caso de se ter uma única aplicação ou uma aplicação dominante em termos do espaço ocupado (superior a 50%), inviabiliza a aplicação do método. Por outro lado, a implementação dos circuitos de auto-teste é efectuada sobre recursos que, em condições normais de funcionamento, podem não ser usados pela aplicação aí implementada. Como a estratégia de teste é orientada à aplicação, a parte dos recursos por ela não usada nunca é testada, não sendo pois possível garantir o seu bom funcionamento, e, consequentemente, o da lógica de auto-teste que os possa vir a usar, o que pode levar a conclusões erradas.

O conceito de auto-teste orientado à aplicação foi igualmente aproveitado para o desenvolvimento de estratégias de teste tendo como alvo a detecção de faltas de atraso em aplicações específicas. A ideia base é que, para o projectista, o que importa não é se o auto-teste dos recursos da FPGA à sua máxima frequência de operação não apresentou problemas, mas sim se a sua aplicação concreta opera correctamente à frequência de relógio pretendida. A lógica de auto-teste é implementada temporariamente sem alterar a configuração das aplicações do utilizador, quer por partição destas, no caso de ocuparem parte significativa do espaço lógico de configuração, quer por troca com outras aplicações, no caso de o espaço ser partilhado, efectuando-se o teste das diferentes partições ou aplicações de forma faseada [Krasniewski, 99] e [Krasniewski, 00]. Como os tempos de transição de um sinal são diferentes caso a transição se efectue no sentido descendente ou ascendente, a determinação do atraso máximo de propagação de um sinal que atravessa um bloco de lógica combinatória, implementado através de uma cascata de tabelas de consulta, deve ter em conta a polaridade das transições que ocorrem ao longo desse bloco aquando da aplicação de um determinado vector de teste. De notar que a polaridade da transição à entrada do bloco não determina a polaridade das transições nas interligações – por exemplo, uma transição ascendente numa das entradas de uma tabela de consulta que implemente uma função ‘OU-Exclusivo’ pode produzir uma transição ascendente ou descendente na saída da tabela, dependendo do valor das restantes entradas. Um método para melhorar a detectabilidade de faltas de atraso em tabelas de consulta é apresentado em [Krasniewski, 01].

Um dos problemas apontados às soluções de teste *off-line* é o tempo excessivo de latência de uma falta, entendido como o intervalo máximo em que uma falta emergente pode manter-se activa (podendo perturbar ou mesmo bloquear o funcionamento do sistema), até ser detectada por um procedimento de teste. Por implicar a paragem do sistema, a aplicação dos testes tende a ser muito

espaçada no tempo, sob pena de uma diminuição incomportável na disponibilidade<sup>20</sup> do sistema. Contudo, em sistemas críticos, quer a paragem do sistema para teste, quer um elevado tempo de latência na detecção de faltas, não são admissíveis. A solução é a realização de testes *on-line* concorrentes, nos quais o processo de teste se desenrola sem que a operação do sistema seja perturbada.

Tipicamente, o teste *on-line* concorrente implica a utilização de circuitos auto-verificáveis, com capacidade para aferirem automaticamente da existência de alguma falta sem necessidade de aplicação externa de estímulos, permitindo a sua detecção durante a operação normal do circuito. Contudo, esta abordagem implica a inserção de lógica redundante, envolvendo a duplicação ou triplicação de circuitos e a utilização de blocos votadores, ou o recurso a códigos de detecção e correcção de erros, e, consequentemente, à inserção de toda a lógica associada à sua implementação. Em ambos os casos, há um dispendioso aumento dos recursos de hardware necessários [Santos, 99] e [Lala, 01].

Várias propostas que apontam para a inclusão automática de lógica auto-verificável durante a síntese do circuito foram apresentadas. Uma destas propostas [Burress et al, 97] recorre à utilização de um conjunto único de células com saídas complementares, que, em caso de erro, produzem valores idênticos na saída. As células são interligadas em cascata e combinadas numa célula final, cujas saídas se ligam a um verificador de saídas complementares (*two-rail checker*)<sup>21</sup>. Esta técnica é capaz de detectar faltas dentro de um CLB ou nas interligações, com a vantagem de permitir não só a detecção de faltas permanentes, mas também transitórias. Partindo do mesmo pressuposto, [Mitra et al., 98] propõem uma técnica para a detecção de faltas e o seu posterior diagnóstico com recurso a auto-teste *off-line*. Uma técnica também baseada na utilização de verificadores de saída complementares, adaptada à detecção de faltas transitórias e de *crosstalk*<sup>22</sup> afectando as interligações da FPGA, é apresentada em [Metra et al., 01].

De notar que só a lógica usada pelos circuitos implementados é testada, o que, em componentes reconfiguráveis, não é suficiente para garantir que, após uma reconfiguração, o circuito configurado opere sem falhas, uma vez que este pode ser implementado sobre recursos lógicos previamente não usados (e portanto não testados). Embora, *a priori*, todos os circuitos a serem configurados num

<sup>20</sup> Probabilidade de um sistema funcionar correctamente e de estar disponível para realizar uma lista de funções num dado instante  $t$  [Laplante, 98].

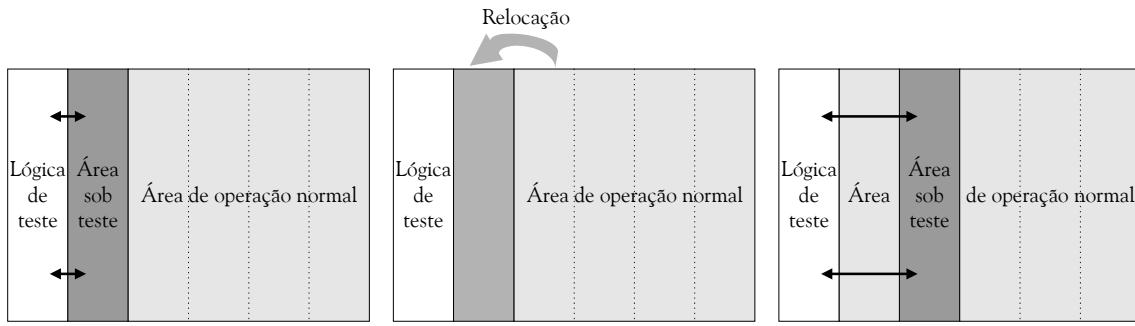
<sup>21</sup> Circuito com dois grupos de entrada ( $X$  e  $Y$ ) e duas saídas ( $f$  e  $g$ ), em que os sinais observados na saída devem sempre ser complementares sse para cada par  $(x_i, y_i)$ ,  $y_i = \bar{x}_i$  [Lala, 01].

<sup>22</sup> Interferência capacitiva entre duas linhas de transmissão paralelas, nas quais um sinal numa linha afecta o sinal na outra [Laplante, 98].

dispositivo lógico programável devam ser auto-verificáveis, só após a configuração se sabe se o circuito trabalha correctamente.

A capacidade de reconfiguração parcial das FPGAs abriu novas perspectivas de abordagem a este problema, posteriormente reforçadas com a possibilidade de essa reconfiguração se poder efectuar de forma dinâmica. Tirando partido da não ocupação total do espaço lógico configurável, que tipicamente se observa nos sistemas baseados em FPGAs, é possível proceder nessas zonas não ocupadas à configuração de esquemas de teste que vão testando estruturalmente os recursos livres, à medida que os restantes circuitos em uso pelo sistema mantêm o seu funcionamento normal. Por transferência desses circuitos entre as áreas ocupadas e as áreas livres, é possível o teste da totalidade dos recursos da FPGA. O conceito de base consiste em ter somente uma porção relativamente pequena dos recursos da FPGA *off-line* e sob teste, enquanto o resto do dispositivo continua *on-line*, operando normalmente. Ao nível do sistema, que é o que interessa neste caso, haverá um teste *on-line* concorrente, visto que o processo de teste detecta faltas durante a operação normal, embora, ao nível dos recursos lógicos internos da FPGA, se possa considerar que se trata de um teste *off-line* estrutural. Desta forma, ficam resolvidos os dois problemas identificados na implementação de estratégias de teste *on-line* em FPGAs, com recurso a sistemas auto-verificáveis – o dispêndio exagerado de recursos e a impossibilidade de detecção antecipada de defeitos.

Aparecendo como um trabalho pioneiro na área do auto-teste *on-line* para FPGAs é apresentada em [Shnidman et al., 97] e [Shnidman et al., 98] uma proposta que, embora conceptual, serviu de base a alguns trabalhos inovadores que mais tarde se afirmaram nesta área. O objectivo é a detecção de faltas numa FPGA, sem utilização de hardware de teste externo ao componente e de forma não intrusiva. A ideia subjacente a esta proposta é a utilização dos próprios recursos internos da FPGA para implementação de uma estrutura de auto-teste que, ao invés das propostas anteriores para o teste *off-line*, testa apenas uma pequena porção dos recursos configuráveis de cada vez, não interferindo com a operação normal do sistema. A alocação de demasiados recursos reduziria a funcionalidade da FPGA, enquanto a alocação de uma parte muito reduzida teria como resultado um teste muito lento, pelo que é assumida a coluna de blocos lógicos como unidade de teste. Desta forma, a estratégia proposta ocupa duas colunas de CLBs numa FPGA, uma que se encontra sob teste a cada instante e outra para implementação da lógica de teste. Através de uma estratégia de varrimento, diferentes partes dos circuitos configurados numa das colunas da FPGA são movidos para a coluna recentemente testada, permitindo assim a sua libertação para o teste, conforme ilustrado na Figura 3.19.



**Figura 3.19:** Metodologia de implementação de auto-teste *on-line*

Assim, ao testar a nível estrutural a FPGA, minimiza-se a probabilidade de ocorrência de uma falha, qualquer que seja o circuito que posteriormente nela seja reconfigurado. A implementação desta estratégia implica consideráveis alterações na arquitectura dos blocos lógicos, com a inclusão em cada bloco de uma memória com a respectiva configuração da tabela de consulta e um duplicado do *flip-flop*. Na prática, duplica-se a memória de configuração da própria FPGA, servindo o conteúdo dessa memória de termo comparativo com o original durante o teste de cada CLB da coluna. Esta duplicação permite ainda que os dados de configuração de uma célula específica sejam acedidos sem que a operação da célula seja afectada<sup>23</sup>. Durante a operação normal, todas as operações de escrita no *flip-flop* são também efectuadas no *flip-flop* de configuração.

Para além destas alterações, o principal problema que se coloca à implementação prática desta proposta é a relocação dos circuitos activos, entendidos como aqueles que estão efectivamente a ser utilizados pelo sistema, nas áreas previamente testadas, sem que a sua funcionalidade seja afectada durante essa relocação. A implementação da estratégia segue um algoritmo que comprehende três fases distintas: cópia, teste e deslocamento. Na primeira fase, é criado um duplicado funcional da próxima coluna a ser testada, com a transferência dos dados de configuração, das tabelas de consulta e dos *flip-flops* a ser efectuada através de um barramento de dados de configuração. Após a cópia, os sinais de entrada e de saída da coluna relocada são comutados para a nova coluna. Esta operação é relativamente simples em FPGAs com arquitectura assimétrica ou em linha, bastando a ligação de cada entrada e saída dos CLBs da nova coluna às correspondentes linhas do barramento onde as entradas e saídas dos CLBs copiados se encontram ligadas. A mesma operação é, no entanto, bastante mais complexa em arquitecturas simétricas ou em ilha, onde é necessário proceder ao reencaminhamento, um a um, dos sinais. No passo seguinte, os valores de estado presentes nos *flip-flops* da coluna a testar são enviados através do barramento de dados de

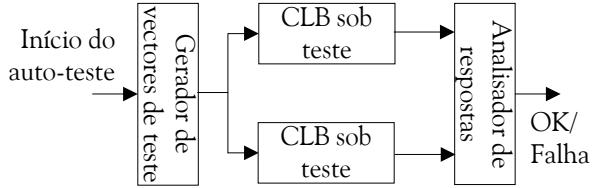
<sup>23</sup> As FPGAs mais recentes permitem a leitura da sua configuração directamente a partir da memória de configuração, sem que isso afecte o seu funcionamento, pelo que esta proposta de duplicação, tendo em mente esse objectivo, deixou de fazer sentido.

configuração e escritos nos *flip-flops* da nova coluna. Este passo, aparentemente repetido, justifica-se por poder existir uma alteração do valor de estado durante o intervalo que decorre entre a cópia da configuração e a colocação em paralelo das entradas. Se existir uma tentativa de escrita enquanto os valores de estado estão a ser copiados, a operação de escrita tem prioridade sobre a de cópia. Uma vez que ambos os *flip-flops* têm as mesmas entradas, ambos são actualizados com o mesmo valor. Esta regra garante a coerência dos dados entre os dois *flip-flops*, o original e a cópia. Ambas as colunas se mantêm activas com as mesmas entradas, saídas e funcionalidade, durante pelo menos um ciclo de relógio, para evitar a ocorrência de transitórios na saída. As saídas da coluna copiada são então libertadas.

A fase seguinte consiste na realização do teste, com a interligação das entradas da coluna sob teste à saída de um contador que controla igualmente a memória presente em cada CLB. As saídas das tabelas de consulta são comparadas com os valores presentes na memória, o mesmo ocorrendo com o *flip-flop* e o seu duplicado no bloco lógico, em busca de diferenças que indiciem a presença de faltas, sendo o processo repetido para a configuração inversa da tabela de consulta e do valor de estado guardado no *flip-flop*. Desta forma, tabela de consulta e *flip-flops* são testados exaustivamente contra faltas do tipo sempre-a e a possível ocorrência de SEUs. A última fase envolve de novo a transferência do bloco copiado para a sua posição original, repetindo-se basicamente o processo de cópia inicial, mas agora em sentido inverso.

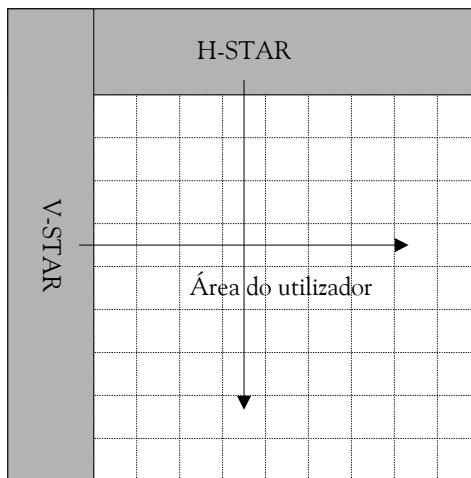
Esta abordagem, ainda que conceptual e incipiente em vários aspectos, alerta para vários problemas ocorrentes na relocação de elementos activos, sem afectar o seu funcionamento. Uma das questões a considerar é o facto de os circuitos-cópia verem as suas entradas e saídas colocadas em paralelo com as dos circuitos originais sem que haja total certeza de que os valores de estado nos *flip-flops* de ambos são iguais.

Apesar disso, a ideia de alojar uma pequena parte da FPGA para realizar o seu auto-teste interno, abrangendo eventualmente, por varrimento e de forma não intrusiva, toda a sua área, serviu de base à proposta, apresentada em [Abramovici et al., 99], de uso de áreas auto-testáveis (*Self-Testing AReas* - STARs). Estas áreas são compostas por unidades mais pequenas, completamente autónomas em termos de teste, designadas por BISTERS e ilustradas na Figura 3.20, que incluem um gerador de vectores de teste, um analisador de resposta por comparação e os próprios blocos lógicos sob teste, a exemplo das estruturas de auto-teste em aplicações de teste *off-line* revisitadas anteriormente. A repetida reconfiguração destas unidades, incluindo o gerador de vectores, quando uma diferente configuração dos blocos sob teste requer diferentes vectores, garante o teste estrutural da totalidade dos blocos lógicos que as constituem, de forma pseudo-exaustiva, em todos os modos de operação.



**Figura 3.20:** Unidade autónoma de teste (BISTER)

Para implementação desta estratégia de teste, o espaço de configuração da FPGA é inicialmente dividido numa área contínua, reservada para as funções do utilizador, e em duas zonas, uma vertical, designada V-STAR, constituída por duas colunas, e a outra horizontal, a H-STAR, constituída por duas linhas. Embora apenas uma destas áreas seja testada de cada vez, as duas são necessárias para permitir o teste dos segmentos verticais e horizontais e das interligações entre eles, facilitando, além disso, o diagnóstico (embora uma fosse suficiente para efectuar o teste da totalidade dos blocos lógicos). O teste da totalidade da FPGA obriga a um deslocamento completo de ambas as STARs, conforme se ilustra na Figura 3.21, o que implica um crescimento linear do tempo de latência das faltas com o aumento do tamanho da FPGA.



**Figura 3.21:** Posição e deslocamento das áreas de auto-teste dentro da FPGA

A implementação das STARs introduz um dispêndio de recursos que pode ser significativo no caso de FPGAs mais pequenas, mas que é atenuado em FPGAs maiores. Por outro lado, o deslocamento das STARs ao longo da FPGA obriga a que as funções implementadas na área do utilizador sejam fraccionadas e que os sinais que interligam as várias fracções as atravessem, introduzindo um atraso adicional que, dependendo do tipo de ligações, dedicadas ou não dedicadas, pode conduzir a um aumento significativo do atraso de propagação e consequente diminuição da frequência máxima de funcionamento. A execução do teste e o movimento das STARs são controlados por um processador exterior, o mesmo que em operação normal controla a reconfiguração da FPGA, através da infra-estrutura BS, evitando a duplicação de recursos de processamento e a ocupação de

pinos de E/S, e a consequente existência de ligações ao nível da carta de circuito impresso, com funcionalidades de teste.

As STARs são formadas por unidades de teste autónomas disjuntas, que executam simultaneamente dentro de cada uma das suas áreas as mesmas funções de teste. Uma vez que cada unidade testa apenas os blocos lógicos sob teste, é necessário reconfigurar várias vezes cada uma delas, por forma a que cada bloco lógico esteja sob teste em pelo menos uma das configurações. Visto que o tempo total de teste é dominado pelos tempos de reconfiguração, o objectivo é a redução do número de blocos lógicos em cada unidade de teste, propondo os autores unidades com seis blocos: três adstritos à geração de vectores, dois sob teste e um para comparação das respostas. Com o objectivo de maximizar a capacidade de detecção de faltas, cada bloco é testado duas vezes, sendo em cada uma comparados os seus resultados com um bloco diferente. A Figura 3.22 ilustra as seis configurações propostas para o auto-teste dos blocos lógicos de cada unidade. No caso de detecção de faltas, são necessárias configurações adicionais para efeitos de diagnóstico.

V   T	T   C	C   T	T   V	V   V	V   V
V   C	V   T	T   V	C   V	T   V	V   T
V   T	V   V	V   V	T   V	C   T	T   C

V - gerador de vectores de teste

C - comparador de respostas

T - bloco sob teste

**Figura 3.22:** Configurações de teste das unidades autónomas

Uma vez testada toda a área abrangida por uma STAR, todas as actividades de teste são interrompidas, sendo esta deslocada para a posição seguinte por relocação da área de implementação das funções do utilizador. Este processo é executado em seis passos [Abramovici et al., 99a]:

1. as funcionalidades implementadas na área a ser abrangida pela STAR são copiadas para a área recém-testada;
2. o relógio do sistema é interrompido;
3. se necessário, procede-se à cópia dos valores de estado presentes nos *flip-flops* dos blocos lógicos copiados;
4. as interligações abrangidas pela relocação das funções são restabelecidas;
5. o relógio do sistema é reiniciado;
6. as unidades autónomas de teste são configuradas na nova área abrangida pela STAR.

Embora as operações de teste estrutural dos blocos lógicos se processem efectivamente *on-line*, a necessidade de relocate a funcionalidade dos blocos da área de trabalho, de modo a, por varrimento, testar-se a totalidade da FPGA, obriga à paragem do sistema, como se vê pelos pontos dois a cinco. Em virtude de as FPGAs comercialmente disponíveis não tratarem os registos como localizações de memória endereçáveis, permitindo de forma directa a sua leitura e escrita, a cópia dos valores de estado das antigas para as novas localizações levanta alguns problemas. Essa cópia obriga ao uso de uma configuração auxiliar que crie um caminho de transferência temporário entre o bloco lógico de origem e o bloco lógico de destino, bem como a aplicação de um impulso de relógio, para que a cópia dos valores se processe. Um procedimento de transferência idêntico é adoptado pelos autores para a transferência de valores entre as células de memória das tabelas de consulta, caso estas estejam a operar em modo memória. Esta última operação, bastante mais complexa, implica a implementação de um controlador de transferência no espaço reservado, mas presentemente não ocupado, da STAR que actualmente não esteja envolvida no processo de deslocamento. Finda a transferência, procede-se ao restabelecimento das interligações da restante lógica, não afectada por este processo de relocação, com os blocos copiados. De notar que, uma vez que o relógio do sistema se encontra neste ponto (quatro) parado, o restabelecimento deve efectuar-se de forma rápida, o que obriga a que a determinação do encaminhamento seja efectuada *a priori* e que todas as configurações necessárias tenham sido previamente computadas e armazenadas numa memória externa (cuja capacidade depende do tamanho da FPGA). Constata-se, assim, que uma parte significativa do processo de reconfiguração é efectuado *off-line*, pelo que o funcionamento do sistema sofrerá interrupções significativas.

Uma das vantagens de o teste se efectuar sobre recursos da FPGA que se encontram *off-line* é que, no caso de detecção de uma falta num deles, não é necessário interromper o funcionamento para se proceder à sua substituição. Refira-se ainda que a realização de um diagnóstico mais preciso da falta, embora mais demorada, não afecta o desempenho do sistema. Por exemplo, no caso de a falta afectar o *flip-flop* de um CLB, a parte combinatória do bloco pode, em caso de necessidade, ser usada para substituir uma parte combinatória defeituosa doutro bloco. Obtém-se, nesta situação, uma degradação mais graciosa do que a que resulta da simples inutilização de todo o bloco, e aumenta-se, em consequência, a vida útil do sistema [Emmert et al., 00]. A degradação do desempenho do sistema, resultante da introdução deste mecanismo de tolerância a faltas, é analisada em [Emmert et al., 00a]. Algumas melhorias foram posteriormente introduzidas, tendo como objectivo a diminuição do número de configurações de teste, sem, contudo, se afectar a capacidade de diagnóstico [Abramovici et al., 00a].

O teste das interligações processa-se de maneira idêntica, substituindo os blocos lógicos sob teste por conjuntos de interligações sob teste, com aplicação restrita à área ocupada a cada momento pela STAR [Stroud et al., 01]. No entanto, há que ter em conta que, não obstante os blocos lógicos dentro da STAR sob teste estarem vazios e, como tal, não requererem o uso de recursos de encaminhamento, muitos segmentos cruzam essa STAR, estabelecendo as ligações entre a lógica presente de um e de outro lado, pelo que nem todos os recursos de interligação dentro dela estão disponíveis para serem testados.

### 3.4. SUMÁRIO

Neste capítulo, abordaram-se as questões relacionadas com o teste de dispositivos lógicos reconfiguráveis, desde os defeitos característicos deste tipo de componentes e da alteração da sua incidência com a evolução tecnológica, passando por uma análise dos modelos de faltas que melhor os representam e que, em resposta a essas alterações, foram sendo propostos, até uma abordagem dos métodos desenvolvidos para o seu teste.

Nesta última parte, foram analisadas em pormenor as últimas soluções propostas para o teste de componentes reconfiguráveis, evidenciando-se as suas vantagens e desvantagens e os problemas que a sua implementação apresenta, de forma a permitir uma identificação dos requisitos que uma proposta alternativa, como a que se formula nesta tese, deve satisfazer.

O próximo capítulo apresentará de forma sistemática esses requisitos, listará os problemas que tais preceitos levantam e proporá uma arquitectura para a solução global do problema.



## **4. UMA METODOLOGIA PARA O TESTE CONCORRENTE VIA RECONFIGURAÇÃO PARCIAL DINÂMICA**



O segundo e terceiro capítulos permitiram enquadrar o leitor no tema da tese, quer na perspectiva do conhecimento dos dispositivos-alvo, as novas FPGAs com possibilidade de reconfiguração parcial dinâmica, quer das questões que se colocam à obtenção de uma elevada disponibilidade dos sistemas nelas baseados.

Partindo de uma perspectiva histórica, o segundo capítulo possibilitou o acompanhamento da evolução dos dispositivos lógicos programáveis e a aquisição de uma maior compreensão dos factores que balizaram o seu aperfeiçoamento até ao estádio actual. Desta forma, pretendeu-se proporcionar uma visão mais intrínseca e, como tal, mais aprofundada da arquitectura das FPGAs com capacidade de reconfiguração parcial dinâmica, em particular do modelo específico usado para validar este trabalho.

O aumento da disponibilidade dos sistemas baseados nestas novas FPGAs obriga à compreensão dos mecanismos que estão na origem da sua degradação. Deles se deu conta no terceiro capítulo, tendo-se ainda assinalado que essa degradação pode dever-se não só a factores endógenos, consequência do próprio desenvolvimento tecnológico com a utilização de escalas submicrométricas e o emprego de mais baixas tensões de alimentação, mas também exógenos, dele consequentes ou não, como um aumento da susceptibilidade das FPGAs à radiação ou ao aparecimento de defeitos físicos quando operadas em ambientes adversos e sujeitas a reconfigurações constantes. Tal implica uma atenção permanente ao seu funcionamento, através da implementação de adequados mecanismos de teste que, embora permitindo supervisionar a sua operação e corrigir eventuais falhas, não perturbem nem deteriorem o desempenho do sistema.

A partir da compreensão desses problemas torna-se possível identificar os requisitos aos quais deverá atender uma proposta de solução que vise incrementar o grau de disponibilidade, permitindo, adicionalmente, compreender as dificuldades e limitações que a própria arquitectura coloca ao seu desenvolvimento.

Algumas soluções propostas anteriormente por diferentes autores, adaptadas de outras já existentes, desenvolvidas sobretudo para o teste de dispositivos não configuráveis, ou de cariz mais inovador, tendo como alvo específico os diversos dispositivos configuráveis, foram identificadas, no sentido de mais facilmente se reconhecer os aspectos menos conseguidos e aqueles cujas potencialidades merecem o seu aproveitamento numa proposta ulterior.

A partir da análise de certa forma multidisciplinar a que se procedeu nos anteriores capítulos tornou-se possível extrair a informação que se sistematiza no início deste e foram criadas as condições para o entendimento dos objectivos que se pretendem atingir com esta proposta, na qual se destacam os pontos distintivos em relação às propostas anteriores, com os quais se procura cobrir

as lacunas e limitações nelas identificadas, e as características-alvo dos sistemas que visa atingir. A proposta é, em seguida, repartida nas suas componentes principais, servindo a sua análise de base à implementação prática, que será objecto de estudo nos capítulos subsequentes.

## 4.1. SOLUÇÃO GLOBAL

A proposta aqui exposta apresenta uma nova metodologia de teste para FPGAs com capacidade de reconfiguração parcial dinâmica, com o objectivo de assegurar o seu funcionamento continuado. Não se pretendendo que o sistema entre em colapso por causa de um problema que possa surgir na FPGA, não seria curial que, para se atingir tal objectivo, se propusesse um método que obrigasse à paragem do sistema para que o teste fosse realizado. Além disso, este teria de ser aplicado tão mais frequentemente quanto maior fosse o grau de fiabilidade pretendida para o sistema, o que na prática implicaria, no limite, uma total indisponibilidade do sistema para a função para a qual tinha sido concebido.

Para ultrapassar essa situação, uma das opções é a implementação de um teste *on-line* concorrente, com recurso a circuitos auto-verificáveis. Este tipo de circuitos possui a capacidade de aferir automaticamente, durante a sua operação normal, se existe alguma falta que afecte a sua lógica interna, sem necessidade de aplicação externa de estímulos de teste. Existem vários métodos para permitir a implementação de auto-verificação, como o uso de códigos de detecção de erros, no âmbito dos quais o mais simples é o cálculo da paridade, ou a utilização de lógica redundante, de que são exemplo os verificadores de saída complementares e a redundância modular tripla (*Triple Modular Redundancy - TMR*) [Lala, 01]. Este tipo de teste apresenta algumas vantagens, uma das quais consiste na detecção do erro no preciso instante em que este afecta o funcionamento do circuito, permitindo a detecção não só de faltas permanentes mas também, em certas circunstâncias, de faltas transitórias. Outra das vantagens é a tolerância a faltas inserida por algumas das estratégias usadas na sua implementação, como os circuitos com TMR ou os códigos de detecção-correcção de erros, o que permite, dentro de certas condições, a continuação do seu funcionamento normal, com a sinalização para o exterior da presença de uma falta que deverá ser reparada antes que outras faltas possam surgir e coloquem em causa a operação do sistema. Em qualquer dos casos, está sempre implícita a utilização de mais recursos do que os estritamente necessários à persecução dos objectivos pretendidos para o circuito, estando esse aumento relacionado com a capacidade de tolerância a faltas do circuito, o que, no caso da TMR, implica o dispêndio de um volume de recursos superior ao triplo do mínimo necessário.

Contudo, o seu uso com lógica reconfigurável acarreta algumas desvantagens. Desde logo, o dispêndio extra de recursos, mas também algumas características do suporte onde é implementado e a forma como este é usado. A utilização de FPGAs como plataformas de computação reconfigurável implica a constante reconfiguração parcial dinâmica de variadas funções, às quais são alocados, para a sua implementação, diferentes recursos em áreas diversas do espaço lógico, de cada

vez que o seu uso é requerido pelas aplicações. O bom estado estrutural dos recursos lógicos que ocupam é, *de per si*, uma garantia do seu próprio bom funcionamento, salvaguardada a possibilidade de ocorrência de erros na transmissão de dados para a memória de configuração, os quais podem ser detectados com recurso a um código de verificação por redundância cíclica (*Cyclic Redundancy Check* – CRC), como é o caso nas famílias Spartan e Virtex [Xilinx, 98] e [Xilinx, 00]. De notar, porém, que qualquer uma das estratégias de teste anteriormente referidas apenas pode assegurar o bom estado estrutural dos recursos ocupados, desde que, em caso de defeito, o seu próprio funcionamento exerce, em algum instante, a falta ou faltas consequentes desse defeito. Isto significa que, mesmo que o circuito considerado ocupe uma área com defeito, este pode não ser detectado. Desta forma, uma nova função, parte de uma dada aplicação, pode ser implantada sobre recursos defeituosos e falhar imediatamente a sua operação, o que, dependendo do tipo de falta e da capacidade de tolerância a faltas implementada, pode implicar a paragem da aplicação.

Outra das questões tem a ver com os diferentes recursos usados para a implementação de lógica redundante dentro de uma FPGA. Os recursos alocados num dado instante para a implementação de uma nova função, embora sejam suficientes para, por exemplo, no caso da redundância modular tripla, permitirem a implantação de três circuitos funcionalmente iguais, não garantem que o tempo de propagação de cada um seja aproximado dos restantes, podendo existir diferenças significativas, facto que deve ser tido em conta no desenho do circuito votador. Obviamente que, por imposição do projectista, essa diferença pode ser limitada a uma janela temporal muito estreita, mas tal implicará o uso de determinados recursos específicos para atingir esse objectivo. Estes podem não estar imediatamente disponíveis quando a execução da função é requerida pela aplicação, atrasando a sua implementação na FPGA e, consequentemente, a própria execução da aplicação.

Em conclusão, a utilização de estratégias de teste *on-line* concorrente clássicas, com a incorporação de lógica de detecção-correcção de erros em todos os circuitos que possam vir a ser implementados na FPGA, para além do dispêndio significativo de recursos que implica, não garante por si só a detecção, *a priori*, de defeitos em sistemas reconfiguráveis.

Uma alternativa para contornar o problema da detecção dos defeitos antes da sua manifestação como falhas nas funções implementadas seria o recurso a estratégias de teste *on-line* não-concorrentes. Segundo [Abramovici et al., 90], o teste *on-line* não-concorrente é efectuado enquanto o sistema está num estado de espera, podendo ocorrer a qualquer instante a retoma do funcionamento normal do circuito, situação em que o processo de teste é imediatamente interrompido. Esta definição, produzida tendo como horizonte os circuitos não configuráveis, aplica-se igualmente, com ligeiras adaptações, à área dos circuitos configuráveis. Apesar de num

sistema onde uma FPGA é partilhada simultaneamente por várias aplicações no domínio espacial e temporal, conforme exposto na secção 2.4.2, se poder considerar o espaço lógico não ocupado num determinado instante como equivalente ao estado de espera referido na definição apresentada, levantam-se algumas questões quanto à eficácia do teste nestas condições. Desde logo, não é admissível que esse espaço possa ser ocupado sem o teste estrutural dessa área estar completo, o que, podendo obstar à imediata implementação de novas funções, originaria atrasos na execução das aplicações. Por outro lado, a não introdução de atrasos impossibilitaria o teste de áreas temporariamente desocupadas, mas cujo intervalo de desocupação fosse tão pequeno que não permitisse a total aplicação do teste. Mesmo admitindo a hipótese de que a sequência de tarefas a executar pela FPGA permite a existência de intervalos temporais que possibilitem a aplicação do teste, colocava-se a questão da sua cobertura e latência. Seria muito pouco provável que, num espaço de tempo relativamente curto e completamente determinístico, a totalidade do espaço lógico fosse, fraccionadamente, disponibilizada para o teste. Mais ainda, a inconstância do volume de recursos sob teste ao longo do tempo, aliada à consequente indeterminação dos meios necessários adstritos a cada instante à aplicação do próprio teste, inviabilizam esta solução.

Ao invés do teste estrutural dos recursos a serem utilizados por uma nova função, uma outra opção é o teste funcional da própria função, antes de esta iniciar o seu funcionamento normal. Enquadram-se nesta perspectiva os testes orientados à aplicação, pertencentes ao grupo dos testes designados de *off-line* funcionais, dos quais alguns exemplos foram também apresentados no capítulo anterior. No entanto, esta abordagem enferma dos mesmos problemas que uma estratégia de teste *on-line* não-concorrente. De facto, embora a implementação dos circuitos na FPGA não tenha como consequência um aumento dos recursos ocupados, como acontece neste último, a aplicação do teste deve efectuar-se antes da entrada em operação das aplicações recém-configuradas, de modo a se garantir a não existência de faltas que afectem o seu desempenho. Essa entrada será atrasada se o intervalo entre a disponibilização de espaço de implementação e a necessidade da sua execução pela aplicação a que pertence for inferior ao somatório dos tempos de configuração da função e aplicação do teste. Para além do mais, a implementação de novas funções e a obrigatoriedade do seu teste imediato conduzem a um número indeterminado de funções sob teste simultaneamente mas em diferentes fases, sendo imprevisível o número de recursos a adstringir ao teste em cada instante, o que de igual forma inviabiliza a solução. De notar ainda que, se por um lado se assegura o bom funcionamento inicial, não se previne o não aparecimento posterior de uma falha que obste à sua permanência em serviço, uma vez que os recursos ocupados por essa função não voltam a ser testados, pois isso implicaria a interrupção da sua funcionalidade.

As primeiras propostas para o teste de dispositivos lógicos, revisitadas no capítulo anterior, foram desenvolvidas para serem aplicadas durante o teste de fabrico, apontando obviamente para a utilização de estratégias de teste *off-line* estrutural, quer através do recurso a técnicas de auto-teste, tirando partido da capacidade de reprogramação do componente, quer por aplicação externa de vectores. O posterior reaproveitamento, nos testes realizados ao longo da vida útil do componente, destas estratégias implica a paragem do sistema, devido à exigência de reconfiguração da sua funcionalidade. Atente-se que a necessidade de realização de testes, posteriormente aos testes de fabrico do próprio componente ou aos testes globais após a sua montagem num sistema mais vasto, advém da possibilidade de manifestação dos designados defeitos de fiabilidade, revistos no capítulo anterior, e que resultam sobretudo da crescente importância dos defeitos ténues, decorrentes do emprego de tecnologias submicrométricas nos novos processos de fabrico de CIs, tais como:

- a diminuição pontual da espessura das interligações metálicas, que aumenta a ameaça de electromigração e consequente aparecimento em operação de circuitos abertos;
- as imperfeições em linhas de metal muito próximas, que inicialmente não se encontram em contacto, mas que, devido ao aumento da temperatura durante a sua operação normal e à consequente dilatação, causam a perfuração do dieléctrico de óxido de metal originando que os átomos de metal se interliguem estabelecendo uma ponte permanente;
- as imperfeições no isolamento da camada de óxido da porta dos transístores, propiciando o aparecimento de uma corrente e a quebra do isolamento.

Não sendo na generalidade, conforme já foi referido, suficientemente importante para se denunciar durante os testes iniciais, esta classe de defeitos pode manifestar-se após um longo período de operação, colocando em evidência a importância que os testes realizados ao longo da vida útil do componente têm para um aumento da fiabilidade dos sistemas [Shnidman et al, 98].

Por outro lado, a diminuição de escala contribui para um aumento da susceptibilidade à radiação cósmica, a qual afecta sobretudo os dispositivos que, como as FPGAs, têm por base de definição da sua funcionalidade uma matriz de memória. As faltas provocadas por SEUs manifestam-se como faltas permanentes, pois, ao alterarem o conteúdo das células que controlam a funcionalidade da FPGA, implicam, na prática, uma mudança na funcionalidade do circuito atingido. No entanto, a causa da falha é efectivamente transitória, uma vez que o fenómeno que a origina é pontual e dele não resulta nenhum dano permanente na estrutura da FPGA, pelo que o seu efeito pode ser revertido [Huang et al., 01].

Conclui-se do exposto que, após a realização dos testes de fabrico, as faltas posteriores possíveis numa FPGA resultam de dois motivos: umas derivam de defeitos ténues que, em consequência do

seu uso, se vêm a manifestar como faltas; outras são provocadas por SEUs após a configuração. As do primeiro tipo podem ocorrer em qualquer das partes constituintes da FPGA, enquanto as segundas se restringem à matriz de memória, que compreende as células de memória presentes na memória de configuração, nos blocos de memória, nas tabelas de consulta e nos eventuais registo configurados na parte lógica.

Os requisitos requeridos para uma solução de teste que abranja a totalidade dos diferentes tipos de faltas e a possibilidade de estas se manifestarem e serem corrigidas ao longo de toda a vida útil da FPGA resumem-se, por conseguinte, nos seguintes pontos:

1. o teste dos recursos da FPGA deve ser do tipo estrutural;
2. a aplicação do teste deve ser transparente para o funcionamento das funções implementadas na FPGA;
3. o teste deve detectar todas as faltas permanentes emergentes antes, durante e após a configuração de novas funções;
4. todos os recursos da FPGA devem ser testados, independentemente da sua ocupação, num intervalo de tempo pré-determinado;
5. as funções implementadas sobre recursos que se venham a manifestar defeituosos devem ser relocalizadas em recursos previamente testados;
6. o processo de teste deve permitir igualmente a correcção da funcionalidade de circuitos afectados por SEUs;
7. a implementação da estratégia de teste deve implicar um mínimo dispêndio extra de recursos, quer em termos de área ocupada e pinos da própria FPGA, quer dos recursos externos necessários à sua execução.

Pela análise dos dois primeiros pontos da lista verifica-se que a obrigatoriedade do teste estrutural da FPGA compõe a uma estratégia de teste *off-line*, enquanto a não perturbação do funcionamento das funções implementadas implica um teste *on-line*. A solução reside na conjugação das duas num teste simultaneamente estrutural e concorrente, dependendo apenas do ponto de vista do observador. Partindo do princípio de que não é admissível afectar o funcionamento do sistema, a estratégia a seguir seria o teste *on-line* concorrente; contudo, a nível interno da FPGA, o teste deve ser *off-line* estrutural. Para que esta simbiose seja possível, é necessário que os recursos sob teste estejam livres, de forma a que a totalidade da sua estrutura possa ser testada. No entanto, a quantidade de recursos livres exigidos para esse fim não deve ser muito elevada, caso contrário não sobraria espaço para a implementação das funções necessárias ao continuado funcionamento do

sistema. Por outro lado, uma ocupação demasiado grande do espaço lógico com funções de teste inviabilizaria a solução devido a um aumento exagerado dos recursos despendidos, e consequentemente, do seu custo.

Vários factores devem ser ponderados na escolha da dimensão da área sob teste:

- ela não deve ser tão grande que afecte a disponibilidade de espaço para a implementação das funções requeridas pelas aplicações;
- as funções de teste nela implementadas devem ser acessíveis do exterior para aplicação de vectores de teste/recolha de respostas;
- a infra-estrutura requerida para o acesso exterior deve ocupar um espaço mínimo dentro da FPGA;
- a área sob teste deve percorrer todo o espaço de configuração da FPGA num intervalo de tempo perfeitamente determinado, cobrindo todos os recursos lógicos e de encaminhamento dos sinais.

Este último factor está relacionado com o ponto quatro dos requisitos e implica a capacidade de, sequencialmente, serem disponibilizadas para o teste novas áreas do espaço lógico de configuração, de forma determinística, conduzindo à obtenção de uma latência fixa de teste para a totalidade dos recursos. No entanto, como já foi explicitado, não é lícito pensar que a sequência própria de funcionamento do sistema permita alcançar esse objectivo, pelo que é necessária a implementação de um mecanismo que, sucessivamente, permita a relocação das funções actualmente em execução (funções activas), replicando a sua funcionalidade em áreas previamente testadas e libertando continuadamente para o teste os recursos que se encontrem ocupados. Esta relocação deve ter em conta os seguintes factores:

- o funcionamento das funções não pode ser interrompido durante a relocação;
- a relocação não pode implicar uma degradação no desempenho da função.

A sequência de relocação deve conduzir a uma rotação da área sob teste, abrangendo a totalidade da área de configuração da FPGA. A área sob teste deve ser primeiramente testada e posteriormente reconfigurada com a funcionalidade implementada na próxima área a testar. Segundo esta sequência, a relocação far-se-á sempre para uma área previamente testada, pelo que, na eventualidade de durante o teste ser detectada uma falta num determinado recurso da FPGA, a funcionalidade previamente implementada nesse recurso já terá sido relocalizada num recurso não defeituoso, cumprindo-se desta forma o ponto cinco da lista de requisitos.

As sucessivas reconfigurações parciais da FPGA, necessárias à rotação da área sob teste, obrigam à geração de ficheiros de configuração parciais. Estes ficheiros podem ser obtidos a partir da leitura da configuração presente naquele instante na FPGA, e sua posterior alteração, ou a partir do ficheiro de configuração original guardado numa memória exterior não-volátil<sup>1</sup>, cuja existência é obrigatória visto tratar-se de FPGAs baseadas em memória estática volátil. Neste último caso, qualquer alteração devida a SEUs, que afecte pontualmente a funcionalidade dos circuitos envolvidos na relocação, será corrigida, uma vez que a geração do ficheiro de configuração parcial se efectua a partir do original, cobrindo assim o requisito exposto no ponto seis da lista.

Resumindo, o teste deve realizar-se sequencialmente, varrendo a totalidade da FPGA, sem paragens e com um tempo de latência determinado. Findo o varrimento, o processo deverá poder ser repetido, nos mesmos moldes, um número infinito de vezes, para que qualquer falta que ocorra antes, durante ou após a configuração de novas funções, ao longo de toda a vida útil do componente, seja detectada, cobrindo-se, assim, a exigência formulada no ponto três da lista de requisitos.

A nível da FPGA, os custos de implementação de uma solução que cubra as questões focadas dividem-se em dois tipos:

- espaço ocupado pela área sob teste;
- infra-estrutura necessária para implementação da estratégia de teste.

Relativamente ao primeiro ponto, a escolha do tamanho da área sob teste deve ter em conta não só o espaço que ocupa, o que influi directamente no custo, mas também todos os factores associados à replicação não intrusiva de circuitos activos. A relocação de grandes áreas da FPGA, por replicação da funcionalidade nelas implementada, conduz à alteração simultânea de um elevado número de interligações, o que pode criar dificuldades ao encaminhamento dos sinais. Igualmente provável é a necessidade de transferência dos valores de estado de um grande número de registo, sendo obrigatório garantir que eles possam ser actualizados a qualquer instante no decorrer do processo de relocação, sem que disso resulte qualquer incoerência entre os dados presentes nos registos replicados e nas respectivas réplicas no final do processo.

A implementação da solução pretendida implica a reconfiguração parcial da FPGA e a aplicação de vectores às configurações de teste configuradas na área sob teste, bem como a posterior recolha das

---

<sup>1</sup> O termo memória exterior não-volátil é usado aqui em sentido lato, abrangendo não só as tradicionais memórias EPROM, mas também outros dispositivos de armazenamento de dados, como discos magnéticos, ópticos ou outros, locais ou à distância (nos casos em que a gestão do sistema é efectuada remotamente, por exemplo, via Internet).

respostas. Ambas as tarefas podem ser executadas através da infra-estrutura BS, parte integrante da FPGA.

A utilização de uma FPGA como base de um sistema de computação reconfigurável exige que, local ou remotamente, exista um sistema de controlo da sua utilização, tipicamente um microprocessador<sup>2</sup>, um microcontrolador ou um computador, responsável pela gestão do espaço tridimensional de configuração, ao qual podem ser acrescentados os mecanismos necessários à implementação da estratégia de teste que seguidamente se propõe e que basicamente se traduz num aumento da capacidade de memória disponibilizada pelo sistema, variável com o tamanho da própria FPGA.

## 4.2. DEFINIÇÃO DA METODOLOGIA

A proposta de solução a seguir apresentada divide-se genericamente em três partes, que satisfazem os requisitos identificados: a relocação da funcionalidade, por replicação dos elementos activos, a rotação da área consignada ao teste, varrendo a totalidade da FPGA num intervalo de tempo determinado, e o teste estrutural dessa área. A Figura 4.1 ilustra genericamente estes três passos, que se desenrolam sucessiva e ininterruptamente até que a totalidade da FPGA esteja testada [Gericota et al., 01]. Testada toda a FPGA, o processo pode ser repetido imediatamente ou após um determinado intervalo de tempo, dependendo da estratégia global de teste adoptada para o sistema.

Cada uma das três partes que compõem a metodologia de teste proposta apresenta especificidades de execução que importa analisar mais cuidadosamente, de forma a se compreender melhor as opções tomadas e os problemas associados à sua implementação prática. Devido à sua diferente natureza, é tratada separadamente a aplicação da metodologia aos blocos lógicos e aos recursos de interligação.

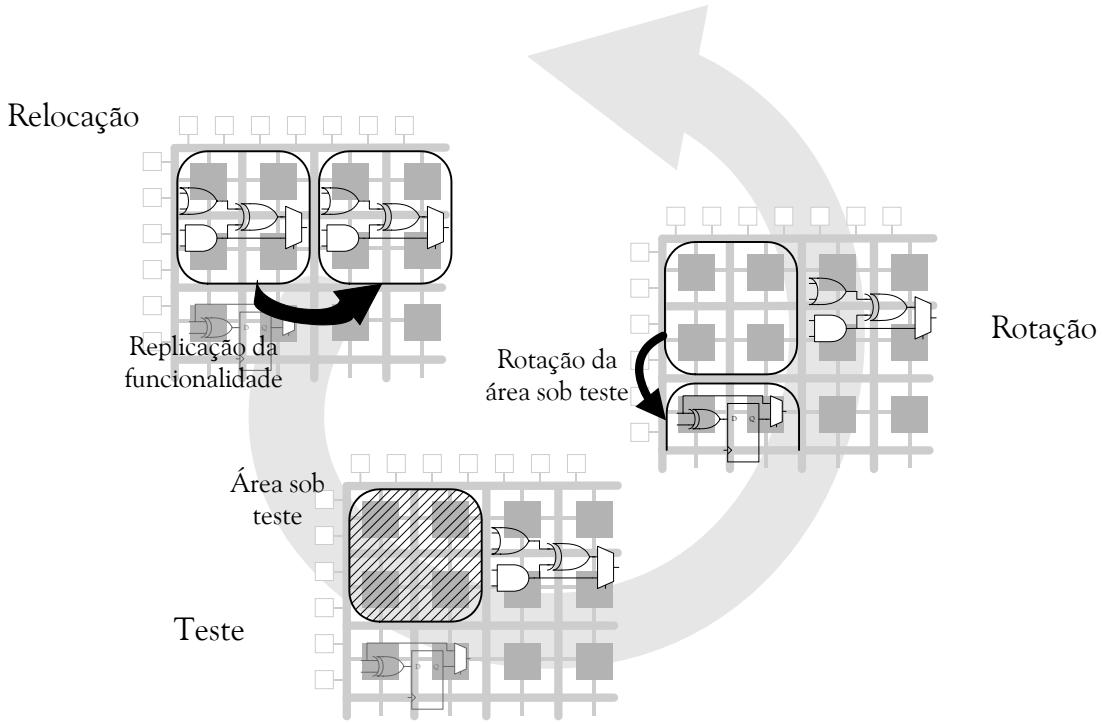
### 4.2.1 RELOCAÇÃO DOS BLOCOS FUNCIONAIS

A relocação dinâmica de uma função activa, entendida como a relocação de uma função presentemente a ser executada pelo sistema e cuja operação não deve ser interrompida, não é uma tarefa trivial. Para além da criação de uma réplica da função por cópia da sua especificação funcional, também as interligações da réplica com o resto do circuito têm de ser estabelecidas, respeitando as interligações da função original. Dependendo da funcionalidade implementada,

---

<sup>2</sup> Em algumas FPGAs mais recentes, caso das Virtex-II Pro [Xilinx, 02], este processador poderá inclusivamente ser um dos processadores embutidos na própria FPGA.

pode igualmente ser necessário preservar na réplica a informação interna de estado da função replicada, tudo sem se perturbar o seu normal funcionamento.



**Figura 4.1:** Metodologia de teste proposta

De notar que a questão da transferência de estado coloca um problema particular, uma vez que as FPGAs actualmente disponíveis permitem a leitura do estado actual de cada elemento de retenção (configurável como *flip-flop* ou *latch*) presente nos blocos lógicos, por intermédio de uma operação de leitura do conteúdo da memória de configuração, mas não permitem de igual forma a sua escrita. Todavia, mesmo que tal fosse possível, qualquer actualização de estado que se efectuasse durante o intervalo que decorre entre a leitura do estado dos elementos de retenção da função a replicar e a escrita nos da sua réplica seria perdida, resultando numa incoerência dos dados no final do processo de relocação.

Adicionalmente, coloca-se a questão da definição da área a replicar, em virtude da quantidade de alterações que o processo de relocação envolve ao nível da configuração da FPGA. Uma área demasiado grande, por exemplo uma coluna de blocos lógicos, como proposto em [Abramovici et al., 99a], implica a relocação de um grande número de CLBs e de todas as interligações das funções neles configuradas, além da transferência da informação interna de estado de um número potencialmente elevado de elementos de retenção. Para obstar a estes problemas, aqueles autores propõem a paragem do sistema durante um curto intervalo de tempo, para permitir a relocação

segura da lógica presente na coluna a relocate para a última coluna testada. Todavia, tal paragem é contrária aos requisitos definidos na secção anterior para o teste de uma FPGA inserida numa plataforma de computação reconfigurável.

De referir que a proposta de relocation de uma coluna enquadra-se bem na estrutura da memória de configuração das FPGAs, que se assemelha a uma matriz rectangular de bits agrupados em vectores verticais, com um bit de largura, estendendo-se do topo ao fundo da matriz (isto é, ao longo de todo o comprimento de uma coluna). O vector é a unidade atómica de configuração, a mais pequena porção da memória de configuração endereçável individualmente que pode ser escrita e/ou lida. Os vectores agrupam-se paralelamente em unidades maiores designadas colunas de configuração. É a divisão da totalidade da memória de configuração da FPGA em vectores endereçáveis individualmente que possibilita a sua configuração parcial dinâmica. Cada vector de configuração, ao abranger a totalidade da coluna, permite a reconfiguração numa única vez da funcionalidade de todos os blocos lógicos que a integram. No entanto, o elevado número de blocos lógicos que uma coluna pode conter, entre a dezena e a centena de CLBs, a multiplicar pelo número de entradas e saídas que interliga cada um ao resto do circuito, implica que a quantidade de alterações envolvidas na definição da configuração da FPGA cause perturbações no seu funcionamento, tornando, na prática, inexequível a relocation segura da funcionalidade de uma coluna, sem impelir à paragem do sistema.

Outra unidade candidata natural ao processo de relocation é o bloco lógico. A replicação de um bloco lógico implica um número relativamente pequeno de interligações com o resto do circuito. Cada CLB possui, nas Virtex, um total de 28 entradas e 14 saídas, das quais nem todas estabelecem ligações a outros blocos do circuito através da matriz de encaminhamento<sup>3</sup>. De reparar que no caso da replicação de uma coluna, mesmo que uma parte substancial das interligações se processe dentro da própria coluna, o número de interligações a ser estabelecido com o resto do circuito seria certamente muito mais elevado, tendo em conta que a mais pequena das FPGAs da família Virtex da Xilinx possui 9 CLBs por coluna [Xilinx, 02a]<sup>4</sup>. Por outro lado, tendo em atenção que cada CLB possui quatro elementos de retenção, seria necessário transferir, no pior dos casos, informação de estado referente a 36 elementos de retenção. A assunção de um bloco lógico como unidade a

---

<sup>3</sup> Duas das catorze saídas são saídas dedicadas (linhas de transporte) que não passam pela matriz de encaminhamento de saída.

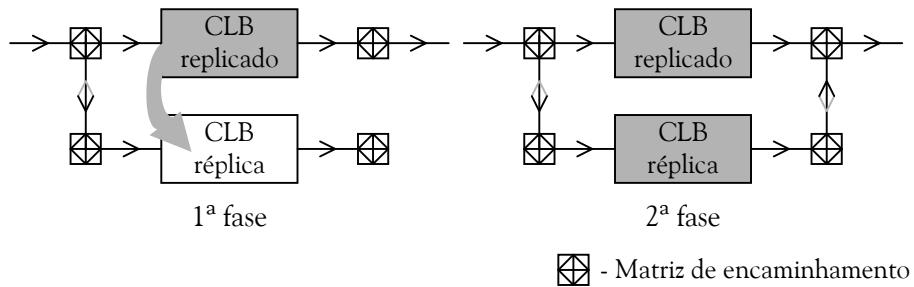
<sup>4</sup> No caso da série 4 da família ORCA da Lattice, cada bloco lógico possui 53 entradas e 20 saídas, o que multiplicado pelo número de CLBs de uma coluna (26 para o caso da mais pequena FPGA da família), constitui um bom indicador das dificuldades e perturbações que a replicação de uma coluna, com o estabelecimento simultâneo de tão elevado número de interligações, certamente acarretaria. Além disso, a necessidade de transferir a informação de estado referente aos oito elementos de retenção de cada um dos CLBs complicaria ainda mais a tarefa [Lattice, 02a].

replicar obriga à transferência do valor de estado de apenas quatro elementos de retenção, o que torna viável a implementação de um esquema de transferência de estado garantindo a coerência dos dados no final da replicação, mesmo que durante o processo de transferência os valores sejam actualizados.

O processo de replicação implica a reconfiguração parcial de uma ou mais colunas da FPGA, pelo facto de o restabelecimento dos sinais de entrada e saída do bloco lógico (bem como os da sua réplica) poder naturalmente implicar o cruzamento de uma ou mais colunas, antes de atingir a sua origem ou destino. É ainda importante reter que mesmo a alteração de um único bit de uma coluna de configuração implica a rescrita de um vector de dados de configuração que se estende ao longo de todo o comprimento dessa coluna. No entanto, a rescrita dos mesmos dados de configuração não gera quaisquer sinais transitórios, pelo que este processo não afecta os restantes recursos cobertos por esta operação.

Qualquer acção de reconfiguração deve assegurar que as interligações do bloco lógico replicado não são quebradas antes de serem totalmente restabelecidas a partir da sua réplica, caso contrário, a interrupção desses sinais perturbaria (podendo mesmo interromper) o seu funcionamento. Mais ainda, a operação da réplica deve encontrar-se num estado perfeitamente estável antes de as suas saídas serem interligadas ao resto do circuito, para evitar o aparecimento de transitórios na saída.

Uma solução plausível para ultrapassar estas questões é a divisão do processo de replicação em duas fases, conforme ilustrado na Figura 4.2 [Gericota et al., 02a] e [Gericota et al., 02b].



**Figura 4.2:** As duas fases do processo de replicação

Na primeira fase, a configuração interna do CLB é replicada na nova posição e as entradas de ambos os CLBs, replicado e réplica, são colocadas em paralelo. Este passo implica a reconfiguração parcial da FPGA, que é efectuada através da interface BS, conforme definido na secção anterior. A configuração através desta interface é relativamente lenta se comparada com a velocidade de operação do sistema, pelo que, na segunda fase, as saídas do CLB réplica estão já perfeitamente estáveis quando são ligadas ao circuito.

Após esta fase é já possível libertar o espaço ocupado pelo CLB replicado para nele serem implementadas as funções de teste. A relocação do bloco lógico fica então completa, sem que tenha havido interrupção do funcionamento do circuito.

O mesmo processo é válido para a relocação de recursos de interligação, sendo as interligações primeiramente duplicadas e posteriormente libertadas, podendo, através de um processo em tudo semelhante ao usado para os blocos lógicos, ser testadas.

As tabelas de consulta nos CLBs podem ser configuradas pelo utilizador para operar como módulos de memória distribuídos. Contudo, não é viável a extensão do conceito de replicação às tabelas de consulta configuradas nesse modo. Com efeito, embora o conteúdo dessas tabelas possa ser lido e escrito através das correspondentes operações sobre a memória de configuração, não existe nenhum mecanismo, a não ser a paragem do sistema, capaz de assegurar a coerência entre os valores das tabelas de consulta do CLB replicado e da sua réplica, se uma operação de escrita nas tabelas tiver lugar durante o intervalo de replicação [Huang et al., 01]. Mais ainda, dado que cada vector de configuração cobre uma coluna inteira de CLBs, as mesmas posições de memória das tabelas de consulta de todos os CLBs da coluna são escritas com um único comando. Embora a escrita do mesmo valor não gere perturbações para o funcionamento dos circuitos, o facto de algumas das tabelas de consulta dessa coluna poderem funcionar como blocos de memória implica que se possa estar a rescrever em algumas das tabelas valores diferentes dos actualmente presentes. É pois necessário assegurar, antes da replicação, que os dados em todas as tabelas de consulta da coluna que contém o CLB a replicar e daquela para onde ele vai ser replicado (no caso de não ser a mesma) se mantêm constantes, ou (em caso contrário) que são também alterados externamente por meio de uma reconfiguração parcial [Xilinx, 00]. Significa isto que, mesmo não sendo replicados, CLBs contendo tabelas de consulta a operar como blocos de memória não devem residir em colunas que possam ser afectadas pelo processo de reconfiguração. De igual forma, deve atender-se a este facto aquando da simples alteração do encaminhamento de sinais. De facto, a informação contida em cada um dos vectores de configuração mistura os dados referentes a todos os elementos programáveis dentro dessa coluna (valores presentes nas tabelas de consulta, elementos de controlo dos CLBs, dos IOBs e das interligações), pelo que qualquer processo que implique a reconfiguração de uma coluna devido a alterações no encaminhamento de sinais que nela têm origem ou destino, ou que simplesmente a atravessam, deve ter em conta os valores presentes nesse instante nas tabelas de consulta configuradas como memórias (já que o uso do ficheiro inicial de configuração reporá os valores iniciais dos dados, sem ter em conta as alterações posteriores). Na prática, esta restrição limita o emprego da metodologia proposta a circuitos cujas tabelas de consulta apenas implementem lógica combinatória.

Por motivos semelhantes, também não é possível o teste dos blocos de memória RAM. Embora, neste caso, o endereçamento directo para leitura e escrita nesses blocos seja possível, mantém-se o problema da garantia da coerência dos dados após o processo de replicação.

#### 4.2.2 ESTRATÉGIA DE ROTAÇÃO

Uma das principais consequências da relocação, a nível da configuração da FPGA, é a alteração dos caminhos que foram inicialmente definidos pela ferramenta de apoio ao projecto. Esse processo de distribuição da lógica a implementar pelos recursos da FPGA, bem como o encaminhamento dos sinais a ela associados, deu origem a uma determinada velocidade de funcionamento do circuito, a qual pode inclusive ter obedecido a eventuais restrições temporais impostas pelo projectista, com o objectivo de assegurar uma dada frequência de funcionamento. O processo de relocação, ao alterar a distribuição inicial da lógica, provoca alterações no encaminhamento dos sinais, que impõem novos atrasos de propagação. Se o processo de relocação implicar a criação de caminhos mais extensos (em termos temporais) que os inicialmente definidos, tal terá como consequência um aumento dos tempos de propagação dos sinais, resultando numa diminuição da frequência máxima de funcionamento da função.

Por outro lado, o objectivo final é testar todos os recursos da FPGA, pelo que a área sob teste deve rodar por todos os blocos lógicos, seguindo um percurso pré-definido, de modo a garantir um determinado tempo de latência do teste. A estratégia adoptada para essa rotação deve privilegiar a minimização do impacto que tal acarreta sobre a frequência máxima de funcionamento, de modo a não perturbar o funcionamento geral do sistema.

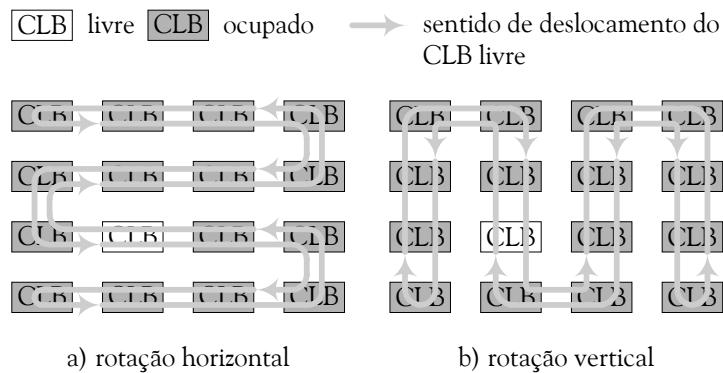
Não menos importante, embora sem impacto directo no funcionamento do circuito, é o próprio custo da reconfiguração. A implementação do processo de relocação obriga à geração de vários ficheiros parciais de configuração, cujo tamanho depende do número de vectores de configuração necessários para replicar e libertar cada CLB. Um número grande de vectores implica não só tempos de reconfiguração mais longos, e consequentemente maiores tempos de latência de teste, mas também maiores recursos de memória [Gericota et al., 01b].

Três possibilidades foram inicialmente consideradas para o estabelecimento de uma regra para a rotação da área sob teste ao longo da matriz de CLBs:

1. aleatória;
2. horizontal;
3. vertical.

A primeira estratégia apresenta vários pontos negativos que levaram à sua rápida rejeição. Se o algoritmo que gera a atribuição de recursos para a implementação de uma função na FPGA tende, numa tentativa de reduzir os tempos de propagação dos sinais, a agrupar em blocos lógicos contíguos a totalidade dos componentes da lógica necessária à sua implementação, não seria sensato dispersá-la, não só por causa do aumento dos tempos de propagação, mas também por tal colocar demasiada pressão nos limitados recursos de encaminhamento da FPGA. Mais ainda, uma rotação aleatória da área sob teste implicaria igualmente uma latência imprevisível na cobertura de defeitos, o que não é aceitável.

A segunda hipótese, é a utilização de uma estratégia de rotação horizontal, conforme ilustrado na Figura 4.3 - a). Após o teste, o CLB fica livre para que a funcionalidade de um dos CLBs que lhe é horizontalmente contíguo possa ser para ele deslocada, libertando por sua vez esse CLB para o teste. Esta sequência horizontal de varrimento deve prosseguir até que todo o espaço lógico de configuração tenha sido testado. De modo idêntico, a terceira hipótese, ilustrada na Figura 4.3 - b), segue uma sequência de varrimento vertical. Em qualquer dos casos, o CLB que após o teste fica livre vai-se movendo segundo o sentido de deslocamento indicado na Figura 4.3, até ter percorrido toda a matriz de CLBs.



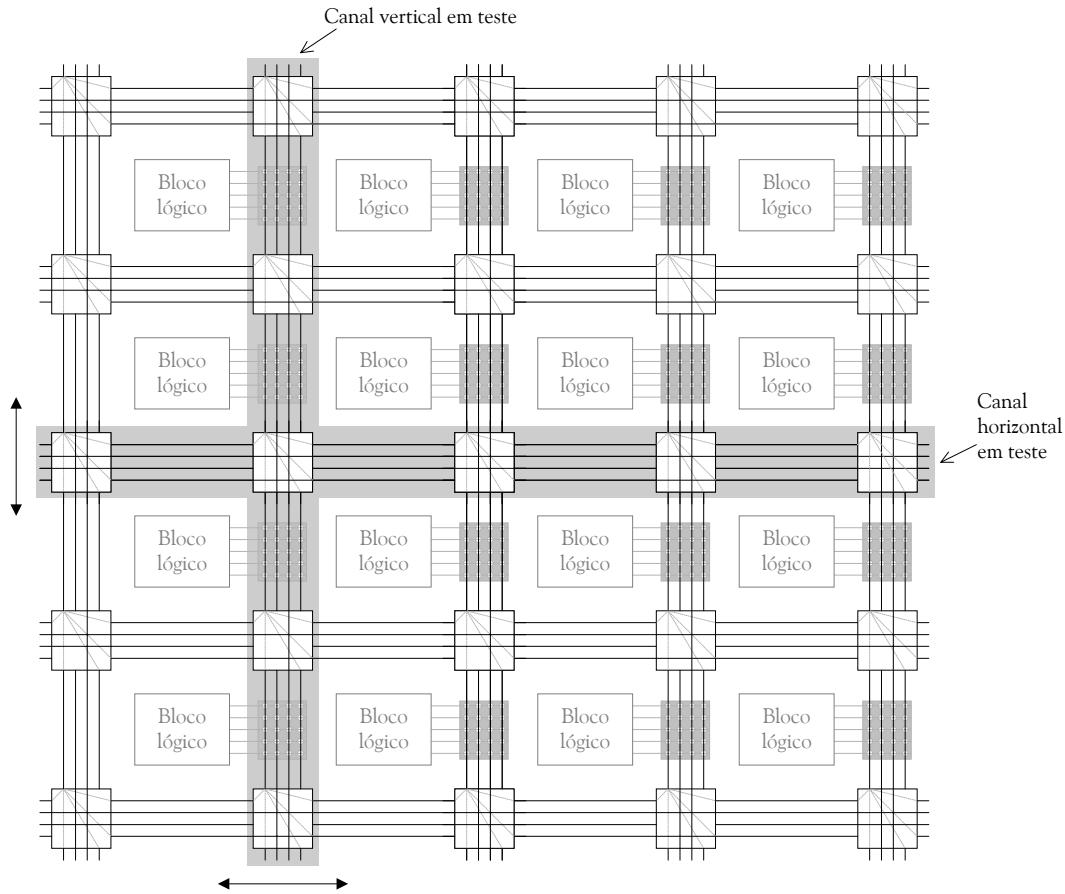
**Figura 4.3:** Estratégias de rotação

Tendo a relocação lugar entre CLBs contíguos, não só a pressão exercida sobre os recursos de encaminhamento é menor, como se previne o aumento dos tempos de propagação dos sinais dentro das funções implementadas e se evita a dispersão dos componentes da sua lógica pela matriz de CLBs. O sentido do deslocamento do CLB livre inverte-se, em ambos os casos, após um varrimento completo da matriz. Tal permite que, após concluir dois varrimentos, a distribuição da lógica implementada e o encaminhamento dos sinais retome a sua configuração inicial.

A escolha entre uma ou outra das estratégias de rotação depende de vários factores, nomeadamente o tipo de funções lógicas implementadas na FPGA e a utilização que elas fazem dos recursos de

encaminhamento dedicados, a forma predominante da distribuição dos componentes da lógica a ela associada (rectangular, quadrada, circular) e a sua orientação (horizontal ou vertical), e a existência de sinais com elevado número de derivações [Gericota et al., 01a].

O teste dos recursos de interligação global, por seu lado, é independente do teste dos CLBs, uma vez que, desenvolvendo-se os segmentos de vários comprimentos ao longo de canais, o seu teste obriga a um varrimento horizontal e vertical desses canais, conforme ilustrado na Figura 4.4. Este varrimento implica dentro de cada canal um número de replicações que depende sobretudo do nível de ocupação dos segmentos desse canal e de canais subjacentes para onde interligações activas possam ser desviadas, respeitando o imperativo da sua não interrupção. De notar que não é obrigatório que o encaminhamento original seja restabelecido após o teste de determinado segmento, podendo inclusive o processo de teste ser aproveitado simultaneamente para a optimização dos recursos de encaminhamento.



**Figura 4.4:** Varrimento adoptado para os recursos de interligação

O deslocamento horizontal e vertical dos canais em teste permite não só o teste dos segmentos da estrutura de encaminhamento global, mas também dos recursos de encaminhamento dentro das matrizes de encaminhamento globais. O teste dos recursos de encaminhamento local e dos

segmentos de interligação entre as entradas e saídas dos CLBs sob teste é coberto pelo teste do próprio CLB, visto que o acesso a esses recursos só é possível por seu intermédio. Além disso, é através desses recursos que são aplicados os vectores de teste ao CLB e capturadas as suas respostas. Na realidade, o teste da totalidade dos CLBs implica já, por si só, implicitamente, o teste de uma parte significativa dos recursos de encaminhamento.

Repara-se, no entanto, que se os recursos de encaminhamento estiverem demasiado saturados, pode não ser possível garantir o seu teste na totalidade, por impossibilidade da sua replicação.

