

Universidade do Porto
Faculdade de Engenharia

**Contribuição das Infra-estruturas
de Teste para a Implementação de
Sistemas Digitais Tolerantes a Falhas**

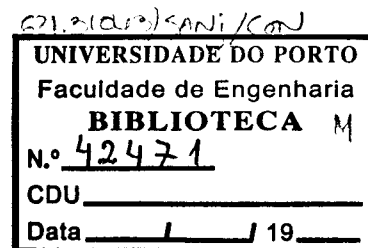
José Miguel Lopes Vieira dos Santos

**Dissertação apresentada na Faculdade de Engenharia da
Universidade do Porto para obtenção do grau de Doutor
em Engenharia Electrotécnica e de Computadores**

Porto
Agosto de 1999

José Miguel Lopes Vieira dos Santos

Contribuição das Infra-estruturas de Teste para a Implementação de Sistemas Digitais Tolerantes a Falhas



Dissertação submetida para 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

Agosto de 1999

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

*À Tina, ao Pedro e ao Tiago,
pela paciência e tolerância às minhas falhas.*

“I believe engineers always tend to optimize their design, even though they are not aware of the necessary technologies. [...] In this sense, I think the process of fault-tolerant system design may rather belong more to the domain of art than to the domain of engineering [...].”

Hirokazu Ihara

(Hitachi Space System Division)

Fault Tolerant Computing Symposium, 1992

Resumo

Palavras chave: Normas IEEE 1149, circuitos integrados com monitorização concorrente, circuitos integrados tolerantes a falhas permanentes, fiabilidade da informação na infra-estrutura 1149.1.

A evolução recente dos sistemas electrónicos tem permitido utilizá-los num número crescente de aplicações, com graus diversos de exigência na segurança de funcionamento, que não permitem algumas características dos sistemas confiáveis tradicionais, nomeadamente o custo elevado, bem como o volume e consumo de energia significativos. Na origem dessa evolução, temos a crescente densidade de integração permitida pela redução das geometrias, factores que no entanto degradam a fiabilidade dos circuitos e o seu tempo de vida, motivando um interesse crescente pela monitorização em serviço. Em paralelo, a infra-estrutura IEEE 1149.1 é cada vez mais utilizada nos testes de produção, mas raramente é activada durante o funcionamento dos componentes.

Nesta tese, e após uma introdução geral à área dos sistemas confiáveis, apresentamos em primeiro lugar a solução que denominamos *CST (Concurrent Scan Test)*, que reutiliza a infra-estrutura 1149.1 e os vectores de teste definidos na produção, para efectuar uma monitorização concorrente do circuito de missão. Com uma sobrecarga de teste diminuta, o CST permite ainda uma tolerância a falhas restrita aos vectores de teste memorizados no controlador da infra-estrutura de teste.

Com o objectivo de expandir a aplicabilidade do CST ao projecto de sistemas tolerantes a falhas, apresentamos de seguida a arquitectura denominada *XMR (Redundância Modular Incompleta)*, uma proposta destinada a circuitos integrados que devem tolerar as próprias falhas físicas em aplicações críticas, com um comportamento semelhante ao de uma arquitectura TMR (Redundância Modular Tripla) com intervalo de latência. A utilização de apenas duas réplicas do circuito de missão, associadas a uma infra-estrutura CST evoluída, permite confinar o erro nas células BST e votar, de forma discreta, com a informação dos vectores utilizados na monitorização.

Finalmente, e para responder às exigências que estas duas propostas colocam à fiabilidade da informação veiculada pela infra-estrutura BST, propomos o controlo de erros na transmissão, através de um bit de paridade, sem introduzir novos estados no controlador do TAP.

As propostas apresentadas foram validadas por simulação e implementação em dispositivos lógicos programáveis de média complexidade, sendo incluídas em anexo as respectivas especificações formais e os resultados da simulação.

Abstract

Keywords: IEEE 1149 standards, concurrent monitoring, integrated circuits with fault tolerance to permanent faults, reliability of the information in the IEEE 1149.1 infrastructure.

Recent advances in microelectronics technology enabled its usage in a growing number of applications, with varying requirements concerning the safety of operation. However, market reasons dictate that some characteristics of traditional dependable systems are not acceptable in some applications, namely high cost and energy consumption. Sub-micron technologies and their inherent higher integration levels are the two main factors underlying the expansion of microelectronics to fault-tolerant systems, but these same factors are responsible for lower circuit reliability and a shorter life span, leading in turn to an increased interest for circuit monitoring during on-line operation. Independent of this fact, but of relevance in the context of this work, boundary scan became an important production test technology, steadily growing in acceptance since its approval as an IEEE standard in 1990. However, and in spite of the powerful operating modes provided by this test infrastructure, it is very seldomly used during normal circuit operation.

Following a general introduction to the characteristics of dependable systems, the *Concurrent Scan Test* (CST) proposal is presented, which reuses the 1149.1 infrastructure and the production test vectors, to enable concurrent monitoring of the mission circuits during system operation. The CST proposal has very low overhead in relation to the standard boundary scan infrastructure and enables a discrete fault tolerance mode, synchronous to the operation of the mission circuit and restricted to those vectors stored in the test controller memory.

The *Incomplete Modular Redundancy* (XMR) proposal is then presented, enabling the design of fault-tolerant systems based in the CST infrastructure. The XMR architecture addresses an application scope consisting of those integrated circuits required to withstand internal physical faults in safety-critical applications, and its operating characteristics are similar to those of a *Triple Modular Redundancy* (TMR) system with a latency interval. The use of only two replicas of each mission circuit, associated to an extended CST infrastructure, enables fault confinement in the boundary scan cells and the implementation of a discrete voting procedure, using the information contained in the set of test vectors used for concurrent monitoring. Besides enabling the detection of common mode faults in the replicas of the mission circuit, the XMR architecture can be integrated in a single component, with a better cost-benefit ratio, when compared to the alternatives traditionally available.

The information transmitted through the boundary scan infrastructure during concurrent monitoring has higher reliability requirements, due to the problems that may be caused by an error in the bit stream. To meet this requirement, the last chapter in this thesis proposes error detection through a parity control bit. This method does not require additional TAP controller states and retains the traditional state transition path for each test vector, if an error in the bit stream is allowed into the update stage during a short time interval, or enables detection before update, if two additional state transitions are acceptable.

Résumé

Mots clé: Normes IEEE 1149, circuits intégrés contrôlés en ligne, circuits intégrés tolérant les fautes permanentes, fiabilité de l'information dans l'infrastructure 1149.1.

Dans les dernières années, l'évolution de la microélectronique a permis d'utiliser les systèmes électroniques dans nombreuses applications demandant la sûreté de fonctionnement, mais n'admetant pas certaines des caractéristiques inhérentes aux systèmes fiables conventionnels, notamment le coût élevé, le surcoût en surface important et la haute consommation de courant. D'une part, la réduction des géométries, à l'origine de la densité croissante d'intégration, impact la fiabilité et le temps de survie des circuits, mettant en valeur les avantages d'utilisation des techniques de test en ligne. D'autre part, l'infrastructure 1149.1, utilisée fréquemment en fabrication, reste inactive pour la plus part de la vie des composants.

Dans cette thèse, suite à une introduction générale au domaine des systèmes fiables, on propose la solution identifiée comme *CST* (*Concurrent Scan Test*, test de balayage concurrent), qui réutilise l'infrastructure 1149.1 et les vecteurs définis pour le test de fabrication afin d'assurer le test concurrent du circuit lors de l'exécution de l'application. Soumettant le circuit à une surcharge à peine réduite du point de vue test, le CST permet d'avantage une tolérance aux fautes discrète, synchronisée avec la fonctionnalité, bien que limitée aux vecteurs de test retenus en mémoire.

Ayant comme objectif d'élargir l'application du CST à la conception de systèmes tolérant aux fautes, on présente dans la suite l'architecture nommée *XMR* (Redondance Modulaire Incomplète). Cette solution, avec un comportement semblable à celui d'une architecture TMR avec latence, s'applique aux circuits qui, intégrés dans une application critique, doivent tolérer ses propres fautes physiques. L'utilisation (à peine) de deux copies du circuit, associées à une infrastructure CST améliorée, permet de confiner l'erreur dans les cellules BST et de voter, de forme discrète, à partir des vecteurs utilisés pour le contrôle en ligne de la fonction. En plus de détecter les fautes mode commun parvenant sur les copies, l'architecture XMR peut être intégrée dans un seul composant avec un rapport coût-bénéfice avantageux face aux alternatives conventionnelles.

À la fin, et en réponse aux demandes posées par ces deux propositions à la fiabilité de l'information traitée par l'infrastructure BST, on propose la vérification d'erreurs de transmission qui, de manière transparente à la norme, utilise un bit de parité sans introduire de nouveaux états au contrôle du TAP 1149.1.

Les solutions proposées ont été validées par des simulations d'implémentations basées sur des dispositifs en logique programmable de complexité moyenne. Les annexes présentent les spécifications formelles et les résultats de simulation associés à ces implémentations.

Agradecimentos

Este trabalho teria sido mais difícil de levar a bom termo, sem a colaboração de um conjunto de pessoas que, a diversos níveis, facilitaram a troca de ideias e auxiliaram num sem número de tarefas acessórias que trabalhos desta natureza sempre implicam. Assim, e correndo o risco de ser injusto para alguém, começo por agradecer ao Prof. Dr. Martins Ferreira o tema sugerido, que me levou a descobrir melhor a área da Tolerância a Falhas, com um imenso prazer. Para além disso, todo um sem número de orientações que foram inegavelmente proveitosas. Ao Eng. Gustavo Alves agradeço diversas sugestões na área do teste e ao Eng. Manuel Gericota a paciência na revisão final. De forma semelhante, agradeço também ao Eng. Miguel Santiago o auxílio prestado em diversas ocasiões e à Dra. Inês Cambeiro a inesgotável paciência face às inelutáveis agruras burocráticas.

Agradeço também ao Doutor Marcelo Lubaszewski as palavras encorajadoras, ao Eng. J. S. Augusto, do IST, pelo auxílio em diversas ocasiões, e a muitas outras pessoas que por correio electrónico me transmitiram sugestões e responderam a múltiplas questões. Agradeço ainda ao Prof. Dr. Silva Matos a disponibilidade demonstrada numa fase prévia do meu envolvimento neste trabalho.

Nota ao Leitor

Na escrita de um documento que abrange duas áreas onde predomina a língua inglesa, e onde a obrigatoriedade das comunicações serem efectuadas nesta língua condiciona novas denominações, é forçoso o conflito entre o purismo linguístico e a terminologia corrente. Embora a tradução nem sempre conduza a um resultado fiel, optámos pela utilização quase integral de termos em língua portuguesa, procurando conciliar a terminologia das normas do Instituto Português da Qualidade, NP-2626-191 de 1994 (edição de Abril de 1995) e NP-3003-8 de 1991 (edição de Junho de 1993), com a que encontrámos em trabalhos já efectuados (nas áreas abrangidas) em português. O resultado da eliminação quase integral de termos anglófonos arreigados (*hardware*, *software*, *self-checking*, *checker* e *watchdog*, por exemplo), pode ser conflituoso nalguns casos e até dificultar inicialmente a compreensão, mas a leitura torna-se mais fluida, uma vez que o leitor se integre na terminologia.

No documento, são representados em *itálico* os termos em língua estrangeira e os que, em português, são introduzidos pela primeira vez. A representação a **negro** será utilizada para realçar indicações relevantes.

Índice

1. INTRODUÇÃO	3
1.1 ÂMBITO DO TRABALHO E OBJECTIVOS PRINCIPAIS.....	3
1.2 ORIGINALIDADE E CONTRIBUIÇÃO INOVADORA	4
1.3 ORGANIZAÇÃO DA TESE.....	5
2. CONFIABILIDADE E TESTABILIDADE.....	9
2.1 INTRODUÇÃO À CONFIABILIDADE	11
2.1.1 <i>Etapas de sobrevivência</i>	12
2.1.2 <i>Breve história da Tolerância a Falhas</i>	13
2.1.3 <i>Sumário de custos e benefícios</i>	15
2.2 TESTABILIDADE.....	16
2.2.1 <i>Modelos de Falhas</i>	18
2.2.2 <i>Geração do Teste</i>	21
2.2.3 <i>Minimização do custo de aplicação do teste</i>	25
2.2.4 <i>Projecto para a Testabilidade</i>	28
2.3 AS INFRA-ESTRUTURAS IEEE 1149.X.....	31
2.3.1 <i>A norma IEEE 1149.1</i>	31
2.3.2 <i>Outras normas</i>	36
2.4 CONCLUSÃO	38
3. PROJECTO PARA A CONFIABILIDADE.....	41
3.1 CONCEITOS DE CONFIABILIDADE	41
3.1.1 <i>Parâmetros da Confiabilidade</i>	42
3.1.2 <i>Medidas preventivas</i>	46
3.2 ARQUITECTURAS TOLERANTES A FALHAS	49
3.2.1 <i>Circuitos Tolerantes a Falhas</i>	49
3.2.1.1 <i>Arquitecturas Estáticas</i>	50
3.2.1.2 <i>Arquitecturas Dinâmicas</i>	53
3.2.1.3 <i>Arquitecturas Mistas</i>	54
3.2.1.4 <i>Arquitecturas Auto-reparáveis</i>	55
3.2.1.5 <i>Arquitecturas específicas</i>	57
3.2.2 <i>Tolerância a Falhas em Programação</i>	57
3.2.3 <i>Sistemas de Tempo-Real</i>	59
3.2.4 <i>Exemplos</i>	60
3.3 TÉCNICAS DE PROJECTO BASEADAS EM CODIFICAÇÃO	67
3.3.1 <i>Erros e Códigos</i>	67
3.3.2 <i>Circuitos Auto-Verificáveis</i>	69
3.3.3 <i>Circuitos Imunes a Falhas</i>	72
3.4 TESTE EM SERVIÇO.....	73
3.4.1 <i>Testes a nível de funcionamento lógico</i>	73
3.4.2 <i>Testes a nível paramétrico</i>	75
3.5 CONCLUSÃO	76

4. CIRCUITOS INTEGRADOS CONFIÁVEIS COM INFRA-ESTRUTURA BST: REQUISITOS E CONDICIONANTES.....	79
4.1 ANÁLISE DE REQUISITOS E CONDICIONANTES	80
4.1.1 <i>Recomendações e orientação</i>	80
4.1.2 <i>Caracterização dos requisitos de Confiabilidade</i>	81
4.1.3 <i>A Confiabilidade nos componentes VLSI</i>	83
4.1.4 <i>Análise comparativa das arquitecturas Tolerantes a Falhas</i>	87
4.1.5 <i>Utilização da norma 1149.1 na Monitorização</i>	89
4.1.6 <i>Características do teste concorrente e dos vectores de teste</i>	92
4.2 ESPECIFICAÇÃO.....	96
5. INFRA-ESTRUTURA CST PARA DETECÇÃO CONCORRENTE DE ERRO....	101
5.1 DEFINIÇÃO DA INFRA-ESTRUTURA CST.....	102
5.1.1 <i>A infra-estrutura CST ao nível do Módulo</i>	103
5.1.2 <i>A infra-estrutura CST ao nível do Circuito de Missão</i>	104
5.2 PRINCÍPIO DE FUNCIONAMENTO DA INFRA-ESTRUTURA CST	107
5.2.1 <i>Caracterização do controlo de captura</i>	108
5.2.2 <i>Constituição das células</i>	113
5.2.3 <i>Constituição do bloco de captura (CB)</i>	113
5.3 ANÁLISE DO COMPORTAMENTO DINÂMICO E OPTIMIZAÇÃO DA INFRA-ESTRUTURA CST	116
5.3.1 <i>Tempo de teste (T_t)</i>	117
5.3.2 <i>Seleção dos vectores de teste</i>	124
5.3.3 <i>Optimização do funcionamento da infra-estrutura CST</i>	124
5.4 APLICABILIDADE DA INFRA-ESTRUTURA CST	127
5.4.1 <i>Sobrecarga da infra-estrutura CST</i>	127
5.4.2 <i>Testabilidade entre registos</i>	128
5.4.3 <i>Aplicação à Tolerância a Falhas discreta</i>	130
5.4.4 <i>O caso dos circuitos sequenciais</i>	133
5.5 DETERMINAÇÃO DA FREQUÊNCIA MÍNIMA DE TCK.....	134
5.6 INTRODUÇÃO AO EXEMPLO DO ANEXO A.....	137
5.7 CONCLUSÃO.....	138
6. ARQUITECTURA XMR: REDUNDÂNCIA MODULAR INCOMPLETA	141
6.1 SISTEMAS DUPLEX COM CST.....	142
6.1.1 <i>Considerações sobre o mascaramento e detecção de erro</i>	143
6.1.2 <i>Definição de arquitecturas com Redundância Fraccionária</i>	144
6.2 A ARQUITECTURA XMR	146
6.2.1 <i>Princípio de funcionamento da arquitectura XMR</i>	147
6.2.2 <i>Modos de operação</i>	148
6.2.3 <i>Características dinâmicas</i>	149
6.2.4 <i>As células de decisão</i>	150
6.3 ANÁLISE DE FIABILIDADE DA ARQUITECTURA XMR	157
6.4 EXTENSÃO DA ARQUITECTURA XMR À CONFIGURAÇÃO 2XMR.....	160
6.5 CONCLUSÃO.....	162

7. FIABILIDADE DA INFORMAÇÃO NA INFRA-ESTRUTURA IEEE 1149.1	165
7.1 A FIABILIDADE DE TRANSMISSÃO NA INFRA-ESTRUTURA BST	165
7.1.1 <i>Processos para determinar a Paridade</i>	168
7.1.2 <i>Análise do diagrama de estados do TAP</i>	169
7.2 CONCLUSÃO	172
8. CONCLUSÃO	175
8.1 METODOLOGIA DE IMPLEMENTAÇÃO DAS PROPOSTAS APRESENTADAS	176
8.2 COMENTÁRIOS SOBRE O BS μ C NECESSÁRIO PARA MONITORIZAÇÃO	178
8.3 DIRECÇÕES FUTURAS.....	179
9. REFERÊNCIAS CITADAS NO TEXTO	183
10. ANEXOS	206
10.1 ANEXO A: O CIRCUITO INTEGRADO COM INFRA-ESTRUTURA CST	207
10.1.1 <i>Esquemas do grupo de entrada (GRPIN5)</i>	209
10.1.2 <i>Esquemas do grupo de saída (GRPCPT)</i>	210
10.1.3 <i>Esquema do circuito de controlo de captura (CB)</i>	211
10.1.4 <i>Multiplexer para TDO (MUXTDO)</i>	212
10.1.5 <i>Esquemas e diagrama do controlador do TAP</i>	213
10.1.6 <i>Diagrama temporal dos ciclos de detecção e captura</i>	217
10.2 ANEXO B: O CIRCUITO INTEGRADO COM ARQUITECTURA XMR	219
10.2.1 <i>Esquemas dos blocos de saída e decisão</i>	222
10.2.2 <i>Diagramas temporais</i>	224
10.3 ANEXO C: O CI COM ARQUITECTURA XMR E CONTROLO DE PARIDADE	229
10.3.1 <i>Esquemas dos blocos de paridade</i>	231
10.3.2 <i>Diagramas temporais</i>	232
10.3.3 <i>Relatório de projecto gerado pelo compilador do MAX+PLUS II</i>	236

Lista de Figuras

<i>Figura 1-1. Áreas abrangidas por esta tese.....</i>	<i>5</i>
<i>Figura 2-1. Hierarquia de projecto considerada nesta tese.....</i>	<i>10</i>
<i>Figura 2-2. As duas grandes fases da vida de um circuito.....</i>	<i>16</i>
<i>Figura 2-3. Arquitectura IEEE 1149.1.....</i>	<i>32</i>
<i>Figura 2-4. Células de deslocamento.....</i>	<i>32</i>
<i>Figura 2-5. Diagrama de Estados do TAP.....</i>	<i>33</i>
<i>Figura 2-6. Infra-estrutura IEEE 1149.4.....</i>	<i>37</i>
<i>Figura 2-7. Conversão entre as normas 1149.1-1149.5.....</i>	<i>38</i>
<i>Figura 3-1. Evolução da taxa de falhas num componente.....</i>	<i>42</i>
<i>Figura 3-2. Disposição de componentes em série e em paralelo.....</i>	<i>50</i>
<i>Figura 3-3. Arquitecturas TMR com CVM único e triplo.....</i>	<i>51</i>
<i>Figura 3-4. Comparação de fiabilidade BF - TMR.....</i>	<i>52</i>
<i>Figura 3-5. Redundância Comparativa.....</i>	<i>54</i>
<i>Figura 3-6. Redundância auto-purgável.....</i>	<i>55</i>
<i>Figura 3-7. Redundância Híbrida.....</i>	<i>56</i>
<i>Figura 3-8. Redundância Hierárquica.....</i>	<i>56</i>
<i>Figura 3-9. Estrutura do Teste no AE11.....</i>	<i>64</i>
<i>Figura 3-10. Arquitectura do computador tolerante a falhas para o ATV.....</i>	<i>65</i>
<i>Figura 3-11. Arquitectura do processador do G4-FT.....</i>	<i>66</i>
<i>Figura 3-12. Esquema geral de um circuito auto-verificável.....</i>	<i>69</i>
<i>Figura 3-13. Verificadores auto-testável e auto-verificável.....</i>	<i>71</i>
<i>Figura 4-1. Universo de Vectores.....</i>	<i>93</i>
<i>Figura 5-1. Hierarquia de projecto.....</i>	<i>102</i>
<i>Figura 5-2. Acesso independente aos CM do Módulo.....</i>	<i>103</i>
<i>Figura 5-3. Infra-estrutura BST no circuito de missão.....</i>	<i>104</i>
<i>Figura 5-4. Relações entre os vectores de funcionamento e de teste.....</i>	<i>105</i>
<i>Figura 5-5. Grupos de células de E/S e deslocamento simultâneo de VT_i e V_o.....</i>	<i>107</i>
<i>Figura 5-6. Sinais associados ao Bloco de Captura (CB).....</i>	<i>108</i>
<i>Figura 5-7. Diagrama que identifica os estados onde a captura é interdita.....</i>	<i>109</i>
<i>Figura 5-8. Situações em que ocorre a activação e desactivação de CB.....</i>	<i>110</i>
<i>Figura 5-9. Leitura de CB através do registo de instrução.....</i>	<i>111</i>
<i>Figura 5-10. Fluxograma correspondente ao funcionamento da infra-estrutura CST.....</i>	<i>112</i>
<i>Figura 5-11. Células de detecção.....</i>	<i>113</i>
<i>Figura 5-12. Sinais utilizados para controlar a captura.....</i>	<i>114</i>
<i>Figura 5-13. A lógica de controlo de captura.....</i>	<i>114</i>
<i>Figura 5-14. Constituição do CB.....</i>	<i>115</i>
<i>Figura 5-15. Funcionamento autónomo de um registo.....</i>	<i>116</i>
<i>Figura 5-16. Relação entre os intervalos t_v, T_v e T_d.....</i>	<i>118</i>
<i>Figura 5-17. Curvas (T_v/ϕ) em função de n_i e N.....</i>	<i>121</i>
<i>Figura 5-18. Comparação das curvas associadas a T_v e a T_d (para $f = 2\phi$).....</i>	<i>122</i>
<i>Figura 5-19. Número de vectores de teste verificados por BF em $T_L = 50ms$ ($\phi = 10\text{ MHz}$).....</i>	<i>123</i>

Figura 5-20. Deslocamento de CB juntamente com V_o	125
Figura 5-21. Contador de registos: deslocamento dos bits e selecção dos registos.....	126
Figura 5-22. Constituição das células dedicadas à leitura do contador.....	126
Figura 5-23. Deslocamento do bit menos significativo do contador.....	127
Figura 5-24. Relações de testabilidade inter-registos.....	129
Figura 5-25. Definição de múltiplos grupos num mesmo registo.....	130
Figura 5-26. Injecção de vectores de teste sincronizada.....	131
Figura 5-27. Diagrama temporal para injecção de erros e correcção discreta da saída.....	131
Figura 5-28. Controlo do sinal MODE na injecção de erros.....	132
Figura 5-29. Controlo de Update-DR para substituição discreta do circuito de missão.....	132
Figura 5-30. Os quatro grupos a monitorar no modelo de Huffman.....	133
Figura 5-31. T_v como função de T_L e V	135
Figura 5-32. Intervalo t_v em função de T_v e N	135
Figura 5-33. Frequência mínima de TCK para verificar os vectores necessários.....	136
Figura 5-34. Esquema do circuito ISCAS'85 C17.....	137
Figura 6-1. Arquitectura duplex com CST.....	142
Figura 6-2. Quantidade de informação num circuito.....	144
Figura 6-3. Intervalo de detecção e correcção versus encargo associado.....	146
Figura 6-4. Arquitectura XMR.....	146
Figura 6-5. Princípio de funcionamento da arquitectura XMR.....	147
Figura 6-6. Tempo médio de recuperação num CI XMR.....	150
Figura 6-7. Esquema da célula de decisão.....	151
Figura 6-8. Localização do circuito de retenção e selecção.....	153
Figura 6-9. Redefinição do bloco de decisão nas células.....	153
Figura 6-10. Diagrama geral do circuito de retenção e selecção.....	154
Figura 6-11. Sequência de estados do bloco de retenção.....	155
Figura 6-12. Disposição dos componentes na arquitectura XMR.....	157
Figura 6-13. Fiabilidade do CM e das arquitecturas TMR e XMR.....	159
Figura 6-14. Acréscimo da fiabilidade das arquitecturas TMR e XMR, relativa ao CM.....	160
Figura 6-15. A configuração 2XMR.....	161
Figura 7-1. Utilização de interrupções.....	166
Figura 7-2. Matriz de acessos aos blocos funcionais.....	167
Figura 7-3. Sequência temporal de dados e do bit de paridade.....	168
Figura 7-4. Geração de paridade em série e em paralelo.....	169
Figura 7-5. Controlo de paridade com transição de Exit-1 para Update.....	170
Figura 7-6. Controlo de paridade com transição via Pause e Exit-2.....	171
Figura 8-1. Etapas da metodologia de implementação.....	177
Figura 10-1. Esquema do CI com CST.....	207
Figura 10-2. Terminais do CI com CST.....	208
Figura 10-3. Esquema do circuito ISCAS'85 C17.....	208
Figura 10-4. Grupo de células de entrada e detecção (GRPIN5).....	209
Figura 10-5. Esquema da célula de detecção.....	209
Figura 10-6. Grupo de células de saída e captura (GRPCPT).....	210
Figura 10-7. Célula de saída e captura.....	210
Figura 10-8. Circuito de controlo de captura (CB).....	211
Figura 10-9. Circuito de multiplexagem para TDO (MUXTDO).....	212
Figura 10-10. Controlador do TAP IEEE-P1149.1 (TAP).....	213
Figura 10-11. Diagramas temporais de funcionamento do controlador do TAP 1149.1.....	214

<i>Figura 10-12. Parte 1 do controlador do TAP</i>	<i>215</i>
<i>Figura 10-13. Parte 2 do controlador do TAP</i>	<i>216</i>
<i>Figura 10-14. Diagrama de detecção e captura do circuito com infra-estrutura CST</i>	<i>218</i>
<i>Figura 10-15. Diagrama de blocos do CI com arquitectura XMR</i>	<i>219</i>
<i>Figura 10-16. Entradas e saídas do CI com arquitectura XMR</i>	<i>220</i>
<i>Figura 10-17. Duplo circuito de missão, com injeção de faltas (CUTXMR)</i>	<i>221</i>
<i>Figura 10-18. Circuito C17 com entrada para injeção de uma falta</i>	<i>221</i>
<i>Figura 10-19 . Grupo de Saída e Decisão (GRPVOT)</i>	<i>222</i>
<i>Figura 10-20. Célula de Decisão</i>	<i>223</i>
<i>Figura 10-21 . Lógica de Decisão</i>	<i>223</i>
<i>Figura 10-22. Circuito de retenção e direccionamento da saída (LGCRET)</i>	<i>224</i>
<i>Figura 10-23. Comportamento sem faltas</i>	<i>226</i>
<i>Figura 10-24. Comportamento com a falta fa</i>	<i>227</i>
<i>Figura 10-25. Comportamento com a falta fb</i>	<i>228</i>
<i>Figura 10-26. Esquema de um CI XMR com controlo de paridade</i>	<i>229</i>
<i>Figura 10-27. Pinos do CI utilizado na simulação</i>	<i>230</i>
<i>Figura 10-28. Pinos necessários no CI final</i>	<i>230</i>
<i>Figura 10-29. Circuito de geração de paridade série (PARDSR)</i>	<i>231</i>
<i>Figura 10-30. Ciclo temporal correspondente</i>	<i>231</i>
<i>Figura 10-31. Circuito que prolonga o sinal ENABLE (ENABLP)</i>	<i>232</i>
<i>Figura 10-32. Diagrama temporal do CI sem faltas</i>	<i>233</i>
<i>Figura 10-33. Diagrama temporal do CI com a falta fa</i>	<i>234</i>
<i>Figura 10-34. Diagrama temporal do CI com a falta fb</i>	<i>235</i>

Lista de Tabelas

<i>Tabela 2-1. Nomenclatura utilizada em sistemas confiáveis.....</i>	<i>11</i>
<i>Tabela 2-2. Exemplo de resultados obtidos com o algoritmo FAN.....</i>	<i>25</i>
<i>Tabela 2-3. Número de vectores de teste para alguns circuitos de referência.....</i>	<i>28</i>
<i>Tabela 2-4. Instruções obrigatórias da norma 1149.1.....</i>	<i>34</i>
<i>Tabela 3-1. Taxa de falhas λ.....</i>	<i>44</i>
<i>Tabela 3-2. Características da arquitectura TMR.....</i>	<i>53</i>
<i>Tabela 3-3. Métodos de detecção de erro.....</i>	<i>53</i>
<i>Tabela 3-4. Características de confiabilidade em diversas aplicações.....</i>	<i>63</i>
<i>Tabela 3-5. Caracterização dos erros mais relevantes nos CI.....</i>	<i>67</i>
<i>Tabela 3-6. Tipos de códigos mais relevantes.....</i>	<i>68</i>
<i>Tabela 3-7. Tipos de códigos para detecção.....</i>	<i>68</i>
<i>Tabela 4-1. Níveis de confiabilidade e de tratamento de erro.....</i>	<i>82</i>
<i>Tabela 4-2. Razões a favor e limitações da confiabilidade nos VLSI.....</i>	<i>86</i>
<i>Tabela 4-3. Arquitecturas tolerantes a falhas e encargo correspondente.....</i>	<i>87</i>
<i>Tabela 4-4. Número mínimo conhecido de vectores de teste para circuitos ISCAS'85.....</i>	<i>94</i>
<i>Tabela 4-5. Número mínimo conhecido de vectores de teste para circuitos ISCAS'89.....</i>	<i>95</i>
<i>Tabela 4-6. Características do ASIC referido em [Zor98].....</i>	<i>96</i>
<i>Tabela 4-7. Parâmetros de referência a considerar no projecto.....</i>	<i>98</i>
<i>Tabela 5-1. Valores de t_v em função de TCK e de n_i.....</i>	<i>120</i>
<i>Tabela 5-2. Vectores de detecção para o circuito ISCAS'85 C17.....</i>	<i>137</i>
<i>Tabela 6-1. Características dos Modos de Funcionamento.....</i>	<i>149</i>
<i>Tabela 6-2. Matriz de decisão.....</i>	<i>152</i>
<i>Tabela 6-3. Tabela de verdade para o circuito que gera E e S.....</i>	<i>154</i>
<i>Tabela 6-4. Informação associada aos sinais E e S.....</i>	<i>155</i>
<i>Tabela 6-5. Tabela de verdade do circuito que gera o sinal X.....</i>	<i>156</i>

Lista de Acrónimos

λ	Taxa de falhas
ABM	<i>Analog Boundary Module</i>
AHDL	<i>Altera Hardware Description Language</i>
ALU	<i>Arithmetic and Logic Unit</i>
ASIC	<i>Application Specific Integrated Circuit</i>
ATAP	<i>Analog Test Access Port</i>
BF	Bloco Funcional
BIST	<i>Built-In Self-Test</i>
BP	<i>Bypass</i> (registro de)
BS	<i>Boundary Scan</i>
BS μ C	Controlador da infra-estrutura BS
BSDL	<i>Boundary Scan Description Language</i>
BST	<i>BS Test</i>
CB	Bloco de Captura
CCI	Carta de Circuito Impresso
CI	Circuito Integrado
CLA	<i>Carry Lookahead</i>
CM	Circuito de Missão (ou Monitorizado)
CMOS	<i>Complementary Mos-fet transistor</i>
COTS	Componentes “ <i>Commercial-off-the-Shelf</i> ”
CPU	<i>Central Processing Unit</i>
CS	<i>Carry Select</i>
CST	Teste Concorrente por Varrimento (<i>Scan</i>)
CVM	Circuito Votador por Maioria
dc	Cobertura de Defeitos
DfR	<i>Design for Reliability</i>
DfT	<i>Design for Testability</i>

DfTT	<i>Design for Thermal Testing</i>
DR	<i>Data Register</i>
E/S	Entrada/Saída
ECC	<i>Error Correcting Code</i>
ECL	<i>Emitter Coupled Logic</i>
EDC	<i>Error Detecting Code</i>
EDIF	<i>Electronic Design Interchange Format</i>
Ep	Entrada paralelo
EP	Entrada Primária
Es	Entrada série
fc	Cobertura de Faltas
FF	Elemento de memória (<i>Flip-flop</i>)
fmc	Falhas de modo comum (ou faltas, de acordo com o contexto)
FPGA	<i>Field-Programmable Gate-Array</i>
FTCS	<i>Fault-Tolerant Computing Symposium</i>
IB	Bit de Injecção
IEEE	<i>Institute of Electrical and Electronics Engineers</i>
IP	<i>Intellectual Property Module</i>
IR	<i>Instruction Register</i>
ISCAS	<i>International Symposium on Circuits and Systems</i>
JEDEC	<i>Joint Electron Device Engineering Council</i>
JTAG	<i>Joint Test Action Group</i>
LFSR	<i>Linear-Feedback Shift Register</i>
MCM	<i>Multi-Chip Module</i>
MISR	<i>Multiple Input Shift Register</i>
MTBF	<i>Mean Time Between Failures</i>
MTM	<i>Module Test and Maintenance bus</i>
MTTF	<i>Mean Time to Failure</i>
MTTR	<i>Mean Time to Repair</i>
NMR	Redundância Modular de ordem N
PPI	<i>Pseudo-Primary Input</i> , acessível através da infra-estrutura BST
ppm	partes por milhão

PPO	<i>Pseudo Primary Output</i> , acessível através da infra-estrutura BST
R(t)	Fiabilidade (<i>Reliability</i>)
RAM	<i>Random Access Memory</i>
RCF	Região de Confinamento de Falhas
ROM	<i>Read Only Memory</i>
SOC	System on a Chip
Sp	Saída paralela
SP	Saída Primária
Ss	Saída série
TAP	<i>Test Access Port</i>
TBIC	<i>Test Bus Interface Circuit</i>
TCK	<i>Test Clock</i>
TDI	<i>Test Data In</i>
TDO	<i>Test Data Out</i>
TF	Tolerância a Falhas
TINT	<i>Test Interrupt</i>
TMR	Redundância Modular Tripla
TMS	<i>Test Mode Select</i>
TRST	<i>Test Reset</i>
VHDL	<i>Very High Speed IC Hardware Description Language</i>
VLSI	<i>Very Large Scale Integration</i>
VSD	Votador e Selector de Desacordo
VT	Vector de Teste
WSI	<i>Wafer Scale Integration</i>
XMR	Redundância Modular Incompleta

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

Capítulo 6

Capítulo 7

Capítulo 8

Bibliografia

Anexos

Introdução

1. Introdução

Este capítulo visa apresentar e situar o tema escolhido, elucidar sobre a motivação do trabalho e os aspectos inovadores que integra, terminando com a descrição da forma como a tese se encontra organizada.

1.1 *Âmbito do trabalho e objectivos principais*

Ao longo de cinco décadas, os sistemas Tolerantes a Falhas têm sido utilizados em aplicações que necessitam de um funcionamento confiável, pelo menos durante os períodos em que uma avaria pode ser catastrófica ou ter custos elevados. No entanto, o elevado número de componentes necessários aos mecanismos de redundância implícitos, tem restringido os sistemas tolerantes a falhas às aplicações que justificam, e permitem, o custo acrescido, não só material mas também de projecto, bem como o volume e consumo de energia proporcionais ao encargo global, e que podem ainda aceitar o tempo de vida mais curto que o do sistema base.

A evolução da microelectrónica, aliada à miniaturização, tem permitido a expansão dos sistemas electrónicos a novas áreas, algumas delas críticas, que aumentaram a necessidade de sistemas tolerantes a falhas mais baratos e simples, características que, de certa forma, se tornaram possíveis com os componentes VLSI (*Very Large Scale Integration*). No entanto, se por um lado esta integração é vantajosa em termos de custo e capacidade dos sistemas, por outro, a evolução no sentido de utilizar cada vez menos componentes exige maior fiabilidade individual e, no limite, um sistema implementado num só componente, será inutilizado por qualquer falha que lhe destrua as capacidades de sobrevivência internas, deixando de prestar o serviço especificado. Resulta daqui a necessidade de encontrar soluções para monitorar os subsistemas, de forma tão próxima quanto possível das condições de funcionamento reais, o que pode requerer pontos de teste a circundar cada subsistema, e eventualmente isolá-lo, se avariar. Esta é a filosofia subjacente à utilização neste trabalho da infra-estrutura IEEE 1149.1 [IEEE93], apoiada numa

cadeia de células acedidas sequencialmente por quatro linhas apenas, que permitem observar e controlar (onde necessário) os pinos e pontos de teste convenientes. A aplicabilidade da infra-estrutura BST (*Boundary Scan Test*) aos sistemas confiáveis é aqui proposta com base em três ideias centrais:

- Monitorar o circuito de missão, em regime concorrente, como forma de minimizar a probabilidade de funcionamento com erros.
- Proporcionar características de tolerância a falhas com sobrecargas de confiabilidade que, sendo inseridas no componente, não lhe degradem significativamente o tempo de vida.
- Conservar a compatibilidade com a norma IEEE 1149.1 e, tanto quanto possível, manter custos semelhantes aos dos componentes equivalentes actuais.

1.2 *Originalidade e contribuição inovadora*

Nesta tese são propostas três contribuições, que surgiram de forma evolutiva em resposta às necessidades encontradas no domínio dos circuitos integrados VLSI confiáveis. Embora tendo sido reunidas num circuito integrado único, são na realidade independentes e cada uma tem a sua aplicação específica.

O contexto dos temas tratados nesta tese está representado na figura 1.1, com o núcleo central constituído pelo relacionamento *tolerância a falhas - componentes VLSI* e as três áreas periféricas, com ênfase para a infra-estrutura BST.

Entre parêntesis, estão indicados os capítulos em que se aborda cada uma das áreas, sendo de destacar o capítulo quatro, dedicado aos componentes VLSI, que levantam inúmeras questões na área da tolerância a falhas. Os capítulos cinco e seis documentam as duas principais propostas, e o capítulo sete inclui ainda uma abordagem à fiabilidade da infra-estrutura BST, também tratada genericamente no capítulo dois.

A principal contribuição inovadora pode ser sintetizada no facto de a infra-estrutura BST passar a ser utilizada para novas funções, fornecendo informação do exterior que é associada à tomada de decisões na própria infra-estrutura, o que permite simplificar o encargo nos componentes. Este aspecto é essencial, na medida em que limita o custo e melhora o tempo de vida dos mesmos componentes. Quanto às propostas apresentadas, temos em primeiro lugar, a infra-estrutura CST (*Concurrent Scan Test* ou *Teste Concorrente por Varrimento*), que permite monitorar circuitos de forma concorrente, aproveitando os vectores gerados para o teste de produção.