#### 4.2.3 APLICAÇÃO DO TESTE

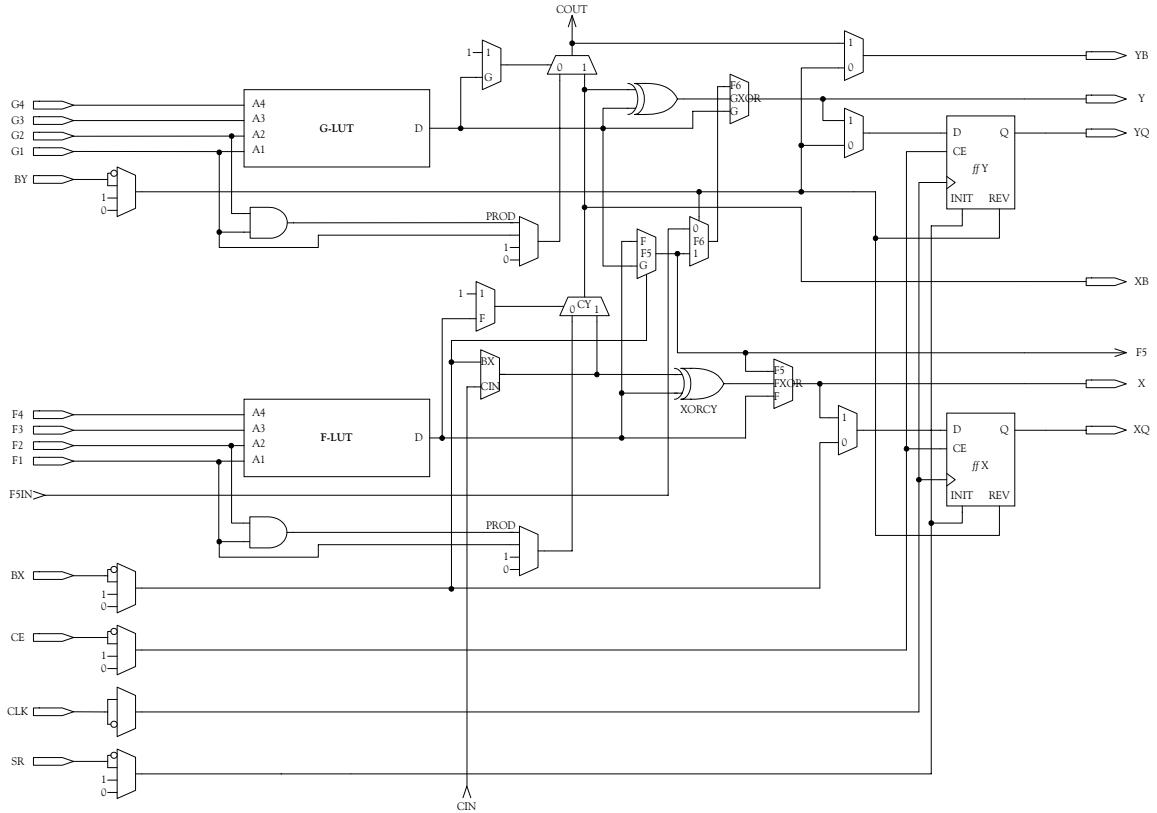
O modelo de teste adoptado para o bloco lógico configurável segue a arquitectura do CLB das famílias Virtex e Spartan da Xilinx [Xilinx, 02a], tendo-lhe sido retirada a lógica de controlo da sua operação em modo memória (blocos de endereçamento e de controlo de leitura e escrita das posições de memória da tabela), em face da impossibilidade da extensão do conceito de replicação às tabelas de consulta configuradas nesse modo.

Cada bloco lógico é constituído por duas partes (*slices*) independentes e exactamente iguais<sup>5</sup>. O modelo de teste, representando apenas uma das *slices* e tendo já em conta as restrições enunciadas, encontra-se ilustrado na Figura 4.5.

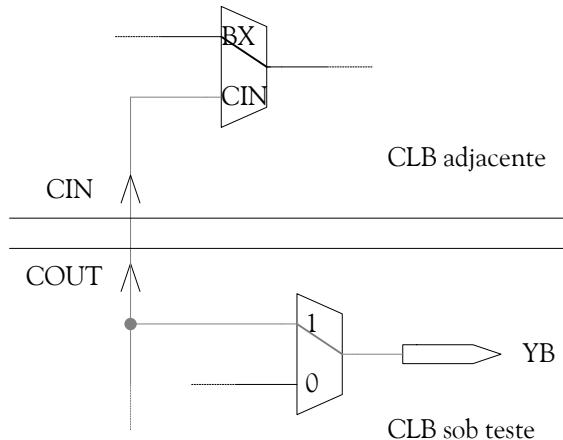
A replicação e teste de apenas um bloco lógico levanta o problema do acesso aos recursos de encaminhamento dedicados, afectos à implementação de sinais de transporte. No sentido de reduzir ao máximo o atraso de propagação desses sinais, permitindo a implementação de lógica aritmética muito rápida (acumuladores, subtractores, multiplicadores, comparadores, contadores, registos de deslocamento), as linhas de transporte interligam directamente a saída da lógica de transporte de um CLB com a entrada do mesmo tipo de lógica no CLB que lhe está superiormente adjacente na coluna. Como tal, é impossível a aplicação de estímulos de teste a esta entrada e a recolha de respostas na sua saída, a não ser através dos CLBs que lhe estão imediatamente adjacentes. No entanto, é possível recolher as respostas resultantes da aplicação de estímulos à linha COUT e, consequentemente, à linha CIN do CLB superiormente adjacente na coluna, através da saída YB, conforme ilustrado na Figura 4.6. De notar que mesmo que esta linha estivesse a ser usada, anteriormente à replicação, pela função aí implementada, a introdução, entre ambos os CLBs, do CLB sob teste implica a sua quebra e o consequente reencaminhamento dos sinais de transporte através de recursos genéricos.

---

<sup>5</sup> Ver ponto 2.4.1.



**Figura 4.5:** Modelo de teste



**Figura 4.6:** Recolha de estímulos do teste da linha de transporte

A utilização de uma estratégia de auto-teste interno implicaria o uso de recursos da própria FPGA para a sua implementação. Dado que esses circuitos ocupariam, em princípio, determinados recursos fixos dentro do espaço de configuração, poderiam eles próprios ser afectados por defeitos emergentes, que supostamente detectariam no CLB livre, mas não na sua própria lógica, deturpando eventualmente os resultados do teste. A implementação de circuitos de auto-teste auto-testáveis ou auto-verificáveis conduziria a um aumento dos recursos internos da FPGA consagrados ao teste, bem como da sua própria complexidade de operação. Tendo em conta os

factores enunciados, optou-se pela aplicação do teste ao CLB livre a partir do exterior, tirando partido da infra-estrutura de teste BS pré-existente na FPGA.

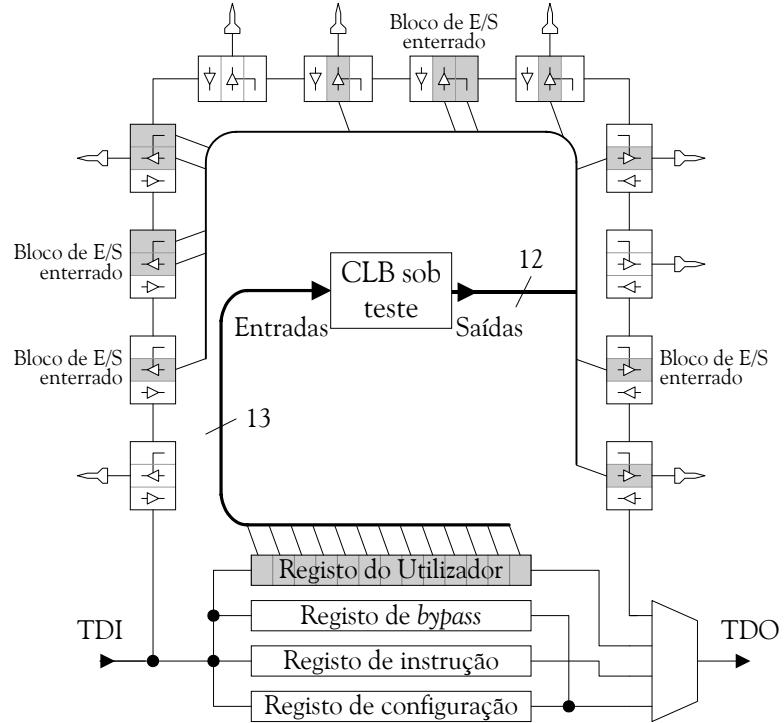
A infra-estrutura de teste BS é usada para aplicar os vectores de teste e capturar as respostas, com as saídas do CLB sob teste a serem encaminhadas para células do registo BS associadas a blocos de E/S. Nas FPGAs da família Virtex, todos os blocos de E/S (enterrados e não enterrados) são considerados como pinos bidireccionais, contribuindo com três células para este registo, independentemente da configuração individual de cada um. Na prática, dependendo da configuração do bloco, muitas dessas células são redundantes. Por exemplo, um pino configurado como entrada necessita apenas de uma única célula. Todavia, as duas restantes não são removidas da cadeia, podendo, por isso, ser aproveitadas na captura das respostas aos vectores de teste aplicados ao CLB sob teste, e, como tal, evitar a ocupação de recursos extra para este propósito.

Contudo, não é possível aplicar os vectores de teste através do registo BS sem afectar os valores presentes em cada entrada da lógica, pelo que essa aplicação requer o uso de um registo alternativo, que, embora definido pelo utilizador, é igualmente controlado através da infra-estrutura BS. A família Virtex da Xilinx [Xilinx, 02a] permite a definição de dois destes registos, enquanto as ORCA série 4 da Lattice [Lattice, 02a] possibilitam a definição de quatro, que são construídos usando os recursos internos configuráveis da própria FPGA. O número de células deste Registo do Utilizador é igual ao número de entradas (treze) do modelo de teste ilustrado na Figura 4.5. Mesmo que vários CLBs sejam testados em paralelo, o comprimento do registo é constante, uma vez que todas as *slices* recebem sempre os mesmos vectores de teste. O número de blocos lógicos usados na implementação do Registo do Utilizador (sete), mais o próprio bloco lógico sob teste, constituem, no limite, o número máximo de recursos do componente que é necessário para implementação desta metodologia de teste. A Figura 4.7 ilustra a implementação na FPGA do procedimento de teste.

A arquitectura do Registo do Utilizador baseia-se na arquitectura definida para as próprias células BS, permitindo o deslocamento e aplicação dos vectores de teste. Este deslocamento é relativamente rápido, em virtude do reduzido comprimento do registo, mas a rapidez do deslocamento das respostas capturadas no registo BS depende da posição dentro da cadeia das células usadas para essa captura. No pior caso, aquele em que a célula mais próxima de TDI é usada para a captura de um bit da resposta, o comprimento do vector a deslocar para se recolher todas as respostas é igual ao comprimento do próprio registo BS [Gericota et al., 01a].

As respostas obtidas são comparadas com as respostas esperadas, devendo, no caso da detecção de uma falta, o recurso ser “marcado” e o seu uso interditado em futuras reconfigurações. A

implementação de um procedimento de tolerância a faltas implica, contudo, a existência de recursos sobresselentes na FPGA, que possam substituir os recursos faltosos. A grande vantagem é que, ao contrário da redundância modular em sistemas não reconfiguráveis, os recursos sobresselentes são, neste caso, do tipo genérico, podendo ser configurados para substituir qualquer função.



**Figura 4.7:** Procedimento de teste de um bloco lógico

Em virtude da estrutura reconfigurável do CLB, a implementação de um teste estrutural obriga ao uso de várias configurações de teste, cujo número depende sobretudo do número máximo de bits de configuração dos multiplexadores programáveis. A geração dos vectores de teste para cada uma das configurações é efectuada com base num modelo híbrido (sempre-a/funcional, discutido no ponto 3.2.2), devido ao desconhecimento da estrutura interna de implementação das primitivas que constituem o bloco lógico.

Sendo cada CLB constituído por duas *slices* iguais e independentes, a geração dos vectores de teste é efectuada para apenas uma delas, com base no modelo de teste apresentado, e de modo a garantir uma cobertura de faltas de 100%, segundo o modelo de faltas considerado. Os vectores de teste são aplicados paralelamente a ambas as *slices*, sendo a captura das suas saídas independente, o que permite uma resolução da falta a uma *slice*, sem a necessidade de um diagnóstico adaptativo posterior.

Para o teste de interligações, o princípio a aplicar é idêntico, mas com a substituição do bloco lógico sob teste por uma fieira de interligações sob teste, tendo a geração de vectores por base os modelos de faltas:

- segmento sempre-aberto;
- ponte.

Uma vez que não se conhece o desenho exacto dos recursos de encaminhamento e a sua posição relativa nas várias camadas de metal, considera-se como possível a existência de um curto-circuito (ponte) entre quaisquer dois segmentos  $S_i$  e  $S_j$  de um mesmo canal.

Qualquer das interligações sob teste estabelece uma ligação entre uma das células do Registo do Utilizador, a partir da qual lhe são aplicados os vectores, e uma célula BS, onde é efectuada a captura da resposta. Cada uma destas interligações é tipicamente composta por vários segmentos e atravessa uma matriz de interligação local (pertencente ao bloco lógico que implementa a célula do Registo do Utilizador) e uma ou mais de interligação global. Transístores de passagem, controlados pelo estado da sua célula de configuração, permitem o encaminhamento dos sinais dentro das matrizes de interligação. Estes transístores podem ser afectados por faltas do tipo transístores sempre-fechados ou sempre-abertos, por defeito no próprio transístor ou na célula da memória de configuração que define o seu estado, estabelecendo, respectivamente, uma ligação permanente entre dois segmentos da matriz ou uma interligação em aberto.

As faltas na matriz, que originam um comportamento igual ao de uma ligação permanente, são cobertas pelo modelo de faltas em ponte, enquanto as que se manifestam como interligações em aberto são cobertas pelo modelo de faltas de segmento sempre-aberto [Portal, 99].

A concepção de um procedimento de teste que varra de igual forma toda a memória de configuração é inviabilizado pelo facto de a escrita e leitura de dados nas suas células não poder ser isolada da configuração da própria FPGA. Embora o uso de várias configurações de teste para os blocos de memória e os recursos de interligação e encaminhamento garanta logo à partida o teste conjunto de um elevado número de bits dessa memória, bem como da sua interface, deve proceder-se à sua leitura completa e comparação com o ficheiro de configuração após a configuração total ou parcial da FPGA para garantir a não existência de faltas que a afectem.

#### 4.3. SUMÁRIO

Este capítulo procurou identificar os pontos positivos, bem como os pontos fracos ou omissos, encontrados em anteriores abordagens às metodologias de teste para FPGAs, com ênfase especial

nas parcialmente reconfiguráveis, de modo a facilitar a identificação dos requisitos que serviram de base à proposta aqui apresentada. A partir destes requisitos estabeleceu-se um conjunto de propostas de solução faseadas que, em conjunto, estabelecem uma proposta coerente de uma nova metodologia para o teste concorrente de FPGAs com capacidade de reconfiguração parcial dinâmica.

Esta nova proposta é aplicável às FPGAs disponíveis comercialmente e não requer qualquer modificação da sua arquitectura. O teste completo da totalidade dos blocos lógicos da FPGA é garantido, dentro das limitações devidas à impossibilidade de replicação de tabelas de consulta operando em modo memória, sem afectar o funcionamento de qualquer uma das funções presentemente implementadas, requerendo para tal um dispêndio mínimo de recursos ao nível do componente. Adicionalmente, é possibilitado o teste das interligações, sujeito a restrições, no sentido em que, dependendo da sua ocupação, pode não ser exequível o teste da totalidade destes recursos.

Uma vez que o relógio do sistema não é interrompido durante o processo de teste, é possível a optimização conjunta dos recursos de encaminhamento e mesmo a desfragmentação do espaço de configuração, com a computação *on-line* de novas colocações de blocos e interligações, sem que tal implique qualquer atraso na execução das funções actualmente presentes na FPGA [Gericota et al., 02] e [Gericota et al., 03a]. Este benefício adicional que é proporcionado pela metodologia proposta é particularmente importante em diversos contextos, o que justifica algumas considerações que a este respeito são apresentadas no capítulo concluente desta dissertação.

Do ponto de vista do sistema, todo o processo de teste (e a possível optimização dos recursos da FPGA) é efectuado *on-line* e de forma completamente transparente. Aliás, sendo o próprio processador do sistema a controlar a implementação da metodologia, uma gestão cuidada do recurso impede o aparecimento de qualquer conflito.

O acesso à FPGA para a implementação das diferentes fases do método e do seu controlo é efectuado usando exclusivamente a infra-estrutura BS, não sendo necessário o dispêndio extra de recursos ao nível da carta, e implica apenas a implementação de um pequeno registo que ocupa recursos do espaço de configuração da própria FPGA sob teste.

Foi excluído da análise efectuada o teste concorrente dos blocos de E/S, pela impossibilidade da sua replicação e libertação para o teste, visto que esse processo implicaria o reencaminhamento de sinais ao nível da própria carta de circuito impresso o que não é viável no contexto considerado [Gericota et al., 03], e da própria infra-estrutura de teste, matéria extensivamente abordada em [Ferreira, 92a].



## **5. REPLICAÇÃO DE CIRCUITOS ACTIVOS**



O objectivo da replicação é possibilitar a libertação para o teste de CLBs presentemente ocupados, copiando o seu conteúdo para CLBs previamente testados, sem que isso implique uma paragem do sistema ou afecte a sua funcionalidade, mesmo que esses CLBs estejam presentemente activos, isto é, sejam parte de uma função actualmente a ser usada pelo sistema.

O mecanismo de replicação depende da funcionalidade configurada no CLB, distinguindo-se basicamente dois grupos: o dos CLBs que implementam circuitos puramente combinatórios e o dos CLBs que implementam circuitos onde são usados elementos de retenção, *flip-flops* ou *latches*. A distinção deve-se ao facto de não ser suficiente a cópia da funcionalidade para os CLBs do segundo grupo, por ser também necessário transferir o valor de estado presente nos elementos de retenção, sem que isso afecte o normal funcionamento do circuito ou, em algum momento, se possam colocar problemas de coerência entre os valores de estado presentes no CLB replicado e na sua réplica.

Por outro lado, a própria estrutura da memória de configuração e o mecanismo de configuração que lhe está associado levantam algumas dificuldades à manutenção dos recursos de encaminhamento entre os diversos CLBs adstritos a uma mesma função activa, aquando da replicação de um deles.

A resolução destes problemas levou à consideração de uma estratégia de replicação em duas etapas, complementada com o uso de um bloco auxiliar de replicação quando é necessário proceder à transferência de valores de estado dos elementos de retenção.

A replicação em duas etapas evita que os sinais recebidos e propagados do CLB a ser relocado sejam interrompidos em algum momento, perturbando o funcionamento do sistema. O emprego do bloco auxiliar de replicação garante a coerência entre os valores de estado dos elementos de retenção replicados, sem colocar quaisquer restrições à sua actualização, em qualquer momento do processo de replicação. Contudo, a replicação de CLBs faltosos, por aplicação do método proposto, pode conduzir a situações em que saídas com diferentes valores lógicos, no CLB replicado e na sua réplica, sejam colocadas em curto-circuito. Estas situações são extensivamente analisadas, quanto ao seu comportamento e às possíveis consequências para o componente e para o funcionamento do sistema.

O mecanismo de replicação implementado possibilita só por si, igualmente, a recuperação de erros nas células de memória (elementos de retenção e memória de configuração) provocados por incidência de radiação, que pode causar a inversão do valor armazenado nessas células.

A principal vantagem da abordagem proposta é a sua independência face ao estado geral do sistema. Em qualquer momento, um CLB activo pode ser replicado num CLB livre e a sua funcionalidade transferida, independentemente das tarefas que esteja a realizar, sem necessidade de implementação de complexos sistemas de verificação do seu estado actual.



## 5.1. CIRCUITOS COMBINATÓRIOS

A replicação de elementos activos (CLBs e interligações que fazem parte de funções activas) levanta alguns problemas de ordem prática, como sejam<sup>1</sup>:

1. a possível interrupção temporária de linhas activas;
2. o aparecimento de transitórios provocados pela alteração dinâmica da configuração da FPGA;
3. a transferência do valor de estado, no caso da utilização de elementos de retenção, *flip-flops* ou *latches* (problema a ser analisado na secção seguinte).

A possibilidade de interrupção temporária de linhas activas deve-se à disposição arquitectónica da memória de configuração. A configuração da FPGA é efectuada segundo um esquema de colunas, como se mostra na Figura 5.1, em que, no caso das colunas  $C_1$  a  $C_n$ , cada um dos vários vectores de configuração abrange unicamente uma coluna de *slices* de cada CLB. A informação contida em cada um desses vectores mistura os dados de configuração de todos os elementos programáveis dentro dessa coluna (valores presentes nas tabelas de consulta, elementos de controlo dos CLBs e IOBs e das interligações), tornando-se a nova configuração activa à medida que os vectores vão sendo escritos na memória. Várias experimentações práticas mostraram que, em virtude da sequencialidade do mecanismo de configuração, o processo de replicação teria forçosamente de ser dividido em várias partes. A sua execução de uma só vez, isto é, usando um único ficheiro de reconfiguração parcial, conduzia a que algumas linhas de entrada e saída do CLB replicado fossem quebradas antes de serem restabelecidas a partir do CLB réplica, ou antes de este estar devidamente configurado, tendo como consequência o aparecimento de transitórios que perturbavam (e por vezes alteravam) o funcionamento de todo o sistema. De referir que a rescrita dos mesmos dados de configuração não gera sinais transitórios, mas a rescrita de novos dados pode conduzir ao seu aparecimento, especialmente se os valores presentes nas tabelas de consulta, ou o encaminhamento dos sinais, forem alterados [Xilinx, 00].

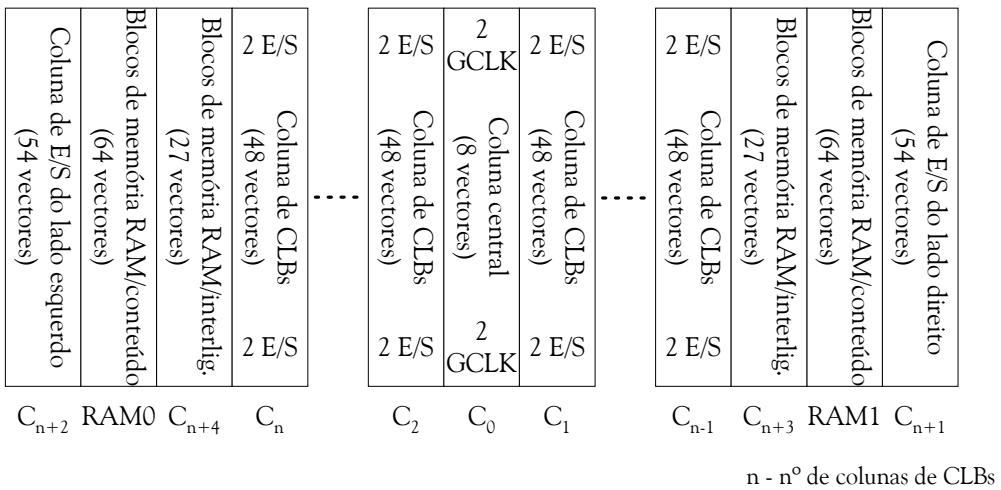
Tendo em consideração os problemas descritos, equacionou-se a hipótese de se proceder à replicação dos CLBs activos em duas fases distintas [Gericota et al., 02e]:

---

<sup>1</sup> Ignora-se nesta fase o problema da transferência dos valores presentes nas tabelas de consulta quando estas funcionam em modo memória, uma vez que já foram analisadas no capítulo anterior as razões que conduzem à impossibilidade da sua replicação.

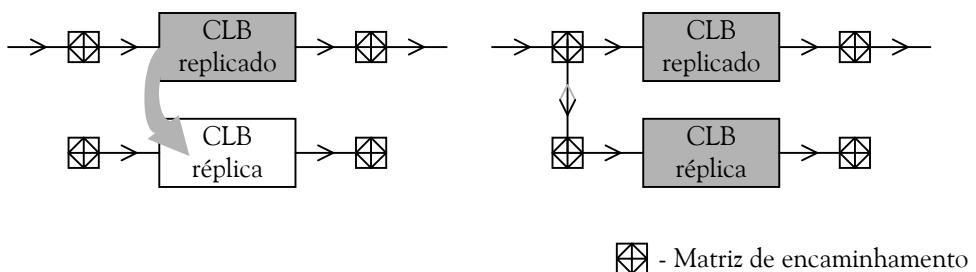
**1<sup>a</sup> fase** cópia da configuração interna e colocação em paralelo das entradas do CLB replicado e da sua réplica;

**2<sup>a</sup> fase** colocação em paralelo das saídas do CLB replicado e da sua réplica.



**Figura 5.1:** Alocação dos vectores aos recursos configuráveis

A 1<sup>a</sup> fase, esquematizada na Figura 5.2, engloba num único passo a cópia da funcionalidade do CLB replicado para o CLB réplica e a colocação em paralelo das entradas activas. Desta forma, é efectuada a completa configuração do CLB réplica e das suas entradas, sem que a operação do CLB replicado, e logo a totalidade da operação do circuito, sofra qualquer interferência.



**Figura 5.2:** Colocação em paralelo das entradas dos dois CLBs envolvidos na replicação

Só posteriormente, na 2<sup>a</sup> fase, é que são colocadas em paralelo as saídas. O tempo que medeia entre a 1<sup>a</sup> e a 2<sup>a</sup> fase garante a estabilização das saídas do CLB réplica e evita o aparecimento de transitórios nefastos para o funcionamento do circuito.

A colocação em paralelo das saídas levantou experimentalmente alguns problemas, em virtude de, arquitectonicamente, a FPGA não estar preparada para que as saídas dos seus CLBs sejam “curto-circuitadas”. Na maioria das implementações de circuitos digitais é comum a existência de uma linha que liga simultaneamente a várias entradas, mas é encarado como um erro a ligação de duas ou mais saídas a uma mesma linha, por ser factor potencial de geração de conflitos no caso de os

valores presentes no conjunto dessas linhas serem diferentes. A tentativa de união das saídas deparou-se pois, na prática, com problemas, quer ao nível das ferramentas de projecto, que proibiam tal prática, quer devido à indisponibilidade de recursos físicos vocacionados para a sua concretização. Essas dificuldades foram, no entanto, ultrapassadas através do recurso a ferramentas desenvolvidas expressamente para esse fim e que permitem a manipulação directa das interligações sem verificação eléctrica posterior. Esse conjunto de ferramentas, denominado *Jbits* [Xilinx, 00a], é constituído por classes de *Java*<sup>2</sup> que proporcionam uma *Application Programming Interface* (API) que permite a manipulação directa dos ficheiros de configuração, em formato binário, das FPGAs da família Virtex da Xilinx.

A divisão em duas fases do processo de replicação permite, assim, ultrapassar os problemas relacionados com o possível aparecimento de transitórios, que resultariam em perturbações para o sistema, mediante a aplicação das mesmas entradas a ambos os blocos lógicos e à interligação de ambas as saídas durante o processo. Uma vez que os blocos são configurados de forma idêntica, esta técnica resulta nos mesmos sinais de saída em ambos os blocos. O bloco replicado pode então ser retirado do circuito, assumindo o bloco réplica a sua funcionalidade [Shnidman et al., 98].

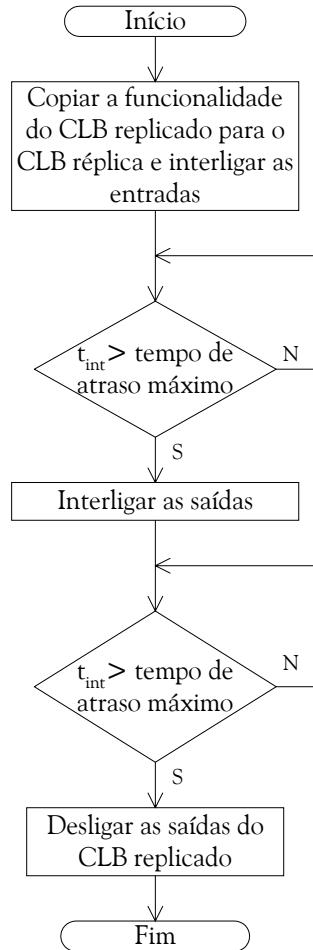
O fluxograma da Figura 5.3 descreve todos os passos necessários à implementação do processo de replicação de blocos lógicos que implementem uma função puramente combinatória. A cada rectângulo do fluxograma corresponde um ficheiro de configuração parcial, pelo que, no total, são necessários três ficheiros para se concretizar a replicação do bloco lógico. De referir que, na fase final do processo, as saídas do CLB replicado devem ser desligadas do circuito antes das entradas, caso contrário, a sequencialidade de configuração pode conduzir a que entradas sejam desligadas antes das saídas, provocando o aparecimento de transitórios nas saídas, antes destas serem desligadas do circuito, o que poderia perturbar o funcionamento do sistema. O passo seguinte, que será detalhado no capítulo sete e compreende o desligar das entradas, não aparece neste fluxograma, uma vez que está já englobado na parte correspondente ao estabelecimento da primeira configuração de teste do CLB que foi replicado, e na qual é implementada toda a estrutura necessária para o seu teste.

A reconfiguração da FPGA é efectuada através da infra-estrutura de teste BS, a uma frequência que, no máximo, atinge os 33 MHz [Xilinx, 02a]. Sendo uma interface de configuração do tipo série, a reconfiguração é, em comparação com os tempos de propagação dentro da FPGA, relativamente lenta. Daí que as condições expressas no fluxograma da Figura 5.3 não necessitem, na prática, de verificação, pois é garantido que o intervalo entre a programação de dois ficheiros de

---

<sup>2</sup> Linguagem de programação interpretada, orientada ao objecto, multiplataforma e independente desta.

configuração parcial,  $t_{int}$ , excede largamente o tempo de propagação dos sinais no primeiro caso entre as entradas e saídas do CLB e, no segundo, entre estas e os nós de interligação. Cada ficheiro de configuração parcial possui, para além dos bits de configuração propriamente ditos, um cabeçalho constituído por várias palavras de inicialização e de endereçamento do vector de configuração, cuja especificação se encontra em anexo a esta tese, inserida na descrição arquitectónica das Virtex. O deslocamento deste cabeçalho, associado ao deslocamento da instrução de configuração para o registo de instruções da infra-estrutura de teste, obriga à aplicação de 206 impulsos do relógio de teste (sendo 14 correspondentes ao deslocamento da instrução de configuração para o registo de instrução e início da leitura dos dados e 192 ao cabeçalho do ficheiro de configuração parcial), aos quais, a uma frequência máxima de 33MHz, corresponde um intervalo de 6,2  $\mu$ s, uma ordem de grandeza acima de  $t_{int}$  para a maioria das funções implementadas. Desta forma, é condição necessária e suficiente para que a replicação se processe sem problemas, o envio consecutivo, sem interrupções, dos três ficheiros de configuração parcial [Gericota et al., 02d].



**Figura 5.3:** Fluxograma do processo de replicação de um bloco de lógica combinatória

Em resumo, a divisão do processo de configuração em duas fases evita a possibilidade de interrupção de linhas activas durante o seu decurso, bem como a interligação com o resto do circuito de linhas de saída antes de o CLB réplica estar devidamente configurado, ou de os sinais de entrada estarem devidamente ligados e os sinais de saída previamente estabilizados, evitando assim o aparecimento de transitórios.

Contudo, a aplicação da mesma técnica à replicação de circuitos sequenciais, em que é necessário proceder não só à replicação funcional do CLB mas igualmente à transferência dos valores de estado associados aos quatro elementos de retenção presentes, configuráveis como *flip-flops* ou *latches*, resolve apenas parcialmente o problema. A FPGA apresenta algumas limitações quanto à capacidade de transferência desses valores, nomeadamente pelo facto de não possibilitar a sua escrita, embora permita a leitura directa do estado de cada um desses elementos de retenção. De salientar também que, dada a relativa lentidão do mecanismo de leitura e reconfiguração face às velocidades de funcionamento permitidas para a FPGA, se durante o intervalo entre a leitura dos valores presentes nos elementos de retenção do CLB replicado e a sua rescrita nos do CLB réplica (supondo que tal fosse possível) ocorresse uma operação de escrita, isso originaria problemas de coerência entre os valores presentes nos elementos replicados e nas suas réplicas. Em [Abramovici et al., 99] é proposta a utilização de caminhos temporários que permitam efectuar a transferência dos valores entre os quatro pares de elementos de retenção dos dois CLBs envolvidos na replicação, sob controlo exterior. Contudo, o problema da existência de um intervalo de tempo entre o momento da transferência e o da activação do CLB réplica (efectiva substituição no circuito activo), durante o qual qualquer actualização de estado seria perdida conduzindo a uma situação de incoerência, foi resolvido por aqueles autores parando o relógio do sistema. Embora o caminho a seguir seja inevitavelmente, dada a impossibilidade de escrita directa, o do estabelecimento de caminhos temporários entre os elementos de retenção replicados e as suas réplicas, a paragem do relógio do sistema é uma técnica intrusiva, pelo que é necessário o estudo de um mecanismo de transferência alternativo.

## 5.2. CIRCUITOS SEQUENCIAIS

No estudo do problema da transferência do valor do estado interno dos elementos de retenção, *flip-flops* ou *latches*, há que ter em conta o tipo de implementação associado ao uso de cada elemento concreto, sendo possível distinguir três casos [Gericota et al., 02d]:

1. a replicação de circuitos síncronos, com elementos de retenção configurados como *flip-flops* (isto é, cujo conteúdo é actualizado à transição positiva ou negativa do sinal de relógio), sem sinal de habilitação de relógio e com ou sem sinal de inicialização síncrono ou assíncrono;

2. a replicação de circuitos síncronos, com elementos de retenção configurados como *flip-flops*, com sinal de habilitação de relógio e com ou sem sinal de inicialização síncrono ou assíncrono;
3. a replicação de circuitos assíncronos que usam elementos de retenção configurados como *latches* (isto é, cuja saída segue a entrada quando o sinal de habilitação de entrada está activo), com sinal de habilitação de entrada e com ou sem sinal de inicialização.

A análise das características próprias de cada um dos três casos e a solução proposta para a sua replicação é efectuada separadamente nas secções seguintes.

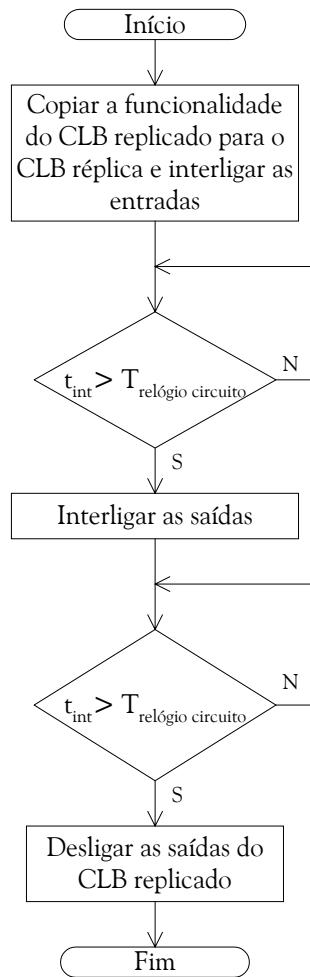
### 5.2.1 CIRCUITOS SÍNCRONOS COM SINAL DE RELÓGIO LIVRE

Nos blocos lógicos que implementam circuitos síncronos com sinal de relógio livre, em que os elementos de retenção configurados como *flip-flops* são actualizados sempre à transição activa do sinal de relógio do circuito, o processo de replicação não levanta qualquer tipo de problema específico, para além dos que já se colocavam à replicação de circuitos combinatórios, sendo reutilizável a solução então encontrada para a sua replicação.

Após a cópia da funcionalidade e a colocação em paralelo das entradas do CLB replicado e da sua réplica basta esperar que um impulso de relógio proveniente do próprio sistema actualize os valores de estado presentes em ambos os CLBs, replicado e réplica, antes de se proceder à colocação em paralelo das saídas. Obviamente, este tempo de espera deve ser conhecido (frequência de funcionamento), pelo que a utilização desta solução só é viável neste caso. De notar que este tempo de espera é apenas aparente, uma vez que o intervalo que medeia entre a primeira fase, onde é efectuada a configuração da funcionalidade do CLB réplica e se colocam em paralelo as suas entradas com as do CLB replicado, e a segunda, onde se colocam em paralelo as saídas, é, em virtude do próprio processo de reconfiguração, mais que suficiente para a actualização do estado dos *flip-flops*, mesmo que a velocidade do sistema seja relativamente baixa. De facto, e uma vez que na implementação global da metodologia se optou por efectuar a configuração por intermédio da infra-estrutura IEEE 1149.1, como já foi observado para o caso da replicação de funções puramente combinatórias, é necessário um mínimo de 206 impulsos do relógio de teste entre o final da primeira configuração e o início da segunda, pelo que o relógio do sistema poderá ser até 206 vezes mais lento do que aquele. Se optarmos pela frequência máxima possível para o relógio de teste (33 MHz), o relógio do sistema poderá ter uma frequência tão baixa como 160 KHz, valor muito pouco usual nos sistemas actuais. Repare-se igualmente que desta forma não se corre o risco de aparecimento de transitórios nas saídas. Uma vez que os *flip-flops* de ambos os CLBs já capturaram

o mesmo valor de estado, os sinais presentes nas saídas do CLB réplica apresentam o mesmo valor que os presentes nas saídas do CLB replicado e estão já perfeitamente estabilizados, aquando da colocação em paralelo das saídas.

O fluxograma da Figura 5.4 descreve todos os passos necessários à implementação do processo de replicação de blocos lógicos que implementem uma função sequencial síncrona com sinal de relógio livre. A cada rectângulo do fluxograma corresponde um ficheiro de configuração parcial, pelo que, tal como no caso dos circuitos combinatórios, são necessários três ficheiros para se concretizar a replicação do bloco lógico.



**Figura 5.4:** Fluxograma do processo de replicação de um bloco de lógica síncrona com relógio livre

De notar que a existência ou não de sinal de inicialização, síncrono ou assíncrono, é irrelevante neste caso, uma vez que, ao efectuar-se o paralelo das entradas, este sinal será igualmente aplicado a ambos os CLBs, replicado e réplica, pelo que a sua activação a qualquer instante afectará de igual forma os elementos envolvidos e, em consequência, as suas saídas.

### 5.2.2 CIRCUITOS SÍNCRONOS COM SINAL DE RELÓGIO BLOQUEÁVEL

No caso de existência de sinal de habilitação de relógio, não se tem a garantia de que, durante o intervalo de tempo entre o paralelo das entradas e das saídas, o *flip-flop* capture o valor presente na entrada, pois esse sinal pode manter-se desactivado. Essa garantia tem de ser fornecida pela própria estratégia de replicação.

No sentido de se perceber a forma como os recursos de configuração de cada CLB são utilizados, procedeu-se a uma análise cuidada dos resultados da implementação prática de um subconjunto de 14 circuitos<sup>3</sup> do conjunto de 22 circuitos designados por “ITC’99 benchmarks”, desenvolvidos pelo Grupo de CAD do Politécnico di Torino [Politécnico di Torino, 99] e [Corno et al., 00]. Trata-se de um conjunto de circuitos-padrão, reunido com a cooperação de várias empresas da área da electrónica, destinado a ser usado na experimentação de técnicas de concepção para a testabilidade e na geração automática de vectores de teste. Este conjunto de circuitos-padrão apresenta como principais características:

- descrições em VHDL, ao nível da transferência de registos (RTL), totalmente sintetizáveis;
- descrições segundo modelos comportamentais, com um ou mais processos concorrentes, contendo em alguns casos também código estrutural;
- inexistência de directivas específicas de compilação;
- circuitos puramente síncronos, com um só sinal de relógio, de fase única, ligado directamente aos elementos de retenção;
- elementos de retenção com uma entrada de inicialização global, permitindo a inicialização dos circuitos a qualquer momento e sempre da mesma forma;
- inexistência de blocos de memória interna (excepto bancos de registos), de barramentos com terceiro estado e de ligações do tipo *wired* (*wired-AND*, *wired-OR*)<sup>4</sup>;
- tamanho e complexidade largamente variável.

Dessa análise resultou uma enumeração da ocupação-tipo de cada *slice*:

1. *slice* de passagem<sup>5</sup>;

---

<sup>3</sup> A restrição a um subconjunto de 14 elementos deve-se exclusivamente à limitação imposta pela capacidade da FPGA utilizada na fase experimental.

<sup>4</sup> Circuitos que realizam, sem recurso a uma porta lógica específica, operações lógicas a partir da interligação de saídas do tipo colector aberto (*wired-AND*) ou emissor aberto (*wired-OR*) [Sandige, 90].

2. lógica combinatória não registada;
3. registo simples sem sinal de habilitação de relógio e com sinal de inicialização assíncrono;
4. lógica combinatória registada sem sinal de habilitação de relógio e com sinal de inicialização assíncrono;
5. registo simples com sinal de habilitação de relógio e com sinal de inicialização assíncrono;
6. lógica combinatória registada com sinal de habilitação de relógio e com sinal de inicialização assíncrono.

Nos primeiros quatro casos enumerados, a estratégia de replicação em duas fases, já apresentada anteriormente, permite replicar os CLBs envolvidos sem necessidade de recurso a outros meios. Contudo, nos dois últimos, a existência de um sinal de habilitação de relógio não garante que, durante o intervalo de tempo entre o paralelo das entradas e das saídas, o *flip-flop* capture o valor presente na entrada, pois esse sinal pode estar desactivado.

De notar que nos casos três e quatro, durante o processo de replicação, não se pode propriamente falar de uma “transferência do estado” entre os registo dos CLB replicado e réplica. Trata-se antes, isso sim, de possibilitar que ambos os conjuntos de registo acedam simultaneamente às mesmas entradas e, consequentemente, capturem o mesmo valor.

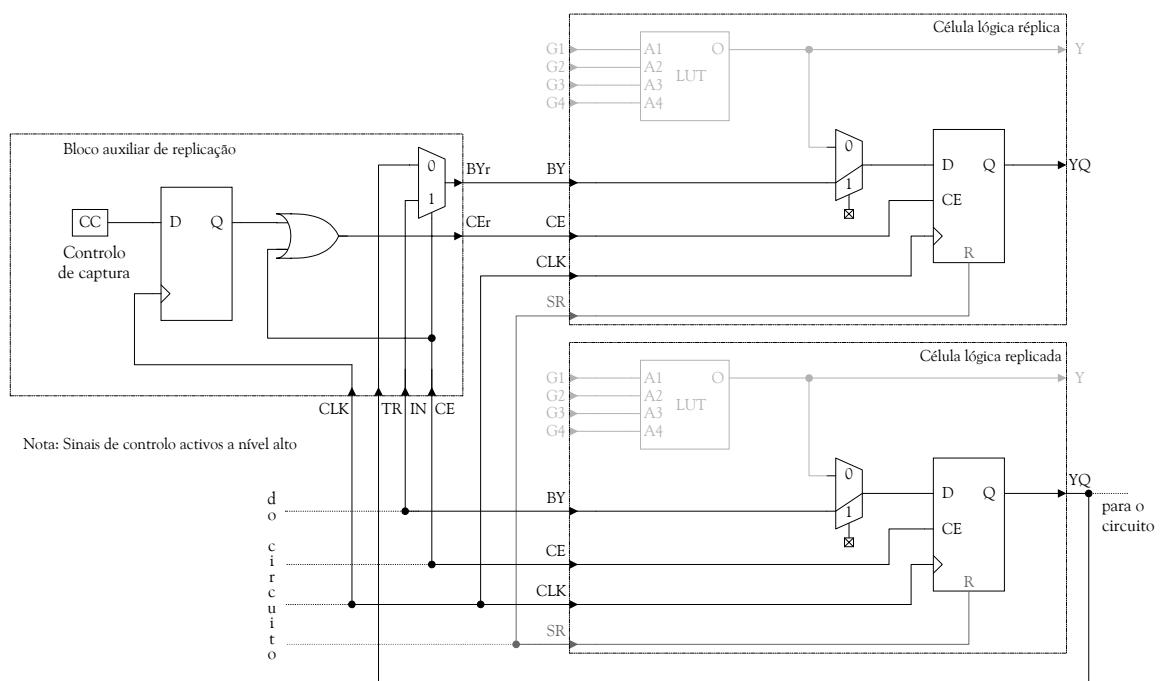
Nos casos cinco e seis, a simples colocação em paralelo das entradas e a geração, como parte do processo de replicação, de um sinal de habilitação para os registo do CLB réplica, de modo a que ele capture nesse momento as entradas, não permite assegurar que ambos os conjuntos de registo armazenem o mesmo estado. Entre o instante de captura das entradas pelos registo do CLB replicado e a sua captura, sob controlo do processo de replicação, pelos do CLB réplica, nada garante que os valores presentes nas entradas não tenham mudado, pelo que, no final do processo, ter-se-iam valores diferentes nos registo do CLB replicado e nos da sua réplica. Nestes casos, é necessário implementar uma real “transferência do estado” entre os registo dos CLB replicado e réplica, não esquecendo que, durante o decurso de todo o processo de replicação, os registo podem ser actualizados uma ou mais vezes pelo próprio circuito implementado nos CLBs envolvidos. Desta forma, será necessário criar um caminho entre a saída do *flip-flop* do CLB replicado e a entrada do *flip-flop* do CLB réplica, que permita a transferência do valor armazenado, do primeiro para o segundo, sem perturbar a operação normal da funcionalidade implementada nem impedir a actualização de ambos os registo. Uma vez que a ocupação do CLB depende da funcionalidade

<sup>5</sup> Um ou mais sinais atravessam as *slices* do CLB sem serem alterados por qualquer operação lógica. Os recursos internos são usados exclusivamente para encaminhar os sinais e/ou actuar como *buffers*.

implementada, as ligações a estabelecer são distintas para cada um dos casos. No entanto, no sentido de manter o mecanismo de replicação o mais simples possível, por forma a não aumentar a complexidade e o tempo de execução, procurou-se uma solução que, com um número mínimo de alterações, satisfizesse não só estes dois casos, mas abarcasse também a replicação de circuitos assíncronos, cuja análise se efectuará na próxima secção.

Desde logo se colocou a necessidade da existência de um bloco auxiliar, que se denominou *bloco auxiliar de replicação* e que, neste caso, permite gerar o sinal de habilitação de relógio que vai possibilitar ao registo do CLB réplica capturar o valor transferido, sem entrar em conflito com o sinal de habilitação de relógio da aplicação implementada, e seleccionar entre a entrada do *flip-flop* no CLB réplica e o caminho de transferência da saída do *flip-flop* do CLB replicado.

A principal diferença para os casos cinco e seis prende-se com o acesso ao valor presente na entrada do *flip-flop* do CLB réplica, que pode ser capturado a qualquer instante por ordem do sistema. O caso cinco encontra-se ilustrado na Figura 5.5, na qual, por simplicidade de desenho e de percepção da solução, se encontra representada apenas uma versão simplificada (sem perda de generalidade) de uma célula lógica, das duas que constituem cada um dos dois *slices* que compõem um CLB. O tratamento da replicação das restantes células lógicas contidas em cada CLB é exactamente igual ao do exemplo apresentado e decorre em paralelo temporal com este.



**Figura 5.5:** Replicação de uma célula contendo um registo simples com sinal de habilitação de relógio e com sinal de inicialização síncrono ou assíncrono

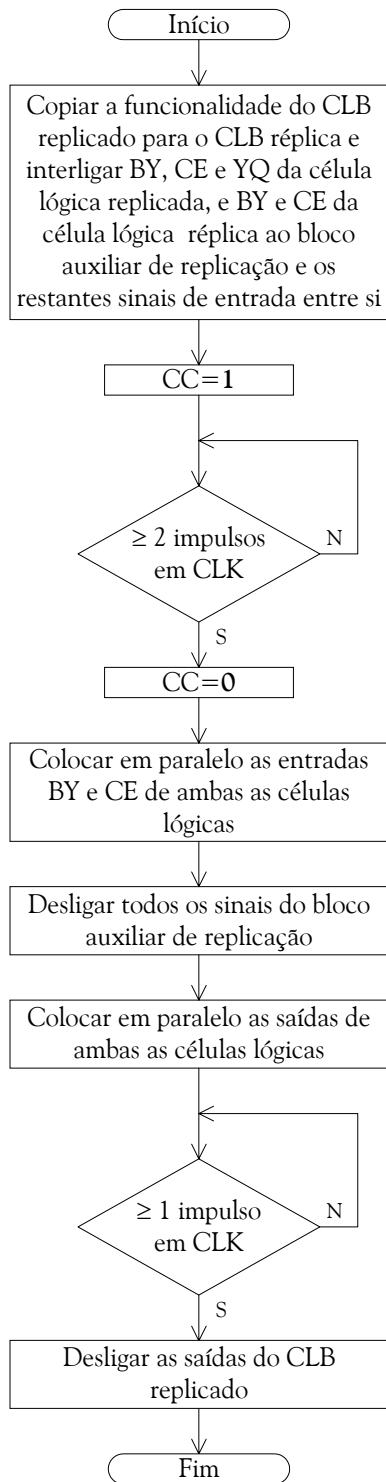
No caso cinco, a entrada para o *flip-flop* é efectuada directamente a partir da entrada BY<sup>6</sup> de cada célula lógica, pelo que é esta entrada que é ligada à entrada IN do bloco auxiliar de replicação. A saída do *flip-flop* do CLB replicado é ligada à entrada TR do mesmo bloco, permitindo transferir o seu conteúdo de um para outro dos *flip-flops*. Ambos os sinais são ligados às entradas de um multiplexador cuja selecção é efectuada pelo sinal de habilitação de relógio (CE). No caso de este sinal não estar activo, na entrada do *flip-flop* do bloco réplica ter-se-á o valor armazenado no *flip-flop* replicado. A activação do sinal de controlo de captura (CC) permite ao *flip-flop* réplica capturar o valor do *flip-flop* replicado ao primeiro impulso de relógio do próprio sistema (CLK). Após a captura, é desactivado o sinal de controlo de captura e colocadas em paralelo as entradas dos sinais BY e CE de ambos os blocos. Posteriormente, são desligados todos os sinais de e para o bloco auxiliar de replicação e colocadas em paralelo as saídas, completando-se a replicação. As diferentes fases deste processo estão esquematizadas no fluxograma da Figura 5.6.

De notar que a activação a qualquer instante do sinal de habilitação de relógio (CE) pela aplicação implementada colocará na entrada do *flip-flop* da célula réplica o valor que está igualmente presente na entrada do *flip-flop* replicado, capturando ambos esse valor ao primeiro impulso de relógio (CLK).

O facto de o sinal de habilitação de relógio activado no decurso do processo de replicação (CC) ser registado prende-se apenas com um pormenor de implementação. Para que o controlo da captura não ocupe recursos de E/S (pinos), optou-se pelo seu controlo através da memória de configuração, mantendo-se, em consequência, também neste caso, a interface BS como única interface de acesso e controlo de todo o processo de teste. A activação do sinal CC é efectuada através da alteração do bit de programação do multiplexador presente na entrada BY, conforme se observa na implementação numa *slice* do bloco auxiliar de replicação, apresentada na Figura 5.7. O encaminhamento deste sinal com um dispêndio mínimo de recursos até uma das saídas do bloco lógico obriga ao atravessamento do elemento de retenção. Por essa razão, e tendo em conta o atraso de um ciclo de relógio introduzido por este elemento na habilitação do relógio do elemento de retenção do *flip-flop* réplica, aparece na primeira condição presente no fluxograma da Figura 5.6 a indicação de um mínimo de dois ciclos de relógio para garantir, a partir do momento em que o bit do multiplexador (CC) é programado, a captura do valor de estado. A imposição da segunda condição constante do mesmo fluxograma tem como único objectivo garantir a estabilidade dos sinais nas saídas antes de terminado o processo de replicação.

---

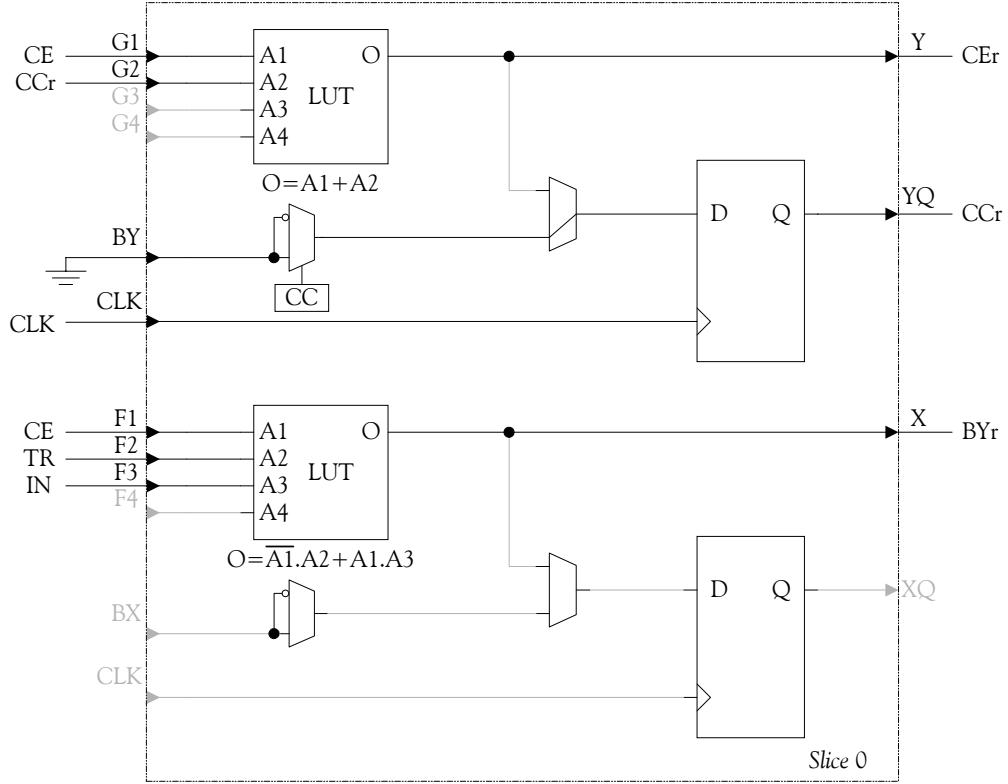
<sup>6</sup> A notação usada advém da célula lógica empregue como exemplo, correspondendo na outra às entradas e saídas que se lhe sobrepõem (ver Figura 2.31).



**Figura 5.6:** Fluxograma do processo de replicação de uma célula contendo um registo simples com sinal de habilitação de relógio e com sinal de inicialização síncrono ou assíncrono

A necessidade de dois ciclos de relógio para assegurar a transferência do valor de estado entre os elementos de retenção do CLB replicado e da sua réplica conduz, com base na mesma análise efectuada anteriormente para a replicação de blocos lógicos implementando circuitos síncronos com sinal de relógio livre, a uma frequência mínima de funcionamento do circuito de 320 KHz. Esta

frequência garante que, em qualquer circunstância, a replicação ocorre sem problemas, mesmo que se opte pela frequência máxima possível para o relógio de teste, 33 MHz.



**Figura 5.7:** Implementação numa *slice* do bloco auxiliar de replicação

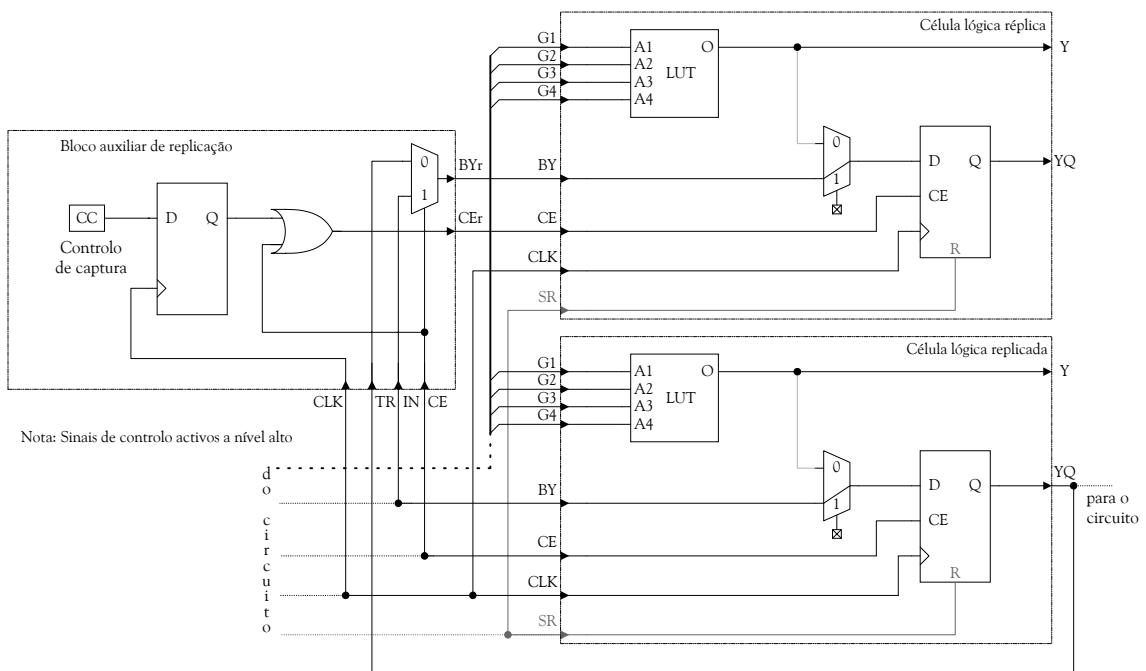
Sendo cada CLB constituído por quatro células lógicas e necessitando cada célula lógica de um bloco auxiliar de replicação, a solução proposta na Figura 5.7, que ocupa um *slice*, exige o uso de dois CLBs para a sua implementação.

Em resumo, o bloco de replicação é responsável, durante o processo de replicação, pela colocação à entrada do elemento de retenção (neste caso um *flip-flop*) da célula réplica do valor actualizado que este deverá guardar, quer esse valor seja proveniente do elemento de retenção da célula replicada, quer advenha da lógica interna (a montante) da aplicação implementada.

Atente-se que, neste caso, mesmo que a tabela de consulta seja utilizada para a implementação de uma qualquer função lógica não registada, a metodologia seguida continua a ser válida, pois a tabela funciona como um bloco combinatório independente. Deste modo, as suas entradas devem ser colocadas em paralelo, juntamente com as restantes entradas não reencaminhadas para o bloco auxiliar de replicação, e posteriormente também as suas saídas, juntamente com as restantes saídas da célula lógica. A Figura 5.8 ilustra um destes casos.

Para o caso seis a solução é em tudo idêntica, como ilustra a Figura 5.9, alterando-se o ponto de entrada para o *flip-flop*, que deixa de ser directo a partir da entrada BY para passar a ser a saída da

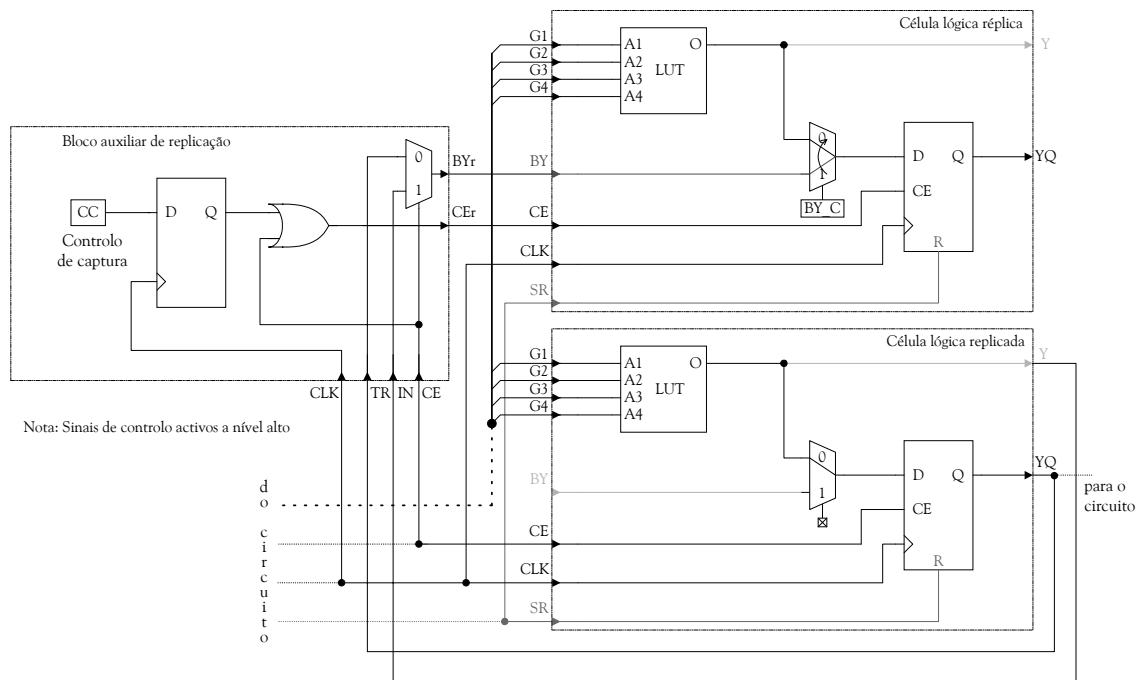
tabela de consulta, a qual implementa uma função combinatória. É esse sinal, recolhido na saída Y da célula lógica replicada, que deve agora ser aplicado à entrada IN do bloco auxiliar de replicação. Atente-se no facto de que o acesso a esse valor a partir da saída Y da célula replicada não influí na funcionalidade normal da célula, esteja ou não esta saída a ser usada. Em alternativa, o valor pode igualmente ser encaminhado a partir da saída Y do próprio bloco lógico réplica, uma vez que ambas as tabelas possuem já, na altura da transferência do estado, a mesma funcionalidade. Esta alternativa, como será visto mais tarde, tem vantagens no caso de o valor de estado presente no *flip-flop* replicado não ser o correcto, devido à existência de uma falta na tabela de consulta. A saída do *flip-flop* da célula lógica replicada é ligada à entrada TR do mesmo bloco, tal como no caso anterior.



**Figura 5.8:** Replicação de uma célula contendo um registo simples com sinal de habilitação de relógio e com sinal de inicialização síncrono ou assíncrono e um bloco combinatório independente

A principal diferença em relação ao caso cinco verifica-se na forma como a saída do multiplexador, a cujas entradas estes sinais são ligados, é encaminhada dentro da célula lógica réplica para a entrada do *flip-flop*. O acesso é efectuado através da entrada BY. Visto que, no caso desta implementação, o valor de estado é fornecido ao elemento de retenção a partir da tabela de consulta e não directamente do exterior, esta linha encontra-se livre, conclusão retirada das observações efectuadas à implementação dos circuitos definidos nas “ITC’99 benchmarks”. Esta entrada compete com a saída da tabela de consulta no acesso à entrada do *flip-flop* através de um multiplexador programável. Em condições normais de funcionamento será a saída da tabela de consulta a estar ligada à entrada do *flip-flop*, mas, durante o processo de transferência do estado,

terá de ser a entrada BY a aceder à entrada do *flip-flop*. Desta forma, tem igualmente de alterar-se o valor presente na célula de selecção desse multiplexador (BY\_C), controlada através da memória de configuração, durante o intervalo de transferência. A activação do sinal de controlo de captura habilitará o relógio do *flip-flop* réplica, que capturará o valor do *flip-flop* replicado ao primeiro impulso de relógio. Após a captura, é desactivado o sinal de controlo de captura e o multiplexador reencaminha a saída da tabela de consulta para a entrada do *flip-flop* réplica, sendo posteriormente colocada em paralelo a entrada CE de ambas as células. Depois, são desligados todos os sinais de e para o bloco auxiliar de replicação e colocadas em paralelo as saídas, completando-se o processo de replicação. As diferentes fases deste processo estão esquematizadas no fluxograma da Figura 5.10.

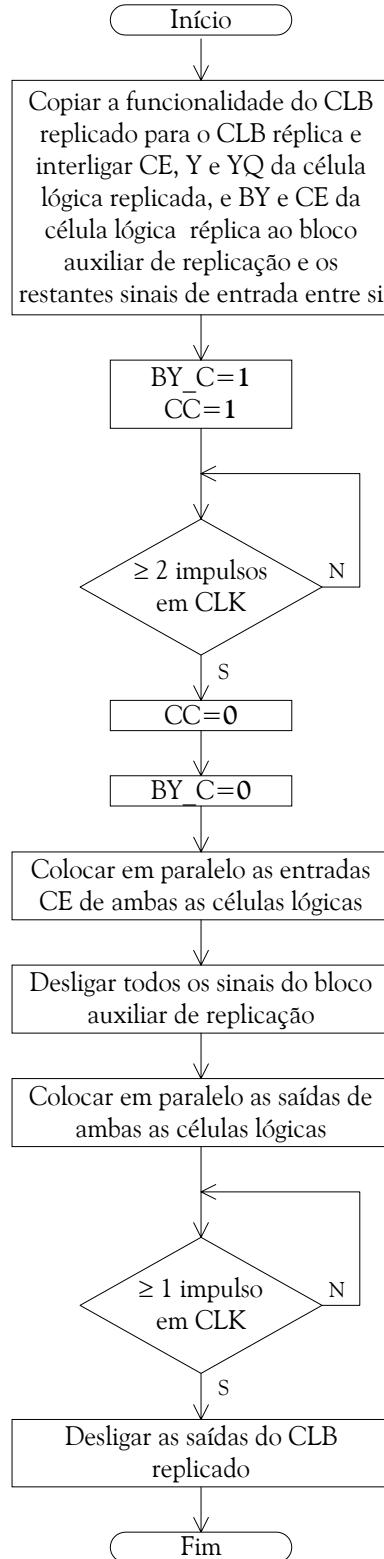


**Figura 5.9:** Replicação de uma célula contendo lógica combinatória registada com sinal de habilitação de relógio e com sinal de inicialização síncrono ou assíncrono

A activação do sinal de habilitação de relógio (CE) pela aplicação implementada, durante qualquer uma das fases do processo, colocará na entrada do *flip-flop* da célula réplica o valor presente na entrada do *flip-flop* replicado, capturando ambos esse valor ao primeiro impulso de relógio. De notar que, se essa captura se efectuasse sempre a partir da saída do *flip-flop* replicado, a actualização do *flip-flop* réplica sofreria um atraso de um impulso de relógio, o que poderia conduzir a situações de incoerência de valores caso CE e CC mudassem de valor dentro do mesmo período do relógio do sistema, logo após uma actualização do valor de estado.

Em ambos os casos, a existência ou não de um sinal de inicialização, quer este seja síncrono ou assíncrono, é irrelevante para o processo de replicação, visto ambas as células lógicas envolvidas

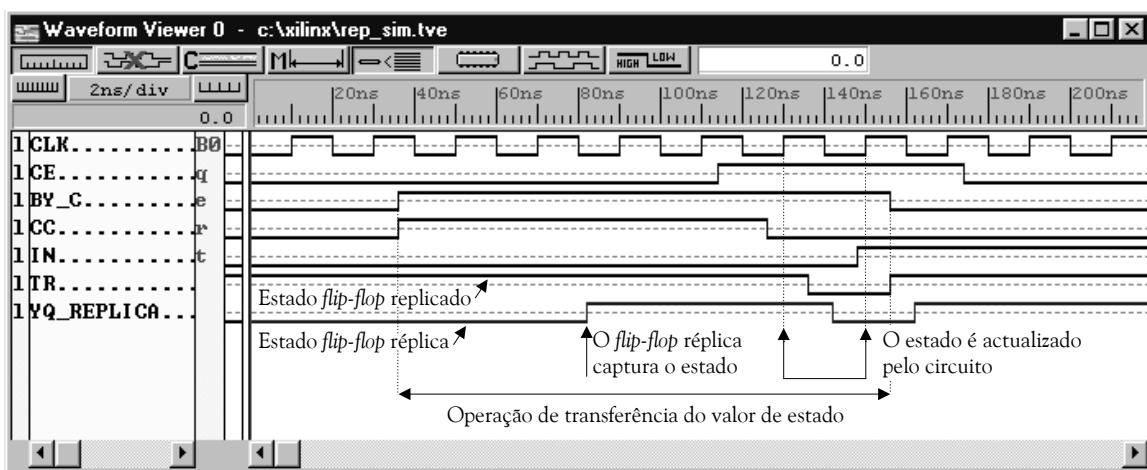
terem essa entrada em paralelo desde o seu início. Nesse sentido, a sua activação a qualquer instante afectar-las á simultaneamente.



**Figura 5.10:** Fluxograma do processo de replicação de uma célula contendo lógica combinatória registada com sinal de habilitação de relógio e com sinal de inicialização síncrono ou assíncrono

A existência de múltiplos sinais de relógio, de diferente fase ou frequência (desde que não inferior ao valor mínimo de 320 KHz), não coloca qualquer obstáculo à execução do processo de replicação da forma descrita, uma vez que o sinal de relógio é único dentro de cada *slice* e, sendo a replicação efectuada *slice a slice*, apenas um dos sinais de relógio estará envolvido a cada momento no processo.

Uma das grandes vantagens deste bloco auxiliar de replicação é a simplicidade da sua implementação e da sua utilização, pois a transferência pode ser executada a qualquer momento sem que isso perturbe o funcionamento da lógica e sem que haja necessidade de qualquer mecanismo auxiliar de verificação do seu estado, garantindo-se a coerência dos valores entre *flip-flops* do CLB replicado e da sua réplica no final do processo. A Figura 5.11 ilustra a simulação de um processo de replicação em que, durante e após a operação de transferência, ocorrem actualizações do estado por parte do sistema, mantendo-se sempre a coerência nos valores de estado presentes no *flip-flop* replicado e na sua réplica, apesar do atraso que se observa entre a actualização dos dois valores. Esse atraso, que se deve a diferentes tempos de propagação, é contudo muito inferior a um ciclo de relógio.



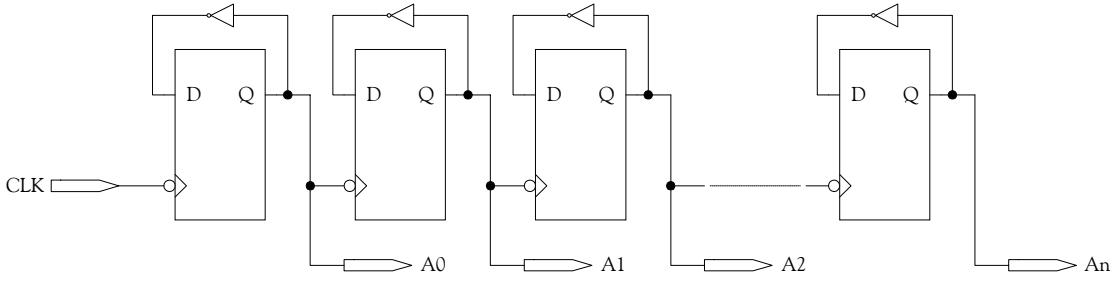
**Figura 5.11:** Simulação de uma operação de transferência e actualização durante o processo de replicação

### 5.2.3 CIRCUITOS ASSÍNCRONOS

Na análise dos circuitos assíncronos temos de distinguir entre dois tipos diferentes: os circuitos assíncronos implementados com *flip-flops* e apresentando múltiplos ‘pseudo’ sinais de relógio e os circuitos assíncronos implementados com *latches* e, obviamente, sem sinal de relógio.

Um exemplo típico de circuitos assíncronos com múltiplos ‘pseudo’ sinais de relógio é o do contador binário assíncrono (*ripple counter*), esquematizado na Figura 5.12. Neste circuito, o ‘pseudo’ sinal de relógio para o próximo *flip-flop* na cadeia é obtido a partir da saída do *flip-flop* anterior, pelo que a frequência do sinal de relógio é dividida por dois de andar para andar. Desde que o valor mínimo

da frequência do ‘pseudo’ sinal de relógio (o do bit mais significativo) não seja inferior a 320 KHz, o processo de replicação pode ser executado sem problemas da mesma forma descrita para o caso da replicação de circuitos síncronos com sinal de relógio livre.