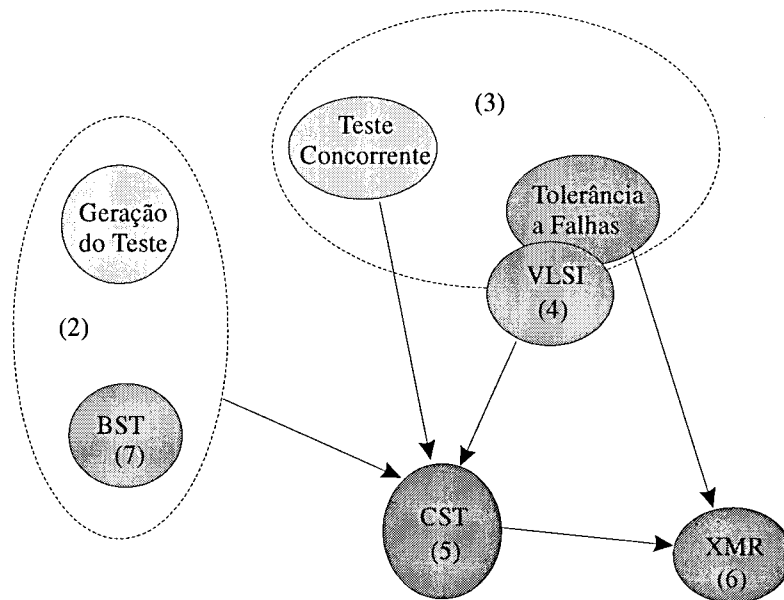


Figura 1-1. Áreas abrangidas por esta tese

A arquitectura XMR (*Redundância Modular Incompleta*), por seu lado, está vocacionada para uma gama de circuitos dedicados a aplicações críticas, que permitem a duplicação do circuito de missão no componente como ponto de partida para tolerar falhas permanentes. Também aqui, os vectores anteriormente definidos para o teste de produção, são aproveitados como forma de levar informação adicional ao integrado, permitindo resolver a indecisão subsequente a situações de erro.

Finalmente, apresentamos o *controlo de paridade* na infra-estrutura BST, uma solução que surgiu em resposta à necessidade de garantir uma maior fiabilidade à informação veiculada através dessa infra-estrutura. Esta proposta é independente das anteriores, podendo ser incluída na norma 1149.1, sem conflitos, e com uma sobrecarga diminuta.

1.3 Organização da tese

Este documento foi dividido em oito capítulos, considerados convenientes para a repartição adequada dos temas a abordar e que podemos agrupar em duas partes. A primeira parte engloba os três capítulos iniciais, onde procurámos sintetizar as bases em que assenta o trabalho desenvolvido. Para além do presente capítulo, introdutório, o segundo apresenta informações genéricas respeitantes à evolução dos sistemas confiáveis e à testabilidade, com breve incursão

nas áreas do teste mais relevantes num trabalho vocacionado para a tolerância a falhas, como as técnicas de projecto para a testabilidade e a infra-estrutura IEEE 1149.1. O terceiro capítulo é dedicado a uma abordagem aprofundada dos conceitos envolvidos no projecto para a confiabilidade e no teste em serviço, análise necessária para fundamentar uma escolha criteriosa das soluções a implementar.

A segunda parte inicia-se com o capítulo quatro, onde se analisam e definem os requisitos dos componentes, com ênfase nas condicionantes de implementação da tolerância a falhas em tecnologia VLSI, tal como as entendemos hoje. O capítulo cinco apresenta a primeira etapa desenvolvida, o Teste Concorrente por Varrimento (CST), e no capítulo seis é explanada a segunda proposta, a arquitectura de Redundância Modular Incompleta (XMR) para circuitos integrados confiáveis, que permite uma simbiose onde o CST é adaptado às arquitecturas de tolerância a falhas mais convenientes, para satisfazer os requisitos essenciais em muitas das novas aplicações de componentes VLSI.

No capítulo sete introduzimos a técnica que melhora a fiabilidade da informação veiculada pela infra-estrutura BST, por recurso ao controlo de paridade, incluindo-a também na arquitectura de um CI que engloba as duas propostas anteriores. No oitavo e último capítulo são apresentadas as conclusões e direcções futuras.

Seguem-se a Bibliografia e os Anexos, que incluem esquemas e diagramas, bem como o relatório de síntese do circuito.

Este documento é ainda acompanhado por um CD-ROM que contém a versão electrónica deste texto (em formatos PDF e PS), bem como os ficheiros utilizados na simulação dos componentes indicados nos anexos.

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

Capítulo 6

Capítulo 7

Capítulo 8

Bibliografia

Anexos

Confiabilidade e Testabilidade

2. Confiabilidade e Testabilidade

Neste capítulo apresentaremos uma introdução às duas áreas que, apesar de terem percursos diferentes, estão hoje, e cada vez mais, interligadas: a tolerância a falhas e a testabilidade.

Um sistema é projectado de acordo com a especificação, que define o serviço a prestar, e quando deixa de o assegurar dizemos que está avariado. A *avaria* resulta de um *erro* nos dados utilizados na máquina de estados do sistema, originado por uma *falha* nos circuitos físicos que a implementam. Estes três factores podem ser caracterizados da seguinte forma [And81]:

- **Falhas:** podem ter causa *física* (interferências externas ou avaria dos componentes) ou *lógica* (se forem causadas por erros de especificação ou de implementação). A falha pode ainda ser caracterizada quanto à natureza temporal como:
 - *Permanente:* a falha instala-se degradando o componente de forma irreversível, quase sempre por se ultrapassarem os limites máximos especificados pelo fabricante.
 - *Temporária:* neste caso, a falha pode ainda ser de dois tipos:
 - *Transitória:* se resulta de influências esporádicas que não danificam o componente a nível físico, de forma permanente.
 - *Intermitente:* se a manifestação ocorre ciclicamente, quando se reúnem condições pontuais que afectam (de forma não destrutiva) o desempenho de um ou mais componentes.

Admite-se que, em média, as falhas permanentes representam entre 1 e 10% do total, dependendo da aplicação, e que as transitórias são responsáveis por mais de 70% das temporárias [Sie91].

- **Erros:** são a consequência das falhas, na informação tratada pela máquina de estados. No caso de esta ter elementos de memória, a informação errada ficará *latente*, até ser utilizada e originar uma avaria, ou ser rescrita (e substituída).
- **Avárias:** situações imprevistas, *catastróficas* ou não, que ocorrem quando um sistema deixa de cumprir o serviço definido na especificação que lhe deu origem.

Temos utilizado o termo *sistema* de forma genérica, sem detalhar a respectiva estrutura, mas no desenvolvimento deste trabalho consideramos, por conveniência de projecto, seis níveis hierárquicos:

- *Transístor*: com interesse circunscrito quase só à fase de produção dos circuitos integrados, este nível está fora da área desta tese.
- *Porta lógica*: importante no teste, este nível é geralmente o mais baixo utilizado no tratamento da tolerância a falhas actual, mas tem um interesse limitado por originar frequentemente soluções com elevada sobrecarga.
- *Bloco Funcional (BF)*: circuito (ou parte dele que pode não ter uma existência física/lógica autónoma) relevante no tratamento da testabilidade por opção de projecto, frequentemente associado ao conceito de *cone testável* e ao de *supergates* [Set85b].
- *Circuito Monitorizado* (ou circuito de *Missão*, *CM*): é o circuito projectado, após identificação dos BF necessários para assegurar as características de testabilidade especificadas. Aparentemente ténue, a distinção entre bloco funcional e circuito monitorizado justifica-se porque a testabilidade supõe (frequentemente implica) partição do circuito, originando blocos funcionais que tendem a ser demasiado pequenos para justificar uma infra-estrutura de teste privativa (*vide* secção 2.3.1).
- *Módulo*: corresponde ao nível de subsistema, geralmente uma carta de circuito impresso (CCI), um componente VLSI ou um módulo multi-componente (*MCM*), e pode conter um ou mais circuitos monitorizados e os mecanismos de confiabilidade associados.
- *Sistema*: o nível de topo, abrangendo qualquer tipo de equipamento digital ou misto, que desempenha uma dada tarefa para que foi projectado e executa ou não um programa.

Neste trabalho, os níveis mais relevantes serão o módulo, o circuito de missão e o bloco funcional, como se mostra na figura 2-1.

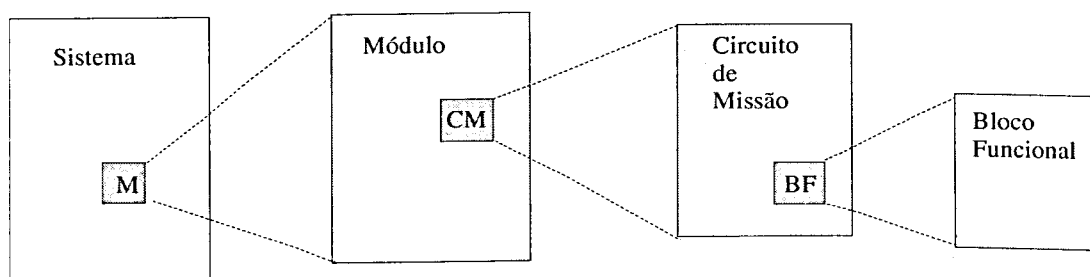


Figura 2-1. Hierarquia de projecto considerada nesta tese

2.1 Introdução à Confiabilidade

Os conceitos *Fiável*, *Confiável*, *Tolerante a Falhas* ou ainda *Robusto*, são frequentemente utilizados como sinónimos, mas o termo *Confiável* (*Dependable computing*), introduzido em [Lap85], é o mais abrangente e identifica um sistema construído para assegurar, *na presença de falhas*, um funcionamento compatível com a especificação. Em relação aos restantes termos, a literatura especializada atribui-lhes o significado que indicamos na Tabela 2-1.

Tabela 2-1. Nomenclatura utilizada em sistemas confiáveis

<i>Terminologia</i>	<i>Características</i>
<i>Componente Fiável</i>	Característica intrínseca: um circuito integrado da série militar é mais <i>fiável</i> que um comercial, por ser sujeito a testes mais completos, senão mesmo a processos de fabrico mais rigorosos.
<i>Sistema Robusto</i>	Um sistema construído com componentes de boa qualidade, <i>fiáveis</i> , é mais <i>robusto</i> que outro idêntico construído com componentes comerciais. Robustez e custo estão directamente relacionadas.
<i>Sistema Tolerante a Falhas</i>	Sistema implementado com recurso a métodos e técnicas destinadas a assegurar comportamentos <i>fiáveis</i> , a partir de componentes de fiabilidade limitada. Esta necessidade resulta de não existirem componentes ideais e de os bons componentes serem caros, para além de que nunca será possível evitar as falhas completamente.
<i>Sistema Confiável</i>	Sistema em cuja construção foram aplicados os cuidados acima referidos e, como tal, é <i>robusto</i> e <i>tolerante a falhas</i> . Neste caso, admite-se o termo <i>Fiável</i> , embora Laprie [Lap85] proponha <i>Confiável</i> como mais correcto e abrangente.

Um componente *fiável* e um sistema *robusto* apenas garantem, dentro de certos limites de distribuição estatística, que a probabilidade de falhar é menor, mas nada podem fazer quando a falha ocorre. Nos sistemas *tolerantes a falhas*, o objectivo é assegurar que existe informação adicional suficiente para atenuar ou corrigir o erro resultante de uma falha, de forma a evitar que se propague à saída do sistema e origine uma avaria. A necessidade das técnicas de tolerância a falhas resulta das limitações dos componentes reais, de fiabilidade limitada, sendo também empregues com o objectivo de permitir reduzir as exigências impostas aos componentes utilizados, possibilitando actualmente a aplicação de componentes comerciais na indústria espacial [Cal98].

2.1.1 *Etapas de sobrevivência*

O processo de sobrevivência, ou de recuperação num sistema tolerante a falhas, pode incluir várias das 10 etapas possíveis referidas a seguir [Avi75, Sie91]:

1. *Detecção*: geralmente implementada com códigos e circuitos de detecção de erros. Pode ser *calendarizada* (ou *intercalada*), efectuada periodicamente com o circuito desactivado, ou *concorrente*, tendencialmente mais rápida nas acções que permite.
2. *Confinamento*: um módulo confiável deve impedir que o erro se propague, pois se não for *confinado* (ou *contido*), pode tornar-se indetectável a jusante, tanto no circuito como na programação. Como se compreende, só a detecção concorrente permite confinamento.
3. *Mascaramento*: se o circuito possuir informação redundante suficiente, um erro pode ser eliminado por votação de maioria, mas a redundância necessária neste processo onera o seu custo e dificulta a localização de falhas, podendo degradar os testes.
4. *Recuperação*: retomar o funcionamento correcto é um processo imediato com o mascaramento, mas nos sistemas baseados na detecção existe geralmente um intervalo de *latência* entre a detecção e o reinício do funcionamento. A recuperação, por sua vez, pode ser *completa*, *degradada*, ou apenas o *fecho em segurança*.
5. *Diagnóstico*: é o processo de análise ao nível da falha, só justificado nos casos em que o sistema pode aproveitar esta informação para desactivar o componente avariado, ou para o substituir por outros disponíveis.
6. *Repetição*: na maioria dos casos, em que a falha é temporária e o diagnóstico dispensável, pode ser suficiente recuperar o funcionamento de um ponto prévio, em que a informação é reconhecida como correcta.
7. *Reinicialização*: forma limite de recuperação, necessária quando a falha, não destrutiva, se mantém enquanto o sistema estiver alimentado. Neste caso, desligar a alimentação é geralmente a única solução.
8. *Reparação*: se existir, o processo de reparação deve permitir retirar o módulo avariado e substituí-lo por um novo, de forma transparente, seja por processos manuais ou automáticos.
9. *Reconfiguração*: falhas permanentes obrigam a desactivar o módulo avariado, para evitar que o erro se propague, independente de estar prevista a reparação ou a passagem a um modo de funcionamento em degradação controlada.
10. *Reintegração*: após um módulo de reserva ser integrado, o sistema deve ser testado, em particular se funcionou em modo degradado, e desejavelmente de forma transparente.

2.1.2 Breve história da Tolerância a Falhas

As primeiras referências aos sistemas “binários” tolerantes a falhas surgem na década de 40, com os equipamentos de comutação a relés para linhas telefônicas, onde a preocupação central era evitar uma frequência (*taxa*) de avarias demasiado elevada para as aplicações em vista [Car87, Max87]. Os sistemas electrónicos com válvulas desenvolvidos durante a Segunda Grande Guerra para criptologia e cálculos balísticos, vieram colocar novas exigências e originar um incremento nas técnicas de análise e tratamento da tolerância a falhas dos sistemas digitais. Sendo as válvulas muito mais rápidas que os relés, a complexidade dos cálculos possíveis passa a ser restringida pela *taxa de falhas* (na ordem de 1 ppm), que obriga a implementar técnicas de tratamento de erros [Avi87]. Para o computador ENIAC [Bur47], são referidos períodos de 5 a 10 minutos em funcionamento, intercalados com outros de 10 a 20 minutos para reparação.

A análise efectuada por von Neumann em 1956, estabelecendo as bases dos sistemas *digitais* com redundância de informação para sobreviver a falhas dos componentes, é apontada como a origem dos métodos de tratamento sistemático da tolerância a falhas (em [Iye95]), dando origem ao conceito de *sistema digital fiável construído com componentes não fiáveis*. Vinte e dois anos mais tarde, A. Avizienis publica uma análise extensiva da tolerância a falhas intitulada “*Fault-Tolerance: The Survival Attribute of Digital Systems*” [Avi78].

O aparecimento do transistor (1948) e posteriormente do circuito integrado (1959), com taxas de falhas milhares de vezes inferiores às das válvulas, melhorou a fiabilidade dos sistemas para níveis que tornaram desnecessária a implementação de unidades tolerantes a falhas, uma vez que o (ainda pequeno) número de componentes envolvidos permitia uma confiabilidade suficiente e não justificava o acréscimo de custo, complexidade de projecto e componentes adicionais. Na origem do impulso verificado nesta área, que ainda hoje se mantém, esteve o programa espacial tripulado americano, onde as avarias, maioritariamente catastróficas, justificavam todos os custos de projecto dos sistemas tolerantes a falhas.

A contínua redução de custo levou os sistemas digitais para fora das salas de ambiente controlado: unidades industriais e automóveis são exemplo de aplicações onde o ruído, electromagnético e nas linhas de alimentação, a temperatura, humidade e poeiras, põem cada vez mais exigências. Por seu lado, a crescente responsabilização das tarefas que estes equipamentos desempenham, em todos os níveis em que se inserem, reactivou a importância da confiabilidade num número crescente de sistemas digitais.

Há muitas datas ou épocas relevantes na história da tolerância a falhas, entre as quais escolhemos as que em nossa opinião são mais representativas:

1946 - Na Grã-Bretanha são relatados os problemas e possíveis soluções nas primeiras unidades de comutação telefónica a relés.

1947 - ENIAC: é construído o primeiro “computador” com válvulas, que apresentava em média uma falha por milhão de operações.

Década de 50 - Projecto do foguetão Saturno V: a aceleração e vibração impõem elevadas cargas mecânicas aos componentes, discretos e de massa apreciável. O “computador” de voo utiliza componentes em configuração quádrupla, permitindo a falha de dois deles. O objectivo é não falhar nos minutos do lançamento.

1956 - John von Neumann estabelece a noção de *redundância de informação digital*, originando o conceito de sistemas com lógica binária tolerante a falhas, construídos com componentes falíveis.

1959 - A Texas Instruments fabrica o primeiro circuito integrado.

Década de 60 - O programa espacial americano tripulado traz novas exigências e canaliza elevadas verbas para a tolerância a falhas. Na cápsula Apolo, confiabilidade e baixo consumo de energia são fundamentais.

O supersónico Concorde é projectado com o primeiro sistema de voo comercial controlado por computador.

Década de 70 - O supercomputador Cray1 avaria 10% do tempo de processamento e o elevado custo das paragens justifica todo o investimento possível em melhorar a tolerância a falhas. Projecto do Vaivém espacial americano.

1971 - O primeiro *Symposium on Fault-Tolerant Computing (FTCS)* propõe-se fazer o ponto da situação na tolerância a falhas e procurar soluções pragmáticas.

1978 - “*Fault-Tolerance: the Survival Attribute of Digital Systems*” de A. Avizienis constitui um marco na tolerância a falhas, integrando uma análise global e conduzindo a uma perspectiva sistemática.

Década de 80 - Os circuitos VLSI trazem vantagens e novos problemas à tolerância a falhas: soluções baseadas na replicação interna de componentes têm baixo custo mas são sensíveis aos defeitos de fabrico. Primeiros projectos de aviões comerciais sem actuação mecânica entre a consola e os actuadores (*fly-by-wire*).

1991 - Na edição desse ano da FTCS, organiza-se um grupo de trabalho para fazer o ponto da situação. As conclusões, apresentadas na edição do ano seguinte, referem que a tolerância a falhas continua a ser orientada pelos objectivos imediatos da aplicação em vista, o que impede uma abordagem integrada.

Para além de inúmeros marcos que ficam por referir, ao longo da década actual há também diversos casos dignos de nota. No entanto, aqueles que consideramos mais relevantes no domínio dos CI tolerantes a falhas, serão apresentados como exemplos na secção 3.2.4.

2.1.3 Sumário de custos e benefícios

A tolerância a falhas é a capacidade de sobrevivência do sistema com características de construção especiais, resultantes da inclusão no projecto de unidades redundantes (físicas ou de informação), que permitem evitar a avaria, restringindo ou eliminando as consequências das falhas. Uma vez que a redundância não melhora o desempenho normal do sistema, a implementação da tolerância a falhas tem de ser encarada num ponto de equilíbrio entre o elevado custo inicial e os benefícios daí decorrentes, por reduzir o número das avarias e os custos da manutenção correctiva e, provavelmente mais importante que os restantes, por evitar o custo indirecto das falhas. No caso de vidas humanas, este custo justifica certamente o melhor sistema possível.

A favor de sistemas tolerantes a falhas podemos considerar o facto de a reparação e manutenção assumirem custos cada vez mais elevados, fazendo com que valha a pena evitar a paragem do sistema, pelo menos durante algum tempo e mesmo que funcionando em degradação controlada. Em paralelo, a cada vez mais rápida obsolescência dos equipamentos faz com que os técnicos encontrem poucas vezes avarias idênticas, o que aumenta o tempo de localização das falhas, caso não haja métodos auxiliares de diagnóstico. Por outro lado, os custos da manutenção e reparação dependem da situação em que são efectuados, com destaque para os sistemas móveis e todos aqueles em que períodos de trabalho intenso alternam com períodos de repouso (ou menos exigentes). Refira-se também, que o custo médio da reparação de um componente defeituoso é multiplicado por uma ordem de grandeza, conforme a fase em que a avaria é detectada [Tex95a]:

- no teste do CI $x1$
- no teste da CCI $x10$
- no teste do sistema $x100$
- no utilizador final $x1000$

Adicionalmente, tendências recentes de contenção de custos e a cada vez maior necessidade de ter os sistemas prontos em prazos curtos, têm levado os projectistas a utilizar componentes comerciais mesmo na indústria aeroespacial, onde os prazos actuais de desenvolvimento de um satélite são de 18-24 meses e já não de 5-10 anos, como nas décadas passadas [Cal98].

Entre as desvantagens estão a maior complexidade do projecto, que encarece o desenvolvimento e origina maior tempo de execução, bem como a sobrecarga de confiabilidade, que aumenta a probabilidade de falhas e afecta o tempo de missão. Em consequência, e para reduzir o custo inicial, muitos sistemas são projectados sem redundância física, com tolerância só a falhas temporárias. Acresce por outro lado a dificuldade do teste, no que respeita à geração dos vectores de teste e ao seu tempo de aplicação, dado que as técnicas de tolerância tendem a mascarar as falhas, assunto que nos leva a considerar o domínio da testabilidade.

2.2 Testabilidade

O ciclo de vida de um circuito pode ser dividido em duas grandes fases: a de projecto/produção e a de vida útil, esquematizadas na figura 2-2. A testabilidade incide geralmente na primeira, enquanto a tolerância a falhas lida com a utilização dos componentes durante a sua vida útil. Em termos simplistas, podemos dizer que na primeira fase pretendemos *localizar* as falhas (ou *defeitos*, num sentido mais lato), mas durante a vida útil, se a monitorização for implementada, é suficiente detectá-las (ou os erros que originam).

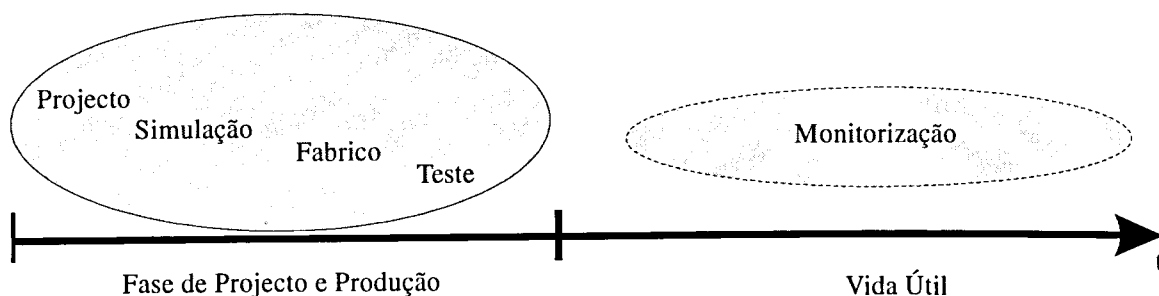


Figura 2-2. As duas grandes fases da vida de um circuito

O *teste* pode ser definido como o processo de estimular as entradas de um sistema e analisar as saídas, para verificar se o funcionamento corresponde à especificação. Se forem detectados erros e houver interesse, procede-se ao *diagnóstico*, com estímulos específicos para localizar a causa.

A verificação do projecto pode ser efectuada num *protótipo*, que permite testes em tempo real mas é moroso e dispendioso, podendo as alterações ser complexas. Além disso, um protótipo construído com componentes discretos não é representativo de um circuito implementado em geometrias micrométricas, onde projecto e tecnologia são indissociáveis [Abre98]. Na *simulação*, o protótipo é substituído por um modelo do circuito, que usualmente inclui dois tipos de informação:

- *Funcional*, que define as relações lógicas.
- *Temporal*, que estabelece as relações dependentes dos atrasos dinâmicos.

Esta separação facilita o tratamento independente de ambos os modelos e permite diferentes implementações com base num mesmo modelo funcional, embora seja necessário verificar os aspectos temporais subsequentes a cada implementação [Sch92, Sim94]. O teste pós-fabrico deve incidir em todo o sistema, abrangendo os níveis da carta de circuito impresso e do circuito integrado, e entre as principais causas de erro estão a especificação incompleta ou inconsistente, os defeitos de fabrico, as falhas físicas e as influências externas. Em resultado, são utilizados vários níveis de teste, desde os paramétricos, que dependem da tecnologia e são importantes no fabrico, onde as falhas podem ser complexas [Red85, Ren98b], passando pelos de instrução, para microprocessadores, que podem recorrer ao tratamento a nível de transferência entre registos [Goo90], até ao nível de sistema, centrado nas interligações e transferência de informação entre os módulos constituintes [Cha86], que pode abranger complexos protocolos de comunicação, sempre que existe redundância.

Neste trabalho interessam essencialmente os testes de natureza lógica, que podem ser *funcionais*, se baseados na especificação de funcionamento do sistema, ou *estruturais*, se lidam com a informação da interligação entre as primitivas de baixo nível que integram o sistema. Os testes estruturais são executados na fase de produção e por vezes como teste inicial, que o sistema executa sempre que é inicializado. Durante a vida útil dos sistemas o teste em serviço é geralmente funcional, como forma de monitorização, e pode ser executado em regime:

- Calendarizado.
- Concorrente.

Os testes de tipo calendarizado são efectuados em intervalos temporais determinados, entre fases de funcionamento normal. Podem ainda ser *intrusivos*, se suspendem a operação normal do circuito, ou *não-intrusivos*, se aproveitam intervalos de relaxamento na actividade deste, uma solução que tem sido explorado nas aplicações em que a gestão de energia é relevante [Flo98a].

Nos testes concorrentes há dois processos fundamentais:

- Comparar as saídas, e outros eventuais pontos de teste, com uma unidade de referência.
- Verificar propriedades invariantes dos blocos funcionais.

As propriedades que interessa verificar não existem de forma inata nos sistemas e são difíceis de obter, implicando técnicas de projecto que consideraremos no capítulo seguinte. Por agora, e uma vez que o processo de teste, a nível lógico, se destina a analisar as falhas, vamos abordar o tratamento destas.

2.2.1 *Modelos de Falhas*

A diversidade de falhas que têm de ser analisadas num circuito dificulta um tratamento adequado e torna conveniente passar a um nível de abstracção que permita representá-las de forma simplificada. Foi esta necessidade que conduziu ao desenvolvimento de *modelos de falhas*, que simplificam a geração dos vectores de teste, mas normalmente à custa de se omitir informação sobre as características das falhas [Hay85, Hay87], que pertencem maioritariamente a três grupos [Wad78, Cou81]:

- Curto-circuito à alimentação.
- Curto-circuito entre linhas de sinal (dados ou controlo).
- Circuitos-abertos.

As primeiras têm como consequência forçar os nós afectados a valores próximos da massa ou de V_{cc} e são tratadas convenientemente pelo modelo de falhas *sempre-a*, ou $s@$ [Sch72], mas os outros dois casos dependem da tecnologia envolvida. Os curto-circuitos entre linhas de sinal alteram a topologia do circuito e geram novas funções lógicas, que podem transformar um circuito combinatório em sequencial, ou um síncrono em assíncrono, pelo que são tratados à parte como falhas *em ponte*. Os circuitos-abertos poderão ser interpretados como 0 ou 1, dependendo da tecnologia de implementação, mas podem igualmente originar valores indeterminados, incluindo oscilação.

Ao considerar um modelo de falhas há que distinguir as duas grandes fases da vida do componente:

- Fase de Projecto e Produção: os defeitos de fabrico conduzem frequentemente a falhas múltiplas, pelo que um modelo limitado a falhas simples pode ser inadequado. Esta situação é tida em conta pelos fabricantes, em particular para falhas que interagem entre si, e conduz a modelos adaptados a essas condições [She85].

- Vida útil: teoria e experiência permitem admitir que na fase de vida útil as falhas ocorrem uma de cada vez, aleatoriamente [Cou81, Roq98], e em condições de funcionamento normais podemos aceitar modelos de faltas simples. Nesta fase os três grupos indicados acima continuam válidos, embora os processos de degradação que ocorrem durante o funcionamento [Cou81], façam com que falhas passem normalmente por uma fase resistiva, antes de se tornarem francas.

Na tolerância a falhas interessa-nos sobretudo o segundo caso, sendo necessário definir com rigor os tipos de falhas que um sistema deve tolerar: um modelo amplo origina custos elevados, em tempo de projecto e de execução, podendo ainda inviabilizar uma depuração completa [Fav91, Gho91]. Esta abordagem é no entanto desnecessária, se for implementada uma *estratégia de teste frequente*, que permita a reparação antes de ocorrer segunda falha, e que é geralmente prevista nos sistemas confiáveis.

O modelo de faltas *sempre-a*, cujas faltas $s@v$, $v \in \{0,1\}$, se referem aos valores lógicos 0/1 atribuídos à tensão (dentro das margens de ruído correspondentes), é o mais utilizado devido à facilidade de tratamento e ao facto de permitir modelar um grande número de outras falhas [Fri74]. Contudo, este modelo tem sido questionado na tecnologia CMOS, particularmente nas geometrias submicrométricas, que originam falhas de modelação complexa [Fav91, Zem96, Abre97]. Casos análogos, também resultantes da evolução técnica, foram resolvidos com novos encapsulamentos e melhores processos de fabrico [Abre97], mas abaixo de $0,25\mu\text{m}$ as distribuições estatísticas tornam-se esbatidas, podendo deixar de ser válidas [Fer98], e nas geometrias de 50 e 30 nm, investigadas por grandes fabricantes, são necessárias novas orientações nos métodos de produção [Cali98].

O modelo de faltas *sempre-a* presume curtos-circuitos francos (ou pelo menos bem definidos), mas certos defeitos resistivos e algumas condições iniciais podem gerar valores indefinidos (u), cuja representação exige três estados e requer tabelas de verdade próprias [Ska95, Ren98b]. No entanto, apesar das limitações referidas, este modelo continua a ser útil na tolerância a falhas, devido à lentidão relativa dos mecanismos de degradação acima referidos.

• Outros Modelos

Para além dos modelos anteriores, pode haver necessidade de considerar modelos para faltas *múltiplas*, com interesse na produção, onde os defeitos de fabrico são frequentemente múltiplos e podem conduzir a falhas latentes que afectam a detecção de outras. Este tipo de falhas é, no

entanto, pouco relevante em operação, excepto em sistemas com unidades replicadas, uma vez que podem ocorrer *falhas* (e subsequentes *faltas*) *de modo comum (fmc)*.

Num circuito com n nós, há $3^n - 1$ faltas múltiplas possíveis, tornando este modelo demasiado oneroso e restringindo-o a ser considerado só na hipótese de *mascamamento*, que ocorre quando uma falta $f2$, posterior a outra $f1$, origina uma falta múltipla que impede a detecção de $f1$. Adicionalmente, os algoritmos que permitem detectar este tipo de faltas têm limitações em circuitos com derivações internas (*fan-out*), que constituem o caso geral [Fuj85].

Durante a vida útil, a *acumulação* é a principal causa de falhas múltiplas, mas esta situação pode ser ultrapassada com testes a intervalos curtos, muito inferiores ao da provável ocorrência de duas falhas consecutivas. Por último, deve-se ter em conta que uma falta múltipla tem elevada probabilidade de ser detectada pelos vectores de teste que detectam qualquer das faltas individuais que a compõem.

As faltas *em ponte* resultam geralmente dos defeitos de fabrico e, na medida em que alteram a topologia do circuito, não podem ser representadas pelas faltas *sempre-a*. O modelo usual para este tipo de faltas, associa ao curto-circuito entre linhas de sinal uma nova função $Z(x, y, \dots)$, que tem a propriedade $Z(a, a) = a$, mas na qual $Z(a, a)$ depende da tecnologia, sendo geralmente indeterminada em CMOS (entre duas linhas apenas). Apesar destas faltas poderem criar malhas de realimentação e transformar um circuito combinatório em sequencial, alterar um sequencial ou mesmo gerar oscilações, são facilmente detectadas por muitos dos testes que detectam faltas *sempre-a*, na maioria dos casos por um só vector [Abr85, Xu85].

O tratamento dos atrasos na propagação recorre também a modelos que os separam da função lógica, destacando-se para as portas lógicas os modelos de *atrasos de transporte* e os de *inércia*. Este tipo de modelos pode ser alargado a blocos funcionais, com atrasos diferenciados entre cada par entrada-saída, e às pistas, que actualmente podem introduzir atrasos superiores ao das portas lógicas, particularmente na tecnologia CMOS [Dag98]. Existem ainda modelos para falhas temporárias, que implicam conhecer a respectiva probabilidade de ocorrência, para determinar quantas vezes repetir o teste de forma a obter uma probabilidade de detecção p . No entanto, o tratamento deste tipo de falhas é feito de preferência em regime concorrente [Kei97].

• *Modelos de Faltas Realistas*

Os modelos de faltas *sempre-a* e *em ponte* assumem faltas *equiprováveis* mas, conhecendo-se a arquitectura do circuito, muitas delas podem ser eliminadas por impossíveis ou improváveis, reduzindo o número de faltas a tratar. Os modelos que consideram estes aspectos, dificilmente

são aplicáveis na simulação e, *a posteriori*, o seu custo raramente se justifica, mas as directivas que geram podem orientar o projecto no sentido de facilitar a testabilidade. Este tema é complexo na medida em que interagem múltiplos parâmetros, mas diversas tentativas de associar projecto e testabilidade mostraram resultados promissores, que se estendem à área do projecto conjunto de código e circuitos dedicados (*Hardware/Software co-design*) [Cel96, Mar97, Dia98].

Outros modelos consideram as características *resistivas* das falhas, que podem ser interpretadas de forma distinta pelas entradas lógicas, uma vez que tendo funções de transferência diferentes, dão resultados dependentes da entrada usada e originam faltas de comportamento variável com a temperatura, com reflexos também no domínio temporal [Mut76, Sant97, Ren98b].

Apesar das vantagens potenciais destes modelos, foi demonstrado que o modelo de faltas *sempre-a* permite resultados próximos dos que outros modelos mais complexos proporcionam, mesmo para circuitos sequenciais, com um custo computacional entre uma e duas ordens de grandeza inferior, para além de que os modelos de faltas mais complexos nem sempre podem ser aplicados [Cor97].

2.2.2 Geração do Teste

Nesta secção, pretendemos abordar a área que mais interessa ao processo de teste concorrente em que assenta uma das propostas desta tese: a minimização do custo de aplicação. Entenda-se por teste *T* um conjunto de vectores de teste, que inclui os *estímulos* e as *respostas* esperadas, e que pode ser gerado para detecção ou para diagnóstico de faltas, com base em dois processos distintos:

- Especificação funcional: é independente da implementação, mas nos processos conhecidos conduz à duplicação da informação e, sendo utilizados vectores de teste, o seu número é geralmente elevado.
- Simulação em modelo: o conhecimento da estrutura do circuito permite obter testes mais compactos, mas impõe geralmente restrições nas faltas tratáveis.

A duplicação da informação útil inerente ao primeiro processo, permite um teste completo com a vantagem de dispensar um modelo de faltas. No segundo caso, ao reduzir o número de vectores de teste, reduz-se também a informação contida em *T*, e teremos de aceitar alguma indefinição na detecção das faltas. Uma vez que a duplicação é geralmente impraticável nos CI correntes,

devido ao custo e sobrecarga associados, recorre-se a testes compactos, sendo necessário avaliar a qualidade de T , determinando a respectiva *cobertura de defeitos*, definida como:

dc = *probabilidade de qualquer defeito no circuito ser detectado por T .*

Este parâmetro, que equivale à cobertura de falhas, é difícil de calcular [Cel96], pelo que na prática se determina a *cobertura de faltas* do teste para um modelo, definida como:

fc = *razão entre o número de faltas detectadas pelo teste e o total de faltas possíveis, de acordo com o modelo utilizado.*

Apesar de fc e dc estarem correlacionadas e geralmente próximas, um teste T com cobertura de faltas de 100% não assegura a detecção da totalidade das falhas, nem de faltas exteriores ao modelo. Por outro lado, qualquer teste compacto deve ser classificado em função de dois parâmetros:

- *Custo de geração*: Sendo efectuado uma só vez, este custo é fixo, depende da complexidade do circuito, beneficia das técnicas de projecto para a testabilidade e dilui-se pelo número de circuitos fabricados.
- *Qualidade*: determinar a qualidade do teste pode ser tão complexo como gerá-lo, sendo mais relevantes nesta avaliação os seguintes factores:
 - *Cobertura de faltas*: idealmente de 100%.
 - *Custo de aplicação*: quanto mais compacto, mais rápido e menos oneroso.
 - *Consumo de energia*: relativamente recente, este parâmetro é importante nos equipamentos portáteis, associado à geração de testes compactos e optimizados, bem como à minimização do ruído eléctrico gerado pelo teste [Flo98a], que pode ser crítico em equipamentos médicos inseridos no paciente.

A qualidade do teste influencia a qualidade do produto final, entendido como os componentes considerados bons e vendidos como tal. Na impossibilidade de o teste ser exaustivo, a dispersão que sempre existe origina a aprovação de componentes parcialmente defeituosos e a rejeição de uma pequena fracção de componentes bons. A tradução matemática dos resultados experimentais associados a estes factores, resulta na equação [Wil81]:

$$D = 1 - Y^{(1-d)}$$

Esta equação, onde Y é o rendimento do processo de fabrico (*yield*), D a probabilidade de aceitar como bom um circuito integrado defeituoso e d a cobertura de defeitos, evidencia que as exigências colocadas ao teste variam numa relação inversa com a qualidade do processo de fabrico, sob pena de uma percentagem elevada de componentes defeituosos escapar aos testes de

produção. Esta equação é ainda extensível ao teste das cartas de circuito impresso, onde a utilização de componentes defeituosos tem efeitos cumulativos [Agr88].

Na realização de um teste concorrente, que constitui o objectivo que temos em vista, interessa minimizar o custo de aplicação e, como tal, interessa-nos determinar um teste compacto, derivado da simulação e com o menor número de vectores possível [Ste91, Arm93]. Para analisar a minimização do custo de aplicação do teste, começemos por uma referência aos métodos básicos de o gerar.

- ***Geração Aleatória***

Esta abordagem baseia-se na geração de vectores por algoritmos pseudo-aleatórios, independentes da estrutura do circuito, que geram testes com um número de vectores várias vezes superior ao de um teste determinístico, para cobertura de faltas equivalente. As principais restrições a este tipo de testes, são a limitação aos circuitos combinatórios e a impossibilidade de garantir num intervalo de tempo limitado uma cobertura de faltas pré-fixada, que geralmente pode apenas ser estimada a partir da probabilidade de detecção, com base nas probabilidades dos sinais [Par75, Sav84, Dav98].

- ***Geração Determinística***

A geração determinística recorre ao modelo estrutural do circuito, sendo por isso mais complexa e morosa, mas gera testes mais curtos e garante a cobertura de faltas especificada, podendo ou não ser orientada para faltas específicas [Abr84].

- *Geração independente das faltas*: é uma solução de complexidade intermédia, útil na fase inicial do processo de geração para obter um teste que determine um grupo alargado de faltas, sem visar qualquer uma individualmente. Os algoritmos utilizados aproveitam o facto de que metade das faltas *sempre-a* que podem ocorrer no percurso crítico de um vector de teste, serem sempre detectadas, pelo que procuram gerar vectores que produzam longos percursos críticos.
- *Geração orientada à falta*: neste caso, o algoritmo tem por objectivo gerar vectores para faltas específicas, quase sempre residuais por não terem sido detectadas pelos métodos anteriores. Na maior parte dos casos, este tipo de problemas deve-se à existência de redundância introduzida pelas derivações internas e é tratado recursivamente por tentativa e erro (*backtracking*), sendo hoje conhecidos algoritmos com heurísticas eficientes [Flo98a].

• *Geração Mista*

Um sistema típico de geração automática do teste tem como objectivo maximizar a cobertura de faltas e minimizar o custo, bem como o número de vectores de teste, e geralmente engloba mais do que um dos métodos anteriores: primeiro, um algoritmo independente das faltas é utilizado para se obter um teste primário com cobertura de faltas entre 60 e 80%, em função das características do circuito, e na segunda fase um algoritmo orientado à falta procura vectores de teste para as faltas residuais. Desta forma, e optimizando a repartição entre os dois algoritmos, minimiza-se o tempo e os custos de geração.

• *Algoritmos para Geração do Teste*

Qualquer algoritmo determinístico envolve decisões e, para as orientar, necessita de *funções de custo*, que são definidas para avaliar a testabilidade de nós. A função ideal é o cálculo exacto mas, tendo custo idêntico à solução, leva ao recurso a estimadores que fornecem apenas medidas relativas, particularmente duas:

- *Controlabilidade*: estima a dificuldade relativa de forçar num nó um determinado valor, permitindo seleccionar o percurso mais longo¹ e, para este, começar pela entrada que permite melhor controlabilidade.
- *Observabilidade*: avalia a dificuldade relativa de propagar o valor lógico presente num nó até uma saída primária, permitindo seleccionar primeiro as portas lógicas que facilitam mais essa propagação.

Estes estimadores podem ser *estáticos* ou *dinâmicos*, neste último caso sendo continuamente recalculados pelo algoritmo, para as portas lógicas ainda não ensaiadas. No entanto, ambos devem ter custo inferior ao benefício que trazem e, geralmente, são usadas funções de custo simplificadas, como as baseadas na distância. Estas medidas assumem que as entradas têm controlabilidade máxima e as saídas a melhor observabilidade e, em termos aproximados, estes custos crescem com a profundidade de cada nó [Mur88].

Uma análise de vários algoritmos encontra-se em [Agr88], sendo hoje aceite que as regras essenciais foram já desenvolvidas e que não se esperam saltos qualitativos apreciáveis. Entre os algoritmos tradicionais realçamos o *Algoritmo D*, originalmente introduzido em [Rot66], o *PODEM* (*Path-Oriented Decision Making*) [Goe81] e o *FAN* (*Fanout-Oriented Test Generation Algorithm*) [Fuj83]. O PODEM e o FAN são por vezes ainda tidos como referência, quer no

¹ - O percurso que permite detectar maior número de faltas.

tempo de geração quer nos vectores de teste que produzem. Na tabela 2-2 indicamos alguns resultados obtidos com o algoritmo FAN em diversos circuitos, cuja descrição se pode encontrar em [Fuj85]. Os aspectos que nos interessa realçar neste trabalho são a cobertura de faltas e o pequeno número de vectores de teste ($\#VT$), face ao tamanho de um teste exaustivo ($\#V_i =$ número de vectores de entrada), assunto que analisaremos com mais detalhe no capítulo quatro.

Tabela 2-2. Exemplo de resultados obtidos com o algoritmo FAN

Circuito	# Portas lógicas	# Nós	# EP	# SP	#Faltas	$fc \%$	#VT	$\#V_i$ (*)
Lógica de codificação	718	1925	33	25	1871	99,52	151	2^{33}
ALU	1003	2782	233	140	2748	95,74	159	2^{233}
ALU	1456	3572	50	22	3428	96,00	215	2^{50}
Controlo de ALU	2002	5429	178	123	5350	98,90	195	2^{178}
ALU	2982	7618	207	108	7550	98,20	283	2^{207}

(*) Supondo um teste exaustivo.

Actualmente, as máquinas com centenas de MIPS² permitem gerar vectores de teste com um custo não significativo [Ham98a] e centrar a optimização em aproximações integradas [Bha89, Tei91, Mar97, Dia98], bem como em algoritmos com heurísticas eficientes [Flo98b, Bom98]. Outros processos de reduzir o custo de geração são os aceleradores físicos acoplados ao computador hospedeiro, com arquitecturas total ou parcialmente dedicadas, que permitem ganhos de uma ordem de grandeza [Alv97, Chi98], ou particionar o circuito, ou grupo de faltas, e distribuir a geração por n processadores em rede.

2.2.3 Minimização do custo de aplicação do teste

Ao efectuar um teste concorrente numa aplicação de tolerância a falhas, procuramos saber se o circuito funciona bem, não sendo, em caso contrário, relevante discriminar a causa para tratar o erro de forma imediata. Só nos interessa portanto o teste de detecção, mas os requisitos inerentes a este tipo de aplicações incluem ainda a necessidade de se minimizar o tempo de aplicação, mesmo que à custa de um acréscimo no custo de geração. A optimização é efectuada com técnicas de compactação que podem incidir no tratamento das faltas ou nos vectores de teste.

² - MIPS: Milhões de instruções por segundo.

- *Tratamento de faltas*

Este tipo de tratamento é essencialmente vocacionado para reduzir o custo de geração do teste, com técnicas de colapso e amostragem.

- *Colapso de faltas*

O número de faltas que é necessário detectar, pode ser reduzido em relação ao universo do modelo, por colapso de faltas, efectuado a dois níveis:

- *Colapso por equivalência de faltas*: as faltas funcionalmente equivalentes originam saídas iguais e nenhum teste permite distingui-las (as faltas f e g são funcionalmente equivalentes num teste T sse as saídas $Z_f(t)=Z_g(t)$ ³, para qualquer vector $t \in T$). Embora seja um problema NP-completo, este tipo de colapso é útil na detecção, uma vez que permite considerar somente uma representante de cada classe de faltas equivalentes, reduzindo a cerca de metade o número de faltas a tratar.
- *Colapso por dominância de faltas*: dado um teste T_g que detecta a falta g , diz-se que uma falta f domina g se são ambas funcionalmente equivalentes sob T_g . Assim, deve ser escolhido um modelo cujas faltas sejam dominadas pelas de outros modelos, permitindo detectar faltas não explicitamente procuradas. O interesse no modelo *sempre-a* advém de estas faltas serem dominadas pela maioria das outras, sendo a excepção mais relevante as que introduzem, ou alteram, malhas de realimentação.

É possível estimar o número de faltas *sempre-a* num circuito [Abr90] e dessa análise conclui-se que para circuitos grandes (aqueles em que este processo é mais importante e que apresentam também valores médios mais estáveis [Brg85]), o colapso permite reduzir o número global de faltas a tratar em cerca de 50% [Agr88]. Como termo de comparação, o modelo para faltas em ponte conduz a um número de faltas possíveis muito superior e, tomando por exemplo um circuito com 5000 portas lógicas, implicará tratar cerca de 500 vezes o número de faltas colapsadas do modelo *sempre-a*.

- *Amostragem de faltas*

Tendo em conta que a complexidade dos circuitos tem crescido continuamente, o número de faltas é elevado e frequentemente são tratadas por amostragem. Num circuito com M faltas após

³ - $Z_i(t)$ representa a função lógica implementada pelo circuito, quando a falta i está presente.

colapso, sendo K o número das que são detectadas pelo teste T , teremos uma cobertura de faltas F dada por K/M . Para reduzir o custo da geração do teste, pode-se simular apenas faltas de uma amostra aleatória m , desejavelmente $m \ll M$, e neste caso, para o universo m da amostra, determinaremos de forma equivalente k e $f=k/m$. O objectivo é relacionar f e F , conhecidos m e M , ou mais precisamente determinar a relação que m deve satisfazer face a M , para que f esteja dentro de uma margem de erro ε de F [Set85a, Dae91]. A análise de distribuição efectuada em [Abr90], supondo $m \ll M$, permite determinar o erro de aproximação através da seguinte equação:

$$\varepsilon = 3 \cdot [F(1-F) \cdot (1/m - 1/M)]^{1/2}$$

Esta mesma análise permite inferir um erro quase independente de M , especialmente para coberturas de faltas elevadas, sugerindo que a amostragem é um processo válido, embora sem permitir obter informação acerca das faltas excluídas da amostra.

Outra alternativa aos métodos de simulação exactos é a simulação estatística, em que se estima a probabilidade de cada falta ser detectada e a partir daí a cobertura de faltas global [Set82, Jai85, Kei97].

• *Tratamento dos Vectores de Teste*

Em relação aos vectores de teste, as duas técnicas de redução do custo de aplicação mais usuais são a compactação e o teste pseudo-exaustivo.

• *Compactação dos vectores de teste*

Os vectores de teste obtidos por geração orientada são, em regra, apenas parcialmente definidos, o que permite compactar vectores que não colidam, num processo que pode ser *estático* ou *dinâmico*. O primeiro, efectuado *a posteriori*, não introduz perda de informação, mas a determinação do grupo óptimo de vectores de teste é um processo quase sempre heurístico [Hsi99]. Na compactação dinâmica, cada vector é reutilizado após ser gerado, na tentativa de aproveitar os bits não definidos para lhe aumentar a capacidade de detecção. Entre os algoritmos de geração de teste recentes, destacamos pelo interesse particular que as abordagens propostas têm no teste concorrente, os seguintes:

- COMPACTEST: algoritmo baseado em dicionários de faltas e exaustivo [Pom91, Pom97].
- MTP: é um algoritmo de compactação dinâmica que procura um teste mínimo, tanto em número de vectores de teste como na actividade forçada no circuito [Flo98b].

- RIBTEC: algoritmo que desenvolve técnicas de alto nível com o objectivo de explorar a testabilidade de núcleos IP internos [Kim98a]. Este algoritmo gera, para alguns circuitos descritos nessa referência, os resultados apresentados na tabela 2-3, entre os quais destacamos o reduzido número de vectores de teste para uma cobertura de faltas de 100%.

Tabela 2-3. Número de vectores de teste para alguns circuitos de referência

Circuito	fc (%)	# VT	# V_i (*)
Somador 16-bit CLA	100	44	2^{33}
Somador 32-bit CLA	100	235	2^{66}
Somador 16-bit CS	100	61	2^{33}
Somador 32-bit CS	100	241	2^{66}
ALU 16-bit	100	74	2^{33}
ALU 32-bit	100	276	2^{66}
Comparador 16-bit	100	114	2^{33}
Comparador 32-bit	100	222	2^{66}
c880 ALU (8 bit+controlo)	100	196	2^{17}
c7552 ALU (34-bit)	100	166	2^{70}

(*) Supondo teste exaustivo.

- *Teste Pseudo-Exaustivo*

O teste exaustivo de um circuito combinatório com n entradas primárias implica 2^n vectores de teste, o que na prática limita os testes exaustivos a cerca de 20 entradas primárias. As alternativas *pseudo-exaustivas* reduzem significativamente o número de vectores de teste [McC84, Bar87], sendo a técnica mais corrente a de explorar a dependência parcial, nos circuitos em que nenhuma saída primária depende de todas as entradas primárias. Neste caso, basta aplicar combinações que activam as saídas, embora fiquem por testar as faltas que façam uma saída primária depender de entradas primárias não previstas [McC81].

2.2.4 *Projecto para a Testabilidade*

O principal objectivo do projecto para a testabilidade consiste em facilitar o teste, tornando-o mais compacto e rápido, aspectos que são essenciais no teste concorrente, recorrendo para este

fim a técnicas que facilitam a geração do teste e melhoram a cobertura de faltas [Wil83]. Para além de melhorar a *observabilidade* e *controlabilidade* já referidas, estas técnicas devem também incidir na *previsibilidade*, a propriedade que assegura a repetição das respostas aos estímulos fornecidos, uma característica que pode ser afectada pelo estado inicial, corridas, oscilações e outras condições nem sempre previsíveis. No entanto, devemos levar em conta que a sobrecarga de teste associada a estas técnicas aumenta a complexidade, o número de portas lógicas e a potência dissipada, pelo que as vantagens resultantes devem compensar o impacto no rendimento de fabrico e no tempo de vida do produto acabado.

Conforme a universalidade da sua aplicação, as técnicas utilizadas dizem-se *estruturadas* (que nem sempre fornecem a solução óptima, mas facilitam a sistematização) ou *ad-hoc*, entre as quais destacamos sumariamente as mais correntes:

- *Inserção de Pontos de Teste*

Os conceitos básicos utilizados neste caso regem-se pelos estimadores já referidos da observabilidade e controlabilidade. A prática corrente é estabelecer no algoritmo limiares de custo para as duas medidas e, uma vez atingidos, abortar a geração do teste e indicar a necessidade de inserir pontos de teste nesses nós, denominados *Pseudo-EP* ou *Pseudo-SP* (*PPI* e *PPO* em inglês) [Hay85, Kie98]. Relacionado com estes procedimentos, foi demonstrado que nos microprocessadores a testabilidade do barramento de dados é relevante, mas a do barramento de endereços tem poucas vantagens, podendo mesmo ser contraproducente [Reb98].

- *Evitar lógica redundante*

A redundância pode ser inserida inadvertidamente e permanecer indetectada (identificá-la é um problema NP-completo), ou intencionalmente, seja para evitar instabilidade ou por necessidades de tolerância a falhas, casos em que deve ser possível desactivá-la durante os testes (sem degradar as funções a testar). Em ambos os casos, a lógica deste tipo afecta a determinação da cobertura de faltas e degrada a geração do teste.

- *Técnicas de partição*

O número de vectores de teste pode ser reduzido por uma partição cuidada do circuito de missão, em blocos funcionais com um número de entradas reduzido. Esta abordagem é útil tanto para os circuitos combinatórios como para os sequenciais, caso em que um contador de n bits, dividido em dois iguais durante o teste, será testado em $2^{(n/2+1)}$ ciclos de relógio, se o teste for individual, ou em $2^{n/2}$, se for em paralelo [Lit91].

- *Inicialização e inibição*

Esta regra recomenda projectar circuitos que sejam facilmente levados a um estado pré-definido, permitam inibir os circuitos osciladores, e se evitem sequências concebidas pelo projectista e dificilmente tratáveis por algoritmos de geração do teste, bem como soluções indutoras de oscilações, ou com resultado indefinido.

- *Circuitos C-Testáveis*

Estas técnicas são aplicáveis a circuitos estruturados, formados por células combinatórias idênticas interligadas e saídas directamente observáveis, em que cada célula pode ser tratada como um segmento testável exaustivamente, se a falta não tornar sequencial o seu funcionamento. Um circuito *C-testável* [Fri73] tem a propriedade de poder ser testado por um conjunto de vectores de teste que não depende do número de células e que pode conter apenas cinco vectores de teste [Hay85], ou mesmo três em determinados circuitos [Fuj85]. Deve no entanto referir-se que a sobrecarga de teste necessária afecta o desempenho do circuito e degrada a fiabilidade [Shr81a, Shr81b, Str97].

- *Técnicas para circuitos sequenciais*

No caso dos circuitos sequenciais, o teste deve distinguir o comportamento de uma máquina de estados do de outra com idêntico número de entradas e saídas primárias, bem como de estados. Uma sequência de inicialização para estes casos deve ser independente do estado inicial, verificar os estados esperados e ainda cada entrada da tabela de estados, pelo que só é aplicável a circuitos reduzidos. Para além disso, nos circuitos sequenciais, a geração do teste implica múltiplas iterações temporais, que crescem de forma exponencial com o número de elementos de memória [Agr88], pelo que terá sempre custos excessivos para circuitos grandes.

Resulta daqui a necessidade de aplicar técnicas de projecto para a testabilidade aos circuitos sequenciais, uma área onde as soluções individuais de cada projectista foram durante muito tempo frequentes, mas onde há uma regra genérica: inserir nós de controlabilidade que permitam transformar um circuito sequencial em combinatório, durante o teste [Che89, Dev90, Ism90, Pom92, Che93, Pom93]. Esta abordagem é utilizada na versão de 93 dos circuitos sequenciais de referência ISCAS'89 [Add93, Ham98a] estando prevista a apresentação de um conjunto de novos circuitos de referência testáveis nestas condições [Ben99]. Entre as ferramentas de síntese para circuitos sequenciais aplicáveis nesta área, podemos destacar o *SIS (System for Sequential Systems Synthesis)* [Sen92, Mil98, Mos98], as que são baseadas em algoritmos genéticos

[Kei98], bem como os métodos de simulação diferencial de faltas (*DSIM* [Che90]) e de análise estatística da probabilidade dos sinais [Kei97].

Entre as técnicas estruturadas, destacamos o BIST (*Built-In Self-Test*) e conceitos derivados [Sal88, Kim91, Zor92, Ren98a], que optámos por considerar apenas no próximo capítulo atendendo ao facto de serem particularmente úteis no teste em serviço, e ainda as infra-estruturas de teste por varrimento, actualmente na base da maior parte dessas técnicas, que analisaremos seguidamente.

2.3 As infra-estruturas IEEE 1149.x

O teste de cartas de circuito impresso recorre frequentemente a emuladores que substituem os circuitos e a agulhas de contacto (*Bed-of-Nails*), em associação com programas de diagnóstico. No entanto, a miniaturização impõe restrições físicas crescentes, para além de que as forças mecânicas necessárias para garantir bons contactos em CIs com 500, 1000 ou 2000 pinos, são potencialmente destrutivas [Doo98] e forçam a procura de alternativas [Zor98], entre as quais a mais utilizada actualmente é a infra-estrutura de teste por varrimento, que abordaremos agora.

2.3.1 A norma IEEE 1149.1

A infra-estrutura BST definida na norma IEEE 1149.1 é hoje suficientemente bem conhecida para se tornar redundante descrevê-la em pormenor neste documento, pelo que remetemos o leitor para a respectiva norma [IEEE93], para referências genéricas [Mau90, Ben91, Jon91a, Mau91, Par92, Sed94] ou para teses onde foi descrita com detalhe [Fer92a, Lub94, Tav95, Mont98]. No nosso caso iremos apenas referir aspectos menos citados que vamos explorar adiante, acompanhados pelas figuras pertinentes, a começar pela organização da infra-estrutura IEEE 1149.1, representada na figura 2-3 e que utiliza um Porto de Acesso para Teste (**TAP**) que compreende quatro linhas obrigatórias:

- Duas de controlo: TCK e TMS (relógio de teste e selecção do modo de teste)
- Duas de dados: TDI e TDO (entrada e saída de dados de teste)

Entre os registos indicados na figura 2-3, destacamos o registo de Instrução, o de Bypass e o de Identificação (do componente/fabricante). O registo Periférico e os de Utilizador, são constituídos por células de deslocamento que permitem observar e (normalmente) controlar os nós em que estão inseridas.

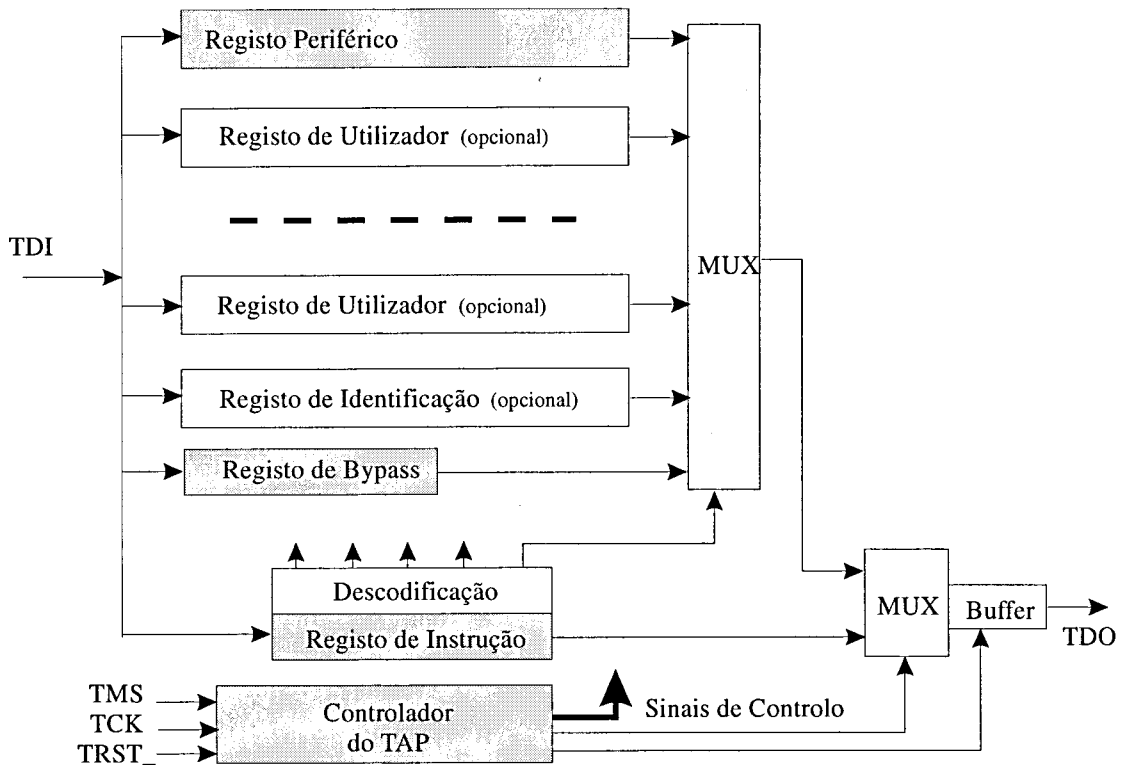


Figura 2-3. Arquitectura IEEE 1149.1

Nestas células, esquematizadas na figura 2-4 para os casos da controlabilidade e observabilidade ou somente de observabilidade, os sinais E_p e S_p representam a entrada e a saída de sinal (paralela), E_s e S_s a entrada e a saída série utilizada no deslocamento sequencial, sendo os restantes sinais de controlo, em conformidade com a norma.

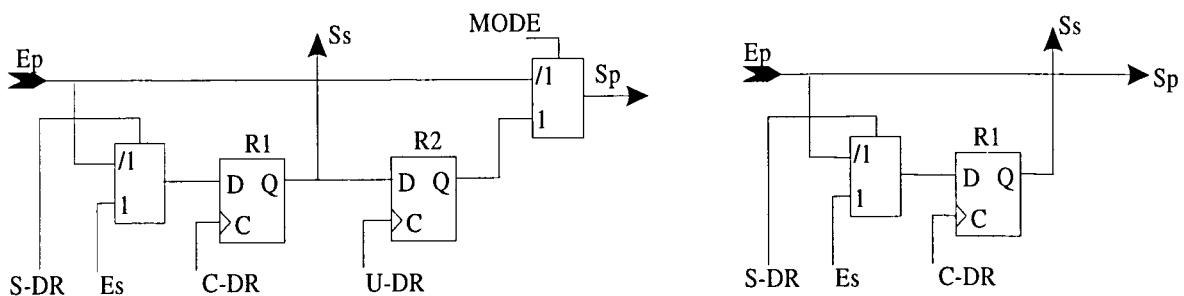


Figura 2-4. Células de deslocamento

Embora o encargo destes elementos raramente seja referido, esta informação é para nós relevante, uma vez que teremos de calcular as sobrecargas de teste e de confiabilidade das soluções propostas, bem como a fiabilidade relativa da infra-estrutura, face ao circuito de missão

que monitoriza e a taxa de falhas daí decorrente. O encargo referido corresponde a uma complexidade equivalente aproximada de [Doo98]:

- 470 ~ 500 portas lógicas no controlador do TAP.
- 13 portas lógicas nas células de observabilidade e controlabilidade.
- 7 portas lógicas nas células de observabilidade.

Como exemplo, num circuito integrado com 100 pinos de sinal, teremos um total variável entre 1200 e 1800 portas lógicas, que representa uma sobrecarga de teste relevante (tomando valores de referência de 5%) se o circuito de missão tiver menos de 20.000 portas lógicas. Embora a norma forneça apenas orientações de implementação e seja lícito minimizar a sobrecarga inserindo a infra-estrutura no circuito de missão, do ponto de vista da tolerância a falhas é necessária uma boa separação relativamente a este. Notemos ainda que o encargo das células de observação representa cerca de metade do das que permitem também controlabilidade (7/13), com a vantagem de não introduzirem atrasos no percurso de sinal e de ser menor a probabilidade de induzirem falhas no circuito de missão, uma vez que não possuem o multiplexer de saída. O diagrama de estados do controlador do TAP está representado na figura 2-5, com destaque para os ciclos de deslocamento de dados (DR) e de instruções (IR).

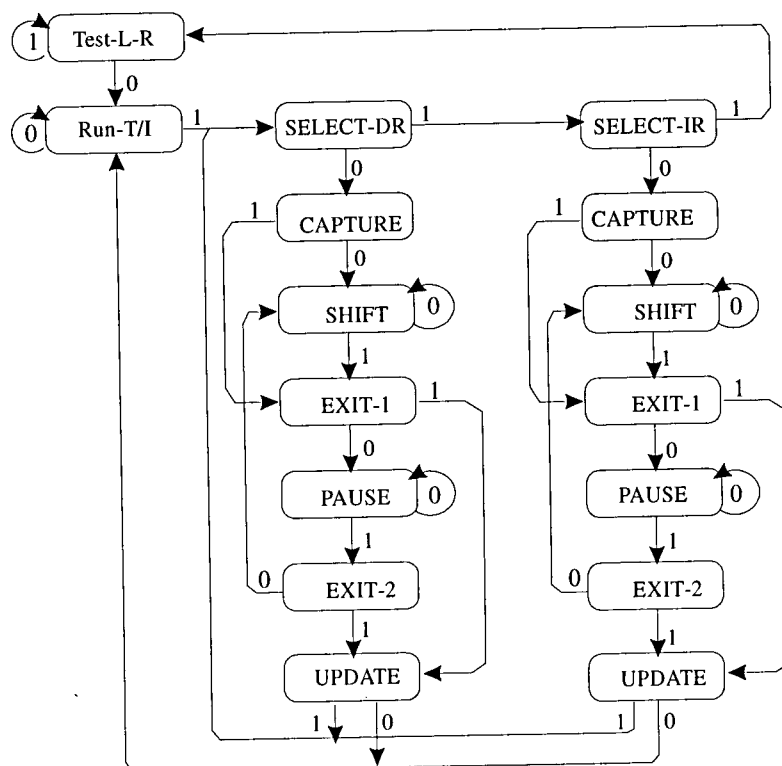


Figura 2-5. Diagrama de Estados do TAP

O armazenamento das instruções efectua-se no registo de instrução, que tem geralmente oito bits, e na tabela 2-4 estão representadas as instruções obrigatórias. Para além destas, o projectista pode implementar instruções opcionais, tais como *Intest*, *IDcode*, *Clamp*, *Runbist* ou *HighZ*, também descritas na norma e a que faremos referência onde seja necessário, e quaisquer outras de que tenha necessidade.

Tabela 2-4. Instruções obrigatórias da norma 1149.1

Instrução	Abreviatura	Código	Acção
Bypass	BP	1...1	Selecciona o registo de BP.
Extest	Ext	0...0	Selecciona o registo periférico, em modo intrusivo.
Sample/ Preload	S/P	Não definido	Selecciona o registo periférico, em modo transparente.

• *Testabilidade da infra-estrutura BST*

A utilização da infra-estrutura BST envolve um compromisso entre o tempo de teste, área de silício acrescida, cobertura de faltas e número de pinos para funções de teste, que é vantajoso para o utilizador (ou de contrário não seria utilizada), mas levanta a questão da testabilidade da própria infra-estrutura [Jon91b].

Como a infra-estrutura BST só funciona com o circuito de missão alimentado, não permite determinar antecipadamente eventuais falhas que, no caso de curto-circuitos, podem destruir as saídas e células afectadas. Quanto aos diversos tipos de falhas que podem ocorrer na infra-estrutura, os circuitos abertos nas ligações a TDI e TMS são interpretados como faltas *sempre-a-1*, em consequência das regras definidas na norma, e as falhas em ponte devem ser interpretadas em função das características da tecnologia. Assim, quer a infra-estrutura seja testada após a inicialização do circuito ou ciclicamente, nas aplicações confiáveis, permite um teste decisor da respectiva operacionalidade [Mai90]. Notemos ainda que a norma 1149.1 prevê que a infra-estrutura BST seja reiniciada em cinco períodos de TCK se TMS for mantido no estado “1”, bem como uma alternativa redundante através do pino opcional de reinicialização (*TRST*), e que ambas colocam a infra-estrutura num modo transparente para o circuito de missão. Nos barramentos com pinos bidireccionais ou de alta impedância, deve ser incluída uma célula adicional, dedicada, para controlar e identificar para o exterior o modo de funcionamento actual.

- ***Inserção da infra-estrutura BST***

Uma análise dos componentes comercializados e da informação reportada pela indústria, permite identificar três alternativas principais para disponibilizar uma infra-estrutura 1149.1, a saber:

- *CIs dedicados*: estes componentes, de que é exemplo a série SCOPE da Texas Instruments [Tex99], permitem substituir outros funcionalmente idênticos e geralmente orientados ao barramento, acrescentando-lhe a infra-estrutura BST. No entanto e num contexto de tolerância a falhas, se forem incluídos somente para melhorar a testabilidade, a sobrecarga que representam e o número de ligações que requerem podem afectar a fiabilidade, pelo que a sua inclusão deve ser ponderada face às vantagens esperadas [Cro89]. Estes componentes permitem efectuar análise de assinatura mas, sendo este processo dependente da sequência temporal dos vectores de teste, não é aplicável num teste de tipo concorrente, como o que procuramos.
- *Componentes programáveis pelo utilizador*: a maioria dos componentes deste tipo, *FPGA* e *CPLD* principalmente, dispõem actualmente de infra-estrutura BST nativa, que pode ser utilizada para os programar, ou quando assim não sucede é comum o fabricante facultar bibliotecas que simplificam a respectiva inclusão [Tex93]. No primeiro caso, o utilizador tem acesso à funcionalidade prevista na norma sem encargos mas, para aceder a pontos de teste internos, terá de implementar uma nova infra-estrutura que introduz uma sobrecarga de teste indesejável. Pensamos que este problema é transitório, até os fabricantes incluírem maior flexibilidade na infra-estrutura nativa, como a possibilidade de gerir registos do utilizador. Entre os vários fabricantes deste tipo de componentes, podemos referir a Texas Instruments [Tex99], a Xilinx [Xil99], a Cypress [Cyp99] e a Altera [Alt99].
- *Embutida no fabrico*: é o caso ideal, uma vez que os fabricantes desenvolveram técnicas para inserir células de baixo custo e com características vantajosas na tolerância a falhas, que passam por implementar transístores sob as linhas de alimentação, para minimizar o silício extra. Limitados a um menor número de camadas e de maiores dimensões, estes transístores *submersos* originam portas lógicas mais lentas, que limitam a frequência de TCK, mas que são também mais robustas [Abre97]. A limitação de TCK, tipicamente a cerca de metade da frequência máxima do circuito de missão, é pouco relevante no teste de produção, mas condiciona a utilização da infra-estrutura em regime concorrente e será tida em conta na solução proposta.

Os circuitos comerciais disponíveis estão vocacionados para o teste fora de serviço, mas podemos admitir que futuros projectos venham a incluir novas infra-estruturas ou a determinar a evolução das actuais, se viáveis economicamente [Lof96]. Um destes circuitos é o *Data Bus Monitor* (DBM [Whe90, Tex99]), vocacionado para capturar e armazenar a sequência dos valores que ocorrem nas suas 16 entradas, após detectar um padrão de referência, permitindo deslocar posteriormente o conteúdo da respectiva memória interna, por varrimento através da infra-estrutura BST. No entanto, esta operação é demasiado morosa para qualquer acção em tempo útil, para além de que o DBM não permite corrigir erros.

Existem actualmente diversas ferramentas para inserção automática da infra-estrutura BST no circuito de missão, de que são exemplos o *ABSIT* (*Automatic Boundary Scan Insertion Tool*) [Abs96] e o *JTAG ActiveTest* [JTA98].

2.3.2 Outras normas

Para além da infra-estrutura 1149.1, consideramos conveniente fazer um referência a três normas adicionais, que embora não sejam incorporadas no trabalho proposto estão com ele relacionadas.

- **A norma IEEE 1149.4**

Esta norma, aprovada recentemente, estende a infra-estrutura BST aos circuitos mistos, através de dois pinos analógicos adicionais que se acrescentam ao TAP (AT1 e AT2, que recebem a designação de ATAP, ou TAP analógico) e de blocos de lógica de teste adicionais, com relevo para os ABM (módulos periféricos analógicos) e para o TBIC (circuito de interface ao barramento de teste) [IEEE99a, Mat98], como se representa na figura 2-6.

A aplicação desta infra-estrutura representa uma sobrecarga elevada, que deve ser confrontada com outras alternativas de teste [Vef91, Med98]. No entanto, nos componentes mistos, a parte analógica representa geralmente um fracção pequena da lógica, mas o custo do respectivo teste pode ultrapassar o da parte digital, pelo que a sobrecarga de teste analógica tende a diluir-se e tornar-se compensadora.

Esta norma permite a realização de testes paramétricos [Sze96, Sze98, Sil98], uma vasta área ainda pouco explorada, e tal como aconteceu com a norma 1149.1, o tempo tenderá a definir as aplicações em que a infra-estrutura 1149.4 se justifica. Na área da confiabilidade podemos antever potenciais aplicações, mas é importante notar que a tolerância a falhas em que esta tese se centra diz respeito ao domínio digital. No domínio analógico, para o qual está prevista uma

evolução mais lenta do que a do digital [SIA97], algumas das regras diferem e é necessário definir com cuidado os benefícios que compensam a sobrecarga. Refira-se ainda que foram propostas soluções alternativas para o teste da parte analógica, com destaque para [Whe96] e [Rus96].

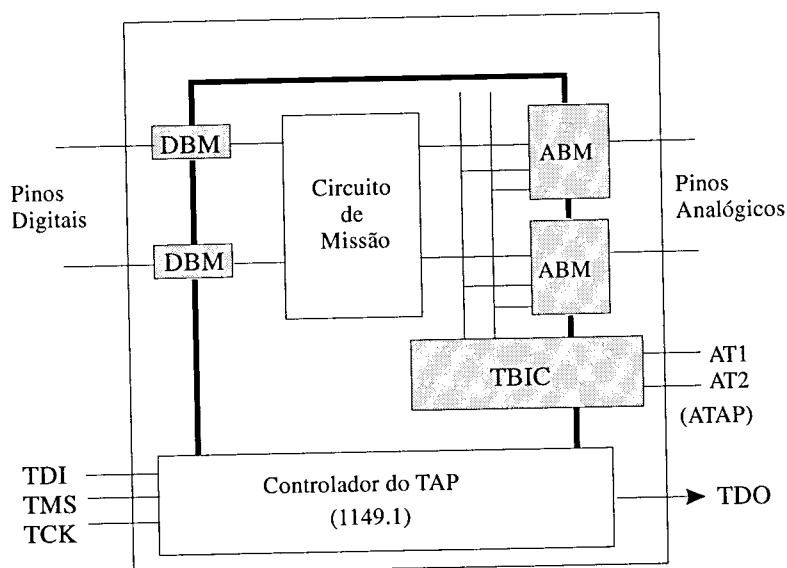


Figura 2-6. Infra-estrutura IEEE 1149.4

- **A Norma IEEE 1149.5**

A norma 1149.1 foi definida a nível do circuito integrado e vocacionada para a carta de circuito impresso, sem prever encadeamento hierárquico. No entanto, num sistema tolerante a falhas, os erros devem ser reportados verticalmente e deverá ser utilizado um barramento de comunicação, que pode ser o proposto na norma 1149.5 [IEEE95]. Esta configuração, ilustrada na figura 2-7, permite a um controlador central gerir múltiplos escravos nas respectivas cartas de circuito impresso.

Embora não utilizemos este barramento, interessa realçar o facto de a aplicação desta norma requerer um conversor de protocolo 1149.1-1149.5, que pode acumular (sem encargos relevantes) essas funções com as de controlador BST residente, inevitavelmente necessário no teste concorrente [Bha91, Bro91, Der91, Pet97]. Note-se que foram também apresentadas soluções alternativas à norma 1149.5, baseadas na extensão da infra-estrutura 1149.1 ao nível de sistema, com relevo para as efectuadas em [Whe92, Whe93].

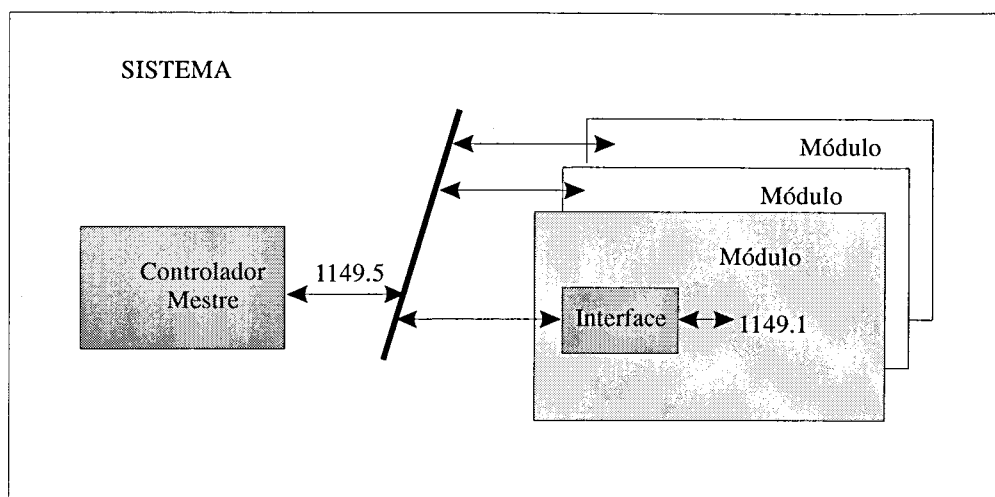


Figura 2-7. Conversão entre as normas 1149.1-1149.5

- ***A proposta de norma IEEE P1500***

A proposta IEEE P1500 [IEEE99b] está vocacionada para o teste de módulos multi-pastilha (MCM), mormente para os núcleos IP reutilizáveis, um processo cuja dificuldade se agrava quando estes são provenientes de múltiplos fornecedores. Entre os objectivos definidos para esta futura norma, está a generalização a todos os tipos de módulos e a aplicabilidade a nível hierárquico.

2.4 Conclusão

Neste capítulo, para além de uma breve introdução à área da confiabilidade, procurámos mostrar que temos hoje técnicas e ferramentas para projectar circuitos testáveis e gerar testes eficientes e compactos, característica que será aproveitada no desenvolvimento deste trabalho e aprofundada no capítulo quatro. Interessa realçar a noção de que um teste concorrente, funcional, pode prescindir do modelo de faltas, uma vez que monitoriza as saídas do circuito em tempo real, comparando-as com a especificação. Contudo, esta abordagem recorre a uma quantidade de informação dificilmente praticável com processos de teste vectorial, o que aumenta o interesse num teste de detecção compacto, gerado com base num modelo de faltas.