**Figura 5.12:** Contador binário assíncrono (*ripple counter*)

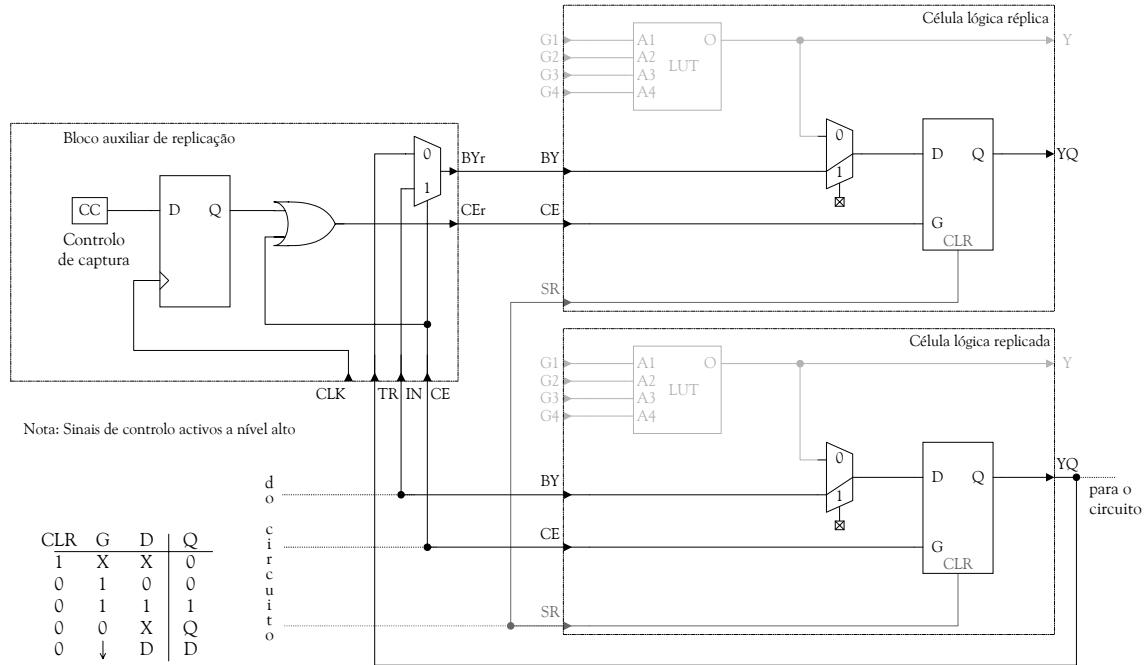
No caso dos circuitos assíncronos que empregam *latches* em vez de *flip-flops*, o facto de a sua resposta ser ao patamar e não existir sinal de relógio possibilita a sua replicação sem qualquer problema, usando-se exactamente o mesmo bloco de replicação utilizado para o caso dos circuitos síncronos com sinal de habilitação de relógio. Esta situação está ilustrada nas Figura 5.13 e Figura 5.14, respectivamente para blocos em que apenas é usada a *latch* e para blocos em que a entrada provém de um bloco lógico combinatório implementado na tabela de consulta da própria célula lógica a replicar. Em ambas as figuras, a *latch* aparece implementada apenas com uma linha de inicialização assíncrona (CLR), embora o processo se aplique de igual modo no caso da existência conjunta ou em alternativa de uma linha de *preset* (PRE). Os fluxogramas de implementação do procedimento de replicação são os mesmos que foram apresentados na Figura 5.6 e na Figura 5.10, respectivamente.

A manutenção no bloco auxiliar de replicação de um *flip-flop* serve apenas para simplificar o projecto, ao generalizar para todo o tipo de implementações o mesmo bloco. No entanto, se a aplicação for puramente assíncrona, isto é, se não existir nenhum relógio no sistema, o *flip-flop* pode ser substituído por uma *latch* transparente com a entrada G ligada à massa, sem qualquer alteração da sequência do processo de replicação.

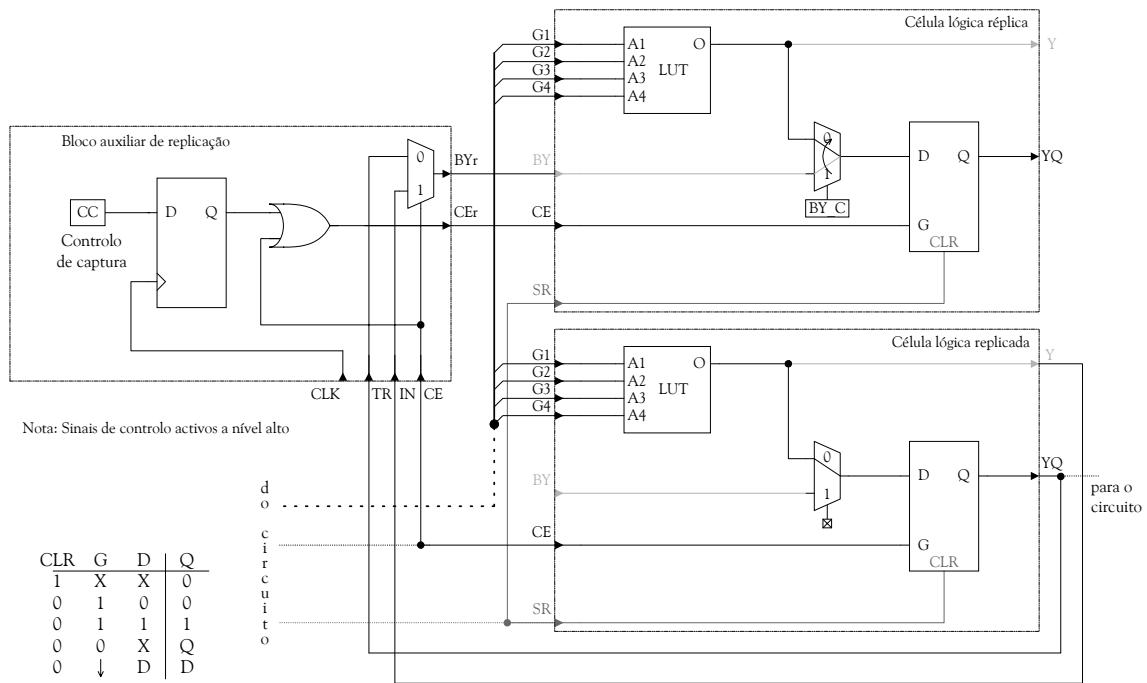
### 5.3. INTERLIGAÇÕES

A replicação de interligações apresenta-se como um processo bastante mais simples, em que é necessário assegurar apenas que as linhas activas que forem replicadas não sejam momentaneamente interrompidas durante o seu decurso. De notar que esta preocupação está já patente na replicação de blocos lógicos, pois, nesse caso, não só é necessário proceder à replicação do bloco em si, mas também ao reencaminhamento das linhas que a ele chegam e que dele partem.

Nesse sentido, pode afirmar-se que, abstraindo o atravessamento do bloco lógico pelos sinais, existe já uma replicação de linhas activas.



**Figura 5.13:** Replicação de uma célula contendo uma *latch* simples com sinal de habilitação de entrada e com sinal de inicialização assíncrono



**Figura 5.14:** Replicação de uma célula contendo lógica combinatória e uma *latch* com sinal de habilitação de entrada e com sinal de inicialização assíncrono

Assim, os problemas então enunciados — a possível interrupção temporária de linhas activas e o aparecimento de transitórios provocados pela alteração dinâmica da configuração da FPGA — e as suas causas, aplicam-se integralmente ao caso da replicação de linhas activas. De igual modo, a solução proposta é a divisão em duas fases do processo de replicação:

**1<sup>a</sup> fase** configuração de um caminho alternativo em paralelo com aquele que se pretende replicar;

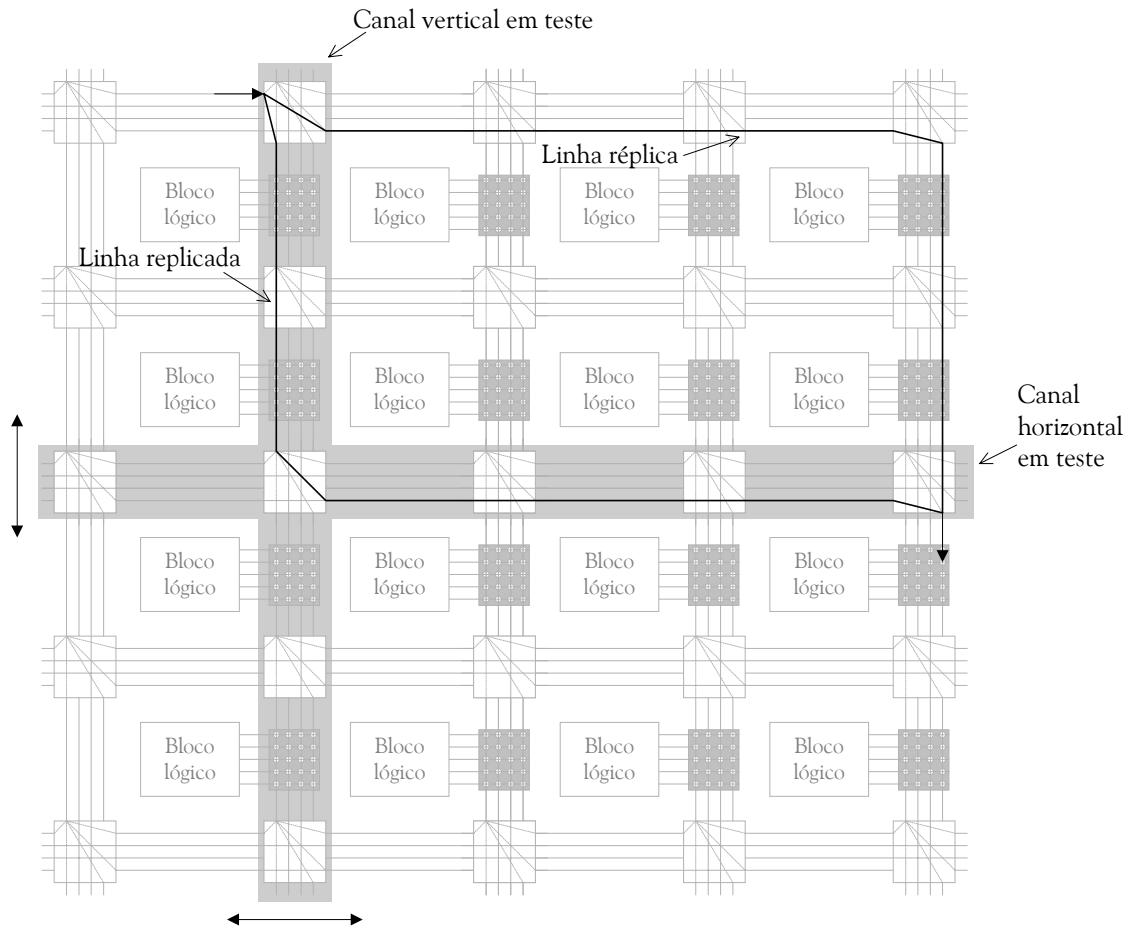
**2<sup>a</sup> fase** libertação do caminho replicado.

A Figura 5.15 ilustra a libertação para o teste de uma interligação que atravessa os canais verticais e horizontais em teste. A interligação é replicada nessa parte por outra que se encaminha através de recursos situados fora dessa zona. A aplicação continuada deste processo a todas as linhas situadas dentro dos canais em teste e o varrimento de todos eles, permite o teste da totalidade dos recursos de interligação global. É importante referir que não é possível libertar simultaneamente para o teste todos os recursos de encaminhamento situados dentro dos canais de teste, quer devido ao custo da replicação de todas as linhas e do seu teste em simultâneo, quer porque os sinais de entrada ou saída nos CLBs periféricos a esses canais têm, obrigatoriamente, que ser conduzidos através de recursos de encaminhamento neles situados. A libertação é, por isso, efectuada de forma faseada, substituindo e testando, em várias etapas, os diferentes recursos.

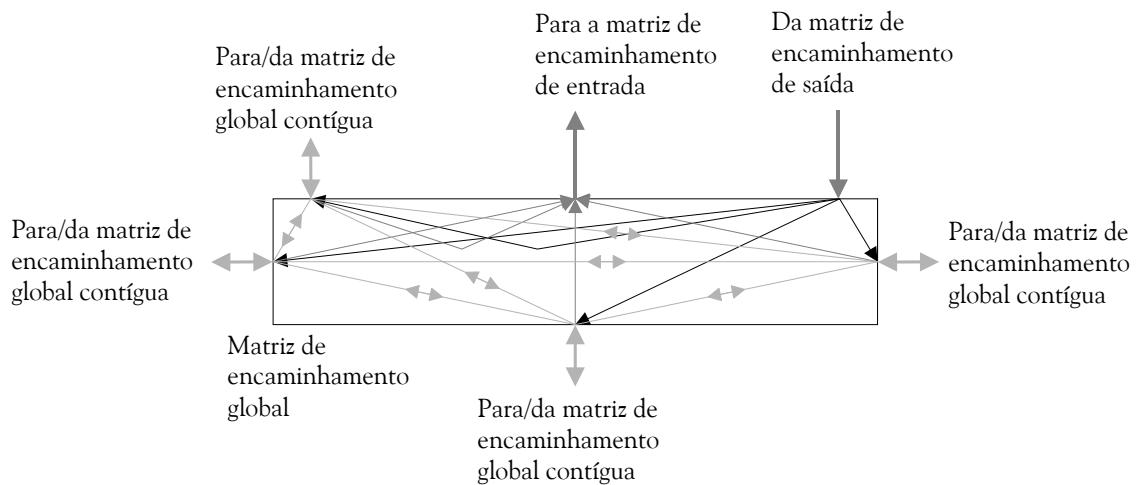
Enquanto cada canal vertical possui, em termos práticos, um total de 96 linhas, cada canal horizontal tem 60 linhas. Na realidade, cada linha é constituída por um ou mais segmentos que, para efeitos de teste, se interligam, sempre que possível, de modo a abrangerem a totalidade da linha ou da coluna e dessa forma diminuir o tempo de teste. Seria igualmente possível ligar várias dessas linhas, inclusive criando uma única, mas tal reduziria a capacidade de detecção de faltas, tornando, por exemplo, redundantes algumas faltas em ponte [Renovell et al., 98c].

A interligação dos segmentos nas matrizes de encaminhamento global tem também como objectivo o teste dos pontos de interligação programável que possibilitam o encaminhamento dos sinais. Desde logo, o teste da totalidade destes pontos de interligação em cada matriz não seria possível com apenas uma configuração, pelo que, mesmo que fosse exequível o teste simultâneo de todas as linhas (entendidas como conjuntos de um ou mais segmentos interligados para formarem uma linha que se estende vertical e/ou horizontalmente ao longo de toda a matriz lógica programável), seria sempre necessário o recurso a várias configurações de teste para se cobrir a totalidade dos pontos de interligação programável. A Figura 5.16 esquematiza as diferentes possibilidades de encaminhamento dos sinais, consoante a sua origem e destino dentro da matriz de encaminhamento global. De referir que, na replicação e teste de interligações, o objectivo é apenas

cobrir as interligações globais, uma vez que as interligações locais para e das matrizes de encaminhamento local, entrada ou saída, são cobertas pelo teste do próprio CLB.



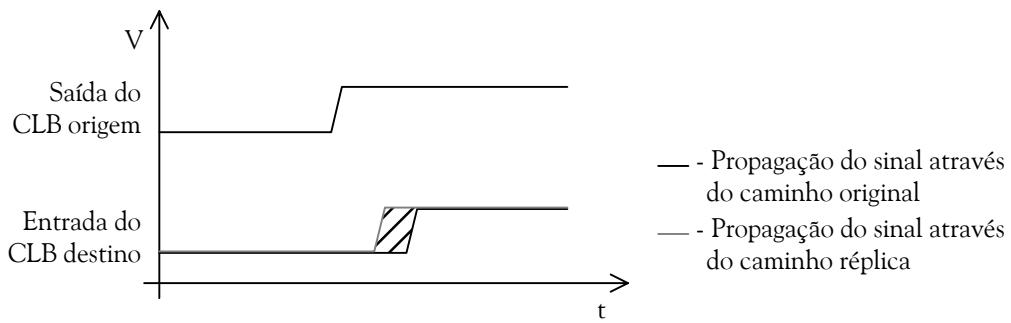
**Figura 5.15:** Replicação de uma interligação



**Figura 5.16:** Recursos de interligação nas matrizes de encaminhamento global

O estabelecimento, na replicação de interligações activas, de um caminho réplica em paralelo com o caminho original, usando diferentes recursos de encaminhamento, reflecte-se no tempo de

propagação dos sinais envolvidos. Durante um certo intervalo de tempo coexistem dois caminhos que embora conduzam o mesmo sinal entre a mesma saída do CLB origem e a mesma entrada do CLB destino, apresentam tempos de propagação diferentes. Isto significa que, quando o nível de tensão na saída do CLB origem muda, o sinal à entrada do CLB destino exibe um intervalo de indeterminação, conforme ilustrado na Figura 5.17. Contudo, a impedância dos PIPs ao longo dos dois caminhos limita o valor da corrente na interligação, e, por isso, este comportamento não causa qualquer dano à FPGA. No entanto, numa análise ao comportamento transitório do circuito deve considerar-se para o atraso de propagação o maior dos dois caminhos [Gericota et al., 03a].



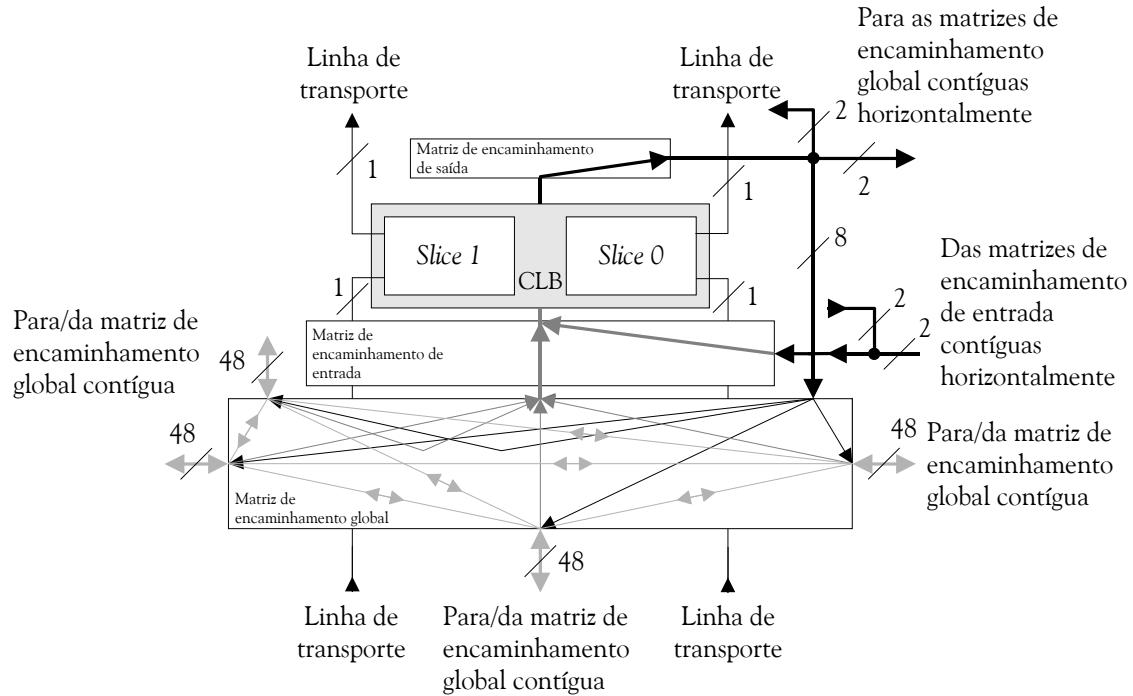
**Figura 5.17:** Atraso de propagação durante a replicação de recursos de interligação

#### 5.4. RECUPERAÇÃO DE ERROS E RESTRIÇÕES

A colocação em paralelo das entradas e saídas do CLB replicado e do CLB réplica não levanta qualquer tipo de problema desde que o comportamento de ambos os CLBs seja idêntico. A sequência natural do processo de teste implica que o CLB onde vai ser configurada a réplica seja primeiramente testado. Depois de se ter verificado que esse CLB não apresenta qualquer defeito, inicia-se o processo de replicação. O objectivo de replicar nesse CLB a configuração doutro CLB é libertar este último para ser testado, sem afectar o funcionamento do sistema. Tal significa que, aquando da colocação em paralelo dos dois CLBs, apenas temos a garantia de que um deles não apresenta nenhum defeito, aquele que foi testado e onde a réplica é configurada. Por sua vez, o CLB replicado, que ainda não foi testado, poderá encontrar-se com defeito. No caso de esse defeito originar valores diferentes nas suas entradas ou saídas, a colocação em paralelo dos dois CLBs poderá conduzir à interligação de linhas com valores lógicos diferentes. As consequências de tal situação para o funcionamento do circuito dependem de vários factores, tais como a direccionalidade dos recursos de encaminhamento e a impedância das interligações.

#### 5.4.1 DIRECCIONALIDADE DOS RECURSOS DE ENCAMINHAMENTO

Cada CLB possui três matrizes de encaminhamento associadas: duas locais (entrada e saída) e uma global. Os recursos de encaminhamento nessas matrizes podem ser unidireccionais ou bidirecionais, conforme esquematizado na Figura 5.18. Com excepção de dois pares de linhas que da matriz de encaminhamento de saída ligam directamente à matriz de encaminhamento de entrada dos CLB contíguos horizontalmente, um par para cada, e da linha de transporte que interliga cada slice à slice homóloga situada no CLB imediatamente acima, não existem outros recursos de encaminhamento que das matrizes de encaminhamento local permitam estabelecer interligações com outros CLBs sem passar pela matriz de encaminhamento global. Desta forma, as interligações que colocam entradas e saídas em paralelo durante o processo de replicação têm de ser estabelecidas através da matriz de encaminhamento global.

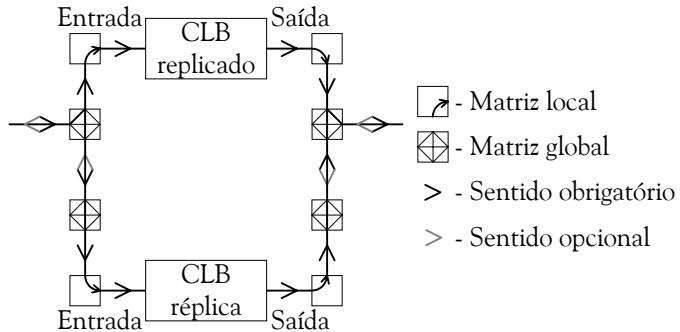


**Figura 5.18:** Recursos de encaminhamento associados a um CLB

Como se observa na Figura 5.18, apenas recursos de interligação unidireccionais estão disponíveis para o estabelecimento de interligações entre a matriz global e as matrizes locais. Por outro lado, os recursos que permitem a interligação entre matrizes globais são em parte bidirecionais e em parte unidireccionais.

Conforme ilustrado na Figura 5.19, o estabelecimento do paralelo entre as entradas durante o processo de replicação efectua-se através de:

- recursos unidireccionais entre as matrizes globais associadas aos dois CLBs envolvidos, replicado e réplica, e as respectivas matrizes locais de entrada;
- recursos unidireccionais (no sentido das entradas do CLB replicado para as do CLB réplica) ou bidireccionais entre as matrizes globais.



**Figura 5.19:** Colocação em paralelo das ligações aos CLBs durante a replicação

No que diz respeito às saídas, as interligações entre as matrizes globais podem igualmente ser unidireccionais, das saídas do CLB réplica para as saídas do CLB replicado, ou bidireccionais, conforme se observa na mesma figura. De outro modo, uma vez que os sinais não se propagam em sentido inverso nas ligações unidireccionais, não atingem a entrada do CLB réplica nem as saídas de ambos são colocadas em paralelo. Como resultado, não se estabelece o paralelo e os sinais não se propagam para o resto do circuito após o CLB replicado ser desligado do sistema, pelo que o funcionamento normal seria interrompido.

Este conjunto de restrições não traz só desvantagens, mas apresenta também vantagens a nível da propagação e correção de erros. Uma falta presente na entrada do CLB replicado não se propaga à entrada do CLB réplica, devido à unidirecionalidade da interligação entre matriz global e local. Desta forma, como as suas entradas não são afectadas pela interligação, mesmo que no CLB replicado exista uma falta, as entradas do CLB réplica exibirão sempre os valores correctos.

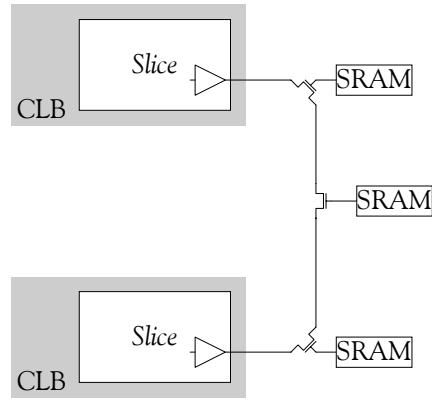
Esta simples análise permite verificar que existe um risco real de, durante o processo de interligação, em consequência de uma falta que afecte o CLB replicado, se interligarem entradas ou saídas com níveis de tensão diferentes, tornando necessário perceber qual o comportamento da interligação e quais as suas consequências.

#### 5.4.2 IMPEDÂNCIA DAS INTERLIGAÇÕES

Para se proceder à análise do comportamento do paralelo na presença de um defeito, considerou-se um modelo de faltas do tipo sempre-a, pelo que, na eventualidade da sua existência, uma das entradas ou uma das saídas do CLB replicado se encontra a um nível lógico fixo (0 ou 1). Esta

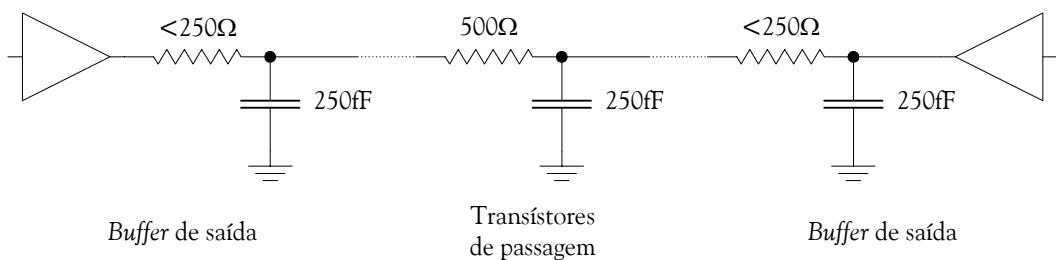
situação leva a que, ao estabelecer-se o paralelo dos dois CLBs com a mesma configuração, mas em que um deles apresenta um defeito, se crie um conflito entre saídas ou se forcem diferentes valores lógicos nas entradas. No entanto, há que notar que uma interligação interna numa FPGA não tem o mesmo significado eléctrico que se lhe atribui, por exemplo, numa carta de circuito impresso. Neste último caso, a impedância da ligação é praticamente nula, enquanto no caso das FPGAs a impedância da interligação decorre do somatório das impedâncias dos PIPs que a estabelecem, o que, em caso de conflito, limitará o valor da corrente.

A topologia da interligação entre as saídas de dois CLBs baseia-se na existência de segmentos de encaminhamento curtos, interligados por meio de PIPs, constituídos por transístores de passagem controlados através da memória de configuração, conforme se representa na Figura 5.20.



**Figura 5.20:** Topologia da interligação entre as saídas de dois CLBs

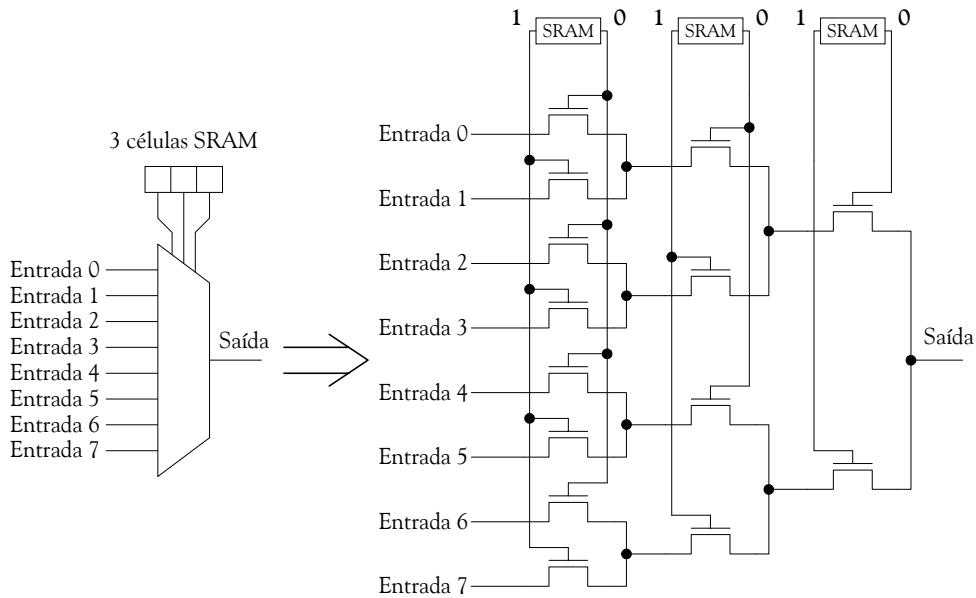
O circuito eléctrico equivalente desta topologia encontra-se representado na Figura 5.21, onde os valores indicados são típicos e retirados de [Betz et al., 99]. Atente-se na existência, em cada uma das saídas, de um *buffer*, que em conjunto com os transístores de passagem, vai limitar o valor da corrente na interligação e contribuir para a divisão de tensão aquando da presença de valores lógicos diferentes em ambas as saídas.



**Figura 5.21:** Circuito eléctrico equivalente que decorre da interligação entre as saídas de dois CLBs

Cada vez que uma linha atravessa uma das matrizes de encaminhamento, em qualquer das direcções, verifica-se a passagem por um ou mais PIPs, e, consequentemente, o atravessamento de

uma ou mais impedâncias. O valor das impedâncias ao longo das diferentes matrizes de encaminhamento é variável, dependendo do tipo de implementação. Por exemplo, à saída de cada CLB, a matriz de encaminhamento local é constituída por oito multiplexadores controlados pela memória de configuração. Cada saída dos multiplexadores pode seleccionar várias saídas do CLB e conduzir esses sinais para os recursos de encaminhamento globais. Uma possível implementação para este tipo de multiplexadores é apresentada na Figura 5.22.

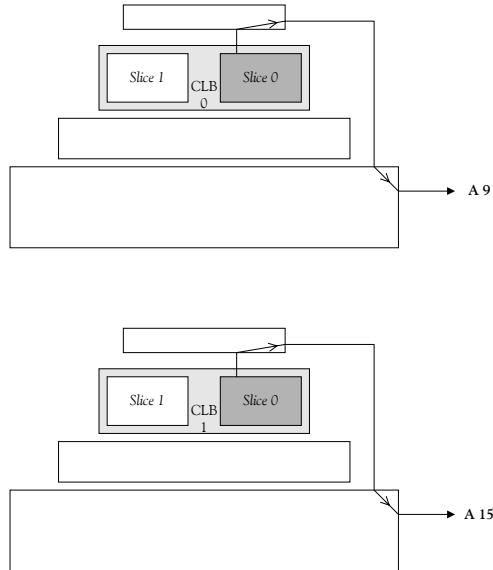


**Figura 5.22:** Esquema eléctrico de um multiplexador de 8 entradas controlado pela memória de configuração

Com o objectivo de validar a análise teórica, realizou-se um primeiro conjunto de ensaios experimentais para se comprovar o comportamento do circuito no caso da interligação de saídas com valores lógicos diferentes e aferir da eventual existência de danos para o circuito. Sendo impossível medir os sinais directamente nos pontos de interligação, estes foram reencaminhados para o exterior, o que implicou a existência de cargas suplementares decorrentes da introdução da derivação. No entanto, tal situação aproxima-se das condições reais de funcionamento, pois também nas implementações reais existirão derivações a partir das interligações dos dois CLBs envolvidos na replicação. Outra das consequências é que, devido ao facto de atravessar vários *buffers*, ao exterior chegam sempre níveis lógicos bem definidos, independentemente dos valores intermédios que a tensão possa apresentar ao longo da interligação, resultado do divisor de tensão criado. Da análise do comportamento, em termos de nível lógico, das formas de onda da tensão em diferentes pontos de medida é possível aferir do seu desenvolvimento ao longo da interligação.

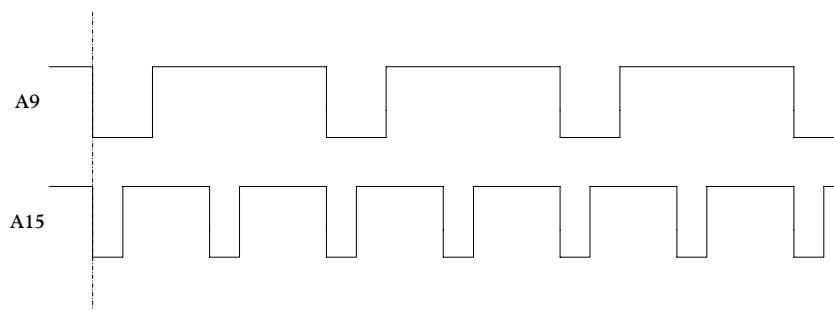
A representação esquemática presente na Figura 5.23 ilustra as ligações dos dois CLBs utilizados na experiência e mostra a colocação dos pontos de medida. As indicações A9 e A15 correspondem à designação dos pinos exteriores para onde esses sinais foram encaminhados. As “caixas” a branco

representam as matrizes de encaminhamento local e global, onde se encontram os PIPs que permitem reencaminhar os sinais ao longo dos recursos de interligação disponíveis. As linhas a negro representam as ligações por onde os sinais são propagados.



**Figura 5.23:** Representação esquemática das ligações e dos pontos de medida antes da interligação das saídas

No primeiro caso, o da Figura 5.23, apenas existem interligações entre a saída de cada um dos CLBs e os pontos de medida, representando-se o comportamento dos CLBs antes da interligação das saídas<sup>7</sup>. Na Figura 5.24 estão representadas as formas de onda de saída de cada um dos CLBs. Estes sinais são diferentes em cada saída, o que possibilita analisar o comportamento do circuito ao efectuar-se a sua interligação.



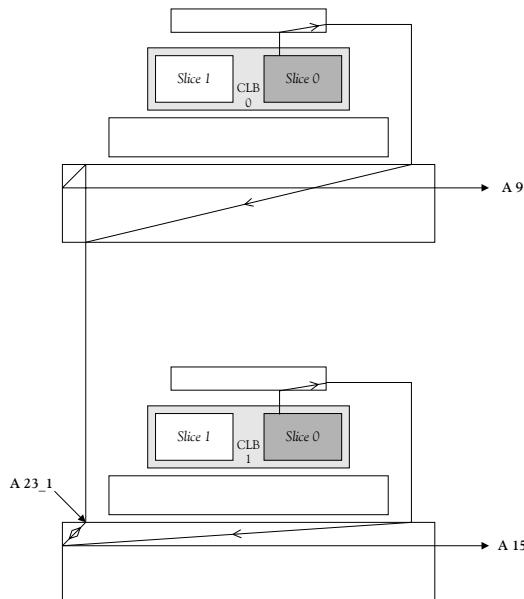
**Figura 5.24:** Diagrama de tensões nos pontos referenciados na Figura 5.23<sup>8</sup>

<sup>7</sup> Para a análise que se pretende efectuar, o circuito implementado para se proceder aos ensaios de interligação das saídas é irrelevante, pelo que não é descrito.

<sup>8</sup> Os diagramas de tensão representados são baseadas nas formas de onda de tensão recolhidas durante a fase experimental e que são apresentadas em anexo a esta tese.

O conjunto de experiências realizadas difere entre si no número de segmentos e no encaminhamento da linha que efectua a interligação entre as saídas dos dois CLBs, bem como nos pontos onde se visualizou a forma de onda.

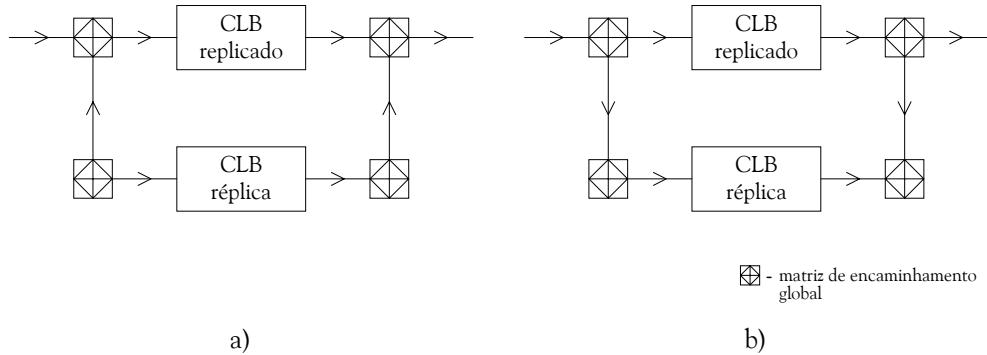
A primeira experiência apresenta uma interligação com seis segmentos, como se ilustra na Figura 5.25. Cada segmento é ligado ao seguinte através de um ou mais PIPs, em cada uma das matrizes de encaminhamento que atravessa. A série de impedâncias apresentada pelos PIPs comporta-se como um divisor de tensão quando o valor lógico é diferente nas duas saídas interligadas, limitando a corrente entre elas. Este comportamento é muito importante, porque devido a esta estrutura da interligação é possível ligar duas saídas que apresentem níveis lógicos diferentes, sem que isso danifique a FPGA.



**Figura 5.25:** Representação esquemática das ligações e dos pontos de medida

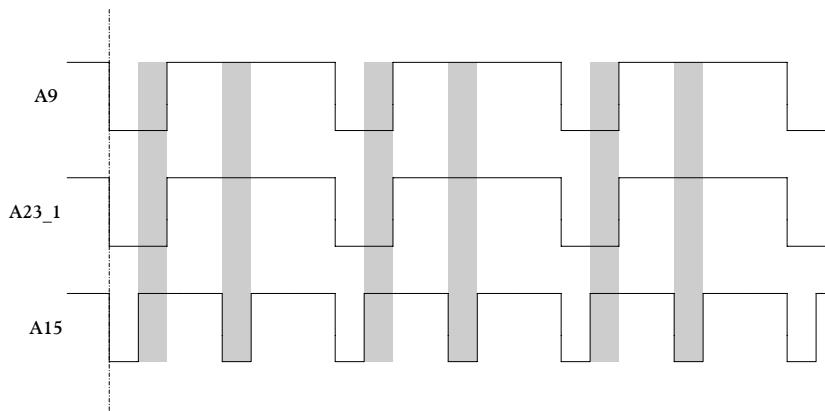
Outra característica das matrizes de encaminhamento é a direccionalidade dos PIPs, assinalada na figura através de pequenas setas ao longo das ligações nas matrizes de encaminhamento. Como se pode observar, a ligação até aos pontos de medida A9 e A15 (as saídas do sinal proveniente de cada CLB) é unidireccional, isto é, a interligação entre as saídas do CLB e a saída da matriz de encaminhamento global (ponto onde o sinal é derivado para o resto do sistema) é unidireccional. A interligação estabelecida entre os pontos de derivação de cada um dos sinais, destinada a realizar o paralelo entre os dois CLBs, foi efectuada, no primeiro conjunto de ensaios, através de PIPs bidirecionais, caso contrário, o sentido do sinal na interligação seria único. Este facto originaria alguns problemas ao funcionamento do paralelo, como facilmente se comprehende pela análise da Figura 5.26. No caso a), a unidireccionalidade da linha que realiza o paralelo da entrada inviabiliza que os sinais presentes na entrada do CLB replicado atinjam o CLB réplica, pelo que as entradas do

CLB réplica se mantêm, na prática, desligadas do circuito. No caso b), a unidireccionalidade da linha que interliga a saída do CLB réplica com a do CLB replicado implica que os sinais presentes na saída do primeiro não atinjam o ponto de derivação, pelo que, depois de se reconfigurar para o teste o CLB replicado, este ponto fica sem sinal.



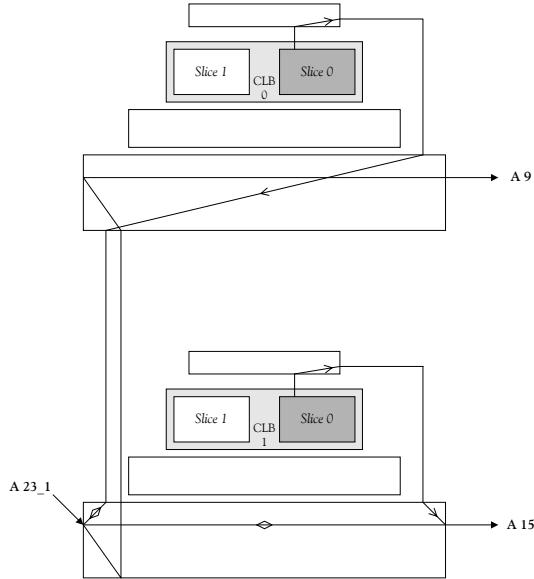
**Figura 5.26:** Paralelo unidireccional dos CLBs envolvidos na replicação

A Figura 5.27 ilustra as formas de onda da tensão nos pontos considerados. A sombreado estão indicadas as zonas em que o nível lógico nas duas saídas é diferente. De notar que, embora no esquema da Figura 5.25 estejam referenciados três pontos de medida, os pontos A9 e A23\_1 são coincidentes do ponto de vista eléctrico, uma vez que se despreza, dada a diferença de grandeza relativamente à impedância dos transístores de passagem, a impedância das linhas de encaminhamento. Esta assunção é verificada correctamente ao longo de todo o ensaio. Uma vez que o sinal, para poder ser lido no exterior, atravessa vários *buffers*, o valor lido da tensão apresenta um nível lógico bem definido, pelo que é através do comportamento das ondas em função dos valores lógicos conhecidos ao longo da interligação das saídas dos dois CLBs que se extrapola o comportamento interno do divisor de tensão. Desta forma, observa-se que a forma de onda em A15 é igual à tensão à saída do CLB antes da interligação, o mesmo se passando com a forma de onda em A9.



**Figura 5.27:** Diagrama de tensões nos pontos referenciados na Figura 5.25

Procedeu-se a uma segunda experiência que consistiu no estudo do comportamento da interligação, no caso do circuito cujo esquema se ilustra na Figura 5.28. Repare-se que agora o número de segmentos envolvidos na interligação das duas saídas é de sete.

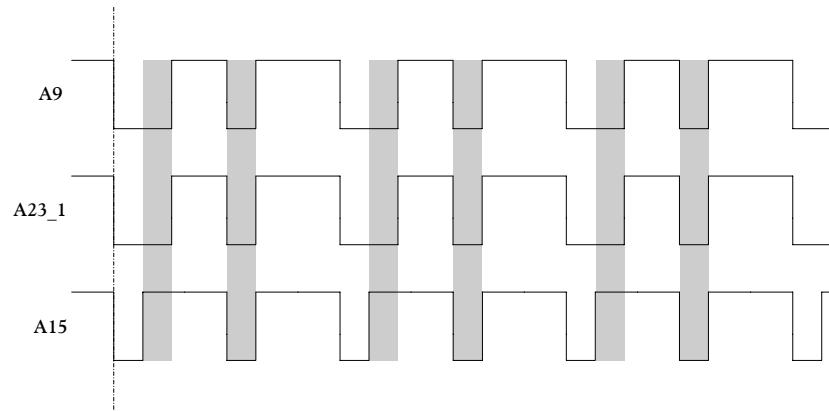


**Figura 5.28:** Representação esquemática das ligações e dos pontos de medida

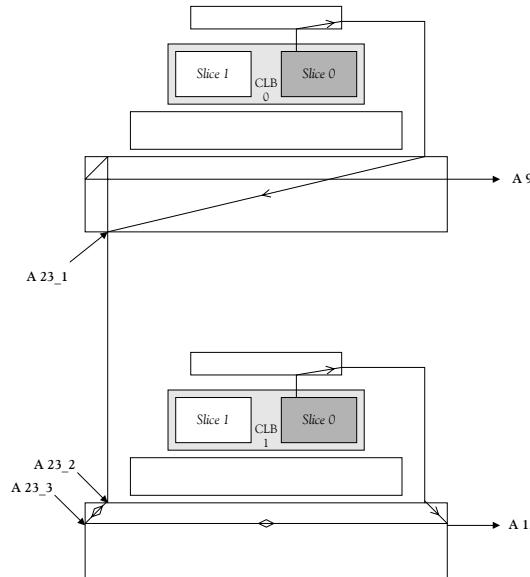
Neste caso, existem igualmente dois pontos de medida que consideramos electricamente iguais, A9 e A23\_1. No entanto, ao contrário do que se passava anteriormente, a forma de onda em A9 é diferente da que se obtinha quando não existia interligação entre as saídas, mantendo-se invariável a forma de onda em A15. Pelo comportamento da tensão infere-se que, no caso de a saída do CLB 0 se encontrar num nível lógico baixo e a do CLB 1 a nível lógico alto (primeira zona sombreada na Figura 5.29), o comportamento das ondas medidas em A9 e em A15 é idêntico ao verificado antes da interligação das saídas. Contudo, no caso inverso (segunda zona sombreada na Figura 5.29), a tensão no ponto A9 é afectada pela interligação, ou seja, a tensão é forçada a um nível lógico baixo. Esta diferença de comportamento é facilmente compreendida se alterarmos os pontos em que efectuamos as medições, embora sem alterarmos o encaminhamento da interligação entre as duas saídas, como se esquematiza na Figura 5.30.

A inserção de mais um ponto de medida permite uma análise mais precisa da divisão da tensão na interligação e do comportamento do sinal ao longo dela. De notar que A9, A23\_1 e A23\_2 são electricamente o mesmo ponto. A observação das formas de onda apresentadas na Figura 5.31 torna perceptível um comportamento diferenciado da forma de onda em A23\_3, consoante a direcção da diferença de nível lógico entre as duas saídas. Essa diferença deve-se à não simetria, em relação à tensão de alimentação, do valor da tensão de transição entre níveis lógicos. A Figura 5.32 ilustra essa não simetria, comparando as duas situações possíveis, saída do CLB 0 a nível lógico 0 e

saída do CLB 1 a nível lógico 1 e vice-versa, e colocando em evidência a existência de uma zona de transição em que o nível lógico 0 predomina sobre o nível lógico 1. Assim sendo, como se observa na Figura 5.31, um nível lógico 0 no CLB 0 e um nível lógico 1 no CLB 1 resulta num nível lógico 0 em A23\_3. De igual forma, mas com valores lógicos inversos nos CLBs, o nível lógico em A23\_3 é também 0. Conclui-se assim que, embora as formas de onda em A9 e em A15 se mantenham inalteráveis relativamente ao comportamento antes da interligação, sinal de que estes pontos se situam fora da zona de transição, A23\_3 sofre influência de ambas as saídas, evidenciando a existência dessa zona ao apresentar uma tensão nula sempre que o nível lógico à saída de qualquer um dos dois CLBs envolvidos é 0.



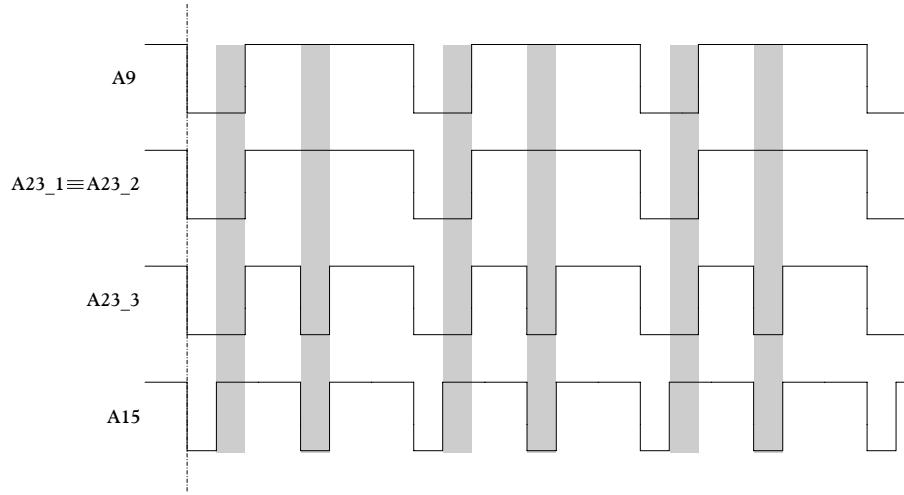
**Figura 5.29:** Diagrama de tensões nos pontos referenciados na Figura 5.28



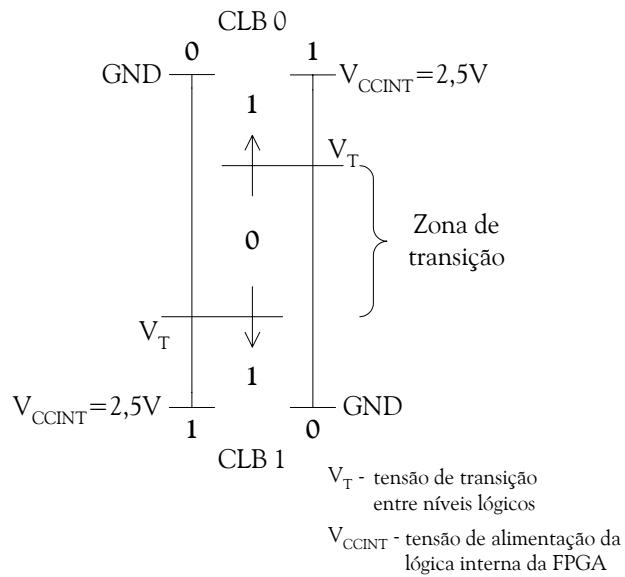
**Figura 5.30:** Representação esquemática das ligações e dos pontos de medida

A Figura 5.33 esquematiza a evolução da tensão ao longo da interligação, mostrando o comportamento das formas de onda da tensão em três pontos distintos, dois fora da zona de

transição e o outro dentro. Conclui-se que o nível lógico na interligação depende do nó onde for efectuada a leitura.



**Figura 5.31:** Diagrama de tensões nos pontos referenciados na Figura 5.30

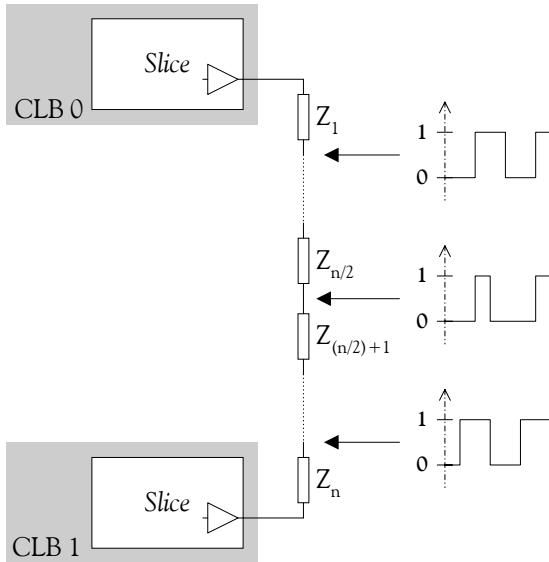


**Figura 5.32:** Tensão de transição entre níveis lógicos

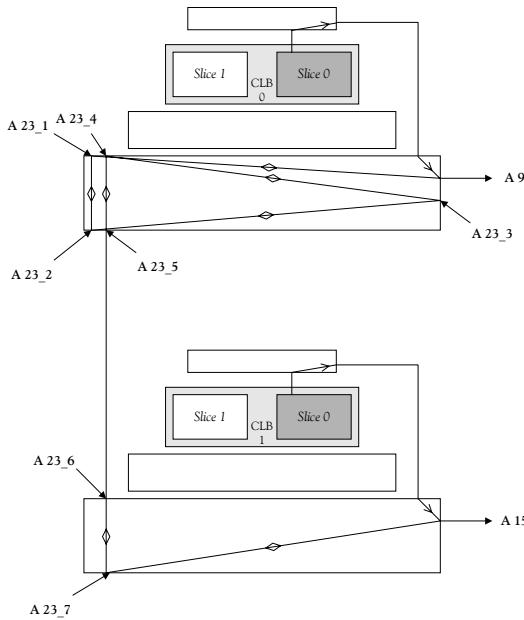
Ao longo da realização das experiências foi-se aumentando o número de segmentos e de pontos de leitura, tendo os resultados sido coerentes com a análise exposta. A última experiência efectuou-se sobre um circuito com uma linha de interligação constituída por doze segmentos, com nove pontos de medida distribuídos ao longo de toda a sua extensão, conforme se esquematiza na Figura 5.34. Neste circuito, o par de pontos de medida A23\_5 e A23\_6 é electricamente coincidente.

Na Figura 5.35 estão representadas as formas de onda obtidas nos pontos de medida referenciados na Figura 5.34. É visível um aumento do número de pontos onde as ondas sofrem influência de

cada uma das entradas, consequência do já referido aumento do número de segmentos, o que coloca mais pontos de medida dentro da zona de transição.



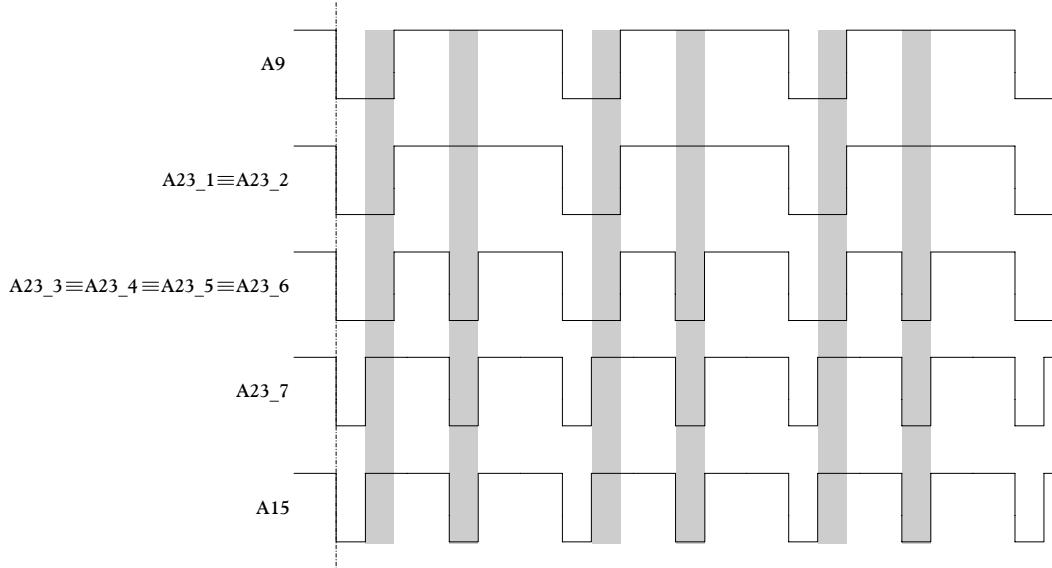
**Figura 5.33:** Variação da forma de onda da tensão ao longo da interligação



**Figura 5.34:** Representação esquemática das ligações e dos pontos de medida

Embora fosse possível continuar a juntar segmentos à interligação das duas saídas, essa situação afastaria-se das condições reais em que decorre a replicação, dado que esta se processa entre CLBs adjacentes e, por consequência, o número de segmentos envolvidos no paralelo desses CLBs é reduzido. Aliás, nem outra situação seria comprehensível, pois traduzir-se-ia num aumento insustentável dos recursos de interligação a ocupar durante a replicação e consequente inviabilidade

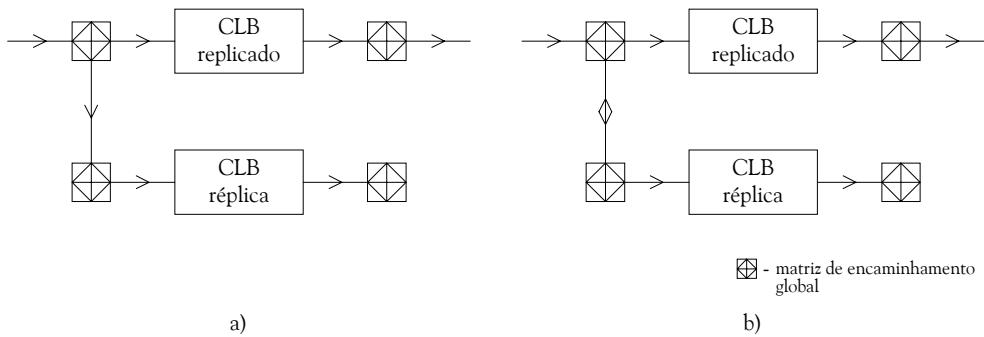
da solução, não só devido à falta de recursos disponíveis, mas também ao atraso de propagação, que implicaria a redução inaceitável da frequência de funcionamento da função afectada.



**Figura 5.35:** Diagrama de tensões nos pontos referenciados na Figura 5.34

#### 5.4.3 CORRECÇÃO DE ERROS DURANTE A AQUISIÇÃO DE ESTADO

Para se perceber qual é a influência no processo de replicação da existência de um defeito caracterizável por uma falta do tipo sempre-a numa linha de entrada ou saída do CLB replicado, é necessário ter presente toda a estrutura de encaminhamento associada a cada CLB. A percepção da influência da existência de uma falta sobre o processo de replicação implica igualmente a compreensão de todo o mecanismo subjacente à sua implementação. A primeira parte do processo consiste na replicação do conteúdo do CLB replicado no CLB réplica e na colocação em paralelo das entradas, como se ilustra na Figura 5.36. Para não obstar à clareza do desenho e porque entre a matriz de encaminhamento global e as entradas e saídas do CLB as interligações são unidirecionais, são omitidas as matrizes de encaminhamento de entrada e saída.

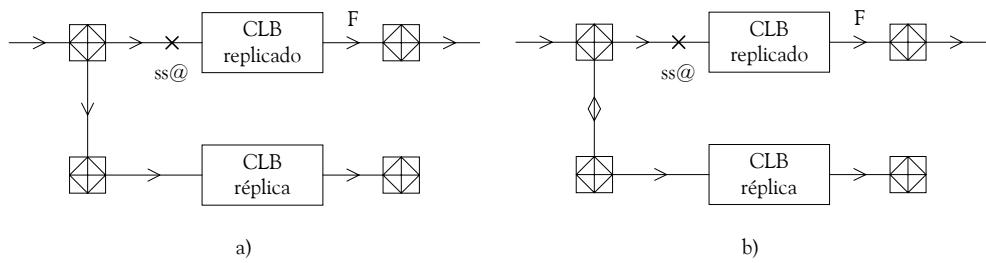


**Figura 5.36:** Paralelo das entradas

No caso da Figura 5.36-a), a interligação entre as duas entradas é unidireccional, no sentido do CLB replicado para o CLB réplica, de tal modo que os sinais que chegam ao primeiro possam igualmente ser derivados para o segundo. No caso da Figura 5.36-b), a bidireccionalidade da interligação garante os mesmos resultados. Como foi exposto anteriormente, uma interligação unidireccional, com sentido contrário ao indicado na Figura 5.36-a), impediria a propagação dos sinais de entrada do CLB replicado para o CLB réplica, pelo que não é considerada.

A colocação em paralelo das entradas garante a estabilização dos sinais na saída, antes da sua própria colocação em paralelo, bem como o não aparecimento de transitórios, resultado do corte e restabelecimento dos vários sinais, garantindo um funcionamento contínuo e sem perturbações do sistema. Nesta primeira fase, verifica-se igualmente a aquisição do estado dos *flip-flops* no caso dos circuitos síncronos com sinal de relógio livre, assegurando a coerência dos valores armazenados.

No caso da existência de um defeito, caracterizável por uma falta do tipo sempre-a, entre a matriz de encaminhamento global e as entradas da lógica interna do CLB replicado, conforme se ilustra na Figura 5.37, a falta poderá ser propagada para a saída F, mas nunca o será para trás, devido à unidireccionalidade da interligação entre essa matriz e as entradas do CLB. Consequentemente, garante-se que o nível lógico dos sinais presentes na entrada do CLB réplica é sempre o correcto, e, por inerência, também correcto o valor de estado adquirido pelos seus *flip-flops*. Note-se que este comportamento é independente do facto de a interligação entre as matrizes de encaminhamento global associadas aos dois CLBs ser unidireccional ou bidireccional (os dois casos da Figura 5.37). Comportamento idêntico ocorre no caso da presença de uma falta do tipo circuito aberto ou ponte.

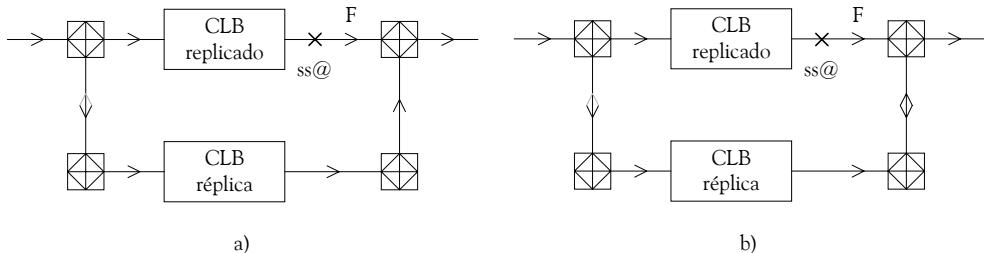


**Figura 5.37:** Comportamento do paralelo das entradas em caso de defeito na entrada do CLB replicado

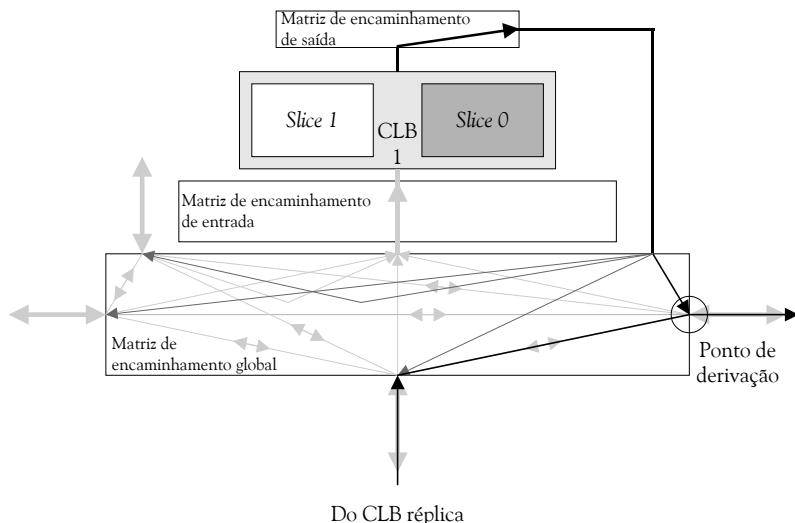
A presença de uma falta na saída do CLB replicado, conforme ilustrado na Figura 5.38, conduz a um maior número possível de situações. Repare-se, todavia, que o facto de a interligação das entradas ser unidireccional ou bidireccional é irrelevante para qualquer uma dessas situações.

Quando a interligação entre as saídas é unidireccional, caso a) da Figura 5.38, podem ocorrer duas situações distintas, consoante a localização da derivação do sinal para o resto do circuito, de acordo com os resultados observados nos ensaios experimentais. Se a derivação for efectuada logo após o

primeiro PIP atravessado pelo sinal na matriz de encaminhamento global, conforme se exemplifica na Figura 5.39, o sinal não é afectado pela interligação, visto o ponto de derivação se encontrar fora da zona de transição. Significa isto que a falta, a existir, continuará a propagar-se, sem qualquer alteração, para o resto do circuito, até que a nova configuração coloque o CLB replicado fora de serviço, momento a partir do qual o CLB réplica assegurará o bom funcionamento do circuito.



**Figura 5.38:** Comportamento do paralelo das saídas em caso de defeito na saída do CLB replicado

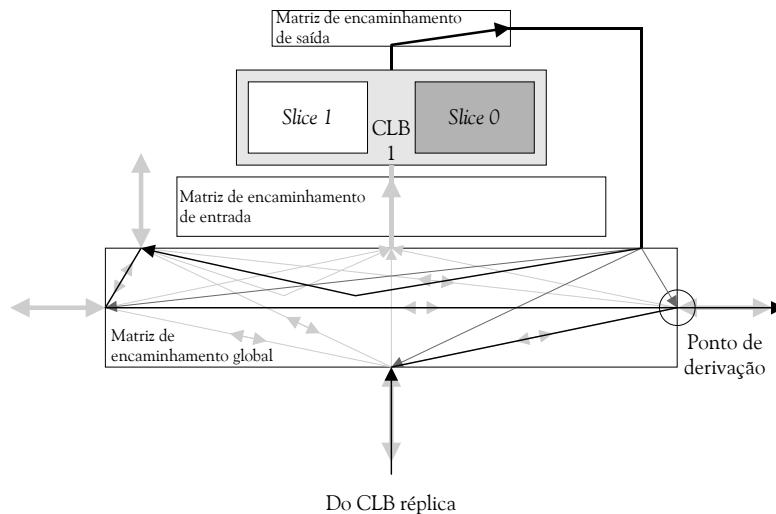


**Figura 5.39:** Derivação imediata do sinal após a chegada à matriz de encaminhamento global

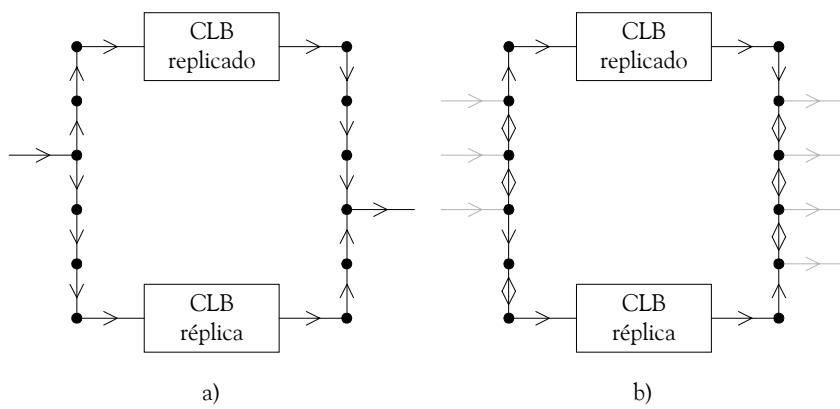
Se a derivação for efectuada depois de o sinal passar por vários PIPs da matriz de encaminhamento global, como se exemplifica na Figura 5.40, pode eventualmente cair na zona de transição, sofrendo o efeito do divisor de tensão resultante da interligação das saídas. Neste caso, a falta, a existir, não só continuará a propagar-se para o resto do circuito, mas fá-lo-á de modo distinto daquele que acontecia antes do paralelo, até que a nova configuração, substituindo o CLB replicado pelo CLB réplica, assegure o bom funcionamento do circuito [Gericota et al., 02c].

No entanto, a zona de transição pode ser deslocada, se, em vez de serem utilizadas ligações bidirecionais na interligação dos dois CLBs envolvidos na replicação, forem usadas ligações unidireccionais. Neste contexto, o valor à saída do CLB é igual ao valor presente ao longo da interligação, desde que esta se mantenha unidireccional. Contudo, tal estratégia está sujeita aos

condicionalismos enunciados anteriormente, pelo que a efectivação do paralelo só com interligações unidireccionais a partir de cada um dos CLBs exige que a derivação seja efectuada no ponto de confluência das interligações unidireccionais, ou, no caso da existência de interligações bidireccionais, em qualquer outro em que exista uma ou mais interligações bidireccionais a unir pontos de confluência das interligações unidireccionais, conforme se ilustra na Figura 5.41. No primeiro caso, todas as interligações são unidireccionais, pelo que só existe um ponto a partir do qual é possível efectuar a derivação, quer na interligação das entradas, quer na interligação das saídas. No segundo caso, existem na saída três interligações bidireccionais que interligam os dois pontos para onde confluem as duas interligações unidireccionais, pelo que são possíveis quatro pontos diferentes de derivação. Na entrada, há igualmente três interligações bidireccionais, mas uma delas (a situada na parte inferior) não pertence ao trecho de ligação entre os pontos de confluência, dado ser precedida por um outro, unidireccional, não confluente. Restam, por conseguinte, apenas três pontos diferentes de derivação.



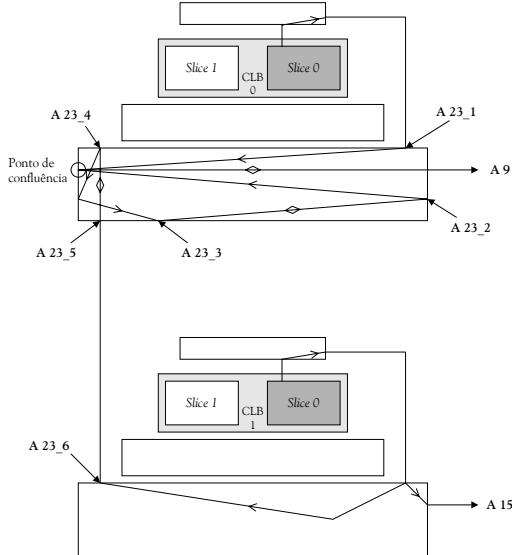
**Figura 5.40:** Derivação do sinal de saída na matriz de encaminhamento global após várias interligações internas



**Figura 5.41:** Pontos de derivação possíveis em interligações unidireccionais ou mistas

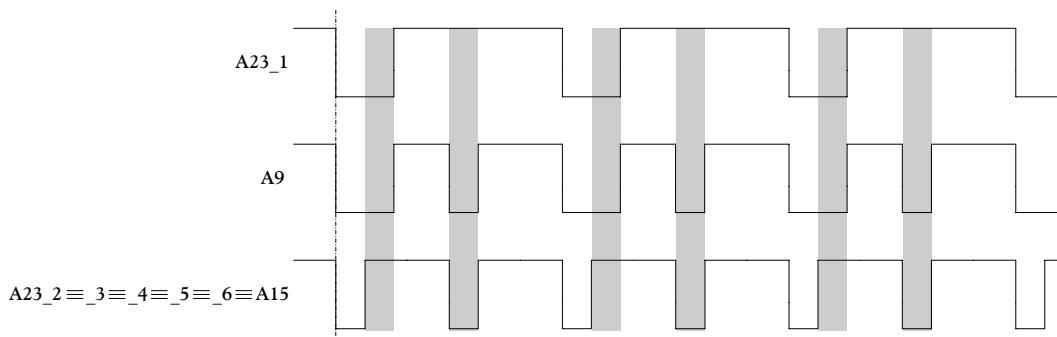
No sentido de comprovar o comportamento do paralelo em presença de interligações unidirecionais, realizou-se um segundo conjunto de três ensaios, usando o mesmo circuito.

Na primeira experiência foram empregues interligações unidirecionais até um ponto de confluência, com a excepção de dois dos segmentos que, sendo bidirecionais, não interligam pontos confluentes, como se ilustra na Figura 5.42.



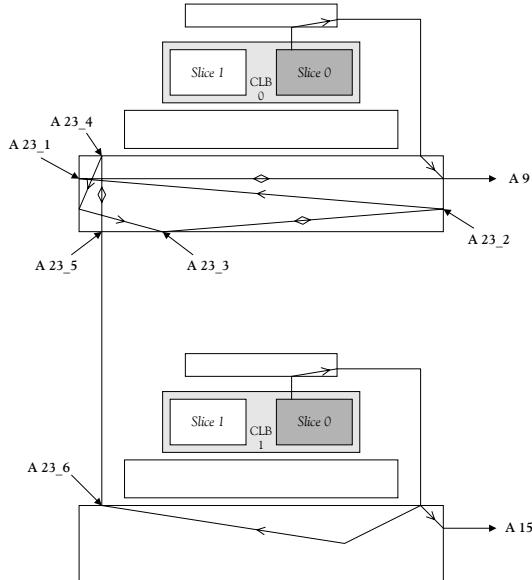
**Figura 5.42:** Representação esquemática das ligações e dos pontos de medida

Na Figura 5.43 estão representadas as formas de onda da tensão ao longo da interligação. Devido à unidirecionalidade imposta até ao ponto de confluência, as formas de onda são iguais às que estão presentes à saída dos CLBs, sendo excepção aquela que é obtida no ponto de confluência, a partir do qual é efectuada a derivação, e que sofre influência de ambas as saídas. Note-se ainda que a presença de interligações bidirecionais intercaladas entre interligações unidirecionais, e não unindo pontos de confluência, não tem, como se esperava, qualquer influência no desenvolvimento da forma de onda da tensão, o que é atestado pelas ondas obtidas entre os pontos A23\_2 e A23\_5.



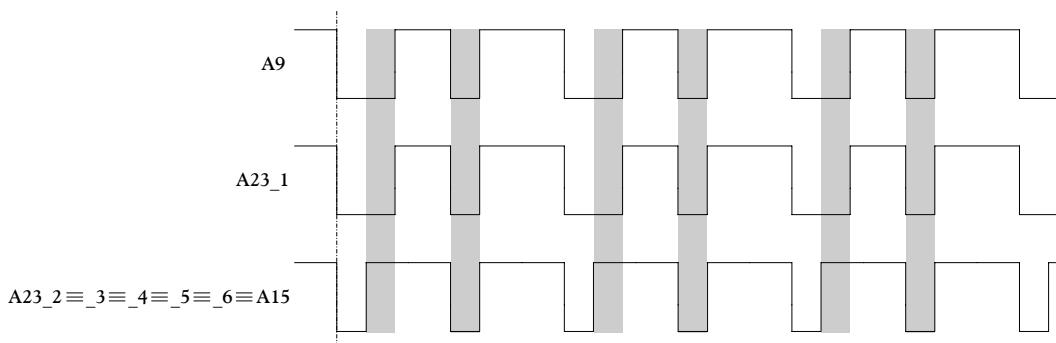
**Figura 5.43:** Diagrama de tensões nos pontos referenciados na Figura 5.42

Na segunda experiência, empregou-se uma interligação bidireccional para unir dois pontos de confluência, conforme se ilustra na Figura 5.44. As restantes interligações são unidireccionais, excepto duas que, sendo bidireccionais, não interligam pontos confluentes, e não influem, por isso, no comportamento da interligação, tal como foi visto no caso anterior.



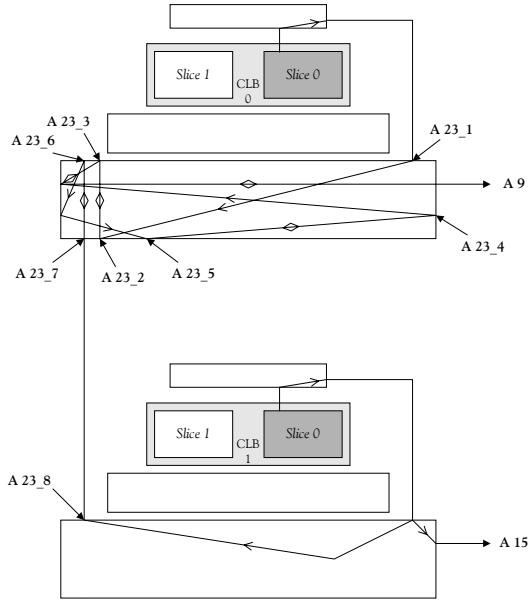
**Figura 5.44:** Representação esquemática das ligações e dos pontos de medida

Na Figura 5.45, estão representadas as formas de onda da tensão ao longo da interligação. Os pontos de medida A9 e A23\_1 são pontos de confluência, interligados por uma interligação bidireccional, pelo que ambos sofrem a influência do divisor de tensão. Os restantes pontos, não sendo pontos de confluência, têm comportamento idêntico ao do caso anterior, ou seja, apresentam a mesma forma de onda que a existente à saída do CLB.



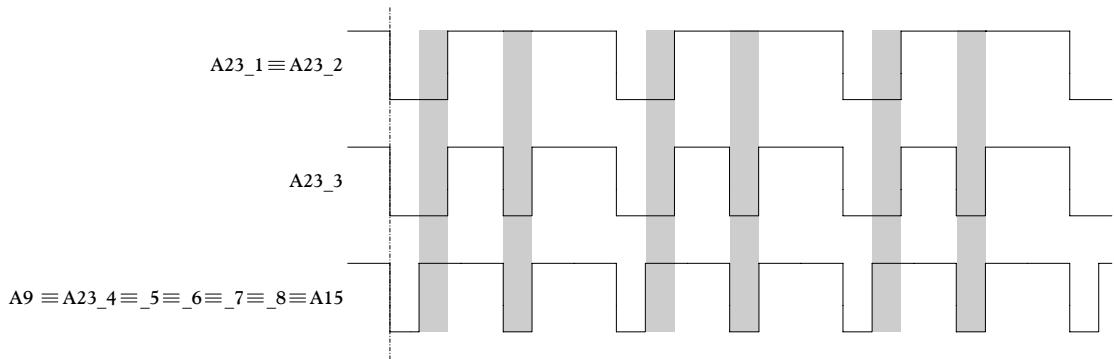
**Figura 5.45:** Diagrama de tensões nos pontos referenciados na Figura 5.44

Na terceira experiência, foram usadas duas interligações bidireccional para unir dois pontos de confluência, conforme se ilustra na Figura 5.46. As restantes interligações são unidireccionais, à excepção de duas das interligações que são bidireccionais, mas não interligam pontos confluentes.



**Figura 5.46:** Representação esquemática das ligações e dos pontos de medida

Na Figura 5.47, estão representadas as formas de onda da tensão ao longo da interligação. Só o ponto de medida A23\_3 sofre a influência do divisor de tensão, ou seja, o ponto intermédio entre as duas interligações bidireccionais que ligam os pontos para onde confluem as interligações unidireccionais que transportam o sinal das saídas de cada CLB. Os restantes pontos têm comportamento idêntico ao caso anterior, isto é, apresentam a mesma forma de onda que a existente à saída do CLB. De referir que, neste caso, os pontos A9 e A23\_2 apresentam valores de tensão iguais aos das respectivas saídas, fazendo-se notar a presença de um divisor de tensão apenas num ponto intermédio, à semelhança do que aconteceu na primeira série de experiências em que as interligações eram todas bidireccionais.



**Figura 5.47:** Diagrama de tensões nos pontos referenciados na Figura 5.46

Deste conjunto de experiências, conclui-se que a existência de uma falta numa interligação unidireccional apenas tem influência sobre os pontos situados a jusante dessa falta, não se propagando para montante. Dada a unidireccionalidade do multiplexador de entrada, qualquer

defeito situado para lá do ponto de derivação de entrada (na matriz de encaminhamento global), não se propaga para trás, pelo que, em qualquer circunstância, defeitos para lá deste ponto não se propagam para as entradas do CLB réplica, garantindo que a este são sempre aplicados os sinais correctos.

Embora fosse provável o aparecimento de alguma instabilidade nos valores lógicos nos pontos intermédios do divisor de tensão, isso não foi observado nos sinais que chegavam ao exterior da FPGA, que mantiveram sempre valores lógicos bem definidos. Tal deve-se provavelmente ao facto de, até atingir o exterior, o sinal atravessar mais do que um *buffer*, que, não sendo exactamente iguais, apresentam valores ligeiramente diferentes para a tensão de transição entre níveis lógicos, conduzindo ao aparecimento de histerese na linha de transmissão.

Apesar de, durante esta fase experimental, se ter forçado a existência de um curto-círcuito entre as saídas dos CLBs, a limitação da corrente na interligação impõe pelas impedâncias dos PIPs evita qualquer dano sobre os recursos da FPGA. De facto, assumindo como valor para a impedância de cada PIP os  $250\ \Omega$  e não considerando a impedância de saída de cada *buffer*, a existência de um único PIP limitaria a corrente através da interligação aos  $10\ mA$ , valor abaixo do máximo de  $24\ mA$ , indicado como limite pelo fabricante. Na realidade, existirá sempre mais que um PIP na interligação, pelo que o valor será sempre bastante inferior. A única consequência previsível é a ocorrência, nestas circunstâncias, de um aumento pontual da temperatura. Trata-se, contudo, de uma situação que só ocorre durante o processo de replicação e no caso de existência de uma falta, pelo que, mesmo a verificar-se um aumento da temperatura, o reduzido espaço de tempo em que o curto se mantém, bem como o seu carácter excepcional, não configuram uma situação que possa pôr em risco a fiabilidade do componente.

#### 5.4.4 CORRECÇÃO DE FALTAS TRANSITÓRIAS

Este resultado mostra que, no caso dos circuitos síncronos com relógio livre, em que os *flip-flops* do CLB réplica adquirem a sua informação de estado durante a replicação directamente das entradas, o valor adquirido é sempre o correcto, independentemente de qualquer erro que possa afectar os valores presentes nos *flip-flops* do CLB replicado e que seria produto de defeitos estruturais ou induzidos por SEUs ou SETs. Consequentemente, após a replicação, um possível erro que anteriormente afectasse os valores de estado fica automaticamente corrigido e as saídas do CLB réplica exibem os valores correctos [Gericota et al., 02a]. Esta correcção é restrita aos circuitos síncronos com relógio livre, onde não existe uma real transferência do valor de estado entre os *flip-flops* replicados e as suas réplicas. No caso dos circuitos síncronos com sinal de habilitação de relógio e no dos circuitos assíncronos, em que se realiza uma verdadeira transferência de estado, os

erros que afectem o valor de estado dos *flip-flops* replicados são transferidos para as suas réplicas, só sendo corrigidos assim que o próprio circuito actualize o estado. De notar que se essa actualização ocorrer já depois da colocação em paralelo das entradas e das ligações ao bloco auxiliar de replicação estarem estabelecidas, os *flip-flops* réplica serão actualizados com o valor correcto, se este for obtido directamente a partir das entradas do CLB réplica (caso das células contendo apenas um registo simples). O mesmo sucede se a actualização tiver lugar a partir da saída da sua própria tabela de consulta (caso das células contendo lógica combinatória registada), apesar de o *flip-flop* replicado receber ainda o valor incorrecto. Dessa forma, quando as saídas de ambos os CLBs forem colocadas em paralelo, já será eventualmente propagado para o resto do circuito o valor correcto, dependendo da posição (na interligação) do nó onde é efectuada a derivação do sinal. Dado que ambas as saídas, a do CLB replicado e a da sua réplica, apresentam valores diferentes, o paralelo das saídas terá um comportamento do tipo divisor de tensão. A propagação ou não de um valor errado para o resto do circuito durante a 2<sup>a</sup> fase da replicação dependerá da posição nessa interligação do nó onde é efectuada a derivação e da influência que o divisor de tensão tem sobre ele, tal como acontecia nos casos analisados na secção anterior. De qualquer forma, depois de garantida a correção do erro que afectava o valor armazenado no *flip-flop*, o valor propagado para jusante será seguramente o correcto, no fim do processo de replicação.

As faltas com origem em fenómenos transitórios que afectam a memória de configuração poderão ou não ser corrigidas durante a replicação, dependendo da estratégia usada para a geração dos necessários ficheiros de reconfiguração parcial. Estas faltas alteram de forma permanente a funcionalidade do circuito, em virtude de alterarem a configuração da lógica. No entanto, o seu carácter não é permanente, no sentido em que, não afectando estruturalmente o componente, a falta pode ser corrigida pela simples reposição do antigo valor de configuração. Essa reposição pode ser efectuada durante o processo de replicação, se, aquando da programação da funcionalidade do CLB replicado no CLB réplica, o ficheiro de configuração parcial usado for gerado com base num ficheiro de configuração guardado numa memória exterior, em vez de a partir dos dados obtidos pela leitura da memória de configuração do próprio componente. De facto, a leitura parcial da área da memória de configuração adstrita à definição da funcionalidade do CLB replicado e a posterior programação desses dados na área da memória de configuração adstrita ao CLB réplica, implicaria igualmente a transferência do valor errado e, consequentemente, a manutenção da falta. No entanto, se a geração do ficheiro de configuração parcial que definirá a funcionalidade do CLB réplica se efectuar a partir do original ou de uma cópia do conteúdo actual da FPGA, guardada numa memória exterior, esse erro será corrigido. Uma vez que num sistema baseado em FPGAs com reconfiguração parcial dinâmica as funções nelas configuradas são frequentemente alteradas, a

melhor solução é a manutenção numa memória externa de uma cópia do conteúdo actual de cada FPGA. Todas as operações de reconfiguração devem ser efectuadas quer sobre a FPGA quer sobre a configuração guardada em memória externa, de modo a se garantir a coerência de ambas. Além disso, a manutenção de uma cópia actualizada da funcionalidade permitirá a recuperação do sistema no caso de ocorrência de uma falha catastrófica. Visto que a FPGA é baseada em tecnologia volátil, a existência de uma memória que conserve a configuração de forma permanente e que permita a sua programação no arranque do sistema é uma exigência do próprio componente, pelo que não se verifica um aumento dos recursos para a implementação deste método.

## 5.5. SUMÁRIO

Neste capítulo, descreveu-se de forma exaustiva a metodologia proposta para a replicação de CLBs activos, sem perturbação da funcionalidade dos circuitos implementados na FPGA. O objectivo da replicação é relocar os CLBs activos em CLBs livres, por forma a libertar os recursos ocupados para o teste.

A análise de um conjunto vasto de implementações serviu de base à identificação dos diferentes tipos de ocupação dos CLBs, permitindo o reconhecimento dos requisitos necessários para a replicação de cada tipo. Em função destes tipos, foram propostas duas soluções que permitem cobrir a sua totalidade, com base num processo de grande simplicidade, mas sem abdicar dos objectivos iniciais.

A grande vantagem das soluções propostas é a sua fácil implementação e universalidade, não sendo necessária a adução de qualquer tipo de lógica de decisão ou controlo para a replicação dos CLBs. O processo pode ser aleatoriamente iniciado e finalizado, independentemente do estado da função a cuja implementação o CLB está adstrito.

O processo de replicação conduz também, e só por si, à possibilidade de correcção de erros, com origem na incidência de radiação que resulta numa inversão do valor armazenado nas células de memória.



## **6. ROTAÇÃO DOS RECURSOS**



Para a execução do teste estrutural é necessária a libertação, por replicação, dos recursos presentemente ocupados, sem que disso resulte perturbação para o normal funcionamento do sistema em que o componente está inserido. Essa libertação deve ocorrer de forma ordenada, de modo a que se possa garantir um intervalo de latência determinado para qualquer uma das faltas abrangidas pelo modelo considerado.

A libertação ordenada dos recursos é o objectivo das estratégias de rotação propostas neste capítulo. A rotação possibilita que todos os recursos sejam varridos pelos procedimentos de teste, com o mínimo de custos para o sistema. Isso implica identificar os factores de custo imediatos associados à relocação de cada elemento a testar e analisar o seu comportamento relocação após relocação, consoante a aplicação das diferentes estratégias de rotação que, à partida, seriam mais viáveis.

A relocação dos blocos lógicos afecta não a lógica em si, mas antes as interligações entre os vários componentes de uma mesma função, distribuídos por vários blocos lógicos configuráveis. A configuração interna de cada um dos blocos lógicos mantém-se inalterada depois de relocada, o mesmo já não acontecendo, porém, com o encaminhamento dos sinais, o que introduz mudanças nos tempos de propagação e pode conduzir a alterações no funcionamento da função. Partindo desta observação, tenta-se determinar, através da análise da aplicação das estratégias a casos reais, os padrões de comportamento e compreender o motivo de determinados desvios a esses padrões, sendo proposta uma métrica de custo que, em face de uma qualquer implementação, possibilita a determinação de qual a estratégia óptima a adoptar. A eficácia da métrica é comprovada pela sua aplicação aos casos reais em estudo, em particular àqueles que fogem ao comportamento geral definido pelo padrão inicial, sendo posteriormente generalizada para qualquer caso.

Por último, é proposta uma estratégia para o varrimento e teste dos recursos de interligação e encaminhamento global, de forma a permitir igualmente a sua libertação faseada e o seu teste num tempo determinado.



## 6.1. ESTRATÉGIAS PARA A ROTAÇÃO DE RECURSOS

A principal ideia subjacente à proposta apresentada nesta tese é a de ter apenas uma porção relativamente pequena do componente sob teste *off-line*, enquanto os restantes recursos continuam a assegurar a funcionalidade normal do sistema. O teste da totalidade do componente é conseguido pela rotação dessa porção sob teste ao longo de toda a FPGA. Se a funcionalidade de um pequeno número de elementos pode ser relocada noutra porção, então esses elementos podem ser colocados *off-line* e sujeitos a um teste estrutural de uma forma completamente transparente, isto é, sem perturbação da operação normal do sistema. Este procedimento, que prossegue copiando e testando outro conjunto de elementos, percorre sistematicamente toda a FPGA em busca de faltas emergentes.

Concertada uma estratégia para a replicação dos elementos activos – CLBs e interligações –, que permite libertá-los das suas funções no circuito e disponibilizá-los para a aplicação do teste sem causar perturbações ao sistema, é agora necessário definir uma estratégia para sequencializar essa libertação. O objectivo é que os recursos sejam sucessivamente libertados, de modo que todos eles possam ser testados dentro de um intervalo de tempo determinado.

Definida a unidade de teste em termos de lógica configurável – um CLB –, pelas razões já apresentadas, a estratégia de relocação sucessiva a definir deverá determinar a forma como o CLB sob teste vai percorrendo todos os blocos lógicos dentro da FPGA. De notar que a definição do CLB como unidade de teste para a lógica configurável não condiciona a definição para as interligações de dois canais sob teste perpendiculares entre si, uma vez que o seu tratamento a nível da replicação, da rotação e do teste se pode efectuar independentemente do teste dos blocos lógicos, ficando apenas sujeito às condicionantes de ordem prática apontadas no capítulo anterior.

Em virtude das características de configuração das FPGA e da sua forma de utilização em sistemas reconfiguráveis, o seu índice de ocupação nunca atinge os 100%, pelo que se pode afirmar que, a qualquer instante, haverá sempre no espaço lógico configurável, um ou mais CLBs livres. A aplicação da estratégia de teste pode começar exactamente por um desses CLBs, configurando nele as funções de teste e reencaminhando os sinais de entrada e saída, respectivamente, para a aplicação dos vectores e recolha das respostas [Gericota et al., 00]. Depois dessa operação, se nenhuma falta estrutural for detectada no CLB, a funcionalidade de outro CLB pode ser nele relocada, libertando-o, por sua vez, para o teste. A pergunta que se coloca é a de saber qual o CLB cuja funcionalidade deve ser relocada no CLB testado. A resposta deverá ter em conta os factores de custo associados à relocação de um CLB, em termos imediatos e em termos globais, de modo a

que sucessivas relocações conduzam, num intervalo de tempo determinado e com o mínimo custo possível, ao teste estrutural de todos os CLBs.

### 6.1.1 FACTORES DE CUSTO

A definição da estratégia a usar para se proceder ao varrimento de toda a área da FPGA pelas funções de teste está condicionada por determinados factores de custo subjacentes ao objectivo de testar estruturalmente toda a FPGA, sem que isso perturbe o seu normal funcionamento, dependendo da sua análise a escolha do caminho a seguir.

A relocação da funcionalidade de um CLB obriga ao reencaminhamento de um número variável de sinais que nele têm origem ou destino. Esse reencaminhamento far-se-á obrigatoriamente através de um outro conjunto de recursos de encaminhamento diferente do definido inicialmente, aquando da síntese do circuito, pelo programa de distribuição e encaminhamento, e que possivelmente obedeceu a um conjunto de requisitos temporais impostos pelo projectista. O uso de outros recursos nesse reencaminhamento altera os tempos de propagação dos sinais, conduzindo, eventualmente, no caso de se registar um aumento desses tempos, a perturbações na operação ou mesmo à alteração da funcionalidade.

Seja  $s_1, s_2, \dots, s_n$  o conjunto de todos os sinais de uma determinada função implementada numa FPGA. A cada um destes sinais corresponde um determinado atraso de propagação,  $t_{sx}$ , que depende do tipo de recursos de encaminhamento usados e do número de PIPs que atravessa entre o seu ponto de origem e de destino. A frequência máxima de funcionamento da função é dada pela Equação 6.1.

**Equação 6.1**

$$\frac{1}{f_{\text{MÁX}}} = t_{p_{\text{MÁX}}} = \text{MÁX}(t_{s_1}, t_{s_2}, \dots, t_{s_n})$$

Atente-se que várias funções partilhando a mesma FPGA podem ter frequências máximas de funcionamento diferentes, condicionadas pelos tempos de propagação dos seus próprios sinais.

Para evitar que o reencaminhamento dos sinais aquando da relocação de um CLB possa perturbar o funcionamento da função à qual está alocado, a relocação não deve provocar um aumento no valor de  $t_{p_{\text{MÁX}}}$ , independentemente de os tempos de propagação adstritos a cada um dos sinais envolvidos poderem aumentar ou diminuir. As possíveis variações introduzidas em  $t_{p_{\text{MÁX}}}$  constituem o primeiro factor de custo a ter em conta na definição de uma estratégia para a rotação dos recursos sob teste.

O segundo factor a ponderar é o custo da própria reconfiguração, entendido como o número de vectores necessários para relocar cada um dos CLBs, dado que a necessidade de um maior número

de vectores implica um intervalo de teste mais longo para os testar a todos. Este custo está intimamente relacionado com a arquitectura da memória de configuração. Sendo a configuração da FPGA efectuada segundo um esquema de colunas, uma estratégia que implique a relocação de um CLB numa coluna diferente terá, como consequência provável, a necessidade de um maior número de vectores de configuração do que é implicado quando a relocação se efectua dentro de uma mesma coluna.

Os mesmos princípios têm igual aplicação na replicação de interligações, pelo que, sempre que possível, é preferível a relocação de uma interligação dentro do canal sob teste, em vez do seu desvio por outros canais, uma vez que, como já foi referido, não é viável o teste da totalidade das interligações de um canal simultaneamente. A replicação de uma interligação num conjunto de recursos de encaminhamento paralelos idênticos no mesmo canal evita aumentos no atraso de propagação dos sinais e a necessidade do uso de um maior número de vectores de configuração.

Um terceiro factor, que, não se traduzindo num custo directo, contribui para a inflação dos dois anteriores, é o reduzido número de recursos de encaminhamento disponíveis numa FPGA. Embora as últimas gerações de FPGAs se caracterizem por uma elevada abundância de recursos de encaminhamento, qualquer estratégia deve ter sempre em conta que o seu número é limitado e que a sua sobreocupação se traduzirá, aquando da relocação, no reencaminhamento dos sinais por caminhos mais longos, de forma a contornar as zonas mais densamente ocupadas. Caminhos mais longos conduzem a atrasos de propagação maiores e implicam o cruzamento de mais colunas, com o consequente aumento do número de vectores de configuração necessários para o seu estabelecimento.

Como ponto de partida para o estudo da estratégia de rotação do bloco lógico sob teste que melhor se ajustasse aos objectivos pretendidos, minimizando os custos, consideraram-se três alternativas [Gericota et al., 01b]:

- o uso de uma estratégia de rotação aleatória;
- o uso de uma estratégia de rotação horizontal;
- o uso de uma estratégia de rotação vertical.

### 6.1.2 ROTAÇÃO ALEATÓRIA

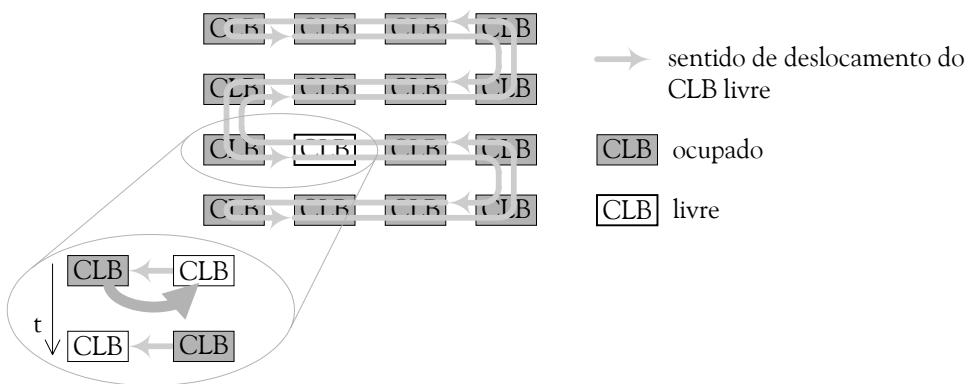
Sendo a estratégia, por definição, uma arte de planeamento e execução de operações, uma estratégia aleatória é, por si só, um contra-senso, por não obedecer a qualquer plano estabelecido previamente. Como consequência, esta ‘pseudo-estratégia’ apresenta custos bastante elevados a nível do aumento dos atrasos de propagação e do número de vectores de configuração.

Se o algoritmo que gera a atribuição de recursos para a implementação de uma função na FPGA tende, numa tentativa de reduzir os tempos de propagação dos sinais dentro da função e de optimizar o mais possível a utilização dos recursos disponíveis, a agrupar em blocos lógicos contíguos a totalidade dos componentes da lógica necessária à sua implementação, não seria sensato dispersá-los, uma vez que isso conduziria não só ao aumento imprevisível dos tempos de propagação, e consequente diminuição da frequência máxima de funcionamento, mas também a demasiada pressão nos limitados recursos de encaminhamento da FPGA. No caso de todas as funções verem os seus componentes lógicos, implementados nos CLBs que lhe estão alocados, dispersos por toda a FPGA, criar-se-ia um enorme emaranhado em termos da interligação dos seus sinais, do qual resultaria a rápida saturação dos recursos de encaminhamento.

Outra questão a referir é que uma rotação aleatória da área sob teste é contrária à necessidade de assegurar uma determinada latência na cobertura de defeitos para toda a matriz de CLBs, pois essa situação seria inaceitável do ponto de vista da confiança no teste.

### 6.1.3 ROTAÇÃO HORIZONTAL

Na segunda opção, a relocação da funcionalidade do próximo CLB a testar tem lugar para um CLB livre que neste caso lhe deve ser adjacente horizontalmente, à direita ou à esquerda, dependendo do sentido da rotação. Desta forma, a rotação pode ser visualizada como sendo a deslocação de um CLB livre ao longo de toda a matriz de CLBs, conforme ilustrado na Figura 6.1, em sentido inverso ao da relocação dos CLBs a testar.



**Figura 6.1:** Estratégia de rotação horizontal

De referir que a existência de CLBs não ocupados em nada altera a progressão do CLB livre, mas significa apenas que a funcionalidade a relocar é inexistente ou nula, mantendo-se, por conseguinte, a estratégia de rotação e o procedimento de teste dos CLBs.

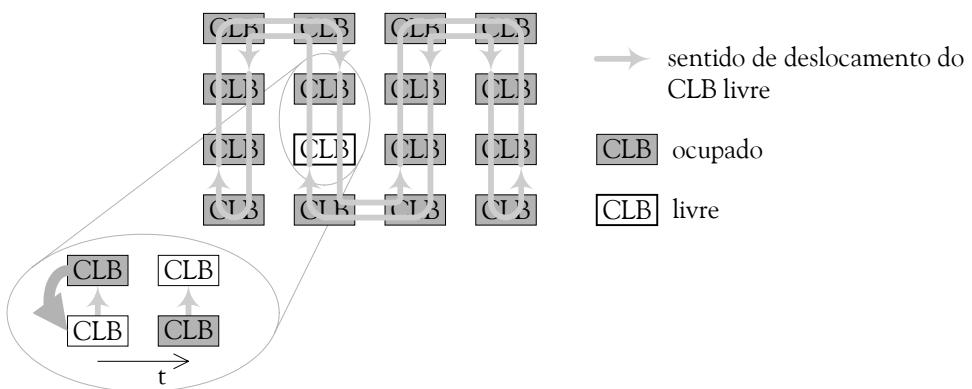
A relocação dos CLBs em CLBs contíguos previne aumentos acentuados no comprimento das interligações, com a consequente possibilidade de aumento dos tempos de propagação e a provável perturbação do funcionamento da função afectada, e acautela uma possível situação de ruptura, evitando uma ocupação mais acentuada dos recursos de encaminhamento.

A estratégia de rotação horizontal permite igualmente determinar um tempo de latência máximo para a cobertura de defeitos nos CLBs, que é calculado tendo em conta as piores condições para o teste: todos os CLBs da matriz, com excepção do próprio CLB sob teste, estão ocupados com funções sequenciais, que implicam o uso de blocos auxiliares de replicação, maximizando, em consequência, o tempo de relocação de cada um. Esta situação extrema não é, obviamente, realista, pelo que o tempo de latência efectivo será sempre inferior ao tempo de latência máximo calculado.

O único senão aparente neste ponto parece ser o facto de a relocação horizontal envolver alterações na configuração que abrangem duas colunas de CLBs, o que implica um custo de reconfiguração relativamente elevado.

#### 6.1.4 ROTAÇÃO VERTICAL

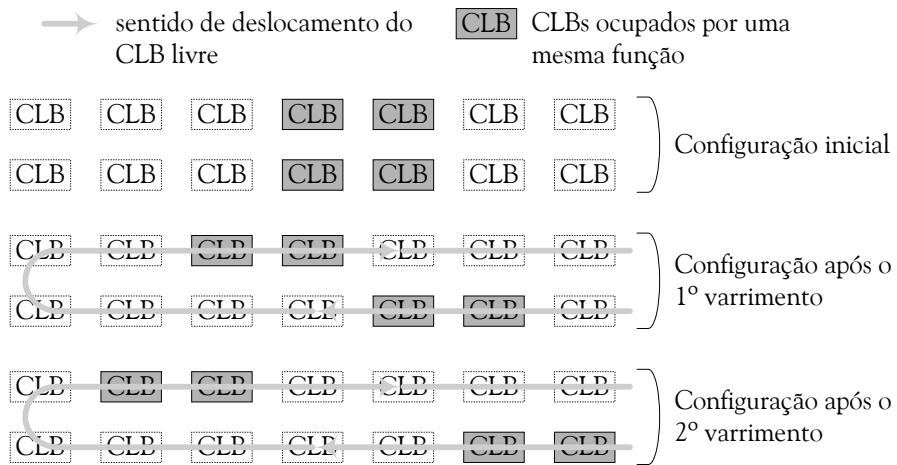
A última questão apontada é resolvida com a opção pelo uso de uma rotação vertical para o CLB livre. Este recebe a funcionalidade do próximo CLB a testar, o qual, neste caso, lhe deve ser adjacente verticalmente, em cima ou em baixo, dependendo do sentido de rotação, conforme ilustrado na Figura 6.2. Esta opção retém todas as vantagens da anterior.



**Figura 6.2:** Estratégia de rotação vertical

A inversão do sentido de deslocamento do CLB livre após um varrimento completo da matriz lógica tem como consequência, em ambos os casos, a reposição da distribuição e encaminhamento iniciais ao fim de dois varrimentos.

Note-se que, se o varrimento da matriz se efectuasse sempre no mesmo sentido, isso implicaria que, atingido um dos extremos, o CLB livre fosse deslocado para o canto diametralmente oposto da matriz de CLBs, registando-se uma troca com a funcionalidade desse CLB. Desse modo, parte da funcionalidade de uma função seria relocada para o extremo oposto, com consequências indesejáveis em termos de atraso de propagação dos sinais dessa função e de aumento de pressão sobre os recursos de encaminhamento. Esta situação repetir-se-ia a cada novo varrimento, acentuando ainda mais o problema. Para lá disso, um deslocamento sempre no mesmo sentido teria também consequências nefastas ao nível da dispersão dos componentes da lógica associada a cada função, mesmo que situadas no miolo da matriz, como se exemplifica na Figura 6.3, para o caso de uma rotação horizontal (o mesmo sucede no caso de uma rotação vertical).



**Figura 6.3:** Dispersão dos componentes de uma mesma função ao longo de sucessivas relocações sempre no mesmo sentido

Este vai-e-vem do CLB livre ao longo da matriz de blocos lógicos implica uma latência de teste que, embora com um intervalo de tempo conhecido, é variável [Gericota et al., 01c]. O tempo que demora a testar de novo um determinado CLB alterna, conforme o sentido de rotação, entre um valor máximo e um valor mínimo, que depende do tamanho da matriz (número de colunas e linhas de CLBs) e dos tempos de relocação e teste. O tempo máximo de latência na detecção de uma falta na matriz de CLBs é dado pela Equação 6.2 e o mínimo, pela Equação 6.3.

**Equação 6.2**

$$\tau_{\text{lat\acute{e}ncia}_{\text{M\'AX}}} = ((\# \text{CLB}_{\text{linhas}} \times \# \text{CLB}_{\text{colunas}}) - 1) \times 2 \times (\text{t}_{\text{reloca\c{c}\~ao}} + \text{t}_{\text{teste}})$$

**Equação 6.3**

$$\tau_{\text{lat\acute{e}ncia}_{\text{m\'in}}} = 2 \times (\text{t}_{\text{reloca\c{c}\~ao}} + \text{t}_{\text{teste}})$$

onde:

$t_{relocação}$ : tempo necessário para completar a relocação de um CLB;

$t_{teste}$ : tempo de teste de um CLB.

O tempo necessário para completar uma relocação,  $t_{relocação}$ , depende da funcionalidade implementada pelo CLB, pelo que, para se considerar a pior situação, abrangendo todas as possibilidades, deve optar-se pelo caso mais longo. Por outro lado, o tempo de teste,  $t_{teste}$ , é independente dessa funcionalidade e praticamente fixo, embora possa registar ligeiros desvios em virtude da possibilidade de existência de variações nos tempos de configuração das interligações entre as entradas/saídas do CLB sob teste e o Registo do Utilizador/células do registo BS, respectivamente. No entanto, apesar de mais do que uma configuração de teste ser necessária para testar o CLB, a configuração dessas interligações é única. Na prática, essa variação não é significativa, dada a mínima importância relativa existente entre esses desvios e o tempo de teste, e entre este e o tempo de relocação.

O processo de varrimento pode ser repetido ou não, dependendo da estratégia global de teste definida para o sistema pelo projectista.

## 6.2. APLICAÇÃO AO CONJUNTO DE CIRCUITOS-PADRÃO

Com o objectivo de testar qual das duas alternativas exequíveis para a definição de uma estratégia de rotação, horizontal ou vertical, apresenta melhor desempenho, foram ambas aplicadas ao conjunto de circuitos-padrão anteriormente usados para avaliar a estratégia de replicação.

A Tabela 6.1 apresenta uma lista das principais características lógicas dos circuitos implementados, sendo o seu conhecimento necessário para se proceder à análise dos resultados. Ao número de entradas primárias acrescem sempre, conforme indicado, mais duas entradas, correspondentes à linha de relógio e à linha de inicialização.

A aplicação da estratégia de rotação vertical ou da de rotação horizontal apresenta custos diversos ao nível do tamanho dos ficheiros de reconfiguração e da degradação que provocam no desempenho do circuito. A Tabela 6.2 apresenta as consequências para cada um dos circuitos da aplicação de cada uma dessas estratégias. Uma das principais ilações é a diminuição, em quase todos os casos, da frequência máxima de funcionamento do circuito em relação à implementação inicial, devido ao aumento dos atrasos de propagação provocados pela reorganização das linhas de encaminhamento

dos sinais. Essa diminuição é mais acentuada no caso da aplicação da rotação horizontal (em média 18% menos contra 7% na vertical<sup>1</sup>), embora existam algumas exceções.

**Tabela 6.1:** Principais características dos circuitos usados no teste da estratégia de rotação

Referência	Circuito		Lógica		Recursos de transporte	
	Entradas primárias	Saídas primárias	Nº de portas	Nº de flip-flops	Linhas	Segmentos
B01	2+2	2	47	5	0	0
B02	1+2	1	29	4	0	0
B03	4+2	4	150	30	0	0
B04	11+2	8	606	66	4	14
B05	1+2	36	977	34	4	16
B06	2+2	6	61	9	0	0
B07	1+2	8	422	49	2	6
B08	9+2	4	168	21	0	0
B09	1+2	1	160	28	0	0
B10	11+2	6	190	17	0	0
B11	7+2	6	484	31	1	4
B12	5+2	6	1037	121	0	0
B13	10+2	10	343	53	1	4
B14	32+2	54	4787	245	11	150

Em relação ao tamanho global dos ficheiros de reconfiguração parcial, resultante do somatório de todos os ficheiros de reconfiguração parcial necessários para a aplicação de cada uma das estratégias a cada circuito, a aplicação da estratégia de rotação horizontal origina ficheiros de tamanho médio superior em cerca de 20%, relativamente à aplicação da estratégia de rotação vertical. Este resultado é facilmente explicável se tivermos em conta a forma de organização dos recursos de configuração na FPGA, onde cada vector de reconfiguração se estende verticalmente desde o topo

---

<sup>1</sup> Em virtude da grande disparidade de características apresentadas pelos vários circuitos não são incorporados, por falta de significado prático, outros valores estatísticos.

até ao fundo, abrangendo apenas uma coluna. Enquanto a rotação vertical dos CLBs afecta apenas uma das colunas, com excepção para a rotação dos CLBs situados em ambos os extremos das colunas, a reconfiguração horizontal afecta sempre duas colunas, necessitando, por isso, de um maior número de vectores de configuração para a sua execução. A diferença só não é mais acentuada devido ao comparativamente elevado número de vectores de reconfiguração necessários para implementar as alterações nos recursos de encaminhamento, que afectam outras colunas além da coluna onde se localiza o CLB a replicar. Verifica-se assim uma diluição da diferença entre a aplicação de uma ou outra das estratégias. De notar, neste caso particular, o comportamento do circuito B02, que ocupa apenas um CLB. A sua relocação para um CLB adjacente horizontalmente implica um conjunto de ficheiros de reconfiguração que é, em tamanho, 50% maior que o necessário para a sua relocação dentro da mesma coluna.

A Tabela 6.3 apresenta os resultados obtidos quando se divide o tamanho global dos ficheiros de reconfiguração pelo número de CLBs ocupados pelos respectivos circuitos, notando-se uma tendência, embora pouco acentuada, de aumento dessa relação, quando aumenta o número de CLBs ocupados. Esse facto reflecte uma maior dificuldade no encaminhamento dos sinais afectados pela relocação, devido a uma maior densidade de ocupação dos recursos de interligação. Neste caso, a vantagem continua a estar do lado da estratégia de reconfiguração vertical. A estratégia horizontal apresenta, em média, ficheiros de reconfiguração por CLB de tamanho superior em cerca de 9,5%.

Uma análise mais pormenorizada dos dados oferecidos pela Tabela 6.2 faz ressaltar algumas particularidades, que em alguns casos acentuam e noutras casos fogem à regra geral enunciada – *maior diminuição da frequência máxima e maiores ficheiros de reconfiguração com a aplicação da estratégia de reconfiguração horizontal* –, permitindo uma análise mais aprofundada determinar o conjunto de regras que, em face das especificidades de cada implementação, deve presidir à escolha da estratégia a adoptar.

### 6.2.1 IMPACTO SOBRE O TEMPO DE PROPAGAÇÃO

Dos circuitos que acentuam a regra geral, em particular no que diz respeito à variação da frequência entre as duas estratégias, encontram-se os circuitos B04, B05, B07, B11, B13 e B14. Da observação da Tabela 6.1, conclui-se que este conjunto de circuitos tem como particularidade comum o recurso ao uso de linhas de transporte dedicadas na sua implementação.

Em anteriores gerações de dispositivos lógicos programáveis, a lógica aritmética era implementada colocando os blocos somadores num gerador de funções e a geração do valor do sinal de transporte

noutro. Uma vez que a velocidade das operações aritméticas é dominada pela velocidade da cadeia de transporte, a introdução de lógica de transporte e de linhas de encaminhamento dedicadas para este fim aumentou a velocidade das operações, enquanto duplicava a sua densidade [Trimberger, 93]. A quebra das cadeias de transporte constitui, por isso, uma das situações que maior impacto tem sobre a alteração dos tempos de propagação numa função.

**Tabela 6.2:** Variação da frequência e relação entre o tamanho global dos ficheiros de reconfiguração parcial com a aplicação das duas estratégias de rotação

Círcuito	Variação da freq. máxima (%)		Tamanho global dos ficheiros de reconfiguração parcial (bytes)		Relação percentual entre o tamanho global dos ficheiros de reconfiguração parcial (horizontal > vertical)
	Vertical	Horizontal	Vertical	Horizontal	
B01	-5,5	0,0	48 350	56 102	16,0
B02	0,0	0,0	7 016	10 623	51,4
B03	-1,9	-4,9	120 705	138 484	14,7
B04	-6,1	-29,3	548 595	665 419	21,3
B05	-17,3	-36,9	1 130 985	1 286 031	13,7
B06	-2,7	0,0	45 291	53 503	18,1
B07	-23,6	-37,8	354 367	425 214	20,0
B08	-5,8	-5,8	150 093	178 339	18,8
B09	-1,8	-4,9	112 107	129 855	15,8
B10	-7,5	-7,6	195 571	245 455	25,5
B11	-10,5	-36,0	500 261	614 093	22,8
B12	0,0	-1,2	1 275 804	1 631 953	27,9
B13	-4,3	-42,8	258 827	332 954	28,6
B14	-13,5	-47,8	5 195 444	6 070 485	16,8

No sentido de evidenciar esse impacto, recorreu-se à implementação de um circuito que, embora não faça parte do conjunto de circuitos-padrão em análise, permite confirmar essa importância. Trata-se de um contador binário de 24 bits, cujo modelo comportamental descrito em VHDL se apresenta na Figura 6.4. Após a síntese, distribuição e encaminhamento dos sinais, o contador

ficou colocado numa coluna de 12 CLBs, que, ocupando em cada CLB a *slice* 0, aproveita as linhas de transporte dedicadas entre CLBs adjacentes de uma mesma coluna, como se ilustra na Figura 6.5.

**Tabela 6.3:** Tamanho médio dos ficheiros de reconfiguração parcial por CLB ocupado com a aplicação das duas estratégias de rotação

Círcuito	Nº de CLBs ocupados	Tamanho médio dos ficheiros de reconfiguração por CLB (bytes)		Relação percentual entre o tamanho médio dos ficheiros de rec. por CLB (horizontal>vertical)
		Vertical	Horizontal	
B01	6	8 058	9 350	16,03
B02	1	7 016	10 623	51,41
B03	11	10 973	12 589	14,73
B04	54	10 159	12 322	21,30
B05	103	10 980	12 485	13,71
B06	5	9 058	10 700	18,13
B07	31	11 431	13 716	19,99
B08	17	8 829	10 490	18,82
B09	12	9 342	10 821	15,83
B10	20	9 778	12 272	25,51
B11	39	12 827	15 745	22,75
B12	119	10 721	13 713	27,92
B13	37	6 995	8 998	28,64
B14	333	15 601	18 229	16,84

Nas MPGAs ou nos ASICs, uma colocação próxima de todos os elementos constituintes de uma função significava quase sempre uma boa distribuição. Nas FPGAs, devido à topologia da estrutura de encaminhamento, pode ser preferível, como acontece no contador, uma distribuição que alinhe os elementos ao longo dos segmentos dedicados, em vez de os agrupar.

Da análise temporal efectuada a esta implementação, resultou uma frequência máxima de funcionamento para a configuração inicial do contador de aproximadamente 144MHz.

```

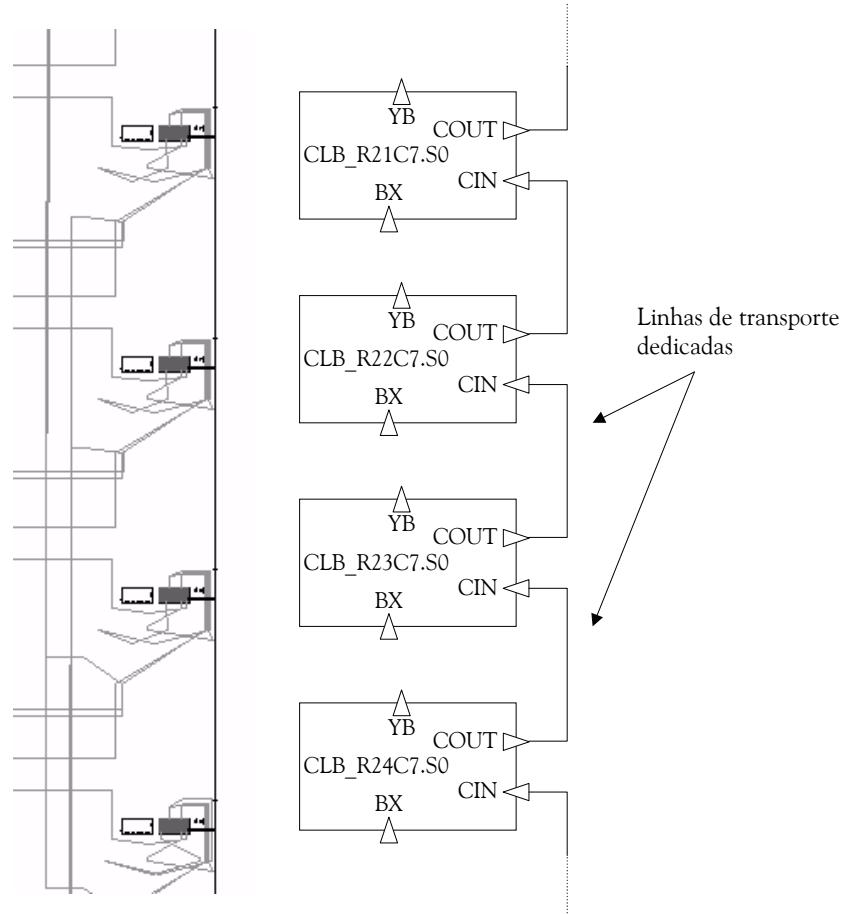
library IEEE;
use IEEE.std_logic_1164.all;

entity contador is
    port (
        CLK: in STD_LOGIC;
        Saida: out INTEGER range 0 to 16777215
    );
end contador;

architecture contador_arch of contador is
signal A: INTEGER range 0 to 16777215;
begin
    process(CLK, A)
    begin
        if (CLK='1' and CLK'event) then
            A <= A+1;
        end if;
        Saida<=A;
    end process;
end contador_arch;

```

**Figura 6.4:** Modelo comportamental do contador binário de 24 bits



**Figura 6.5:** Implementação física do contador e sua representação simplificada

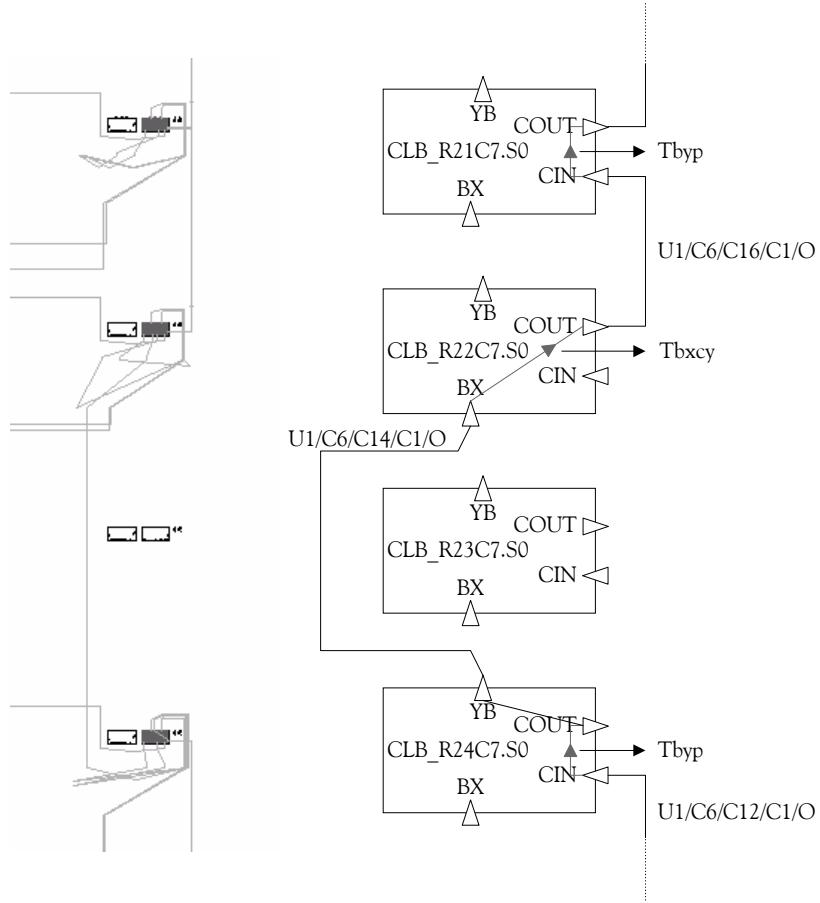
A aplicação da estratégia de rotação vertical resultou numa diminuição inicial da frequência máxima, devido à quebra de uma das linhas de transporte dedicadas e à sua substituição por recursos genéricos (entre os dois CLBs do fundo da coluna), tendo como consequência um aumento do tempo de propagação do sinal de transporte. Um exemplo concreto da distribuição desse acréscimo temporal é ilustrado na Figura 6.6, onde se mostra parte da implementação do contador de 24 bits, numa altura em que, empregando a estratégia de rotação vertical, o CLB da linha 23, coluna 7, é libertado para ser testado. Repare-se que, entre os CLBs que lhe estão adjacentes, a linha de transporte recorre a recursos genéricos de encaminhamento para a interligação da saída COUT do CLB\_R24C7.S0<sup>2</sup> com BX (entrada alternativa a CIN quando se recorre a recursos genéricos de encaminhamento para os sinais de transporte) do CLB\_R22C7.S0. De notar ainda, e ao contrário do que acontece com a ligação interna entre COUT e YB (CLB\_R24C7.S0), a ligação interna entre BX e COUT ( $T_{bxxy}$ ) apresenta um aumento no tempo de propagação, devido ao maior número de portas lógicas internas do CLB que o sinal tem de atravessar. Uma análise ao resultado da simulação temporal, cuja parte do ficheiro, correspondente ao atraso na propagação que se verifica no exemplo dado, é apresentada na Figura 6.7, permite concluir que, em relação à utilização de recursos de encaminhamento dedicados, o uso de recursos genéricos introduz na linha de transporte um aumento significativo no tempo de propagação dos sinais.

Na aplicação da estratégia de rotação vertical, em cada deslocamento de um CLB, a linha dedicada é quebrada num dos segmentos ao introduzir-se o CLB livre, mas repõe-se a ligação no ponto onde havia sido quebrada anteriormente. Deste modo, a cada instante, a linha de transporte não utiliza recursos dedicados entre apenas dois dos CLBs adjacentes abrangidos por essa linha. Logo, o atraso que se verifica aquando do primeiro deslocamento mantém-se constante ao longo do deslocamento de todos os restantes CLBs que constituem o contador, como se pode observar no gráfico da Figura 6.8. Este gráfico foi obtido a partir dos valores da frequência máxima de funcionamento do contador após cada deslocamento de um CLB, partindo da frequência máxima achada com a configuração inicial, pela aplicação quer da estratégia de rotação vertical, quer da de rotação horizontal. No caso do emprego de uma estratégia de rotação horizontal sobre o mesmo contador, a frequência máxima de funcionamento é cada vez mais baixa, pois a cada deslocamento horizontal é retirado um dos segmentos dedicados da linha de transporte, que, ao contrário do que acontece na rotação vertical, não volta a ser reposto, conduzindo a um acumular do atraso introduzido. Conclui-se, deste modo, que a utilização de uma estratégia de rotação horizontal provoca uma

---

<sup>2</sup> Tipo de elemento (CLB) Linha (R24) Coluna (C7) Slice (S0).

diminuição inaceitável na frequência de funcionamento de circuitos constituídos por funções aritméticas que façam uso de linhas de transporte dedicadas.



**Figura 6.6:** Parte da implementação física do contador e sua representação simplificada com um dos CLBs libertado para ser testado empregando uma estratégia de rotação vertical

```

Xilinx TRACE, Version D.23

Device speed:          xcv200,-4 (FINAL 1.111 2002-09-17)
Report level:          verbose report
-----
:
CLB_R24C7.S0.CIN    net (fanout=1) 0.000R      U1/C6/C12/C1/O
CLB_R24C7.S0.COUT   Tbyp                  0.109R      SAIDA<10>
CLB_R22C7.S0.BX   net (fanout=1) 1.497R  U1/C6/C14/C1/O
CLB_R22C7.S0.COUT Tbxcy                1.053R  SAIDA<12>
CLB_R21C7.S0.CIN    net (fanout=1) 0.000R      U1/C6/C16/C1/O
CLB_R21C7.S0.COUT   Tbyp                  0.109R      SAIDA<14>
:

```

**Figura 6.7:** Parte do ficheiro de simulação temporal do contador de 24 bits após uma das rotações

Percebe-se, por conseguinte, que os circuitos B04, B05, B07, B11, B13 e B14, ao empregarem na sua implementação linhas de transporte dedicadas, são grandemente sensíveis, em termos do atraso de

propagação, à sua quebra por aplicação das estratégias de rotação. A estratégia de rotação vertical, ao quebrar apenas um dos segmentos dedicados de cada vez, contribui de forma muito menos acentuada para a degradação da frequência máxima de funcionamento. Na estratégia horizontal, a cada relocação dos CLBs envolvidos na cadeia de transporte é quebrado um dos segmentos, tendo como resultado, após a relocação de todos esses CLBs, a quebra da totalidade dos segmentos. Esse processo origina um aumento cumulativo do atraso de propagação e uma consequente degradação acentuada da frequência máxima de funcionamento.

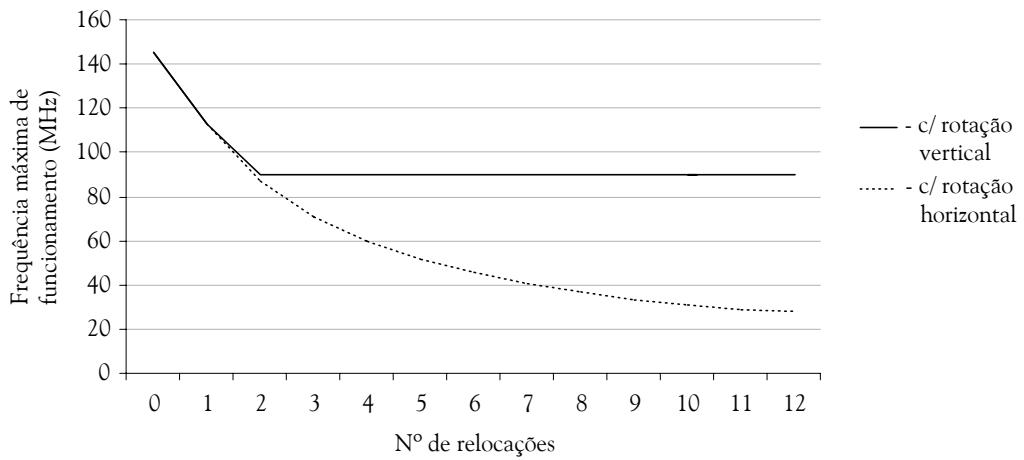


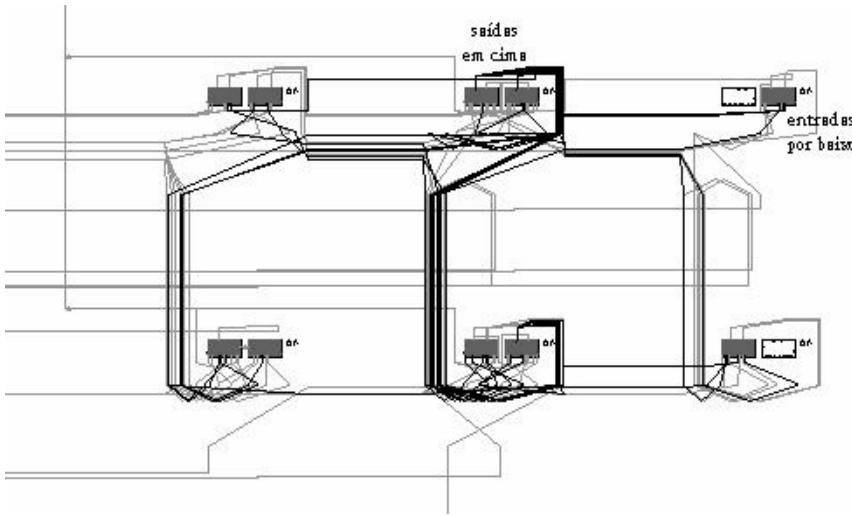
Figura 6.8: Variação da frequência máxima de funcionamento do contador

### 6.2.2 VECTORES DE PROXIMIDADE COMO MÉTRICA DE CUSTO

Nem todos os circuitos apresentam, contudo, um comportamento similar quanto à variação do atraso de propagação com a aplicação das estratégias de rotação. Da análise da Tabela 6.2, ressalta um conjunto de excepções, em particular nos circuitos B01 e B06, e B05 e B07. No primeiro conjunto, o dos circuitos B01 e B06, não existe degradação da frequência máxima de funcionamento aquando da aplicação da estratégia de rotação horizontal, o mesmo já não acontecendo com a aplicação da estratégia vertical, o que contraria o que se passa em todos os outros. No outro conjunto, o dos circuitos B05 e B07, já observados a propósito da quebra das linhas de transporte, os valores de degradação da frequência máxima de funcionamento são coerentes com os restantes, mas apresentam, mesmo na rotação vertical, valores bastante mais elevados, levantando dúvidas quanto à ideia de ser as linhas de transporte a única justificação para tão elevada diferença.

No caso da implementação do circuito B01, a disposição dos elementos deste circuito dentro da FPGA, efectuada pelo programa de distribuição e encaminhamento, ilustrada na Figura 6.9, adopta um arranjo predominantemente horizontal, resultando num número de segmentos horizontais

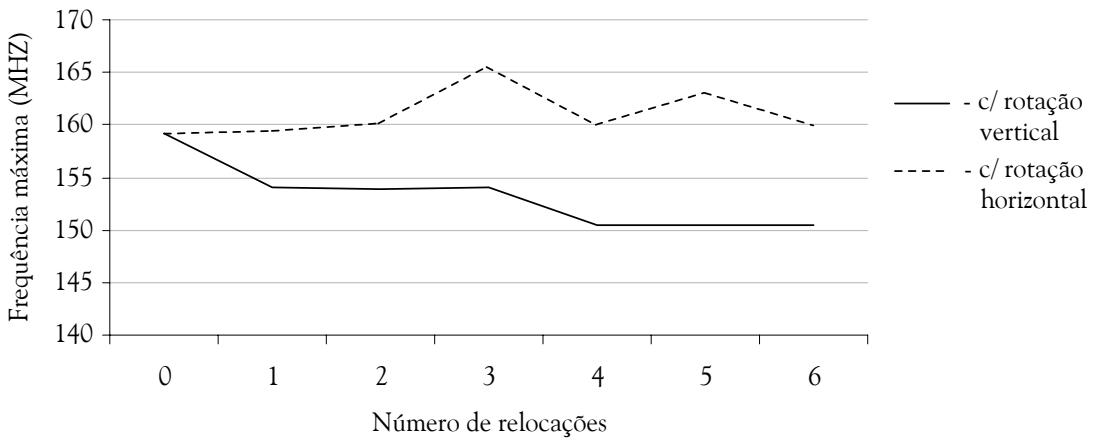
superior ao número de segmentos verticais. De salientar também a existência de três linhas com um número elevado de derivações, cuja origem se situa nos dois CLBs centrais da implementação, como se pode igualmente observar na Figura 6.9, onde essas linhas estão salientadas a cor negra. A sua centralidade e o arranjo adoptado derivam da necessidade de redução das distâncias entre o ponto de origem e os diversos pontos de destino, conduzindo a pequenos atrasos de propagação em todas as direcções. De notar que esta é a única hipótese de explicação plausível, uma vez que não são usados recursos de encaminhamento dedicados que imponham outro tipo de distribuição, como é o caso das disposições puramente verticais adoptadas no exemplo do contador, que derivam dos recursos disponibilizados para acelerar o funcionamento da lógica aritmética.



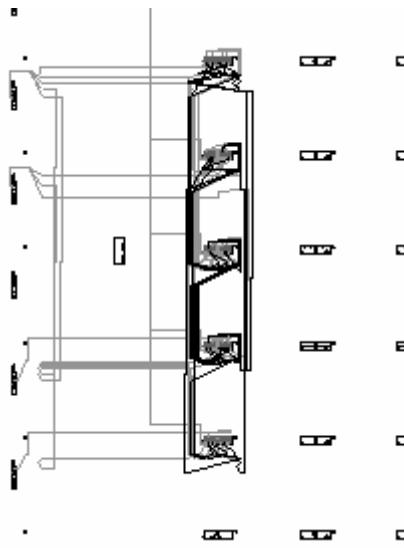
**Figura 6.9:** Implementação física do circuito B01

A aplicação de uma ou outra das estratégias de rotação conduz, como ilustrado no gráfico da Figura 6.10, a variações do valor da frequência máxima de funcionamento, distintas à medida que se vão processando as reconfigurações. Tal traduz uma diferente influência das duas estratégias sobre a variação dos tempos de propagação dos sinais, em especial daqueles que apresentam maior número de derivações, devido à consequente deslocação relativa dos blocos lógicos onde os mesmos têm origem. Estas observações levaram à consideração da influência do arranjo e da existência de linhas com elevado número de derivações, como possível explicação para a inversão, neste circuito, da tendência de variação da frequência com a aplicação das duas estratégias.

A implementação deste mesmo circuito, mas impondo ao programa de distribuição e encaminhamento restrições quanto à distribuição dos elementos de forma a obter-se uma disposição puramente vertical, conforme se ilustra na Figura 6.11, conduz aos resultados apresentados na Tabela 6.4.



**Figura 6.10:** Variação da frequência com a aplicação das duas estratégias de rotação ao circuito B01



**Figura 6.11:** Implementação física, com restrições à distribuição, do circuito B01

Como se observa, a variação da frequência aquando da aplicação de uma ou outra estratégia em relação à frequência inicial é positiva, o que significa que, ao contrário do que acontecia antes, não só as estratégias de rotação não introduzem degradação no funcionamento do circuito, mas o melhoram em relação à disposição inicial efectuada pelo programa de distribuição e encaminhamento, sem imposição de qualquer restrição. Todos os valores de frequência máxima obtidos pela aplicação, quer de uma quer de outra estratégia, são iguais ou superiores ao valor inicial, verificando-se que no final da aplicação da estratégia de rotação vertical, a frequência máxima é mesmo superior à inicial do circuito sem restrições. A não existência de linhas de transporte leva a que, até no caso da rotação horizontal, não se verifique uma degradação da frequência, mesmo impondo uma restrição de verticalidade na distribuição dos elementos. No entanto, a imposição de restrições à distribuição conduz, desde logo, a um valor inicial para a frequência máxima de funcionamento inferior em 7,1% relativamente ao obtido no circuito sem

restrições, o que, obviamente, afecta ambas as estratégias. Desta forma, conclui-se que a degradação ocorrida pela imposição de restrições à distribuição é superior à que resulta da simples aplicação da estratégia de rotação vertical à distribuição sem restrições. Este resultado é motivado pela perda de centralidade imposta, logo à partida, pelas restrições aplicadas à colocação dos blocos lógicos onde as linhas com elevado número de derivações têm origem, relativamente aos CLBs de destino de cada derivação. Essas restrições provocam aumentos nos atrasos de propagação, não só pelo aumento do comprimento da ligação, com pouco significado no aumento do atraso, mas principalmente porque obrigam ao travessamento de um maior número de PIPs para encaminhamento do sinal. O aumento do número de PIPs comprehende-se pela pequena distância a percorrer pelos sinais, dois ou três CLBs de permeio, que conduz ao uso de segmentos de pequeno comprimento que unem matrizes de encaminhamento global adjacentes, acontecendo que o acréscimo de um segmento implica imediatamente o acréscimo de, pelo menos, mais um PIP, ao invés de segmentos mais longos que se estendem ao longo de seis CLBs. No entanto, a aplicação das estratégias de rotação conduziu a uma optimização do encaminhamento, permitindo melhorar a situação inicial.

**Tabela 6.4:** Valores da frequência máxima encontrada para o circuito B01 com restrições

Rotação	Frequência máxima (Hz)	
	Vertical	Horizontal
Inicial	147 973	147 973
1	147 973	147 973
2	156 495	151 469
3	151 515	148 743
4	149 343	148 743
5	161 264	150 376
Variação	9,0 %	1,6 %

Para sistematizar esta análise, recorreu-se a um conceito análogo ao do cálculo do centro de massa de um conjunto de partículas, considerando-se cada ponto de destino de cada uma das derivações como as coordenadas de uma partícula, cuja ‘massa’ é igual para todas as derivações, e determinando os respectivos vectores de posição. Afere-se, desta forma, o seu grau de centralidade e os desvios que esse grau de centralidade sofre com a aplicação de ambas as estratégias, bem como as suas implicações.

Se a cada uma das derivações de uma linha de saída de um CLB se associar um vector que une a saída do CLB de origem com a entrada do CLB destino, denominado vector de proximidade, fazendo corresponder um factor de proximidade a cada um (comprimento do vector),  $f_{p_x}$ , proporcional à distância entre esses dois CLBs, conclui-se que a colocação óptima do CLB de origem, de modo a reduzir o atraso na propagação do sinal nessa linha, implica a minimização do somatório de cada factor associado a cada um dos vectores dessa linha, de tal forma a que o factor de proximidade dessa linha,  $F_{p_{lx}}$ , seja o menor possível.

**Equação 6.4**

$$F_{p_{lx}} = \min \sum_d (f_{p_1}, f_{p_2}, \dots, f_{p_d})$$

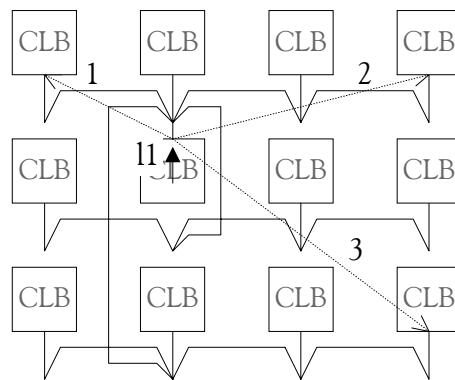
Genericamente, o factor de proximidade global de um circuito,  $F_{pg}$ , resultado do somatório dos factores de proximidade de todas as linhas, deve ser minimizado para reduzir os atrasos na propagação dos sinais nesse circuito, aumentando a sua frequência máxima de funcionamento.

**Equação 6.5**

$$F_{pg} = \min \sum_l (f_{p_{l1}}, f_{p_{l2}}, \dots, f_{p_{ll}})$$

A análise do impacto da aplicação de cada uma das estratégias de rotação sobre esse factor de proximidade global conduz a um conjunto de resultados que ajuda a compreender a razão dos desvios verificados em alguns circuitos e é determinante para a opção da estratégia a adoptar.

Na Figura 6.12, encontra-se o exemplo de uma distribuição optimizada dos recursos de forma a minimizar o factor de proximidade global de uma linha de saída, l1 (na figura apenas estão representados 3 dos 11 vectores de proximidade correspondentes à linha l1, por uma questão de clareza do desenho).



**Figura 6.12:** Disposição predominantemente horizontal com linha com elevado número de derivações

O CLB origem da linha l1 encontra-se numa posição central, minimizando os factores de proximidade associados a cada um dos vectores ( $f_{p_1}, f_{p_2}, f_{p_3}, \dots$ ), o que contribui para a minimização

do factor de proximidade da linha l1,  $F_{p_{l1}}$ , e consequente minimização do atraso de propagação associado ao sinal presente nessa linha. Se se usar um sistema por unidade (p.u.), em que uma unidade corresponde ao espaçamento de uma linha ou de uma coluna de CLBs, verifica-se que o comprimento do vector 1 será:

**Equação 6.6**

$$f_{p_1} = \sqrt{1^2 + 1^2} = \sqrt{2}$$

De igual forma para o vector 2:

**Equação 6.7**

$$f_{p_2} = \sqrt{1^2 + 2^2} = \sqrt{5}$$

e assim sucessivamente para os 11 vectores possíveis, considerando o caso genérico em que a linha l1 tem derivações para todos os CLBs.

A aplicação, neste exemplo, de restrições à distribuição, forçando a uma disposição predominantemente vertical, conduz à situação ilustrada na Figura 6.13. Como se observa, a restrição na distribuição dos recursos conduz a uma distribuição não optimizada, com consequente aumento do factor de proximidade global associado à linha l1, resultante do aumento do factor de proximidade associado a cada uma das suas derivações e implicando um aumento do atraso de propagação associado ao sinal presente nessa linha, se comparada com a situação anterior. Essa influência será tanto maior quanto maior for o número de derivações de uma linha, pois os recursos a atingir situar-se-ão cada vez mais longe, com o consequente aumento do comprimento dos vectores.

De igual forma, a aplicação da estratégia de rotação implica alterações no comprimento dos vectores e, como tal, uma variação do factor de proximidade da linha l1, resultando numa variação mais ou menos acentuada da frequência máxima de funcionamento do circuito implementado. A Figura 6.14 mostra o resultado da aplicação da estratégia de rotação vertical ao exemplo apresentado. Observa-se claramente que a rotação implica um aumento do factor de proximidade da linha l1, embora, num vector ou outro, se possa encontrar uma diminuição do valor do factor de proximidade associado. A Figura 6.15 ilustra a aplicação da estratégia de rotação horizontal ao mesmo exemplo.

Em termos numéricos, tem-se para o valor do factor de proximidade inicial da linha l1:

**Equação 6.8**

$$F_{p_{l1}} = f_{p_1} + f_{p_2} + \dots + f_{p_{l1}} = 16,13$$

Após a aplicação da estratégia vertical:

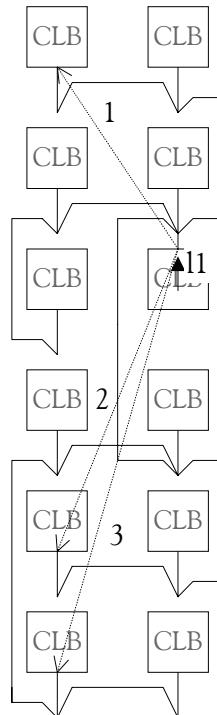
**Equação 6.9**

$$F_{p_{11}} = f_{p_1} + f_{p_2} + \dots + f_{p_{11}} = 22,10$$

e da estratégia horizontal:

**Equação 6.10**

$$F_{p_{11}} = f_{p_1} + f_{p_2} + \dots + f_{p_{11}} = 20,45$$



**Figura 6.13:** Disposição predominantemente vertical com linha com elevado número de derivações

Como se verifica pela comparação entre a Equação 6.9 e a Equação 6.10, a degradação do factor de proximidade da linha l1, ocorrida com a aplicação da estratégia vertical, é superior à resultante da aplicação da estratégia horizontal.