Necessários ao teste de produção e gerados com custo apreciável, os vectores de teste tornam-se geralmente inúteis após aquela fase. A nossa proposta passa pela sua reutilização em modo concorrente, numa solução de custo limitado (quase) somente à infra-estrutura BST, justificando desta forma, o esforço necessário para gerar o melhor grupo de vectores de teste possível.

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

Capítulo 6

Capítulo 7

Capítulo 8

Bibliografia

Anexos

Projecto para a Confiabilidade

3. Projecto para a Confiabilidade

Este capítulo é dedicado a uma análise dos conceitos utilizados na área da confiabilidade e às arquitecturas e técnicas utilizadas no projecto de sistemas tolerantes a falhas, com vista a fundamentar a escolha das propostas a apresentar, assunto que será tratado no capítulo quatro.

3.1 Conceitos de Confiabilidade

Na terminologia do grupo de trabalho 10.4 da IFIP⁴ [Lap85, IFI99], quando um sistema funciona correctamente, diremos que *assegura o serviço previsto na especificação* e, caso contrário, que a prestação do serviço foi *interrompida*. A alternância entre os dois estados resulta da *avaria* e da *restauração* do serviço, sendo a *qualidade* do serviço assegurado avaliada, do ponto de vista do utilizador, por dois parâmetros principais:

- **Fiabilidade** $R(t)$ (*Reliability*)
- **Disponibilidade** $A(t)$ (*Availability*)

A *fiabilidade* avalia a continuidade do serviço ao longo de um intervalo de tempo, enquanto a *disponibilidade* o qualifica tendo em conta as interrupções, se aceitáveis durante o intervalo de funcionamento previsto, denominado como *período* (ou *tempo*) de *missão* [Joh89]. Para além dos restantes parâmetros que analisaremos em seguida, deve ainda ser considerada a Testabilidade, que ganhou já estatuto de área autónoma, como vimos no capítulo anterior.

Na construção de um sistema confiável, é necessário conjugar as acções de dois grupos de métodos, classificados da seguinte forma:

- *Implementação*:
 - Prevenção (*Fault-avoidance*): métodos para construir sistemas robustos.
 - Tolerância a Falhas (*Fault-tolerance*): métodos para implementar mecanismos de redundância que permitem ultrapassar as consequências das falhas.

⁴ - IFIP - International Federation for Information Processing.

- **Validação:**
 - Depuração (*Error-removal*): métodos para verificar e corrigir as falhas presentes, com ênfase particular no tratamento das falhas latentes.
 - Previsibilidade (*Error-forecasting*): métodos para estimar e avaliar as consequências dos erros.

Os dois primeiros métodos visam garantir a construção de sistemas que cumprem a especificação, enquanto os dois últimos servem para assegurar a confiança no serviço que o sistema fornece. Vejamos primeiro os parâmetros utilizados para aferir a confiabilidade.

3.1.1 Parâmetros da Confiabilidade

Para definir os parâmetros utilizados na confiabilidade consideremos uma amostra de N componentes idênticos em condições de funcionamento semelhantes, seja $S(t)$ o número de *sobreviventes* ao fim do intervalo t e $F(t)$ o número dos que *falharam* até esse instante.

- **Fiabilidade - $R(t)$**

Define-se fiabilidade como a probabilidade de sobrevivência, dada por $R(t) = S(t)/N$

De igual modo, a não-fiabilidade é dada por $Q(t) = F(t)/N$

Como $S(t) + F(t) = N$,

temos $R(t) + Q(t) = 1$.

A *taxa de falhas*, $Z(t)$, é definida como o número de falhas por unidade de tempo em relação ao número de sobreviventes:

$$Z(t) = \frac{1}{S(t)} \cdot \frac{dF(t)}{dt}$$

A esta equação está associada a curva da figura 3.1, que descreve as três fases de evolução da taxa de um componente: mortalidade infantil, período de vida útil e envelhecimento.

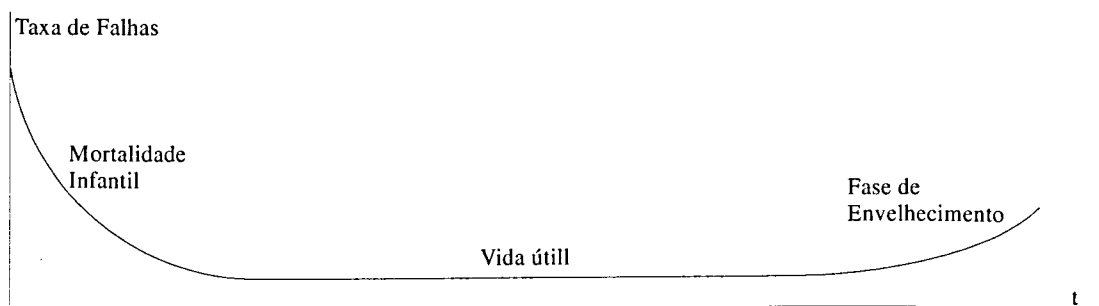


Figura 3-1. Evolução da taxa de falhas num componente

Na primeira e última fases a taxa de falhas é influenciada respectivamente pelos defeitos de fabrico e pelo desgaste, mas durante a vida útil as falhas ocorrem de modo aleatório, permitindo admitir uma taxa de falhas constante [Cou81]:

$$Z(t) = \lambda$$

Como

$$R(t) = 1 - F(t)/N$$

a fiabilidade durante a vida útil pode ser calculada por [Lal85]:

$$R(t) = e^{-\lambda \cdot t} \quad (\text{Eq. 3-1})$$

Esta equação é conhecida como *lei de falhas exponencial* e, tendo em conta que durante o período de missão o produto $\lambda \cdot t$ é muito pequeno, pode ser aproximada pelos dois primeiros termos da correspondente série de Euler, o que dá $R(t) \approx 1 - \lambda \cdot t$ e permite representar a fiabilidade por 0,99999... ou 9_i, onde *i* indica o número de nove à direita da vírgula e é denominado *índice de confiabilidade* [Joh89].

• Determinação prática da taxa de falhas

A forma de cálculo usual da *taxa de falhas* λ em componentes electrónicos está definida na norma MIL-HDBK-217 de 1965 [Mil65], a que se seguiram as revisões B (1974) e C (1979), e baseia-se na análise de resultados obtidos de componentes reais. A revisão MIL-HDBK-217B define uma equação estimativa da taxa de falhas constante, válida durante a fase de vida útil de um CI, com base em sete parâmetros:

$$\lambda = \pi_L \cdot \pi_Q \cdot (C_1 \cdot \pi_T + C_2 \cdot \pi_E) \cdot \pi_P \quad (\text{Eq. 3-2})$$

A taxa de falhas determinada pela equação 3-2 mede-se em falhas por 10⁶ Horas, sendo atribuído aos parâmetros que a integram o seguinte significado:

- O factor de *aprendizagem* π_L é considerado 1 para processos de fabrico maduros e 10 para processos novos, com pouca experiência e menos fiáveis.
- O factor de *qualidade* π_Q tem quatro classes de base, A-D, e varia de 1 a 300 em função da classe de testes a que o CI é sujeito pelo fabricante. A e B destinam-se a aplicações militares (MIL), a que são atribuídos os factores 1 e 2 respectivamente. A classe C, com $\pi_Q=16$, é atribuída aos componentes comerciais de boa qualidade, em que todos os componentes são testados em valores nominais. A classe D atribui um factor de 150 aos CI testados sumariamente e 300 àqueles em que só amostras da produção são testadas pelo fabricante.

- O factor de *temperatura* π_T é descrito por uma equação que considera a tecnologia, a temperatura de funcionamento, o tipo de encapsulamento e a potência dissipada pelo CI, distinguindo os componentes lineares dos digitais, sendo que para estes últimos π_T toma o valor 0,1 a 25°C.
- O factor *ambiental* π_E pretende traduzir a agressividade do meio, ponderando influências como temperatura, humidade, vibração, agressão química e outras. Este factor varia de 0,2, num ambiente de temperatura controlada, a 4-5 em regime industrial, 6 num satélite ou avião, até 10 nos sistemas sujeitos a elevadas acelerações, como é o caso dos mísseis.
- A influência do *número de pinos* está representado por π_P , que toma o valor 1,0 para os CIs com menos de 25 pinos, 1,1 de 25 até 64 pinos, e 1,2 se o CI tiver mais de 64 pinos. Para além do número de pinos, este factor depende também da tecnologia envolvida e do encapsulamento utilizado.
- Os factores de Complexidade C_1 e C_2 são definidos por equações que ponderam a complexidade do circuito, em função do número de portas lógicas ou de transístores nos circuitos lineares, utilizando coeficientes crescentes com esse número, que pretendem reflectir as dificuldades de projecto e teste.

A tabela 3-1 indica a taxa de falhas (em falhas/ 10^6 H) para vários componentes, calculada de acordo com a revisão B da norma, supondo $\pi_L=1$, $\pi_Q=16$, $\pi_T=0,35$, $\pi_E=0,2$ e $\pi_P=1$, considerados valores médios para estes parâmetros, em circuitos comerciais a operar numa sala [Joh89, Cro89].

Tabela 3-1. Taxa de falhas λ

Circuitos digitais		Memórias RAM	
Número de portas lógicas	λ	Número de bits	λ
50	0,1527	1K	0,8837
100	0,2312	2K	1,3491
200	0,3655	8K	3,1453
500	1,4483	16K	4,8033
1000	14,4880	32K	7,3362

• Tempo Médio até Falhar - MTTF

O tempo médio até falhar (*Mean Time To Failure*) estima o intervalo de funcionamento esperado (durante o período de vida útil) até à primeira falha (catastrófica) e é definido pelo seguinte integral [Phi96]:

$$MTTF = \int_0^\infty e^{-\lambda t} dt$$

Integrando entre 0 e um determinado instante t , resulta para este parâmetro um valor que corresponde ao inverso da taxa de falhas [Lal85]:

$$MTTF = 1/\lambda \quad (\text{Eq. 3-3})$$

Este parâmetro, por vezes denominado *Tempo de vida* do componente, é relevante para falhas permanentes, uma vez que as temporárias permitem geralmente a recuperação. Recorrendo aos valores de λ apresentados na tabela 3-1, a equação 3-3 permite-nos estimar valores de MTTF entre 10 e 600 anos de vida.

• Tempo Médio de Reparação - MTTR

O MTTR (*Mean Time to Repair*) ou tempo médio de reparação é calculado por estimativa ou, de preferência, determinado experimentalmente e inclui duas etapas:

- *Passiva*: tempo para localizar a falha, dependente da experiência do técnico.
- *Activa*: tempo de substituição da peça defeituosa, previsível com rigor.

Calculando a média de uma amostra de reparações, o fabricante define o MTTR, ou o seu inverso μ , a *taxa de reparações*, expressa em reparações por hora. Se não existirem mecanismos auxiliares de diagnóstico no sistema, a componente passiva é quase sempre dominante.

• Nível de Manutenção - M(t)

O nível de manutenção é a probabilidade de o sistema ser reparado dentro de um dado intervalo de tempo, estando relacionado com a taxa de reparações através da equação:

$$M(t) = 1 - e^{-t/MTTR} = 1 - e^{-\mu t}$$

• Tempo Médio entre Falhas - MTBF

O MTBF (*Mean Time Between Failures*) é definido como:

$$MTBF = MTTF + MTTR.$$

Nos sistemas actuais espera-se que o MTTF seja muito superior ao MTTR, pelo que o MTBF resulta próximo do MTTF. Contudo, em muitos sistemas e particularmente nas aplicações de

acesso difícil ou condicionado, estes valores podem diferir apreciavelmente, como é caso de centrais e unidades nucleares, onde o MTTR pode atingir semanas.

- **Disponibilidade - $A(t)$**

Define-se $A(t)$ (*Availability*) como a relação entre o período de funcionamento e tempo total (funcionamento + avaria), sendo dada por

$$A(t) = \frac{MTTF}{MTBF}$$

Uma vez que o MTBF é definido por (MTTF+MTTR), a disponibilidade de um sistema aumenta quando se diminui o MTTR. Muitos sistemas tolerantes a falhas (implementados com mascaramento de erro) asseguram uma disponibilidade total ($A(t)=1$) durante o período de missão, mas um sistema não precisa de ser tolerante a falhas para ter elevada *disponibilidade*, bastando construí-lo com componentes de boa qualidade e fácil reparação.

- **Operacionalidade - $P(L, t)$**

O nível de operacionalidade qualifica a capacidade do sistema em assegurar uma determinada qualidade de serviço, com limiar L durante um intervalo t , alargando o conceito de fiabilidade aos casos de funcionamento em regime degradado.

- **Segurança - $S(t)$**

O factor de segurança avalia, de forma essencialmente qualitativa, a probabilidade de o sistema operar de forma correcta, ou falhar de maneira segura, sem risco para os utentes. No essencial, esta propriedade implica que o sistema tenha capacidade dar conhecimento atempadamente da ocorrência de falhas, permitindo que sejam reparadas e evitando que se acumulem.

3.1.2 *Medidas preventivas*

Num sistema sem redundância, a única forma de melhorar a fiabilidade é implementar medidas de prevenção, que minimizem a taxa de falhas. Estas medidas podem ser implementadas na fase de projecto, abrangendo qualquer dos tipos descritos nas duas secções seguintes, ou *a posteriori*, como a *manutenção preventiva*, onde se procura substituir os componentes ainda dentro do seu intervalo de vida útil estimada, ou logo que começam a apresentar sintomas não esperados, detectáveis por monitorização. Apesar de ter sido descurada durante muito tempo, a possibilidade actual de monitorização automática, associada a sensores económicos, reavivou o interesse nesta forma de prevenção [Vid93].

- **Processos Conservativos**

Estes tipo de procedimentos destina-se a reduzir a taxa de falhas com medidas de natureza passiva, que não afectam o funcionamento do sistema, sendo de dois tipos básicos:

- A *Previsão*, que incide nos seguintes pontos:
 - Manter o sistema tão simples quanto possível, desde que sejam garantidos os requisitos da especificação.
 - Minimizar a frequência de trabalho, limitando a potência dissipada e a temperatura de funcionamento, em benefício da vida útil.
 - Definir um modelo das faltas adequado aos requisitos: quanto mais abrangente for o modelo, maior será o custo e o tempo de desenvolvimento do projecto, o que implicará também mecanismos mais complexos, que degradarão inevitavelmente o MTTF do sistema. Um exemplo, é o facto de as falhas permanentes serem 10 a 100 vezes menos prováveis que as temporárias, o que permite dispensar redundância física em muitas aplicações que de outra forma teriam mais componentes, maiores custos e menor tempo de vida.
- A nível *Físico* consideramos:
 - Seleccionar componentes de elevada qualidade, se possível da série militar, ou submeter os que haja disponíveis a testes de despistagem antes de os utilizar. Teremos naturalmente de levar em conta os limites económicos, porque o custo dos componentes sobe rapidamente com a sua qualidade.
 - Utilizar componentes bem conhecidos, “maduros”, em relação aos quais tenham sido já detectados eventuais problemas, uma vez que componentes recentes representam um risco acrescido. Exemplos paradigmáticos são o veículo robótico utilizado na exploração do planeta Marte (*MARS Rover* [NAS97]), que utiliza o microprocessador 80C85 (bem conhecido e que se presume já totalmente depurado), e no outro extremo, os processadores da família Pentium, cuja complexidade e juventude faz com que sejam conhecidas dezenas de falhas internas (algumas sem solução prevista), que lhe vedam para já a utilização em aplicações aeronáuticas e espaciais [Avi98, Sos98].
 - Projectar com margens paramétricas e temporais adequadas, para que futuras substituições de componentes, na produção ou reparação, por outros de séries de fabrico ou fabricantes diferentes, não originem situações de mau funcionamento, dificilmente detectáveis [Ren98b].

- Incluir blindagens contra condições ambientais agressivas, nomeadamente para radiação electromagnética e de alta energia, controlo de temperatura e humidade, e outros factores que por vezes são negligenciados.

• **Diversidade de Projecto**

A ideia subjacente à *diversidade de projecto* (em sentido lato) consiste em providenciar percursos de projecto alternativos e comparar os resultados, com medidas a quatro níveis principais:

- *Especificação*: nenhuma especificação é perfeita, seja qual for a aplicação alvo, mas equipas independentes podem comparar resultados e tirar ilações, de forma *parcial* se cada equipa aborda uma parcela do problema, ou *integral* se trabalham em paralelo o mesmo problema. Apesar das vantagens, esta variante é onerosa e restringe-se a aplicações excepcionais.
- *Projecto*: aqui a situação é análoga e, não raras vezes, a equipa que define a especificação desenvolve também o projecto, o que é questionável. A diversidade permite neste caso utilizar módulos alternativos distintos, se for implementada redundância no sistema, ou apenas seleccionar o projecto mais adequado aos requisitos, em fiabilidade ou economia.
- *Componente*: sempre que haja réplicas no sistema, a diversidade recomenda utilizar componentes de séries ou de fabricantes diferentes, para minimizar a probabilidade de os defeitos de fabrico surgirem em simultâneo, originando falhas de modo comum.
- *Desfasamento temporal*: as falhas mais vulgares são as temporárias, e o recurso a módulos que funcionam desfasados no tempo permite detectar ou tolerar as falhas de duração inferior ao desfasamento. Alguns sistemas requerem, no entanto, intervalos de sincronização obrigatória, que impõem limites a este desfasamento.

A diversidade de projecto é sempre uma solução de custo elevado, mas recomendável nas fases iniciais do projecto, porque o custo da correcção de erros é tanto maior quanto mais cedo estes forem inseridos. Os erros de especificação, os primeiros no processo, tendem a ser os mais onerosos, porque uma especificação deficiente origina erros que geralmente só serão detectados pelo utilizador. Outro aspecto importante a referir neste contexto são as *alterações* que, ao contrário do projecto, onde se segue uma linha de raciocínio coesa, surgem sempre como um mal necessário. Muitas delas têm consequências dificilmente previsíveis, que podem ser activadas muito tempo depois de efectuada a alteração, pelo que, em projectos de responsabilidade elevada, se deve garantir que mais de um projectista analise as consequências de cada alteração.

3.2 *Arquitecturas Tolerantes a Falhas*

A prevenção permite minimizar a taxa de falhas, mas não torna o sistema *isento de*, ou *à prova* de falhas, pelo que é necessário recorrer a mecanismos redundantes que permitem tratar o erro sem actuar ao nível da falha, particularmente quando os parâmetros envolvidos na equação 3.2 são agravados pelas condições de funcionamento ou ambientais. Estes mecanismos, que analisaremos nas próximas secções, podem ser implementados a nível do circuito físico (também denominado *espacial*) ou em programação.

3.2.1 *Circuitos Tolerantes a Falhas*

O objectivo último da tolerância a falhas é evitar que o *erro se propague e origine uma avaria*. Para tal, o erro deve ser contido no módulo em que é gerado, uma vez que pode tornar-se indetectável fora dele, sendo dada a um módulo com estas características a designação de *Região de Confinamento de Falhas* [Lal94]. O tratamento de erro pode ser efectuado a nível do circuito de missão ou do bloco funcional, sendo correntemente aceite que não se prevêem novas soluções [Joh89], para além das duas conhecidas:

- Mascaramento.
- Detecção e recuperação.

Ao primeiro caso estão associadas as arquitecturas denominadas *estáticas* e ao segundo as *dinâmicas*, havendo também arquitecturas *mistas*, que conjugam algumas das melhores características de ambas.

Conforme os objectivos visados num projecto genérico, há duas configurações básicas para os componentes utilizados, como se representa na figura 3-2, onde suporemos que a fiabilidade de cada componente (sejam blocos funcionais ou circuitos de missão) é igual e dada por $R(t)$, ou abreviadamente por R :

- *Série*: todos os elementos têm de funcionar, numa sequência normalmente forçada pelo encadeamento dos componentes, sendo a fiabilidade do conjunto dada por

$$R_c = \prod_{i=1}^n (R_i) \quad (\text{Eq. 3-4})$$

- *Paralelo*: configuração necessária se a falha de um componente não deve interromper o serviço, permitindo uma fiabilidade calculada por

$$R_c = 1 - \prod_{i=1}^n (1 - R_i) \quad (\text{Eq. 3-5})$$

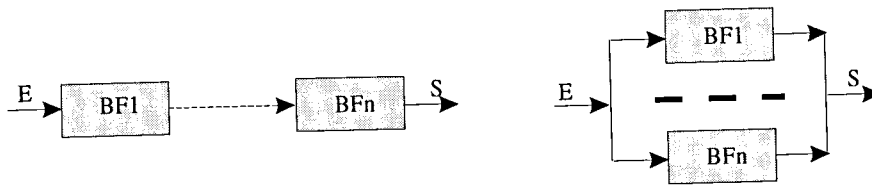


Figura 3-2. Disposição de componentes em série e em paralelo

A configuração em paralelo tem um tempo de missão superior à série, o que seria de esperar atendendo ao facto de ter redundância disponível. Note-se contudo que, no caso dos sistemas duplos com comparação, se não existirem mecanismos auxiliares de localização da falha, o funcionamento do paralelo equivale ao da série, uma vez que uma falha num dos blocos funcionais é suficiente para que o sistema avarie. Estas configurações podem ser combinadas para aproveitar as vantagens de cada uma delas, nas variantes *paralelo-série* ou *série-paralelo*, cuja fiabilidade é calculável a partir das duas anteriores [Lal85].

3.2.1.1 Arquitecturas Estáticas

Estas arquitecturas, também denominadas *passivas* por não implicarem a intervenção explícita dos mecanismos de tolerância a falhas, pertencem ao grupo da *Redundância Modular de ordem N (NMR)*, que confinam o erro por mascaramento no *Circuito Votador por Maioria (CVM)*. O número N de blocos funcionais, necessário para tolerar n blocos avariados, é dado pela equação:

$$N = (2n+1) \quad (\text{Eq. 3-6})$$

Daqui obtemos o número de avariados que N blocos funcionais permitem corrigir, dado pela equação⁵:

$$n = \lfloor (N-1)/2 \rfloor \quad (\text{Eq. 3-7})$$

Estas equações implicam um mínimo de três blocos funcionais para mascarar uma falta, justificando a impossibilidade de uma arquitectura duplicada poder corrigir um erro: em caso de desacordo (modelo de falta simples), um sistema com *redundância de ordem dois* não permite decisão imediata. Note-se que, mesmo tendo em conta que nestas arquitecturas deve ser prevista uma reparação rápida para evitar a acumulação de blocos faltosos, a redundância de ordem superior a três tem um interesse limitado devido às implicações em termos de custo, volume e potência dissipada. Teoricamente, aumentar N permite tolerar mais falhas, mas a complexidade

⁵ - $\lfloor X \rfloor$ representa o maior inteiro não superior a X .

do CVM cresce com N e sendo este elemento único (em série), o aumento de complexidade degrada a fiabilidade global de forma acentuada.

Notemos ainda que os códigos correctores de erro podem ser considerados implementações alternativas das arquitecturas NMR, na medida em que fornecem também informação adicional com acção imediata, mas são no entanto de aplicação complexa [Nel87, Joh89].

• Redundância Modular Tripla - TMR

A arquitectura com redundância de ordem três é a variante NMR mais usual, suportada pelas hipóteses de falta única, aleatória, com reparação rápida, num intervalo inferior ao MTTF. Existem duas alternativas para esta arquitectura, como se mostra na figura 3-3:

- *CVM único*: sendo o ponto fraco, por não replicado, deve detectar falhas próprias e ter uma fiabilidade muito superior à dos blocos funcionais votados, numa relação que varia na razão inversa do nível hierárquico a que é implementado, e que desaconselha por exemplo a implementação destas arquitecturas a nível de porta lógica [Lal85].
- *CVM triplo*: permite melhor fiabilidade na votação, útil para reconstituir sinais, mas as três saídas obrigam a relegar a decisão final para unidades a jusante, geralmente mais robustas.

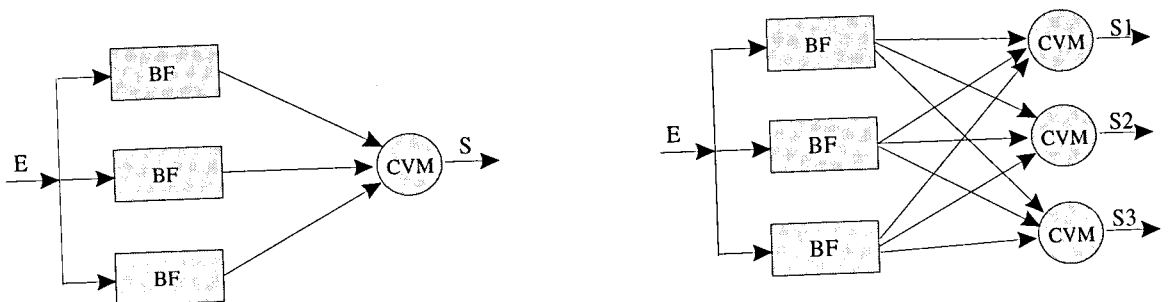


Figura 3-3. Arquitecturas TMR com CVM único e triplo

A fiabilidade de sistemas com vários blocos funcionais e modos de funcionamento, é geralmente analisada com diagramas de Markov [Joh89, Phi96], mas o funcionamento de um TMR pode ser interpretado directamente [Lal85]. Supondo o CVM único e ideal, e como na figura 3-3 os blocos funcionais estão em paralelo, a fiabilidade de um TMR será dada pela probabilidade de todos, ou apenas dois, estarem a funcionar correctamente:

$$\begin{aligned}
 R_{\text{TMR}} &= R_c^3 + 3 \cdot R_c^2 \cdot (1 - R_c) \\
 &= 3 \cdot R_c^2 - 2 \cdot R_c^3
 \end{aligned}$$

ou $R_{TMR} = R_c^2 \cdot (3 - 2 \cdot R_c)$

Considerando agora um CVM real com fiabilidade R , e atendendo a que se encontra em série no percurso de sinal, teremos:

$$R_{TMR} = R \cdot R_c^2 \cdot (3 - 2 \cdot R_c) \quad (\text{Eq. 3-8})$$

Durante o período de vida útil dos componentes, o CVM e os blocos funcionais terão taxas de falhas de, respectivamente, λ e λ_c , associadas a fiabilidades $R = e^{-\lambda \cdot t}$ e $R_c = e^{-\lambda_c \cdot t}$, que permitem rescrever a equação anterior como

$$R_{TMR} = e^{-(\lambda + 2\lambda_c) \cdot t} \cdot (3 - 2 \cdot e^{-\lambda_c \cdot t}) \quad (\text{Eq. 3-9})$$

A esta equação corresponde o gráfico da figura 3-4, que representa a evolução da fiabilidade do bloco funcional genérico e da arquitectura TMR dele derivada, supondo ainda que a taxa de falhas do CVM é dez vezes inferior à do bloco funcional ($\lambda = \lambda_c / 10$).

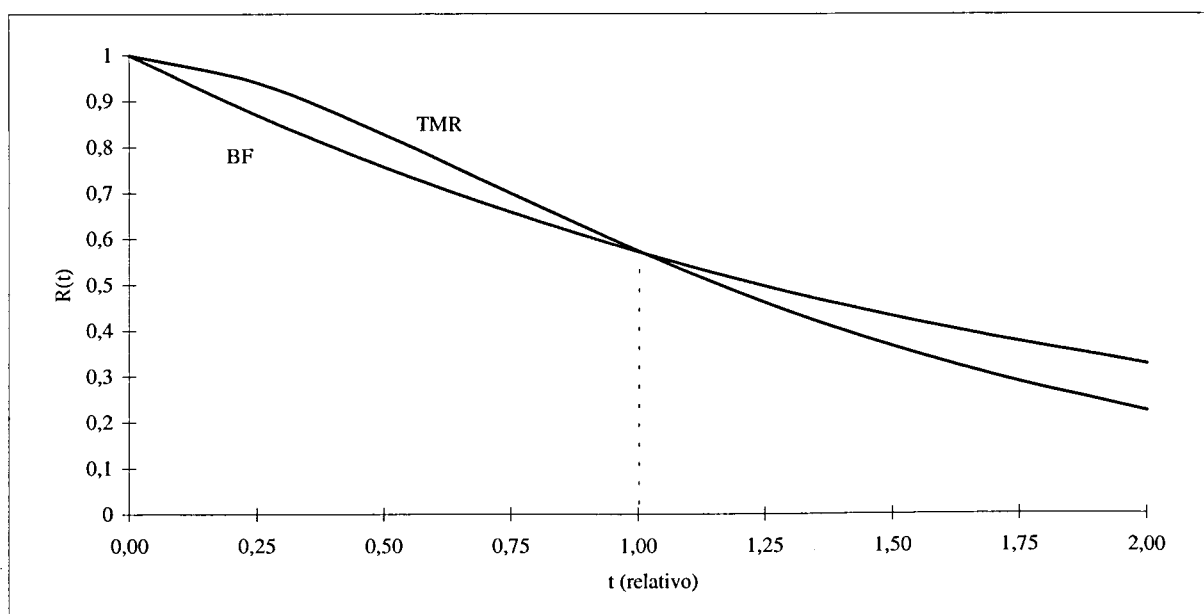


Figura 3-4. Comparação de fiabilidade BF - TMR

Neste gráfico, o eixo das abcissas representa a evolução temporal, em unidades relativas ao período em que as duas fiabilidades se igualam, uma vez que, devido ao maior número de componentes, o TMR tem uma taxa de falhas superior à do bloco funcional, que acaba por conduzir a uma fiabilidade inferior à do bloco funcional. Como tal, as arquitecturas TMR são previstas para um período de missão que, para um votador ideal, deve ser inferior a $MTTF_{TMR} = 0.693 \cdot MTTF_c$ [Phi96]. A tabela seguinte resume as características principais desta arquitectura.

Tabela 3-2. Características da arquitectura TMR

<i>Vantagens</i>	• Mascara falhas permanentes e temporárias de forma imediata. • Facilidade de projecto, por cópia do bloco funcional. • Não requer detecção explícita do erro. • Facilmente aplicável a circuitos sequenciais.
<i>Desvantagens</i>	• Necessidade de três réplicas, com os custos inerentes. • Período de missão restrito, condicionado pela complexidade do sistema. • Elevadas exigências no projecto do CVM. • Não tem protecção contra falhas de modo comum.

Finalmente, há duas conclusões importantes a tirar desta análise:

- Os blocos funcionais devem ser construídos com componentes fiáveis, sob pena de uma arquitectura TMR deles derivada ter um tempo de missão muito reduzido.
- A fiabilidade do CVM é fundamental, devendo ser tão simples (e robusto) quanto possível.

Este segundo aspecto é particularmente importante na prática e levou ao desenvolvimento das arquitecturas mistas, que veremos mais adiante.

3.2.1.2 Arquitecturas Dinâmicas

A relação entre os diversos tipos de falhas depende muito das condições de funcionamento, mas em regra as falhas permanentes são muito menos frequentes que as temporárias e, nalgumas aplicações, as réplicas podem ser evitadas, detectando o erro e reiniciando o sistema. No entanto, se a especificação abranger falhas permanentes, a replicação física é necessária e os blocos funcionais serão substituídos à medida que avariarem. Desta forma, estas arquitecturas têm menor sobrecarga de confiabilidade ou, para o caso de sobrecargas semelhantes, permitem um tempo de missão superior, se os blocos redundantes permanecerem desactivados até ao momento de substituir um que avarie. A desvantagem maior é o intervalo de latência entre a detecção e a recuperação, mas em contrapartida dispensam o votador, substituído por mecanismos de detecção de erro que se obtêm com os dois métodos indicados na tabela 3-3.

Tabela 3-3. Métodos de detecção de erro

<i>Duplicação e comparação</i>	• Facilidade de projecto, por cópia do bloco funcional. • Elemento comparador mais simples que um CVM.
<i>Códigos de detecção de erro</i>	• Aplicáveis só a estruturas normalizadas (e.g. barramentos). • Difícil extensão a lógica não estruturada e de controlo.

Nestas arquitecturas, a detecção e correcção do erro são problemas distintos: a correcção requer informação adicional, que pode ser fornecida com um terceiro bloco funcional (o que nos leva de novo à arquitectura TMR) ou com detecção de erro individual num sistema duplo, neste caso denominado *duplex*. Como exemplo de uma arquitectura deste tipo temos a Redundância Comparativa [Phi96], que utiliza três blocos funcionais replicados, mas evita o CVM comparando apenas dois e passando o controlo para o terceiro bloco funcional, em caso de erro.

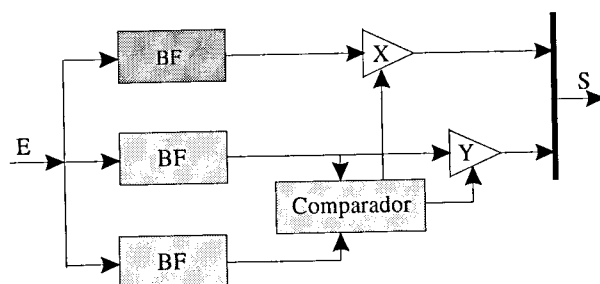


Figura 3-5. Redundância Comparativa

A análise matemática evidencia que um sistema de redundância dois tem um tempo de missão superior a um de ordem três. No entanto, o terceiro bloco funcional (sobre o qual não há qualquer informação) poderá ter já uma falha latente (e indetectada), o que impede, quanto a nós, esta solução de ser classificada como segura. Uma vez que a hipótese de falta única é pressuposta (tal como num TMR), a primeira falta *tem de* ser detectada (o que acontece implicitamente num TMR) e, de acordo com os critérios de segurança, consideramos que só a paragem é segura.

3.2.1.3 Arquitecturas Mistas

As arquitecturas mistas procuram essencialmente prolongar o tempo de missão e reduzir o custo através de uma simbiose das vantagens das duas anteriores. Em termos gerais, são arquitecturas NMR com elementos redundantes e reconfiguráveis, onde se procura melhorar a fiabilidade dos blocos funcionais e do elemento de decisão, e podem ser representadas pela arquitectura de *Redundância Auto-Purgável (Self-Purging, SP)*, que se ilustra na figura 3-6.

Nesta variante NMR, o CVM é substituído por um elemento mais simples denominado *votador e selector de desacordo (VSD)*. Esta simplificação é possível porque cada bloco funcional passa a ter associado um bloco de detecção de erro entre a sua saída e a do VSD (*Sel*), que o desactiva

em caso de desigualdade. Evita-se assim ao VSD a função de desactivar os blocos avariados, simplificando-o e tornando-o em consequência mais fiável.

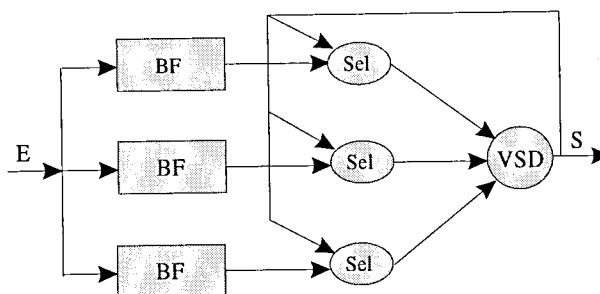


Figura 3-6. Redundância auto-purgável

No entanto, para evitar resultados incorrectos, o VSD terá um limiar mínimo de entradas necessário para funcionamento, normalmente $\lfloor (N/2)+1 \rfloor$. Desta forma, a representação simbólica $SP(4,3)$ identifica um sistema com quatro blocos funcionais e limiar três, uma arquitectura apropriada para aplicações críticas de vida longa e predominante em sistemas aeroespaciais [Skl76, Lal85].

Uma configuração alternativa, é a Redundância Modular “Sift-Out” (SMR), onde as funções de detecção e desactivação são passadas a um circuito independente, que controla o VSD e permite, tanto a este como aos blocos funcionais, a máxima fiabilidade relativa [Lal85].

3.2.1.4 Arquitecturas Auto-reparáveis

As arquitecturas auto-reparáveis diferem das anteriores essencialmente por disporem de blocos funcionais redundantes, que se mantêm desactivados até serem necessários para substituição. Um exemplo é a *Redundância Híbrida* ilustrada na figura 3-7, abreviadamente denominada $H(N,S)$, onde N é o nível de redundância activa e S o número de elementos suplentes. Note-se que um sistema $H(3,1)$ terá um encargo idêntico ao de um sistema $SP(4,3)$, mas o quarto bloco funcional mantém-se aqui inactivo. Esta diferença é importante porque permite um tempo de vida (bastante) superior, mas em contrapartida, o processo de reintegração do bloco funcional inactivo implica um elemento de selecção mais complexo (*Sel*).

Uma das principais vantagens desta arquitectura é permitir que seja efectuada primeiro a substituição e depois o teste do bloco funcional suposto defeituoso e, se a falha foi temporária, reintegrar de novo o bloco funcional no grupo S .

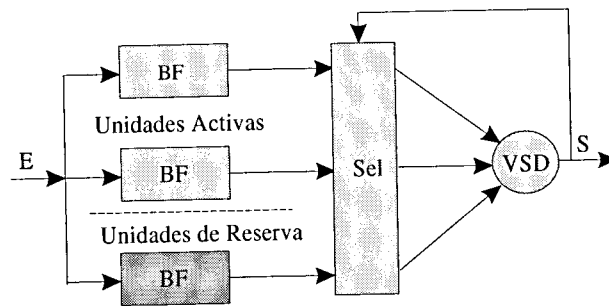


Figura 3-7. Redundância Híbrida

O número óptimo de réplicas está intimamente relacionado com o tempo de vida previsto para o sistema, uma vez que um número excessivo afecta a complexidade do votador e a fiabilidade dos elementos de detecção e comutação [Thi92]. Nos casos mais relevantes, N é quase sempre igual a três, mas pode ser superior se for necessário considerar a possibilidade de ocorrerem *fmc* em dois ou mais blocos activos, levando o VSD a rejeitar todos os suplentes que estejam bons.

Uma implementação alternativa deste grupo denomina-se *5MR Reconfigurável*, diferindo de uma 5MR normal (que tolera duas falhas) pelo facto de, na primeira falha, passar a H(3,1). A segunda falha converte-o em TMR, o que permite ainda suportar uma terceira falha. Outra variante é a *Reconfiguração Hierárquica* ilustrada na figura 3-8, onde o sistema é constituído por um conjunto de módulos (M), sucessivamente divididos em circuitos de missão (CM) e estes em blocos funcionais (BF), até um nível de profundidade conveniente à implementação.

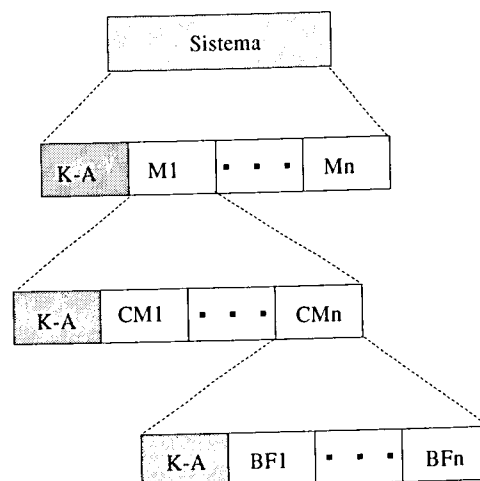


Figura 3-8. Redundância Hierárquica

Nesta arquitectura, cada nível tem réplicas redundantes, que podem substituir as avariadas, sob o controlo do elemento decisor, aqui denominado *Kill-area* (K-A), o que configura uma aplicação encadeada da redundância híbrida. Esta solução tem interesse teórico mas a sua aplicação é dificultada pelas interligações necessárias, que nos circuitos VLSI implicam um custo excessivo em silício e rendimento de fabrico muito baixo [Thi92, Chen93].