De igual modo, calcula-se o factor de proximidade da linha l1 para o caso da imposição de restrições à distribuição, de forma a obter-se uma disposição predominantemente vertical (Figura 6.13). Em termos numéricos, tem-se para o valor do factor de proximidade inicial da linha l1:

**Equação 6.11**

$$F_{p_{11}} = f_{p_1} + f_{p_2} + \dots + f_{p_{11}} = 20,46$$

Após a aplicação da estratégia vertical:

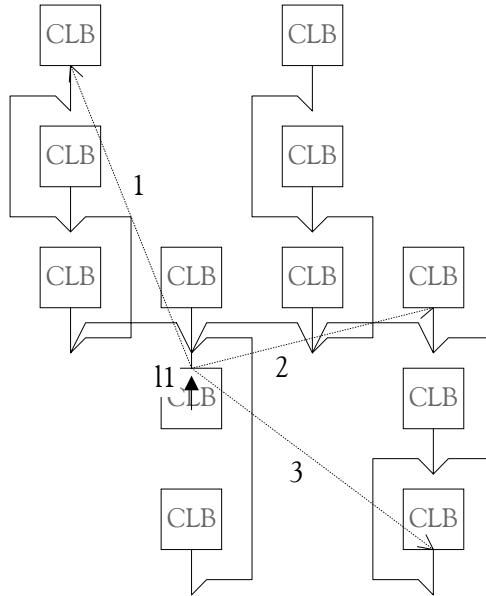
**Equação 6.12**

$$F_{p_{11}} = f_{p_1} + f_{p_2} + \dots + f_{p_{11}} = 22,35$$

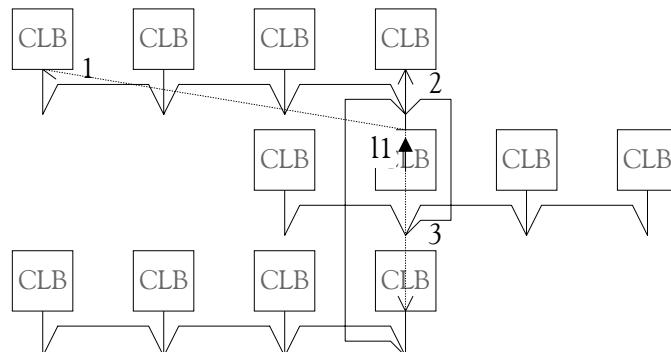
e da estratégia horizontal:

**Equação 6.13**

$$F_{p_{11}} = f_{p_1} + f_{p_2} + \dots + f_{p_{11}} = 23,54$$



**Figura 6.14:** Aplicação da estratégia de rotação vertical a uma disposição predominantemente horizontal com linha com elevado número de derivações



**Figura 6.15:** Aplicação da estratégia de rotação horizontal a uma disposição predominantemente horizontal com linha com elevado número de derivações

Pela comparação dos valores do factor de proximidade inicial da linha l1 em ambos os casos, conclui-se que a imposição de restrições, de forma a alterar uma disposição predominantemente horizontal para uma predominantemente vertical, deteriora logo à partida o valor do factor de proximidade, explicando o decréscimo verificado na frequência máxima, no caso da implementação do circuito B01 com restrições.

Em relação aos valores obtidos com a aplicação das estratégias de rotação a ambos os casos, constata-se uma variação inversa dos valores do factor de proximidade da linha l1. No caso de uma

disposição predominantemente horizontal, o factor de proximidade, após a aplicação da estratégia horizontal, é inferior ao mesmo factor, no caso da aplicação da estratégia vertical (Equação 6.9 e Equação 6.10), verificando-se o inverso no caso de uma disposição predominantemente vertical (Equação 6.12 e Equação 6.13). Esta tendência está em consonância com as variações verificadas, na prática, na frequência máxima de funcionamento do circuito B01, tendo a aplicação destes conceitos ao referido circuito conduzido aos resultados apresentados na Tabela 6.5.

**Tabela 6.5:** Quadro resumo da aplicação dos factores de proximidade ao circuito B01

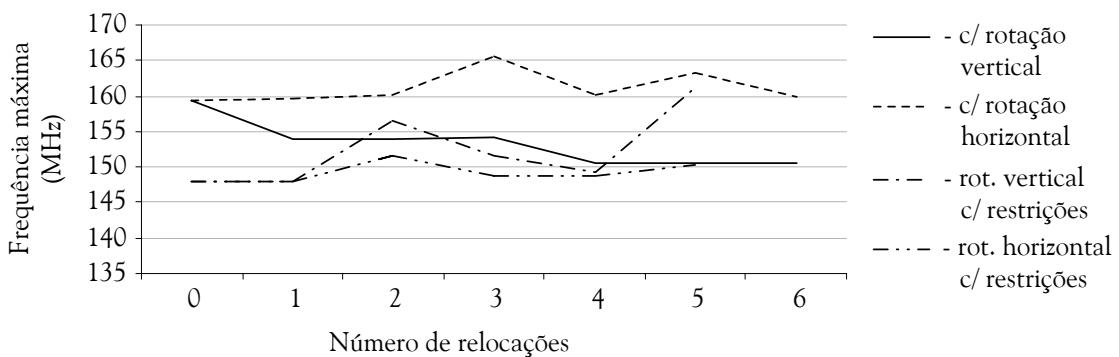
	sem restrições	com restrições
Nº de segmentos horizontais	25	0
Nº de segmentos verticais	19	29
Factor de proximidade global do circuito		
Inicial	36,97	50
Após a aplicação da estratégia de rotação horizontal	54,36	70,70
Após a aplicação da estratégia de rotação vertical	63,52	50

Analizando a primeira situação, a implementação sem restrições, verifica-se que o número de segmentos horizontais é superior ao número de segmentos verticais, pelo que o factor de proximidade global do circuito, após a aplicação de ambas as estratégias, segue as conclusões retiradas anteriormente. Daí a degradação verificada na frequência máxima com a aplicação da estratégia vertical.

Na análise da segunda situação, a implementação com restrições, desaparecem os segmentos horizontais, pelo que não existe alteração do factor de proximidade com a aplicação da estratégia vertical. No entanto, e tendo em conta a presença neste circuito de três linhas com elevado número de derivações, observa-se uma degradação do factor de proximidade global inicial do circuito, resultado da perda de centralidade dos blocos lógicos onde essas linhas têm origem, conforme se observa na Figura 6.11, e o consequente aumento do factor de proximidade associado a cada uma dessas linhas.

Conclui-se, desta forma, que a degradação da frequência máxima segue a tendência verificada nos valores do factor de proximidade global do circuito, isto é, o aumento do factor de proximidade implica uma diminuição da frequência máxima de funcionamento e vice-versa. De notar, no entanto, que o valor do factor de proximidade global, depois da aplicação da estratégia de rotação vertical, no caso com restrições, é mais pequeno que qualquer um dos outros no final da aplicação das estratégias de rotação, quer no caso com restrições, quer no caso sem restrições. Tal resultado está em consonância com os valores obtidos para a frequência máxima, listados na Tabela 6.4.

A Figura 6.16 mostra a variação da frequência máxima com a aplicação de ambas as estratégias de rotação aos casos da implementação do circuito B01 sem e com restrições.



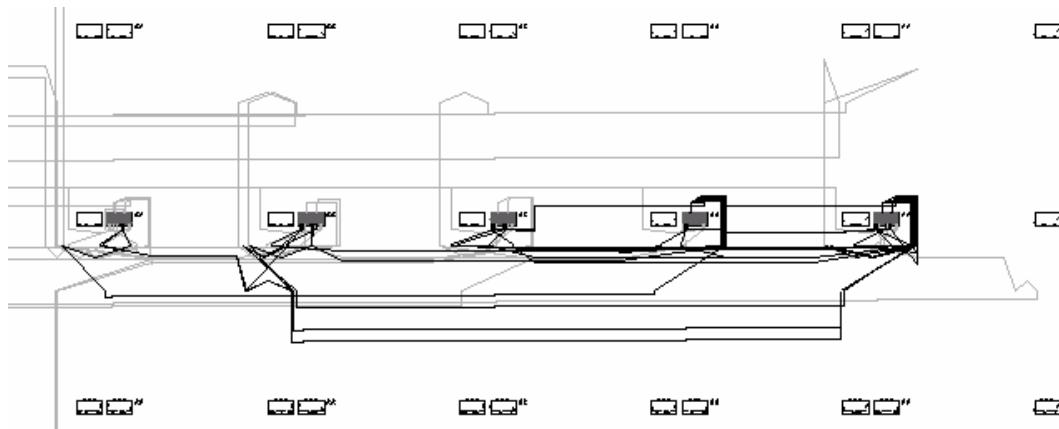
**Figura 6.16:** Variação da frequência máxima com a aplicação das duas estratégias de rotação à implementação sem e com restrições à distribuição do circuito B01

Em síntese, verifica-se uma excelente consonância entre os resultados obtidos pela métrica proposta e a variação, na prática, da frequência máxima.

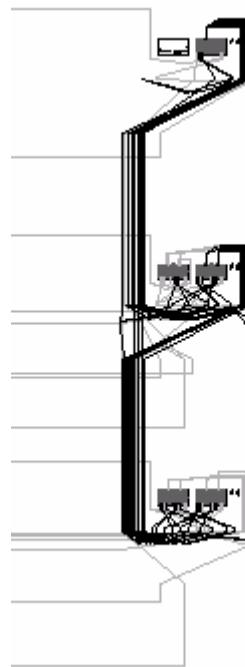
Na implementação do circuito B06, a disposição dos elementos, ilustrada na Figura 6.17, é puramente horizontal, apresentando igualmente três linhas, salientadas a negro na figura, com um número elevado de derivações. De notar que a aplicação da estratégia horizontal a este circuito mantém a sua horizontalidade intacta, ao contrário do que sucede com a aplicação da estratégia vertical, pelo que se verifica, na aplicação desta última estratégia, uma diminuição da frequência máxima.

Impondo igualmente a este circuito uma restrição de verticalidade para a disposição dos componentes, obtemos como resultado a distribuição ilustrada na Figura 6.18. Com a imposição de restrições, embora o número de *slices* usadas seja o mesmo, o número de blocos lógicos ocupados pelo circuito é menor, pelo que o factor de proximidade global do circuito diminui em relação ao valor obtido para a colocação anterior, como se mostra na Tabela 6.6. Isso reflecte-se na frequência máxima de funcionamento do circuito, que cresce em ambos os casos, como se indica na

Tabela 6.7, e cuja evolução comparativa com a aplicação de cada uma das estratégias se pode observar na Figura 6.19. Tal como no caso de B01, o valor relativo dos factores de proximidade obtido é um bom indicador da tendência de variação da frequência máxima de funcionamento com a aplicação de ambas as estratégias e da melhoria que, neste caso, a implementação com restrições trouxe ao funcionamento do circuito.



**Figura 6.17:** Implementação física do circuito B06



**Figura 6.18:** Implementação física, com restrições à distribuição, do circuito B06

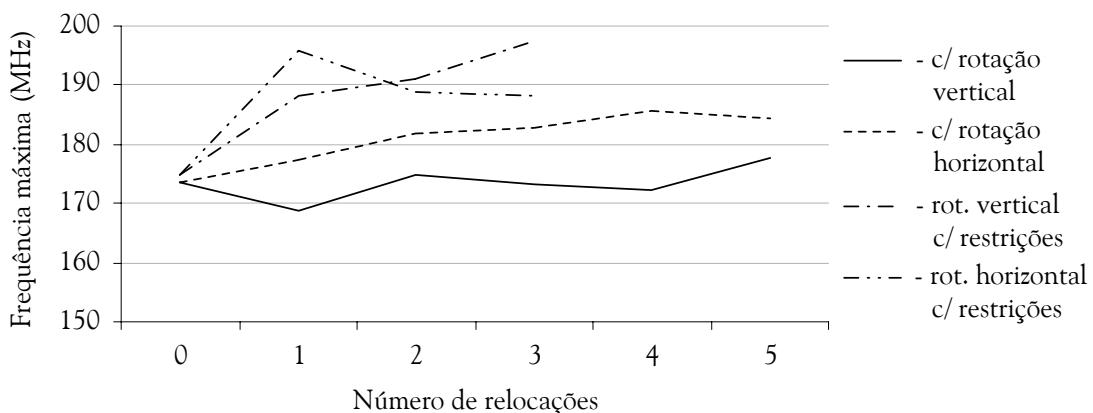
A simples observação da Tabela 6.6 permite decidir, claramente e de imediato, por qual das implementações e estratégia de rotação se deve optar para esta função.

**Tabela 6.6:** Quadro resumo da aplicação dos factores de proximidade ao circuito B06

Factor de proximidade global	sem restrições	com restrições
Inicial	41	21
Após a aplicação da estratégia de rotação horizontal	41	26,38
Após a aplicação da estratégia de rotação vertical	51,44	21

**Tabela 6.7:** Valores máximos da frequência de funcionamento de B06 em ambos os casos após a aplicação das estratégias de rotação

Tipo de implementação	Frequência máxima (Hz)	
	Vertical	Horizontal
sem restrições	177 651	184 332
com restrições	197 511	188 324
%	3,66	0,82



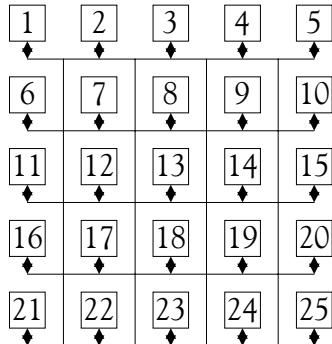
**Figura 6.19:** Variação da frequência máxima com a aplicação das duas estratégias de rotação à implementação sem e com restrições à distribuição do circuito B06

### 6.2.3 INFLUÊNCIA DO NÚMERO DE DERIVAÇÕES E DA DISPOSIÇÃO SOBRE O CUSTO

Pela análise da aplicação da teoria exposta aos casos dos circuitos B01 e B06, depreende-se que a variação do factor de proximidade de cada linha, com a aplicação de cada uma das estratégias,

depende da posição do CLB onde essa linha tem origem, bem como do número de derivações que ela apresenta.

Considerando, por generalização, um circuito de 25 blocos com disposição em quadrado, em que de cada bloco sai uma linha com derivação para todos os outros, conforme se esquematiza na Figura 6.20, conclui-se, pela análise do gráfico apresentado na Figura 6.21, que a variação do valor do factor de proximidade global para cada uma das linhas, com a aplicação de ambas as estratégias, depende da posição dentro da distribuição do CLB onde essas linhas têm origem. De notar que para os CLBs 1, 7, 13, 19 e 25, que estão situados na diagonal da implementação, a variação do factor de proximidade global das linhas, resultante da aplicação de ambas as estratégias, é igual. Em dez das distribuições, a estratégia vertical conduz a factores de proximidade inferiores aos resultantes da aplicação da estratégia horizontal, verificando-se o contrário nas restantes dez. No total, a estratégia de rotação vertical apresenta uma vantagem de 0,4% no factor de proximidade global do circuito. O gráfico apresentado na Figura 6.22 mostra a relação percentual entre a aplicação das duas estratégias em função da posição de cada CLB, indicando os valores negativos vantagem para a estratégia vertical e os positivos vantagem para a horizontal.

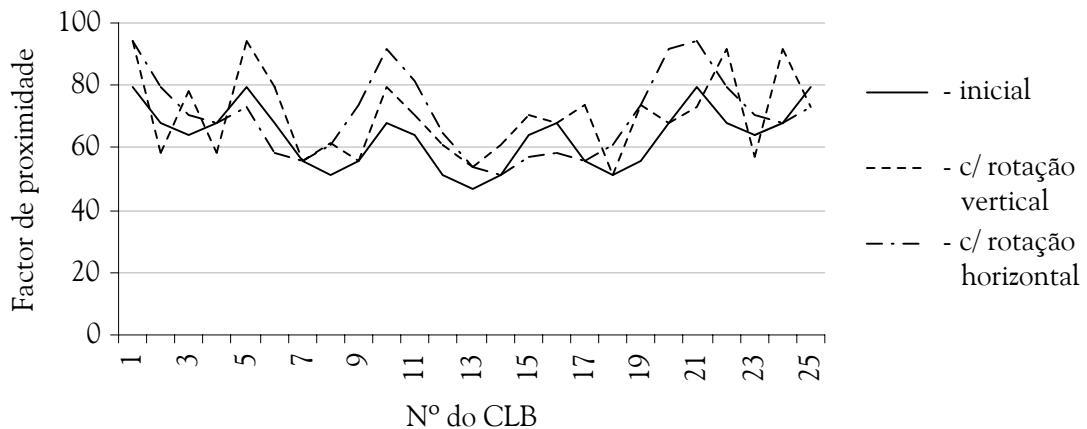


**Figura 6.20:** Circuito com 25 blocos lógicos, onde cada um apresenta uma linha com derivação para todos os restantes

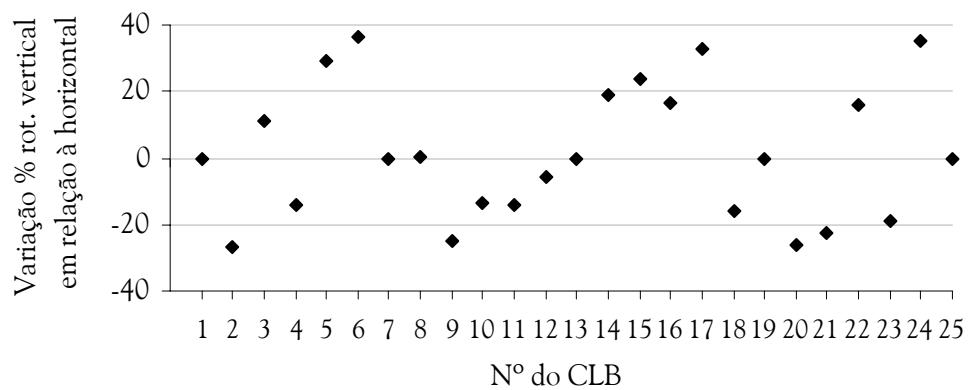
Como o factor de proximidade está intimamente relacionado com o atraso de propagação, uma não centralidade dos CLBs onde as linhas com elevado número de derivações têm origem conduz a valores de atraso elevados nessas linhas, os quais se degradam ainda mais com a aplicação de qualquer uma das estratégias. A variação da frequência máxima de funcionamento, que representa o principal custo da aplicação das estratégias de rotação (tendo em conta que a sua variação pode afectar o funcionamento dos circuitos), depende, pois, do número de linhas com elevado número de derivações e da posição dos CLBs em que essas linhas têm origem.

De igual modo, e como se pode verificar no caso dos circuitos B01 e B06, a disposição dos blocos lógicos é também importante. Generalizando para o caso de um circuito com 10 CLBs e uma

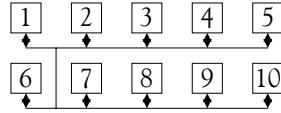
disposição predominantemente horizontal, em que cada CLB é origem de uma linha com derivações para todos os outros, conforme se ilustra na Figura 6.23, verifica-se alguma vantagem na aplicação da estratégia horizontal, como se comprova pelo gráfico apresentado na Figura 6.24. Com excepção dos CLBs situados em cada um dos extremos opostos (em que a estratégia vertical é menos penalizadora) ou dos situados no centro da distribuição (situação em que é indiferente o uso de uma ou outra estratégia), verifica-se uma acentuada diminuição do factor de proximidade de cada linha, inclusive para valores inferiores aos da configuração inicial. Pelo contrário, a aplicação da estratégia vertical traduz-se sempre numa deterioração desse factor em relação ao seu valor inicial, apresentando quase sempre piores resultados relativamente à estratégia horizontal. Considerando o factor de proximidade global do circuito, a estratégia horizontal resulta num valor 4% inferior, face ao valor obtido após a aplicação da estratégia vertical.



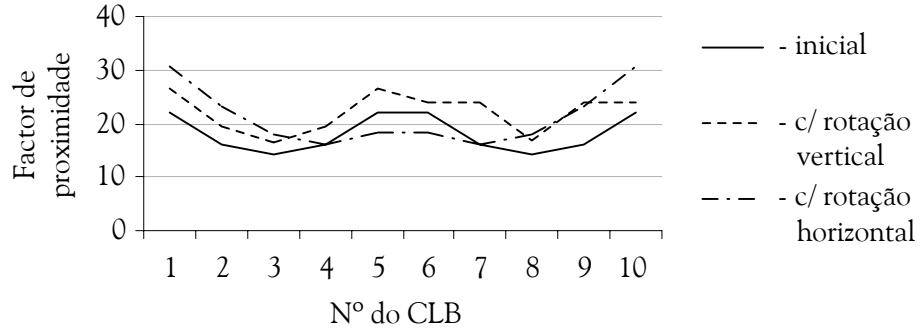
**Figura 6.21:** Valor do factor de proximidade para uma linha saindo de um CLB com derivação para todos os outros



**Figura 6.22:** Diferença percentual entre os factores de proximidade para cada CLB, resultante da aplicação da estratégia vertical relativamente à horizontal (valores negativos indicam vantagem da aplicação da estratégia vertical)

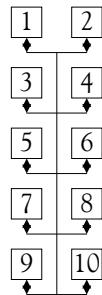


**Figura 6.23:** Circuito com 10 blocos lógicos, onde cada um apresenta uma linha com derivação para todos os restantes, com uma disposição predominantemente horizontal



**Figura 6.24:** Comparação entre os valores do factor de proximidade para uma linha saindo de um CLB com derivação para todos os outros, numa disposição predominantemente horizontal

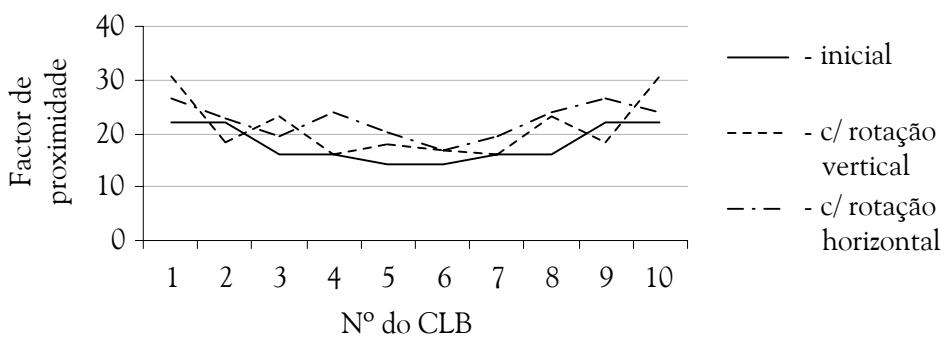
Por outro lado, uma disposição dos blocos lógicos predominantemente vertical, conforme se ilustra na Figura 6.25, reverte numa clara vantagem da aplicação da estratégia vertical, com excepção dos CLBs mais longe do centro da distribuição, como se mostra no gráfico apresentado na Figura 6.26. Mais uma vez se verifica que a centralidade dos blocos com elevado número de derivações tem influência decisiva no valor do factor de proximidade e, consequentemente, na degradação do valor da frequência máxima com a aplicação das estratégias de rotação. Considerando o factor de proximidade global do circuito, a estratégia vertical resulta num valor 5% inferior, face ao valor obtido após a aplicação da estratégia horizontal.



**Figura 6.25:** Circuito com 10 blocos lógicos, onde cada um apresenta uma linha com derivação para todos os restantes, com uma disposição predominantemente vertical

No caso dos circuitos B05 e B07, a elevada degradação do valor da frequência máxima com a aplicação da estratégia vertical pode igualmente ser interpretada à luz deste estudo. Em relação ao circuito B05, a disposição dos CLBs na implementação é tendencialmente vertical, sendo

determinada pela existência de 4 linhas de transporte que utilizam os recursos de encaminhamento dedicados disponibilizados pelas Virtex. Essas linhas de transporte são responsáveis pela elevada degradação da frequência máxima verificada aquando da aplicação da estratégia de rotação horizontal, porque esta origina a sua quebra, passando o encaminhamento a ser efectuado por recursos genéricos. No entanto, em virtude da existência de cinco linhas que apresentam um grande número de derivações (55, 86, 90 e duas com 93 derivações), a implementação apresenta igualmente uma quantidade elevada de segmentos horizontais que, devido à sua quebra com a aplicação da estratégia de rotação vertical e consequente aumento do factor de proximidade global do circuito, são responsáveis pela considerável degradação verificada no valor da frequência máxima de funcionamento com a aplicação da estratégia vertical.



**Figura 6.26:** Comparação entre os valores do factor de proximidade para uma linha saindo de um CLB com derivação para todos os outros, numa disposição predominantemente vertical

Por sua vez, o comportamento do circuito B07 é determinado pela presença de três linhas que apresentam um grande número de derivações (43, 22 e 16), que resultam numa quantidade elevada de segmentos horizontais, acentuada pela disposição predominantemente horizontal, originando um acentuado incremento do factor de proximidade global e a consequente degradação do valor da frequência máxima de funcionamento do circuito com a aplicação da estratégia de rotação vertical. Duas linhas de transporte, ainda que de comprimento bastante curto (três segmentos cada), são responsáveis pela elevada degradação da frequência máxima verificada com a aplicação da estratégia de rotação horizontal. Em termos de ficheiros de reconfiguração e em qualquer dos dois casos, a vantagem vai para a rotação vertical.

#### 6.2.4 DETERMINAÇÃO DA ESTRATÉGIA DE MAIS BAIXO CUSTO

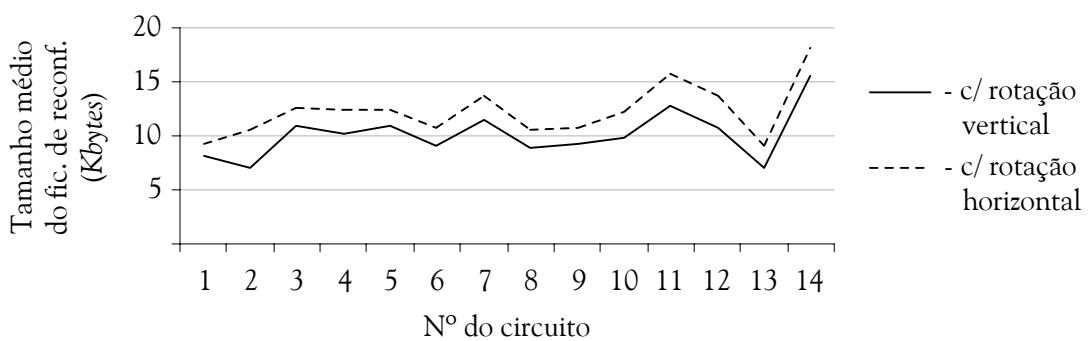
Em conclusão, pode considerar-se que os principais factores que afectam a variação do valor do factor de proximidade global de um circuito, aquando da aplicação de uma das estratégias de rotação, são:

- a posição dos blocos lógicos onde têm origem linhas com elevado número de derivações;
- a forma da distribuição dos blocos lógicos constituintes do circuito.

Estes pontos devem ser tidos em conta na escolha da estratégia de rotação a adoptar para uma determinada implementação de um circuito, de forma a evitar uma elevada degradação do seu funcionamento com a aplicação das estratégias de rotação. Exceptua-se o caso dos circuitos que apresentem linhas de transporte, pois, nesta situação, deve ser sempre adoptada a estratégia vertical.

Para além de permitir uma rápida decisão sobre qual a estratégia de rotação a utilizar no sentido de minimizar a degradação na frequência máxima de funcionamento, a métrica apresentada tem ainda a vantagem de ser facilmente integrável nas ferramentas de síntese, distribuição e encaminhamento actualmente disponíveis.

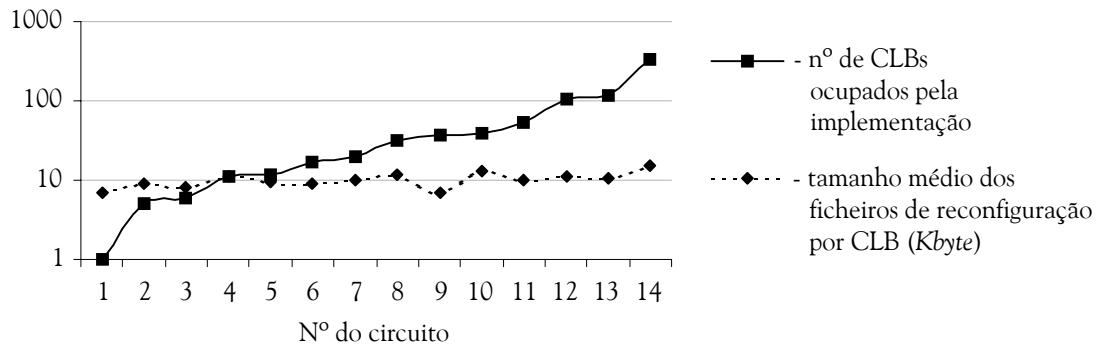
É ainda importante reter que a aplicação da estratégia horizontal resulta sempre em ficheiros de reconfiguração parcial maiores do que os exigidos para a aplicação da estratégia vertical. O gráfico da Figura 6.27 apresenta os valores médios dos ficheiros de reconfiguração por CLB para cada um dos circuitos B01 a B14. Nota-se uma tendência para um ligeiro aumento do tamanho dos ficheiros com o aumento do tamanho do circuito, provocado por uma maior dificuldade em reencaminhar as linhas afectadas pela relocação dos CLBs. No entanto, a um aumento exponencial do tamanho dos circuitos corresponde um aumento linear, pouco acentuado, do tamanho dos ficheiros, conforme se ilustra na Figura 6.28, para o caso da estratégia de rotação vertical. O mesmo acontece no caso da aplicação da estratégia de rotação horizontal.



**Figura 6.27:** Tamanho médio dos ficheiros de configuração por CLB para os circuitos B01 a B14, aplicando ambas as estratégias de rotação

É de referir, todavia, que, em termos da aplicação da metodologia de teste, é mais importante a não perturbação do funcionamento do circuito do que um aumento no tempo total de teste, pelo que é preferível a estratégia de mais baixo custo em termos do nível de degradação da frequência máxima

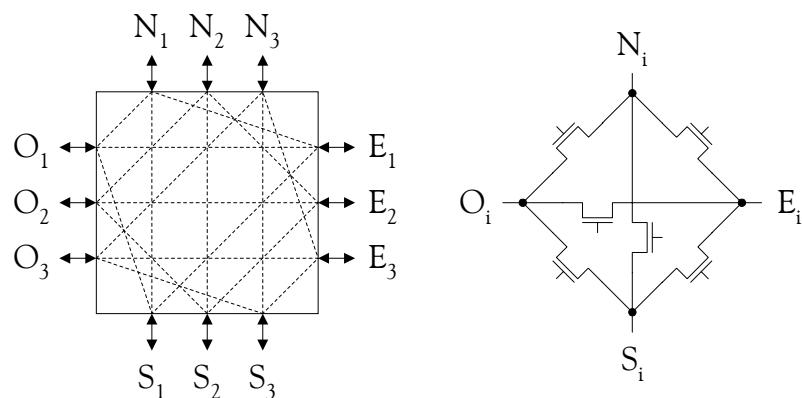
de funcionamento, independentemente de, em alguns casos, poder ser aquela que apresenta um custo mais elevado em termos do tamanho dos ficheiros de reconfiguração.



**Figura 6.28:** Comparação entre o tamanho médio dos ficheiros de reconfiguração por CLB e o tamanho do circuito

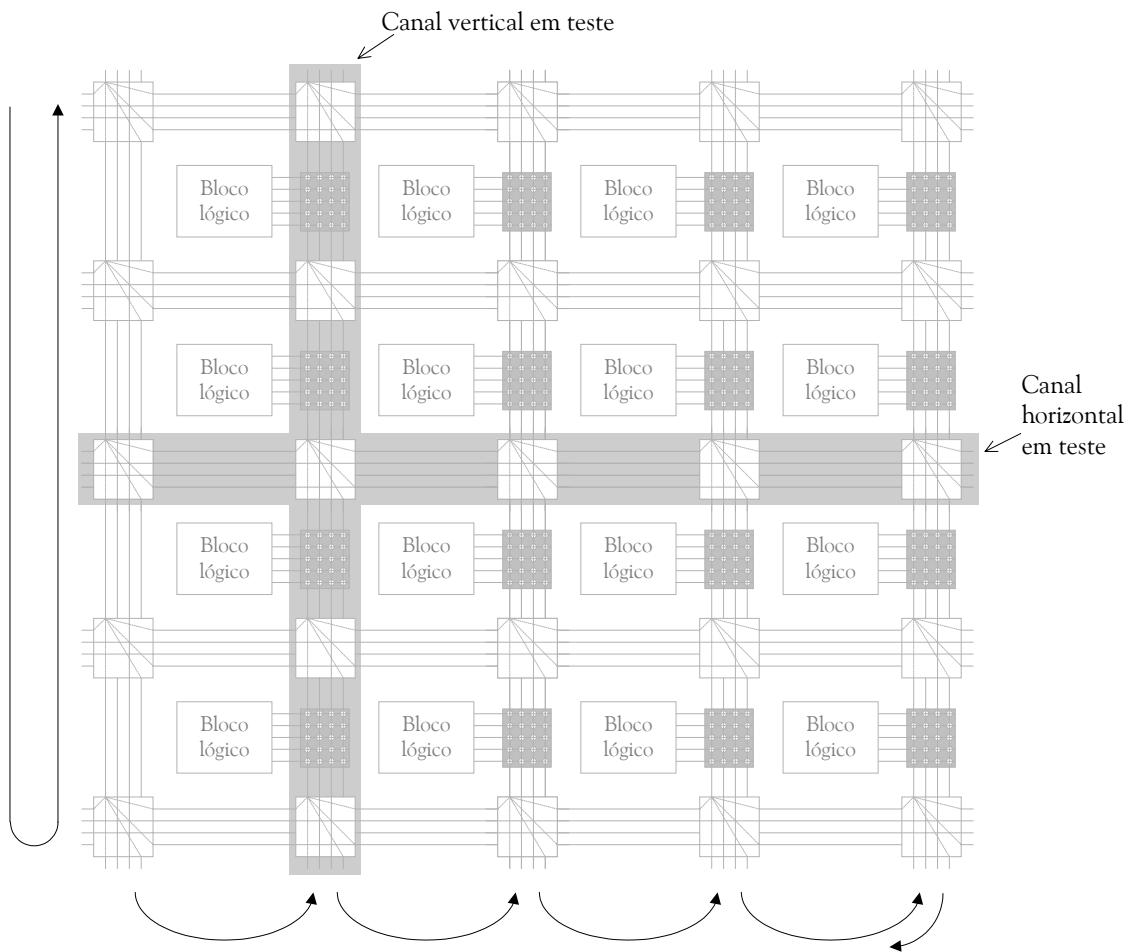
### 6.3. ROTAÇÃO DAS INTERLIGAÇÕES

Os recursos de interligação global que se estendem vertical ou horizontalmente ao longo de canais são constituídos por conjuntos de segmentos de comprimento unitário (um CLB), permitindo estabelecer interligações entre blocos lógicos adjacentes, de comprimento seis CLBs, permitindo estabelecer interligações entre CLBs distantes seis linhas ou colunas entre si, com uma derivação para um CLB a meio da distância, ou com um comprimento que abrange toda a altura ou largura da FPGA, com derivações a cada seis CLBs. As matrizes de interligação global, colocadas no ponto de cruzamento desses canais, permitem efectuar o encaminhamento dos sinais através destes recursos, por intermédio de PIPs que direccionam, em cada matriz, os sinais que aí chegam numa das três direcções possíveis, como simplificadamente se representa na Figura 6.29.



**Figura 6.29:** Modelo simplificado de uma célula da matriz de interligação global

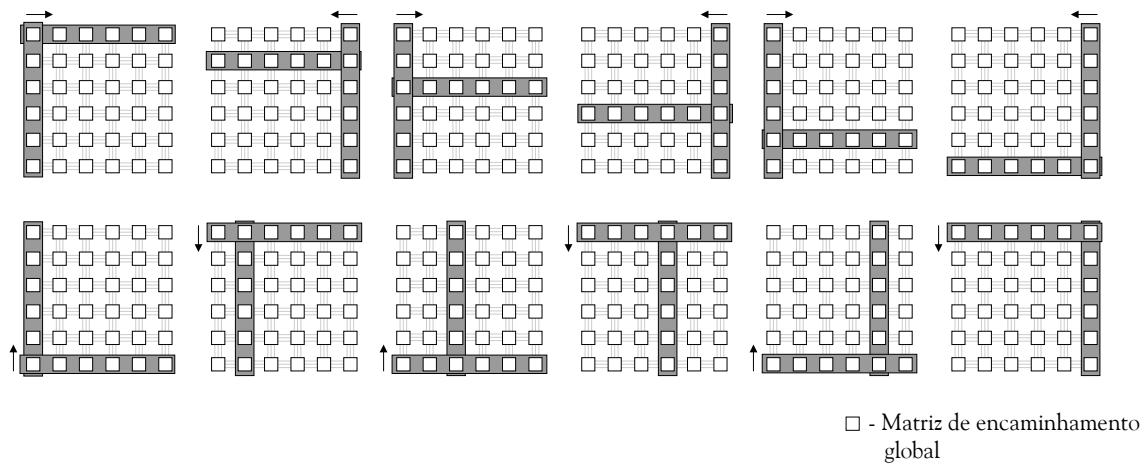
Ao possibilitarem a mudança de direcção dos sinais que chegam à matriz, reencaminhando-os dos canais horizontais para os verticais ou vice-versa, o teste da totalidade dos PIPs de todas as matrizes de interligação global obriga à existência de uma configuração de teste para todos os pares possíveis de posições dos canais horizontal e vertical em teste, sempre com dois canais em teste simultaneamente, conforme ilustrado na Figura 6.30. O teste da totalidade dos canais e dos recursos de encaminhamento das matrizes globais implica um varrimento completo do canal vertical em teste por cada deslocamento do canal horizontal ou vice-versa. A direcção do varrimento inverte-se após cada varrimento completo de um canal.



**Figura 6.30:** Rotação dos canais de interligação sob teste

Deste modo, o teste completo de todas as matrizes implica o uso de um número de configurações de teste igual ao produto do número de linhas pelo número de colunas da matriz. Em FPGAs de grande tamanho, isso conduz a um elevado aumento do tempo de latência para todas as faltas. Uma solução de compromisso para evitar esse acréscimo tão significativo é o teste de apenas um subconjunto dos recursos de encaminhamento por cada sessão de teste da FPGA, entendida como o teste de todos os CLBs, mais o varrimento completo de um dos canais em teste, enquanto se

mantém fixo o outro canal em teste. Assegurando que cada subconjunto é diferente em cada uma das sessões garante-se o teste da totalidade dos recursos de encaminhamento ao fim de um número de sessões igual ao número de canais horizontais ou verticais da FPGA. Uma das possibilidades é apresentada na Figura 6.31. Este compromisso foi já anteriormente assumido por outros autores, visto ser impossível o teste da totalidade dos recursos de interligação sem recorrer a um número elevado de configurações de teste, problema que se agudiza quando se pretende implementar um teste concorrente, em que parte significativa destes recursos não pode ser simultaneamente libertada para a aplicação de testes [Abramovici et al., 01a].



**Figura 6.31:** Exemplo da rotação dos canais em teste

Atente-se que, embora para uma matriz-exemplo de seis-por-seis se apresente na Figura 6.31 um total de doze situações, em que o canal fixo alterna entre o horizontal nas primeiras seis e o vertical nas seis seguintes, o teste da totalidade dos recursos é alcançado ao fim das primeiras seis sessões, repetindo-se nas seis seguintes. É, pois, possível optar por manter fixo o mesmo canal e repetir sempre a mesma sequência em ambos os sentidos. No entanto, tal conduz a um desequilíbrio no número de vezes em que alguns dos recursos são testados. Por exemplo, se se mantiver fixo o canal horizontal, ao fim de seis sessões, cada canal vertical esteve em teste seis vezes, enquanto cada canal horizontal esteve em teste apenas uma vez. Isso não significa que todos os recursos exclusivos do canal vertical tenham sido testados seis vezes, mas que alguns desses recursos, usados para permitir, por exemplo, o encaminhamento dos vectores de teste e a recolha de resultados dentro do canal em teste, são, na prática, testados mais do que uma vez. A troca da escolha do canal fixo tem por objectivo anular este desequilíbrio ao fim do dobro das sessões necessárias ao teste efectivo da totalidade dos recursos de encaminhamento.

## 6.4. SUMÁRIO

Neste capítulo, foram descritas as estratégias propostas para a rotação do CLB livre sem perturbação da operação normal do sistema, permitindo o varrimento pelos procedimentos de teste de todos os blocos lógicos. O objectivo da rotação é o estabelecimento de uma estratégia para a contínua relocação dos CLBs activos no CLB livre que, percorrendo a totalidade dos recursos, permite a sua libertação para o teste.

A análise de um conjunto vasto de implementações serviu de base à identificação de um comportamento-padrão dos circuitos em função das estratégias adoptadas, conduzindo à elaboração de uma proposta de métrica que, em face de qualquer implementação de um circuito, permite determinar a estratégia de mais baixo custo, em termos de degradação do seu funcionamento.

A grande vantagem da métrica proposta é a sua fácil integração com as ferramentas de síntese, distribuição e encaminhamento.



## **7. APLICAÇÃO DO TESTE**



Neste capítulo, é exposta a última componente da trilogia que integra a metodologia de teste proposta para as FPGAs parcial e dinamicamente reconfiguráveis, a implementação dos procedimentos de teste propriamente ditos.

O teste dos dispositivos lógicos programáveis assume-se como uma súmula de todos os aspectos ligados ao teste, ao integrar num único componente recursos lógicos, de encaminhamento, de interligação e ainda uma memória com funções de configuração de toda a estrutura. A abordagem natural é a divisão do procedimento de teste segundo estas várias partes, tendo em atenção as especificidades de cada uma.

A opção pela geração determinística dos vectores de teste, ao invés do recurso à configuração de estruturas de auto-teste interno que integram lógica para esse fim, e a sua aplicação por meio da infra-estrutura de teste por varrimento periférico (IEEE 1149.1 *Boundary Scan*), usada igualmente para a recolha das respostas, no sentido de evitar o dispêndio extra de recursos, caracterizam esta proposta.

A flexibilidade da FPGA cria uma dependência entre a detecção de uma falta e a configuração do circuito, pelo que, faltas indetectáveis com uma dada configuração de teste, podem com outra configuração tornar-se detectáveis. Contudo, cada reconfiguração de teste implica um custo acrescido em termos de memória e tempo de teste, impondo-se, por isso, a minimização do número de configurações de teste e do correspondente conjunto de vectores de teste que permita cobrir todas as faltas estruturais da FPGA, segundo o modelo de faltas considerado para cada uma das partes.

O desconhecimento, a nível físico, da implementação dos vários componentes do bloco lógico conduz à adopção de um modelo de faltas híbrido, que conjuga aspectos estruturais com aspectos funcionais. Para as estruturas de encaminhamento e de interligação, os modelos de faltas adoptados reflectem as opções clássicas para este tipo de recursos.

Por último, é analisada a questão do teste da memória de configuração, que, indissociável da sua função de configuração, levanta particulares problemas.



## 7.1. DESCRIÇÃO DA ESTRATÉGIA DE TESTE

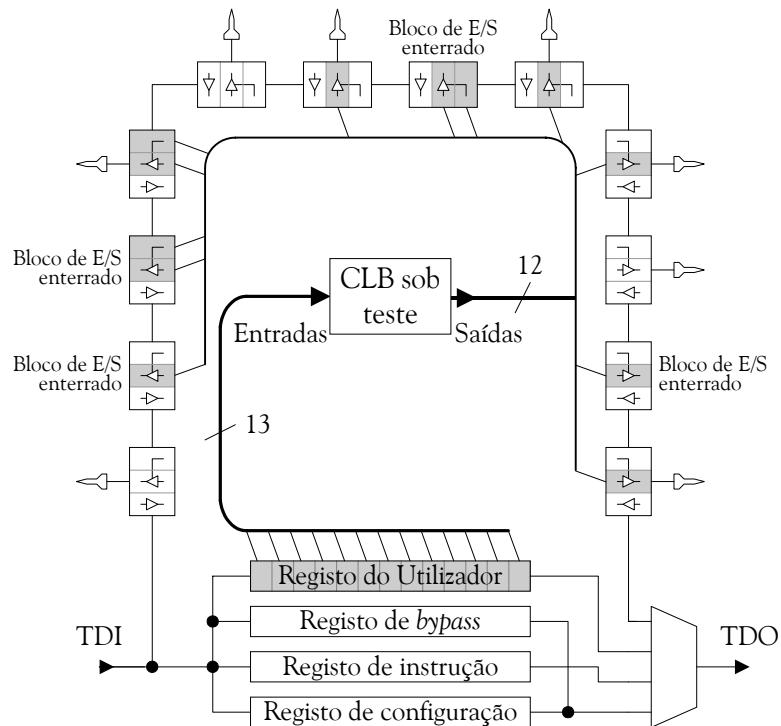
Tendo já sido definido como objectivo o teste estrutural da FPGA, sem afectar a sua operação normal, e descritos os mecanismos conducentes à libertação (para a submissão a esse teste) dos recursos ocupados pelas funções do sistema, sem interromper ou alterar a sua funcionalidade, resta, neste capítulo, observar como, depois de libertos, esses recursos são testados.

No sentido de minimizar os recursos extra necessários para o teste, bem como o seu custo em termos de tempo de aplicação, optou-se pela geração e aplicação de vectores de teste, em vez de se recorrer a uma estratégia de auto-teste interno, por esta conduzir não só à ocupação de recursos internos da FPGA (embora tal pudesse ser, dependendo da estratégia global de teste, apenas uma ocupação temporária), mas também a um maior número de vectores de teste e, consequentemente, a uma maior duração do próprio teste, resultado do número substancialmente mais elevado de vectores a aplicar. Uma vez que, sendo a FPGA uma estrutura regular e tendo definido como unidade de teste da lógica o CLB, o seu reduzido tamanho propicia e incentiva mesmo aquele tipo de abordagem.

A aplicação dos vectores de teste e a recolha das respostas efectua-se através da infra-estrutura BS, pré-existente na FPGA e definida na norma IEEE 1149.1 [IEEE 1149.1, 01], com as entradas do CLB ou os recursos de encaminhamento sob teste a receberem os estímulos a partir de um registo de teste, denominado Registo do Utilizador, e as respostas a serem encaminhadas para células do registo BS associadas aos blocos de E/S, como se exemplifica na Figura 7.1, para o caso do teste de um bloco lógico [Gericota et al., 01]. Uma vez que se trata de estruturas reconfiguráveis, o teste estrutural de cada CLB e dos recursos de encaminhamento exige o uso de várias configurações de teste, podendo essa configuração ser igualmente efectuada através da mesma interface com a infra-estrutura BS, tal como ocorria no caso da relocação da funcionalidade dos CLBs e no reencaminhamento de sinais, possibilidade normalizada pela norma IEEE 1532 [IEEE 1532, 01]. Desta forma, toda a metodologia proposta utiliza como única interface entre o componente e o exterior a infra-estrutura BS e o seu porto de acesso, pelo que não são requeridas alterações a nível da carta de circuito impresso.

O comprimento do Registo do Utilizador, controlado igualmente através da infra-estrutura BS, é imposto pelo número de entradas de uma *slice*, treze, às quais é possível aplicar estímulos de teste. Mesmo admitindo a hipótese do teste simultâneo de mais do que um bloco lógico, o comprimento do Registo do Utilizador não necessita de ser alterado, uma vez que todas as *slices* de todos os CLBs sob teste podem receber em paralelo os mesmos vectores. Aliás, é já isso que se passa com as duas

slices de cada CLB, a cujas entradas são aplicados simultaneamente os mesmos vectores. Pelo contrário, as respostas aos vectores, recolhidas nas suas saídas, devem ser adquiridas independentemente, para permitir efectuar o diagnóstico de uma possível falta por simples análise dessas respostas, sem necessidade de recurso a métodos de diagnóstico adaptativos [Stroud et al., 96a] e [Wang et al., 99]. O comprimento do Registo do Utilizador poderia eventualmente ser aumentado para possibilitar o teste em simultâneo de um maior número de recursos de encaminhamento. No entanto, o aumento do seu comprimento implica igualmente o aumento dos recursos ocupados para a sua implementação, pelo que a viabilidade desse aumento deve ser estudada caso a caso, consoante o índice de ocupação da FPGA.



**Figura 7.1:** Infra-estrutura para o teste de um bloco lógico

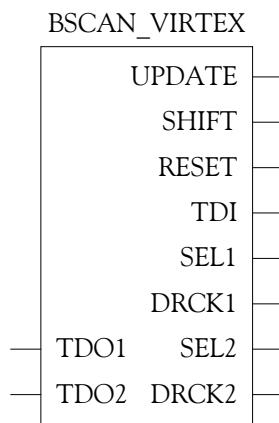
### 7.1.1 APLICAÇÃO DE ESTÍMULOS

A necessidade de emprego de um registo extra, definido pelo utilizador, para aplicação dos estímulos, deve-se à impossibilidade de utilizar o registo BS para esse fim sem afectar os valores presentes em cada entrada da FPGA. A norma IEEE 1149.1 define uma instrução opcional denominada INTEST que permite colocar o registo BS entre a entrada TDI e a saída TDO do componente, para permitir a deslocação de valores para o interior desta cadeia e, posteriormente, a sua aplicação em paralelo à lógica interna. Contudo, e conforme determinado na norma, todos os portos de saída passam a ter o seu estado definido pelo andar de retenção das células BS e todos os sinais de entrada da lógica funcional são definidos pelos andares de retenção dos pinos de entrada

(ver Figura 2.32), pelo que a lógica implementada dentro da FPGA fica isolada do exterior pela própria cadeia BS, bloqueando o funcionamento do sistema.

A família Virtex da Xilinx [Xilinx, 02a] permite a definição pelo utilizador de um máximo de dois registos de dados adicionais (USER1 e USER2), que são válidos apenas após a configuração. Estes registos, uma vez criados, podem ser acedidos através dos pinos do TAP, da mesma forma que os restantes registos de dados, seleccionando as instruções opcionais USER1 e USER2.

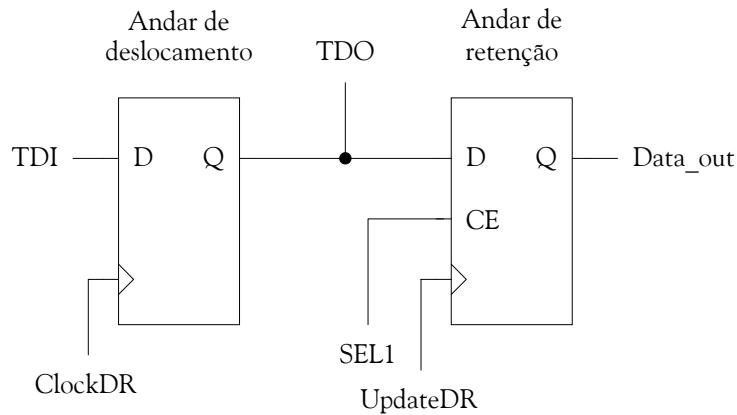
Dado que os pinos e a lógica de teste associados à infra-estrutura BS são dedicados, não é necessária a adição ao projecto de qualquer elemento para possibilitar o seu uso. Contudo, ao contrário do que se passa com a simples utilização da infra-estrutura BS, a definição e uso de registos próprios obriga à sua declaração explícita no projecto através do uso de uma *macro* para esse fim, cujo símbolo se encontra representado na Figura 7.2. Essa *macro* possui dois pinos, SEL1 e SEL2, que determinam qual o registo a aceder, dois pinos de relógio independentes para cada registo, DRCK1 e DRCK2, e pinos de saída partilhados que representam o estado do controlador do TAP, RESET, SHIFT e UPDATE, para além de um pino de saída TDI e dois pinos de entrada, TDO1 e TDO2.



**Figura 7.2:** Representação simbólica da *macro* BSCAN\_VIRTEX

Os sinais na entrada TDO1 são passados para o pino TDO externo do componente quando a instrução USER1 está activa, caso em que o sinal na saída SEL1 passa a nível alto. A saída DRCK1 permite que o Registo do Utilizador seleccionado por USER1 aceda ao relógio do registo de dados (*Clock Data Register – ClockDR*) gerado pelo controlador do TAP. TDO2 e SEL2 realizam funções similares quando a instrução USER2 se encontra activa. Os pinos RESET, UPDATE (*UpdateDR*) e SHIFT (*ShiftDR*) descodificam o correspondente estado da máquina de estados do controlador interno. O pino TDI proporciona acesso ao sinal TDI externo do porto de acesso, para deslocamento dos dados para os registos do utilizador. A criação de registos definidos pelo utilizador enquadrar-se dentro das permissões contempladas no ponto 9.1.1 da norma IEEE 1149.1 [IEEE 1149.1, 01].

A célula-base do Registo do Utilizador é constituída por dois *flip-flops* do tipo D, correspondendo um ao andar de deslocamento e o outro ao andar de retenção. Na Figura 7.3, encontra-se a representação esquemática desta célula.



**Figura 7.3:** Representação esquemática da célula-base do Registo do Utilizador

Cada um dos vectores a aplicar é deslocado em série, entrando por TDI e saindo por TDO. Quando termina o deslocamento, um impulso em UpdateDR leva as saídas paralelas a assumir os valores presentes no primeiro *flip-flop* e, consequentemente, a aplicar o vector à lógica a testar. Na Figura 7.4, encontra-se a descrição em VHDL do Registo do Utilizador, apresentando-se na Tabela 7.1 um resumo do espaço ocupado por este registo numa XCV200.

O Registo do Utilizador pode ser adicionado a qualquer projecto, incluindo o conjunto de restrições à disposição para uma ocupação optimizada dos recursos. Um exemplo da implementação física do registo numa XCV200 é mostrado na Figura 7.5.

**Tabela 7.1:** Resumo do espaço ocupado pela implementação do Registo do Utilizador numa XCV200

Device utilization summary:		
Number of errors:	0	
Number of 4 input LUTs:	0 out of 4704	0%
Number of SLICEs:	14 out of 2352	1%
Number of Slice Flip Flops:	26 out of 4704	1%
Number of BSCANs:	1 out of 1	100%
Total equivalent gate count for design: 256		

```

-- *** Celula-base do registo do utilizador ***

library IEEE;
use IEEE.std_logic_1164.all;

entity BS_Cell is
    port (Sin, Clk, Update, Enable: in STD_LOGIC;
          Sout, Dout: out STD_LOGIC
        );
end BS_Cell;

architecture BS_Cell_arch of BS_Cell is
begin
    DESLOCAMENTO:process (Clk)
    begin
        if (Clk'event and Clk='1') then
            q1 <= Sin;
        end if;
    end process;

    RETENCAO: process (Enable, Update)
    begin
        if Enable='1' then
            if (Update='0' and Update'event) then
                q2 <= q1;
            end if;
        end if;
    end process;

    Sout <= q1;
    Dout <= q2;
end BS_Cell_arch;

-- *** Criacao de um registo do utilizador com 13 celulas ***

library IEEE;
use IEEE.std_logic_1164.all;

entity UserReg is
    port (
        Scan_in, ClkVT, UpdateVT, EnableVT: in STD_LOGIC;
        Scan_out: out STD_LOGIC;
        Data_out: out STD_LOGIC_VECTOR(12 downto 0)
    );
End UserReg;

```

**Figura 7.4:** Descrição em VHDL do Registo do Utilizador

```

architecture UserReg_arch of UserReg is

component BS_Cell
port (Sin, Clk, Update, Enable: in STD_LOGIC;
      Sout, Dout: out STD_LOGIC
    );
end component;

signal S: STD_LOGIC_VECTOR(13 downto 0);
signal C, U, E: STD_LOGIC;

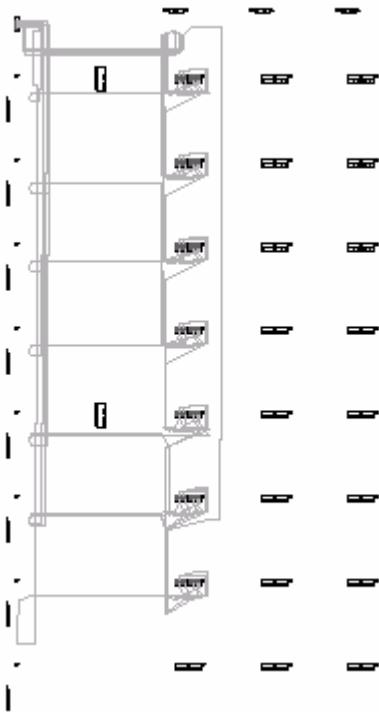
begin
S(0) <= Scan_in;
Scan_out <= S(13);
C <= ClkVT;
U <= UpdateVT;
E <= EnableVT;

GK: for K in 0 to 12 generate
  FG: BS_Cell port map (Sin=>S(K), Clk=>C, Update=>U,
                        Enable=>E, Sout=>S(K+1), Dout=>Data_out(K));
end generate GK;

end UserReg_arch;

```

**Figura 7.4:** Descrição em VHDL do Registo do Utilizador (cont.)



**Figura 7.5:** Implementação física do Registo do Utilizador

### 7.1.2 CAPTURA DAS RESPOSTAS

Nas FPGAs da família Virtex, todos os blocos de E/S (enterrados e não enterrados) são considerados como pinos bidireccionais independentes com terceiro estado, pelo que cada um contribui com três células para o registo BS, independentemente da sua posterior configuração individual. Tal acontece porque a norma IEEE 1149.1 [IEEE 1149.1, 01] determina, no seu ponto 9.2.1-d), que, nos componentes programáveis, o comprimento do registo BS seja independente da forma como o componente é programado. Na prática, dependendo dessa configuração, muitas das células desse registo tornam-se redundantes, facto que está de acordo com o especificado no ponto 11.8 da mesma norma. Por exemplo, um pino configurado como entrada necessita apenas de uma única célula BS. Contudo, as duas restantes não são removidas da cadeia, podendo, por isso, ser aproveitadas para capturar as respostas aos vectores de teste aplicados ao CLB sob teste ou ao conjunto de linhas sob teste e, como tal, evitar a ocupação de recursos extra para este propósito. A observação dessas respostas é efectuada através do uso da instrução SAMPLE que permite capturar o estado das entradas e saídas do componente sem interferir com a sua operação normal.

O pequeno número de células do Registo do Utilizador possibilita um rápido deslocamento e aplicação dos vectores de teste. No entanto, o tempo de deslocamento das respostas capturadas no registo BS depende da posição dentro da cadeia das células usadas para a sua captura. Independentemente do comprimento desta, o comprimento do vector a deslocar para o exterior correspondente à distância entre TDO e a célula que contém o último bit da resposta. No pior caso, aquele em que a célula mais próxima de TDI é usada para a captura de um bit da resposta, o comprimento do vector a deslocar para se recolher todas as respostas é igual ao comprimento do próprio registo BS.

## 7.2. TESTE DOS BLOCOS LÓGICOS CONFIGURÁVEIS

Definida a forma de aplicação e de recolha dos vectores de teste, é necessário efectuar o estudo dos blocos lógicos configuráveis, com vista a determinar as várias etapas do seu teste estrutural, tendo em conta a sua configurabilidade. Para o estudo deste problema, é importante esclarecer alguns termos que serão usados ao longo da sua abordagem:

- configuração de teste ( $T_C$ ): uma configuração particular do CLB que tem como objectivo facilitar o teste da sua estrutura<sup>1</sup>;

---

<sup>1</sup> Por razões de simplificação da escrita, e desde que não haja lugar a equívocos, será usado o termo *configuração* com o significado de *configuração de teste*.

- reconfiguração de teste: procedimento de programação num CLB de uma dada configuração de teste;
- aplicação do teste ( $T_A$ ): procedimento de aplicação de um vector a um CLB configurado com uma dada configuração de teste e captura das respostas;
- fase de teste: conjunto constituído por uma configuração de teste e por um certo número de aplicações de teste;
- sessão de teste ( $T_S$ ): conjunto de um dado número de fases de teste que permitem testar a totalidade do CLB, sob o modelo de faltas considerado;
- interligações de teste: conjunto de interligações necessárias para a interligação do CLB com o Registo do Utilizador e as células BS atribuídas à captura das respostas.

O teste dos dispositivos lógicos programáveis coloca vários desafios, visto que requer a implementação de várias configurações de teste. Devido à flexibilidade da FPGA, a detecção de uma falta depende da configuração do circuito. De facto, uma falta pode ser indetectável numa dada configuração de teste, mas tornar-se detectável noutra configuração. Contudo, cada reconfiguração implica um custo acrescido em termos de memória e tempo de teste, pelo que se coloca a questão de determinar o número mínimo de configurações de teste e a correspondente sequência de aplicações de teste, que permitam cobrir todas as faltas estruturais da FPGA, segundo o modelo de faltas considerado.

Uma sessão de teste implica, além da fase inicial de estabelecimento das interligações de teste, o uso de um conjunto de configurações, englobando cada várias aplicações de teste, e que se pode resumir na Equação 7.1.

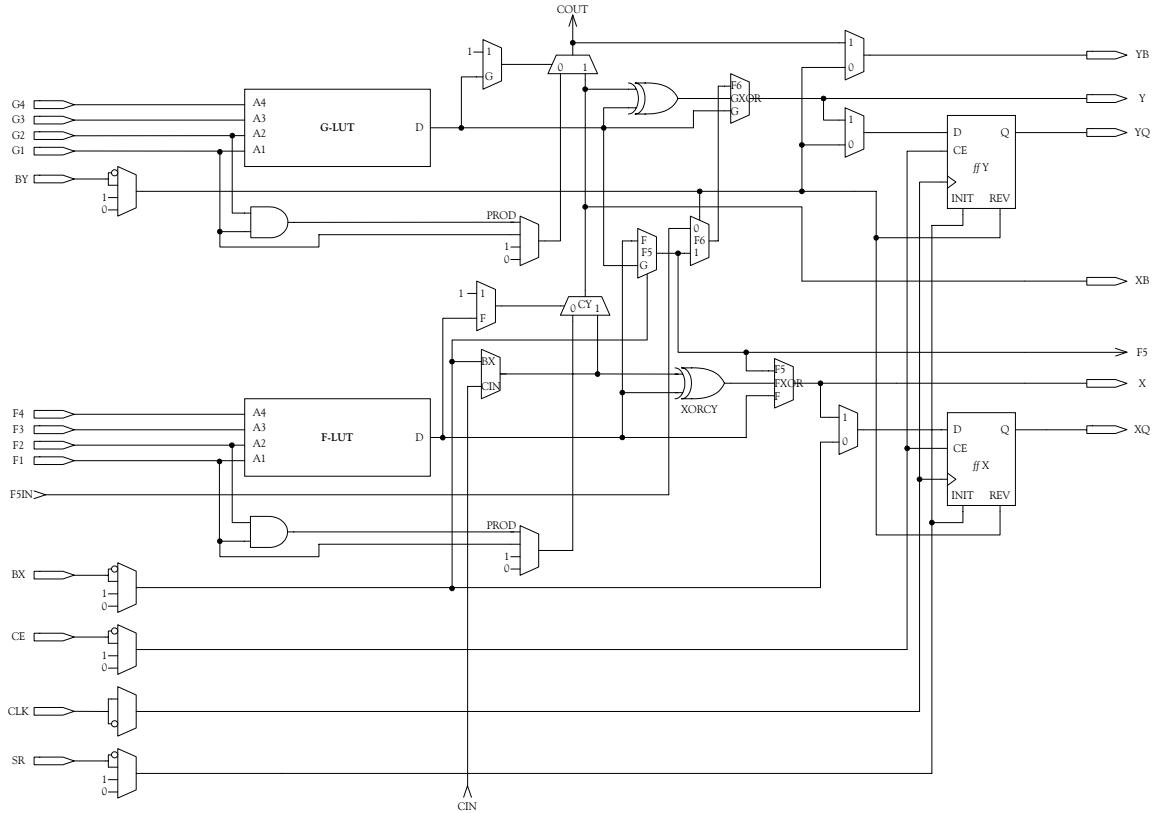
**Equação 7.1**

$$T_S = (T_{C_1}, (T_{A_{11}}, T_{A_{12}}, \dots)) + (T_{C_2}, (T_{A_{21}}, T_{A_{22}}, \dots)) + \dots + (T_{C_n}, (T_{A_{n1}}, T_{A_{n2}}, \dots))$$

É, pois, imperioso estudar a estrutura e as possibilidades de configuração do bloco lógico, de modo a determinar o número mínimo de configurações de teste necessárias em cada sessão, bem como o número mínimo de aplicações de teste, que minimizem o tempo total da sessão. A geração dos vectores é efectuada tendo em conta estes dois pontos e procura alcançar a máxima cobertura de faltas, segundo o modelo híbrido já referido e amplamente aplicado por outros autores [Fawcett, 94], [Huang et al., 96], [Huang et al., 96a], [Wang et al., 97], [Huang et al., 98], [Metra et al., 98], [Mitra et al., 98], [Renovell, 98] e [Renovell et al., 98a].

### 7.2.1 ESTRUTURA DOS BLOCOS LÓGICOS CONFIGURÁVEIS

Cada bloco lógico é constituído por duas *slices* independentes e exactamente iguais<sup>2</sup>. O modelo de teste, representando apenas uma das *slices* e tendo já em conta as restrições impostas pela impossibilidade de replicar as tabelas de consulta que operem em modo memória, encontra-se ilustrado na Figura 7.6.



**Figura 7.6:** Modelo de teste de uma *slice* do CLB

Cada *slice* é constituída por um conjunto de diferentes primitivas funcionais que, à luz do modelo de faltas considerado, podem ser analisadas separadamente:

- multiplexadores;
- tabelas de consulta;
- elementos de retenção;
- portas lógicas simples ('E' e 'OU-Exclusivo').

Os multiplexadores e as tabelas de consulta são tipicamente estruturas configuráveis, enquanto os elementos de retenção e as portas lógicas são não configuráveis. Contudo, os elementos de

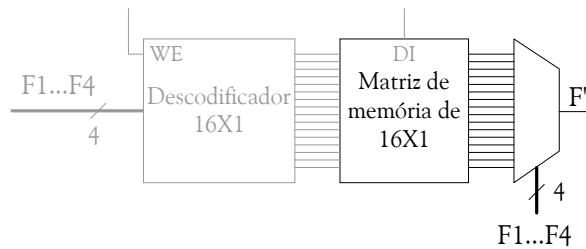
---

<sup>2</sup> Ver ponto 2.4.1.

retenção, no caso das Virtex, possuem dois bits de configuração: um dedicado à selecção entre o modo síncrono e assíncrono de inicialização e o outro à selecção do modo de funcionamento, *flip-flop* ou *latch*.

Os multiplexadores sem linhas de selecção presentes no bloco lógico são programáveis, sendo a selecção da entrada efectuada pelo valor presente nas células respectivas da memória de configuração.

A estrutura interna das tabelas de consulta é, de certa forma, similar à dos multiplexadores, como se observa na Figura 7.7. De facto, não considerando a estrutura que permite transformar as tabelas de consulta em memórias distribuídas, elas resumem-se a um conjunto de células de memória ligadas às entradas de um multiplexador.



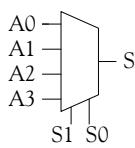
**Figura 7.7:** Estrutura interna de uma tabela de consulta

A geração de um procedimento de teste para as diferentes primitivas funcionais, tendo em conta o modelo considerado, é analisada separadamente.

### 7.2.2 TESTE DOS BLOCOS ELEMENTARES

Para o teste estrutural dos multiplexadores, é usado um modelo funcional, no qual são assumidas faltas do tipo sempre-a nas entradas e saída da primitiva, incluindo as entradas configuráveis nos multiplexadores programáveis, uma vez que a estrutura da sua lógica interna é desconhecida. A validação desta abordagem, necessária porque esta aproximação não garante *a priori* a detecção de faltas no interior da primitiva por aplicação de um conjunto de vectores de teste gerados com base no modelo, é apresentada em [Portal, 99]. Um conjunto de vectores de teste, gerados para detectar 100% das faltas sempre-a-0 e sempre-a-1 nas entradas e na saída do multiplexador, foi aplicado a três tipos de implementação diferentes (implementação lógica em dois planos, implementação lógica em árvore e implementação eléctrica em árvore), tendo demonstrado a eficácia deste procedimento na detecção de faltas estruturais no interior da primitiva. Este resultado é bastante importante, pois possibilita que, quer os multiplexadores programáveis, quer as tabelas de consulta, sejam testados com base nesta abordagem.

A sequência de teste ilustrada na Figura 7.8, gerada para testar as faltas do tipo sempre-a nas entradas e saídas do multiplexador ilustrado na mesma figura, pode, do mesmo modo, ser usada para o teste dos multiplexadores programáveis presentes no bloco lógico. Nestes multiplexadores, as entradas de selecção são substituídas por células de memória que fazem parte da própria memória de configuração do circuito, pelo que a aplicação imediata daquele conjunto de vectores, seguindo a mesma sequência, obriga ao uso de oito configurações de teste diferentes. É evidente que esta solução não se enquadra na obrigatoriedade de minimizar o número de configurações de teste, tanto mais que existe uma repetição dos valores presentes nas células de memória de selecção, resultando no aparecimento de cada configuração duas vezes.



Vectores de teste						Faltas detectadas	S
A3	A2	A1	A0	S1	S0		
X	1	1	0	0	0	A0 sempre-a-1, S0 sempre-a-1, S1 sempre-a-1	0
0	1	X	0	1	0	A2 sempre-a-0, S0 sempre-a-1, S1 sempre-a-0	1
1	X	0	1	0	1	A1 sempre-a-1, S0 sempre-a-0, S1 sempre-a-1	0
1	0	0	X	1	1	A3 sempre-a-0, S0 sempre-a-0, S1 sempre-a-0	1
1	0	X	1	1	0	A2 sempre-a-1, S0 sempre-a-1, S1 sempre-a-0	0
X	0	0	1	0	0	A0 sempre-a-0, S0 sempre-a-1, S1 sempre-a-1	1
0	1	1	X	1	1	A3 sempre-a-1, S0 sempre-a-0, S1 sempre-a-0	0
0	X	1	0	0	1	A1 sempre-a-0, S0 sempre-a-0, S1 sempre-a-1	1

**Figura 7.8:** Sequência de teste para um multiplexador

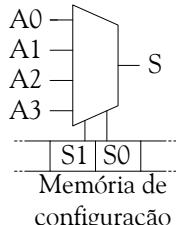
O objectivo de minimização do número de configurações obriga a uma alteração da sequência de aplicação dos vectores, que conduz ao agrupamento dos vectores que apresentam os mesmos valores nas linhas de selecção, como se ilustra na Figura 7.9.

O multiplexador é primeiramente configurado com a configuração  $T_{C1}=\{0, 0\}$ <sup>3</sup>, sendo-lhe, depois aplicada uma sequência de teste constituída por dois vectores  $T_{A1}=\{X110, X001\}$ . Este procedimento é repetido para as configurações  $T_{C2}=\{1, 0\}$ ,  $T_{C3}=\{0, 1\}$  e  $T_{C4}=\{1, 1\}$  e respectivos  $T_{A2}$ ,  $T_{A3}$  e  $T_{A4}$ .

Formalmente, os conjuntos  $(T_{Cn}, T_{An})$  representam os pares configuração-vectores para um procedimento optimizado de teste para o caso dos multiplexadores programáveis. A generalização para um multiplexador programável de  $n$  bits de selecção (células de memória de selecção) é

<sup>3</sup>  $T_{Cn}=\{S1, S0\}$

imediata, existindo nessa situação  $2^n$  conjuntos de pares configuração-vectores de teste. É interessante notar que qualquer que seja o número de bits de endereçamento,  $\#T_{An}=2$ .



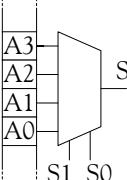
Vectores de teste						S
A3	A2	A1	A0	S1	S0	
X	1	1	0	0	0	0
X	0	0	1	0	0	1
0	1	X	0	1	0	1
1	0	X	1	1	0	0
1	X	0	1	0	1	0
0	X	1	0	0	1	1
1	0	0	X	1	1	1
0	1	1	X	1	1	0

Figura 7.9: Sequência de teste para um multiplexador programável

A análise efectuada para o caso do multiplexador serve igualmente de base para o estudo das tabelas de consulta. Neste caso, são as entradas de dados que são ligadas a células da memória de configuração, ao invés das linhas de selecção, que agora são entradas primárias do bloco lógico. A aplicação da mesma sequência de vectores conduz a um total de oito configurações de teste:  $T_{C1}=\{X, 1, 1, 0\}$ ,  $T_{C2}=\{X, 0, 0, 1\}$ ,  $T_{C3}=\{0, 1, X, 0\}$ ,  $T_{C4}=\{1, 0, X, 1\}$ ,  $T_{C5}=\{1, X, 0, 1\}$ ,  $T_{C6}=\{0, X, 1, 0\}$ ,  $T_{C7}=\{1, 0, 0, X\}$  e  $T_{C8}=\{0, 1, 1, X\}$ .

Tal como no caso dos multiplexadores, é necessário definir um número mínimo de configurações de teste, isto é, de pares configuração-vectores. Uma vez que as oito configurações de teste apresentam valores indeterminados (X) para alguns dos bits, é utilizada uma técnica de compactação que, diminuindo o número de configurações, define esses valores. O resultado da compactação encontra-se representado na Figura 7.10.

Através de uma escolha criteriosa dos vectores a compactar, que tem em conta os valores indeterminados, consegue-se obter apenas duas configurações de teste e em consequência dois conjuntos de pares configuração-vectores, ( $T_{C1}=\{0, 1, 1, 0\}$ ,  $T_{A1}=\{00, 10, 11, 01\}$ ) e ( $T_{C2}=\{1, 0, 0, 1\}$ ,  $T_{A2}=\{00, 10, 11, 01\}$ ). O mesmo procedimento de compactação não é, porém, viável para o caso dos valores a aplicar nas linhas de selecção, uma vez que, como se observa na Figura 7.9, estes não possuem valores indeterminados.



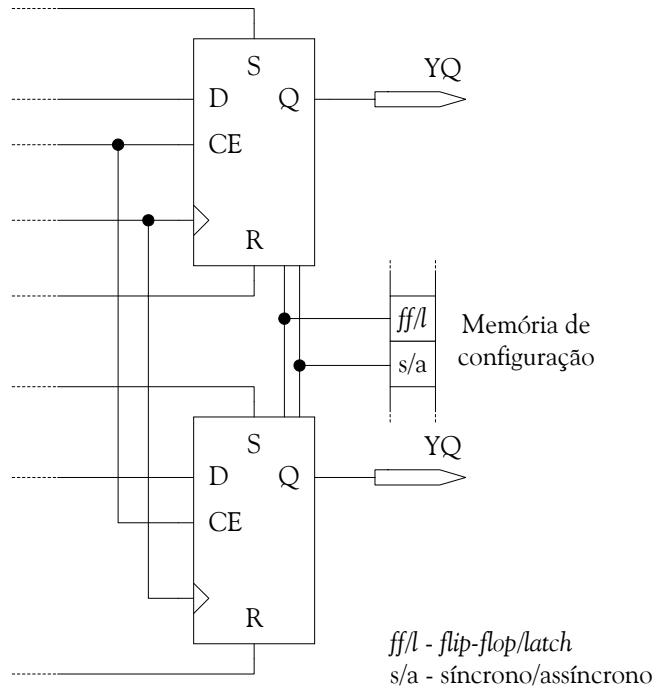
	A3	A2	A1	A0		A3	A2	A1	A0
	X	1	1	0		X	0	0	1
	0	1	X	0		1	0	X	1
	0	X	1	0		1	X	0	1
	0	1	1	X		1	0	0	X
$T_{C1} =$	0	1	1	0		$T_{C2} =$	1	0	0

**Figura 7.10:** Optimização das configurações de teste para uma tabela de consulta

As sequências de configuração  $T_{C1}=\{0, 1, 1, 0\}$  e  $T_{C2}=\{1, 0, 0, 1\}$  correspondem à programação na tabela de consulta de uma função ‘OU-Exclusivo’ e ‘Não-OU-Exclusivo’, respectivamente.

A generalização a uma tabela de consulta com  $n$  bits de selecção é, também neste caso, imediata, sendo evidente que, qualquer que seja o número de bits, somente duas configurações de teste, configuração das funções ‘OU-Exclusivo’ e ‘Não-OU-Exclusivo’ na tabela de consulta, são necessárias. Pelo contrário, o número de vectores de teste a aplicar associados a cada uma das configurações é  $2^n$ . Só assim é possível seleccionar, ao menos uma vez, cada uma das entradas da tabela. Outra característica é que, com aquelas configurações, a simples inversão de uma das entradas da tabela origina a inversão do valor lógico na sua saída.

O teste dos elementos de retenção presentes na família Virtex exige igualmente mais que uma configuração de teste, dado que o par existente em cada *slice* partilha duas células da memória de configuração, que controlam o modo de funcionamento do elemento: uma dedicada à selecção entre o modo síncrono e assíncrono de inicialização (s/a) e outra para selecção entre o funcionamento como *flip-flop* ou *latch* (*ff/l*), como se ilustra na Figura 7.11. No entanto, uma vez que o número de células de configuração associadas é apenas de duas, o número mínimo de configurações de teste, para que seja possível testar o elemento de retenção em todos os modos de funcionamento, é somente quatro. Uma particularidade é que, embora por manipulação directa dos ficheiros de configuração seja teoricamente possível configurar os elementos de retenção como *latches* e seleccionar simultaneamente o modo síncrono de inicialização, este tipo de funcionalidade não faz sentido, uma vez que as *latches* não têm, por natureza, sinal de sincronismo. Os elementos de retenção são assim testados considerando um modelo de faltas funcional, que usa apenas três configurações de teste.



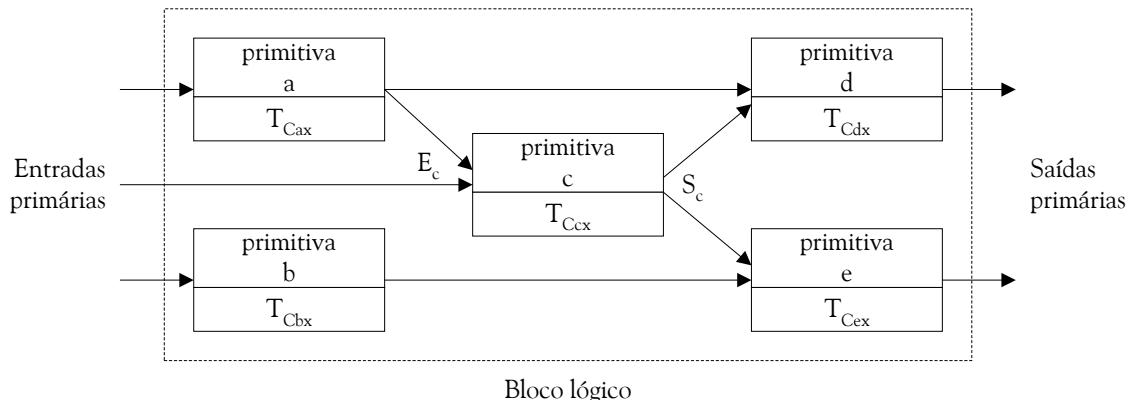
**Figura 7.11:** Estrutura dos elementos de retenção presentes na *slice*

### 7.2.3 TESTE DE UM BLOCO LÓGICO

Da secção anterior conclui-se que o número final de configurações a usar para o teste da totalidade do bloco lógico depende fortemente do número de configurações necessárias para testar cada uma das primitivas. A minimização do número de configurações para cada primitiva individual constituinte do bloco lógico resulta na minimização do número de configurações do próprio bloco.

#### 7.2.3.1 DETERMINAÇÃO DO NÚMERO DE CONFIGURAÇÕES DE TESTE

Um bloco lógico configurável pode ser representado como um conjunto de primitivas configuráveis interligadas entre si, como se exemplifica na Figura 7.12.



**Figura 7.12:** Exemplo de uma possível interligação de primitivas num bloco lógico

A configuração de uma configuração de teste na *primitiva c* ( $T_{Ccx}$ ) não levanta qualquer problema, uma vez que o acesso à memória de configuração é garantido durante a fase de reconfiguração de teste. No entanto, a aplicação dos vectores a essa primitiva e a captura das respostas mostram-se mais problemáticas, pois algumas das suas entradas e a totalidade das suas saídas não são entradas ou saídas primárias do bloco lógico. Isso implica que os vectores de teste tenham de ser aplicados e as respostas recolhidas através dos outros módulos, impondo restrições à sua própria configuração. Para ser possível testar a *primitiva c* é, por isso, necessário reunir um determinado conjunto de condições de testabilidade:

1. todas as configurações de teste da *primitiva c* devem-lhe ser aplicadas;
2. a controlabilidade da *primitiva c* deve ser garantida pelas configurações das primitivas que lhe estão a montante;
3. a observabilidade da *primitiva c* deve ser garantida pelas configurações das primitivas que lhe estão a jusante.

O conjunto de configurações das primitivas que cumprem as condições enunciadas constitui uma configuração de teste do bloco lógico. Significa isto que, ao nível do bloco, o objectivo deve ser a aplicação dos conjuntos de configurações adequadas ao teste de cada primitiva, de modo a minimizar o número de configurações de teste do bloco. Para tal, é preciso conjugar as condições de controlabilidade e observabilidade impostas às primitivas a montante e a jusante com as suas próprias configurações, de modo a que, simultaneamente, várias primitivas reunam as suas próprias condições de teste e possam ser testadas com uma única configuração do bloco lógico.

Recuperando o estudo anteriormente efectuado dos multiplexadores programáveis, é fácil verificar que, qualquer que seja a sua configuração, a sua saída está sempre ligada a uma entrada e, como tal, é completamente controlável. Em contrapartida, qualquer configuração não permite mais do que a observação da entrada seleccionada. De modo semelhante, o estudo efectuado para as tabelas de consulta permite concluir que, com uma configuração de teste ‘OU-Exclusivo’ ou ‘Não-OU-Exclusivo’, a saída é completamente controlável a partir de qualquer entrada e que qualquer entrada é completamente observável a partir da saída. No caso dos elementos de retenção presentes nos CLBs das Virtex, a sua saída é uma saída primária, pelo que a sua observabilidade está sempre garantida.

No que se refere à *slice* de teste representada na Figura 7.6, o número total de bits de configuração é de 26, possibilitando  $2^{26}$  configurações diferentes. Em [Renovell et al., 99] e [Renovell et al., 99a], é apresentado um método tabular para a resolução do problema da minimização das configurações de teste, a partir da análise das condições de testabilidade para cada uma das primitivas e tendo em

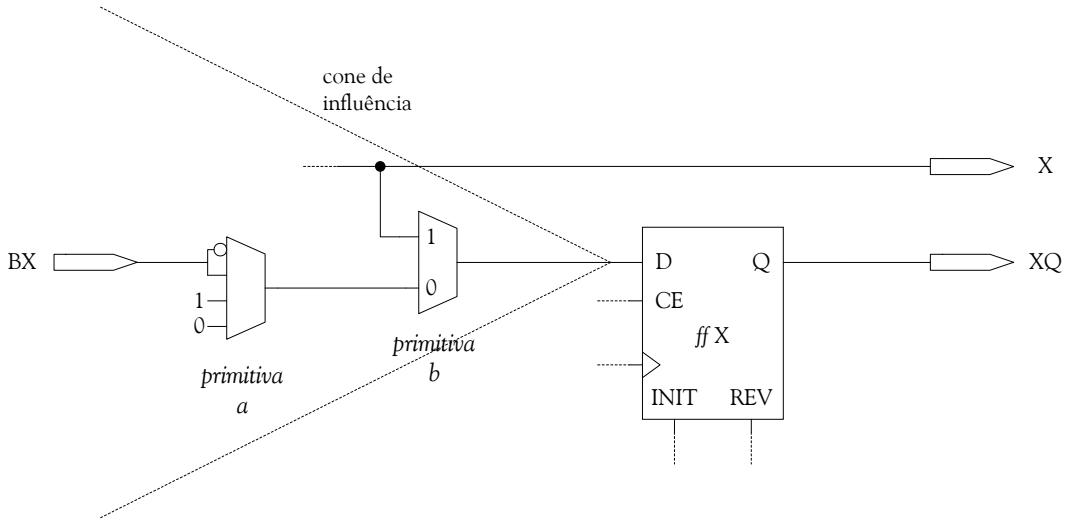
conta cada uma das possíveis configurações de teste do circuito. Contudo, a sua aplicação a este caso mostra-se inviável, dado o elevado número de configurações possíveis. Aliás, o carácter exponencial do aumento das configurações com o aumento do número de primitivas rapidamente inviabiliza o seu uso.

De resto, uma análise mais atenta do problema leva à rápida eliminação da maioria das possibilidades, uma vez que parte substancial das primitivas não são interdependentes entre si em termos de controlabilidade das entradas e observabilidade das saídas, pelo que é possível a segmentação do circuito em cones de influência, numa aplicação da técnica apresentada em [McCluskey, 84]. Entende-se como cone de influência de uma primitiva todas as entradas primárias ou saídas de outras primitivas a montante, que influenciam a saída da primitiva considerada. O valor presente na saída de uma primitiva pode ser controlado e observado numa saída primária através da aplicação de um determinado vector no seu cone de influência e de uma certa configuração de teste nas primitivas que lhe estão a montante e a jusante. É exactamente a última parte que interessa para esta discussão.

Dos elementos configuráveis da *slice*, aqueles que necessitam de um maior número de configurações de teste são os multiplexadores de 4:1, com quatro configurações diferentes, pelo que, logo à partida, se pode afirmar que o número mínimo de configurações será igual ou superior a quatro, uma vez que a primeira condição imposta para uma primitiva ser testada é a aplicação da totalidade das suas configurações. Se as entradas e saídas de cada um dos multiplexadores de 4:1 fossem entradas e saídas primárias do circuito, o número mínimo de configurações de teste seria quatro, dado que o cone de influência da saída dessa primitiva ficava restrito a ela própria. O número mantém-se nos casos em que as entradas são antecedidas por primitivas não configuráveis ou quando as saídas forem precedidas pelo mesmo tipo de primitivas.

Tomando em consideração a *slice* de teste, verifica-se que todas as entradas do multiplexador presente na entrada SR, são entradas primárias e que a sua saída, que controla o sinal de inicialização dos elementos de retenção, é perfeitamente observável quando este elemento está programado como *latch* ou como *flip-flop* (em modo de inicialização assíncrono ou síncrono). A sua observação torna-se, assim, independente da configuração da primitiva que lhe está a jusante. O mesmo não se passa se se considerar o multiplexador da entrada BX e se se pretender tornar a sua saída observável através do elemento de retenção, pois entre essa saída e a entrada do elemento de retenção existe uma outra primitiva, um multiplexador de duas entradas, como se observa na Figura 7.13. Seguindo as condições enunciadas anteriormente para o teste de uma primitiva, sabe-se que o teste da *primitiva a* implica, para além das suas quatro configurações de teste, a configuração da *primitiva b*, de tal modo que a saída da *primitiva a* atinja a entrada do elemento de

retenção. Por seu lado, duas das configurações de teste da *primitiva a* permitem a controlabilidade de uma das entradas da *primitiva b* com uma das suas próprias configurações de teste, pelo que esta pode ser testada em simultâneo com a primeira. No entanto, a outra configuração de teste da *primitiva b* é incompatível com as configurações de teste da *primitiva a*, pois torna a sua saída inobservável. Esta situação exige então um número mínimo de configurações de teste igual a cinco. De facto, o cone de influência da saída da *primitiva b* abrange ambas as primitivas, o que as torna interdependentes.

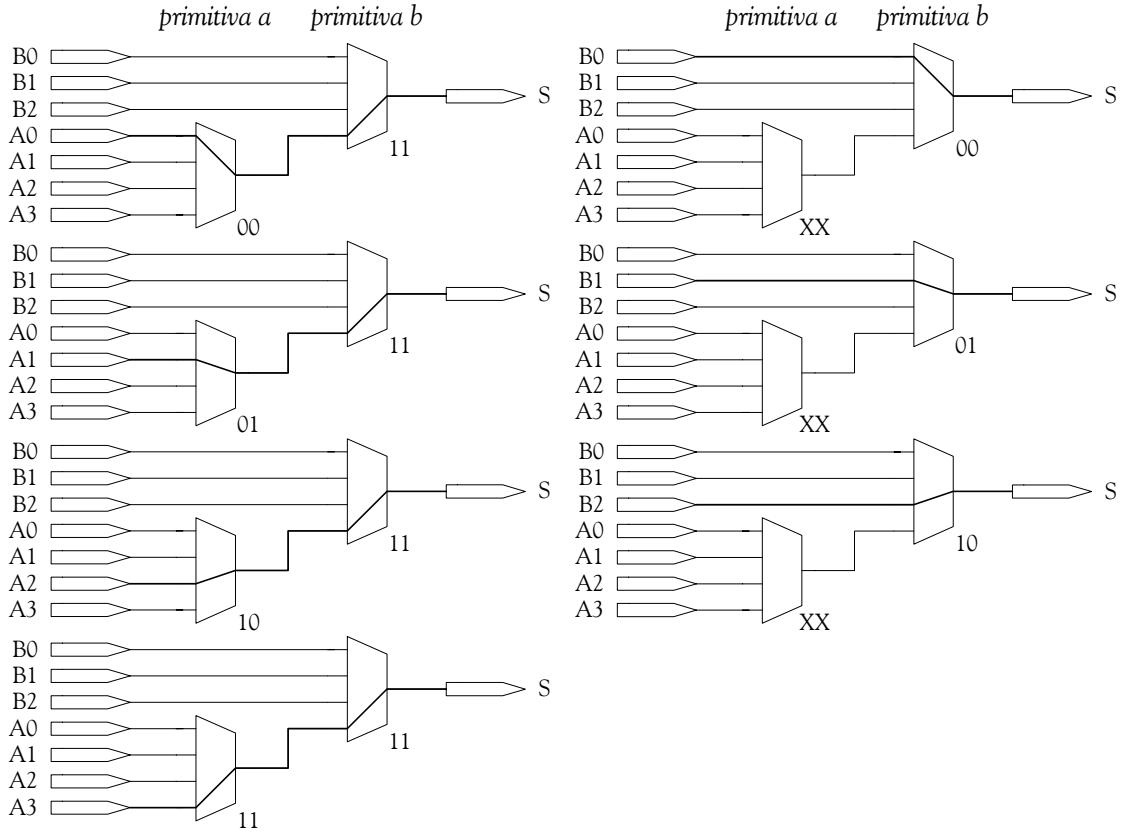


**Figura 7.13:** Exemplo da segmentação do circuito por cones de influência

De notar que, se a *primitiva b* fosse um multiplexador de 4:1, com as restantes entradas como entradas primárias do circuito, o número mínimo de configurações de teste do conjunto subiria para sete, com as quatro configurações de teste da *primitiva a* a serem adicionadas às três restantes da *primitiva b*, que não podem ser testadas simultaneamente com as da *primitiva a* por tornarem a sua saída inobservável. A totalidade das configurações de teste e respectivos percursos lógicos de observação e controlo das entradas e saídas das primitivas, para este conjunto, encontra-se ilustrada na Figura 7.14.

Desta observação, conclui-se que o aumento do número de primitivas configuráveis, associado ao aumento do número mínimo de configurações de teste de cada uma, no cone de influência de cada uma das saídas, conduz ao aumento do número mínimo de configurações de teste. Por conseguinte, é lícito enunciar o princípio de que:

- a saída primária, cujo cone de influência apresente o maior número de primitivas (configuráveis) com maior número mínimo de configurações de teste cada, determinará o número mínimo de configurações de teste do circuito, independentemente do número total de primitivas.



**Figura 7.14:** Configurações de teste dentro do cone de influência

A observação da Figura 7.6, permite ilustrar que a pior situação se verifica na saída YB, cujo cone de influência apresenta uma cascata de três multiplexadores, representados na Figura 7.15. Neste subcírcuito, a *primitiva c*, situada no cone da saída YB, não é configurável. Contudo, a sua linha de seleção depende de uma primitiva configurável, a *primitiva b*, pelo que, indirectamente, contribui para o número mínimo de configurações de teste deste cone de influência. Atente-se ainda no facto de que a tabela de consulta representada na Figura 7.15 (G-LUT), apesar de ser uma primitiva configurável, não faz parte, com base no princípio enunciado anteriormente, do cone de influência da saída YB, para determinação do número mínimo de configurações de teste do circuito, ao contrário do que acontece com as primitivas configuráveis *a*, *b* e *d*. A consideração de um cone de influência compreendendo a tabela de consulta e as primitivas configuráveis *b* e *d* conduziria a um mesmo número de primitivas, mas a um somatório menor do número de configurações mínimas de teste de cada uma, visto que o número mínimo de configurações de teste da tabela é de apenas dois, inferior ao número mínimo de configurações de teste da *primitiva a*, que é de quatro. O mesmo se passa com as entradas *c*<sub>1</sub> e *d*<sub>0</sub>, respectivamente das primitivas *c* e *d*.

Uma vez que o número total de bits de configuração deste subcírcuito é agora quatro, é já viável a utilização do método tabular apresentado em [Renovell et al., 99] e [Renovell et al., 99a], para a obtenção do número mínimo de configurações de teste que satisfazam as condições de testabilidade

das três primitivas configuráveis nele representadas, tendo em atenção que o número possível de configurações é de  $2^4=16$ .

Começando pela primitiva mais a montante, a *primitiva a*, a primeira condição é o uso de todas as suas configurações mínimas de teste (quatro); a segunda condição, a da controlabilidade de todas as suas entradas dever ser garantida pelas configurações das primitivas que lhe estão a montante, condição satisfeita por não existirem primitivas configuráveis a montante, embora nem todas as suas entradas sejam entradas primárias; e a terceira condição, a da observabilidade da sua saída dever ser garantida pela configuração das primitivas *b* e *d*. A concretização destas condições para cada uma das entradas da *primitiva a* resulta nas equações lógicas:

**Equação 7.2**

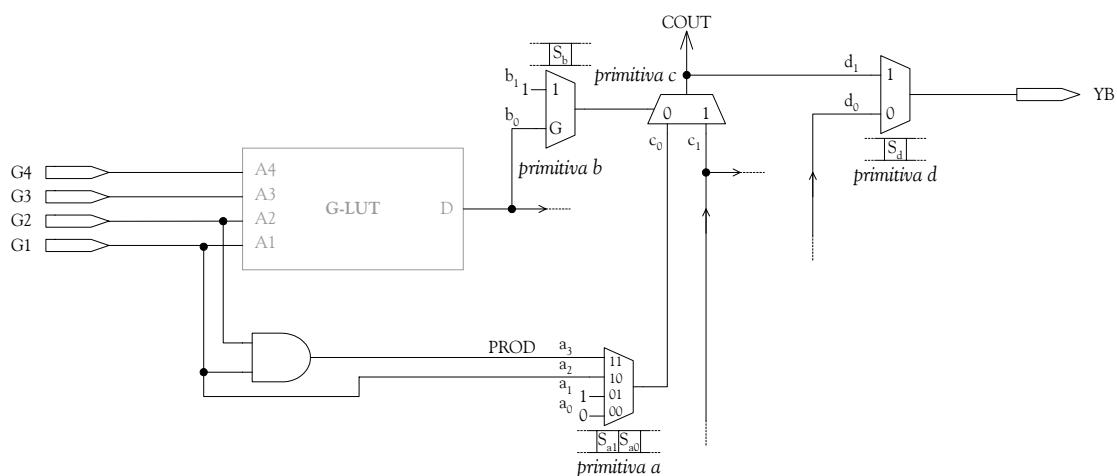
$$T_{a0} = \overline{S_{a1}} \cdot \overline{S_{a0}} \cdot \overline{S_b} \cdot S_d$$

$$T_{a1} = \overline{S_{a1}} \cdot S_{a0} \cdot \overline{S_b} \cdot S_d$$

$$T_{a2} = S_{a1} \cdot \overline{S_{a0}} \cdot \overline{S_b} \cdot S_d$$

$$T_{a3} = S_{a1} \cdot S_{a0} \cdot \overline{S_b} \cdot S_d$$

O valor de  $S_b=0$  em todas as expressões é o único que garante a controlabilidade da linha de seleção da *primitiva c*, permitindo que a saída da *primitiva a* seja encaminhada para a *primitiva d* e, consequentemente, para uma saída primária, garantindo a sua observabilidade. Repare-se, contudo, que, para que essa situação aconteça, é necessário que  $b_0=0$ . Todavia, trata-se de uma condição sempre obtinível, independentemente da configuração de teste presente na tabela de consulta ('OU-Exclusivo' ou 'Não-OU-Exclusivo'), uma vez que as suas entradas são totalmente controláveis, pelo facto de serem entradas primárias do circuito e ainda porque a simples inversão de uma delas permite determinar o valor na sua saída.



**Figura 7.15:** Cone de influência da saída YB

De forma idêntica, obtém-se para as entradas da *primitiva b* as equações lógicas:

**Equação 7.3**

$$T_{b0} = \overline{S_b} \cdot S_d$$

$$T_{b1} = S_b \cdot S_d$$

e, para as da *primitiva d*, as equações lógicas:

**Equação 7.4**

$$T_{d0} = \overline{S_d}$$

$$T_{d1} = S_d$$

Ainda que  $c_1$  e  $d_0$  não sejam entradas primárias, as primitivas que as antecedem não fazem parte do cone de influência da saída YB para determinação do número mínimo de configurações de teste do circuito, pelas mesmas razões expostas para o caso da tabela de consulta. De facto,  $c_1$  é também uma saída primária do modelo de teste da *slice* (XB), cujo cone de influência apresenta apenas duas primitivas, tantas quantas o cone de influência em que  $d_0$  se insere, pelo que o número mínimo de configurações de teste é inferior ao número mínimo obtido. Desta forma, o seu valor é totalmente controlável por recurso a uma configuração não necessária para o seu próprio teste, mas imposta pelo cone de influência da saída, que determina o número mínimo de configurações de teste da *slice*.

Com todas as condições de testabilidade determinadas, torna-se possível resolver o problema da minimização do número de configurações de teste para o subcircuito em análise. Na Tabela 7.2, as linhas indicam a totalidade das configurações possíveis, representando as colunas as condições de testabilidade de cada entrada. Uma cruz indica que a condição de testabilidade é satisfeita pela configuração correspondente. O uso desta representação torna o problema equivalente a um problema de cobertura clássico. Uma das possíveis soluções está indicada pelos círculos na tabela.

**Tabela 7.2:** Tabela de cobertura para determinação do número mínimo de configurações de teste

$S_{a1}$	$S_{a0}$	$S_b$	$S_d$	$T_{a0}$	$T_{a1}$	$T_{a2}$	$T_{a3}$	$T_{b0}$	$T_{b1}$	$T_{d0}$	$T_{d1}$
0	0	0	0	*						*	*
0	0	0	1					*			*
0	0	1	0						*		
0	0	1	1					*			*
0	1	0	0						*		
0	1	0	1	*							*
0	1	1	0		*						
0	1	1	1			*					*
1	0	0	0					*			
1	0	0	1		*						*
1	0	1	0			*					
1	0	1	1				*				*
1	1	0	0			*					
1	1	0	1		*						*
1	1	1	0				*			*	
1	1	1	1					*			*

Repare-se, contudo, que a solução indicada não é a única. As condições de testabilidade das entradas  $a_0$ ,  $a_1$ ,  $a_2$  e  $a_3$  impõem quatro configurações que cobrem também as condições de testabilidade das entradas  $b_0$  e  $d_1$ . Para a entrada  $b_1$ , pode escolher-se entre quatro configurações diferentes e para a entrada  $d_0$  entre seis. Todavia, qualquer que seja a escolha, o número mínimo de configurações é seis. Pelas razões apresentadas anteriormente, este será o número mínimo de configurações da *slice* de teste.

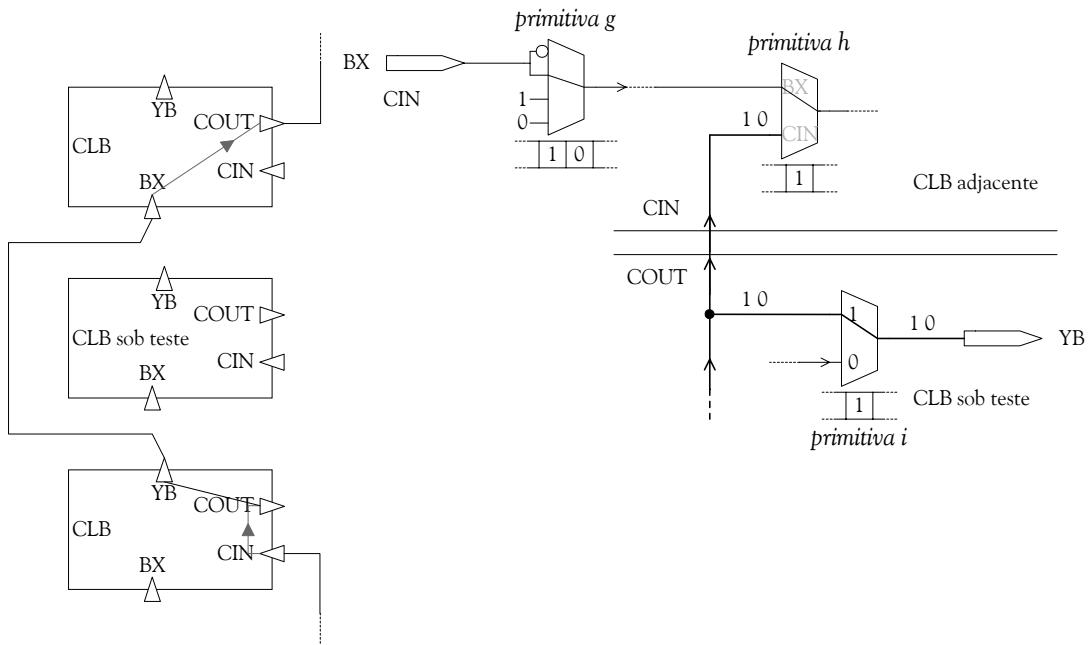
#### 7.2.3.2 TESTE DAS LINHAS DE TRANSPORTE E DA LÓGICA ASSOCIADA

As linhas de transporte visam reduzir o atraso de propagação deste tipo de sinais, normalmente associados a implementações aritméticas (acumuladores, subtractores, multiplicadores, comparadores, contadores, registo de deslocamento), factor com forte impacto no tempo de execução dessas funções. Para alcançar esse objectivo, os blocos lógicos dispõem de linhas dedicadas que interligam directamente a saída da lógica de transporte de um CLB com a entrada do mesmo tipo de lógica no CLB que lhe está superiormente adjacente (na coluna), evitando a utilização de recursos de interligação genéricos e a passagem pelas matrizes globais de encaminhamento. A impossibilidade de acesso à entrada e à saída primária da linha de transporte, como se observa na Figura 7.6, limita não só a sua capacidade de teste, mas também a da lógica que lhe está associada em cada *slice*, ao não permitir a aplicação de estímulos de teste e a recolha de respostas, a não ser através dos CLBs que lhe estão imediatamente adjacentes, situação inviável dentro da metodologia proposta.

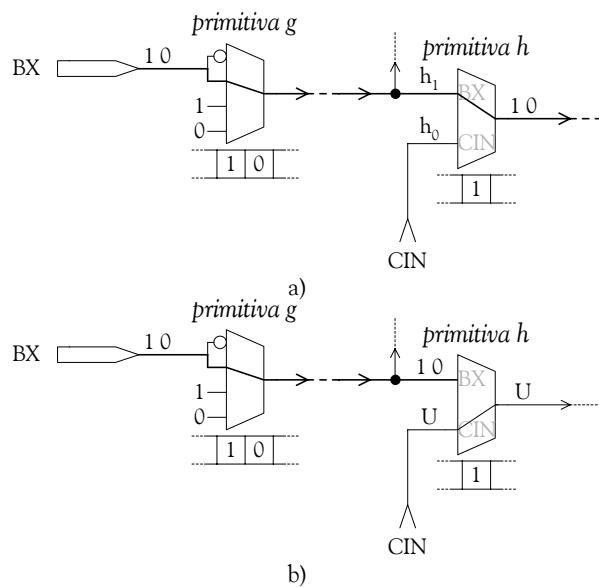
Contudo, é possível recolher as respostas resultantes da aplicação de estímulos à linha COUT e, consequentemente, à linha CIN do CLB superiormente adjacente, através da saída YB, conforme ilustrado na Figura 7.16. De notar que, mesmo que esta saída estivesse a ser usada anteriormente à replicação pela função aí implementada, a introdução, entre ambos os CLBs, do CLB sob teste implica a sua quebra e o consequente reencaminhamento dos sinais de transporte através de recursos genéricos, passando o sinal de transporte a ser encaminhado para o interior do CLB superiormente adjacente através da entrada BX, o que condiciona a configuração das primitivas  $g$  e  $h$  que o encaminham internamente, como se observa igualmente na Figura 7.16. Em consequência, a linha é completamente libertada e o seu teste torna-se possível.

A impossibilidade de acesso à entrada primária CIN impede, contudo, a totalidade do teste funcional da *primitiva h*. De facto, como se ilustra na Figura 7.17-a), é possível testar a entrada  $h_1$  e a sua saída, mas não a entrada  $h_0$ , como se observa na Figura 7.17-b). Esta entrada é proveniente da saída COUT do CLB inferiormente adjacente ao CLB sob teste, que mantém o seu funcionamento normal, apesar de, como se observa na Figura 7.16, o sinal de transporte ser

desviado para a saída YB, uma vez que as saídas primárias COUT e YB são electricamente iguais. Desta forma, como a funcionalidade da FPGA é mantida inalterável durante o decurso do teste, a entrada  $h_0$  exibirá um nível de tensão bem definido, mas que, no entanto, é desconhecido da lógica de teste. A única certeza resultante da aplicação do teste ilustrado na Figura 7.17-b) é que, sendo 1 o valor programado na *primitiva h* para selecção da saída, o multiplexador não estabelece continuidade entre a entrada  $h_1$  e a saída. Não é, no entanto, possível ter certezas quanto ao estabelecimento da continuidade entre  $h_0$  e a saída.



**Figura 7.16:** Recolha de estímulos do teste da linha de transporte



**Figura 7.17:** Condicionantes ao teste da linha de transporte

De reparar contudo que, segundo o modelo de faltas considerado para o multiplexador e exposto na secção 7.2.2, a detecção de faltas do tipo sempre-a nas suas entradas e saída, incluindo as entradas configuráveis, é, segundo [Portal, 99], suficiente para a detecção de faltas estruturais no interior desta primitiva. Ora acontece que, conjugando o resultado do teste dos CLBs adjacentes, ambas as entradas e a saída da primitiva são efectivamente controladas e observadas no que diz respeito à possibilidade de exibirem faltas do tipo sempre-a. Daqui resulta que, embora não seja possível durante o teste do CLB, aplicar valores à entrada  $h_0$ , o seu teste contra o modelo de faltas considerado foi já efectuado, ou será efectuado, consoante o sentido de deslocamento do CLB sob teste, durante o teste do CLB inferiormente adjacente.

Como os CLBs colocados no fundo das colunas não possuem a possibilidade de ligação de um sinal à entrada da linha de transporte, por inexistência de um CLB inferiormente adjacente na coluna, esta consideração não lhes é aplicável. Resulta pois que a cobertura de faltas, sob o modelo considerado, ao nível do CLB é efectivamente de 100%.

#### 7.2.3.3 GERAÇÃO DOS CONJUNTOS DE VECTORES DE TESTE

A geração dos vectores de teste para cada uma das configurações segue, a exemplo do método para obtenção do número de configurações mínimas, a partição da *slice* em cones de influência para cada uma das saídas [McCluskey, 84]. Em cada uma das configurações, as entradas que afectam uma determinada saída são testadas tendo em conta os vectores determinados na secção 7.2.2 para o teste de cada uma das primitivas. O conjunto final de vectores para cada cone é obtido pela compactação dos vectores de cada uma das primitivas que constituem o cone, os quais são, por sua vez, compactados com os dos restantes cones.

Um exemplo desta compactação encontra-se ilustrado na Figura 7.18, onde vários vectores são compactados para permitir o teste em simultâneo da tabela de consulta  $G$ , configurada como ‘OU-Exclusivo’, da porta lógica ‘E’ e das entradas  $a_3$  da *primitiva a* e respectiva saída, da entrada  $c_0$  da *primitiva c* e respectiva saída, conjuntamente com a linha de transporte, e da entrada  $d_1$  da *primitiva d* e respectiva saída, coincidente com a saída primária YB. A entrada  $b_0$  da *primitiva b* e a respectiva saída, a entrada  $e_0$  da *primitiva e* e a respectiva saída, coincidente com a saída primária Y, e a entrada  $f_1$  da *primitiva f* e a respectiva saída, que encaminha os sinais de saída da tabela de consulta para o elemento de retenção, permitindo o seu teste em simultâneo, aparecem na Figura 7.18 apenas parcialmente testadas, uma vez que estão somente representados um quarto dos dezasseis vectores a aplicar nas entradas da tabela de consulta, para o seu teste completo nesta configuração. O outro quarto resulta no mesmo conjunto de vectores à entrada das primitivas abrangidas, enquanto a metade restante provoca a alteração do valor na saída da tabela de consulta

e a consequente alteração do valor na linha de selecção da primitiva  $c$ , permitindo completar o teste das entradas  $b_0$  e  $e_0$  e respectivas saídas das suas primitivas, bem como a saída primária  $Y$ , e eventualmente efectuar o teste da entrada  $c_1$ . Este último teste está contudo dependente da configuração de outras primitivas, não constantes da Figura 7.18, nesta configuração de teste. Note-se que a configuração da tabela de consulta como ‘Não-OU-Exclusivo’ repete a sequência na sua saída, pelo que o teste de  $c_1$  continua viável nessa situação, podendo ser relegado para essa configuração de teste, em função da configuração das outras primitivas.

Um dos possíveis conjuntos de vectores, abrangendo as entradas representadas na Figura 7.18, a aplicar com esta configuração de teste, encontra-se representado na Tabela 7.3. O valor  $c_1$  na saída  $YB$  depende, como referido, de outras entradas do bloco lógico, diferentes das apresentadas na figura.

Em duas das configurações de teste do bloco, a tabela de consulta obriga à aplicação sequencial de 16 vectores de teste, num total de 32 vectores. O teste dos elementos de retenção em modo síncrono efectua-se simultaneamente com o teste deste bloco, uma vez que esses 32 vectores permitem a aplicação de 16 impulsos de relógio ao elemento de retenção, possibilitando o teste total da sua funcionalidade neste modo.

O número de vectores a aplicar em cada uma das restantes quatro configurações de teste é de apenas dois. Estes vectores permitem testar as restantes entradas dos multiplexadores cuja configuração inicial facultava o encaminhamento dos sinais da saída das tabelas de consulta, durante o seu teste, até uma das saídas primárias do circuito.

A aplicação dos vários conjuntos de vectores de teste a cada uma das configurações conduz ao teste múltiplo de várias primitivas e interligações, aumentando a sua eficácia na detecção de defeitos não explícitos no modelo [Hughes et al., 84] e [Hughes, 88].

O teste dos recursos de interligação e das matrizes de encaminhamento local efectua-se conjuntamente com o teste dos CLBs, uma vez que é através desses recursos que são aplicados os vectores de teste e recolhidas as respostas. As matrizes de encaminhamento de entrada e saída são implementadas como desmultiplexadores programáveis que permitem encaminhar um sinal de entrada para um determinado número possível de saídas. O teste dessas matrizes obriga, por isso, ao uso de várias configurações, dependente do número máximo de saídas para onde o sinal que chega à matriz pode ser encaminhado – seis nas matrizes de encaminhamento de entrada e oito nas de saída – conduzindo a igual número de configurações de teste para cada uma. Daí resulta um aumento do tempo de latência para todas as faltas, situação idêntica à verificada também com o teste das matrizes de encaminhamento global.

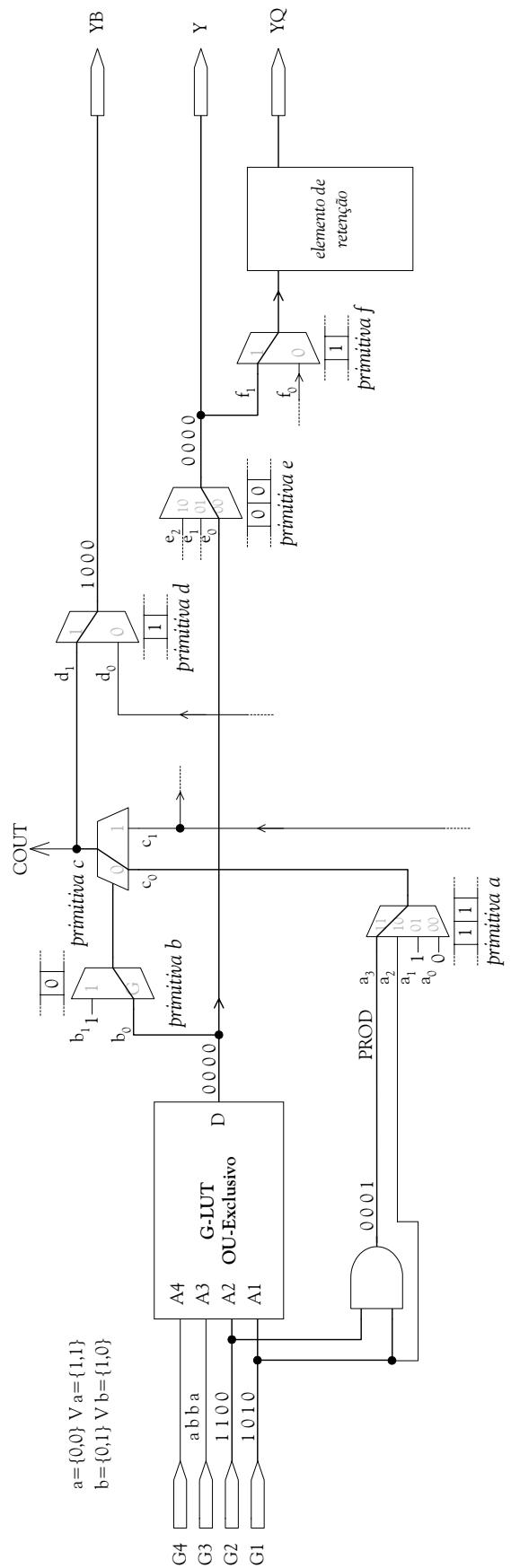


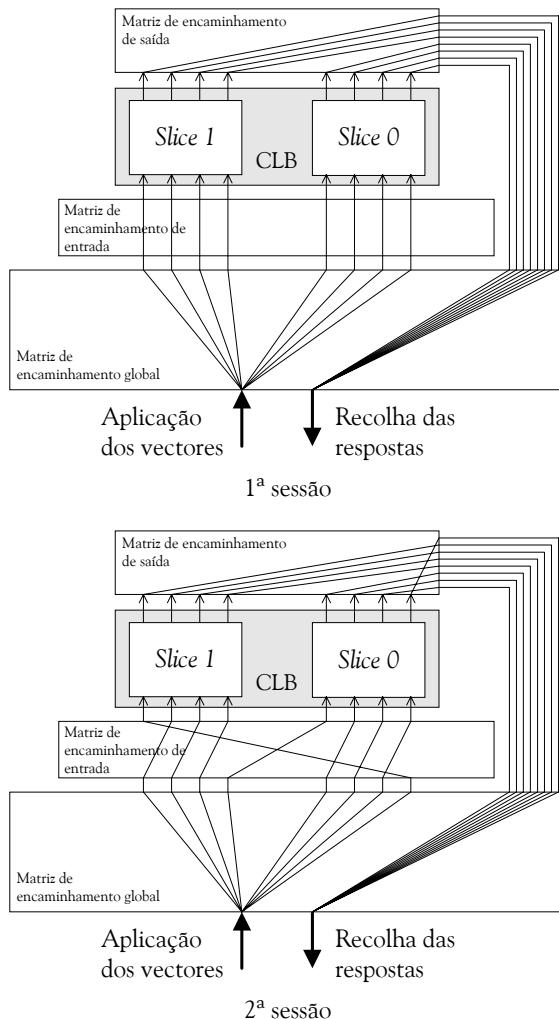
Figura 7.18: Exemplo da compactação de vectores de teste

**Tabela 7.3:** Conjunto de vectores de teste

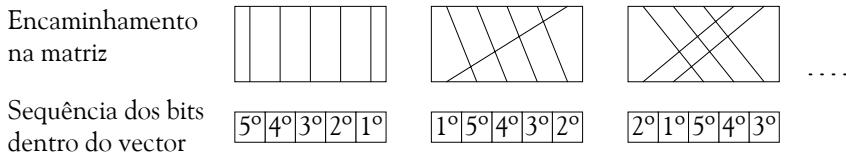
Vector	G4	G3	G2	G1	YB	Y
1	0	0	0	0	0	0
2	0	1	0	1	0	0
3	0	1	1	0	0	0
4	0	0	1	1	1	0
5	0	1	0	0	$c_1$	1
6	0	0	0	1	$c_1$	1
7	0	0	1	0	$c_1$	1
8	0	1	1	1	$c_1$	1
9	1	0	0	0	$c_1$	1
10	1	1	0	1	$c_1$	1
11	1	1	1	0	$c_1$	1
12	1	0	1	1	$c_1$	1
13	1	1	0	0	0	0
14	1	0	0	1	0	0
15	1	0	1	0	0	0
16	1	1	1	1	1	0

Uma solução de compromisso para contrariar esse acréscimo é o teste de apenas um subconjunto dos recursos de encaminhamento por cada sessão de teste do CLB, evitando o dispêndio de tempo em reconfigurações. Em cada sessão, a configuração dos recursos de encaminhamento das matrizes de entrada e saída mantém-se constante, sendo testados cerca de 16,7% dos recursos de encaminhamento da matriz de entrada e 12,5% dos recursos da matriz de saída. Assegurando que cada subconjunto é diferente em cada uma das seis sessões seguintes, garante-se o teste da totalidade dos recursos da matriz de entrada e obtém-se em apenas oito sessões a mesma garantia para a matriz de saída. Este aumento do tempo de latência só afecta o teste dos recursos de encaminhamento dentro da matriz e não as interligações entre as matrizes locais e os blocos lógicos, que são totalmente testadas em cada sessão.

De notar que a alteração da configuração das matrizes locais de encaminhamento não implica a alteração do encaminhamento das linhas que conduzem os vectores de teste a serem aplicados entre a matriz local e o Registo do Utilizador, passando pela matriz global, nem das linhas que capturam as respostas e as conduzem às células de captura escolhidas no registo BS, como se exemplifica para duas sessões na Figura 7.19. Contudo, uma vez que existe uma troca de sequência motivada pela alteração de encaminhamento nas matrizes locais, é igualmente necessário alterar a sequência de bits dentro do vector a aplicar antes da sua transferência para o Registo do Utilizador, bem como ter em atenção a alteração de sequência quando se procede ao deslocamento dos resultados obtidos através da cadeia BS. A forma mais simples será a implementação de um movimento circular na alteração do encaminhamento e, em consequência, na sequência dos bits dentro do vector a aplicar, em que o primeiro de uma sequência passa a ser o último da sequência seguinte, com o encaminhamento respeitando a sequência de alteração que se observa na Figura 7.20. Essa reflecte-se de igual forma no encaminhamento das respostas e na sequência pela qual elas são capturadas e deslocadas para o exterior.



**Figura 7.19:** Exemplo do teste das matrizes locais de encaminhamento



**Figura 7.20:** Sequência de teste das matrizes locais

Refira-se que, dependendo dos recursos de encaminhamento disponíveis, a sequência pela qual as respostas são deslocadas para o exterior pode não coincidir, logo à partida, com a posição relativa das saídas no CLB. As saídas são encaminhadas para células livres do registo BS, não sendo rígido que a sua posição dentro do registo obedeça à mesma ordem observada nas saídas do CLB.

#### 7.2.4 RESOLUÇÃO DO TESTE E TOLERÂNCIA A FALTAS

Dado que a recolha das respostas aos vectores é efectuada separadamente para cada saída do bloco, a existência de uma falta conduz imediatamente à identificação do cone de influência e, como tal, do conjunto de primitivas e interligações internas e dos recursos de interligação e encaminhamento da matriz de encaminhamento local onde a falta está localizada, sem necessidade do emprego de mais vectores de teste para diagnóstico dessa falta, por se considerar que os recursos de encaminhamento global que conduzem os vectores até à matriz de entrada, bem como as respostas da matriz de saída até ao registo BS, foram previamente testados e não exibem qualquer falta. Caso contrário, uma falta que afectasse uma dessas linhas impediria o teste correcto do CLB e induziria um erro de diagnóstico.

A implementação de um mecanismo de tolerância a faltas não requer mais resolução do que esta no diagnóstico de uma possível falta. Depois da sua detecção, pode-se substituir o CLB faltoso na sua totalidade por outro não usado, ou substituir apenas a *slice* afectada pela falta. Todavia, há que ter em conta que, neste último caso, dificilmente a *slice* não afectada voltará a ser usada, a não ser que ocorra uma falta numa das *slices* de algum CLB adjacente. Caso assim não aconteça, é necessário recorrer a um rearranjo da disposição das funções implementadas para o seu aproveitamento, uma vez que a transferência para essa *slice* boa da funcionalidade de uma outra *slice* faltosa, situada num CLB afastado, conduziria a um aumento dos tempos de propagação da função implementada nessa *slice* faltosa, o que poderia afectar o seu funcionamento.

O estudo de uma estratégia para o rearranjo das funções implementadas em caso de detecção de uma falta, variável consoante a primitiva afectada e a sua área de influência, e a definição do posicionamento na matriz de CLBs de blocos sobresselentes que permitam a substituição de CLBs faltosos, com o mínimo de custos para o funcionamento das funções afectadas, devidamente

adaptada às características da metodologia de teste exposta, constitui uma das linhas a explorar no futuro.

### 7.2.5 TESTE DAS TABELAS DE CONSULTA EM MODO DE MEMÓRIA

Apesar de, pelas razões indicadas anteriormente, nomeadamente a impossibilidade da sua relocação, não ser viável a utilização da metodologia apresentada para o teste das tabelas de consulta configuradas em modo de memória, nada impede que, sendo de interesse, seja programada uma configuração para testar os recursos envolvidos nesse modo. Esses recursos compreendem alguma lógica adicional destinada a controlar o processo de escrita nas células da tabela, bem como algumas linhas extra, nomeadamente uma linha de habilitação de escrita (WE), uma de entrada de dados (DI) e uma linha a mais de endereço, A5, devido à possibilidade de associação das duas tabelas de consulta de uma mesma *slice*, para formação de uma memória de 32x1 bit. Note-se que, enquanto as operações de escrita na tabela/memória são síncronas com o sinal de relógio, as operações de leitura são completamente assíncronas, revelando nisso um comportamento idêntico ao que possuem quando actuam como simples geradoras de funções, na medida em que o porto de saída reflecte a cada momento o valor da posição de memória especificada no barramento de endereços ( $F1=A1$ ,  $F2=A2$ ,  $F3=A3$ ,  $F4=A4$  e de igual forma para as entradas  $Gx$ ). O conteúdo inicial das células de memória da tabela pode ser especificado no ficheiro de configuração, tal como acontece quando aí são implementadas funções de lógica combinatória, sendo, por omissão, a memória inicializada com todas as suas posições a 0. É possível a utilização das tabelas de consulta para constituição de blocos de memória com palavras de maior comprimento e maior número de posições, bem como de memórias de dupla porta, mas tal implica a utilização de recursos genéricos de configuração, já anteriormente testados. Desta forma, as metodologias de teste clássicas para o teste de memórias, como os *march tests*, podem ser reaproveitadas [Papachristou et al., 85], [Dekker et al., 88], [Mazumder et al., 96], [Azimane et al., 98], [Renovell et al., 98b] e [Bushnell et al., 00].

### 7.2.6 LATÊNCIA DO TESTE

A latência do teste depende sobretudo da estratégia de varrimento adoptada para cada recurso e, por isso, tem mais a ver com a estratégia de rotação do que propriamente com os procedimentos de teste em si. Recorde-se que as equações que permitem determinar o tempo de latência de qualquer falta dentro de um bloco lógico foram já apresentadas no capítulo seis, final da secção 6.1.4 (Equações 6.2 e 6.3). Deve, no entanto, acrescentar-se que, nos casos em que simultaneamente se testem os recursos de encaminhamento da matriz de entrada e da matriz de saída, o procedimento exposto não implica aumento do tempo de teste, uma vez que não obriga à aplicação de vectores

extra em cada sessão ou ao estabelecimento de configurações de teste extra. Por seu lado, o tempo máximo de latência de uma falta num PIP na matriz de encaminhamento é igual ao tempo máximo de latência de uma falta num CLB (dado pela Equação 6.2) multiplicado pela metade do número de sessões de teste necessárias para testar cada matriz (seis para a matriz de entrada e oito para a de saída), visto que o tempo máximo corresponde a dois varrimentos completos da matriz – um num sentido e o outro noutro – e, como tal, a duas sessões de teste.

### 7.3. TESTE DAS INTERLIGAÇÕES

O princípio a aplicar para o teste de interligações é idêntico ao usado no dos blocos lógicos, devendo, no entanto, registar-se a substituição do bloco sob teste por uma fieira de linhas sob teste, e ter em atenção que o comprimento do vector a aplicar é, no máximo, de treze bits, limite imposto pelo comprimento do Registo do Utilizador para a aplicação dos vectores de teste (definido em função do número de entradas de teste do bloco lógico). De notar que esta limitação não tem relação directa com o número de linhas sob teste simultaneamente, uma vez que, dependendo do algoritmo usado, várias linhas podem estar ligadas à mesma célula. O número de pontos de recolha depende apenas da quantidade de recursos livres no registo BS que podem ser usados para esse fim e da possibilidade de encaminhamento desses pontos até eles.

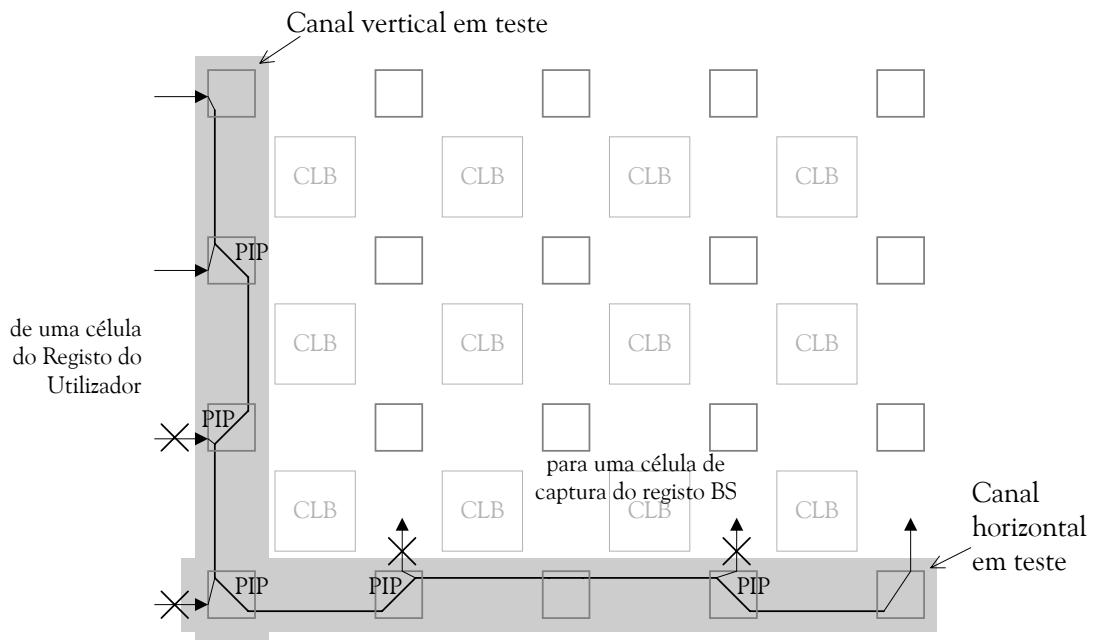
O teste dos recursos de interligação numa FPGA apresenta algumas particularidades relativamente ao mesmo tipo de teste efectuado em estruturas não reconfiguráveis, como ASICs ou placas de circuito impresso. A possibilidade de configuração implica, à partida, a necessidade de ter em conta as faltas específicas que podem afectar os elementos configuráveis da estrutura, PIPs, que permitem a interligação dos segmentos. Esses PIPs podem ser basicamente afectados por dois tipos de faltas: sempre-fechados e sempre-abertos, indicando a presença de um transístor sempre fechado ou sempre aberto, respectivamente, por defeito no próprio transístor ou na célula da memória de configuração que define o seu estado. Para os segmentos em si, são de esperar faltas do tipo segmento aberto, curto entre dois segmentos e entre um segmento e uma das linhas de alimentação (faltas em ponte).

É evidente que as faltas do tipo segmento aberto cobrem as faltas do tipo PIP sempre-aberto, da mesma forma que as faltas em ponte cobrem as faltas do tipo PIP sempre-fechado e os segmentos sempre-a-0 ou sempre-a-1, pelo que o modelo de faltas para o teste das interligações se resume a faltas do tipo:

- segmento sempre-aberto;
- ponte.

As fíeiras de linhas sob teste são constituídas por linhas resultantes da união de vários segmentos ao longo de um dos canais em teste, no caso de segmentos mais curtos que a extensão do canal, que por sua vez são interligadas na matriz de encaminhamento global com linhas que se desenvolvem ao longo do outro canal em teste, perpendicular ao primeiro.

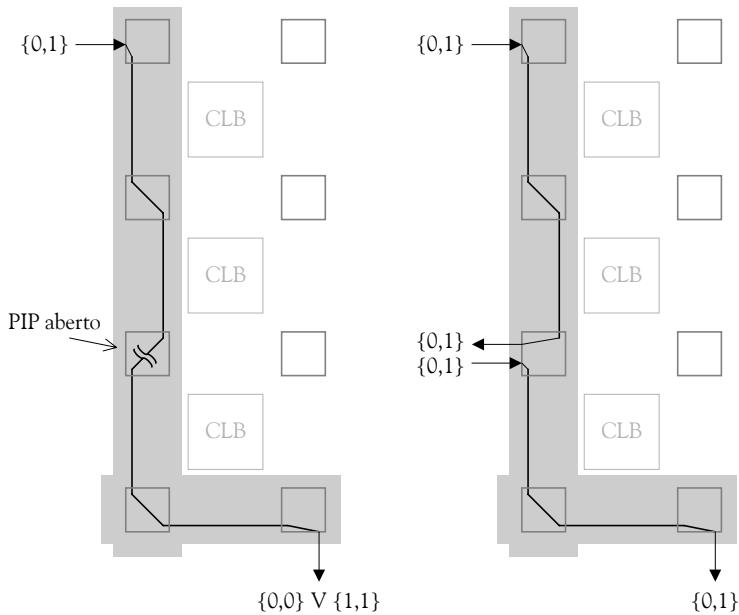
Uma linha constituída por vários segmentos comporta-se como uma associação em série de impedâncias, conforme discutido na secção 5.4.2, e não como um fio em que todos os pontos são electricamente iguais. Desta forma, o teste de uma linha só se pode considerar efectivo se a aplicação e a recolha do valor se efectuarem algures nos seus segmentos extremos, como se indica na Figura 7.21. Caso contrário, a existência de uma falta num dos extremos pode levar ao aparecimento de um divisor de tensão que provoque uma leitura errada do valor na linha sob teste, se esta se efectuar algures a meio (este comportamento foi longamente discutido na secção 5.4.2). Obviamente, as linhas que se estendem ao longo de todo o comprimento do canal não apresentam este problema. Estas linhas devem ser interligadas durante o teste, nas matrizes globais que tal possibilitem, com linhas transversais que, como elas, se estendem ao longo de todo o comprimento do canal, ou com linhas constituídas por interligações de segmentos, que se estendam dessa matriz até ao ponto de captura da resposta aos vectores de teste.



**Figura 7.21:** Pontos possíveis de aplicação e captura de teste numa linha sob teste

Como não se conhece o desenho exacto dos recursos de encaminhamento e a sua posição relativa nas várias camadas de metal, considera-se possível a existência de um curto-circuito (ponte) entre quaisquer dois ou mais segmentos de um mesmo canal.

A detecção de uma falta do tipo segmento sempre-aberto numa linha sob teste obriga à aplicação num dos extremos de uma sequência de teste  $\{0, 1\}$  e à observação da sequência capturada no outro extremo. Esta sequência deve ser igual na ausência de uma falta, ou um valor estável ( $0$  ou  $1$ ), qualquer que seja o vector de entrada, no caso de um dos segmentos que constitui a linha sob teste se encontrar em aberto, ou de existir uma falta do tipo sempre-aberto num dos PIPs que deveria estabelecer a interligação entre os vários segmentos para a formar. A determinação do segmento ou PIP faltoso obriga à sucessiva divisão da linha sob teste, processando-se esta através da sua abertura, por configuração de um PIP intermédio, e a aplicação a ambas as partes da mesma sequência de teste. O desaparecimento da falta indica o seu tipo (sempre-aberto) e permite determinar a sua localização (o PIP que antes estabelecia a ligação entre as duas partes). Este caso encontra-se ilustrado na Figura 7.22. Se a falta não desaparecer deve continuar-se com o processo de divisão e teste da parte na qual a falta se mantém, até ser possível o seu diagnóstico. No limite, a falta situar-se-á num dos segmentos individuais que restar no final das sucessivas divisões. Esta situação encontra-se retratada na Figura 7.23.



**Figura 7.22:** Diagnóstico de um PIP com uma falta do tipo sempre-aberto

O teste de PIPs contra faltas do tipo sempre-fechado obriga à sua configuração como PIPs abertos e à aplicação de uma sequência de teste  $\{0, 1\}$  sobre o segmento de entrada. A observação do vector obtido no ou nos segmentos de saída, atendendo a que, nas matrizes de encaminhamento, uma entrada pode ser encaminhada para várias saídas, permite concluir se o PIP está efectivamente aberto ou não. Em caso de falta, o vector observado em cada uma das saídas é idêntico ao vector aplicado na entrada. Contudo, este procedimento de teste obriga à configuração de um PIP de cada

vez e à aplicação de uma sequência de teste, o que implica um tempo de teste significativamente longo, devido ao grande número de configurações envolvidas.

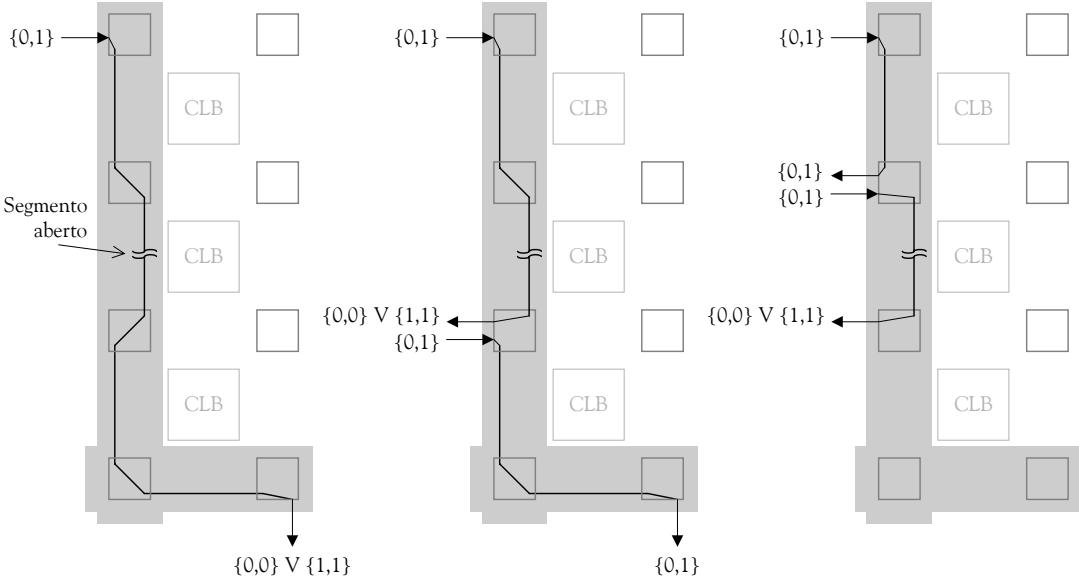
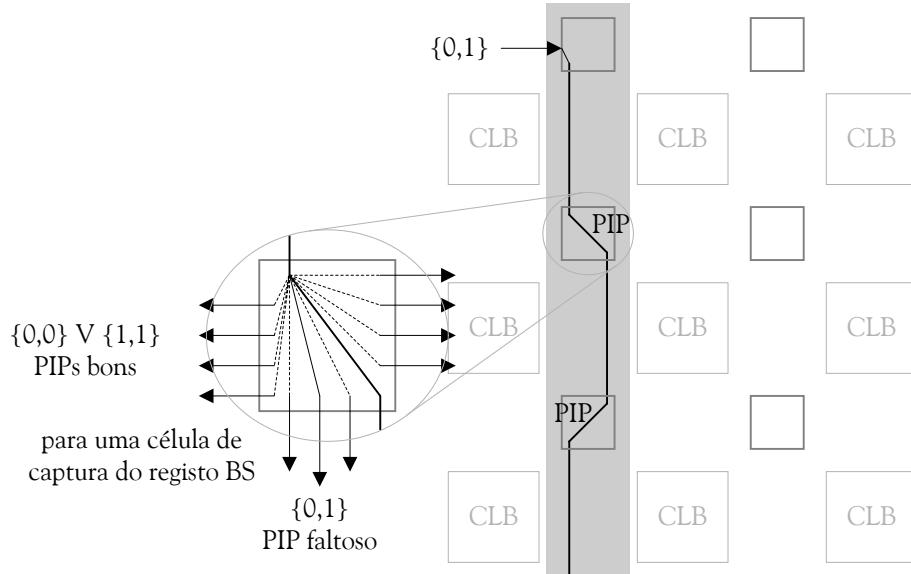


Figura 7.23: Diagnóstico de um segmento aberto

No entanto, o teste de PIPs contra faltas do tipo sempre-fechado pode efectuar-se simultaneamente com o teste contra faltas do tipo segmento sempre-aberto numa linha sob teste. A sequência de teste aplicada propaga-se a todas as entradas dos PIPs que interligam os vários segmentos que constituem a linha sob teste. Cada uma dessas entradas tem a possibilidade de encaminhar o sinal de entrada não só para a saída que está activa, mas para várias outras, mantidas desligadas pela configuração dos PIPs. No caso de uma falta do tipo sempre-fechado afectar um desses PIPs, a saída exibirá essa sequência de teste, em vez de apresentar um valor estável, como se ilustra na Figura 7.24. Por esse facto, durante o teste das linhas contra faltas do tipo segmento sempre-aberto, a captura de todas as possíveis saídas, para a entrada ocupada em cada uma das matrizes atravessadas pela linha sob teste, permite detectar igualmente possíveis PIPs sempre fechados. Em consequência, o teste da totalidade dos segmentos contra faltas do tipo sempre-aberto implica igualmente o teste da totalidade dos PIPs contra faltas do tipo sempre-fechado.

Um PIP sempre-fechado pode causar uma falta em ponte entre duas linhas sob teste. Todavia, este caso, equivalente à existência de uma ponte entre dois segmentos de duas linhas sob teste, não está coberto pelo teste anterior. A detecção de faltas deste género, que se podem comportar como portas lógicas do tipo 'E' ou 'OU', obriga à aplicação de sequências complementares entre si a todos os segmentos sob teste. Estas sequências podem ser compactadas com a anterior sequência  $\{0, 1\}$ , para permitir o teste simultâneo contra todas as faltas do modelo considerado. O resultado é igual àquele que proviria da aplicação do Algoritmo de Partição Binária apresentado em [Kautz, 74].

Este método, que permite o teste simultâneo e em paralelo de várias faltas em ponte, é suficiente para cobrir a totalidade das faltas uma vez que, pelo menos num dos vectores, existem valores lógicos complementares aplicados a cada uma das linhas sob teste. A Tabela 7.4 apresenta um exemplo de um conjunto de vectores de teste obtido com base na utilização deste algoritmo.



**Figura 7.24:** Teste de PIPs contra faltas do tipo sempre-fechado

**Tabela 7.4:** Conjunto de vectores para o teste das interligações

Vector 1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
Vector 2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1
Vector 3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
Vector 4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

Da observação da Tabela 7.4 conclui-se que, ao contrário do que acontece com todos os outros, os 1º e último bit de cada vector se mantêm constantes ao longo dos quatro vectores, pelo que esta sequência não detecta uma possível falta do tipo curto entre as linhas a que esses bits são aplicados e uma das linhas de alimentação. Para evitar tal situação, essas sequências de valores lógicos (vectores de teste sequenciais) sempre-a-0 e sempre-a-1 não devem ser usados no teste.