3.2.1.5 *Arquitecturas específicas*

Há diversos casos em que podem ser utilizadas arquitecturas com particularidades específicas, como a *Arquitectura inspirada na Biologia* [Tem97], um caso particular de uma área emergente de sistemas auto-reparáveis, denominada *Evolvable Hardware* [Gar95], que se inspira nos sistemas biológicos para explorar a flexibilidade de reprogramação das FPGA (principalmente em versão RAM). No entanto, este tipo de soluções conduz a um aproveitamento da lógica da FPGA inferior a 10%, e a algoritmos de reconstituição complexos e, por enquanto, totalmente dependentes de sistemas hospedeiros (exteriores ao CI), contrário ao conceito original. Para além disso, estas arquitecturas aplicam-se somente a estruturas reticulares ou matriciais e, tal como no caso anterior, exige uma elevada densidade de conexões disponíveis.

Outro caso é a *Degradação Controlada*, em que o sistema não dispõe de redundância espacial (ou é mínima), e em caso de falha passa a funcionar com as capacidades parcialmente afectadas, desactivando os blocos avariados até um limiar mínimo de operacionalidade. Esta solução tem a vantagem de poupar réplicas, onde não sejam justificadas devido ao custo, volume ou energia, mas o conceito de operacionalidade é difícil de especificar, principalmente se as funções perdidas forem vitais.

Podemos ainda referir as unidades regulares, do tipo memórias e barramentos, que não envolvem o processamento da informação e permitem a utilização de códigos detectores ou correctores de erro, com custos limitados [Karr93].

3.2.2 *Tolerância a Falhas em Programação*

A programação, reconhecidamente a área mais crítica hoje em dia na confiabilidade, afasta-se do tema central desta tese, a tolerância a falhas física a nível de CI. No entanto, consideramos que há interesse em referir alguns tópicos, úteis na medida em que a existência de microprocessadores num sistema permite geralmente reduzir a redundância espacial, em particular se existem tempos mortos de processamento. Para este fim, utilizam-se diversas

técnicas de redundância em programação, como a *verificação de integridade* de rotinas e dados (com valores conhecidos e seus duais, ou com operandos deslocados) [Lev94, Silva98], o *processamento redundante* com replicação de cálculos ou execução concorrente e verificação, o recurso a *rotinas diferentes* para obtenção dos mesmos resultados, a *duplicação* no armazenamento com limitação de escrita [Silv98], ou ainda por rastreio (*trace*) de acontecimentos excepcionais e captura das respectivas condições, o que permite caracterizar situações de funcionamento não completadas correctamente e facilita a depuração posterior.

Na prática, os sistemas confiáveis estão sujeitos a exigências de fiabilidade tão restritivas que a depuração é inviável em condições normais. Deste modo, e para acelerar o processo de depuração, particularmente na programação, utilizam-se métodos de *injecção* de faltas, através de pinos dedicados ou por alteração de rotinas ou parâmetros, ou ainda com radiação de alta energia [Mad94]. É ainda importante não esquecer as interfaces com o utilizador, tendo em conta que os graus de liberdade de actuação no sistema devem estar preparados para que a inexperiência, ou o descuido pontual do operador, não permitam acções passíveis de levar a estados não previstos e não testados. Uma interface por menus, que conduza apenas às sequências testadas, é mais segura que botões de selecção e, provavelmente, as falhas latentes nunca se manifestarão.

• *Salvaguarda Periódica e Recuperação*

O armazenamento periódico da informação processada, suposta boa (*Checkpointing*), constitui uma das técnicas mais utilizadas em programação, que permite recuperar de um erro sem ser necessário reinicializar completamente o sistema. Os períodos de salvaguarda podem ser fixos ou variáveis, permitindo estes sobrecarregar menos o sistema. Para salvaguardar a integridade da informação armazenada, efectua-se uma cópia com protocolo de escrita limitada, podendo esta informação ser utilizada para a recuperação por duas formas:

- *Retrocesso (Roll-Back)*: a rotina recupera a partir da última salvaguarda e repete as instruções decorridas, mas algumas partes do programa podem ser particularmente críticas e gerar informação errónea, se repetidas.
- *Antecipação (Roll-Forward)*: neste caso a informação armazenada é aproveitada para recomençar o processamento numa fase posterior [Pra94], mas esta solução pode induzir instabilidade no sistema controlado.

A recuperação envolve um conjunto de operações automáticas ou manuais, que permitem retomar a tarefa de forma *Completa* ou *Degradada*, ou ainda, se tal já não for possível,

desencadear o *Fecho em Segurança*. Outra questão é o nível de actividade no sistema influenciar significativamente a taxa de falhas [Lev82] e, existindo elementos de memória, muitos erros originados por falhas temporárias são armazenados e ficam latentes, só se propagando se utilizados. Para evitar a contaminação daí resultante, estima-se o MTBF e criam-se mecanismos para *refrescar* o programa, encerrando-o em segurança e reiniciando-o a intervalos inferiores ao MTBF estimado, numa técnica denominada *Rejuvenescimento da programação*. Foi ainda proposta a *micro-recuperação* no circuito, uma técnica a nível de instrução [Ora96], com aplicação em processadores recentes [Cle98].

3.2.3 *Sistemas de Tempo-Real*

Conforme o tipo de aplicação para que são projectados, os sistemas podem ser de *utilização corrente*, em que o tempo é considerado um recurso não prioritário (inesgotável) e a optimização centrada nos recursos espaciais, ou de *tempo-real*, que impõem limites temporais (*deadlines*), devendo o projectista garantir padrões mínimos de confiabilidade e de previsibilidade [Kim98b]. Por outras palavras, um sistema confiável pode deixar de o ser quando sujeito a restrições temporais, que podem resultar de três razões principais:

- Fornecer determinada capacidade de processamento num tempo limitado: a teoria de controlo evidencia a sensibilidade dos sistemas físicos aos atrasos na malha de realimentação, onde as unidades electrónicas geralmente se inserem, e os cálculos têm de ser garantidos em tempo útil ou podem gerar instabilidade.
- Sincronização em sistemas com redundância, em particular quando se utiliza diversidade espacial e/ou de programação, ou quando se força um desfasamento intencional [Lal85].
- Nos casos em que é implementada a salvaguarda periódica, particularmente se esta for a intervalos fixos, uma situação mais restritiva do que com intervalos flexíveis.

A maioria das aplicações de tempo-real críticas, de que são exemplo a aviação [Lal94] e o vaivém europeu [Roq98], é projectada com intervalos de actualização que permitem ao sistema de controlo falhar um ou dois tempos limites consecutivos, mas não mais, sob pena de originar falhas aleatórias (também denominadas *bizantinas*), tendo sido propostas duas alternativas principais para contornar este problema:

- *Cálculos Imprecisos*: a teoria das *Imprecise Computations* [Liu94] divide os programas em rotinas *obrigatórias* e *opcionais*: as primeiras, críticas, asseguram um

funcionamento seguro e devem ser executadas sempre, enquanto as segundas, acessórias, permitem refinar a qualidade de serviço mas podem ser abandonadas, se ocorrer uma sobre-ocupação momentânea inesperada ou se o sistema estiver em funcionamento degradado. O problema desta abordagem é assegurar a previsibilidade na repartição de tarefas, o que pode necessitar de estimadores dinâmicos para escalonar as rotinas acessórias.

- *Votação por Aproximação*: Em sistemas com réplicas distintas por diversidade de projecto, ou com sensores e canais independentes, tende a existir alguma diferença nos resultados gerados e, neste método, define-se uma *janela de aceitação* em bits menos significativos. Nos aviões F16 e B737, por exemplo, certas unidades de controlo permitem este tipo de desacordo [Lal94], mas esta forma de votação aproximada só é aplicável a sistemas que tratam dados ordenados. Refira-se ainda que a largura da janela é importante, para evitar falsos alarmes, mas sem deixar de detectar faltas que originem erros superiores a um determinado valor crítico.

3.2.4 Exemplos

Face à impossibilidade de analisar extensivamente as inúmeras referências a sistemas tolerantes a falhas [Toy78, Lal85, Joh89, Sie90, Sie91], optámos por uma abordagem a três níveis: primeiro a referência a três aplicações representativas, a que se segue uma breve descrição de três sistemas que foram desenvolvidos com objectivos próximos da área desta tese, e por último a análise de três aplicações recentes, que envolveram CIs tolerantes a falhas próximos dos nossos objectivos.

- **Aplicações representativas**
- A aviação comercial é tradicionalmente a aplicação mais exigente dos sistemas confiáveis, envolvendo muitos utilizadores que pretendem não correr riscos e estão dispostos a pagar por isso. As normas de aprovação dos aviões colocam exigências típicas de 10^{-10} falhas por hora, num voo de referência de 10 horas, e consideram-se situações extremas, desde falhas de origem aleatória até impactos de projecteis que danifiquem parte do avião, o que exige a separação física das réplicas [Lal94]. Nos sistemas denominados *fly-by-wire* (sem actuação mecânica entre a consola de comando e os mecanismos actuados), os sistemas de controlo de voo são (pelo menos) *triplicados*. Nos exemplos mais recentes (aviões B767 e A340), o

controlo de lemes tem todos os módulos triplicados até aos motores que accionam cada leme. Estes motores são também triplicados e a votação decidida em cada um deles, por um circuito (linear) que detecta a corrente excessiva resultante de uma eventual oposição aos outros dois. A simplicidade destes últimos circuitos permite-lhes elevada fiabilidade, secundada pelo facto de, mesmo que avarie um deles, a composição de forças entre os motores ser de 2-1.

- O supersónico Concorde é uma caso à parte na aviação comercial. A sua forma esguia, devido às características de voo a velocidade elevada (Mach 2), desloca o centro de gravidade mecânico para trás do centro vectorial das forças em voo e torna o avião instável, impossível de controlar sem o auxílio de sistemas em malha fechada [Ste96]. Como a altitude de cruzeiro é elevada para reduzir o atrito (15-20 Km), a *dose acumulada de radiação* cósmica a que os componentes são sujeitos é também elevada [Vel98]. Uma vez que os sistemas digitais (memória volátil em particular) são mais vulneráveis, foi preferido um sistema analógico secundado por um hidráulico, numa época em que a electrónica digital não era ainda suficientemente fiável. O voo dinamicamente instável tem vindo a ser aperfeiçoado em aviões com asas de geometria variável, para melhor capacidade de manobra, e actualmente é investigado no desenvolvimento de aviões mono-asa, que procuram compatibilizar a poupança de combustível com elevadas velocidades e capacidade de carga, mas coloca severas exigências aos sistemas de controlo, devido à instabilidade dinâmica.
- O vaivém espacial dos EUA é outro exemplo paradigmático no domínio dos sistemas tolerantes a falhas, onde soluções precursoras dos actuais aviões *fly-by-wire*, foram condicionadas pela tecnologia dos anos 70 [Sk176]. Apesar de os computadores terem sido já modernizados, a arquitectura original mantém-se e é um dos raros sistemas previstos para suportar mais de uma falta. O vaivém europeu, actualmente em desenvolvimento, beneficia já das vantagens da tecnologia actual e, mesmo sendo os requisitos semelhantes, as abordagens diferem na simplificação de barramentos, facultada pela redundância a nível de processador [Roq98]. Entre os problemas que os projectistas encontraram, destacam-se a elevada confiabilidade necessária numa missão tripulada e as fases de lançamento e reentrada na atmosfera que, embora de curta duração, são críticas. A arquitectura utilizada implementa diversidade espacial (física) e de programação, com cinco computadores, quatro do mesmo fabricante e idênticos e o quinto de fabricante diferente, a executar um programa simplificado e mais facilmente depurável. Dos quatro primeiros, três funcionam em simultâneo, indistintos em termos de circuitos, programa e tarefas, sendo o quarto utilizado para verificar a

concordância dos três resultados, com funções de *votador*. No entanto, é possível reconfigurar as funções dos quatro computadores na nave, ou a partir de terra, para salvaguardar o ponto mais frágil destas configurações, que é o votador. O quinto computador assegura apenas cálculos de sobrevivência, e só quando o sistema principal deixar de garantir parâmetros de operacionalidade (falha em dois computadores), o controlo passa a manual, auxiliado pelo quinto computador, um comportamento denominado “*fail-safe, fail-soft*” [Sto96]. Há ainda seis barramentos de 64 bits, com um bit de paridade por cada oito bits de dados, numa configuração de duplo TMR⁶.

Os três casos acima pertencem ao grupo de sistemas de mais elevada exigência de confiabilidade, denominado *life-critical*. Um outro grupo, menos crítico, são os *critical-but-not-life-critical*, ou simplesmente *critical*, de que são exemplo os aviões de combate, em que a tripulação dispõe de pára-quedas, o que permite reduzir muito os requisitos de confiabilidade. Outro exemplo é a gestão das entradas de ar nos reactores do Concorde, necessárias devido à variação de velocidade e densidade do ar nas altitudes a que voa, e que na ocorrência de um erro são retidas na última posição segura.

As aplicações com energia nuclear são outro dos casos onde a confiabilidade é necessária, em parte devido à elevada *radiação* ambiente. Nestas aplicações de ciclo lento não são exigidas respostas rápidas, mas há unidades de controlo que só podem ser reparadas com o núcleo desactivado, uma operação de custo elevado e sempre morosa. São ainda representativas as bases de dados bancárias, as centrais telefónicas (não só para assegurar a qualidade de serviço como para garantir a inviolabilidade das comunicações), os armazéns automatizados, unidades químicas ou de fusão de metais e a vigilância de linhas de produção de alimentos e produtos farmacêuticos [Lan92, San97d]. A tabela 3-4 indica os requisitos típicos para os sistemas que referimos, em relação às falhas que podem gerar avarias catastróficas, onde se refere o índice de confiabilidade definido na secção 3.1.1 e os períodos de missão.

É de notar que, para testar um sistema com os requisitos da aviação civil, supondo um teste contínuo, a primeira falha só deveria ocorrer após mais de um milhão de anos, pelo que com 10.000 aviões em teste, seria preciso mais de um século para a primeira falha ser detectada. No entanto, para além do período de missão, o nível de confiabilidade reduz-se drasticamente.

⁶ - As interligações são frequentemente descuradas, mas podem constituir uma das partes mais críticas em qualquer sistema.

Tabela 3-4. Características de confiabilidade em diversas aplicações

Aplicação	Índice de confiabilidade	Período de missão
Aviação Civil	10	voo de 10H
Vaivém espacial dos EUA	10	voo de 2H
Aviação Militar	7-8	voo de 1-2H
Sistemas de controlo nuclear	7-8	semanas - meses
Equipamentos para vigilância médica	6-8	dias - semanas
Sistemas bancários, ATM	> 5	dias - semanas
Indústria Química, Farmacêutica	> 5	dias - semanas

- **Sistemas tradicionais próximos da área desta tese**

Entre os sistemas desenvolvidos com aplicação específica, que mais se aproximam do tema dos CI tolerantes a falhas, podemos realçar os seguintes, representativos por terem sido utilizados em aplicações espaciais, embora algumas das soluções a que recorrem estejam hoje desactualizadas pela evolução recente da microelectrónica:

- O computador *FTMP* (*Fault Tolerant Multi-Processor*) destina-se aos sistemas utilizados na aviação comercial, com requisitos de 10^{-9} falhas por hora numa missão de 10 horas [Hop78]. É constituído por uma arquitectura TMR dupla quanto a componentes, com barramentos redundantes em configuração 5MR reconfigurável e sinais de E/S duplicados. A configuração mínima permitida corresponde a ter um dos TMR completamente operacional.
- O *SIFT* (*Software Implemented Fault Tolerant computer*) é também um computador, desenvolvido com base numa especificação semelhante à do FTMP, mas que implementa técnicas de detecção, diagnóstico e reconfiguração em programação, procurando que a execução seja assegurada mesmo em presença de falhas internas [Wen78]. A redundância física que existe é optimizada implementando uma forma de cálculo impreciso: quanto mais crítica for a rotina, maior o número de subsistemas em que será executada em paralelo.
- O computador *FTBBC* (*Fault Tolerant Building Block Computer*) constitui um dos primeiros casos de sistemas destinados a aplicações críticas de vida longa (interplanetária), construídos a partir de componentes comerciais [Lal85]. A fiabilidade limitada destes componentes, levou a incluir no projecto redundância física e circuitos de monitorização, numa versão auto-verificável.

- **Aplicações recentes com utilização de CIs específicos**

Entre os casos recentes com afinidade na área desta tese, optamos por destacar os seguintes:

- O *AE11 Fail-Stop Controller*, é um microcontrolador desenvolvido por um consórcio europeu (Bosch, Siemens, Porsche, Mercedes, Alcatel e Thompson) [Boh97], que se destina a substituir sistemas tradicionais duplicados, para minimizar custos e espaço em aplicações com grande volume de produção, a mais relevante das quais é a indústria automóvel. A figura 3-9 esquematiza a interdependência dos diversos tipos de teste implementados neste componente, que permite ao AE11 parar o funcionamento num curto espaço de tempo após ocorrer uma falha, o que é compatível com os requisitos dos sistemas automóveis, ferroviários ou de controlo de tráfego.

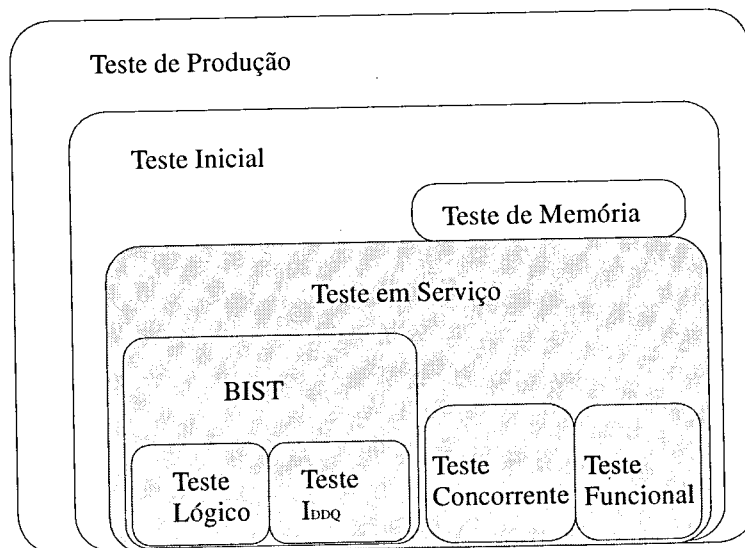


Figura 3-9. Estrutura do Teste no AE11

No essencial, este processador de 8 bits é compatível em programação com o CPU da família de microcontroladores 80C51 e agrega várias técnicas de *detecção* de erro, procurando impedir o funcionamento em presença de falhas (como o nome indica), não sendo classificado como tolerante a falhas. Partindo do pressuposto de que uma cobertura de faltas da ordem dos 99% satisfaz as aplicações alvo, foram implementadas técnicas de detecção por codificação em nós e barramentos, bem como na ALU e memória interna. Alguns elementos sequenciais, em que tal não é possível, foram duplicados e as suas saídas comparadas. Adicionalmente, inclui monitorização de corrente (I_{DDQ}) e também de V_{cc} , para suspender o processamento quando se excedem as margens de segurança. Dispõe também de monitor

térmico numa segunda pastilha de silício, de um circuito guardião (*watchdog*), bem como de infra-estrutura BST, utilizada para activar periodicamente um processo de auto-teste interno (BIST).

- O *Fault-Tolerant Computer for the Automated Transfer Vehicle (FTC-ATV)* corresponde a uma quarta geração de sistemas tolerantes a falhas, desenvolvida pela *Matra Marconi Space* para o vaivém da Agência Espacial Europeia, cuja arquitectura se representa na figura 3-10.

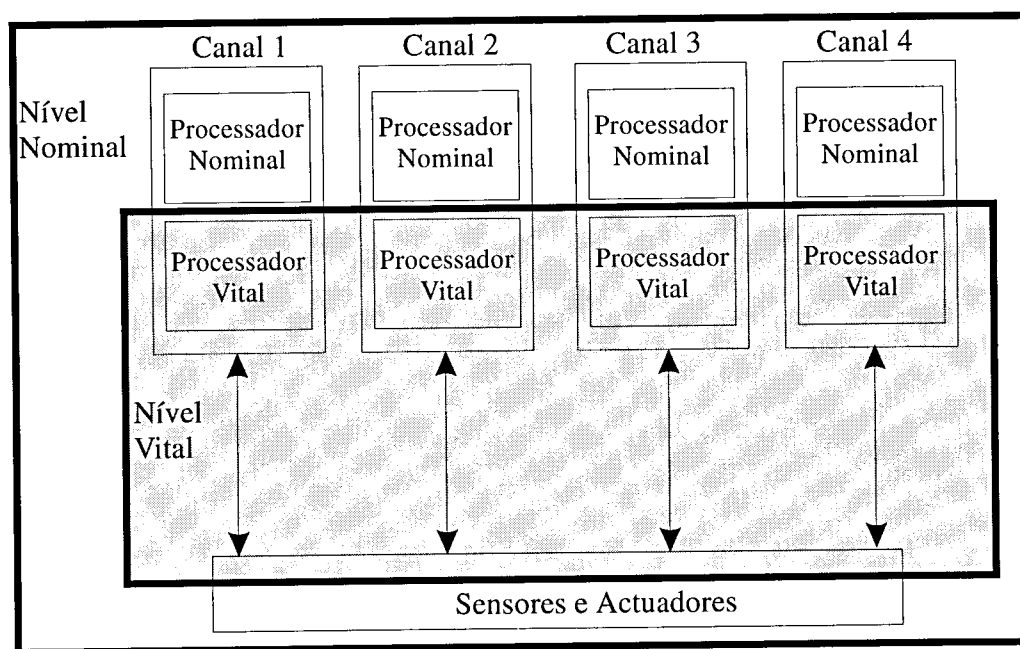


Figura 3-10. Arquitectura do computador tolerante a falhas para o ATV

Este computador integra quatro *canais* (módulos) votados por maioria, que permitem suportar uma falha aleatória e manter uma arquitectura TMR com falha permanente num dos canais [Roq98]. Cada módulo é constituído pelo CPU (arquitectura proprietária de 32 bits), memória e sinais de E/S (para o exterior e de comunicação inter-módulos). O centro nevrálgico desta arquitectura é constituído pelo CPU dual, onde cada réplica tem tarefas distintas:

- *Processador Nominal*: executa as tarefas normais, sem controlo especial de falhas. Desta forma maximiza-se a capacidade de processamento.
- *Processador Vital*: é uma réplica do Nominal, com memória que este pode ler mas não alterar, que mantém a supervisão permanente, pelo que não tem tempo para tarefas externas.

Os quatro módulos estão interligados por barramentos, com largura de banda suficiente para compararem resultados num intervalo sincronizado todos os 20 e 100ms. Em cada módulo, o Processador Vital fornece aos outros resultados codificados do seu gémeo Nominal, sendo a tarefa de comparação e detecção da responsabilidade de cada módulo (logo distribuída). Existem ainda circuitos de monitorização que detectam falhas de comunicação.

No caso de falhar o nível Nominal, o Processador Vital associado permite assegurar apenas as funções críticas, como a descolagem, a reentrada na atmosfera, a aterragem e os sistemas de apoio à vida, mas não as tarefas acessórias. Deve entender-se que estes serviços são assegurados a nível do canal, mas o sistema quádruplo deverá manter a funcionalidade TMR. Se a falha ocorrer no nível Vital, o Processador Nominal deixa de ter intervenção activa no fornecimento de dados, mas pode ainda detectar erros por comparação de resultados com os outros módulos. Um outro sistema, com características semelhantes às deste, é descrito em [Urb98].

- O *G4-FT CMOS Mainframe*, o último dos exemplos aqui apresentados, é um computador baseado num processador de quarta geração, proprietário da IBM, que é o primeiro em tecnologia CMOS com características gerais (desempenho e tolerância a falhas) melhores que as versões anteriores em tecnologia ECL [Spa98]. A densidade de integração permitida em CMOS, possibilitou um processador com as unidades de Instrução (I) e de Execução (E) duplicadas num só circuito integrado, que inclui ainda os mecanismos de comparação, de controlo e a memória, como se representa na figura 3-11.

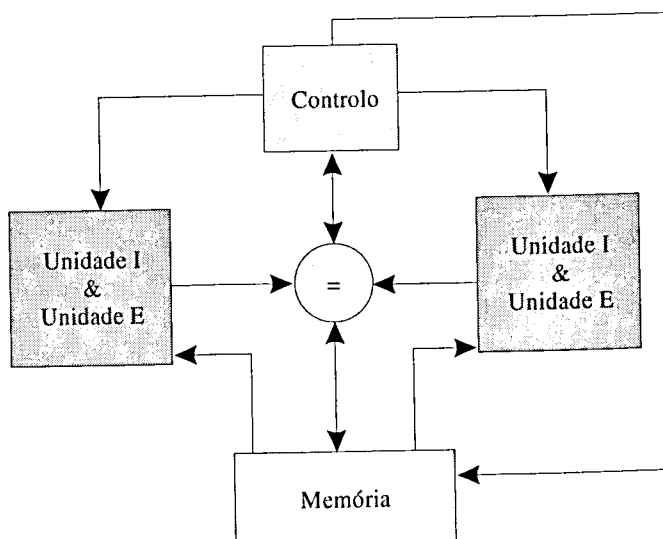


Figura 3-11. Arquitectura do processador do G4-FT

As técnicas de confiabilidade implementadas, que representam uma sobrecarga próxima de 40% no integrado do CPU (para além dos blocos duplicados), permitem projectar cartas tolerantes a falhas que utilizam processadores duplicados. A nível de sistema, estas cartas permitem construir unidades auto-reparáveis, sendo utilizada uma fonte de alimentação independente para cada carta suplente, que serão mantidas inactivas até serem chamadas a substituir uma carta que avarie. No entanto, a acessibilidade limitada, resultante da integração do equivalente a 400 circuitos ECL por CPU num só integrado CMOS, com cerca de oito milhões de transístores, obrigou a desenhar um novo processador, dividido em quatro módulos.

3.3 *Técnicas de projecto baseadas em Codificação*

A dificuldade ou impossibilidade física de inserir réplicas num componente (só recentemente surgiram os primeiros CI que utilizam replicação, com realce para o último exemplo da secção anterior), levou a que as técnicas de tolerância a falhas actualmente mais exploradas a nível de CI, sejam as que se baseiam em códigos detectores de erros. Embora não seja esta a orientação que pretendemos dar às propostas que apresentaremos, devido à reconhecida dificuldade em implementar a codificação, existe um conjunto de conceitos associados a estas técnicas que importa ter em conta a nível de componente, como a capacidade de auto-verificação e a imunidade a falhas. Atendendo a que ambas se destinam a tratar determinados tipos de erro, começaremos por recordar alguns conceitos básicos sobre erros e códigos detectores. Os códigos correctores de erro, embora utilizados em programação e na transmissão ou armazenamento de informação, têm tido uma aplicação restrita a nível de CI.

3.3.1 *Erros e Códigos*

Independentemente das outras particularidades que distinguem os erros, e que são influenciadas pela implementação e pela tecnologia [Nic89, Nic98b], no tratamento a nível espacial consideram-se essencialmente os três tipos de erros apresentados na tabela 3-5.

Tabela 3-5. Caracterização dos erros mais relevantes nos CI

<i>Simple</i>	• Apenas 1 bit da palavra de informação (código ou dados).
<i>Unidireccional</i>	• Número indefinido de bits, todos no mesmo sentido.
<i>Múltiplo</i>	• Número indefinido de bits, em ambos os sentidos.

Os códigos podem ser classificados de acordo com a relação entre os bits ou entre as palavras. Quanto aos bits, que podem ser de informação (b_i) ou de codificação (b_c), os códigos definem-se como *separáveis* e *não-separáveis*. Por seu lado, a relação entre as palavras do código, origina uma classificação em *ordenados* e *não-ordenados*. A tabela 3-6 apresenta as características dos dois tipos de códigos, entre estes quatro, mais relevantes actualmente.

Tabela 3-6. Tipos de códigos mais relevantes

<i>Separável</i>	Os bits de informação (b_i) e os de código (b_c) são distintos.
<i>Não-ordenado</i>	Não existem dois vectores tais que um cubra o outro.

Entre os códigos de detecção conhecidos, os mais utilizados são os mais facilmente implementáveis ou que permitem construir circuitos de verificação mais eficientes, classificados como se representa na tabela 3-7. Em termos aproximados, podemos dizer que os códigos separáveis são mais fáceis de implementar mas originam circuitos de verificação com maior tempo de resposta.

Tabela 3-7. Tipos de códigos para detecção

Código	Tipo	Detecção
Paridade	Separável, não-ordenado	Erros simples
Berger	Separável, não-ordenado	Erros múltiplos, unidireccionais
Duplicação	Separável	Erros múltiplos
Complementar	Separável	Idêntico ao anterior. A réplica representa o complemento lógico.

Os códigos a utilizar em circuitos que *tratam* a informação (para além da transmissão e do armazenamento) devem espelhar as propriedades das operações efectuadas, como por exemplo os códigos de resíduos. No entanto, os circuitos que verificam este tipo de códigos são geralmente complexos, o que tem condicionado a respectiva utilização.

3.3.2 Circuitos Auto-Verificáveis

Os circuitos auto-verificáveis (*Totally Self-Checking*) são implementados sob a forma de blocos funcionais e circuitos verificadores (*Checkers*), como mostra a figura 3-12, para detecção concorrente do erro por meio de técnicas de codificação que asseguram propriedades invariantes na saída do bloco funcional, permitindo evitar o teste explícito das faltas [Smi78].

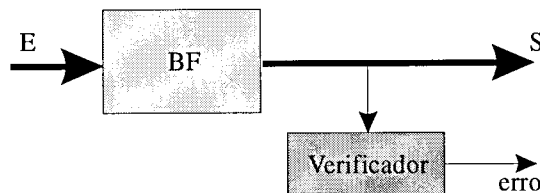


Figura 3-12. Esquema geral de um circuito auto-verificável

Num circuito combinatório com n entradas e s saídas podem ocorrer 2^n vectores (ou *palavras* na terminologia usual nesta área) de entrada e 2^s vectores de saída. Se forem permitidas somente $k < 2^s$ combinações das saídas, as palavras *legais*, as restantes $2^s - k$ palavras “proibidas”, ou *ilegais*, indicam situações anormais.

As desvantagens deste processo consistem na sobrecarga necessária para codificar e também para verificar, que onera o projecto, uma vez que o circuito de missão tem de ser redesenhado. Para além disso, o tempo de resposta dos verificadores pode condicionar a frequência de trabalho no circuito, bem como a contenção do erro, e não detectam faltas que gerem palavras de saída legais.

Tanto os blocos funcionais como os verificadores devem satisfazer um conjunto de propriedades que veremos agora, começando por considerar um circuito C , para o qual se definem:

- X : conjunto de palavras de entrada.
- Y : conjunto de palavras de saída.
- $A \subset X$: conjunto de palavras (válidas) do código de entrada.
- $B \subset Y$: conjunto de palavras do código de saída.
- F : conjunto de faltas do universo considerado (f : uma falta tal que $f \in F$).
- $C(x, f)$: valor da saída correspondente a uma entrada x , em presença de uma falta $f \in F$.
- $C(x, \emptyset)$: valor da saída correspondente a uma entrada x , na ausência de faltas.

- **Propriedades dos blocos funcionais em circuitos auto-verificáveis**

Os blocos funcionais utilizados nos circuitos auto-verificáveis, devem satisfazer as propriedades apresentadas em seguida [Nic88, Lub94, Nic98a]:

Propriedade -BF1: *Auto-Testável*

C é auto-testável para faltas em F se $\forall f \in F, \exists a \in A \mid C(a, f) \notin B$

Esta propriedade garante que em presença de uma falta no circuito, todas as saídas serão ilegais e, como tal, detectáveis.

Propriedade -BF2: *Imune a faltas*

C é imune a faltas em F se $\forall f \in F, \exists a \in A \mid C(a, f) = C(a, \emptyset) \vee C(a, f) \notin B$

Neste caso, a falta interna não afecta a saída, que se mantém igual à que era esperada, ou as palavras de saída são ilegais.

Propriedade -BF3: *Auto-verificável*

C é auto-verificável em F , se for simultaneamente auto-testável e imune a faltas em F .

Esta propriedade assegura que a primeira saída incorrecta será extra-código, mas só pode ser garantida para faltas simples e se forem exercitadas todas as palavras de entrada. Esta suposição implica que tenha de ser aceite (e respeitada na prática, o que pode ser difícil) a:

Hipótese 1: Entre a ocorrência de duas quaisquer faltas de F , decorre sempre um intervalo de tempo suficiente para que todas as palavras do código de entrada A possam ser aplicadas ao circuito.

Como a globalidade das palavras de entrada raramente é exercitada em condições normais e podem ocorrer faltas múltiplas, os circuitos auto-verificáveis são difíceis de implementar. Uma tentativa de aproximação a esta propriedade levou a definir outra [Smi78]:

Propriedade -BF4: *Fortemente imune a faltas*

O circuito C é fortemente imune a faltas, se é auto-verificável ou imune a faltas (para a primeira falta que ocorre) e em face de uma segunda falta, mantém essas propriedades para a falta múltipla resultante.

- **Propriedades dos Verificadores**

Os verificadores, para determinado código, podem por sua vez ser também auto-testáveis ou auto-verificáveis, distinguindo-se por terem duas ou uma só saída de indicação de erro, respectivamente, como se mostra na figura 3-13.

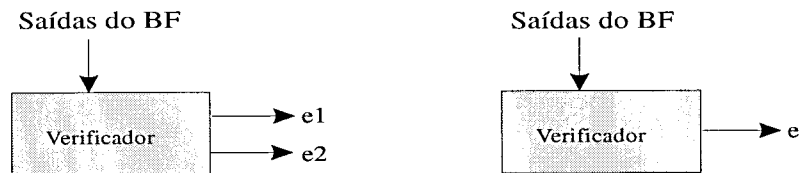


Figura 3-13. Verificadores auto-testável e auto-verificável

A indicação final de erro em qualquer circuito tem sempre que ser do tipo auto-verificável, em saída *passiva*. No entanto, um verificador deste tipo com (por exemplo) saída em “0” para palavras válidas e em “1” para erro, não pode ser auto-testável para faltas *sempre-a-0* na sua saída, só detectáveis se o bloco funcional gerar palavras extra-código, o que implica funcionar incorrectamente. Por seu lado, os verificadores auto-testáveis geram dois sinais, normalmente 01/10 se não há erro e 11/00 em caso de erro nas suas entradas, que não podendo ser directamente utilizados para indicar erro, devem ser convertidos num sinal passivo [Nic89]. O projecto dos verificadores procura que sejam seguros em relação a erros nas entradas e faltas neles próprios e, para além de deverem ter as propriedades auto-testável ou verificável, devem ainda satisfazer a propriedade seguinte:

Propriedade -V1: *Código Disjunto* (CD)

C verifica a propriedade CD se $\forall a \in A, C(a, \emptyset) \in B; \forall x \in (X-A), C(x, \emptyset) \notin B$

Esta propriedade implica que o verificador seja capaz de distinguir nas suas entradas as palavras correctas das erradas, e é suficiente para lhe permitir verificar o bloco funcional, mas deve também ser auto-verificável em relação às suas próprias faltas. Em circuitos complexos, pode haver blocos funcionais e verificadores em série, sendo corrente utilizar internamente verificadores auto-testáveis e relegar a capacidade auto-verificável para o último verificador na cadeia, que deve ser o mais simples e robusto possível [Hir91, Sit93].

A conjugação de blocos funcionais e verificadores que satisfaçam as propriedades indicadas, permite obter circuitos auto-verificáveis na condição de serem exercitadas todas as entradas do bloco funcional *e* do verificador, o que requer a aceitação de uma nova hipótese:

Hipótese 2: Após ocorrer uma falta no verificador, decorre um intervalo de tempo suficiente para que todos os vectores de A sejam aplicados ao bloco funcional, antes de surgir segunda falta no bloco funcional ou no verificador.

Supondo garantidas as hipóteses 1 e 2, um circuito auto-verificável tem o seguinte comportamento:

- Uma falta que ocorra no bloco funcional manter-se-á latente, sem modificar o funcionamento deste, ou gera palavras de saída ilegais, que serão detectadas pelo verificador antes de surgir uma segunda falta.
- Uma falta que ocorra no verificador manter-se-á latente e pelo menos uma das suas entradas, origina a indicação de erro, antes do aparecimento de nova falta no verificador.

Verificadores para circuitos sequenciais são difíceis de implementar, sendo a única abordagem que conhecemos a que é preconizada pelas *Strongly Code-Disjoint Languages* [Jan85], mas sem referências a aplicações práticas.

3.3.3 Circuitos Imunes a Falhas

Os circuitos imunes a faltas (*fail-safe*) foram desenvolvidos numa perspectiva diferente dos auto-verificáveis, procurando confinar as saídas erradas em tempo-real para evitar que se propaguem [Fuj85, Nic89]. Este tipo de circuitos é útil nas aplicações onde há um estado *seguro*, como no controlo de semáforos (vermelho é seguro), no controlo do trem de aterragem de um avião (descido é seguro) e nos casos em que o estado seguro permite avisar o operador em tempo útil.

A teoria dos circuitos imunes a faltas, em que os erros se convertem no estado inactivo, provocando confinamento unidireccional, foi desenvolvida para saídas de um só bit. A sua aplicação a saídas multi-bit implica que, em rigor, estas devam ser tratadas como um conjunto de bits individuais. Uma vez que a maioria dos sistemas utiliza barramentos incompatíveis com a individualização bit a bit, a generalização desta teoria, exposta em [Nic89], implica dividir o conjunto das palavras em subconjuntos de vectores seguros e não-seguros. Embora necessária, esta divisão não caracteriza completamente o tipo de resposta do circuito, que pode gerar uma saída:

- Segura:
 - em vez de outra segura.
 - em vez de uma não-segura.
- Não-segura:
 - em vez de uma segura.
 - em vez de outra não-segura.

As duas primeiras situações são seguras, mas as outras duas só podem ser caracterizadas conhecendo a alteração relativa entre as duas palavras, o que geralmente depende da aplicação e leva a um tratamento complexo dos casos em apreço.

Os circuitos imunes a faltas, que não requerem saídas redundantes nem verificadores, são implementados com estruturas de portas lógicas que geram erros unidireccionais, ou os confinam, assegurando que erros na saída de qualquer porta lógica se propagam até às saídas como valor seguro. São exemplos deste tipo a expansão de Reed-Muller, que utiliza apenas portas lógicas AND e XOR encadeadas, mas origina elevado número de níveis, o projecto OR-AND-OR, que permite reduzir o número de níveis a três e confina as faltas *sempre-a*, ou ainda o aproveitamento de estados redundantes [Bog95]. Sem detecção de erros, desnecessária uma vez que são confinados (unidireccionalmente), estes circuitos dependem da hipótese de falta simples e necessitam de mecanismos que assegurem a detecção num intervalo de tempo inferior ao do provável aparecimento de uma segunda falta, para impedir o circuito de fornecer saídas erradas.

3.4 Teste em Serviço

O teste executado com o circuito em serviço (*on-line*) pode ser, como já referimos, efectuado em regime calendarizado ou concorrente. Estes testes podem incidir no funcionamento lógico ou em medidas paramétricas, com as técnicas que a seguir descrevemos sumariamente, numa perspectiva de utilização nos circuitos VLSI.

3.4.1 Testes a nível de funcionamento lógico

Consideramos neste caso, algumas das técnicas mais correntes de teste baseado no funcionamento lógico, que podem ou não utilizar um modelo de faltas.

- *Calendarizados*

- *BIST* (*Built-In Self-Test*) - nesta técnica de auto-teste incorporado, a lógica auxiliar de teste é interna e testa o circuito de missão gerando a excitação com circuitos dedicados (*LFSR* ou *MISR*). Uma vez que o BIST recorre a técnicas não determinísticas ou de determinismo condicionado, implica a geração de um elevado número de vectores de teste e conduz à análise de assinatura para compactar a resposta, o que introduz problemas de indefinição (*aliasing*) [Iva88]. Um caso relevante deste tipo de circuitos são os *BILBO* (*Built-In Logic Block Observer*), vocacionados essencialmente para testar

circuitos regulares, e que podem ser configurados para gerar a excitação e tratar as respostas do circuito de missão, geralmente deslocadas posteriormente para análise no exterior [Koe79, Kim82]. Esta técnica tem evoluído em várias direcções, de que são exemplo o *CBIST* e o *C2BIST* [Chi97], que utilizam o conceito de autómatos celulares, bem como outras variantes que veremos no modo concorrente, sendo cada vez mais utilizadas técnicas auxiliadas ou controladas pela infra-estrutura BST, de que é exemplo o *scan-BIST* [Abr98, Kie98, Mar91].

- *Concorrentes*

Neste tipo de teste consideramos relevantes a duplicação (com comparação) e os circuitos auto-verificáveis. Estes últimos têm diversas limitações, nomeadamente o número máximo de entradas do verificador, apontado como 15 [Nic88, Lo88, Bar90, Jha93, Kar93, Boh97, Met97]. Para além destas técnicas, existem ainda as seguintes:

- *UBIST (Unified BIST)* [Nic88]: em condições normais, as propriedades auto-verificáveis são difíceis de assegurar, porque as saídas de um bloco funcional dificilmente cobrem as necessidades de teste do seguinte, devido à não ocorrência (proibição) de vectores de teste ilegais. A técnica UBIST procura ultrapassar esta limitação, aplicando ao circuito de missão um teste calendarizado, a intervalos muito menores que o MTTF, inserido ciclicamente para exercitar separadamente os diferentes blocos.
- *B²UBIST (Boundary Scan Board UBIST)* [Lub92, Lub94]: esta técnica é uma evolução do UBIST, em que os circuitos auxiliares testam alternada e ciclicamente os blocos funcionais e os verificadores de ordem par e ímpar, utilizando a infra-estrutura BST para levar a informação dos verificadores até à saída TDO. Embora este aproveitamento facilite uma maior resolução no nível de tratamento de sinal, o B²UBIST requer no entanto três sinais auxiliares, que activam a comutação entre elementos mas impedem a compatibilidade com a norma 1149.1.
- *BICST (Built-In Concurrent Self-Test)* [Sha88]: nesta técnica, um circuito de missão combinatório com n entradas e m saídas é testado em funcionamento por um circuito que procura vectores correspondentes aos que tem armazenados. No entanto, tal como o processo é descrito, está limitado a um pequeno número de entradas (quatro no exemplo) e os tempos de detecção crescem para o fim do teste. O intervalo necessário para completar um teste deste tipo foi já apontado como tendencialmente superior a 10^6 períodos de relógio [Asa96].

3.4.2 Testes a nível paramétrico

Para além de testes ao funcionamento lógico, podem efectuar-se também os de nível paramétrico, baseados em técnicas justificáveis quase só nos componentes VLSI, onde por um lado os custos dos sensores e a sobrecarga de teste se diluem no CI e, por outro, a testabilidade limitada obriga a procurar formas adicionais de monitorização.

- *Monitorização de Corrente (I_{DDQ})*

Esta forma de medição, abrangendo múltiplas portas lógicas, pode ser calendarizada ou concorrente [Bal91, Lo92] e abranger sensores de corrente auto-calibráveis ou não [Mee97]. Entre as vantagens, refere-se a potencial detecção antecipada de falhas [Haw98], sendo a degradação do desempenho do circuito a principal desvantagem, tanto no modo calendarizado como no concorrente [Fer98]. Como forma de ultrapassar esta limitação, foi sugerido testar partes inactivas de forma transparente [Riu97], mas a sua implementação prática tem restrições complexas. Para além disso, esta técnica é ineficiente nas geometrias nanométricas, onde a dispersão estatística (que existe nas geometrias de maior dimensão) deixa de ser válida e aumenta a indefinição do processo de teste [Fer98]. Notemos também, que nestas geometrias estão previstas tensões de V_{cc} entre 1,2V e 0,9V [SIA97], implicando o recurso a sensores específicos para baixas tensões [Miu98] e a uma escolha selectiva dos vectores utilizados neste tipo de teste (quando executado de forma calendarizada) [Gra98].

- *Monitorização Térmica*

Muitas falhas originam aumento da potência dissipada, detectável com sensores térmicos, o que possibilita esta técnica (*DfTT - Design for Thermal Test*), que não degrada a eficiência do circuito. No entanto, a distribuição dos sensores é complexa, e embora a temperatura possa indiciar potenciais falhas, não serve como critério decisório da sua existência. Esta limitação resulta de as falhas que são raramente activadas originarem uma contribuição térmica diminuta, enquanto as resistivas e temporárias só são (potencialmente) detectáveis com uma densidade de sensores incompatível com os custos de produção [Xia92, Mart97]. Embora não sendo uma técnica de resultados precisos, a monitorização térmica permite uma indicação válida no caso de falhas de curto-circuito (embora de resposta lenta nas condições actuais), tendo sido já proposta a sua reunião com a técnica UBIST [Sze96]. Notemos, contudo, que as utilizações actuais recorrem frequentemente a uma segunda pastilha de silício para o sensor.

3.5 *Conclusão*

Neste capítulo procuramos caracterizar a confiabilidade e identificar as técnicas e arquitecturas tolerantes a falhas conhecidas. O primeiro passo para construir um sistema confiável é a *prevenção*, minimizando a probabilidade de ocorrência de falhas no sistema, e só quando estas medidas são insuficientes se recorre às arquitecturas tolerantes a falhas. É importante enfatizar que estas arquitecturas só são eficazes se implementadas com componentes já de si fiáveis, uma vez que a redundância aumenta a taxa de falhas e encurta o tempo de vida útil do sistema.

Finalmente, resumimos as vantagens e inconvenientes das principais técnicas utilizadas no teste em serviço, com mais interesse nas concorrentes, que servirão de base à escolha da solução mais adequada para ser implementada num só CI.

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

Capítulo 6

Capítulo 7

Capítulo 8

Bibliografia

Anexos

**Circuitos Integrados Confiáveis
com Infra-estrutura BST:
Requisitos e Condicionantes**

4. Circuitos Integrados Confiáveis com Infra-estrutura BST: Requisitos e Condicionantes

A utilização de sistemas de controlo electrónicos tem vindo a expandir-se às indústrias automóvel, ferroviária, de equipamentos médicos, controlo de tráfego aéreo, viário e ferroviário, e outras que, ao contrário das aplicações tradicionais de tolerância a falhas (que justificam e permitem todo o investimento para obter o melhor sistema possível), têm volumes de produção e limites de custo que inviabilizam arquitecturas de elevada redundância. Neste contexto, a utilização de sistemas electrónicos depende principalmente de um custo baixo/moderado e comportamento confiável, características que os circuitos VLSI podem proporcionar. No entanto, as arquitecturas confiáveis tradicionais são afectadas por diversas limitações destes componentes, que nos levam a duas questões principais:

- Qual a sobrecarga aceitável num CI para otimizar a relação confiabilidade-custo?
- É possível simplificar as arquitecturas confiáveis tradicionais? Em caso afirmativo, como e até onde?

Para lhes responder, faremos primeiro uma síntese das condicionantes, tal como as entendemos no contexto da tecnologia actual, seguida pela discriminação dos objectivos que estabelecemos e que balizaram o desenvolvimento do trabalho, conduzindo à especificação (genérica) que será apresentada na última secção deste capítulo.

4.1 *Análise de Requisitos e Condicionantes*

No estado de conhecimento actual, não sendo esperadas novas arquitecturas [Joh89], a resposta à primeira questão colocada acima não tem solução única e depende de conceitos recentes sobre os quais não há ainda uma visão global, particularmente no relacionamento das infra-estruturas de teste BST com a tolerância a falhas e desta com os circuitos VLSI. Para identificar as respostas mais adequadas, optámos por uma estratégia alicerçada em seis vectores primordiais:

- Direcções ou recomendações a seguir na tolerância a falhas.
- Necessidades reais das aplicações.
- Condicionantes específicas dos componentes VLSI.
- Arquitecturas de tolerância a falhas mais adequadas aos componentes VLSI.
- Aproveitamento, ou reutilização, da infra-estrutura BST nestas áreas.
- Condicionantes do teste concorrente.

São estes aspectos que passamos agora a analisar nas subsecções seguintes.

4.1.1 *Recomendações e orientação*

Em 1992, ao apresentar no decurso do FTCS'22 as conclusões da análise efectuada durante esse ano, um grupo de trabalho [IFI99] iniciado na edição anterior desta conferência referiu que “*a prática em tolerância a falhas é mais arte que engenharia*” [Wei92], para enfatizar que a generalidade dos projectos tolerantes a falhas continuava a ter um objectivo específico, tal como nas décadas de 50 e 60, que os condicionava e originava soluções particularizadas. As consequências desta situação, dificuldade na reutilização e reduzido número de exemplares em funcionamento, levaram esse grupo a propor as seguintes recomendações:

- Sistematização de metodologias estruturadas e generalizáveis, que permitam ao projectista encaminhar a solução para arquitecturas de eficácia comprovada, devidamente documentadas e ensaiadas, reduzindo os custos de projecto.
- Necessidade de estabelecer critérios bem definidos para identificar os subsistemas críticos e não-críticos, coordenando a implementação da tolerância a falhas de acordo com essas regras, para evitar exigências excessivas no projecto.
- Desenvolver subsistemas reutilizáveis, permitindo o recurso a bibliotecas de módulos confiáveis com propriedades bem documentadas e testadas, para os interligar de uma

forma expedita, em lugar de desenvolver soluções de raiz de eficácia duvidosa e que implicam ensaios exaustivos, frequentemente tão caros como o próprio sistema.

- Reutilização extensiva de subsistemas, permitindo submetê-los a ensaios mais alargados do que seria possível numa aplicação única, facilitando o diagnóstico e optimização dos mecanismos de tolerância a falhas e aumentando a confiança em reutilizações futuras. A divisão do custo de projecto por múltiplos utilizadores, permite e justifica um tratamento exaustivo.
- A utilização de diversidade no projecto deve ser ponderada, uma vez que a experiência obtida nalguns casos parece questionar as potenciais vantagens.

Estas conclusões mantêm-se válidas actualmente e são particularmente interessantes dado que, na vida real, os prazos e as limitações de custo permitem depurar o sistema somente até ao nível aceitável para o cliente, correspondente a avarias suficientemente espaçadas e muito aquém de um tratamento exaustivo.

A acrescentar às conclusões do grupo citado, julgamos que será útil evidenciar a utilização crescente de COTS e módulos IP no projecto de sistemas confiáveis, pela importância que podem representar face à aceitação prática das recomendações propostas.

4.1.2 *Caracterização dos requisitos de Confiabilidade*

Muitas das novas aplicações que procuram sistemas confiáveis de baixo custo, necessitam deles em boa parte porque as condições de funcionamento são agressivas, como é o caso de módulos instalados no compartimento do motor de um automóvel, sujeitos a temperaturas que podem variar entre -20 e +70° Celsius. Situações análogas, com gamas alargadas de temperatura e humidade, ruído electromagnético, poeiras, vibrações e outras condições extremas, degradam e fragilizam os circuitos integrados que, no limite, podem ter de tolerar falhas físicas.

Existe a opinião generalizada, na comunidade da tolerância a falhas, de que o importante é corrigir o erro e a detecção da falha é secundária, mas o custo final e as limitações da própria implementação condicionam este ponto de vista. Como nota [Lap85], o intervalo de interrupção de serviço e a probabilidade de propagação do erro variam no sentido inverso da sobrecarga de confiabilidade, o que significa uma complexidade do projecto influenciada pelo nível de confiabilidade especificado. Uma vez que este nível depende essencialmente da forma como o erro é tratado, consideramos que os quatro níveis de confiabilidade com interesse prático são os indicados na tabela 4-1, por ordem crescente de complexidade.

Tabela 4-1. Níveis de confiabilidade e de tratamento de erro

Nível	Tratamento do Erro	Soluções
1	Deteção temporizada	Teste calendarizado ou amostragem funcional concorrente (*).
2	Confinamento	Duplicação com comparação ou circuitos auto-verificáveis.
3	Recuperação	Módulo capaz de retomar o funcionamento, sem gerar saídas erradas durante o intervalo de latência.
4	Mascaramento	Módulo com <i>correção de erro</i> imediata, sem perda de informação.

(*) - A amostragem tem sido essencialmente teórica, por falta de ferramentas adequadas.

Estes níveis, que permitem enquadrar os requisitos de confiabilidade e evitar exigências excessivas, conduzem a uma resposta à questão da sobrecarga aceitável na relação confiabilidade-custo. Assim, há aplicações que requerem, e apenas justificam, deteção de erro (níveis 1 e 2), mas noutras a correção é essencial e o custo inerente torna-se admissível (níveis 3 e 4), pelo que devemos procurar soluções a nível de CI para satisfazer ambos os casos.

Para além destes níveis, a generalidade das aplicações de tolerância a falhas que temos em vista está sujeita a restrições temporais, pelo que devemos também considerar as características dos sistemas de tempo-real, influenciadas por três factores principais:

- **Intervalo de latência aceitável**

Na generalidade das aplicações, encontramos intervalos de latência entre 10 ms e 2 segundos [Kun97], sendo os mais frequentes na gama 20-100 ms. Por outro lado, muitas aplicações utilizam ciclos de monitorização curtos, mas permitem tempos de resposta de vários ciclos para recuperar. Um dos casos mais restritivos é o controlo de combustível nos reactores dos aviões, geralmente com ciclos de monitorização de 20-40 ms, que podem tolerar três ciclos para recuperar sem problemas apreciáveis [Lal94]. No controlo dos lemes as exigências são semelhantes, mas os intervalos de latência podem ser um pouco maiores, devido à inércia do próprio avião. Nas aplicações aeroespaciais e nos sistemas terrestres, estes intervalos são pelo menos análogos, pelo que tomaremos para referência um intervalo de latência (T_L) de 50 ms.

- **Frequência de operação**

Numa aplicação confiável a frequência é reduzida ao mínimo necessário, sendo referidos na literatura casos desde centenas de KHz, em satélites com tempo de missão longo, a 33 MHz, na

indústria aeronáutica actual. Podemos assim considerar representativas as frequências entre 1 e 33 MHz, que devem ser interpretadas como as frequências de funcionamento nominal em qualquer circuito, não se restringindo à frequência de relógio nos circuitos síncronos.

- **Características dos sinais de entrada**

Nas análises em tolerância a falhas são geralmente supostos sinais de entrada com distribuição equiprovável, o que abrange a falta de informação sobre as características da fonte que os gera. Uma análise de múltiplas aplicações referenciadas permite distinguir três casos:

- Barramentos de dados (*data path*): a distribuição é geralmente equiprovável.
- Lógica não estruturada e de controlo: é impossível definir padrões para os sinais neste tipo de circuitos mas, em muitos casos, podemos no entanto estimar o estado provável (e.g. um sinal de leitura para a memória é normalmente activo em “0”).
- Barramentos associados a sinais físicos: a distribuição normal é frequente.

Uma vez que pretendemos tratar as entradas e saídas do circuito de missão, tomaremos a distribuição equiprovável como representativa, sem esquecer que outras distribuições permitem frequentemente explorar a maior probabilidade de alguns vectores de teste [Sal88].

4.1.3 A Confiabilidade nos componentes VLSI

A utilização de microprocessadores tem sido vantajosa pela flexibilidade permitida mas, em muitos casos, um ASIC (ou FPGA) pode actualmente ser mais adequado e fiável, por envolver menos recursos e frequências mais baixas para tarefas idênticas [Abra98]. Tendo em conta que a programação continua a ser a componente mais difícil de depurar em sistemas confiáveis, é previsível que, no futuro, os microprocessadores venham a ser relegados para aplicações de tratamento intensivo de informação [Roq98]. Existe assim um universo de aplicações que poderão vir a evitar microprocessadores, aumentando o interesse por técnicas de tolerância a falhas que não sejam baseadas em programação.

Ao analisar a confiabilidade dos componentes VLSI, sem excluir os microprocessadores, é de realçar que a fiabilidade e a densidade de integração têm aumentado continuamente, mas também que diversos factores limitam o aproveitamento dessas potencialidades na tolerância a falhas. Recordemos que a redundância nos circuitos integrados pode ter dois objectivos distintos, relacionados com as duas fases da vida dos componentes:

- Fase de projecto e produção: melhorar o rendimento que, para uma dada tecnologia de fabrico, decresce na razão inversa da densidade de integração, podendo descer a menos

de 50% nos componentes VLSI actuais e mesmo a 10% nos que abarcam toda a pastilha de silício (WSI, *wafer-scale integration*) [Haw98]. A inserção de blocos redundantes permite aos fabricantes aproveitar circuitos integrados defeituosos, por substituição selectiva, e melhorar o rendimento. Neste caso, de que são exemplo as memórias fabricadas com linhas e colunas adicionais, os blocos redundantes não ficam acessíveis para a fase de vida útil [She90, Cic91].

- Vida útil: permitir circuitos integrados confiáveis. Na tolerância a falhas pretendemos ter redundância em serviço, disponível e configurável pelo utilizador durante a vida útil do circuito integrado, e o rendimento de produção é menos relevante.

É o segundo tipo de redundância que pode vir a facultar circuitos integrados confiáveis a custo comparável ao dos equivalentes actuais, mas os defeitos de fabrico são geralmente múltiplos e podem afectar as arquitecturas tradicionais de tolerância a falhas, baseadas na hipótese de falta simples. Interessa pois considerar a sobrecarga de confiabilidade inserida no CI sob os dois prismas seguintes:

- **Impacto das falhas nas diversas arquitecturas conhecidas**

Nos circuitos integrados há dois grupos de falhas particularmente críticos: as que originam *fmc*, que afectam as arquitecturas replicadas, e as que geram erros unidireccionais, degradando os sistemas baseados em codificação [Nic91, Str93]. O contínuo avanço para geometrias submicrométricas afecta também a confiabilidade em tecnologia CMOS, com defeitos de fabrico que escapam aos testes de produção e podem reaparecer mais tarde em condições de funcionamento severas, inutilizando os testes baseados no modelo de faltas *sempre-a* [Mer97]. À medida que a densidade de integração cresce e os restantes parâmetros construtivos melhoram, tornam-se mais representativos os defeitos a nível de pinos, nos quais a potência dissipada e problemas térmicos tendem a originar falhas constituídas por ligações defeituosas, que passam a depuração e se manifestam apenas a temperaturas de funcionamento elevadas [Abre97]. O processo de soldadura também degrada as ligações internas e foi já sugerido replicar pinos em aplicações críticas [Str93], mas há fortes limitações de natureza física e prática a este respeito.

Este conjunto de problemas agravou-se nas geometrias abaixo de $0,25\mu\text{m}$ e tem sido referido como muito complexo nas geometrias inferiores a $0,1\mu\text{m}$, o sub-mícron profundo [Koe98, Cal98, Nic98b], que apresenta também uma elevada sensibilidade às partículas cósmicas ionizadas. Este tipo de falhas, até recentemente problemático só em aplicações aeronáuticas e

espaciais, onde a radiação pode ser 8-10 vezes superior à que existe ao nível do mar, é hoje muito mais crítico, tendo sido já detectada a ionização no sistema de controlo de um comboio [Che98]. Resulta daqui que o projecto para a confiabilidade nos componentes VLSI, independentemente de se utilizarem ou não réplicas, deve privilegiar arquitecturas com detecção de erro.

- **Custo final do circuito integrado**

Numa solução TMR implementada num só CI, uma réplica defeituosa inutiliza o integrado e a única solução é recuperar a filosofia utilizada para melhorar o rendimento de fabrico, o que significa produzir circuitos integrados com quatro ou mais réplicas, para se obter um integrado TMR operacional [Kor90]. A título de exemplo, consideremos uma pastilha de silício com 240 circuitos, fabricados com um rendimento (y) de 50%, o que conduz a um total de 120 circuitos integrados sem defeito. Para implementar um circuito integrado TMR, supondo o votador com encargo próximo de um circuito de missão, cada circuito integrado terá uma sobrecarga de confiabilidade comparável a outro com quatro réplicas. Daqui obtemos 60 integrados possíveis ($240/4$), com probabilidade de estarem bons igual à de estarem os quatro blocos bons, que é dada por $y = 0,5^4 = 0,0625$, o que conduz a um número de CI bons por pastilha de $60 \cdot y = 3,75$. Cada um destes circuitos integrados seria então 32 vezes mais caro que o circuito de missão ($120/3,75$), pelo que é preferível utilizar componentes independentes, o que possibilita 30 módulos TMR ($120/4$).

Supondo agora que o fabricante insere outra réplica para melhorar o rendimento de fabrico, cada circuito integrado teria o equivalente a cinco réplicas e a probabilidade de funcionar correctamente é a probabilidade (aproximada) de todos os circuitos estarem bons e de um em cinco poder estar avariado, dada por $y = 0,5^5 + 5 \cdot (1 - 0,5) \cdot (0,5)^4 = 0,1875$, circunstâncias em que se obtêm 9 circuitos bons por pastilha ($0,1875 \cdot 240/5$). Neste caso, o número de integrados TMR bons é quase triplo do anterior, mas a utilização de réplicas disponíveis implica lógica de reconfiguração. Se admitirmos que esta lógica acarreta uma sobrecarga idêntica a (mais) um circuito de missão, temos 40 circuitos potenciais ($240/6$), com $y = 0,5 \cdot (0,5^5 + 5 \cdot 0,5^4) = 0,09375$. Obtemos portanto, neste caso, 3,75 circuitos sem defeito, que é igual ao valor inicial.

Este exemplo ilustra os problemas associados à inclusão de redundância no interior dos CI e a necessidade de se procurar um ponto de equilíbrio. Nas FPGA, contudo, a fracção de recursos internos que geralmente fica subaproveitada, pode ser utilizada para aumentar a confiabilidade do circuito integrado, sem custos adicionais para além dos inerentes ao projecto adequado. A

replicação pode no entanto ser problemática nestes componentes, devido à difícil separação interna de sinais, que frequentemente implica utilizar pinos replicados, sob pena de se degradar a respectiva confiabilidade [Ber96, Mon98]. Resumidamente, a tolerância a falhas em circuitos integrados VLSI apresenta o balanço que indicamos na tabela 4-2.

Tabela 4-2. Razões a favor e limitações da confiabilidade nos VLSI

A favor	A fiabilidade de um VLSI é superior à do mesmo circuito implementado em lógica discreta. Custo das réplicas muito baixo. Redundância de fabrico começa a ser essencial nos VLSI e nos WSI, sendo reaproveitável. Aproveitar a lógica residual para melhorar a confiabilidade nas FPGA.
Limitações	Taxa de falhas cresce com o número componentes integrados. Número de pinos, principalmente nas FPGA. Acessibilidade e fiabilidade na transferência de sinais de E/S. A redundância aumenta o número de vectores de teste necessários. Defeitos de fabrico não detectados podem gerar <i>fmc</i>. Projectos multi-equipa potenciam problemas de testabilidade interna. A redundância implica maior potência dissipada, e temperatura, que afectam o MTTF. Temperatura de caixa elevada afecta a fiabilidade nos contactos dos pinos. Tensão de alimentação baixa, reduz as margens de ruído e afecta o MTBF [Lia96]. Baixa energia de comutação susceptível a ruído e radiação, degrada MTBF.

Por estas razões, mesmo sendo baixo o custo das réplicas, a replicação num só circuito integrado tem interesse limitado, devido à necessidade de se minimizar a sobrecarga de confiabilidade, o que nos leva a analisar as arquitecturas tradicionais. Destas, excluimos os casos particulares das soluções de tolerância a falhas que se baseiam em códigos correctores de erro (de aplicação condicionada e complexa), as baseadas em programação (que representam uma área distinta e se limitam aos sistemas com microprocessador) e os circuitos imunes a erro devido aos condicionamentos que a respectiva aplicação tem nos componentes VLSI [Nic98a]. Em consequência, centraremos a análise da secção seguinte nas arquitecturas mais utilizadas a nível físico, por replicação e com códigos detectores de erro.

4.1.4 Análise comparativa das arquitecturas Tolerantes a Falhas

As implementações mais recentes dos circuitos auto-verificáveis, vocacionadas para canais de dados (*data path*) e circuitos combinatórios modulares, permitem sobrecargas de confiabilidade entre 12 e 85%, dependendo da regularidade e tamanho das estruturas alvo [Dua97]. No entanto, o processo de codificação é oneroso, pode limitar a frequência de operação e, no estado de conhecimento actual, considera-se improvável que venha a ser aplicável de forma modular, particularmente em circuitos sequenciais [Jan85, Nic98c]. Note-se ainda, que a sobrecarga indicada não inclui o circuito para confinar o erro e fornecer uma palavra segura nas saídas (o que pode acrescentar mais 10 a 20%), bem como o aspecto relevante de um circuito auto-verificável permitir apenas parar o módulo, devendo ser *duplex* para tolerar falhas físicas.

Nas arquitecturas por replicação, atribuindo o índice 100 ao circuito de missão, sabemos que o comparador num módulo duplicado pode representar 10-20% e que o votador num TMR é equiparado a 50-100% do circuito de missão, por ser geralmente auto-verificável. A tabela 4-3 representa os encargos médios indicados na literatura mais recente, sendo de enfatizar que o encargo aumenta quando se baixa o tamanho dos blocos funcionais onde são tomadas decisões, atingindo valores máximos ao nível da porta lógica.

Tabela 4-3. Arquitecturas tolerantes a falhas e encargo correspondente

Método	Arquitectura	Encargo (%)
Codificação	Auto-verificável (*)	130-180
	Duplex - Auto-verificável	270-350
Replicação	Duplicação + comparação (*)	> 210
	TMR	350-700
	NMR	> 500
Reconfiguração	Hierárquica, Mista	> 350

As arquitecturas marcadas com asterisco (*) permitem somente detectar o erro, devendo acrescentar-se mais 10 a 20% para o confinar e fornecer uma palavra segura na saída. As restantes alternativas implicam circuitos integrados com três vezes ou mais o encargo do circuito de missão, o que é indesejável mesmo em componentes VLSI.

Quanto às arquitecturas replicadas, susceptíveis a *fmc* como vimos na secção anterior, há propostas de soluções alternativas para os componentes VLSI, mas que são complexas. Podemos referir como exemplo um TMR em que uma réplica é em lógica negativa [Lal85], de que não conhecemos aplicações práticas e que implica projectar dois circuitos de missão distintos (um complementar dos outros dois), mas com tempos de propagação iguais. As características dos circuitos VLSI permitem considerar também a integração de arquitecturas mistas e auto-reparáveis, mas o impacto na fiabilidade é sempre relevante [Mari91, Thi93].

A conclusão que retiramos é que corrigir o erro implica uma sobrecarga de confiabilidade tendencialmente dupla da sobrecarga de teste necessária à detecção, pelo que tentar conciliá-las num circuito integrado implica onerar excessivamente a detecção ou degradar a capacidade de correcção. Mais uma vez, em nossa opinião, a solução procurada deve conduzir a dois resultados independentes:

- Circuitos integrados que detectem as falhas internas, podendo recuperar das temporárias.
- Circuitos integrados que tolerem a primeira falha interna (permanente).

Podemos inferir ainda que a duplicação do circuito de missão é o mínimo necessário para tolerar falhas permanentes, mas encargos próximos de 300% não são economicamente aceitáveis e não conhecemos referências a circuitos integrados com arquitectura TMR, sendo os maiores encargos referenciados correspondentes a 240% do circuito de missão, em circuitos com duplicação total ou parcial para aplicações críticas [Roq98, Spa98].

Em qualquer das soluções apontadas, o objectivo primário é assegurar um circuito de missão fiável, o que podemos conseguir gerindo a testabilidade ao nível dos blocos funcionais e a tolerância a falhas ao nível de módulo, com a adição ao circuito de missão dos mecanismos convenientes. Quanto aos circuitos de missão, podem ter desde centenas até milhões de portas lógicas sendo impossível estabelecer parâmetros. No entanto, acima de 10^4 portas lógicas, a partição é virtualmente obrigatória [Bar87, Zor98] e há experiência suficiente para admitir que a generalidade dos circuitos combinatórios são testáveis, sem restrições significativas, em blocos funcionais com (até) 16 bits de entrada, valores usuais em muitos sistemas [Kuz96, Doo98, Zor98]. Quanto ao número de portas lógicas e à profundidade dos blocos funcionais, vimos no capítulo dois que há ferramentas que permitem inserir células BST, em consonância com a cobertura de faltas especificada e o limiar aceitável para os custos da geração e da aplicação do teste [Glo95, Her98].

4.1.5 Utilização da norma 1149.1 na Monitorização

A crescente utilização da infra-estrutura BST, apoiada em ferramentas de inserção automática [Abs96, JTA98] e bibliotecas dos fabricantes [Tex93, Xil99], justifica o objectivo de aproveitar esta infra-estrutura, que presentemente se mantém inactiva durante o funcionamento do circuito, para efectuar a monitorização em tempo-real e contribuir para a implementação de soluções tolerantes a falhas. No entanto, qualquer expansão da infra-estrutura BST deve originar uma sobrecarga limitada, para além de que propostas implicando o afastamento da norma IEEE 1149.1 têm suscitado alguma oposição [Cha97]. As referências bibliográficas que descrevem a utilização da infra-estrutura BST para melhorar a confiabilidade de funcionamento são escassas, sendo relevante destacar as seguintes, por ordem cronológica:

- A utilização de Paridade e Interrupções foi sugerida por [McH90] para aumentar a fiabilidade das operações de deslocamento, principalmente para as instruções, consideradas a fase mais crítica do processo BST, pela possibilidade de colocar as células em modo de controlabilidade e afectar o circuito de missão. As limitações desta técnica são a utilização de paridade fixa e a necessidade de uma linha adicional para o sinal de interrupção, que implica lógica associada relativamente complexa para ser inserida num controlador BST.
- Em [Wag91] é proposta uma forma de monitorização (concorrente), que permite aumentar até uma ordem de grandeza a probabilidade de detectar falhas através de um teste fora de serviço. O mecanismo sugerido para a monitorização é a infra-estrutura BST, mas o método proposto para melhorar a capacidade de detecção é intrusivo.
- A técnica B²UBIST para módulos auto-verificáveis, proposta em [Lub92], usa a infra-estrutura BST com o objectivo de refinar o nível de discriminação possível no método UBIST: testa alternadamente os blocos funcionais e os verificadores, conduzindo as saídas destes até ao exterior. Na medida em que melhora a resolução a nível de blocos funcionais, esta solução permite reduzir as exigências de codificação, mas necessita de três sinais de controlo adicionais, não sendo compatível com a norma 1149.1. Note-se contudo que as soluções baseadas no processo UBIST devem ser *duplex* para tolerarem falhas permanentes e, nestas condições, o encargo aproxima-se do de um TMR.
- Em [DeH95] é proposta a redundância a nível de TAP para reconfigurar e endereçar individualmente diversos canais BST, no sentido de melhorar a fiabilidade desta infra-estrutura em arquitecturas configuráveis, mas a sobrecarga não é analisada e parece-nos excessiva para ser praticável.

- A injeção de faltas proposta em [Cha94] permite o controlo de faltas *sempre-a* nos pinos, mas não prevê qualquer mecanismo de sincronização em tempo-real com o circuito de missão e nem sempre é possível suspender o processamento para efectuar o varrimento, em particular ao lidar com sistemas dinâmicos e falhas temporárias.
- Módulos com circuitos de missão duplicados [Cha97], alternadamente desligados e testados em regime calendarizado, necessitam de utilizar células modificadas (com encargo duplo das células normais), para isolar o circuito e injectar os sinais de teste. Além de ter problemas na sincronização dos circuitos ao reintegrá-los, esta proposta colide com a norma 1149.1.
- Em [Kun97] é descrita uma aplicação ferroviária que aceita tempos de latência até 2 segundos, em que são utilizados dois ASICs com vários pontos de teste acedidos e comparados continuamente através da infra-estrutura BST. Esta solução é susceptível a *fmc*, se não for prevista diversidade de projecto.
- O processo de teste calendarizado descrito em [Kie98] utiliza a infra-estrutura BST para controlar operações de auto-teste (*scan-BIST*, [Hel95]), sendo de destacar o teste de um circuito de missão decomposto em BFs definidos aleatoriamente (sem recorrer à informação estrutural), com base no critério de se utilizarem registos BST com comprimento não superior a 50 células, um valor que permite acelerar as características dinâmicas deste teste face às soluções de registo único.

Em todos estes exemplos, a quantidade de informação que a infra-estrutura BST pode fornecer é a limitada pela sua natureza série, o que sendo pouco relevante num teste calendarizado, condiciona a monitorização concorrente. Podemos facilmente aperceber-nos da natureza desta limitação, se considerarmos um circuito de missão genérico com n entradas, m elementos de memorização e s saídas, que funciona a uma frequência $f=1/\tau$. Supondo a comparação dos vectores de teste nas entradas e a leitura de resultados nas saídas, em cada intervalo τ a infra-estrutura BST deve de inspeccionar $(n+m)$ pontos de teste. Uma vez que em circuitos integrados reais o máximo exigível é que a frequência de TCK possa igualar a do circuito de missão, no intervalo τ a infra-estrutura BST poderá suprir um bit apenas e a relação entre a quantidade de informação necessária e a fornecida via BST é, no melhor dos casos, $n+m$. Esta limitação só poderia ser compensada aumentando TCK, mas nenhum fabricante estaria disposto a suportar o custo de um circuito integrado 30, 50 ou 200 vezes mais rápido que o necessário na aplicação, admitindo ser possível fabricá-lo.

- **Sobrecarga aceitável da infra-estrutura BST**

A sobrecarga de teste em silício, aceitável para a inserção da infra-estrutura BST com o objectivo de facilitar o teste de produção situa-se entre 4 e 5% [Doo98]. Esta sobrecarga deve-se essencialmente ao controlador do TAP, uma vez que as células do registo periférico podem ser inseridas entre pistas de alimentação e implicam pouco silício extra [Abre97, Yan98]. Desta forma a sobrecarga lógica pode atingir 6-8%, um valor relativamente comum [Cro89, Doo98]. A utilização de registos de utilizador, que representa uma sobrecarga adicional, permite complementar a cobertura de faltas possível com o registo periférico até ao valor especificado [Gun91], ou acelerar o acesso a determinados pontos de teste, colocando as células correspondentes em registos mais curtos [Lie91, Nar93]. Este tipo de optimização pode ser automatizada por algoritmos como o *DYNASTEE*, que recorre ao conceito de *teste dinâmico* para optimizar a testabilidade do circuito de missão, procurando minimizar o custo de geração e aplicação do teste, mesmo para circuitos sequenciais [Glo93], ou o *SCORCH*, que minimiza os registos e procura gerar o menor número de vectores de teste, para a cobertura de faltas especificada pelo utilizador [Pra91, Glo95].

Tendo em conta o valor acrescido das funções adicionais, é lícito admitir na tolerância a falhas uma sobrecarga de confiabilidade superior à indicada acima, mas não há por enquanto referências que suportem esta hipótese. Por comparação com as alternativas, sempre superiores a 50% no CI, consideramos aceitável como ponto de partida, uma sobrecarga de teste no CI até 20% do circuito de missão, para monitorização e detecção de erro. Os restantes 30% serão atribuídos ao controlador da infra-estrutura BST residente (*BSμC*), necessário no modo concorrente e semelhante aos descritos em [Fer92a, Pet97, Mont98], ou a versão simplificada referida no capítulo oito deste documento, podendo ainda acumular funções com a norma IEEE 1149.5. Por outro lado, as recomendações da FTCS'22 mantêm-se actuais (*vide* secção 4.1.1), mas no estágio de conhecimento presente é impossível prever se algum dia serão viáveis técnicas de projecto para a testabilidade e confiabilidade *modulares*, que conduzam a soluções *optimizadas*, dois conceitos frequentemente antagónicos. Outros aspectos a ponderar na sobrecarga aceitável, são o recurso à diversidade de projecto como forma de detectar *fmc* nos componentes VLSI e a definição de estratégias de teste em campo, uma área difícil para o fabricante e desagradável para os técnicos de manutenção, por expor os pontos fracos do sistema ao utilizador. Por estas razões, qualquer ferramenta residente de localização das falhas é importante, em particular se permitir comunicar com um equipamento de manutenção.

4.1.6 *Características do teste concorrente e dos vectores de teste*

O teste calendarizado afecta necessariamente o desempenho do circuito de missão, restringe-se a falhas permanentes e, para além disso, a utilização da infra-estrutura BST neste regime foi já apontada como podendo gerar uma actividade excessiva em circuitos grandes, indutora de erros e sendo mesmo potencialmente destrutiva [Her98]. O teste concorrente, por seu lado, permite a monitorização funcional, que ultrapassa os problemas anteriores, mas é mais difícil de implementar, particularmente se não-intrusivo. Analisando as opções associadas às células BST, encontramos duas alternativas com interesse para a utilização neste domínio:

- *Células de votação*: semelhantes ao votador de um TMR. No entanto, e como as arquitecturas com redundância de ordem três ou superior são de excluir em componentes VLSI, abandonaremos esta opção.
- *Células de detecção*: neste caso, as soluções conhecidas para o teste funcional (lógico) concorrente, baseiam-se em:
 - Codificar e verificar as saídas, mas redesenhar o circuito para inserir codificação contraria os objectivos de optimização e independência, pelo que pretendemos evitá-la.
 - Análise da relação Entradas/Saídas, que pode ser verificada por comparação com:
 - Réplica: permite detecção imediata e dispensa um modelo de faltas, mas implica um encargo superior a 200% e como tal é indesejável.
 - Vectores de teste armazenados em memória: análogo no essencial ao teste de produção, este processo é compatível com a infra-estrutura BST.

Este último caso é o que se revela mais interessante, mas há dois procedimentos que não podem ser controlados em modo concorrente:

- *A sequência de aplicação dos vectores de teste*: uma vez que não é possível controlar os vectores de entrada, fica vedada a gestão (num intervalo em tempo restrito, útil) da sequência em que são encontrados os vectores de teste, impedindo a utilização de técnicas dependentes da ordem destes, como a compactação da resposta por assinatura. Se não forem impostas restrições temporais, então estas técnicas podem ser aceitáveis, mas ficam ainda assim condicionadas pela possível não ocorrência de vectores esperados [Sha88].
- *A aplicação de todos os vectores de teste do modelo*: sendo impossível garantir que todos os vectores de teste aparecem num intervalo aceitável, o teste deve poder terminar

mesmo que não assegure a máxima cobertura de faltas por eles facultada. Esta limitação inviabiliza os métodos de compactação, síndrome incluído, pelas razões apontadas acima, e pode impedir a conclusão do teste [Sax92], razão pela qual não exploraremos a utilização destes métodos.

Concluimos assim que a solução mais adequada às características da infra-estrutura BST, para teste concorrente, é a amostragem. O intervalo de latência associado a este tipo de processos pode ser estimado em função do número de vectores, e fá-lo-emos no capítulo seguinte, mas para já procuraremos relacionar esse número com a complexidade do circuito a monitorar. No entanto, não encontramos referências a estimadores precisos para o número de vectores de teste, qualquer que seja a medida de complexidade utilizada [Agr78, Gol79], e os indicadores quantitativos descritos na literatura podem ser sintetizados no *Índice de Testabilidade*. Determinou-se experimentalmente que este índice, definido pelo logaritmo da soma dos índices de observabilidade e controlabilidade dos nós do circuito de missão, é proporcional ao número máximo de vectores necessários para o testar, mas tem uma representatividade limitada, apesar do apreciável custo computacional que requer [Sav83, Set85a].