Para os 96 segmentos ou linhas paralelas presentes num canal vertical (supondo que seria possível o seu teste simultâneo), necessitamos de sete vectores de teste, gerados de forma idêntica, pelo que seriam possíveis  $27=128$  vectores de teste sequenciais. Assim sendo, e como apenas 96 são necessários, pode-se eliminar as combinações que não permitiriam detectar os curtos com uma das linhas de alimentação. No entanto, devido à limitação imposta pelo comprimento do Registo do

Utilizador, o número de configurações de teste necessárias é superior às sete fixadas pelo número mínimo de vectores. Para a aplicação do primeiro vector necessitamos de pelo menos duas células para aplicar os valores **0** e **1** a cada uma das partes em que se divide o conjunto de linhas sob teste. Para o segundo, com o conjunto dividido em quatro partes, de pelo menos quatro células, e assim sucessivamente. De notar que a partir do quarto vector, inclusive, o número de células necessárias ultrapassa o número de células do Registo do Utilizador, pelo que a aplicação de um vector tem de ser efectuada de forma faseada, correspondendo a cada fase uma nova configuração. A utilização de células independentes para aplicação do mesmo valor a várias partes, em vez da opção pelo uso de uma única célula para todas essas partes, independentemente do seu número, prende-se com a capacidade de diagnóstico de qual a linha afectada pela falta. Caso contrário, a falta seria detectada mas não diagnosticada, sendo preciso recorrer a um teste adaptativo. A Tabela 7.5 sumaria o número de configurações necessárias para a aplicação dos vectores às 96 linhas sob teste.

**Tabela 7.5:** Número mínimo de configurações de teste

Vector	Nº de pontos de aplicação do vector	Nº configurações
1	2	1
2	4	1
3	8	1
4	16	2
5	32	3
6	64	5
7	64	5
Total.....		18

Deste modo, o teste e diagnóstico da totalidade das faltas só ocorre ao fim de 18 configurações diferentes. Para não aumentar exageradamente a latência do teste para as outras faltas, apenas se procede à implementação de seis das configurações por cada vez que o canal está em teste. Desta forma, a totalidade do teste das interligações contra faltas em ponte só ocorre a cada três vezes que a totalidade dos canais se encontre em teste. De referir ainda que este valor é meramente teórico, já que pode não ser possível conjugar os segmentos ocupados com os que é necessário libertar, a cada fase do teste, para garantir que todos os segmentos são testados dentro destas 18

configurações. É pois provável que, dependendo das funções implementadas e do índice de ocupação da FPGA, o número de configurações possa ser superior, o que implica igualmente um aumento do tempo de latência destas faltas. Neste sentido, o teste da totalidade das interligações afigura-se bastante mais trabalhoso que o teste dos blocos lógicos, em termos da sua execução prática.

#### 7.4. TESTE DA MEMÓRIA DE CONFIGURAÇÃO

Numa FPGA com tecnologia de programação baseada em memória estática, é o estado das células da memória de configuração que configura a lógica para desempenhar a funcionalidade pretendida pelo utilizador. Essas células encontram-se distribuídas por entre os elementos lógicos que controlam, para implementar a função lógica e os recursos de encaminhamento que interligam os vários blocos. A possibilidade de reconfiguração de apenas parte da memória exige, no entanto, que essas células possuam uma forma de organização que possibilite o seu endereçamento. No caso das Virtex, como já foi referido, a memória de configuração assemelha-se a uma matriz rectangular de bits, que se encontram agrupados em vectores verticais, com um bit de largura, estendendo-se do topo ao fundo da matriz. Cada um desses vectores controla uma parte dos bits de configuração da lógica interna das *slices* e dos recursos de encaminhamento associados ao bloco lógico em que elas se inserem e constitui a mais pequena unidade endereçável.

O teste da memória de configuração implica, por isso, não só o teste das próprias células, mas também o teste da lógica que possibilita a escrita e a leitura endereçada do seu conteúdo. No entanto, dado que não é possível desacoplar as células da sua função de configuração dos recursos lógicos, não é viável a utilização das técnicas tradicionais de teste de memórias estáticas, pois tal resultaria numa configuração aleatória desses recursos, podendo eventualmente implicar uma configuração lógica que provocasse danos na própria estrutura da FPGA. Além disso, mesmo considerando como viável essa possibilidade, ela não seria compatível com o teste concorrente definido na metodologia aqui apresentada.

Contudo, o teste dos blocos lógicos e dos recursos de encaminhamento permite, igualmente, detectar as faltas que, na memória de configuração, afectem os bits que controlam a configuração desses recursos, porque qualquer uma dessas faltas resultaria num mau funcionamento do recurso sob teste [Stroud et al., 96c] e [Abramovici et al., 99].

O bom estado estrutural dos recursos lógicos é pois, *de per si*, uma garantia do bom funcionamento da memória de configuração, salvaguardada a possibilidade de ocorrência de erros na transmissão de dados para essa memória, os quais são, no caso da família Virtex, detectados com recurso a um

código de verificação por redundância cíclica [Xilinx, 98] e [Xilinx, 00], ou ainda de erros devidos a SEUs e SETs durante a sua operação normal.

Uma outra forma de detecção de faltas na memória de configuração é a leitura da parte da memória recentemente reconfigurada, aproveitando a possibilidade da leitura parcial endereçada dos vectores de configuração, através da infra-estrutura BS, e a comparação do resultado com o ficheiro de configuração parcial enviado. Desta forma é possível diagnosticar a existência de faltas do tipo sempre-a, embora este procedimento conduza a um aumento do tempo de latência das faltas se for consecutivamente efectuado após cada reconfiguração. De notar contudo que, no caso de se verificar a existência de um erro, a sua correcção deve ser tentada anteriormente ao diagnóstico de uma falta estrutural da célula, uma vez que o erro pode ser devido a um SEU ou a um SET.

Com base nas situações mencionadas neste último caso é recomendável a leitura da totalidade da memória, em vez de se restringir à parte recentemente alterada, em busca de erros emergentes, quer permanentes, quer provocados pela incidência de radiação. Uma vez que a aplicação da metodologia de teste apresentada não permite o funcionamento em modo de memória das tabelas de consulta, pode-se, se necessário para corrigir erros não permanentes nas células, reconfigurar dinamicamente parte ou a totalidade da FPGA. Esta possibilidade é válida porque a sua configuração, não sendo estática, apenas é alterada por configuração externa da memória e não acarreta o risco do surgimento de incoerências nos valores aí armazenados. Este procedimento obriga à manutenção em memória externa de uma cópia actualizada da configuração da FPGA, o que não implica, no entanto, um dispêndio acrescido de memória, visto que a sua existência é indispensável quando se usam recursos configuráveis baseados em memória estática volátil. Por seu lado, o aparecimento de erros que se reflictam de forma periódica nas posições de memória ou no conteúdo dos vectores de configuração, indica a presença de faltas na lógica de endereçamento e escrita desses vectores.

## 7.5. SUMÁRIO

Este capítulo apresentou a metodologia adoptada para o teste dos recursos libertados pela transferência da funcionalidade anteriormente aí implementada. O objectivo foi o teste estrutural de todos os recursos: lógicos, de encaminhamento e de interligação.

O facto de a FPGA ser uma estrutura reconfigurável faz depender o teste estrutural da configuração dos seus recursos. No sentido de reduzir ao mínimo o número de reconfigurações necessárias, grandes consumidoras de tempo e, por isso, com um custo elevado, apresentou-se uma metodologia para a determinação do número mínimo de configurações de teste para os blocos lógicos. O

pequeno número de reconfigurações e de vectores de teste a aplicar por cada uma traduz a eficácia da solução proposta. Adicionalmente, são testadas também as matrizes de encaminhamento e as interligações locais, sem aumento de custos, embora à custa de uma maior latência de teste.

Mais moroso é o teste da estrutura de encaminhamento e interligação global, para o qual se propôs uma possível solução.

Por último, apresentou-se um método dirigido à memória de configuração, mais para verificação e possível correcção de erros do que propriamente para o seu teste. A impossibilidade de separar esta memória das suas funções de configuração da estrutura da FPGA limita significativamente as suas possibilidades de teste. Contudo, e uma vez que as faltas nesta memória afectam a própria configuração da estrutura da FPGA, inclusive durante o teste, o bom funcionamento desta última é prenúncio do bom funcionamento da memória de configuração.

## **8. IMPLEMENTAÇÃO E VALIDAÇÃO**



Neste capítulo, discutem-se alguns aspectos ligados à implementação prática da metodologia apresentada, bem como os principais problemas enfrentados na sua concretização e as soluções desenvolvidas. Apresenta-se, depois, um caso prático de incorporação deste método e a sua validação, continuando-se com a análise de alguns dos valores temporais obtidos com a implementação da referida metodologia, por se considerar que ilustram as condições em que esta pode ser uma alternativa viável para o teste de FPGAs inseridas em sistemas reconfiguráveis.

Um dos maiores problemas com que a realização deste trabalho se defrontou foi o da falta de ferramentas que permitissem lidar com as características de reconfiguração parcial dinâmica dos novos componentes programáveis. As ferramentas disponibilizadas pelos fabricantes permitem apenas a concepção de circuitos estáticos clássicos, não suportando a geração de ficheiros parciais e a sua programação. Outro problema foi a falta de informação acerca da estrutura interna dos próprios componentes, que obrigou a um laborioso trabalho experimental para obtenção de um conjunto de elementos que outorgassem não só a concepção das ferramentas, mas também o entendimento da estrutura interna da própria FPGA e das suas possibilidades. Tratava-se de um processo imprescindível para formular correctamente as metodologias expostas e evitar discrepâncias entre a teoria e a realidade, das quais resultaria provavelmente um desenvolvimento metodológico meramente teórico e desconexo da prática.

Deste modo, a validação do trabalho implicou o desenvolvimento de ferramentas de apoio à reconfiguração parcial dinâmica, nas quais se incluem a geração de ficheiros de reconfiguração parcial e a sua programação no componente através da infra-estrutura *Boundary Scan*.

O capítulo prossegue com a exemplificação, num pequeno caso prático, da implementação da metodologia e conclui com a listagem dos resultados que foi possível obter experimentalmente a partir dessa implementação, bem como a sua extrapolação para um caso genérico.



## 8.1. IMPLEMENTAÇÃO

O desenvolvimento e validação da metodologia exposta neste trabalho deparou-se com uma série de problemas relacionados com a parca informação disponibilizada pelo fabricante dos componentes usados, as Virtex da Xilinx, que se deve essencialmente à protecção de soluções que possam eventualmente ser copiadas por outros fabricantes, e com a inexistência de ferramentas que permitissem a sua implementação prática.

O desenvolvimento teórico foi sobretudo baseado num trabalho preliminar de pesquisa de informação acerca do próprio componente, tendo como fontes a folha de características do componente [Xilinx, 00b], associada a um conjunto mais ou menos vasto de elementos de maior detalhe sobre as suas características de configuração e organização interna [Xilinx, 00], [Xilinx, 00c] e [Xilinx, 00d], conjuntamente com exemplos de aplicação [Xilinx, 00e], [Xilinx, 01]. A partir dessa recolha e da realização de experiências de implementação de algumas funções, foi possível reunir um conjunto consistente de informação sobre a arquitectura interna do componente, que permitiu fundamentar a formulação da metodologia de teste proposta.

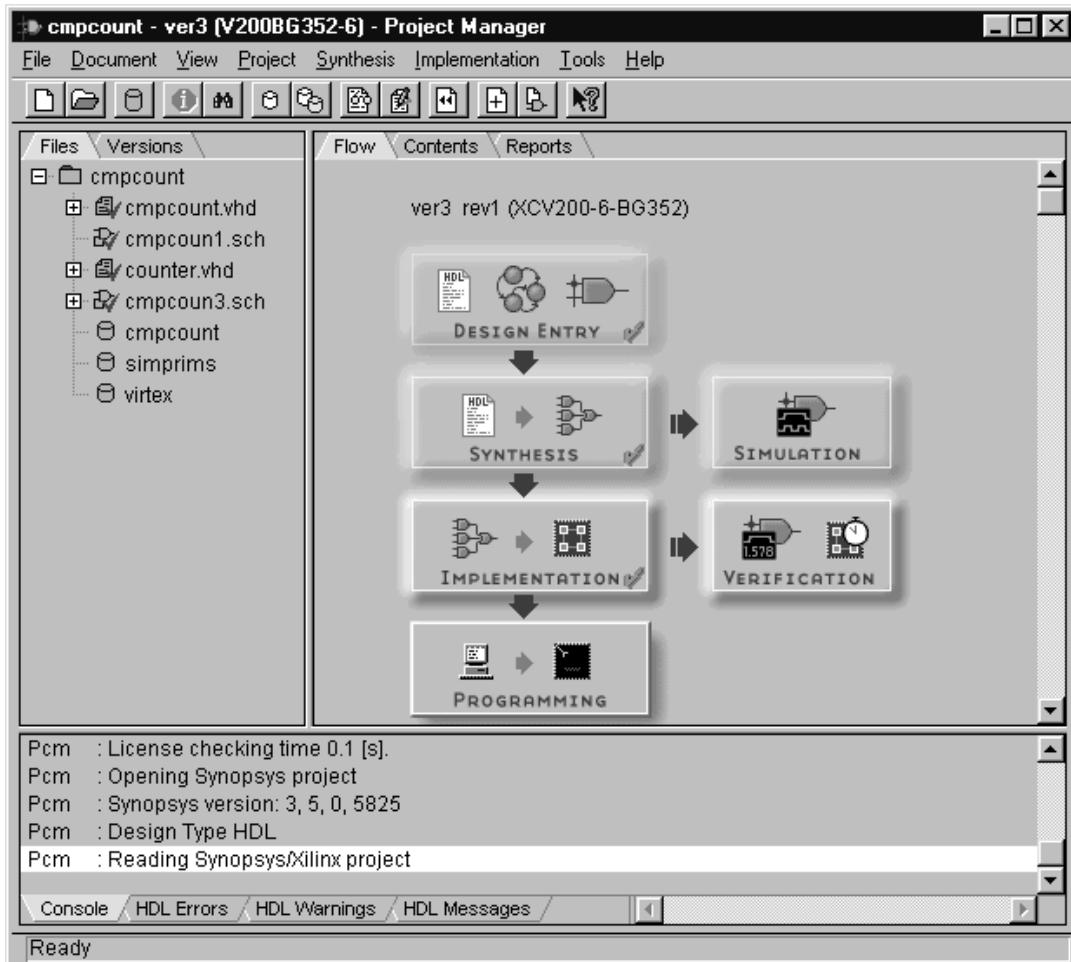
A essa formulação seguiu-se uma etapa experimental em que cada uma das soluções foi ensaiada, começando pela replicação de CLBs configurados com diferentes tipos de funcionalidades. Nessa fase, colocou-se a questão da possibilidade de, durante o paralelo das entradas ou das saídas, se forçar, em virtude da existência de uma falta, o curto-circuito entre nós que exibissem valores lógicos diferentes. O comportamento da interligação nestas condições foi extensivamente estudado, como se deu conta no capítulo cinco, determinando-se quais as suas implicações no funcionamento da função, bem como os problemas que a sua existência, embora momentânea, pudesse provocar no comportamento estrutural da própria FPGA.

A determinação dos custos resultantes da opção por uma ou outra das estratégias de rotação constituiu a fase experimental subsequente, de cujos resultados se fez a descrição no capítulo seis. Para esse fim, recorreu-se à implementação de um subconjunto de 14 circuitos, do conjunto de 22 designados por “ITC'99 benchmarks”, desenvolvidos pelo Grupo de CAD do Politécnico di Torino [Politécnico di Torino, 99] e [Corno et al., 00]. Este subconjunto cobre desde circuitos simples, de pequenas máquinas de estado a conversores de código, até outros mais complexos, como encriptadores de texto e processadores genéricos.

Por último, experimentou-se a infra-estrutura de teste pensada para a aplicação dos vectores e recolha das respostas, com base na reutilização da infra-estrutura BS, já discutida no capítulo sete.

Contudo, a consecução da fase experimental só foi possível após o desenvolvimento de uma ferramenta de software que permitisse de uma forma simples a manipulação directa do ficheiro de configuração da FPGA.

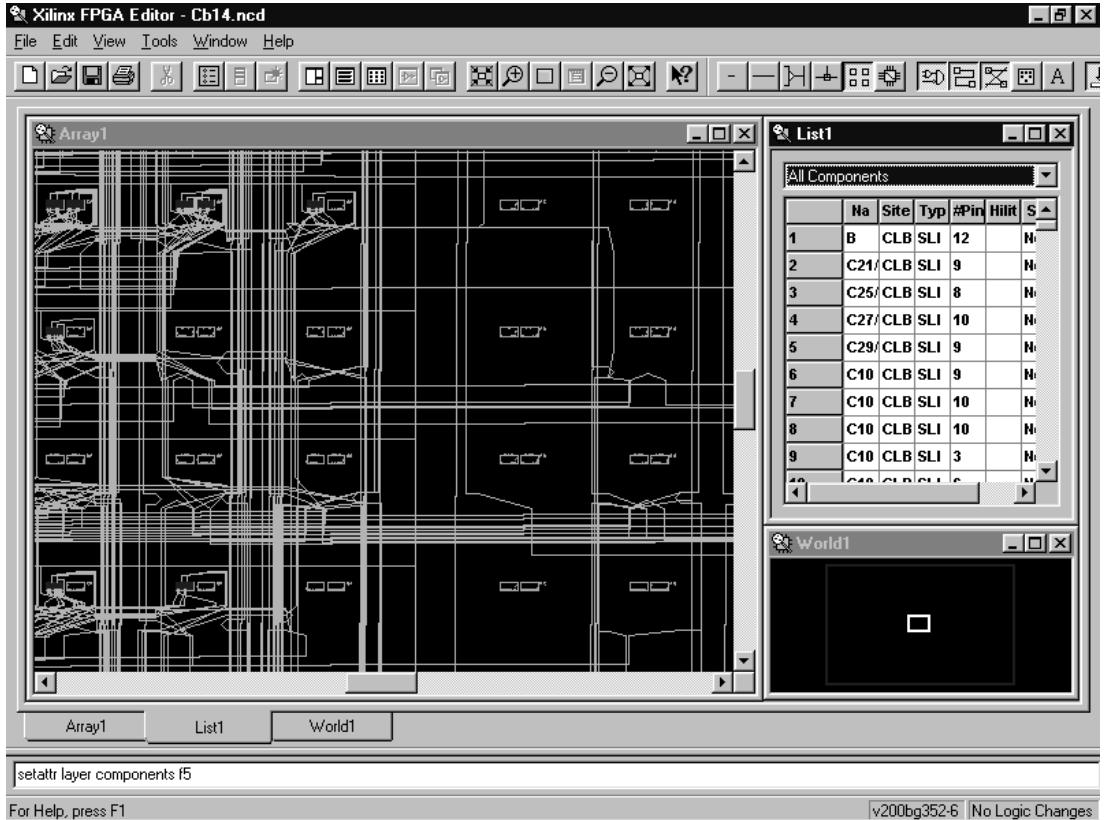
O pacote de ferramentas comercializadas pelo fabricante, denominado *Foundation*, permite a total implementação de um projecto desde a fase de descrição, com entrada por esquemático ou usando uma linguagem de programação (VHDL [IEEE 1076, 02] ou Verilog [IEEE 1364, 01]), até à fase de configuração da FPGA, passando pela síntese, simulação, mapeamento e verificação temporal. A janela de gestão de um projecto neste pacote de ferramentas está ilustrada na Figura 8.1. Todavia, não é permitida nem a geração nem a programação de ficheiros de reconfiguração parcial, mas somente a geração e programação de ficheiros de configuração total.



**Figura 8.1:** Janela de gestão de projecto no *Foundation*

Embora uma das ferramentas do pacote, o *FPGA Editor*, cuja interface se encontra ilustrada na Figura 8.2, permita manipular a posição dos CLBs configurados e das interligações, mantém-se uma dificuldade inultrapassável, que é a impossibilidade de se interligarem duas saídas. Na verdade, uma

vez que em condições normais esta interligação configura uma situação de erro, a ferramenta recusa-se a efectuá-la.

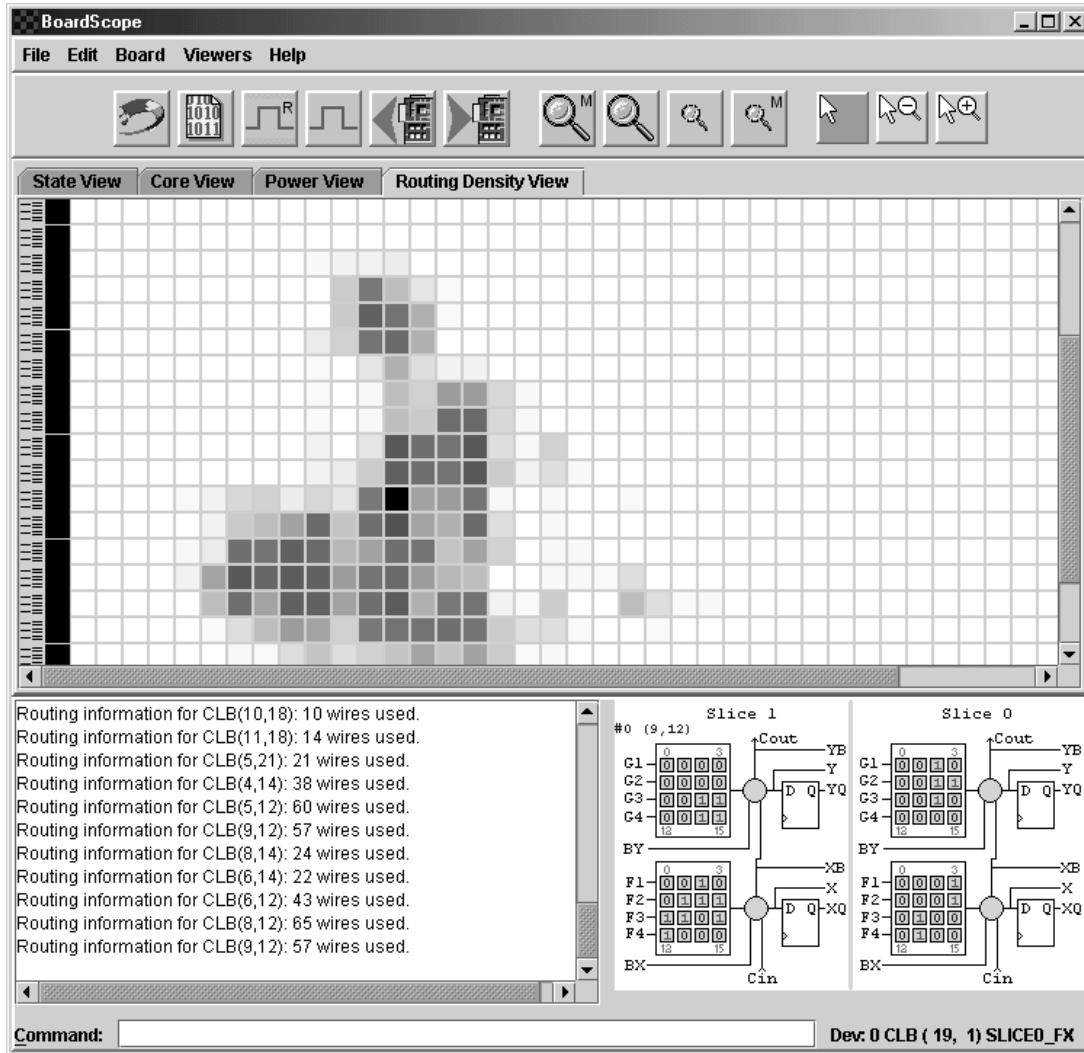


**Figura 8.2:** Janela de interface com o *FPGA Editor*

Por outro lado, se na fase experimental e de verificação da exequibilidade da solução proposta, o recurso a esta ferramenta permitia a obtenção de alguns resultados, embora à custa de um longo trabalho manual, o objectivo de tornar autónomo todo o processo de implementação da metodologia, indispensável à sua viabilidade prática, obrigava a conceber uma ferramenta que automatizasse todo o processo.

Um subsídio relevante na persecução deste objectivo foi o pacote *JBits* disponibilizado pela Xilinx [Xilinx, 00a]. Trata-se de um pacote constituído por um conjunto de classes Java que proporcionam uma *Application Programming Interface* (API), que permite a manipulação directa dos ficheiros binários de configuração das FPGAs. Deste modo, a interface funciona quer sobre ficheiros de configuração gerados pela ferramenta de projecto quer sobre ficheiros resultantes da leitura, através de uma operação de leitura total ou parcial, da memória de configuração da FPGA. Esta operação funciona de forma idêntica à operação de configuração e pode ser executada igualmente através da infra-estrutura BS, tirando partido do registo de configuração. A interface permite que todos os recursos configuráveis, como as tabelas de consulta, o encaminhamento dos sinais e os *flip-flops* na FPGA, sejam individualmente configurados [Guccione et al., 99].

Um dos programas desenvolvidos com base nesta interface é o *BoardScope* (fornecido com o pacote *JBits*), uma ferramenta de depuração de hardware com uma interface gráfica que permite visualizar a distribuição e o estado interno dos circuitos, enquanto o componente mantém o seu funcionamento normal. Os dados são obtidos através de uma operação de leitura e mostrados graficamente na interface, podendo esta operação ser repetida consecutivamente, o que possibilita uma amostragem do comportamento interno dos registos. A Figura 8.3 mostra um aspecto da interface gráfica deste programa, obtido com o circuito B05 dos “ITC’99 benchmarks” [Politécnico di Torino, 99] e [Corno et al., 00].



**Figura 8.3:** Janela de interface com o *BoardScope*

De notar, no canto inferior direito da figura, os dados referentes ao valor armazenado nas tabelas de consulta de um dos CLBs, referenciado pela sua posição na matriz (CLB(19, 1)), e, na janela à sua esquerda, a informação referente ao número de linhas usadas por esse CLB, para encaminhar os sinais que a ele chegam e dele partem. Na janela principal, é possível visualizar a densidade de

ocupação das linhas em redor de cada CLB, com os tons mais escuros a denunciar um maior índice de ocupação.

Uma interface normalizada, denominada XHWIF (*Xilinx HardWare InterFace*), é utilizada para estabelecer a comunicação entre o programa e as FPGAs localizadas numa placa. Esta interface inclui métodos para leitura e escrita dos ficheiros de configuração, incremento do sinal de relógio e leitura e escrita de memórias externas à FPGA, se presentes na placa. Ao normalizar a forma como as aplicações comunicam com o hardware, esta interface permite também o estabelecimento da comunicação entre aplicações como o *BoardScope* e uma grande variedade de placas. Uma das aplicações desenvolvidas pela Xilinx com base nesta interface é o *XHWIF Server*, que possibilita a comunicação, através da *Internet*, entre aplicações do tipo *BoardScope* e placas reconfiguráveis localizadas à distância. As aplicações mantêm totalmente a sua funcionalidade, apenas sendo afectadas pela maior lentidão derivada das condições de acesso.

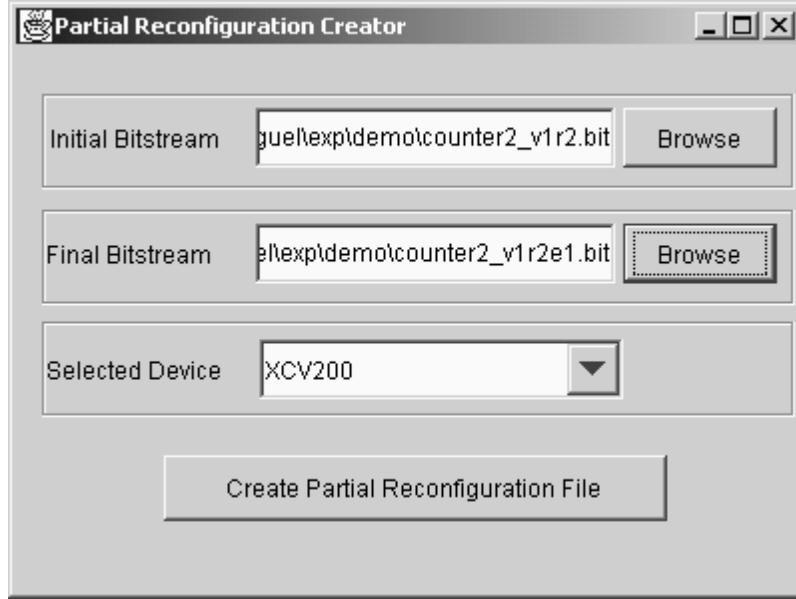
Uma outra possibilidade, no caso de inexistência do suporte físico ou durante operações de depuração, é a utilização da aplicação referida em modo de simulação, carregando directamente para o programa o ficheiro de configuração do circuito obtido a partir das ferramentas de desenvolvimento tradicionais. Em modo de simulação, o *BoardScope* possui a possibilidade de incremento do sinal de relógio e de visualização das formas de onda dos vários sinais.

Apesar dos seus aspectos positivos, nomeadamente para efeitos de depuração, esta aplicação não se coaduna com as necessidades que a experimentação do método proposto requer. Daí a necessidade do desenvolvimento de novas aplicações, que tirem inteiro partido do conjunto de classes *Java* definidas na API.

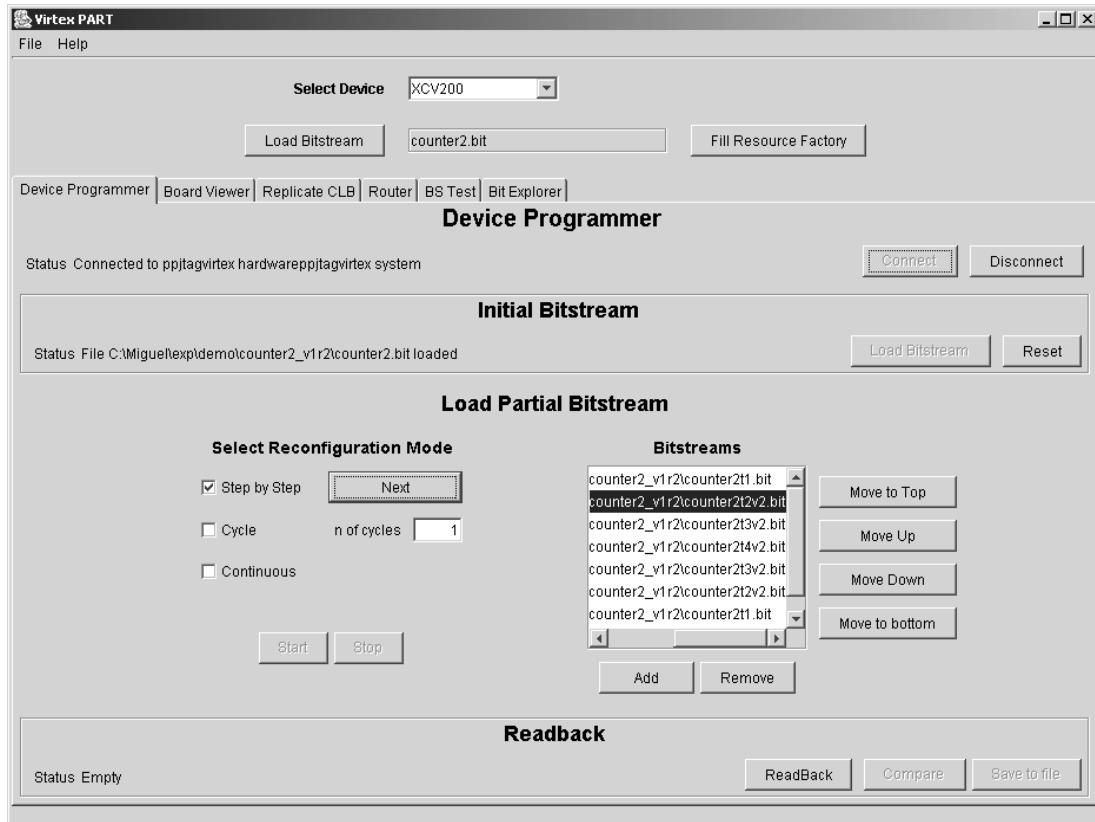
Uma das primeiras aplicações a ser desenvolvida, denominada *Partial Reconfiguration Creator*, foi um gerador de ficheiros de configuração parcial a partir de dois ficheiros de configuração total, cuja janela de interface se ilustra na Figura 8.4. Esta aplicação permitiu a realização das primeiras simulações usando a reconfiguração parcial.

Posteriormente, desenvolveu-se a aplicação *Virtex PART*, cuja interface com o utilizador está ilustrada na Figura 8.5. Entre outras funcionalidades que foram sendo acrescentadas ao longo do trabalho, a aplicação permite a transferência dos ficheiros de reconfiguração parcial de e para uma placa contendo uma FPGA XCV 200, ilustrada na Figura 8.6, através da utilização da infra-estrutura BS como interface de configuração, cujo porto de acesso é visível na parte superior esquerda da referida figura. No entanto, ainda antes do arranque das experiências, foi necessário projectar e construir uma fonte de alimentação comutada, capaz de fornecer os três valores de tensão (5, 3,3 e 2,5 V) nas condições exigidas pelo componente, e cujo esquema é fornecido em

anexo. A partir desta base mínima, procedeu-se à realização das primeiras experimentações práticas.



**Figura 8.4:** Janela de interface com o *Partial Reconfiguration Creator*



**Figura 8.5:** Janela de interface com o *Virtex PART*

Um dos pontos mais trabalhosos durante a fase experimental foi a do reencaminhamento dos sinais durante o processo de relocação dos blocos lógicos. Embora seja possível, com o *FPGA Editor*,

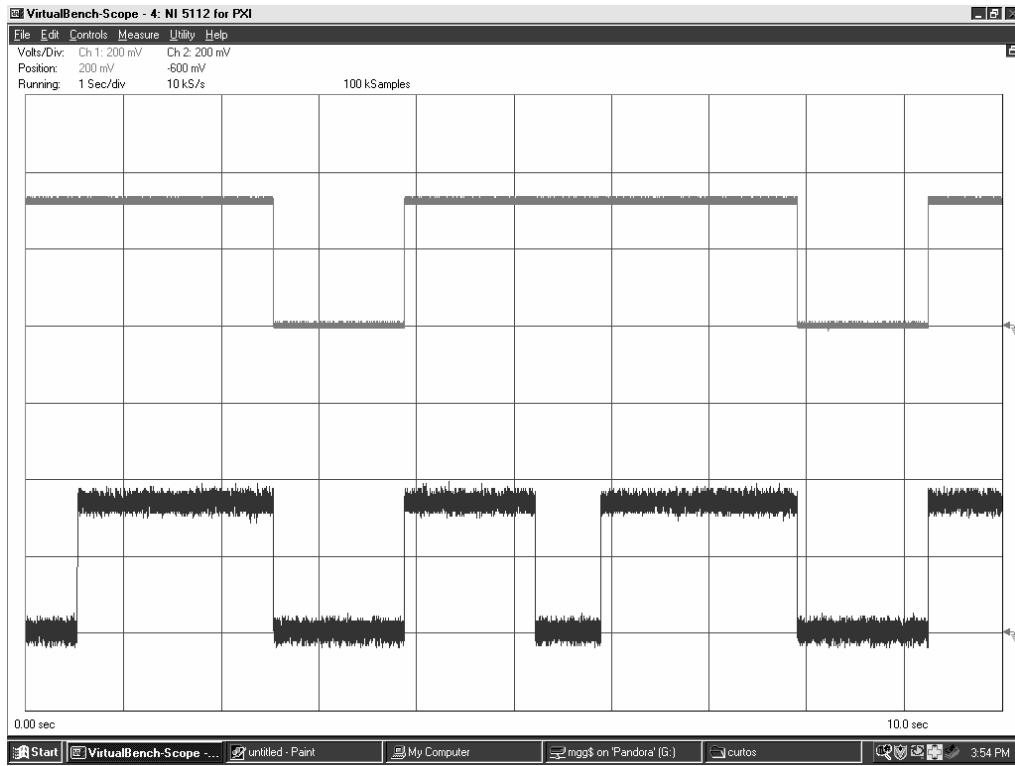
proceder à deslocação dos blocos lógicos e ao restabelecimento das suas ligações, tirando partido da interface gráfica (o que facilita uma primeira abordagem), operações como a colocação em paralelo das saídas, contrárias à prática comum, só podem ser executadas com recurso ao *JBits*, e, como tal, sobre o ficheiro de configuração. Dado que o *FPGA Editor* faz parte de um pacote de ferramentas concebido para o desenvolvimento de projectos num único sentido, não é possível, após a alteração do ficheiro de configuração pelo *JBits*, caminhar em sentido inverso e visualizar o resultado da alteração no *FPGA Editor*. Aliado à escassa informação disponibilizada pelo fabricante sobre os recursos de encaminhamento, esse problema dificultou a tarefa da aprendizagem neste ponto.



**Figura 8.6:** Placa de experimentação

Resolvida a questão da programação dos ficheiros de reconfiguração parcial no componente e do paralelo das saídas, analisou-se posteriormente o seu comportamento na presença de reconfigurações que originassem situações de curto-circuito, ao interligarem-se níveis lógicos diferentes, criados por defeitos que alteraram a funcionalidade implementada nos blocos lógicos. A análise desta situação entravava num obstáculo de monta: a impossibilidade de acesso à interligação, uma vez que tal implicaria o acesso ao interior do próprio componente. A única solução viável foi a do encaminhamento para o exterior dos sinais presentes em cada um dos segmentos da interligação, atravessando um bloco de E/S.

A fase experimental consistiu na interligação de duas saídas, forçadas a níveis lógicos diferentes, de dois CLBs, através de um número variável de segmentos, e na recolha e análise da evolução da tensão ao longo dessa interligação. Duas das formas de onda recolhidas ao longo desta fase experimental encontram-se ilustradas na Figura 8.7, resultando da sua observação as conclusões que se expuseram nas secções 5.4.2 e 5.4.3 deste trabalho.



**Figura 8.7:** Formas de onda recolhidas durante uma das fases experimentais

Uma vez que os resultados destas experiências levaram a concluir pela inexistência de riscos para o componente se se verificasse a ocorrência desta situação, em caso de defeito, a continuação do trabalho foi viabilizada. De notar que, a existir esse risco, a solução proposta deixaria de fazer sentido, pois, se ao colocar-se em paralelo um CLB defeituoso com outro previamente testado e em bom estado, se desse origem a um curto-circuito que danificasse ambos, em vez de se localizar e delimitar um defeito, este acabaria por ser propagado.

Ainda dentro do processo de relocação, a fase experimental seguinte consistiu na comprovação do bom funcionamento das soluções propostas para a replicação de lógica síncrona, sem e com sinal de habilitação de relógio, e de lógica assíncrona, recorrendo-se, para tal, à utilização de várias implementações dos circuitos-padrão definidos nas “ITC’99 benchmarks”, sobre as quais se executou o processo de forma exaustiva. O objectivo principal desta etapa foi a comprovação da eficácia do bloco auxiliar de replicação na correcta transferência dos valores de estado, independentemente da

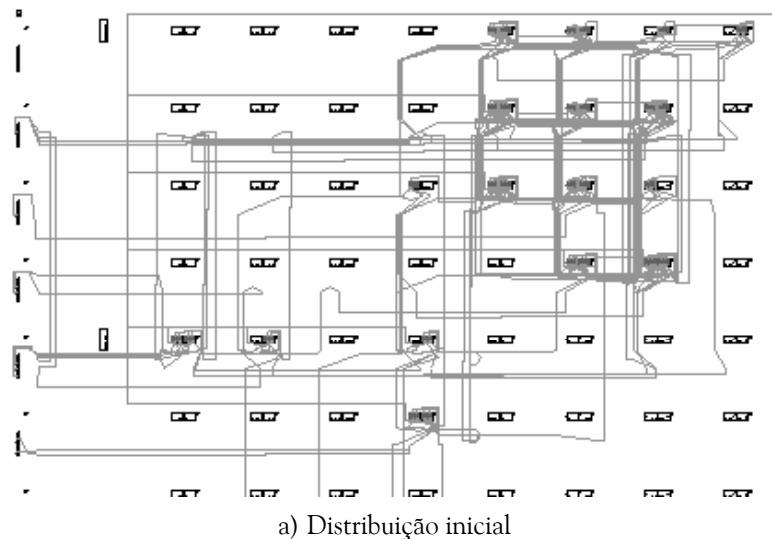
variação dos sinais envolvidos, no decorrer do normal funcionamento do circuito. Neste caso, a validade da solução proposta foi constatada pela simples observação do comportamento das formas de onda dos sinais externos dos circuitos implementados, uma vez que uma falha na transferência do valor implicaria imediatamente uma alteração no funcionamento do circuito, ou mesmo a ocorrência de um bloqueio na sua operação.

A análise de qual a melhor estratégia de rotação a utilizar em cada caso foi um trabalho moroso. A cada uma das implementações dos circuitos-padrão foram aplicadas ambas as estratégias de rotação, horizontal e vertical. A título de exemplo, a Figura 8.8 dá conta dessa aplicação ao circuito B08. Nessa figura, apenas aparecem a distribuição inicial, resultado da síntese a partir do pacote de ferramentas *Foundation*, e a distribuição final, após a aplicação das duas estratégias de rotação. Contudo, a aplicação de cada uma das estratégias a cada um dos 14 circuitos-padrão analisados, ocupando um total de 788 blocos lógicos, implicou a geração de um número de ficheiros de configuração igual ao número de CLBs, multiplicado pelo número de ficheiros necessários à replicação de cada CLB (três, sete ou oito, conforme se tratasse de CLBs implementando funções puramente combinatórias, ou funções síncronas sem ou com sinal de habilitação de relógio, respectivamente<sup>1</sup>), o que dá a ideia da dimensão do trabalho envolvido numa fase em que a automação do processo era ainda muito incipiente. Das conclusões retiradas da análise efectuada aos resultados recolhidos, deu-se conta ao longo de todo o capítulo 6.

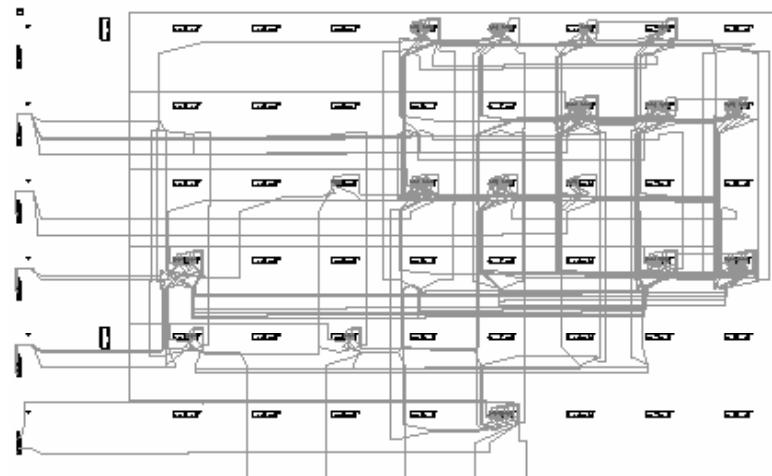
A aplicação do teste obrigou igualmente ao desenvolvimento de uma ferramenta que controlasse o deslocamento dos vectores para o Registo do Utilizador e os aplicasse ao bloco lógico sob teste, previamente configurado com uma das configurações de teste. De igual forma, a mesma ferramenta deveria permitir controlar a captura das respostas e o seu deslocamento através da cadeia BS. Esta ferramenta, cuja interface com o utilizador está ilustrada na Figura 8.9, foi integrada na aplicação *Virtex PART*. A primeira das configurações de teste estabelece os percursos dos sinais desde o Registo do Utilizador até às entradas do CLB sob teste, bem como os percursos desde as suas saídas até blocos de E/S completamente livres, situação em que as células BS associadas às linhas de saída e de controlo do terceiro estado de cada bloco podem ser usadas para recolha das respostas. A célula associada à linha de entrada, mesmo estando livre, não é passível de ser utilizada, dada a sua direccionalidade, de fora para dentro do componente. Por este motivo, não é possível fazer chegar até ao andar de captura/deslocamento da célula BS da linha de entrada um sinal vindo de dentro do componente.

---

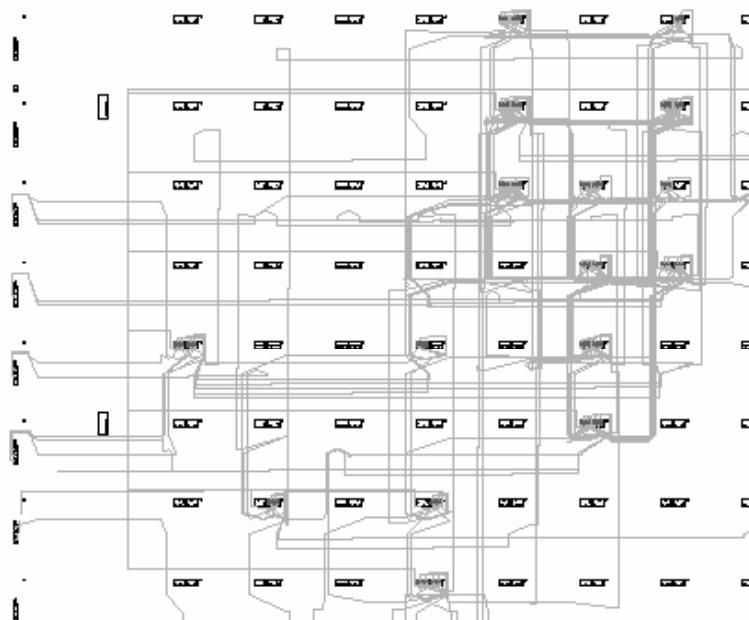
<sup>1</sup> Ver capítulo 5.



a) Distribuição inicial

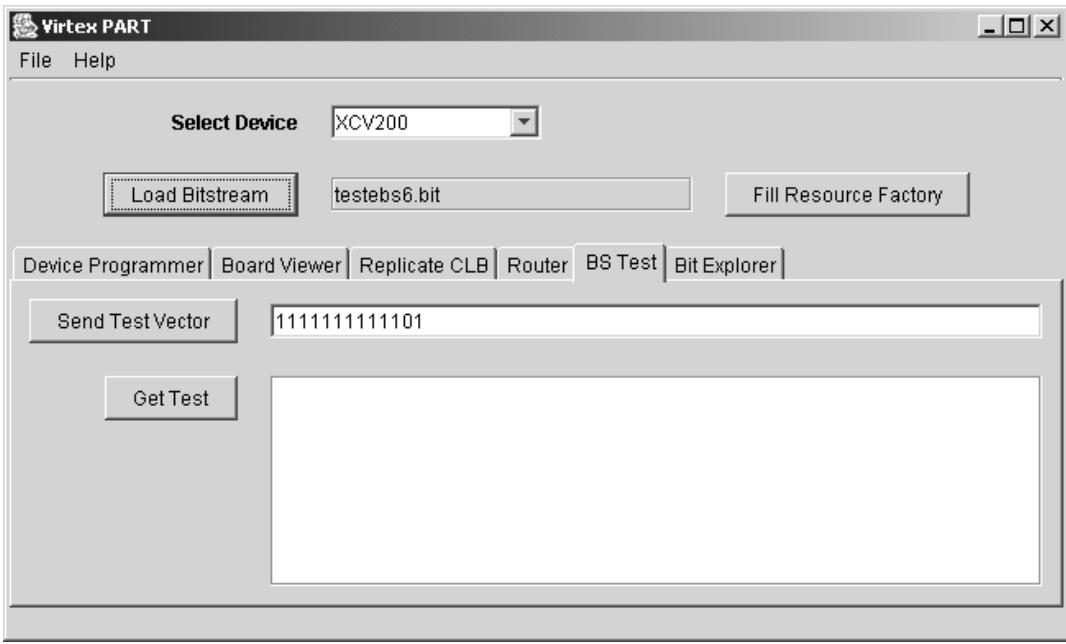


b) Distribuição após a rotação horizontal



c) Distribuição após a rotação vertical

**Figura 8.8:** Resultado da aplicação de ambas as estratégias de rotação à distribuição inicial do circuito B08



**Figura 8.9:** Janela de interface com a ferramenta de aplicação e recolha dos vectores de teste

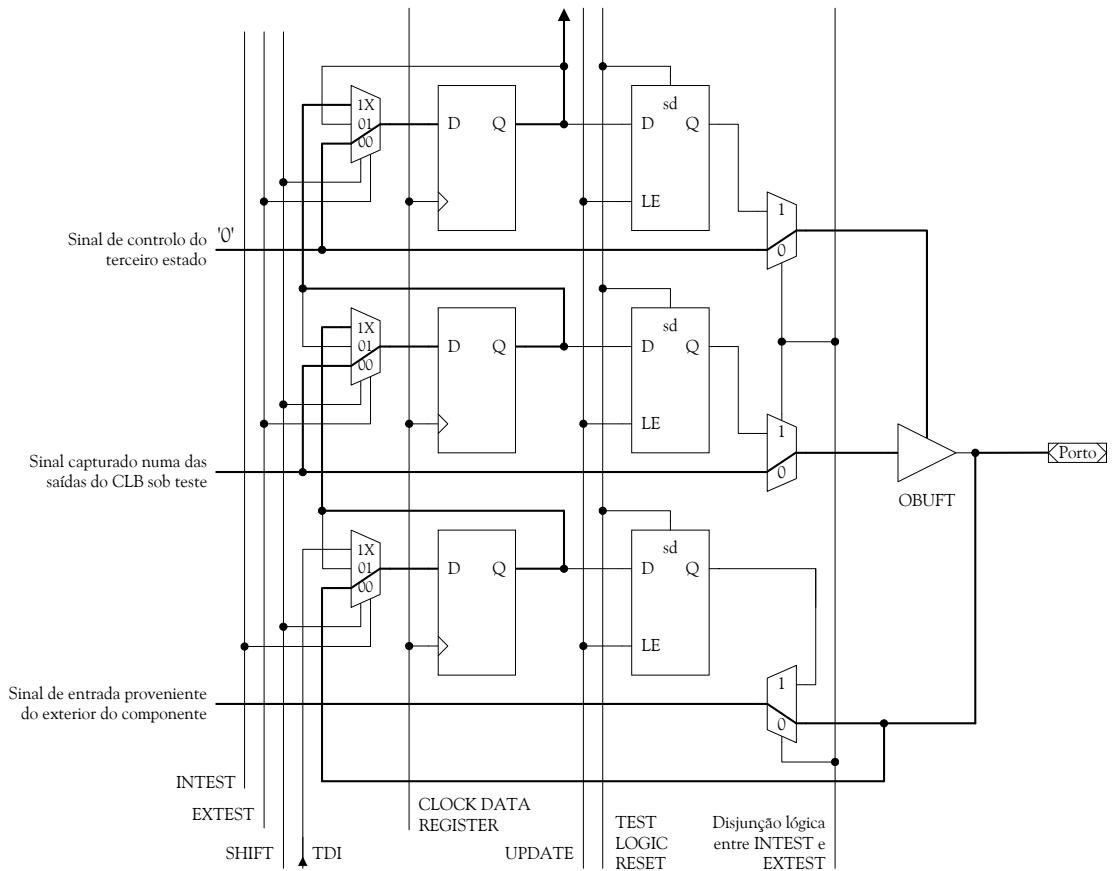
Outra restrição prende-se com o aproveitamento das células BS livres dos blocos configurados como entradas. Neste caso, apenas a célula do registo BS associada ao sinal de saída pode ser usada para recolha das respostas aos vectores de teste, uma vez que a linha associada ao controlo do terceiro estado deve ser mantida com um valor fixo (0), mantendo a saída em terceiro estado e evitando o aparecimento de conflitos com o sinal de entrada. Durante a captura, a instrução SAMPLE torna as células transparentes, pelo que a linha de saída contém o sinal proveniente da captura da resposta aos vectores aplicados. Esse sinal, obviamente, não deve ser aplicado ao porto, na medida em que o *buffer* de saída deve manter-se incondicionalmente em terceiro estado, situação cuja permanência deve ser assegurada pela própria configuração de teste. Na Figura 8.10, estão marcadas a carregado as linhas com sinal activo num bloco de E/S configurado como entrada, cuja célula BS de saída é utilizada para capturar um dos sinais provenientes de uma das saídas do CLB sob teste.

Depois de concluídas todas as experiências relacionadas com cada uma das partes em que se pode dividir a implementação da metodologia proposta (relocação, rotação e teste), procedeu-se à sua integração e aplicação a vários casos concretos, dos quais se apresenta, na próxima secção, um exemplo.

## 8.2. VALIDAÇÃO

A inclusão da metodologia proposta para o teste de FPGAs num sistema reconfigurável, a exemplo de qualquer boa prática de introdução de procedimentos de teste, inicia-se logo na fase de projecto,

com a junção da *macro BSCAN\_VIRTEX* e do Registo do Utilizador. Para lá da adição da descrição em VHDL do Registo, de modo a que este seja sintetizado conjuntamente com os restantes módulos do projecto, é necessário adicionar um ficheiro (extensão *.ucf – user constraints file*) contendo um conjunto de restrições destinadas a optimizar a sua colocação dentro da FPGA. Esse ficheiro, cujo conteúdo está representado na Figura 8.11, garante que o Registo do Utilizador ocupará apenas os sete CLBs situados fisicamente no canto superior esquerdo da matriz de blocos lógicos, junto à *macro BSCAN\_VIRTEX*. Esta permite o acesso aos sinais provenientes da infra-estrutura BS, necessários ao seu controlo, optimizando não só a ocupação dos recursos lógicos, mas também a dos recursos de encaminhamento. A Figura 8.12 ilustra esta situação, sendo igualmente visível o encaminhamento das saídas paralelas do Registo, que aplicam os vectores ao CLB sob teste. A colocação inicial do Registo do Utilizador nesta posição não invalida que, em posteriores reconfigurações durante o teste, possa ocorrer o seu reposicionamento, consoante a disponibilidade dos recursos de encaminhamento em cada instante.



**Figura 8.10:** Linhas activas num bloco de E/S configurado como entrada e usado para capturar uma das saídas do CLB sob teste

```

INST "U1/FG_11/q1_reg" LOC = "CLB_R6C1.S1" ;
INST "U1/FG_10/q1_reg" LOC = "CLB_R6C1.S1" ;
INST "U1/FG_9/q1_reg" LOC = "CLB_R5C1.S1" ;
INST "U1/FG_8/q1_reg" LOC = "CLB_R5C1.S1" ;
INST "U1/FG_7/q1_reg" LOC = "CLB_R4C1.S1" ;
INST "U1/FG_6/q1_reg" LOC = "CLB_R4C1.S1" ;
INST "U1/FG_5/q1_reg" LOC = "CLB_R3C1.S1" ;
INST "U1/FG_4/q1_reg" LOC = "CLB_R3C1.S1" ;
INST "U1/FG_3/q1_reg" LOC = "CLB_R2C1.S1" ;
INST "U1/FG_2/q1_reg" LOC = "CLB_R2C1.S1" ;
INST "U1/FG_1/q1_reg" LOC = "CLB_R1C1.S1" ;
INST "U1/FG_0/q1_reg" LOC = "CLB_R1C1.S1" ;
INST "U1/FG_12/q2_reg" LOC = "CLB_R7C1.S0" ;
INST "U1/FG_11/q2_reg" LOC = "CLB_R6C1.S0" ;
INST "U1/FG_10/q2_reg" LOC = "CLB_R6C1.S0" ;
INST "U1/FG_9/q2_reg" LOC = "CLB_R5C1.S0" ;
INST "U1/FG_8/q2_reg" LOC = "CLB_R5C1.S0" ;
INST "U1/FG_7/q2_reg" LOC = "CLB_R4C1.S0" ;
INST "U1/FG_6/q2_reg" LOC = "CLB_R4C1.S0" ;
INST "U1/FG_5/q2_reg" LOC = "CLB_R3C1.S0" ;
INST "U1/FG_4/q2_reg" LOC = "CLB_R3C1.S0" ;
INST "U1/FG_3/q2_reg" LOC = "CLB_R2C1.S0" ;
INST "U1/FG_2/q2_reg" LOC = "CLB_R2C1.S0" ;
INST "U1/FG_1/q2_reg" LOC = "CLB_R1C1.S0" ;
INST "U1/FG_0/q2_reg" LOC = "CLB_R1C1.S0" ;
INST "U1/FG_12/q1_reg" LOC = "CLB_R7C1.S1" ;

```

**Figura 8.11:** Ficheiro de restrições para implementação do Registo do Utilizador



**Figura 8.12:** Colocação do Registo do Utilizador dentro do espaço físico da FPGA

Para exemplificar o procedimento de aplicação da metodologia proposta, foi implementado um contador, cuja função consiste em dividir a frequência de 50 MHz, gerada por um cristal presente

na placa utilizada para as experiências práticas, de modo a se obter uma frequência de 0,2 Hz, para que a variação do sinal de saída, quando aplicado a um diodo emissor de luz, fosse perceptível à vista. Adicionalmente, a diferença acentuada entre as velocidades de actualização dos diferentes *flip-flops* que constituem o contador permitiu aferir da eficácia do procedimento de replicação proposto, quer em situações em que as actualizações se processavam muito lentamente (em tempo muito superior ao do processo de relocação, podendo eventualmente ocorrer dentro deste), quer em situações em que as actualizações se processavam muito rapidamente (em tempo muito inferior ao do processo de relocação, acontecendo várias vezes durante o seu decorrer).

A operação de relocação é, sem dúvida, uma das mais críticas em todo este processo, por envolver a manipulação de elementos lógicos durante o seu funcionamento normal. A opção por um contador para exemplo de demonstração baseou-se no facto de se tratar de um circuito em que é fácil perceber a existência de algum problema durante o seu funcionamento, que eventualmente conduzisse à perda de informação de estado, pela simples observação do carácter cíclico da forma de onda de saída. De facto, qualquer erro na transferência de estado conduziria a uma imediata e perceptível alteração desse ciclo. A observação dessa saída foi efectuada através da análise, no osciloscópio, da forma de onda do sinal, à medida que a sequência de ficheiros de reconfiguração eram aplicados ao contador, como se ilustra na Figura 8.13.



**Figura 8.13:** Verificação do correcto funcionamento do contador durante a relocação

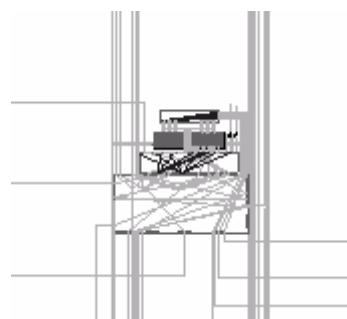
As várias fases da sequência de relocação de um bloco lógico, integrante deste contador, encontram-se ilustradas na Figura 8.14. Repare-se que uma das sequências, a que diz respeito à colocação em paralelo das saídas, não aparece representada nesta sucessão, pela impossibilidade, já apontada anteriormente, de não ser possível visualizar, no *FPGA Editor*, as alterações introduzidas

no ficheiro de configuração através do recurso ao *JBits*. A relocação segue a estratégia de rotação vertical.



**Figura 8.14:** Exemplo de relocação de um dos CLBs do contador

Após a relocação, cada um dos CLBs é sujeito à fase de teste, ilustrada na Figura 8.15, com as saídas do Registo do Utilizador a serem conduzidas em paralelo para as entradas de cada uma das *slices* e as saídas a serem encaminhadas separadamente até células livres do registo BS. A Figura 8.16 mostra o aspecto de uma *slice* configurada com uma das seis configurações de teste.



**Figura 8.15:** CLB sob teste

O pequeno exemplo apresentado, além de servir de demonstração prática da inclusão da metodologia de teste proposta numa FPGA, permite igualmente a sua validação no que diz respeito aos aspectos de implementação e ao seu próprio funcionamento. No entanto, a consideração da sua

validade como alternativa a métodos propostos por outros autores, que foram explicitadas no capítulo 3, fundamenta-se na análise do seu desempenho, nomeadamente através do tempo de latência das faltas, que, se exceptuarmos a cobertura de faltas, questão já analisada no capítulo sete, se mostra como a grandeza mais significativa para a apreciação de qualquer metodologia de teste *on-line*.

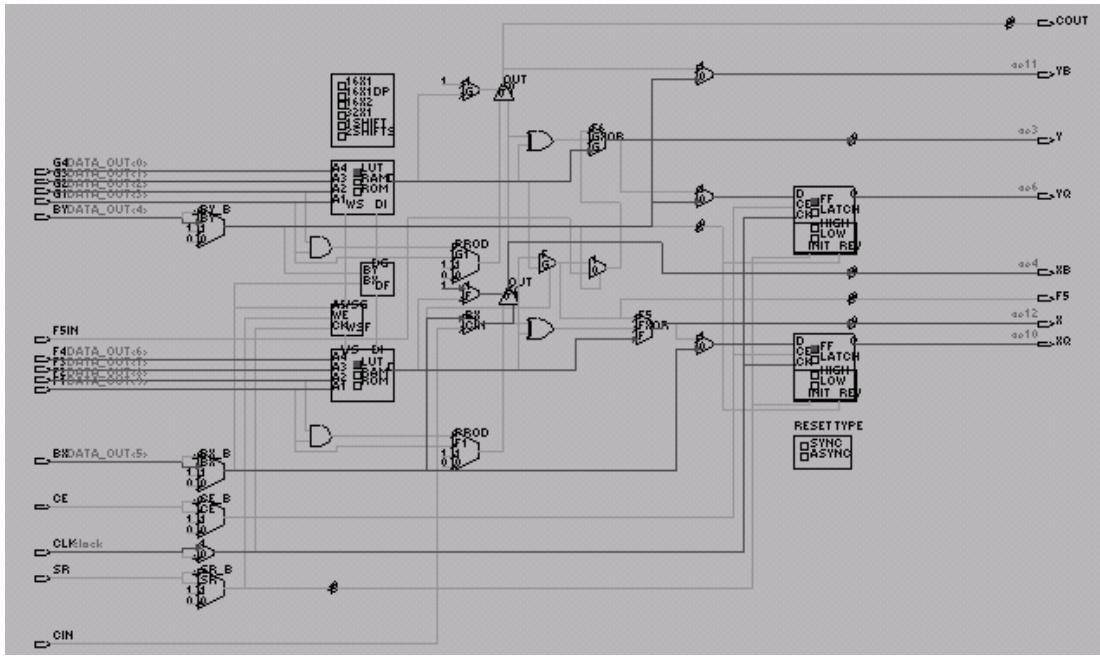


Figura 8.16: Slice com configuração de teste

### 8.3. RESULTADOS TEMPORAIS

A partir da implementação dos circuitos-padrão definidos nas “ITC’99 benchmarks”, apresenta-se um conjunto de médias temporais referentes à aplicação da metodologia proposta, que têm como objectivo fornecer uma percepção do seu desempenho. Desta forma pode aferir-se melhor em que tipos de aplicações a utilização desta metodologia pode ser considerada com vantagem, em alternativa a outras propostas. Os valores publicados apresentam-se todos sob a forma de valores médios, uma vez que a duração temporal de cada uma das três fases, replicação, rotação e teste, está fortemente dependente das próprias aplicações e do índice de ocupação da FPGA.

Os tempos apresentados envolvem quer a consideração do número de impulsos do relógio de teste que é necessário aplicar para controlo da infra-estrutura BS, quer os bytes referentes aos vectores

mudos e às palavras de sincronismo, controlo e mudas, existentes em cada ficheiro de reconfiguração parcial<sup>2</sup>.

A Tabela 8.1 discrimina o número médio de bytes envolvido em cada uma das fases de replicação de um bloco lógico, com o elemento de retenção configurado como *flip-flop* com sinal de relógio bloqueável ou como *latch*, situações que obrigam ao emprego do bloco auxiliar de replicação. Os valores temporais foram obtidos para uma frequência do relógio de teste de 20 MHz, embora, no caso das Virtex, a frequência máxima permitida para TCK seja de 33 MHz. A utilização de uma frequência inferior à máxima prendeu-se apenas com os meios disponíveis para controlar a infra-estrutura BS e não com qualquer limitação imposta pela metodologia adoptada.

**Tabela 8.1:** Tamanho dos vectores e tempo de reconfiguração para cada etapa, na replicação de circuitos sequenciais síncronos com sinal de habilitação de relógio

Replicação com bloco auxiliar de replicação

	Nº de bytes	TCK = 20 MHz Tempo (ms)
Cópia da lógica interna e paralelo das entradas	11 289	9,705
BY_C=1 $\wedge$ CC=1	441	0,379
CC=0	277	0,238
BY_C=0	277	0,238
Colocar em paralelo as entradas CE de ambas as células lógicas	2 145	1,844
Desligar todos os sinais do bloco auxiliar de replicação	2 217	1,906
Colocar em paralelo as saídas de ambas as células lógicas	4 129	3,550
Desligar as saídas do CLB replicado	1 333	1,146
Desligar as entradas do CLB replicado e configurar teste	18 392	15,813
Total	40 500	34,820

<sup>2</sup> Em anexo a esta tese, encontra-se uma descrição mais aprofundada da arquitectura das Virtex, na qual o mecanismo de configuração e a constituição dos vectores são abordados pormenorizadamente.

A Tabela 8.2 indica o número médio de bytes envolvido em cada uma das fases de replicação de um bloco lógico com o elemento de retenção configurado como *flip-flop* com sinal de relógio livre, ou implementando lógica puramente combinatória, situações em que o bloco auxiliar de replicação não é necessário, pelo que o número de fases é menor e, em consequência, é também menor o tempo total de replicação.

**Tabela 8.2:** Tamanho dos vectores e tempo de reconfiguração para cada etapa, na replicação de circuitos sequenciais síncronos com sinal de relógio livre e de circuitos puramente combinatórios

Replicação sem bloco auxiliar de replicação		
	Nº de bytes	TCK = 20 MHz Tempo (ms)
Cópia da lógica interna e paralelo das entradas	12 163	10,457
Colocar em paralelo as saídas de ambas as células lógicas	3 993	3,433
Desligar as saídas do CLB replicado	1 073	0,923
Desligar as entradas do CLB replicado e configurar teste	18 392	15,813
Total	35 621	30,625

Atente-se, por observação da Tabela 8.1 e da Tabela 8.2, que a última etapa da replicação, o desligar das entradas, compreende, no mesmo ficheiro de configuração parcial, a programação da primeira configuração de teste do bloco lógico, bem assim como de toda a infra-estrutura de encaminhamento necessária para a aplicação dos vectores às suas entradas e captura das respostas nas suas saídas. A Tabela 8.3 indica os tempos referentes à configuração das restantes cinco configurações necessárias para se efectuar o teste da totalidade do bloco lógico.

Outra etapa compreendida em cada uma das sessões de teste é o deslocamento para o Registo do Utilizador dos vectores a aplicar ao bloco lógico, em cada uma das configurações de teste. O comprimento desses vectores é igual ao do próprio Registo, treze bits, pelo que o seu deslocamento é, comparativamente, muito rápido, como se dá conta na Tabela 8.4. Já o deslocamento das respostas depende da posição dentro da cadeia das células usadas para a sua captura. Independentemente do comprimento desta, o comprimento do vector a deslocar para o exterior correspondente à distância entre TDO e a célula que contém o último bit da resposta. No pior caso, aquele em que a célula mais próxima de TDI é usada para a captura de um bit da resposta, o

comprimento do vector a deslocar para se recolher todas as respostas é igual ao comprimento do próprio registo BS. No caso do componente usado durante o trabalho experimental, o XCV200, o comprimento desse registo é de 1 022 células, resultando num tempo de deslocamento substancialmente mais elevado quando comparado com o do deslocamento dos vectores a aplicar, como se comprova pela observação da Tabela 8.5.

**Tabela 8.3:** Tamanho dos vectores e tempo de reconfiguração das configurações de teste

Nº da configuração	Nº de bytes	TCK = 20 MHz Tempo (ms)
2 <sup>a</sup>	3 115	2,678
3 <sup>a</sup>	623	0,536
4 <sup>a</sup>	634	0,545
5 <sup>a</sup>	613	0,527
6 <sup>a</sup>	512	0,440
Total	5 497	4,726

**Tabela 8.4:** Tempo de deslocamento dos vectores de teste a aplicar

Nº de vectores	Comprimento (bits)	Total (bits)	TCK = 20 MHz Tempo de aplicação (ms)
40	13	520	0,066

**Tabela 8.5:** Tempo de deslocamento das respostas aos vectores aplicados ao CLB sob teste

Nº células do registo BS no XCV200	Nº de vectores	TCK = 20 MHz Tempo de deslocamento (ms)
1 022	40	4,088

Tendo em consideração a duração temporal de todos os passos que conduzem ao teste de um CLB, é possível determinar um valor médio para a aplicação da metodologia proposta a cada um deles, consoante o seu tipo de ocupação implique a utilização ou não do bloco auxiliar de replicação, ou no caso de este não estar ocupado. Esses valores médios encontram-se listados na Tabela 8.6. A partir destes dados e da análise do tipo de ocupação apresentado pelos circuitos-padrão, pode extrapolar-se um valor médio para o teste da totalidade da matriz lógica de um CLB com 1 176 blocos lógicos, o tamanho do XCV200 utilizado nos testes experimentais.

Segundo o fabricante, a lógica configurável disponível nesse componente equivale a 63 504 portas lógicas. Considerando uma ocupação média de 75% dos blocos lógicos, dos quais um terço requeira a utilização do bloco auxiliar de replicação durante a sua relocação, obtêm-se os valores apresentados na Tabela 8.7, para dois valores diferentes da frequência do relógio de teste: o de 20 MHz, usado durante a experimentação, e o de 33 MHz, valor máximo admissível pelo componente usado (os valores para este caso foram obtidos por simples extrapolação<sup>3</sup>). Estes valores indicam os tempos de latência máximos para as faltas dentro dos blocos lógicos, correspondendo a um limite superior de um intervalo de latência que resulta do movimento de vai-e-vem implementado pela estratégia de rotação, seja ela vertical ou horizontal<sup>4</sup>.

**Tabela 8.6:** Valores médios para o tempo total de teste de um CLB

Tempo total de teste de um CLB com bloco de replicação	43,991 ms
Tempo total de teste de um CLB sem bloco de replicação	39,797 ms
Tempo total de teste de um CLB vazio	24,984 ms

**Tabela 8.7:** Valores para o tempo médio de teste da matriz lógica de uma FPGA

Tempo médio de teste de uma matriz de 1 176 CLBs	
Tipo de ocupação: 25% c/ + 50% s/ + 25% vazio	
43 679,188 ms	TCK = 20 MHz
26 472,235 ms	TCK = 33 MHz

Quanto ao teste das interligações e dos recursos de encaminhamento, não são apresentados tempos, dada a dificuldade da sua extrapolação de condições de laboratório para condições reais de utilização e devido à metodologia proposta não se encontrar ainda automatizada nesta parte. Isso torna inviável retirar, dado o número elevado de configurações que o teste da totalidade desses recursos implica, conclusões seguras, pelo que se optou pela sua não inclusão. De notar, no entanto, que o número elevado de configurações é enganador quanto ao tempo que tal operação demora, uma vez que a relocação de interligações implica ficheiros de reconfiguração parcial bastante mais pequenos e menor número de vectores de teste. Por outro lado, se o seu teste for efectuado intercaladamente com o teste dos CLBs, o tempo de latência das faltas nestes recursos

<sup>3</sup> A situação de 25% dos blocos registados, 50% não registados e 25% vazios, que se apresenta na Tabela 8.7, resulta da análise da ocupação dos recursos durante a implementação dos circuitos-padrão considerados na fase experimental.

<sup>4</sup> Acerca da variação do tempo de latência, ver secção 6.1.4.

será, comparativamente com o tempo de latência no caso dos CLBs, bastante mais elevado. A grande variabilidade dos parâmetros que enformam a relocação e teste destes recursos é outro factor que impede, sem recurso a um grupo exaustivo de implementações práticas, qualquer extração desse teor.