Na impossibilidade de encontrar estimadores analíticos, e como pretendemos apenas um *majorante*, procuraremos a relação entre o número de vectores para um teste exaustivo e para um teste de detecção compacto, a partir de valores referidos por alguns autores. Com o objectivo de nos auxiliar na análise deste problema, começaremos por representar na figura 4-1 a relação entre os grupos de vectores considerados.

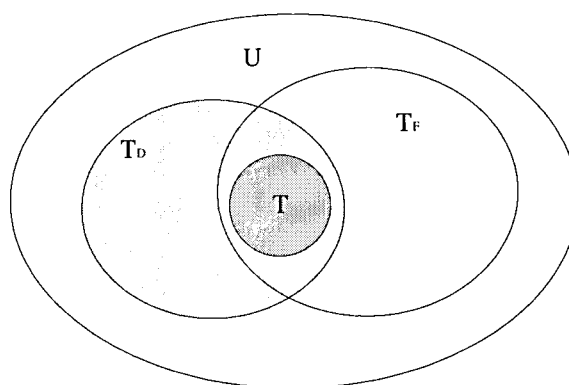


Figura 4-1. Universo de Vectores

Nesta figura, U representa o universo de vectores de entrada, T_D os vectores para um teste de Diagnóstico, T_F corresponde aos vectores para um teste funcional e T o grupo de vectores para o

teste de detecção de faltas (do modelo considerado). Vimos no capítulo dois que, num teste exaustivo, T estará contido em T_D . Por outro lado, como admitimos uma distribuição de entrada equiprovável, todos os vectores de teste devem aparecer em funcionamento normal. Esta condição é aceitável porque, em sistemas não codificados, $T_F \approx U$, caso contrário o sistema é redundante ou foi sobredimensionado (podendo ser simplificado).

Definamos agora o parâmetro $\sigma = \#T/\#U$, correspondente à razão entre o número de vectores do teste para detecção de erro e os do universo de entrada, e procuremos esta relação na literatura, começando por recapitular que no capítulo dois apresentámos tabelas contendo os números de vectores de teste obtidos para diversos circuitos, sem que os autores referidos os apresentem como testes mínimos. Mesmo assim, o número de vectores de teste é relativamente pequeno, apesar da complexidade (em número de portas lógicas) de alguns dos circuitos estar próxima daquela em que se justifica a partição [Kuz96].

As tabelas 4-4 e 4-5, que construímos para circuitos ISCAS'85 [Brg85] e ISCAS'89 [Brg89] respectivamente, com base nos dados referidos em [Pom97] e complementados em [Ham98a], tornam evidente que o número de vectores de teste para a detecção de erro (T), obtidos com os algoritmos actuais referidos no capítulo dois, se restringe geralmente a algumas dezenas, valores compatíveis com o processo de monitorização e com o armazenamento em memória.

Tabela 4-4. Número mínimo conhecido de vectores de teste para circuitos ISCAS'85

ISCAS'85					
Circuito	#VT	#EP	#Portas	#SP	σ
c17	4	5	6	2	0,15625
c432	27	36	160	7	5,24E-10
c499	52	41	202	32	1,86E-11
c880	16	60	383	26	5,20E-17
c1355	84	41	546	32	1,86E-11
c1908	106	33	880	25	3,84E-09
c2670	44	233	1269	140	1,69E-68
c3540	84	50	1669	22	4,44E-14
c5315	37	178	2207	123	4,65E-52
c6288	6	32	2416	32	7,45E-09
c7552	65	207	3513	108	1,01E-60

Na tabela 4-5 temos os resultados para os circuitos sequenciais ISCAS'89 com varrimento total (versão *full-scan* [Add93, Ham98a]), onde merece destaque o pequeno número de vectores de detecção, face ao número pontos de teste a observar ($\#PPI = \#EP + \#FF$ [Kie98]).

Tabela 4-5. Número mínimo conhecido de vectores de teste para circuitos ISCAS'89

ISCAS'89						
Circuito	#VT	#EP	#FF	#Portas	#SP	σ
s208	27	10	8	94	1	1,03E-04
s298	23	3	14	75	6	1,75E-04
s344	13	9	15	160	11	7,75E-07
s349	13	9	15	161	11	7,75E-07
s382	25	3	21	158	6	1,49E-06
s386	63	7	6	159	7	7,69E-03
s400	24	3	21	164	6	1,43E-06
s420	44	18	16	218	1	2,56E-09
s444	24	3	21	181	6	1,43E-06
s510	54	19	6	201	7	1,61E-06
s526	49	3	21	193	6	2,92E-06
s526n	49	35	21	194	6	6,80E-16
s641	21	35	19	379	24	1,17E-15
s713	21	35	19	393	23	1,17E-15
s820	94	18	5	289	19	1,12E-05
s832	94	18	5	287	19	1,12E-05
s838	76	34	32	446	1	1,03E-18
s953	76	16	29	395	23	2,16E-12
s1196	113	14	18	529	14	2,63E-08
s1238	121	14	18	508	14	2,82E-08
s1423	20	17	74	657	5	8,08E-27
s1488	101	8	6	653	19	6,16E-03
s1494	100	8	6	647	19	6,10E-03
s5378	97	35	179	2779	49	3,68E-63
s9234	105	36	211	5597	39	4,64E-73
s13207	233	62	638	7951	152	4,43E-209
s15850	95	77	534	10772	150	1,12E-182
s35932	12	35	1728	16065	320	0,00E+00
s38417	70	28	1636	22179	106	0,00E+00
s38584	111	38	1426	19253	304	0,00E+00

Embora não seja possível definir regras claras, devido aos múltiplos factores envolvidos (número de portas lógicas, profundidade média e máxima, número de pontos de teste), é de esperar para σ um comportamento tendencial sempre decrescente, por influência da partição e do projecto para a testabilidade. Este tipo de comportamento pode ser exemplificado com um ASIC recente, destinado a uma aplicação multimédia, com mais de dois milhões de portas lógicas, cerca de metade utilizadas em lógica combinatória e as restantes em blocos de memória num total superior a 40 Kbytes [Zor98]. Este circuito foi particionado em 12 blocos funcionais, permitindo desta forma que uma estimativa de 30.000 vectores de teste para a detecção de erro (no caso de não haver partição), se reduzisse a cerca de 4.000, um valor baixo para a complexidade indicada na tabela 4-6, onde σ foi calculado dividindo o número de vectores de T pelo de vectores potenciais de U (2^{100}), necessários num teste exaustivo.

Tabela 4-6. Características do ASIC referido em [Zor98]

# Portas lógicas	# EP	# BF	# Vectores de teste	σ
$>2 \cdot 10^6$	>100	$6+4+2 = 12$	≈ 4000	$3,16E-27$

Tomando como ponto de partida os valores de σ indicados nos exemplos apresentados e a análise efectuada ao longo deste capítulo, podemos agora estabelecer as linhas orientadoras do trabalho futuro.

4.2 Especificação

A tendência para integrar as funções de vários componentes discretos num único circuito integrado VLSI, dificulta a testabilidade interna, o que afecta um conjunto de regras da tolerância a falhas que nela se baseiam. A variedade das soluções possíveis actualmente, impede também, para já, uma perspectiva rigorosa do ponto de vista prático e realista, sem visar um problema concreto, como sugerem as recomendações da FTCS'22. Em consequência, pretendemos somente balizar a direcção que, em nosso entender, o projecto de CIs confiáveis deve seguir, face às condicionantes que descrevemos na secção anterior.

As alternativas para projectar um circuito integrado que tolere falhas permanentes estão limitadas pelas *fmc*, tornando desejáveis arquitecturas com detecção de erro, mas no estado de

conhecimento actual não há *uma* estratégia de teste geral e óptima: há várias, com maior ou menor relevância, conforme cada caso específico. Para além disso, os diversos tipos de falhas que afectam os componentes VLSI, e que podem ser influenciados pela aplicação, não permitem (ainda) respostas à questão da validade dos modelos de faltas tradicionais na detecção por amostragem, cada vez mais plausível, dada a complexidade e o elevado número de pinos nestes componentes. Por outro lado, o conceito tradicional de que a complexidade de teste cresce de forma quase exponencial com o número de portas lógicas e de entradas é verdade apenas parcialmente, uma vez que podemos minimizar o número de vectores de teste, por recurso a duas medidas principais:

- *Partição dos circuitos*: melhora a testabilidade e reduz o custo de geração do teste, baixa o número de vectores e o tempo de aplicação do teste, aumentando por outro lado a cobertura de faltas.
- *Independência entre o circuito de missão e os mecanismos auxiliares*: nas soluções tradicionais de tolerância a falhas, os mecanismos auxiliares (codificação ou replicação) tendem a afectar a testabilidade conjunta e originam geralmente um maior número de vectores de teste [Nic88, Bar92]. A utilização da infra-estrutura BST como mecanismo auxiliar é, por si só, garantia de separação sem aumentar o número de vectores de teste.

Sintetizando os diversos pontos realçados neste capítulo, pretendemos critérios globais que evitem onerar desnecessariamente o sistema, pelo que a solução procurada deve ser dividida em duas, com referimos na secção 4.1.4:

- Uma solução de baixo custo, destinada a aplicações de confiabilidade restrita, baseada em componentes que detectam falhas internas e que podem (se a aplicação o permitir) recuperar de falhas temporárias.
- Outra solução destinada a aplicações críticas, que deverá permitir fabricar componentes que tolerem, pelo menos, a primeira falha física interna.

No primeiro caso pretendemos um teste concorrente, por amostragem dos vectores de teste, baseado na infra-estrutura BST. Em relação ao segundo caso, concluímos que a duplicação do circuito de missão é a solução mais adequada e economicamente justificável num circuito integrado que se pretende dever tolerar falhas permanentes. Neste caso, para além da duplicação, consideramos admissível uma sobrecarga de confiabilidade dupla da sobrecarga de teste que indicamos como aceitável no ponto 4.1.5, um valor que conduz a encargos no CI não superiores 250% do circuito de missão, inferior a qualquer das alternativas existentes (*vide* tabela 4-3).

Cada um destes casos será tratado separadamente nos próximos dois capítulos, tomando como referência os parâmetros indicados na tabela 4-7.

Tabela 4-7. Parâmetros de referência a considerar no projecto

Parâmetro	Valor de referência
Intervalo de latência	< 50 ms
Sobrecarga de testabilidade no CI (monitorização concorrente)	< 20% (do circuito de missão)
Sobrecarga de confiabilidade no CI (tolerância a falhas)	< 150% (do circuito de missão)
Número de pontos de teste a observar (entradas) nos blocos funcionais	≤ 16
Número de vectores de teste por bloco funcional	< 250 vectores de teste

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

Capítulo 6

Capítulo 7

Capítulo 8

Bibliografia

Anexos

Infra-estrutura CST
para
Detecção Concorrente de Erro

Notas Prévias:

Neste capítulo apresentaremos uma evolução da infra-estrutura BST, otimizada para o teste em regime concorrente, em duas etapas tal como foi desenvolvida e mantendo o nome que lhe demos quando foi apresentada em publicações na língua inglesa: **CST** - *Concurrent Scan Test*.

Começaremos pelo modo de funcionamento inicialmente desenvolvido com base na utilização da infra-estrutura BST no teste fora de serviço, prosseguindo depois com a optimização da solução inicial, assente em conceitos de funcionamento concorrente.

As figuras que ilustram a exposição, privilegiam a inteligibilidade e a compreensão dos princípios e modo de funcionamento, sendo os detalhes de implementação apresentados no anexo A.

5. Infra-estrutura CST para Detecção Concorrente de Erro

A infra-estrutura CST surgiu como evolução do BST, ao procurarmos adequar a norma IEEE 1149.1 à detecção concorrente de erros, um processo que levanta três dificuldades inexistentes ou pouco relevantes no teste fora de serviço, por não serem críticas nesse contexto:

- Na monitorização não intrusiva, os vectores de teste possíveis são aqueles que surgem aplicados às entradas do circuito de missão, durante o funcionamento. Por esta razão não existe, ou é muito condicionada, a liberdade para escolher os estímulos a aplicar nas entradas.
- As operações em que se acede ao registo de instruções são a fase mais crítica no controlo da infra-estrutura BST porque, em caso de erro no deslocamento, a instrução pode impedir o funcionamento transparente das células (de controlabilidade) e afectar o circuito de missão.
- Durante o funcionamento em modo concorrente, a infra-estrutura BST e o circuito de missão podem induzir mutuamente ruído cruzado⁷. Desconhecemos, no entanto, referências na literatura a este respeito, não tendo sido indicados problemas deste tipo no B²UBIST [Lub94], no *scan-BIST* [Kie98], na utilização do BST para monitorização térmica [Sze98], nem mesmo em aplicações comerciais [Boh97, Kun97, Zor98]. Podemos assim admitir que este problema, quando ocorre, pode ser solucionado [Nar93, Zor90].

O princípio de funcionamento da infra-estrutura CST consiste em aproveitar os vectores de teste, gerados para o teste de produção, para efectuar a monitorização em modo concorrente. Este tipo de operação necessita no entanto de sincronismo com o circuito interno, que a norma 1149.1 não prevê [San97a]. Será esta problemática que trataremos neste capítulo, primeiro para circuitos combinatórios e depois quanto à sua aplicabilidade aos circuitos sequenciais.

⁷ - Este tipo de ruído pode resultar de interferências entre ligações ou de ruído veiculado pela alimentação, em resultado da maior actividade que a utilização da infra-estrutura acarreta.



5.1 Definição da infra-estrutura CST

A estrutura hierárquica dos sistemas que consideramos neste documento corresponde à representada na figura 5-1, onde o *sistema* é constituído por *módulos* que, de acordo com os requisitos de confiabilidade da especificação, agrupam múltiplos *circuitos de missão* (CM), cada um destes podendo ainda ser tratado como um conjunto de *blocos funcionais* (BF).

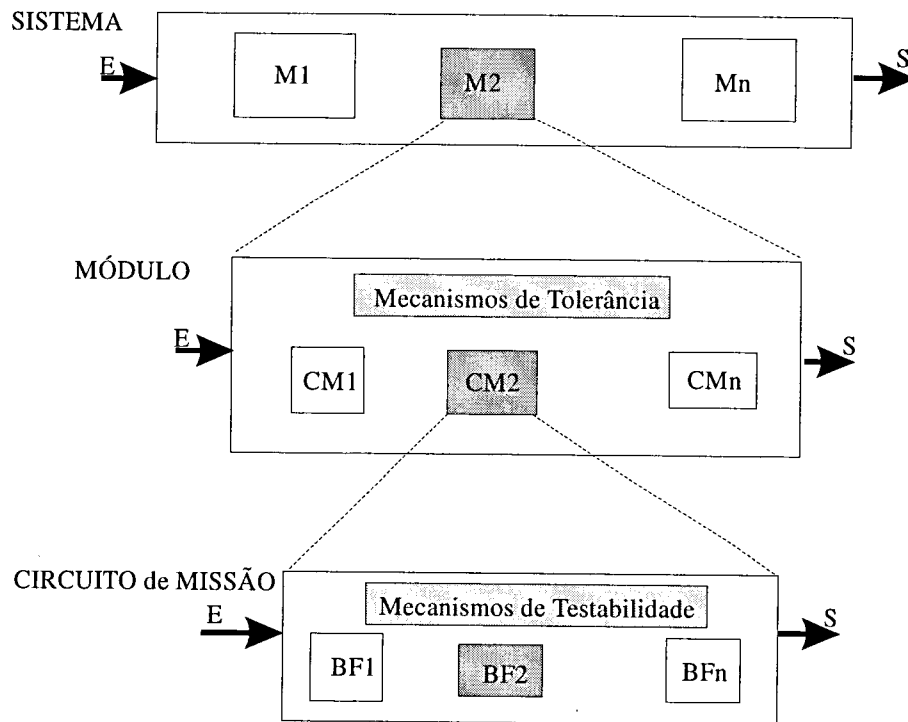


Figura 5-1. Hierarquia de projecto

Recapitulando o que vimos no capítulo dois sobre o circuito de missão e os blocos funcionais, recordemos que a sua definição não introduz custos adicionais na monitorização, uma vez que a divisão em blocos funcionais é ditada por razões de testabilidade, e deverá ocorrer apenas se o valor da cobertura de faltas pretendida não for possível apenas através do registo periférico [Gun91]. Este é, contudo, um problema que virá resolvido já da fase de produção, em consonância com a cobertura de faltas justificada pelos requisitos de confiabilidade.

Assim, este capítulo incide fundamentalmente nos “*Mecanismos de Testabilidade*” a nível do circuito de missão e na análise do respectivo funcionamento em regime concorrente, enquanto os “*Mecanismos de Tolerância*” a nível de módulo serão o tema a tratar no capítulo seis. Notemos, contudo, que na base de ambos os “*Mecanismos*” se encontra a infra-estrutura BST.

5.1.1 A infra-estrutura CST ao nível do Módulo

A nível do módulo, a infra-estrutura CST será gerida por um controlador residente (*BSμC*), como referido nos capítulos dois e quatro, que deve ser considerado um dos custos inerentes à confiabilidade. Este *BSμC*, tem como funções essenciais deslocar continuamente os vectores de teste (programados na memória ROM interna) para os circuitos destinatários e comparar as respostas recolhidas.

No teste fora de serviço a utilização da infra-estrutura BST recorre normalmente a uma só cadeia com todos os CI em série, que minimiza o número de pinos e evita circuitos de selecção. No entanto, esta solução implica a utilização de vectores com elevado número de bits, difíceis de armazenar e que podem degradar a taxa de erros de deslocamento. Estas situações, que importa naturalmente evitar na detecção concorrente de erro, podem ser minimizadas, se endereçarmos separadamente cada circuito de missão, de acordo com a figura 5-2.

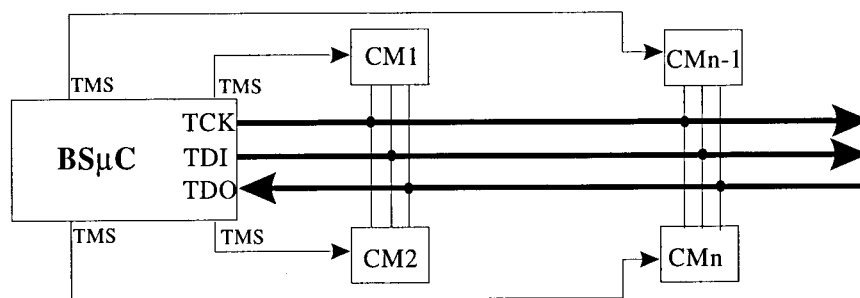


Figura 5-2. Acesso independente aos CM do Módulo

Esta organização está prevista na norma 1149.1, com linhas de TCK, TDI e TDO comuns, e somente linhas de TMS independentes. Notemos também que as saídas TDO ficam em alta impedância quando inactivas, permitindo a ligação em paralelo, e que uma disposição conveniente do controlador e restantes circuitos pode aumentar a fiabilidade global da estrutura, em comparação com o encadeamento série [Tex95b], uma vez que esta organização permite cadeias de varrimento mais curtas e com melhor MTBF.

A configuração apresentada não é obrigatória na nossa proposta, mas permite um funcionamento mais rápido e eficiente da infra-estrutura CST, uma vez que evita a utilização de longas cadeias de bits, bem como o recurso à instrução BYPASS. Finalmente, devemos referir que os circuitos de missão podem corresponder a componentes distintos, ou eventualmente a réplicas, numa aplicação tolerante a falhas.

5.1.2 A infra-estrutura CST ao nível do Circuito de Missão

Cada circuito de missão tem associado um TAP 1149.1, o respectivo controlador e o registo periférico, podendo ainda incluir registos de utilizador, como se ilustra na figura 5-3, que circundam os blocos funcionais necessários para atingir os objectivos de testabilidade especificados.

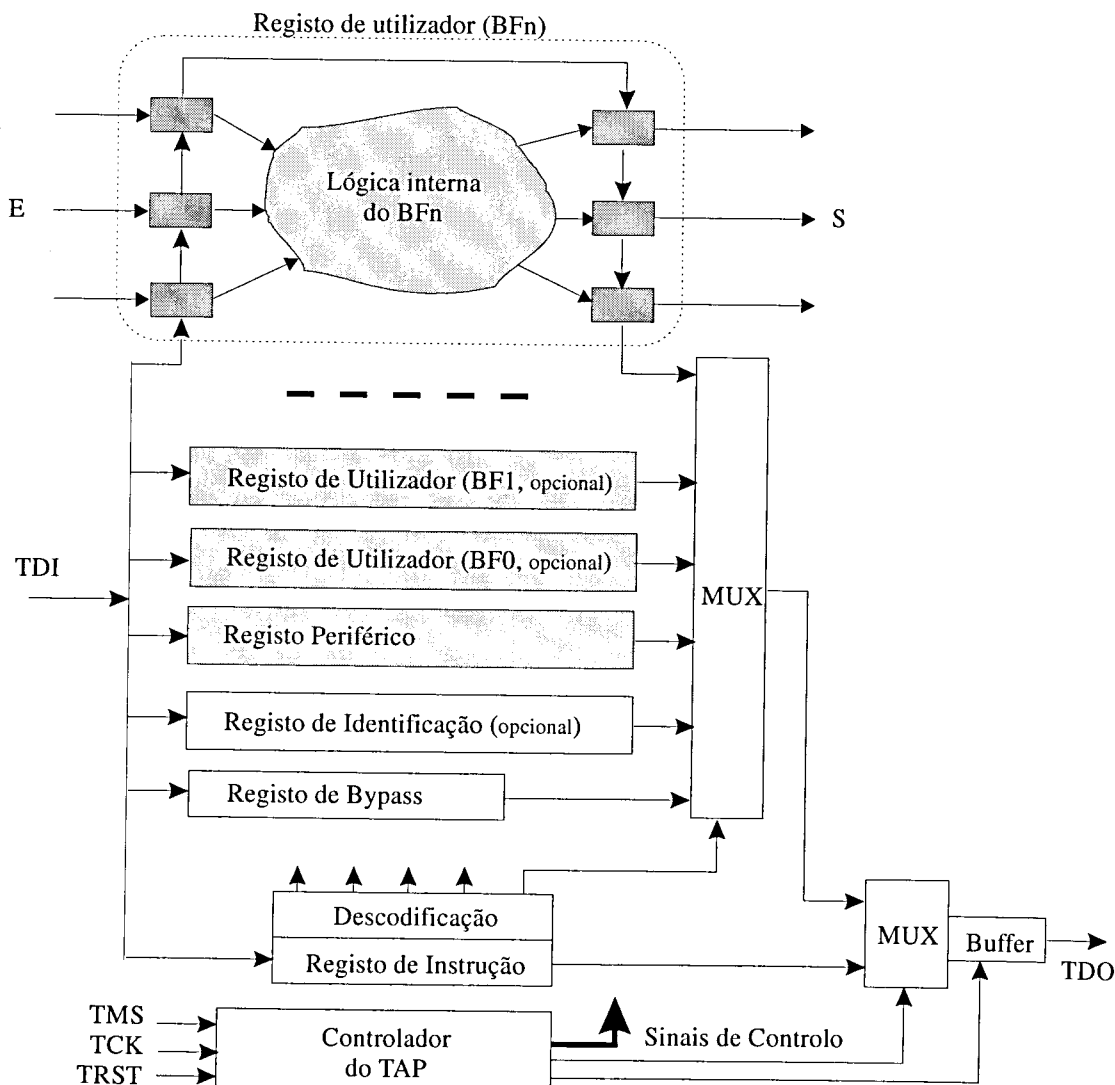


Figura 5-3. Infra-estrutura BST no circuito de missão

A nossa proposta abrange o registo periférico e os de utilizador associados a blocos funcionais, que no seguimento deste capítulo denominaremos apenas por *registos*. Todos os outros registos serão explicitamente identificados, sempre que a eles façamos referência.

Nesta abordagem, consideramos inicialmente que, em regime concorrente, cada registo permite monitorar a lógica que circunda (registo periférico - circuito de missão, registos de utilizador - blocos funcionais), não sendo consideradas para já situações de blocos funcionais circundados por mais do que um registo. Na secção 5.4.1, após a descrição dos princípios de funcionamento, analisaremos as relações deste tipo que são admissíveis.

Uma vez que os registos são seleccionados através do registo de instrução, definimos uma instrução opcional denominada

SELRn - Selecção do Registo n

que coloca entre TDI e TDO o registo de ordem n , com as células num modo de funcionamento idêntico ao que a instrução *Sample/Preload* define para o registo periférico. Quando $n=0$, a instrução SELRn seleccionará o registo periférico.

Para cada registo e circuito interno associado, temos os seguintes vectores de funcionamento e de teste (ou padrões de referência), como se ilustra na figura 5-4:

- V_i - Vector de entrada, em funcionamento.
- V_o - Vector de saída, em funcionamento.
- VT_i - Vector de teste, esperado na entrada em funcionamento correcto.
- VT_o - Vector de teste, esperado na saída em funcionamento correcto.

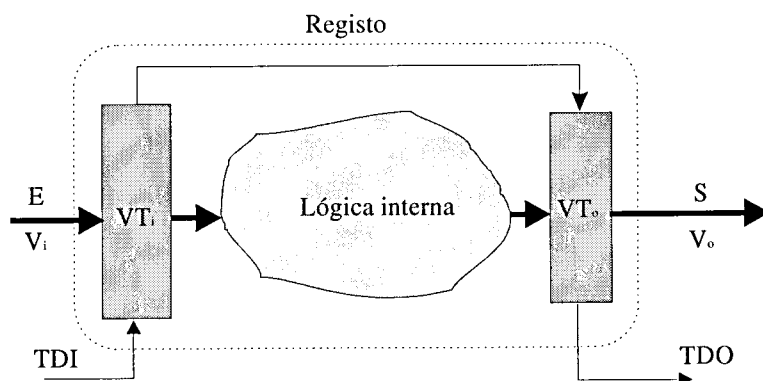


Figura 5-4. Relações entre os vectores de funcionamento e de teste

Os vectores de teste, previamente gerados e armazenados na memória do BSμC, consideram-se disponíveis para comparação com V_i e V_o , permitindo um processo de monitorização que pode ser efectuado de duas formas:

- Capturar os valores presentes nas entradas e saídas do circuito e deslocá-los para análise no BSμC.

- Carregar VT_i para o interior das células de entrada e esperar a ocorrência em funcionamento de um V_i igual.

A opção por uma destas soluções requer uma análise das vantagens e inconvenientes mútuos. No primeiro caso temos a captura por amostragem, aleatória, sob o controlo de TCK, que normalmente não está sincronizado com o circuito. No segundo, cria-se uma referência nas células de entrada e a igualdade entre V_i e VT_i define em que instantes devem ser analisadas as saídas e possíveis pontos de teste internos. A primeira alternativa não beneficia do sincronismo inerente à segunda, o que origina diferenças importantes na *Eficiência de captura* em modo concorrente, que definiremos como:

E_c - razão entre o número de pares de vectores detectados/capturados (V_i/V_o), que pertencem ao conjunto de vectores esperados (VT_i/VT_o), e o número total de capturas.

No caso da amostragem aleatória, assumindo uma distribuição equiprovável dos vectores capturados, E_c tende para a relação ($\#VT_i/\#V_i$). Como o número de vectores de teste deve ser minimizado, para otimizar o intervalo do teste concorrente e reduzir a memória necessária para os armazenar, a percentagem de capturas “úteis” será inferior a 1%, se atendermos aos valores de σ indicados no capítulo quatro. Pelas razões apontadas, esta opção é pouco eficaz e não será mais considerada.

No segundo processo, E_c depende somente da ocorrência dos VT_i em funcionamento e do intervalo de tempo em que estes vectores permanecem nas células. Uma vez que temos controlo sobre este intervalo, podemos ajustar (estatisticamente) aquele coeficiente, como veremos na secção 5.3. Os dois factores que esta solução faculta, auto-sincronismo com o circuito de missão na captura e melhor eficiência de captura, recomendam claramente a monitorização por busca de VT_i . No entanto, como este processo não pode ser executado sob o controlo de TCK, é necessário desenvolver um mecanismo de sincronização entre as células e o circuito de missão, que dará origem ao sinal de detecção **D** indicado na figura 5-5, activo em “0” (quando $V_i = VT_i$) e que descreveremos em pormenor na secção 5.2.

Definido o mecanismo a utilizar na entrada, passemos agora à verificação dos vectores na saída de cada bloco funcional, para os quais dispomos também de duas alternativas:

- Deslocar VT_o juntamente com VT_i .
- Captura de V_o nas células de saída, activada pela detecção $V_i = VT_i$.

Na primeira, a comparação ocorre simultaneamente na entrada e na saída do BF. No entanto, e sendo necessário deslocar todo o registo, esta opção torna-se mais lenta. No segundo caso, o V_o

capturado é lido pelo BS μ C e comparado com o VT_o esperado. Se organizarmos as células em dois grupos, o das entradas junto a TDI e o das saídas junto a TDO, o tempo de varrimento fica dependente apenas do número de células do maior dos dois vectores, como mostra a figura 5-5, onde se assume que VT_i é maior que V_o (ou VT_o).

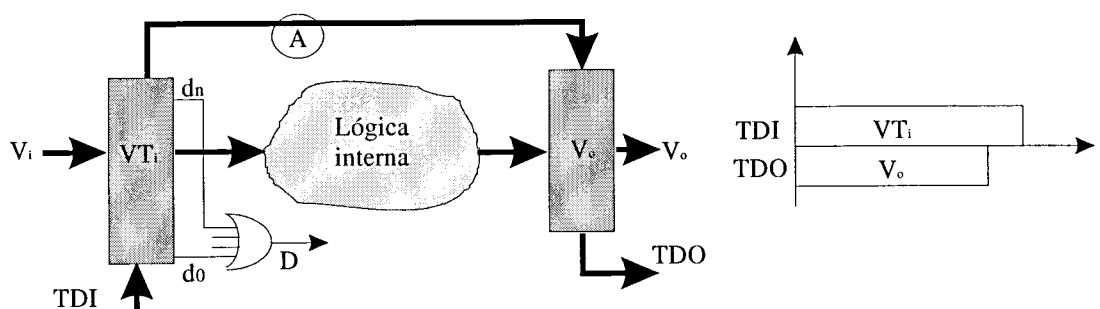


Figura 5-5. Grupos de células de E/S e deslocamento simultâneo de VT_i e V_o .

O agrupamento das células facilita a explanação mas não é obrigatório, e deve notar-se que mesmo nos casos onde a proximidade física não possa ser satisfeita, é sempre possível encadeá-las convenientemente, ao abrigo da secção 10.2.1.m da norma IEEE 1149.1a (1993). As células que forem necessárias só para controlabilidade devem ser colocadas na região A do registo, evitando-se deslocá-las durante a monitorização. Concluimos assim que a segunda opção é a mais vantajosa e a que será considerada no seguimento deste trabalho.

5.2 Princípio de funcionamento da infra-estrutura CST

Dado um circuito em funcionamento (não necessariamente correcto), pretendemos agora:

1. Deslocar um VT_i , armazenado na memória do BS μ C, para o grupo de células de entrada.
2. Detectar uma igualdade $V_i = VT_i$.
3. Capturar o V_o correspondente.
4. Deslocar um novo VT_i e recolher o V_o .
5. Comparar o V_o lido com o VT_o esperado, também armazenado na memória do BS μ C.

Os passos 1, 4 e 5 são necessariamente geridos pelo BS μ C, mas os passos 2 e 3 são executados localmente e de forma autónoma em cada registo da infra-estrutura CST, quando se encontra carregada a instrução SELRn. Podemos ver na figura 5-6 um diagrama genérico onde foi

representado um bloco adicional, denominado **CB** (*Capture Block*), que tem por função controlar a captura e memorizar a respectiva ocorrência. Este circuito recebe o sinal de detecção *D*, das células de entrada, bem como sinais provenientes do controlador do TAP, e gera o sinal que denominámos **C2C** (acrónimo em inglês de *Clock-to-Capture*), que substitui o sinal *Clock-DR* fornecido às células associadas aos sinais de saída.

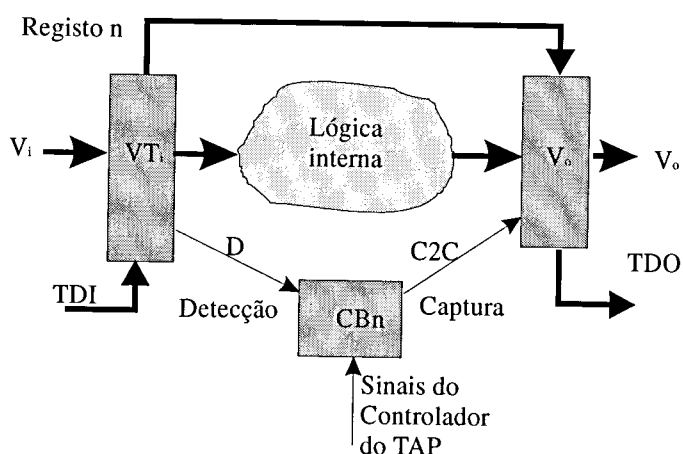


Figura 5-6. Sinais associados ao Bloco de Captura (CB)

Na análise que se segue, consideraremos que todos os acessos aos registos, sejam os de dados ou o de instrução, terminam no estado *Run-Test/Idle*, embora seja também possível passar directamente do estado *Update-DR* para *Select-DR*. A justificação para esta opção reside na necessidade de definir um estado de referência para os ciclos do controlador do TAP.

5.2.1 Caracterização do controlo de captura

Do ponto de vista da infra-estrutura CST, a sequência de operações correspondente ao princípio de monitorização enunciado no ponto anterior é a seguinte:

- Deslocar a instrução *SELRn*, para identificar o registo pretendido.
- Verificar se já ocorreu, neste registo, a captura da resposta ao VT_i anterior.
- Deslocar novo VT_i se a captura já ocorreu, ou passar ao registo seguinte.

O protocolo apresentado reflecte o facto de este procedimento de captura ser assíncrono em relação a *TCK*, o que implica que se verifique periodicamente se este evento já ocorreu. No entanto, se a verificação da ocorrência de uma igualdade ($V_i = VT_i$) implicasse deslocar para o exterior do conteúdo do registo, o V_o eventualmente capturado seria destruído, à passagem por *Capture-DR*, o mesmo sucedendo com o VT_i anteriormente carregado. Caso a captura não

tivesse ainda ocorrido, esta destruição de V_o e VT_i seria naturalmente indesejável e obrigaria a que se deslocasse de novo VT_i para o interior do registo.

Repare-se ainda que a captura assíncrona nas células do registo R_n deverá ser interdita quando $SEL R_n$ for a instrução actual e o controlador do TAP se encontrar num dos estados compreendidos entre Capture-DR e Update-DR (estes dois estados incluídos). Esta imposição decorre da necessidade de sincronização com TCK durante a permanência em qualquer destes estados, que estão ilustrados na figura 5-7. Fora destes estados, a captura é permitida até que o registo seja seleccionado de novo.

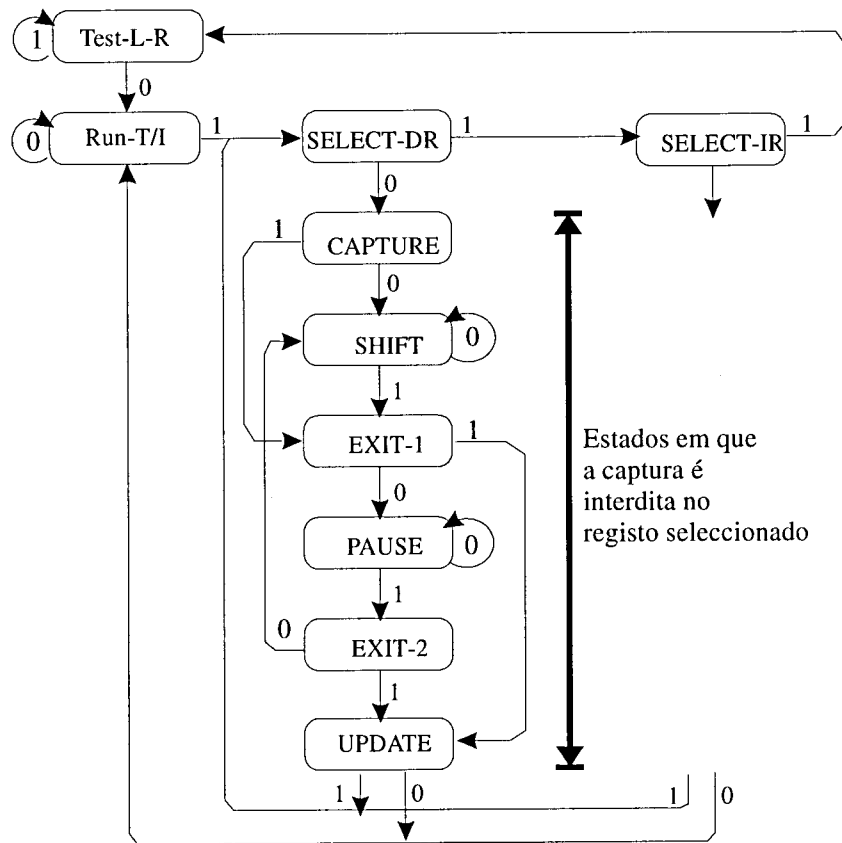


Figura 5-7. Diagrama que identifica os estados onde a captura é interdita

Em conformidade com este diagrama, o circuito CB será *activado* (consideramos activo em 0) à passagem pelo estado Update-DR, em que o novo VT_i é copiado do andar de deslocamento para o de retenção, nas células do registo seleccionado. Por outro lado, a *desactivação* de CB ocorre

em duas situações, indicadas no diagrama temporal que se apresenta na figura 5-8 (sugere-se também a consulta do diagrama temporal 10-14, no anexo A, como complemento):

- *Deslocamento de um novo VT_i* : CB deve ser desactivado quando se inicia o deslocamento de um novo VT_i , sendo a captura permitida até ao estado Capture-DR.
- *Com a detecção*: a igualdade $V_i = VT_i$, inibe também CB para que, no caso de ocorrer mais do que uma detecção entre dois acessos consecutivos do BS μ C ao registo, só a resposta à primeira igualdade seja capturada. Admitindo que haja mais do que uma, a última detecção seria mais actual, mas aumenta a probabilidade de colidir com novo acesso do BS μ C, que tem prioridade e invalida a captura.