#### 8.4. SUMÁRIO

Neste capítulo, procurou transpor-se para o papel a componente prática que suportou as conclusões teóricas explanadas nos capítulos anteriores e que serviram de base à metodologia proposta para o teste dos blocos lógicos. Descreveram-se as experiências laboratoriais realizadas para suportar as hipóteses teóricas colocadas, relataram-se as dificuldades encontradas, quer a nível de falta de informação, quer de ferramentas de trabalho, e procurou-se, através do exemplo apresentado, expor a totalidade da proposta de um ponto de vista prático. Como corolário, apresentaram-se alguns resultados, que permitem aferir do desempenho e eficácia da aplicação da proposta defendida e ajudar na avaliação, em presença de um caso específico, da sua viabilidade como alternativa a outras soluções de teste.



## **9. CONCLUSÕES E PERSPECTIVAS DE TRABALHO FUTURO**



## 9.1. CONCLUSÕES

Retomando o objectivo expresso na introdução, o trabalho descrito nesta dissertação teve por motivação principal a concepção de uma metodologia vocacionada para o teste estrutural concorrente de dispositivos lógicos programáveis, do tipo FPGA, com capacidade de reconfiguração parcial dinâmica, integrados em sistemas reconfiguráveis. Esta metodologia desenvolve-se em três componentes principais:

- a replicação de circuitos activos;
- a rotação de recursos;
- a aplicação do teste.

Se, por um lado, o objectivo de testar estruturalmente o circuito obriga à sua total disponibilidade para sobre ele ser aplicado o teste; por outro, o propósito de o tornar concorrente, ou seja, de não permitir que a sua aplicação perturbe a operação do sistema, implica que a funcionalidade presente na porção dos recursos a testar seja replicada numa porção livre, sem que o funcionamento do sistema seja afectado por este processo.

Na primeira das três componentes, são alvo de estudo e especificação os mecanismos que permitem, tendo em conta o tipo de recursos ocupados e a funcionalidade neles implementada, a replicação das funções activas. Para vários tipos de circuitos, são concebidos e implementados mecanismos que permitem efectuar a transferência da funcionalidade sem perturbar a sua operação. Após a replicação, os recursos ficam livres para sobre eles poder ser aplicado um teste estrutural.

Todavia, essa replicação, que conduz à transferência da funcionalidade, não está isenta de custos, porque o reencaminhamento das interligações que estabelecem as ligações entre as funções replicadas e o resto do circuito são alteradas e porque a replicação implica a reconfiguração de parte da FPGA e, como tal, a geração e configuração de ficheiros de configuração parcial. O tamanho destes ficheiros influencia somente o tempo de teste e não a operação do sistema, ponto fulcral da metodologia proposta, o mesmo já não se passando, contudo, com o reencaminhamento das interligações. Neste caso, a alteração dos recursos de encaminhamento usados implica uma alteração nos tempos de propagação dos sinais afectados, que poderá perturbar o funcionamento da função. A minimização de ambos os custos, em especial do segundo, é objecto de análise quando se aborda a rotação dos recursos, onde são ponderadas as alternativas para, de forma ordenada, se efectuar a replicação das funções implementadas, de modo a garantir que, minimizando os custos, todos os recursos da FPGA foram sujeitos ao teste ao fim de um determinado intervalo de tempo.

Como se trata de um recurso reconfigurável, o teste estrutural da FPGA exige o uso de várias configurações de teste, cujo número deve ser minimizado para se minimizar o tempo de teste, reduzindo a latência das faltas. Uma análise dos recursos lógicos e um método para a obtenção do número mínimo de configurações de teste são apresentados quando se aborda a parte final, referente à aplicação do teste. É igualmente descrita a geração de vectores a aplicar, de forma a garantir uma cobertura de faltas de 100% sob os modelos de faltas considerados, em função das especificidades dos diferentes tipos de recursos da FPGA. A aplicação dos vectores e a recolha dos resultados são efectuadas através da infra-estrutura *Boundary Scan*, evitando-se, deste modo, o dispêndio extra de pinos e, consequentemente, de recursos a nível da carta de circuito impresso. De notar que também a reconfiguração da FPGA se efectua, no que respeita à replicação e configuração de funções de teste, através desta interface. Consegue-se, assim, implementar uma metodologia de teste estrutural concorrente sem que sejam necessárias alterações a nível da carta de circuito impresso. Os resultados apresentados, fruto de um conjunto de experiências práticas, permitem a validação da metodologia proposta.

## 9.2. PERSPECTIVAS DE TRABALHO FUTURO

Dentro do âmbito desta dissertação e do seu objectivo inicial, alguns pontos merecem continuidade de investigação, no sentido de melhorar e aprofundar as soluções encontradas, bem como incorporar novos elementos que possam aumentar a eficácia e o desempenho da metodologia proposta dentro dos sistemas reconfiguráveis.

O estudo efectuado torna evidente que o teste estrutural concorrente da totalidade das interligações se afigura bastante mais trabalhoso do que o teste dos blocos lógicos, podendo por isso beneficiar de um trabalho mais aprofundado quanto à possibilidade da sua optimização. No entanto, a sua prossecução, com vista a garantir menores tempos de latência e a possibilidade de tornar esse tempo independente em relação à ocupação do dispositivo, não deve pôr em causa os pressupostos iniciais de desenvolvimento de uma metodologia de teste não intrusiva.

De igual modo, o desenvolvimento de uma estratégia de tolerância a faltas adaptada às características da metodologia de teste proposta, nomeadamente o rearranjo das funções implementadas em caso de detecção de uma falta, variável consoante a primitiva afectada e a sua área de influência, e a definição do posicionamento na matriz de CLBs de blocos sobresselentes que permitam a substituição de CLBs que falhem o teste, com o mínimo de custos para o funcionamento das funções afectadas, seria um complemento precioso.

A introdução de um método que, em conjunto com os modelos de faltas considerados nesta proposta, permita incorporar o teste de faltas devidas a atrasos de propagação, com origem numa classe de defeitos cuja importância relativa tem vindo a aumentar, como se comprova pelo surgimento de alguns trabalhos pioneiros neste tema [Krasniewski, 01] e [Abramovici et al., 02], é também exemplo de outro campo a explorar.

De momento, o trabalho a finalizar, ainda no âmbito do Projecto POCTI 33842, referido na introdução, prende-se com a conclusão do desenvolvimento da ferramenta *Virtex PART*, mencionada na secção 8.1, que permite a automatização da implementação da metodologia, tal como definida nesta dissertação.

Uma das contribuições inovadoras desta tese relaciona-se com o método apresentado para a replicação de blocos lógicos activos. Trata-se, efectivamente, de um assunto inédito cuja aplicação extravasa o seu uso no teste, como se prova pelos trabalhos entretanto efectuados na área da desfragmentação do espaço lógico, referidos em [Gericota et al., 02] e [ Gericota et al., 03a]. As FPGAs com capacidade de reconfiguração parcial dinâmica, base das plataformas de computação dinamicamente reconfiguráveis, permitem que múltiplas funções partilhem o espaço tridimensional de configuração (plano dos recursos lógicos ao longo do tempo), sendo implementadas quando necessário e substituídas por outras, quando dispensáveis. Se a sequência de tarefas a realizar pelo sistema não é previsível, as decisões de atribuição de recursos a novas funções, cada uma com os seus requisitos próprios, têm de ser efectuadas em tempo real. À medida que os recursos vão sendo alocados a novas funções e, mais tarde, libertados, pequenas áreas não ocupadas vão sendo criadas. Estas porções de recursos não alocados tendem a tornar-se demasiado pequenas, pelo que não são capazes de *per si* de satisfazerem os requisitos das novas funções, continuando, por isso, desocupadas. Ao fim de várias reconfigurações, o seu número tende a crescer e a área por elas ocupada a tornar-se significativa em relação aos recursos disponíveis. Apesar de, eventualmente, o espaço livre ser suficiente para a implementação de uma nova função, a sua dispersão impede a sua utilização – o espaço lógico de configuração fica fragmentado. É, pois, necessário proceder ao rearranjo das funções implementadas, de forma a conseguir-se espaço contíguo suficiente para a implementação de novas funções. A implementação deste rearranjo obriga à relocação de funções, procedimento que pode ser implementado através dos mecanismos de replicação apresentados nesta dissertação, evitando-se a paragem do sistema para a sua efectivação, com a consequente degradação do desempenho. A continuação do estudo dos mecanismos de gestão do espaço lógico que permitam optimizar o seu uso é um dos temas em aberto e que poderá ter seguimento a partir deste trabalho.

Outro cenário possível, referido na secção 3.2, é o estudo de técnicas de “evolução para a testabilidade”. As FPGAs permitiram concretizar o conceito de *hardware evolutivo*, o que, no entanto, levanta uma outra dificuldade: a de, à partida, não se saber sequer qual a função e os recursos que serão ocupados como resultado da geração evolutiva do circuito. As técnicas de hardware evolutivo conduzem, por vezes, a circuitos “bizarros” que, embora cumprindo os requisitos definidos inicialmente, possuem uma arquitectura cujo funcionamento escapa à compreensão dos projectistas [Thompson et al., 99]. A inexistência de técnicas de “evolução para a testabilidade”, no sentido de que o circuito resultante seja testável, é outra área em aberto.

A tendência que se verifica para uma redução, em termos absolutos, no número de ASICs desenvolvidos anualmente evidencia, não uma diminuição na concepção de novos circuitos para novas utilizações, mas antes uma opção pelo uso cada vez mais frequente de plataformas reconfiguráveis, mais genéricas e consequentemente mais flexíveis e com custos menores. Esta preferência coloca em evidência a importância crescente que as FPGAs têm na indústria electrónica e a importância do incremento do seu estudo nas mais diversas vertentes, o que por sua vez reforça o interesse em prosseguir a investigação iniciada neste trabalho.

## **10. REFERÊNCIAS BIBLIOGRÁFICAS**



- Abramovici, M., Breuer, M. A. and Friedman, A. D. 1990. *Digital Systems Testing and Testable Design*. 1<sup>st</sup> ed. New York: Computer Science Press, 1990. 653 p. ISBN 0-7167-8179-4.
- Abramovici, M., Lee, E. and Stroud, C. E. 1997. BIST-Based Diagnostic of FPGA Logic Blocks. *Proc. of the 3<sup>rd</sup> IEEE International On-Line Testing Workshop*, 1997, p. 196-201.
- Abramovici, M., Lee, E., Stroud, C. E. and Underwood, M. 1997a. Self-test for FPGAs and CPLDs requires no overhead. *Electronic Design News*, November 6, 1997, Vol. 42, No. 23, p. 121-128.
- Abramovici, M., Wijesuriya, S., Hamilton, C. and Stroud, C. E. 1998. Built-In Self-Test for Field Programmable Gate Array Interconnect. *Proc. of the 4<sup>th</sup> IEEE International On-Line Testing Workshop*, 1998, p. 37-41.
- Abramovici, M., Stroud, C. E., Wijesuriya, S., Hamilton, C. and Verma, V. 1999. On-Line Testing and Diagnosis of FPGAs with Roving STARs. *Proc. of the 5<sup>th</sup> IEEE International On-Line Testing Workshop*, 1999, p. 2-7.
- Abramovici, M., Stroud, C. E., Hamilton, C., Wijesuriya, S. and Verma, V. 1999a. Using Roving STARs for On-Line Testing and Diagnosis of FPGAs in Fault-Tolerant Applications. *Proc. of the IEEE International Test Conference*, 1999, p. 973-982.
- Abramovici, M. and Stroud, C. E. 2000. BIST-Based Detection and Diagnosis of Multiple Faults in FPGAs. *Proc. of the IEEE International Test Conference*, 2000, p. 785-794.
- Abramovici, M., Stroud, C. E., Skaggs, B. and Emmert, J. M. 2000a. Improving On-Line BIST-Based Diagnosis for Roving STARs. *Proc. of the 6<sup>th</sup> IEEE International On-Line Testing Workshop*, 2000, p. 31-39.
- Abramovici, M. and Stroud, C. E. 2001. BIST-Based Test and Diagnosis of FPGA Logic Blocks. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, February, 2001, Vol. 9, No. 1, p. 159-172.
- Abramovici, M., Emmert, J. M. and Stroud, C. E. 2001a. Roving STARs: An integrated Approach to On-Line Testing, Diagnosis, and Fault Tolerance for FPGAs in Adaptive Computing Systems. *Proc. of the 3<sup>rd</sup> NASA/DoD Workshop on Evolvable Hardware*, 2001, p. 73-92.
- Abramovici, M. and Stroud, C. E. 2002. BIST-Based Delay-Fault Testing in FPGAs. *Proc. of the 8<sup>th</sup> IEEE International On-Line Testing Workshop*, 2002, p. 131-134.
- Actel, 1999. *Actel Digital Library*. Actel Corporation, 1999. (informação actualizada disponível em <http://www.actel.com>)

- Adálio, A. M. S. and Bampi, S. 1999. Reconfigurable Computing: Viable Applications and Trends. *Proc. of the 10<sup>th</sup> International Conference on Very Large Scale Integration*, 1999, p. 583-594.
- Agarwal, V. K. 1980. Multiple Fault Detection in Programmable Logic Arrays. *IEEE Transactions on Computers*, June, 1980, Vol. C-29, No. 6, p. 518-522.
- Agrawal, V. D., Kime, C. R. and Saluja, K. K. 1993. A Tutorial on Built-In Self-Test - Part 1: Principles. *IEEE Design & Test of Computers*, March, 1993, Vol. 10, No. 1, p. 73-82.
- Agrawal, V. D., Kime, C. R. and Saluja, K. K. 1993a. A Tutorial on Built-In Self-Test - Part 2: Applications. *IEEE Design & Test of Computers*, June, 1993, Vol. 10, No. 2, p. 69-77.
- Ahmed, E. and Rose, J. 2000. The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density. *Proc. of the 8<sup>th</sup> ACM International Symposium on Field-Programmable Gate Arrays*, 2000, p. 3-12.
- Alexandrescu, D., Anghel, L. and Nicolaidis, M. 2002. Simulating Single Event Transients in VDSM ICs for Ground Level Radiation. *3<sup>rd</sup> IEEE Latin-American Test Workshop Digest of Papers*, 2002, p. 126-129.
- Allan, A., Edenfeld, D., Joyner, Jr., W. H., Kahng, A. B., Rodgers, M. and Zorian, Y. 2002. 2001 Technology Roadmap for Semiconductors. *Computer*, January, 2002, Vol. 35, No. 1, p. 42-53.
- Altera, 1998. *Altera Data Book*. Altera Corporation, January, 1998, 885 p. (informação actualizada disponível em <http://www.altera.com>)
- Atmel, 1999. *Atmel CD-ROM Data Book*. Atmel Corporation, April, 1999. (informação actualizada disponível em <http://www.atmel.com>)
- Azimane, M. and Ruiz, A. L. 1998. New Short and Efficient Algorithm for Testing Random-Access Memories. *Proc. of the 5<sup>th</sup> IEEE International Conference on Electronics, Circuits and Systems*, 1998, p. 541-544.
- Betz, V. and Rose, J. 1998. How much logic should go in an FPGA logic block ? *IEEE Design & Test of Computers*, January-March, 1998, Vol. 15, No. 1, p. 10-15.
- Betz, V., Rose, J. and Marquardt, A. 1999. *Architecture and CAD for Deep-Submicron FPGAs*. 1<sup>st</sup> ed. Norwell: Kluwer, 1999. 247 p. ISBN 0-7923-8460-1.
- Betz, V. and Rose, J. 1999a. FPGA Routing Architecture: Segmentation and Buffering to Optimize Speed and Density. *Proc. of the 7<sup>th</sup> ACM International Symposium on Field-Programmable Gate Arrays*, 1999, p. 59-68.

- Betz, V. and Rose, J. 1999b. Circuit Design, Transistor Sizing and Wire Layout of FPGA Interconnect. *Proc. of the IEEE Custom Integrated Circuits Conference*, 1999, p. 171-174.
- Bleeker, H., van den Eijnden, P. and de Jong, F. 1993. *Boundary-Scan Test: A Practical Approach*. 1<sup>st</sup> ed. Norwell: Kluwer, 1993. 225 p. ISBN 0-7923-9296-5.
- Breuer, M. A., Sarrafzadeh, M. and Somenzi, F. 2000. Fundamental CAD Algorithms. *IEEE Transactions on CAD*, December, 2000, Vol. 19, No. 12, p. 1449-1475.
- Brown, S. 1996. FPGA Architectural Research: A survey. *IEEE Design & Test of Computers*, Winter, 1996, Vol. 13, No. 4, p. 9-15.
- Brown, S., Khellah, M. and Vranesic, Z. 1996. Minimizing FPGA Interconnect Delays. *IEEE Design & Test of Computers*, Winter, 1996, Vol. 13, No. 4, p. 16-23.
- Brown, S. and Rose, J. 1996a. FPGA and CPLD Architectures: A Tutorial. *IEEE Design & Test of Computers*, Summer, 1996, Vol. 13, No. 2, p. 42-57.
- Brufaut, M., Depreitere, Jo, Meeus, W., Campenhout, J. V., Melchior, H., Annen, R., Zenklusen, P., Bockstaele, R., Vanwassenhove, L., Hall, J., Neyer, A., Wittmann, B., Heremans, P., Koetsem, J. V., King, R., Thienpont, H. and Baets, R. 1999. A multi-FPGA demonstrator with POF-based optical area interconnect. *Proc. of the 12<sup>th</sup> Annual Meeting of the IEEE Lasers and Electro-Optics Society*, 1999, Vol. 2, p. 625-626.
- Burns, J., Donlin, A., Hogg, J., Singh, S. and Wit, M. de 1997. A Dynamic Reconfiguration Run-Time System. *Proc. of the 5<sup>th</sup> Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, 1997, p. 66-75.
- Burress, A. L. and Lala, P. K. 1997. On-Line Testable Logic Design for FPGA Implementation. *Proc. of the IEEE International Test Conference*, 1997, p. 471-478.
- Bursky, D. 1996. Efficient RAM-based FPGAs Ease System Design. *Electronic Design*, January 22, 1996, Vol. 44, No. 2, p. 53-62.
- Bursky, D. 1997. High-Density FPGA Family delivers Megagate Capacity. *Electronic Design*, November 17, 1997, Vol. 45, No. 22, p. 67-69.
- Bursky, D. 2002. Reprogrammable Logic: From Standard Logic Replacement to Unique Solutions. *Electronic Design*, January 7, 2002, Vol. 50, No. 1, p. 42-43.
- Bushnell, M. L. and Agrawal, V. D. 2000. *Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits*. 1<sup>st</sup> ed. Boston: Kluwer, 2000. 690 p. ISBN 0-7923-799-1-8.

- Campenhout, J. V., Marck, H. V., Depreitere, J. and Dambre, J. 1999. Optoelectronic FPGA's. *IEEE Journal of Selected Topics in Quantum Electronics*, March-April, 1999, Vol. 5, No. 2, p. 306-315.
- Cantó, E., Moreno, J. M., Cabestany, J., Lacadena, I. and Insenser, J. M. 2001. A Temporal Bipartitioning Algorithm for Dynamically Reconfigurable FPGAs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, February, 2001, Vol. 9, No. 1, p. 210-218.
- Cardoso, J. M. P. and Neto, H. C. 1999. An Enhanced Static-List Scheduling Algorithm for Temporal Partitioning onto RPUs. *Proc. of the 10<sup>th</sup> International Conference on Very Large Scale Integration*, 1999, p. 485-496.
- Carter, W. S. 1991. The Evolution of Programmable Logic. *Symposium on VLSI Circuits Digest of Technical Papers*, 1991, p. 43-46.
- Chiricescu, S., Leeser, M. and Vai, M. M. 2001. Design and Analysis of a Dynamically Reconfigurable Three-Dimensional FPGA. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, February, 2001, Vol. 9, No. 1, p. 186-196.
- Chow, P., Seo, S. O., Rose, J., Chung, K., Páez-Monzón, G. and Rahardja, I. 1999. The Design of an SRAM-Based Field-Programmable Gate Array, Part I: Architecture. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, June, 1999, Vol. 7, No. 2, p. 191-197.
- Chow, P., Seo, S. O., Rose, J., Chung, K., Páez-Monzón, G. and Rahardja, I. 1999a. The Design of an SRAM-Based Field-Programmable Gate Array, Part II: Circuit Design and Layout. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, September, 1999, Vol. 7, No. 3, p. 321-330.
- Compton, K. 1999. Programming Architectures For Run-Time Reconfigurable Systems. *Master's Thesis*, Northwestern University, IL, USA, December, 1999, 33 p. (disponível em <http://www.ece.northwestern.edu/~kat/>)
- Compton, K. and Hauck, S. 2002. Reconfigurable Computing: A Survey of Systems and Software. *ACM Computing Surveys*, June, 2002, Vol. 34, No. 2, p. 171-210.
- Cong, J., Lei, H., Cheng-Kok, K. and Madden, P. H. 1996. Performance optimization of VLSI interconnect layout. *Integration, The VLSI journal*, November 1, 1996, Vol. 21, No. 1-2, p. 1-94.

- Constancias, C. 1998. Emission d'électrons par effet de champ à partir de micropointes pour écrans plats : Simulations, caractérisations et confrontations expérimentales. *Thèse soutenu pour obtenir le titre de Docteur (spécialité : Physique microélectronique)*, Université Joseph Fourier - Grenoble I, France, Janvier, 1998. (disponível em <http://constancias.chez.tiscal.fr/>)
- Corno, F., Reorda, M. S. and Squillero, G. 2000. RT-level ITC'99 benchmarks and first ATPG results. *IEEE Design and Test of Computers*, July-September, 2000, Vol. 17, No. 3, p. 44-53.
- Crouch, A. L. 1999. *Design-for-Test for Digital IC's and Embedded Core Systems*. 1<sup>st</sup> ed. Upper Saddle River: Prentice Hall, 1999. 349 p. ISBN 0-13-084827-1.
- Darwin, C. 1859. *On the Origin of Species by Means of Natural Selection or the Preservation of Favoured Races in Struggle for Life*. 1<sup>st</sup> ed. London: John Murray, 1859.
- David, R., Chillet, D., Pillement, S. and Sentieys, O. 2002. A Compilation Framework for a Dynamically Reconfigurable Architecture. *Proc. of the 12<sup>th</sup> International Workshop on Field-Programmable Logic and Applications*, 2002, p. 1058-1067.
- DeHon, A. 1994. DPGA-Coupled Microprocessors: Commodity ICs for the early 21<sup>st</sup> century. *Proc. of the IEEE Workshop on FPGA's for Custom Computing Machines*, 1994, p. 31-39.
- Dekker, R., Beenker, F. and Thijssen, L. 1988. Fault Modeling and Test Algorithm Development for Static Random Access Memories. *Proc. of the IEEE International Test Conference*, 1988, p. 343-352.
- Diessel, O., ElGindy, H., Middendorf, M., Schmeck, H. and Schmidt, B. 2000. Dynamic scheduling of tasks on partially reconfigurable FPGAs. *IEE Proc.-Computer Digital Technology*, May, 2000, Vol. 147, No. 3, p. 181-188.
- Doumar, A., and Ito, H. 1999. Design of an automatic testing for FPGAs. *Proc. of the 4<sup>th</sup> IEEE European Test Workshop*, 1999, p. 152-157.
- Doumar, A., Ohmameuda, T. and Ito, H. 1999a. Testing the Logic Cells and Interconnect Resources for FPGAs. *Proc. of the 8<sup>th</sup> Asian Test Symposium*, 1999, p. 369-374.
- Dufaza, C. 1998. Theoretical properties of LFSRs for built-in self test. *Integration, The VLSI journal*, September 1, 1998, Vol. 25, No. 1, p. 17-35.
- Duncan, A. A., Hendry, D. C. and Gray, P. 2001. The CORBA-ABS High-Level Synthesis System for Multi-FPGA Custom Computing Machines. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, February, 2001, Vol. 9, No. 1, p. 218-223.

- El Gamal, A., Greene, J., Reyneri, J., Rogoyski, E., El-Ayat, K. and Mohsen, A. 1988. An Architecture for Electrically Configurable Gate Arrays. *Proc. of the IEEE Custom Integrated Circuits Conference*, 1988, p. 15.4.1-15.4.4.
- Eldred, R. D. 1959. Test Routines Based on Symbolic Logical Statements. *Journal of the ACM*, January, 1959, Vol. 6, No. 1, p. 33-37.
- Emmert, J. M., Stroud, C. E., Skaggs, B. and Abramovici, M. 2000. Dynamic Fault Tolerance in FPGAs via Partial Reconfiguration. *Proc. of the 8<sup>th</sup> IEEE Symposium on Field-Programmable Custom Computing Machines*, 2000, p. 165-174.
- Emmert, J. M., Stroud, C. E., Cheatham, J., Taylor, A. M., Kataria, P. and Abramovici, M. 2000a. Performance Penalty for Fault Tolerance in Roving STARs. *Proc. of the 10<sup>th</sup> Int. Conf. on Field-Programmable Logic and Applications*, 2000, p. 545-554.
- Estrin, G., Bussel, B. and Turn, R. 1963. Parallel Processing in a Restructurable Computer System. *IEEE Transactions on Electronic Computers*, December, 1963, Vol. EC-12, No. 5, p. 747-755.
- Fawcett, B. K., 1994. Taking Advantage of Reconfigurable Logic. *Proc. of the 2<sup>nd</sup> ACM International Workshop on FPGAs*, 1994.
- 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 submetida para obtenção do grau de Doutor*, Faculdade de Engenharia da Universidade do Porto, Abril, 1992, 322 p.
- Ferreira, J. P. C. 1992a. O teste funcional de implementações da norma IEEE 1149.1 - 1990. *Trabalho de síntese submetido a provas de Aptidão Pedagógica e Capacidade Científica*, Faculdade de Engenharia da Universidade do Porto, Agosto, 1992, 102 p.
- Ferreira, J. M., Alves, G. R., Gericota, M. G. and Silva, M. L. 2002. Teste Concorrente para Sistemas Electrónicos Reconfiguráveis (baseados em FPGAs com capacidade de reconfiguração parcial dinâmica). *Relatório intercalar de actividades - Projecto POCTI/33842/ESE/2000*, Faculdade de Engenharia da Universidade do Porto, Abril, 2002, 100 p.
- Fujiwara, H. and Kinoshita, K. 1981. A Design of Programmable Logic Arrays with Universal Tests. *IEEE Transactions on Computers*, November, 1981, Vol. C-30, No. 11, p. 823-828.
- Fujiwara, H. 1984. A New PLA Design for Universal Testability. *IEEE Transactions on Computers*, August, 1984, Vol. C-33, No. 8, p. 745-750.

- Gehring, S. and Ludwig, S. 1998. Fast Integrated Tools for Circuit Design with FPGAs. *Proc. of the 6<sup>th</sup> ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 1998, p. 133-139.
- Gericota, M. G., Alves, G. R., Silva, M. L. and Ferreira, J. M. 2000. The RaT technique for concurrent test of dynamically reconfigurable hardware. *Proc. of the 15<sup>th</sup> Conference on Design of Circuits and Integrated Systems*, 2000, p. 337-340.
- Gericota, M. G., Alves, G. R., Silva, M. L. and Ferreira, J. M. 2001. DRAFT: An On-Line Fault Detection Method for Dynamic and Partially Reconfigurable FPGAs, *Proc. of the 7<sup>th</sup> IEEE On-Line Testing Workshop*, 2001, p. 34-36.
- Gericota, M. G., Alves, G. R., Silva, M. L. and Ferreira, J. M. 2001a. DRAFT: An On-line Concurrent Test for Partial and Dynamically Reconfigurable FPGAs. *Proc. of the 16<sup>th</sup> Conference on Design of Circuits and Integrated Systems*, 2001, p. 553-558.
- Gericota, M. G., Alves, G. R., Silva, M. L. and Ferreira, J. M. 2001b. Dynamically Rotate And Free for Test: The Path for FPGA Concurrent Test, *2<sup>nd</sup> IEEE Latin-American Test Workshop Digest of Papers*, 2001, p. 180-185.
- Gericota, M. G., Alves, G. R., Silva, M. L. and Ferreira, J. M. 2001c. DRAFT: A Scanning Test Methodology for Dynamic and Partially Reconfigurable FPGAs, *6<sup>th</sup> IEEE European Test Workshop Informal Digest*, 2001, p. 113-115.
- Gericota, M. G., Alves, G. R., Silva, M. L. and Ferreira, J. M. 2002. On-line Defragmentation for Run-Time Partially Reconfigurable FPGAs. *Proc. of the 12<sup>th</sup> International Workshop on Field-Programmable Logic and Applications*, 2002, p. 302-311.
- Gericota, M. G., Alves, G. R., Silva, M. L. and Ferreira, J. M. 2002a. Dynamic Replication: The Core of a Truly Non-Intrusive SRAM-based FPGA Structural Concurrent Test Methodology. *3<sup>rd</sup> IEEE Latin-American Test Workshop Digest of Papers*, 2002, p. 70-75.
- Gericota, M. G., Alves, G. R., Silva, M. L. and Ferreira, J. M. 2002b. A Novel Methodology for the Concurrent Test of Partial and Dynamically Reconfigurable SRAM-based FPGAs, *Proc. of the IEEE International Conference on Design, Automation and Test in Europe*, 2002, p. 1126.
- Gericota, M. G., Alves, G. R., Silva, M. L. and Ferreira, J. M. 2002c. On-line Testing of FPGA Logic Blocks Using Active Replication, *Proc. of the Norsk Informatikkonferanse*, 2002, p. 167-178.

- Gericota, M. G., Alves, G. R., Silva, M. L. and Ferreira, J. M. 2002d. Active Replication: Towards a Truly SRAM-based FPGA On-Line Concurrent Testing, *Proc. of the 8<sup>th</sup> IEEE On-Line Testing Workshop*, 2002, p. 165-169.
- Gericota, M. G., Alves, G. R., Silva, M. L. and Ferreira, J. M. 2002e. AR<sup>2</sup>T: Implementing a Truly SRAM-based FPGA On-Line Concurrent Testing, *7<sup>th</sup> IEEE European Test Workshop Informal Digest*, 2002, p. 61-66.
- Gericota, M. G., Alves, G. R., Silva, M. L. and Ferreira, J. M. 2003. Programmable Logic Devices: A Test Approach for the Input/Output Blocks and Pad-to-Pin Interconnections. *4<sup>th</sup> IEEE Latin-American Test Workshop Digest of Papers*, 2003, p. 72-77.
- Gericota, M. G., Alves, G. R., Silva, M. L. and Ferreira, J. M. 2003a. Run-Time Management of Logic Resources on Reconfigurable Systems. *Proc. of the IEEE International Conference on Design, Automation and Test in Europe*, 2003, p. 974-979.
- Ghosh, I. and Jha, N. K. 1998. High-level test synthesis: a survey. *Integration, The VLSI journal*, December 1, 1998, Vol. 26, No. 1-2, p. 79-99.
- Gibson, G., Gray, L. and Stroud, C. 1997. Boundary Scan Access of Built-In Self-Test for Field Programmable Gate Arrays. *Proc. of the 10<sup>th</sup> Annual IEEE International ASIC Conference*, 1997, p. 57-61.
- Gibson, D. J., Vasilko, M. and Long, D. 1998. Virtual Prototyping for Dynamically Reconfigurable Architectures using Dynamic Generic Mapping. *Proc. of the VHDL International Users Forum*, 1998.
- Greene, J., Hamdy, E. and Beal, S. 1993. Antifuse Field Programmable Gate Arrays. *Proceedings of the IEEE*, July, 1993, Vol. 81, No. 7, p. 1042-1056.
- Guccione, S. A., Levi, D. and Sundararajan, P. 1999. JBits: Java based interface for reconfigurable computing. *Proc. of the 2<sup>nd</sup> Annual Military and Aerospace Applications of Programmable Devices and Technologies Conference*, 1999.
- Haase, A., Kretzschmar, C., Siegmund, R., Müller, D., Schneider, J., Boden, M. and Langer, M., 2002. Design of a Reed Solomon Decoder using Partial Dynamic Reconfiguration of Xilinx Virtex FPGAs - A Case Study. *Proc. of the IEEE International Conference on Design, Automation and Test in Europe - Designers' Forum*, 2002, p. 151-157.
- Hanchek, F. and Dutt, S. 1998. Methodologies for Tolerant Cell and Interconnect Faults in FPGAs. *IEEE Transactions on Computers*, January, 1998, Vol. 47, No. 1, p. 15-33.

- Harkin, J., McGinnity, T. M. and Maguire, L. P. 2001. Genetic algorithm driven hardware – software partitioning for dynamically reconfigurable embedded systems. *Microprocessors and Microsystems*, August 20, 2001, Vol. 25, No. 5, p. 263-274.
- Hastie, N. and Cliff, R. 1990. The Implementation of Hardware Subroutines on Field Programmable Gate Arrays. *Proc. of the IEEE Custom Integrated Circuits Conference*, 1990, p. 31.4.1-31.4.4.
- Hauck, S. and Borriello, G. 1995. Logic Partition Orderings for Multi-FPGA Systems. *Proc. of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 1995, p. 32-38.
- Hauck, S. 1998. The Roles of FPGA's in Reprogrammable Systems. *Proceedings of the IEEE*, April, 1998, Vol. 86, No. 4, p. 615-638.
- Hayes, J. P. 1985. Fault Modeling. *IEEE Design & Test of Computers*, April, 1985, Vol. 2, No. 2, p. 88-95.
- He, J. and Rose, J. 1993. Advantages of Heterogeneous Logic Block Architectures for FPGAs. *Proc. of the IEEE Custom Integrated Circuits Conference*, 1993, p. 7.4.1-7.4.5.
- Higuchi, T. and Kajihara, N. 1999. Evolvable Hardware Chips for Industrial Applications: Autonomous, dynamic, and suitable for a variety of real-world applications. *Communications of the ACM*, April, 1999, Vol. 42, No. 4, p. 60-66.
- Horta, E. L., Lockwood, J. W. and Kofuji, S. T. 2002. Using PARBIT to Implement Partial Run-Time Reconfigurable Systems. *Proc. of the 12<sup>th</sup> International Workshop on Field-Programmable Logic and Applications*, 2002, p. 182-191.
- Hsieh, H.-C., Dong, K., Ja, J. Y., Kanazawa, R., Ngo, L. T., Tinkey, L. G., Carter, W. S. and Freeman, R. H. 1988. A 9000-Gate Array User-Programmable Gate Array. *Proc. of the IEEE Custom Integrated Circuits Conference*, 1988, p. 15.3.1-15.3.7.
- Hsieh, H.-C., Carter, W. S., Ja, J. Y., Cheung, E., Schreifels, S., Erickson, C., Freidin, P., Tinkey, L. G. and Kanazawa, R. 1990. Third-Generation Architecture Boosts Speed and Density of Field-Programmable Gate Arrays. *Proc. of the IEEE Custom Integrated Circuits Conference*, 1990, p. 31.2.1-31.2.7.
- Huang, W. K., Meyer, F. J. and Lombardi, F. 1996. Array-Based Testing of FPGAs: Architecture and Complexity. *Proc. of the 8<sup>th</sup> Annual IEEE International Conference on Innovative Systems in Silicon*, 1996, p. 249-258.

- Huang, W. K. and Lombardi, F. 1996a. An Approach for Testing Programmable / Configurable Field Programmable Gate Arrays. *Proc. of the 14<sup>th</sup> IEEE VLSI Test Symposium*, 1996, p. 450-455.
- Huang, W. K., Meyer, F. J., Chen, Xiao-Tao and Lombardi, F. 1998. Testing Configurable LUT-Based FPGA's. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, June, 1998, Vol. 6, No. 2, p. 276-283.
- Huang, W. K., Meyer, F. J. and Lombardi, F. 2000. An approach for detecting multiple faulty FPGA logic blocks. *IEEE Transactions on Computers*, January, 2000, Vol. C-49, No. 1, p. 48-54.
- Huang, Wei-Je and McCluskey, E. J. 2001. A Memory Coherence Technique for Online Transient Error Recovery of FPGA Configurations. *Proc. of the 9<sup>th</sup> ACM Int. Symposium on Field-Programmable Gate Arrays*, 2001, p. 183-192.
- Hughes, J. L. A. and McCluskey, E. J. 1984. An analysis of the multiple fault detection capabilities of single stuck-at fault test sets. *Proc. of the IEEE International Test Conference*, 1984, p. 52-58.
- Hughes, J. L. A. 1988. Multiple Fault Detection Using Single Fault Test Sets. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, January, 1988, Vol. 7, No. 1, p. 100-108.
- IEEE Std 1076 2002. *IEEE Standard VHDL Language Reference Manual*. New York: IEEE Standards Board, May, 2002, 309 p. ISBN 0-7381-3247-0.
- IEEE Std 1149.1 2001. *IEEE Standard Test Access Port and Boundary-Scan Architecture*. New York: IEEE Standards Board, July, 2001, 208 p. ISBN 0-7381-2945-3.
- IEEE Std 1364 2001. *IEEE Standard Verilog® Hardware Description Language*. New York: IEEE Standards Board, September, 2001, 791 p. ISBN 0-7381-2826-0.
- IEEE Std 1532 2001. *IEEE Standard for In-System Configuration of Programmable Devices*. New York: IEEE Standards Board, December, 2001, 139 p. ISBN 0-7381-3102-4.
- Inoue, T., Fujiwara, H., Michinishi, H., Yokohira, T. and Okamoto, T. 1995. Universal Test Complexity of Field-Programmable Gate Arrays. *Proc. of the 4<sup>th</sup> IEEE Asian Test Symposium*, 1995, p. 259-267.
- Inoue, T., Miyazaki, S. and Fujiwara, H. 1997. On the complexity of Universal Fault Diagnosis for Look-up Table FPGAs. *Proc. of the 6<sup>th</sup> IEEE Asian Test Symposium*, 1997, p. 276-281.

- Inoue, T., Miyazaki, S. and Fujiwara, H. 1998. Universal Fault Diagnosis for Look-up Table FPGAs. *IEEE Design & Test of Computers*, January-March, 1998, Vol. 15, No. 1, p. 39-44.
- Kautz, W. H. 1974. Testing for Faults in Wiring Networks. *IEEE Transactions on Computers*, April, 1974, Vol. C-23, No. 4, p. 358-363.
- Kaviani, A. and Brown, S. 1996. Hybrid FPGA Architecture. *Proc. of the 4<sup>th</sup> International Symposium on Field-Programmable Gate Arrays*, 1996, 7 p.
- Kaviani, A. and Brown, S. 1998. Efficient Implementation of Array Multipliers in FPGAs. *Proc. of the 5<sup>th</sup> Canadian Workshop on Field-Programmable Devices*, 1998, 6 p.
- Kaviani, A. and Brown, S. 1999. The Hybrid Field-Programmable Architecture. *IEEE Design & Test of Computers*, April-June, 1999, Vol. 16, No. 2, p. 74-83.
- Klein, B. 1994. Use LFSRs to build fast FPGA-based counters. *Electronic Design*, March 21, 1994, Vol. 42, No. 6, p. 87-100.
- Köster, M. and Teich, J. 2002. (Self-)reconfigurable Finite State Machines: Theory and Implementation. *Proc. of the IEEE International Conference on Design, Automation and Test in Europe - Designers' Forum*, 2002, p. 559-566.
- Krasniewski, A. 1999. Application-Dependent Testing of FPGA Delay Faults. *Proc. of the 25<sup>th</sup> EUROMICRO Conference*, 1999, Vol. 1, p. 260-267.
- Krasniewski, A. 2000. Self-Testing of FPGA Delay Faults in the System Environment. *Proc. of the 6<sup>th</sup> IEEE International On-Line Testing Workshop*, 2000, p. 40-41.
- Krasniewski, A. 2001. Evaluation of Delay Fault Testability of LUT Functions for Improved Efficiency of FPGA Testing. *Proc. of the Euromicro Symposium on Digital Systems Design*, 2001, p. 310-317.
- Krupnova, H. and Saucier, G. 2000. FPGA technology snapshot: Current devices and design tools. *Proc. of the 11<sup>th</sup> IEEE International Workshop on Rapid Systems Prototyping*, 2000, p. 200-205.
- Lach, J., Mangione-Smith, W. H. and Potkonjak, M. 1998. Low Overhead Fault-Tolerant FPGA Systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, June, 1998, Vol. 6, No. 2, p. 212-221.
- Lala, P. K. 2001. *Self-Checking and Fault-Tolerant Digital Design*. 1<sup>st</sup> ed. San Francisco: Morgan Kaufmann, 2001. 216 p. ISBN 0-12-434370-8.
- Laplante, P. A. 1998. Comprehensive Electrical Engineering Dictionary. 1<sup>st</sup> ed. Boca Raton: CRC Press LLC, 1998. 1536 p. ISBN 0-84-933128-5.

Lattice, 2002. ORCA Series 2 Data Sheet. Lattice Semiconductor Corp., August, 2002. 196 p.  
(informação actualizada disponível em <http://www.latticesemi.com>)

Lattice, 2002a ORCA Series 4 Data Sheet. Lattice Semiconductor Corp., April, 2002. 151 p.  
(informação actualizada disponível em <http://www.latticesemi.com>)

Lau, N. and Sklyarov, V. 1999. Dynamically Reconfigurable Implementation of Control Circuits.  
*Proc. of the 10<sup>th</sup> International Conference on Very Large Scale Integration*, 1999, p. 137-148.

Lauwereins, R. 2002. Creating a World of Smart Re-configurable Devices. *Proc. of the 12<sup>th</sup> International Workshop on Field-Programmable Logic and Applications*, 2002, p. 790-794.

Lee, S., Cobb, B., Dworak, J., Grimalia, M. R. and Mercer, M. R. 2002. A New ATPG Algorithm to Limit Test Set Size and Achieve Multiple Detections of all Faults. *Proc. of the IEEE International Conference on Design, Automation and Test in Europe*, 2002, p. 94-99.

Leeser, M., Meleis, W. M., Vai, M. M., Chiricescu, S., Xu, W. and Zavracky, P. M. 1998. Rothko: a three-dimensional FPGA. *IEEE Design & Test of Computers*, January-March, 1998, Vol. 15, No. 1, p. 16-23.

Levi, D. and Guccione, S. A. 1999. GeneticFPGA: A Java-Based Tool for Evolving Stable Circuits. *Reconfigurable Technology: FPGAs for Computing and Applications, Proc. of the Society of Photo-Optical Instrumentation Engineers (SPIE)*, 1999, Vol. 3844, p. 114-121.

Lighthart, M. M. and Stans, R. J. 1991. A Fault Model for PLA's. *IEEE Transactions on CAD*, February, 1991, Vol. 10, No. 2, p. 265-270.

Lima, F., Carro, L., Velazco, R. and Reis, R. 2002. Injecting Multiple Upsets in a SEU tolerant 8051 Micro-controller. *3<sup>rd</sup> IEEE Latin-American Test Workshop Digest of Papers*, 2002, p. 120-125.

Liu, D. L. and McCluskey, E. J. 1988. Design of Large Embedded CMOS PLA's for Built-In Self-Test, *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, January, 1988, Vol. 7, No. 1, p. 50-59.

Liu, T., Huang, W. K., Meyer, F. J. and Lombardi, F. 2000. Testing and Testable Designs for One-Time Programmable FPGAs. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, November, 2000, Vol. 19, No. 11, p. 1370-1375.

López, C., Riesgo, T., Torroja, Y., Uceda, J. and Entrena, L. 2000. Application of Fault Simulation Techniques to Design Validation Quality Measurements. *Proc. of the 15<sup>th</sup> Design of Circuits and Integrated Systems Conference*, 2000, p. 415-420.

- Lu, C.-W., Lee, C. L., Su, C. and Chen, J.-E, 2002. Analysis of Application of the IDDQ Technique to the Deep Sub-Micron VLSI Testing. *Journal of Electronic Testing: Theory and Applications*, February, 2002, Vol. 18, No. 1, p. 89-97.
- Maliniak, L. 1995. Can FPGA design be device independent? *Electronic Design*, January 23, 1995, Vol. 43, No. 2, p. 41-51.
- Marchal, P. 1999. Field Programmable Gate Arrays. *Communications of the ACM*, April, 1999, Vol. 42, No. 4, p. 57-59.
- Marck, H. V., Depreitere, J., Stroobandt, D. and Campenhout, J. V. 1998. A quantitative study of the benefits of area-I/O in FPGAs. *Proc. of the 8<sup>th</sup> Great Lakes Symposium on VLSI*, 1998, p. 392-399.
- Marquardt, A., Betz, V. and Rose, J. 1999. Using Cluster-based Logic Blocks and Timing-Driven Packing to Improve FPGA Speed and Density. *Proc. of the 7<sup>th</sup> ACM International Symposium on Field-Programmable Gate Arrays*, 1999, p. 37-46.
- Marquardt, A., Betz, V. and Rose, J. 2000. Speed and Area Tradeoffs in Cluster-Based FPGA Architectures. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, February, 2000, Vol. 8, No. 1, p. 84-93.
- Maunder, C. M. and Tulloss, R. E. 1990. *The Test Access Port and Boundary-Scan Architecture*. 1<sup>st</sup> ed. New York: IEEE Computer Society Press, 1990. 372 p. ISBN 0-8186-9070-4.
- Maunder, C. M. and Tulloss, R. E. 1991. An Introduction to the Boundary Scan Standard: ANSI/IEEE Std 1149.1. *Journal of Electronic Testing: Theory and Applications*, March, 1991, Vol. 2, No. 1, p. 27-42.
- Maxfield, C. 1996. Logic that mutates while-u-wait. *Electronic Design News*, November 7, 1996, Vol. 41, No. 23, p. 137-142.
- Mazumder, P. and Chakraborty, K. 1996. *Testing and Testable Design of High-Density Random-Access Memories*. 1<sup>st</sup> ed. Boston: Kluwer, 1996. 386 p. ISBN 0-7923-9782-7.
- McCluskey, E. J. 1984. Verification Testing - A Pseudoexhaustive Test Technique. *IEEE Transactions on Computers*, June, 1984, Vol C-33, No. 6, p. 541-546.
- McCluskey, E. J. 1985. Built-In Self-Test Techniques. *IEEE Design & Test of Computers*, April, 1985, Vol. 2, No. 2, p. 21-28.
- McCluskey, E. J. 1985a. Built-In Self-Test Structures. *IEEE Design & Test of Computers*, April, 1985, Vol. 2, No. 2, p. 29-36.

- McCluskey, E. J. 2000. Why defects escape some of our tests. *Proc. of the IEEE International Test Conference*, 2000, p. 1125.
- McCluskey, E. J. and Tseng, C.-W. 2000. Stuck-Fault Tests vs. Actual Defects. *Proc. of the IEEE International Test Conference*, 2000, p. 336-343.
- Merino, J. L. and Bota, S. A. 2000. Reconfigurable SRAM-Based FPGAs: BIST Testing Applications. *Proc. of the XV Design of Circuits and Integrated Systems Conference*, 2000, p. 341-345.
- Metra, C., Mojoli, G., Pastore, S., Salvi, D. and Sechi, G. R. 1998. Novel Technique for Testing FPGAs. *Proc. of the IEEE International Conference on Design, Automation and Test in Europe*, 1998, p. 89-94.
- Metra, C., Pagano, A. and Riccò, B. 2001. On-Line Testing of Transient and Crosstalk Faults Affecting Interconnections of FPGA-Implemented Systems. *Proc. of the IEEE International Test Conference*, 2001, p. 939-947.
- Mitra, S., Shirvani, P. P. and McCluskey, E. J. 1998. Fault Location in FPGA-Based Reconfigurable Systems. *Proc. of the IEEE International High Level Design Validation and Test Workshop*, 1998, p. 143-150.
- Moore, G. E. 1965. Cramming more components onto integrated circuits. *Electronics*, April 19, 1965, Vol. 38, No. 8, 4 p.
- Murgai, R. and Fujita, M. 1999. Speeding up Look-up Table Driven Logic Simulation. *Proc. of the 10<sup>th</sup> International Conference on Very Large Scale Integration*, 1999, p. 385-397.
- Needham, W., Prunty, C. and Yeoh, H. E. 1998. High volume microprocessor test escapes, an analysis of defects our tests are missing. *Proc. of the IEEE International Test Conference*, 1998, p. 25-34.
- Nicolaidis, M. and Zorian Y. 1999. Scaling Deeper to Submicron: On-Line Testing to the Rescue. *Proc. of the IEEE International Conference on Design, Automation and Test in Europe*, 1999, p. 432.
- Otten, R. H. J. M., Camposano, R. and Groeneveld, P. R. 2002. Design Automation for Deepsubmicron: present and future. *Proc. of the IEEE International Conference on Design, Automation and Test in Europe*, 2002, p. 650-657.

- Papachristou, C. A. and Sahgal, N. B. 1985. An Improved Method for Detecting Functional Faults in Semiconductor Random Access Memories. *IEEE Transactions on Computers*, February, 1985, Vol. 34, No. 2, p. 110-116.
- Park, J., Mercer, M. R., Naivar, M., Williams, T. W. and Kapur, R. 1994. Limitations in predicting defect level based on stuck-at fault coverage. *Proc. of the 12<sup>th</sup> IEEE VLSI Test Symposium*, 1994, p. 186-191.
- Parker, K. P. 1992. *The Boundary-Scan Handbook*. 1<sup>st</sup> ed. Norwell: Kluwer, 1992. 262 p. ISBN 0-7923-9270-1.
- Politécnico di Torino, 1999. *Torino ITC'99 benchmarks*, 1999. (descrição dos circuitos disponível em <http://www.cad.polito.it/tools/itc99.html>)
- Porrmann, M., Witkowski, U., Kalte, H. and Rückert, U. 2002. Dynamically Reconfigurable Hardware – A New Perspective for Neural Network Implementations. *Proc. of the 12<sup>th</sup> International Workshop on Field-Programmable Logic and Applications*, 2002, p. 1048-1057.
- Portal, J.-M. 1999. Test des Circuits Configurables de Type FPGA à Base de SRAM. *Thèse présentée à l'Université de Montpellier II Sciences et Techniques du Languedoc pour obtenir le diplôme de Doctorat*, Université de Montpellier II, France, Mai, 1999.
- Powell, T. J., Butler, K. M., Ales, M., Haley, R. and Perry, M. 1994. Correlating defect level to final test fault coverage for modular structured designs. *Proc. of the 12<sup>th</sup> IEEE VLSI Test Symposium*, p. 192-196.
- Renovell, M., Portal, J.-M., Figueras, J. and Zorian, Y. 1997. Testing Unconfigured FPGA Logic Modules. *IEEE European Test Workshop Compendium of Papers*, 1997, p. 128-132.
- Renovell, 1998. SRAM-Based FPGAs: A Structural Test Approach. *Proc. of the XI Brazilian Symposium on Integrated Circuit Design*, 1998, p. 67-72.
- Renovell, M., Azaïs, F. and Bertrand, Y. 1998. Analysing Relationship between Defect and Fault Model. *IEEE European Test Workshop Compendium of Papers*, 1998, p. 151-155.
- Renovell, M., Portal, J.-M., Figueras, J. and Zorian, Y. 1998a. RAM-Based FPGA's: A Test Approach for the Configurable Logic. *Proc. of the IEEE International Conference on Design, Automation and Test in Europe*, 1998, p. 82-88.
- Renovell, M., Portal, J.-M., Figueras, J. and Zorian, Y. 1998b. SRAM-Based FPGA's: Testing the RAM Mode of the LUT/RAM modules. *IEEE European Test Workshop Compendium of Papers*, 1998, p. 5-9.

- Renovell, M., Portal, J.-M., Figueras, J. and Zorian, Y. 1998c. Testing the interconnect of RAM-based FPGAs. *IEEE Design & Test of Computers*, January-March, 1998, Vol. 15, No. 1, p. 45-50.
- Renovell, M., Portal, J.-M., Figueras, J. and Zorian, Y. 1999. Test Configuration Minimization for the Logic Cells of SRAM-Based FPGAs: A case study. *Proc. of the 4<sup>th</sup> IEEE European Test Workshop*, 1999, p. 146-151.
- Renovell, M., Portal, J.-M., Figueras, J. and Zorian, Y. 1999a. SRAM-Based FPGAs: Logic Cell Test Configuration Minimization. *Proc. of the XIV Design of Circuits and Integrated Systems Conference*, 1999, p. 307-312.
- Renovell, M., Portal, J.-M., Figueras, J. and Zorian, Y. 1999b. Testing the Configurable Interconnect/Logic Interface of SRAM-Based FPGA's. *Proc. of the IEEE International Conference on Design, Automation and Test in Europe*, 1999, p. 618-622.
- Renovell, M., Portal, J.-M., Faure, P., Figueras, J. and Zorian, Y. 2000. Analyzing the Test Generation Problem for an Application-Oriented Test of FPGAs. *Proc. of the 5<sup>th</sup> IEEE European Test Workshop*, 2000, p. 75-80.
- Renovell, M., Portal, J.-M., Faure, P., Figueras, J. and Zorian, Y. 2000a. Test Generation Optimization for a FPGA Application-Oriented Test Procedure. *Proc. of the XV Design of Circuits and Integrated Systems Conference*, 2000, p. 330-336.
- Renovell, M., Faure, P., Portal, J.-M., Figueras, J. and Zorian, Y. 2001. IS-FPGA: A New Symmetric FPGA Architecture with Implicit SCAN. *Proc. of the IEEE International Test Conference*, 2001, p. 924-931.
- Rizzo, D. and Colavin, O. 2002. A Video Compression Case Study on a Reconfigurable VLIW Architecture. *Proc. of the IEEE International Conference on Design, Automation and Test in Europe - Designers' Forum*, 2002, p. 540-546.
- Robertson, I., Irvine, J., Lysaght, P. and Robinson, D. 2002. Improved Functional Simulation of Dynamically Reconfigurable Logic. *Proc. of the 12<sup>th</sup> International Workshop on Field-Programmable Logic and Applications*, 2002, p. 152-161.
- Robinson, D. and Lysaght, P. 2000. Methods of exploiting simulation technology for simulating the timing of dynamically reconfigurable logic. *IEE Proc.-Computer Digital Technology*, May, 2000, Vol. 147, No. 3, p. 175-180.

- Rockett, L. R. 2001. A design based on proven concepts of an SEU-immune CMOS configuration data cell for reprogrammable FPGAs. *Microelectronics Journal*, February, 2001, Vol. 32, No. 2, p. 99-111.
- Rose, J., Francis, R. J., Chow, P. and Lewis, D. 1989. The Effect of Logic Block Complexity on Area of Programmable Gate Arrays. *Proc. of the IEEE Custom Integrated Circuits Conference*, 1989, p. 5.3.1-5.3.5.
- Rose, J., El Gamal, A. and Sangiovanni-Vincentelli, A. 1993. Architecture of Field-Programmable Gate Arrays. *Proceedings of the IEEE*, July, 1993, Vol. 81, No. 7, p. 1013-1029.
- Sachdev, M. 1998. *Defect Oriented Testing for CMOS Analog and Digital Circuits*. 1<sup>st</sup> ed. Dordrecht: Kluwer, 1998. 308 p. ISBN 0-7923-8083-5.
- Sánchez, C. O. 2001. Realización de arreglos embriónicos en FPGA Virtex. *Actas del 3<sup>er</sup> Congreso Internacional en Control, Instrumentación Virtual y Sistemas Digitales*, 2001, p. 182-187.
- Sanchez-Elez, M., Fernández, M., Maestre, R., Hermida, R., Bagherzadeh, N. and Kurdahi, F. J. 2002. A Complete Data Scheduler for Multi-Context Reconfigurable Architectures. *Proc. of the IEEE International Conference on Design, Automation and Test in Europe - Designers' Forum*, 2002, p. 547-552.
- Sandige, R. S. 1990. *Modern Digital Design*. International ed. Singapore: McGraw-Hill, 1990. 743 p. ISBN 0-07-100933-7.
- Sangiovanni-Vincentelli, A., El Gamal, A. and Rose, J. 1993. Synthesis Methods for Field Programmable Gate Arrays. *Proceedings of the IEEE*, July, 1993, Vol. 81, No. 7, p. 1057-1083.
- Santos, J. M. V. 1999. Contribuição das Infra-estruturas de Teste para a Implementação de Sistemas Digitais Tolerantes a Falhas. *Dissertação submetida para obtenção do grau de Doutor*, Faculdade de Engenharia da Universidade do Porto, Agosto, 1999, 284 p.
- Sar-Dessai, V. R. and Walker, D. M. H., 1999. Resistive Bridge Fault Modeling, Simulation and Test Generation. *Proc. of the IEEE International Test Conference*, 1999, p. 596-605.
- Sassatelli, G., Torres, L., Benoit, P., Gil, T., Diou, C., Cambon, G. and Galy, J. 2002. Highly Scalable Dynamically Reconfigurable Systolic Ring-Architecture for DSP applications. *Proc. of the IEEE International Conference on Design, Automation and Test in Europe - Designers' Forum*, 2002, p. 553-558.
- Segura, J., Keshavarzi, A., Soden, J. M. and Hawkins, C. F., 2002. The Nature of Parametric Failures in CMOS ICs. *7<sup>th</sup> IEEE European Test Workshop Informal Digest*, 2002, p. 69-74.

Sengupta, S., Kundu, S., Chakravarty, S., Parvathala, P., Galivanche, R., Kosonocky, G., Rodgers, M. and Mak, TM 1999. Defect-Based Test: A Key Enabler for Successful Migration to Structural Test. *Intel Technology Journal*, Q1, 1999, p. 14. (disponível em <http://www.intel.com/technology/itj/q1199.htm>)

Sharma, A. K. 1998. *Programmable Logic Handbook*. 1<sup>st</sup> ed. New York: McGraw-Hill, 1998. 435 p. ISBN 0-07-057852-4.

Shen, J. P., Maly, W. and Ferguson, F. J. 1985. Inductive Fault Analysis of MOS Integrated Circuits. *IEEE Design & Test of Computers*, December, 1996, Vol. 2, No. 6, p. 13-26.

Shnidman, N. R., Mangione-Smith, W. H. and Potkonjak, M. 1997. Fault Scanner for Reconfigurable Logic. *Proc. of the 17<sup>th</sup> Conference on Advanced Research in VLSI*, 1997, p. 238-255.

Shnidman, N. R., Mangione-Smith, W. H. and Potkonjak, M. 1998. On-Line Fault Detection for Bus-Based Field Programmable Gate Arrays. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, December, 1998, Vol. 6, No. 4, p. 656-666.

Sima, M., Vassiliadis, S., Cotofana, S., van Eijndhoven, J. T. J. and Vissers, K. 2002. Field-Programmable Custom Computing Machines – A Taxonomy. *Proc. of the 12<sup>th</sup> International Workshop on Field-Programmable Logic and Applications*, 2002, p. 79-88.

Singh, S. 1995. Architectural Descriptions for FPGA Circuits. *Proc. of the IEEE Symposium on FPGAs for Custom Computing Machines*, 1995, p. 145-154.

Sipper, M., Mange, D. and Sanchez, E. 1999. Quo Vadis Evolvable Hardware? From Darwin to Darware: Why computer Engineers have started listening to Mother (Nature). *Communications of the ACM*, April, 1999, Vol. 42, No. 4, p. 50-56.

Smit, G. J. M., Havinga, P. J. M., Smit, L. T., Heysters, P. M. and Rosien, M. A. J. 2002. Dynamic Reconfiguration in Mobile Systems. *Proc. of the 12<sup>th</sup> International Workshop on Field-Programmable Logic and Applications*, 2002, p. 171-181.

Smith, D. J. 1997. To create successful designs, know your HDL simulation and synthesis issues. *Electronic Design News*, November 6, 1997, Vol. 42, No. 23, p. 135-144.

Smith, M. J. S. 1997a. *Application-Specific Integrated Circuits*. 1<sup>st</sup> ed. Reading: Addison Wesley, 1997. 1026 p. ISBN 0-201-50022-1.

- Srinivasan, V., Govindarajan, S. and Vemuri, R. 2001. Fine-Grained and Coarse-Grained Behavioral Partitioning with Effective Utilization of Memory and Design Space Exploration for Multi-FPGA Architectures. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, February, 2001, Vol. 9, No. 1, p. 140-158.
- Stroud, C. E., Chen, P., Konala, S. and Abramovici, M. 1996. Evaluation of FPGA Resources for Built-In Self-Test of Programmable Logic Blocks. *Proc. of the 4<sup>th</sup> International Symposium on Field-Programmable Gate Arrays*, 1996, p. 107-113.
- Stroud, C. E., Konala, S., Chen, P. and Abramovici, M. 1996a. Built-In Self-Test of Logic Blocks in FPGAs (Finally, A Free Lunch: BIST Without Overhead!). *Proc. of the 14<sup>th</sup> IEEE VLSI Test Symposium*, 1996, p. 387-392.
- Stroud, C. E., Lee, E., Konala, S. and Abramovici, M. 1996b. Selecting Built-In Self-Test Configurations for Field Programmable Gate Arrays. *Proc. of the IEEE Automatic Test Conference*, 1996, p. 29-35.
- Stroud, C. E., Lee, E., Konala, S. and Abramovici, M. 1996c. Using ILA Testing for BIST in FPGAs. *Proc. of the IEEE International Test Conference*, 1996, p. 68-75.
- Stroud, C. E., Lee, E., and Abramovici, M. 1997. BIST-Based Diagnostic of FPGA Logic Blocks. *Proc. of the IEEE International Test Conference*, 1997, p. 539-547.
- Stroud, C. E., Wijesuriya, S., Hamilton, C. and Abramovici, M. 1998. Built-In Self-Test of FPGA Interconnect. *Proc. of the IEEE International Test Conference*, 1998, p. 404-411.
- Stroud, C. E., Lashinsky, M., Nall, J., Emmert, J. M. and Abramovici, M. 2001. On-Line BIST and Diagnosis of FPGA Interconnect Using Roving STARs. *Proc. of the 7<sup>th</sup> IEEE On-Line Testing Workshop*, 2001, p. 27-33.
- Stroud, C. E. 2002. *A Designer's Guide to Built-In Self-Test*. 1<sup>st</sup> ed. Boston: Kluwer, 2002. 319 p. ISBN 1-4020-7050-0.
- Swager, A. W. 1992. Choosing Complex. *Electronic Design News*, September 17, 1992, Vol. 37, No. 19, p. 74-84.
- Teich, M., Fekete, S. and Schepers, J. 1999. Compile-time optimization of dynamic hardware reconfigurations. *Proc. of the International Conference on Parallel and Distributed Processing Techniques and Applications*, 1999, p. 1097-1103.
- Texas, 1997. *Logic Selection Guide and Databook CD-ROM*. Texas Instruments, Inc., January, 1997. (informação actualizada disponível em <http://www.ti.com>)

Thompson, K. M. 1996. Intel and the Myths of Test. *IEEE Design & Test of Computers*, Spring,

1996, Vol. 13, No. 1, p. 79-81.

Thompson, A. and Layzell, P. 1999. Analysis of Unconventional evolved Electronics: Artificial

evolution can produce bizarre circuits that work - but do we need to understand them?

*Communications of the ACM*, April, 1999, Vol. 42, No. 4, p. 71-79.

Treuer, R., Fujiwara, H. and Agarwal, V. K. 1985. Implementing a Built-In Self-Test PLA Design.

*IEEE Design and Test of Computers*, April, 1985, Vol. 2, No. 2, p. 37-48.

Trimberger, S. 1993. A Reprogrammable Gate Array and Applications. *Proceedings of the IEEE*,

July, 1993, Vol. 81, No. 7, p. 1030-1041.

Ubar, R. and Borrione, D. 1999. Design Error Diagnosis in Digital Circuits without Error Model.

*Proc. of the 10<sup>th</sup> International Conference on Very Large Scale Integration*, 1999, p. 281-292.

Vasilko, M. and Ait-Boudaoud, D. 1995. Scheduling for Dynamically Reconfigurable FPGAs. *Proc.*

*of the International Workshop on Logic and Architecture Synthesis*, 1995, p. 328-336.

Vasilko, M. and Ait-Boudaoud, D. 1996. Architectural Synthesis Techniques for Dynamically

Reconfigurable Logic. *Proc. of the 6<sup>th</sup> International Workshop on Field-Programmable Logic and*

*Applications*, 1996, p. 290-296.

Vasilko, M. and Long, D. 1998. RIFLE-62: A flexible environment for prototyping dynamically

reconfigurable systems. *Proc. of the 9<sup>th</sup> IEEE International Workshop on Rapid System*

*Prototyping*, 1998, p. 130-135.

Vasilko, M. 1999. DYNASTY: A Temporal Floorplanning Based CAD Framework for Dynamically

Reconfigurable Logic Systems. *Proc. of the 9<sup>th</sup> International Workshop on Field-Programmable*

*Logic and Applications*, 1999, p. 124-133.

Vasilko, M., Gibson, D. J., Long, D. and Holloway, S. 1999. Towards a Consistent Design

Methodology for Run-Time Reconfigurable Systems. *IEE Colloquium on Reconfigurable*

*Systems Digest*, 1999, p. 5/1-4.

Vasilko, M. 2000. Design Visualisation for Dynamically Reconfigurable Systems. *Proc. of the 10<sup>th</sup>*

*International Workshop on Field-Programmable Logic and Applications*, 2000, p. 131-140.

Vasilko, M. and Benyon-Tinker, G. 2000. Automatic Temporal Floorplanning with Guaranteed

Solution Feasibility. *Proc. of the 10<sup>th</sup> International Workshop on Field-Programmable Logic and*

*Applications*, 2000, p. 656-664.

- Wang, S.-J. and Tsai, T.-M. 1997. Test and Diagnosis of Faulty Logic Blocks in FPGAs. *Proc. of the International Conference on Computer-Aided Design*, 1997, p. 722-727.
- Wang, S.-J. and Tsai, T.-M. 1999. Test and Diagnosis of Faulty Logic Blocks in FPGAs. *IEE Proceedings - Computer Digital Techniques*, March, 1999, Vol. 146, No. 2, p. 100-106.
- Wondolowski, M., Bennetts, R. G. and Ley, A. 1999. Boundary Scan: The Internet of Test. *IEEE Design & Test of Computers*, July-September, 1999, Vol. 16, No. 3, p. 34-43.
- Woods, R., Trainor, D. and Heron, J.-P. 1998. Applying an XC6200 to real-time image processing. *IEEE Design & Test of Computers*, January-March, 1998, Vol. 15, No. 1, p. 30-38.
- XCell, 1995. Introducing the XC6200 FPGA Architecture: The First FPGA Architecture Optimized for Coprocessing in Embedded System Applications. *Xcell Journal*, No. 18, Q3, 1995, p. 22-23. (disponível em <http://www.xilinx.com>)
- XCell, 1998. Virtex Our New Million-Gate 100-MHz FPGA Technology. *Xcell Journal*, No. 27, Q1, 1998, p. 22-23. (disponível em <http://www.xilinx.com>)
- XCell, 2000. New Spartan-II FPGA Family, Kiss Your ASIC Good-bye. *Xcell Journal*, No. 35, Q1, 2000, p. 5-7. (disponível em <http://www.xilinx.com>)
- Xilinx, 1995. XC6200 FPGA Data Sheet. Xilinx, Inc., 1995. 73 p.
- Xilinx, 1998. The Low-Cost, Efficient Serial Configuration of Spartan FPGAs. *Application Note XAPP098*, November 13, 1998, 9 p. (informação actualizada disponível em <http://www.xilinx.com>)
- Xilinx, 2000. Virtex Series Configuration Architecture User Guide. *Application Note XAPP151*, September 27, 2000, 45 p. (informação actualizada disponível em <http://www.xilinx.com>)
- Xilinx, 2000a. *The Xilinx JBits Software Development Kit*, disponível como Limited Distribution Release (para colocar questões ou obter uma cópia do programa: [jbits@xilinx.com](mailto:jbits@xilinx.com))
- Xilinx, 2000b. *Virtex 2.5V Field Programmable Gate Arrays Data Sheet*. Xilinx, Inc., January, 2000. 72 p. (informação actualizada disponível em <http://www.xilinx.com>)
- Xilinx, 2000c. Virtex FPGA Series Configuration and Readback. *Application Note XAPP138*, October 4, 2000, 38 p. (informação actualizada disponível em <http://www.xilinx.com>)
- Xilinx, 2000d. Configuration and Readback of Virtex FPGAs Using (JTAG) Boundary-Scan. *Application Note XAPP139*, February 18, 2000, 15 p. (informação actualizada disponível em <http://www.xilinx.com>)

Xilinx, 2000e. Correcting single-event upsets through Virtex Partial configuration. *Application Note XAPP216*, June 1, 2000, 12 p. (informação actualizada disponível em <http://www.xilinx.com>)

Xilinx, 2001. Linear Feedback Shift Register in Virtex Devices. *Application Note XAPP210*, January 9, 2001, 5 p. (informação actualizada disponível em <http://www.xilinx.com>)

Xilinx, 2002. *Virtex-II Pro Platform FPGA Handbook*. Xilinx, Inc., January, 2002. 586 p. (informação actualizada disponível em <http://www.xilinx.com>)

Xilinx, 2002a. *Xilinx DataSource* CD-ROM. Xilinx, Inc., July, 2002. (informação actualizada disponível em <http://www.xilinx.com>)

Yao, X. 1999. Following the Path of Evolvable Hardware. *Communications of the ACM*, April, 1999, Vol. 42, No. 4, p. 47-49.

Yamada, T. and Nanya, T. 1984. Stuck-At Fault Tests in the Presence of Undetectable Bridging Faults. *IEEE Transactions on Computers*, August, 1984, Vol. C-33, No. 3, p. 758-761.

Yu, Y., Xu, J., Huang, W. K. and Lombardi, F. 1999. Minimizing the Number of Programming Steps for Diagnosis of Interconnect Faults in FPGAs. *Proc. of the 8<sup>th</sup> Asian Test Symposium*, 1999, p. 357-362.

Zhang, X. and Ng, K. W. 2000. A review of high-level synthesis for dynamically reconfigurable FPGAs. *Microprocessors and Microsystems*, August 1, 2000, Vol. 24, No. 4, p. 199-211.

Zorian, Y. 1999. Built-In Self-Test. *Microelectronic Engineering*, November, 1999, Vol. 49, No. 1-2, p. 135-138.

## **11. ANEXOS**



A elevada quantidade de informação relevante que se pretendia colocar em anexo, e também o seu formato, levou à opção por um suporte electrónico para a sua apresentação. Desta forma, o anexo é constituído por um CD-ROM, fixado no verso da contracapa deste trabalho, que contém uma aplicação auto-executável que apresenta, de imediato, os vários elementos armazenados. Desses elementos, fazem parte este documento, uma descrição aprofundada da arquitectura e modo de programação do componente usado na implementação e validação da proposta exposta, os artigos submetidos e aceites em várias conferências e seminários respeitantes a este trabalho, os ficheiros relativos à implementação da proposta, os ficheiros com resultados das experimentações efectuadas, o esquema da fonte de tensão comutada projectada e construída pelo autor para alimentação da FPGA usada na validação deste trabalho, uma cópia do relatório intercalar do projecto mencionado na secção 1.2, uma cópia do *Curriculum Vitae* do autor e uma base de dados contendo as referências de toda a bibliografia reunida pelo autor durante o trabalho de Doutoramento.