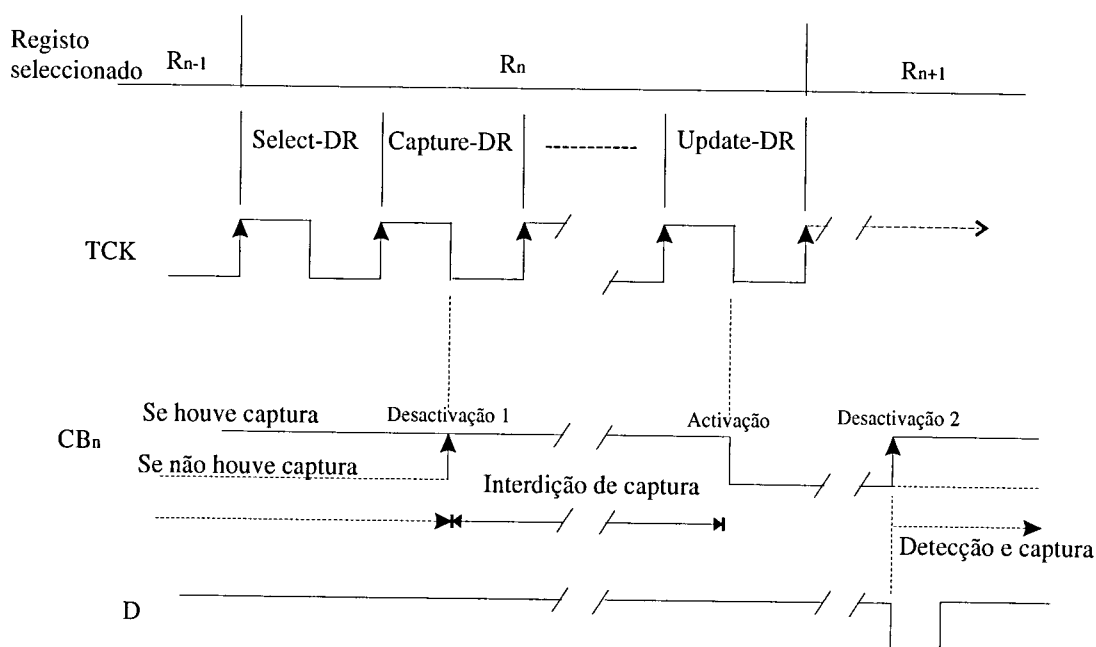


Figura 5-8. Situações em que ocorre a activação e desactivação de CB

Embora possa ser definida uma instrução que evite a captura na passagem pelo estado Capture-DR, impedindo a destruição de V_o , a sequência de estados “Shift-DR→Update-DR” afecta sempre VT_i , uma limitação que o BS μ C só pode resolver após completar e repetir o varrimento, seja para deslocar um novo VT_i ou para repor o anterior, um procedimento moroso e que cria um hiato temporal durante o qual não existe qualquer VT_i útil na entrada. Por outro lado, uma vez que a captura (assíncrona) tem de ser memorizada em CB e este circuito controla a captura, a referida instrução torna-se redundante. Resta, portanto, encontrar alternativas expeditas para determinar se a captura já ocorreu.

Uma solução possível é implementar um mecanismo de *interrupção* associado às capturas mas, para além de implicar pinos adicionais no circuito de missão e no BSμC, a estrutura necessária para gerar e atender interrupções aumenta a complexidade da infra-estrutura CST e do BSμC, sem vantagens apreciáveis na fiabilidade, pelo que não será considerada.

Uma alternativa mais simples consiste em aceder ao conteúdo de CB através do registo de instrução, aproveitando o facto de a norma apenas reservar a utilização dos dois bits menos significativos deste registo, como se mostra na figura 5-9, onde R1 e R2 representam respectivamente os andares de deslocamento e retenção das células deste registo.

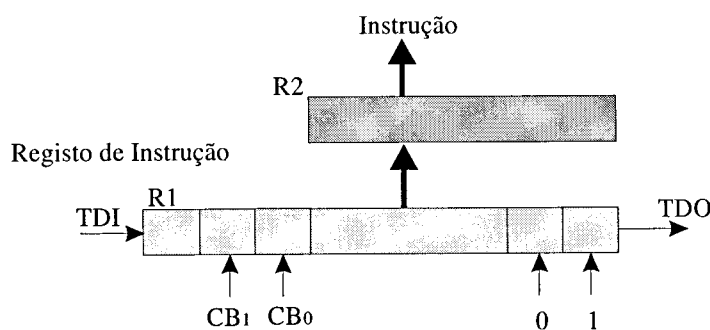


Figura 5-9. Leitura de CB através do registo de instrução

Desta forma, carregando o estado dos CB dos diversos registos em células dedicadas do registo de instrução⁸, permite-se ao BSμC saber se a captura está disponível logo após aceder ao registo de instrução, para deslocar SELR_n, sem afectar VT_i no registo a inspeccionar. Como se mostra no fluxograma da figura 5-10, o princípio de funcionamento fica assim reduzido a uma sequência de acessos ao registo de instrução e, sempre que há uma captura disponível, ao registo onde ela teve lugar, num processo rotativo e fixo.

Pode-se também admitir que, após deslocar a instrução SELR_n, o BSμC procure os bits CB com indicação de captura pronta, para ir de seguida ler cada um dos registos correspondentes. No entanto, abandonámos esta opção em virtude de privilegiar os registos onde a captura seja mais frequente, exigindo também um BSμC mais complexo (e com mais instruções) que o necessário numa análise sequencial dos registos. Para além disso, o programa do BSμC seria também mais complexo, e como tal menos fiável.

⁸ - Estas células dispensam o andar de retenção.

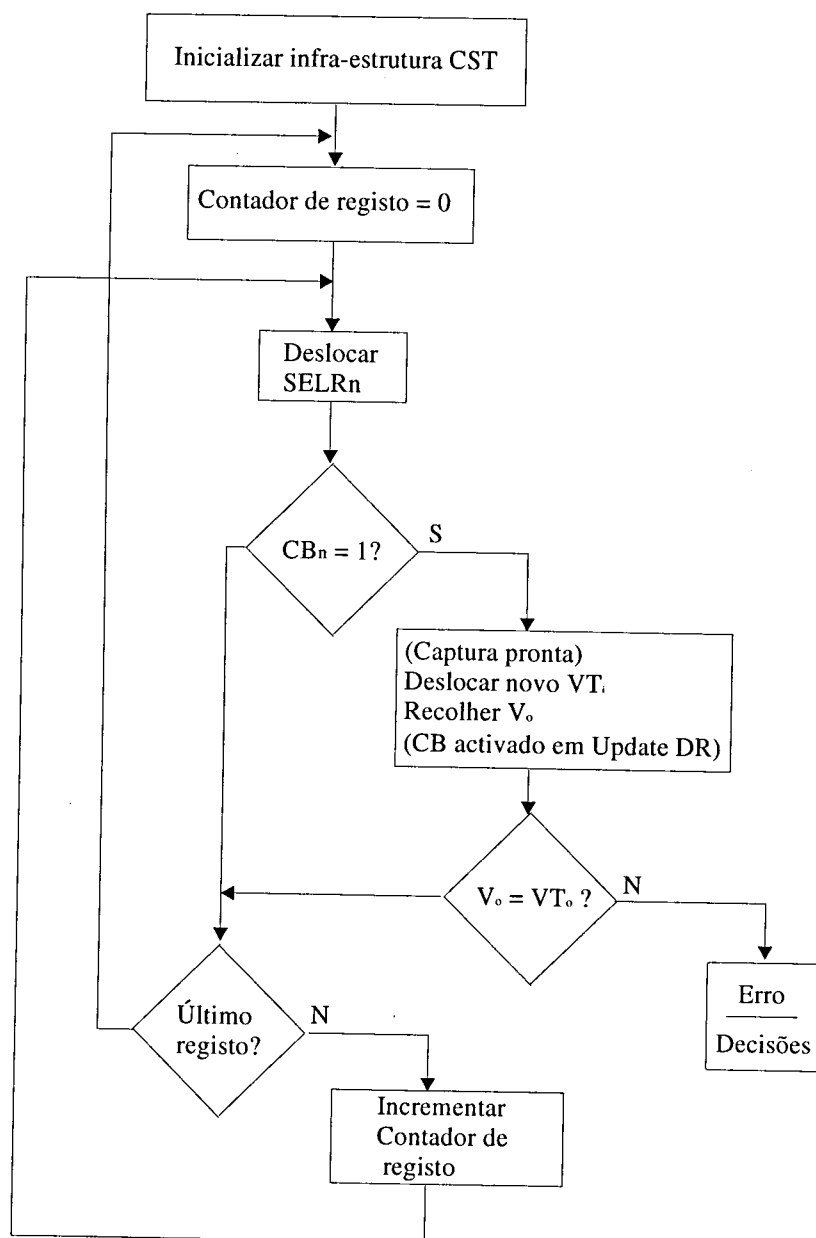


Figura 5-10. Fluxograma correspondente ao funcionamento da infra-estrutura CST

os sinais Shift-DR e Clock-DR, que definem o estado Capture-DR⁹, como se representa na figura 5-12.

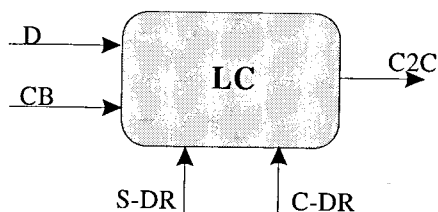


Figura 5-12. Sinais utilizados para controlar a captura

A síntese deste bloco pode ser feita por via analítica, tendo em conta que:

- Os sinais D e CB são activos em “0”.
- O sinal C2C deve reproduzir D quando o CB está activo.
- O sinal C2C deve também reproduzir Clock-DR, excepção feita para o impulso que corresponde ao estado Capture-DR, que pode ser inibido pelo sinal Shift-DR.

O circuito correspondente é descrito pela equação seguinte e está esquematizado na figura 5-13:

$$C2C = (D+CB) \cdot (C-DR+/S-DR)$$

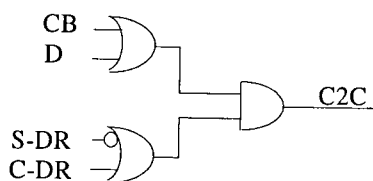


Figura 5-13. A lógica de controlo de captura

A saída deste circuito, o sinal C2C, para além de substituir Clock-DR nas células onde se pretende a captura, permite desactivar CB, em conformidade com o diagrama de estados da figura 5-7. Em resultado das condições de activação (por Update-DR) e desactivação (por C2C), e da sua topologia, CB constitui uma célula fantasma do respectivo registo, sem pino de acesso exterior e compatível com a secção 10.4.1.f da norma IEEE 1149.1a, alterada em conformidade com a figura 5-14.

⁹ - Capture-DR ocorre quando é aplicado um impulso em Clock-DR, sem que Shift-DR esteja activo.

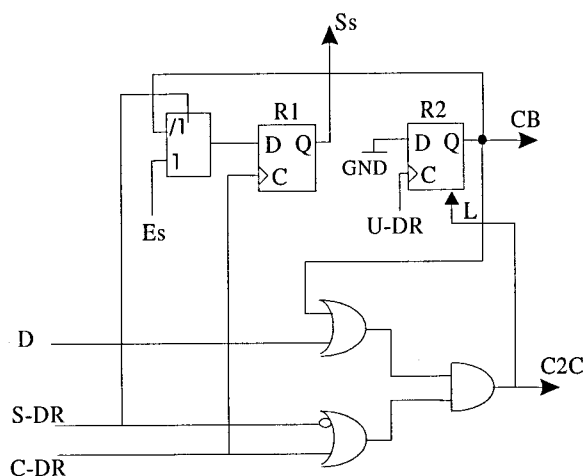


Figura 5-14. Constituição do CB

Neste circuito, além da lógica correspondente ao bloco LC, podemos ver que o andar de retenção é utilizado para memorizar o estado da captura, enquanto o andar de deslocamento mantém a sua função normal, permitindo deslocar para o exterior o conteúdo de R2.

O princípio de funcionamento desta célula (o encargo é semelhante e as alterações pequenas) concilia os dois FF internos com o diagrama de estados do controlador do TAP. Assim, após ser carregada a instrução SELRn e deslocado um VT_i , o andar de retenção (R2) é activado na passagem pelo estado Update-DR. A desactivação, efectuada pelo sinal C2C ligado à entrada assíncrona L de R2 (Limpar, ou *Preset* neste caso), ocorre em conformidade com as duas situações descritas na figura 5-8.

Finalmente note-se que, na passagem pelo estado Capture-DR, o andar de deslocamento (R1) da célula CB copia o andar de retenção (R2). De seguida, R2 é desactivado (se ainda não o foi pela detecção) para impedir uma captura extemporânea, durante os estados interditados no diagrama da figura 5-7. Desta forma torna-se possível deslocar para o exterior a informação que indica se o conteúdo das células de saída é válido (foi capturado) ou não. No entanto, nas condições de funcionamento que descrevemos, esta operação é desaconselhável por destruir VT_i , tendo conduzido à leitura de CB através do registo de instrução.

A figura 5-15 representa a constituição final de um registo na arquitectura CST, com o respectivo CB colocado junto das células do grupo de saída, para simplificar a ligação do sinal C2C.

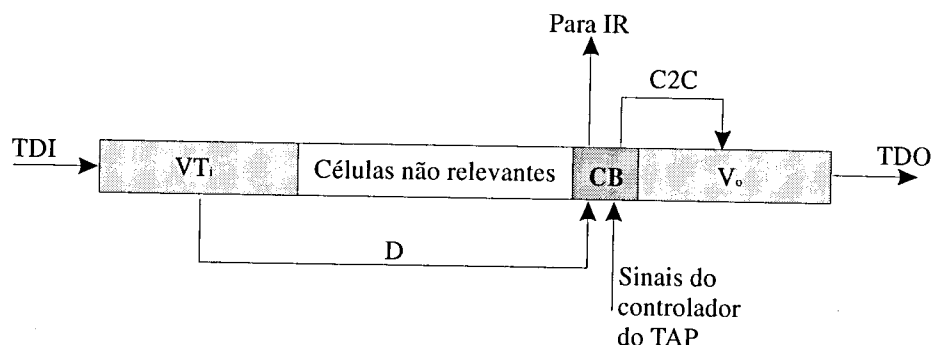


Figura 5-15. Funcionamento autónomo de um registo

A infra-estrutura CST difere da BST pela existência, em cada registo, da célula CB, bem como da lógica que gera o sinal D. Esta pequena alteração, cuja sobrecarga determinaremos na secção 5.4, permite-lhe contudo apresentar características de uma arquitectura distribuída pelos diversos registos, suportando a capacidade de cada um deles procurar os vectores de teste de forma autónoma, independente e síncrona com o bloco funcional respectivo, durante todo o tempo em que o registo não é acedido.

Refira-se ainda que CB foi definido como activo em “0” por razões de segurança, tendo em conta que TDI deve capturar um “1”, quando em circuito aberto. Em conformidade, fora do processo de monitorização, deve ser tratado como “1” e, em caso de erro no valor carregado, activa somente uma busca inútil.

Concluimos aqui o desenvolvimento da infra-estrutura CST, que consideramos possível com base nos conceitos de teste fora de serviço. Qualquer evolução adicional terá de se socorrer de conceitos inerentes ao comportamento dinâmico do modelo de funcionamento descrito para a infra-estrutura CST, o tema a desenvolver na próxima secção.

5.3 *Análise do comportamento dinâmico e optimização da infra-estrutura CST*

A infra-estrutura CST efectua uma monitorização discreta, em que a detecção de V_i e a captura de V_o são em tempo-real, mas a detecção de erros só é possível após a comparação de V_o com VT_o. O facto de este processo originar um intervalo de latência implica que não seja possível confinar erros, mas somente garantir uma probabilidade de os detectar, que é numericamente igual à cobertura de faltas, assegurada no intervalo em que os vectores de teste são verificados

[San97b]. Mesmo tendo em conta que esta cobertura de faltas depende das características dos estímulos de entrada e dos vectores verificados, podemos determinar a cobertura de faltas integrada ao longo desse intervalo, que denominámos *tempo de teste* (T_t), e cuja análise apresentaremos seguidamente.

5.3.1 *Tempo de teste* (T_t)

Em regime concorrente, o tempo de teste T_t do circuito de missão é o intervalo para verificar os vectores de teste (VT_i), desejavelmente todos, num processo em que a sequência dos V_i está condicionada pela aplicação. Assim, para estimar T_t a partir do princípio de funcionamento da infra-estrutura CST apresentado na secção 5.2, considere-se um módulo em relação ao qual são conhecidos os seguintes parâmetros:

- N = número total de registos a gerir pelo BSuC (independentemente do número de circuitos de missão).
- $n_i = \max.[VT_i, VT_o]$: número de bits a deslocar para cada registo R_i .
- n = média dos n_i .
- f = frequência de operação do circuito de missão, com período $\tau = 1/f$.
- ϕ = frequência de TCK, com período $\phi = 1/\phi$.

Finalmente, é também conhecido o intervalo de latência aceitável na aplicação, T_L . Para analisar o processo de monitorização temporal, comecemos por notar que há três componentes distintas:

- t_v - *tempo de varrimento*, correspondente ao acesso a cada registo, para cada vector de teste.
- T_v - *tempo de verificação*, correspondente ao somatório dos t_v de todos registos (N) que o BSuC supervisiona, entre dois acessos consecutivos ao mesmo registo.
- T_d - *tempo de detecção*, definido pelo intervalo médio para ocorrer a detecção de um qualquer VT_i . Note-se que este intervalo não está relacionado com a existência de falhas, dependendo apenas do circuito que gera os vectores de entrada (V_i).

Em regime normal, o BSuC desloca um VT_i para um determinado registo e vai procurar capturas nos restantes. Em condições ideais, deve haver uma captura pronta quando aceder de novo a este registo, o que corresponde a termos um tempo de detecção (T_d) mais rápido (curto) que o de verificação (T_v), como mostra a figura 5-16. Se esta condição não se verificar, a captura tende (em valor médio) a estar disponível só no acesso seguinte e degrada a eficiência de captura.

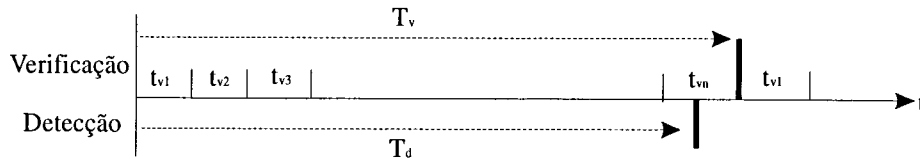


Figura 5-16. Relação entre os intervalos t_v , T_v e T_d

Assim, numa aplicação em que T_v corresponda em média a metade de T_d , a probabilidade de a captura estar pronta no primeiro acesso após ser deslocado um VT_i é de 50%, probabilidade que varia na relação directa da razão entre os dois intervalos, como se depreende da figura 5-16, pelo que, neste modo de funcionamento, definiremos a eficiência de captura como:

$$E_c = T_v / T_d \quad \text{com limite máximo 1 (100\%).}$$

Notemos que o parâmetro σ não degrada E_c neste regime, uma vez que quanto menor for σ , menor será o número (relativo) de vectores de teste e, para T_L fixo, maior será o intervalo de permanência atribuído a cada VT_i . Seguidamente, começaremos por determinar o intervalo de verificação, T_v , e depois analisaremos o de detecção, T_d .

• T_v - tempo de verificação

O intervalo T_v é constituído pelo somatório dos intervalos de varrimento para cada registo:

$$T_v = \sum_{i=1}^N (t_{vi})$$

Com base no modo de operação da infra-estrutura CST descrito na secção 5.2, o BS μ C pode ter de efectuar diversos acessos ao registo de instrução, entre cada acesso ao registo, para determinar através de CB se a captura está disponível. O intervalo de tempo correspondente (t_v) pode ser traduzido matematicamente pela equação 5-1, que permite determinar o número de períodos ϕ de TCK, que o BS μ C necessita por vector, em cada registo:

$$t_v = (T_{DR} + \alpha) + (T_{IR} + \beta) \cdot K_i \quad (\text{Eq. 5-1})$$

onde temos, também em períodos ϕ :

T_{DR} - tempo de um “ciclo DR”, correspondente ao número de períodos necessário para efectuar um acesso a cada registo, de acordo com a figura 5-7.

T_{IR} - tempo de um “ciclo IR”, correspondente ao número de períodos necessário para aceder ao registo de instrução.

α - número de períodos necessários ao BS μ C para detectar a presença de erro no V_o recolhido.

β - número de períodos necessários ao BS μ C para detectar se a activação de CB já teve lugar, para o registo seleccionado (por acesso ao registo de instrução).

K_1 - número de inspecções de CB (através do registo de instrução), entre duas capturas consecutivas num mesmo registo.

Em condições normais, os parâmetros α e β são semelhantes e correspondem ao BS μ C efectuar uma comparação de V_o com VT_o para verificar se há erro (α), ou comparar CB para determinar se há captura disponível (β), seguidos dos saltos condicionais correspondentes no programa. Neste caso, sendo $\alpha \approx \beta$, virá:

$$t_v = T_{DR} + T_{IR} \cdot K_1 + \alpha \cdot (K_1 + 1)$$

Com base na análise efectuada para a eficiência de captura E_c associada à figura 5-16, a situação ideal (mas não obrigatória) é que não haja inspecções inúteis a CB, o que permite minimizar a frequência de TCK. Assim, o funcionamento óptimo (cujas condições pretendemos determinar) corresponde a ter o valor médio de K_1 aproximadamente unitário ($K_1 \approx 1$) e virá

$$t_v = T_{DR} + T_{IR} + 2 \cdot \alpha$$

ou

$$t_v = T_{DR} + T_{IR} + T_{OI}$$

Nesta equação, T_{OI} representa o intervalo de tempo correspondente ao número de períodos TCK necessários a outras instruções BS μ C¹⁰, além das que correspondem às operações de acesso aos registos e ao registo de instrução, na infra-estrutura CST. Tomando como referência o BS μ C descrito em [Fer92b] e as respectivas instruções, e supondo também que a frequência do BS μ C é igual à de TCK¹¹, obtemos:

$$\alpha = 4 \cdot \phi \quad ; \text{ número de períodos necessários à execução da instrução BS}\mu\text{C JPNE} \\ \text{(se há erro vai para a rotina respectiva).}$$

$$\text{Daqui virá} \quad T_{OI} = 2 \cdot 4 \cdot \phi = 8 \cdot \phi$$

Por outro lado, as instruções BS μ C utilizadas nos acessos aos registos, em T_{IR} e T_{DR} , são:

$$\text{TMS (0/1)} \rightarrow 2 \cdot \phi$$

$$\text{LD CNT, c} \rightarrow 6 \cdot \phi \quad ; \text{ onde } c \text{ define o número de bits a deslocar.}$$

$$\text{NSHFCP} \rightarrow \{3 + n_i + 3 \cdot \lceil (n_i/8) \rceil\} \cdot \phi \quad (n_i > 1)$$

¹⁰ - As instruções do BS μ C são aqui designadas por *instruções BS μ C*, para as distinguir das instruções BST (que são deslocadas para o registo de instrução).

¹¹ - Esta relação não é obrigatória, uma vez que a razão entre as frequências de relógio do BS μ C e de TCK são constantes, mas permite simplificar a análise.

Considerando que o controlador do TAP parte sempre do estado *Run-Test/Idle*, os acessos ao registo de instrução e aos registos correspondem respectivamente a:

- Acesso ao registo de instrução: $6 \text{ TMS} \rightarrow 12 \cdot \varphi$
- Acesso a cada registo: $5 \text{ TMS} \rightarrow 10 \cdot \varphi$
- A que se acrescentam ainda os períodos necessários à instrução LD CNT e ao deslocamento dos bits para as células dos registos.

Nos circuitos actuais, compatíveis com a norma 1149.1, o registo de instrução é geralmente de 8 bits, caso para o qual T_{IR} equivale às instruções BSμC:

$$6 \text{ TMS} + \text{LD CNT} + \text{NSHFCP}$$

de onde $T_{IR} = [2 \cdot 6 + 6 + (3 + 8 + 3)] \cdot \varphi$

ou ainda $T_{IR} = 32 \cdot \varphi$

Por seu lado, T_{DR} depende de n_i e corresponde às instruções BSμC:

$$5 \text{ TMS} + \text{LD CNT} + \text{NSHFCP}$$

$$T_{DR} = \{2 \cdot 5 + 6 + [3 + n_i + 6]\} \cdot \varphi \quad ; \text{ para } 8 < n_i \leq 16 \text{ [Fer92b].}$$

$$T_{DR} = (25 + n_i) \cdot \varphi$$

Podemos finalmente escrever a equação:

$$t_v = (32 + 25 + n_i + 8) \cdot \varphi$$

ou $t_v = (65 + n_i) \cdot \varphi$ (Eq. 5-2)

Numa aplicação real, sabendo n_i e a frequência de TCK, determina-se t_v para cada registo. A título de exemplo, apresentam-se na tabela 5-1 alguns valores de t_v para quatro frequências de TCK, considerando os casos de $n_i = 8$ e $n_i = 16$.

Tabela 5-1. Valores de t_v em função de TCK e de n_i

TCK (MHz)	t_v (μs)			
	5	10	20	30
$n_i = 8$	14,6	7,3	3,7	2,4
$n_i = 16$	16,2	8,1	4,1	2,7

O intervalo T_v corresponde ao tempo necessário para aceder uma vez a todos os N registos, sendo definido pela equação seguinte, onde se consideram os períodos de TCK fixos para cada registo (65) e os variáveis em função de n_i :

$$T_v = [65 \cdot N + \sum_{i=1}^N (n_i)] \cdot \varphi$$

Este intervalo, que condiciona a frequência a que cada registo é verificado e está indirectamente representado no gráfico da figura 5-17, onde se apresenta o número de períodos de TCK que lhe correspondem (T_v obtém-se multiplicando este valor por ϕ). As curvas apresentadas permitem obter o tempo de inspecção necessário ao BSμC para aceder a todos os registos, considerando como exemplo os casos de $N=4, 8, 12$ e 16 .

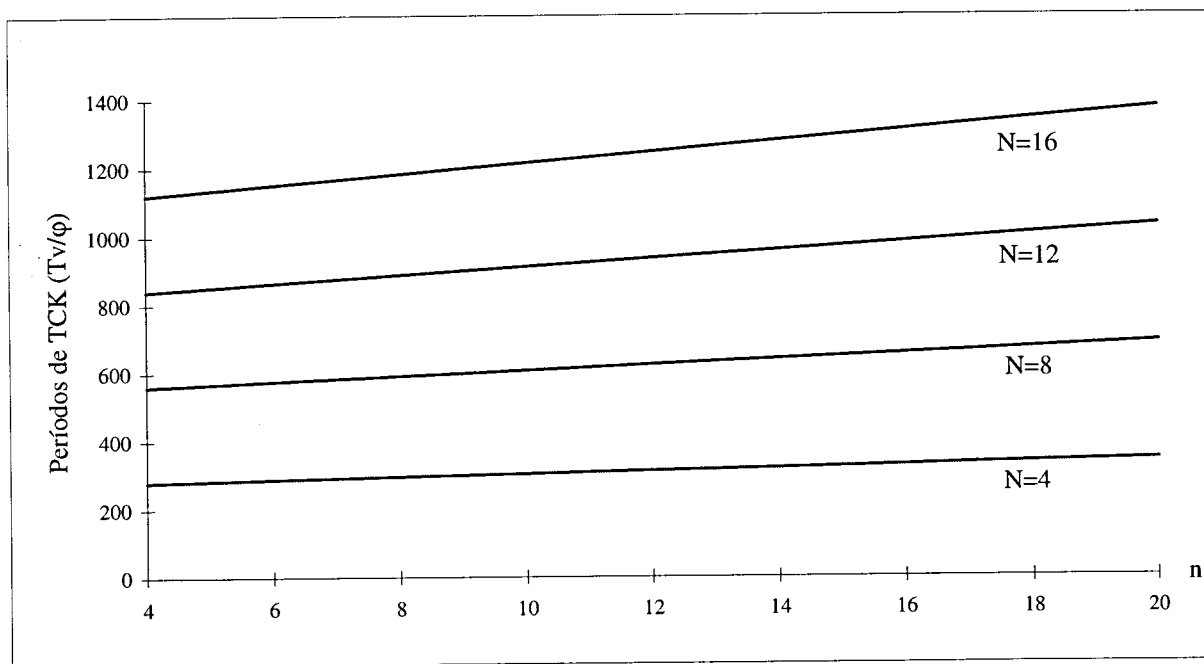


Figura 5-17. Curvas (T_v/ϕ) em função de n_i e N

Como exemplo, para um módulo com 12 registos ($N=12$), supondo deslocar em média 16 bits para cada registo ($n=16$) e uma frequência de TCK $\phi=10\text{MHz}$, o intervalo médio de verificação será de $972/(10 \cdot 10^6)$ segundos, inferior a $100\mu\text{s}$, o que corresponde a aceder a cada um dos 12 registos a mais de 10 KHz, verificando 10.000 vectores de teste potenciais por segundo, ou 120.000 vectores por segundo em todo o circuito de missão (valores estatísticos) [San97c]. Uma vez determinado o intervalo de verificação, passaremos agora a analisar o de detecção.

- **T_d - intervalo de tempo para detecção de um vector de teste**

Para um circuito com x bits de entrada e presumindo distribuição equiprovável dos sinais de entrada, o intervalo temporal associado ao número de períodos de relógio do módulo (ou sistema em que o circuito está inserido), necessário para a ocorrência estatística de um determinado V_i , é dado por (τ representa o período do relógio do módulo)

$$T_d = 2^x \cdot \tau \quad (\text{Eq. 5-3})$$

Para relacionar T_d com T_v , torna-se necessário identificar uma relação entre as frequências f do módulo e ϕ de TCK. Numa aplicação típica, a situação provável é o BSμC gerar o sinal TCK a partir do relógio do módulo, tal como sucede no BSμC descrito em [Fer92b] onde TCK é gerado por divisão em potências de 2. Por outro lado, na generalidade das aplicações, a limitação em frequência das células de deslocamento, resultante das respectivas características construtivas (quando inseridas no fabrico), deverá restringir ϕ a um valor próximo de $f/2$. Tomando a relação $\phi = f/2$ como referência e igualando x ao valor médio de bits de entrada que consideramos ao analisar T_v anteriormente, podemos determinar T_d em função de TCK. Como $\phi = 2 \cdot \tau$, a equação 5-3 pode ser escrita da seguinte forma:

$$T_d = 2^x \cdot \tau = 2^{n_i-1} \cdot \phi$$

Podemos assim sobrepor as curvas associadas a T_v e T_d no mesmo gráfico, representando uma vez mais estes parâmetros de forma indirecta, através do número de períodos de TCK que lhes estão associados, como se ilustra na figura 5-18.

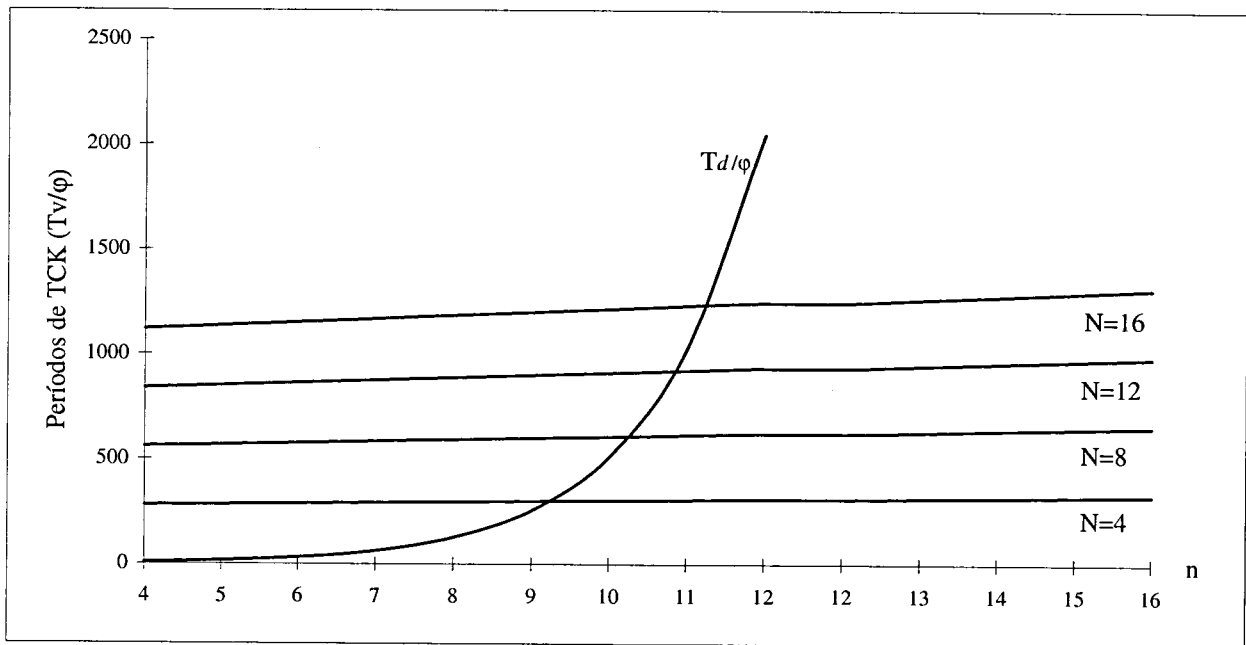


Figura 5-18. Comparação das curvas associadas a T_v e a T_d (para $f = 2\phi$)

Nestas curvas, para um determinado valor de n_i , são desejáveis soluções de projecto (N, n_i) , para as quais T_v seja superior a T_d , garantindo (estatisticamente) que cada acesso a um registo terá uma captura pronta. No entanto, as soluções à direita da curva T_d/ϕ podem ser atractivas em aplicações onde existam vectores de teste com probabilidade de ocorrência superior à média [Sal88].

Sabendo f , ϕ e n para os diversos registos e blocos funcionais associados, podemos determinar o número de vectores de teste verificáveis no intervalo de latência, como mostra a figura 5-19 para $T_L = 50\text{ms}$, $TCK = 10\text{MHz}$ e considerando a relação acima definida $\phi = 2 \cdot \tau$. Neste gráfico vemos que são verificados nesse intervalo de latência mais de 400 vectores de teste ($VT_i + VT_o$) em cada um dos 16 blocos funcionais de um potencial módulo, um valor 66% superior aos 250 vectores de teste que assumimos (no capítulo quatro) como máximo necessário para a monitorização de blocos funcionais com até 16 bits de entrada.

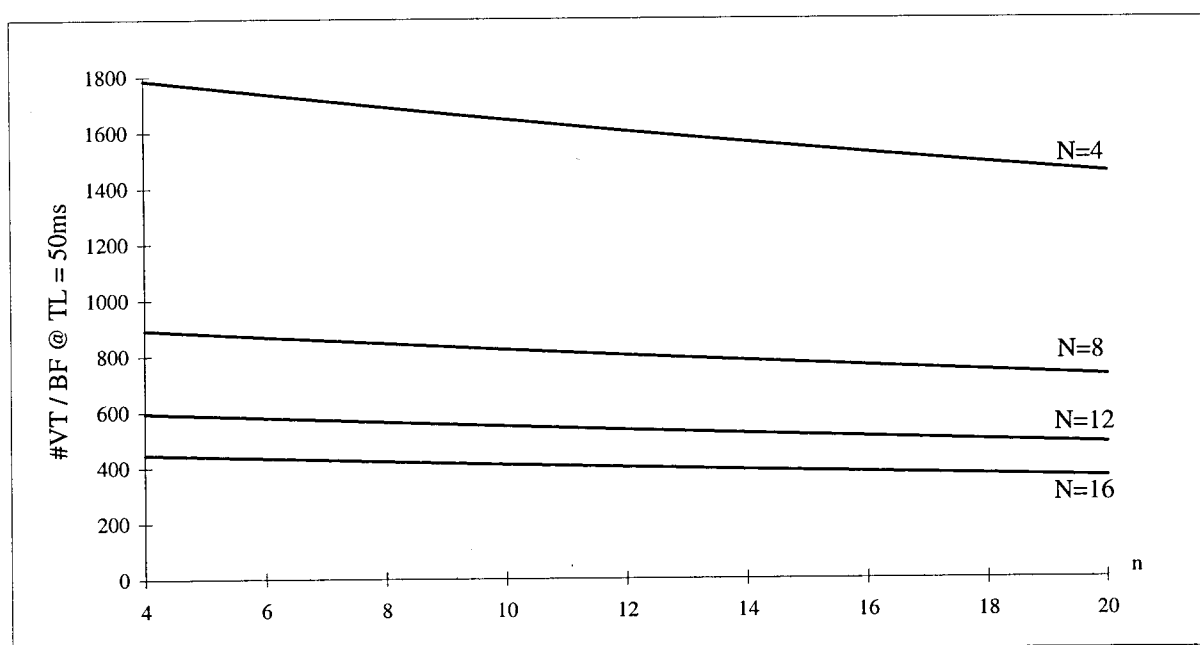


Figura 5-19. Número de vectores de teste verificados por BF em $T_L = 50\text{ms}$ ($\phi = 10\text{ MHz}$)

A análise precedente foi efectuada sem considerar a selecção de TMS, no caso de haver mais do que um circuito de missão no módulo. Esta assunção justifica-se porque a mudança de TMS é efectuada com uma instrução análoga a SELTAP n ¹² [Fer92b], que acresce apenas $2 \cdot \phi$ ao intervalo T_v . Uma vez que este valor será inferior a 1% do intervalo de verificação, como se infere das figuras 5-17 e 5-18, tem uma influência irrelevante no contexto apresentado, motivo porque não foi considerado.

¹² - Selecciona o TAP activo, quando na presença de múltiplas cadeias BST.

5.3.2 *Seleccção dos vectores de teste*

Para além do número de vectores de teste verificados, é necessário que a respectiva utilização seja optimizada no teste concorrente. Assim, a tabela de pares (VT_i, VT_o) será formada pelo conjunto dos vectores definidos para detectar faltas nos testes de produção, ordenados pelas seguintes regras:

- Calcular a cobertura de faltas de cada par $VT_i + VT_o$: ξ_i
- Estimar a probabilidade de ocorrência de cada V_i em funcionamento: ψ_i
 - No caso geral, ou se não houver informação: $\forall_i, \psi_i = 2^{-n}$
- Ordenar os pares pelo seu *coeficiente de eficiência*: $\gamma_i = \xi_i \cdot \psi_i$
- Seleccionar os m V_i com maior γ_i , para a cobertura de faltas pretendida.

O objectivo desejável consiste naturalmente em verificar pelo menos uma vez todos os vectores de teste, dentro do intervalo de latência aceitável, e determinar a frequência mínima de TCK necessária para tal. No entanto, e nos casos em que não seja garantida a verificação de todos os pares da tabela dentro do intervalo de latência, a ordenação proposta para os pares de vectores permitirá optimizar a cobertura de faltas, recomeçando ciclicamente a busca pelo topo da lista.

5.3.3 *Optimização do funcionamento da infra-estrutura CST*

Num processo de monitorização concorrente e não intrusivo, o intervalo de detecção T_d não pode ser alterado, porque depende apenas dos vectores que ocorrem na entrada¹³. Podemos, no entanto, actuar no intervalo de verificação T_v , minimizando o número de períodos TCK em cada acesso a um registo, uma vez que nos cálculos efectuados nesta secção, consideramos o controlo do TAP com início e fim no estado *Run-Test/Idle*. Contudo, a norma 1149.1 permite passar directamente do estado Update-DR ao estado Select-DR, o que reduz t_v . Face ao número de impulsos TCK necessário no acesso a um registo, este ganho será da ordem de $1/(5+16)$, um valor próximo de 5% e portanto não desprezável.

Uma segunda melhoria resulta da análise do comportamento dinâmico da infra-estrutura CST efectuada, que mostra ser possível ajustar a frequência de TCK para aumentar a probabilidade de ter captura pronta na primeira inspecção a CB. Nestas circunstâncias, o acesso ao registo de instrução deixaria de ter interesse, mas só poderá ser evitado se houver alternativas ao

¹³ - Notemos, contudo, que existe a possibilidade de particionar o grupo de células de detecção e procurar múltiplas combinações de VT_i em simultâneo [San97c], o que reduz T_d . Este processo no entanto tem implicações adicionais e não será analisado aqui.

procedimento que se baseia na instrução *SELRn*, para além de continuar a ser necessário confirmar a ocorrência de captura (saber se V_o é válido).

Comecemos por analisar as consequências de suprimir o acesso ao registo de instrução, notando que se eliminarmos os períodos TCK correspondentes, obtemos:

$$t_v = T_{DR} + \alpha$$

ou

$$t_v = (29+n) \cdot \phi \quad (\text{Eq. 5-4})$$

Este resultado, que reduz t_v para valores próximos de metade do número de períodos de TCK anteriormente calculado, permite diminuir a frequência de TCK ou duplicar a de verificação, duas razões que justificam a procura de alternativas para evitar o deslocamento da instrução *SELRn*.

Se não se aceder ao registo de instrução, a solução mais adequada para o BS μ C determinar se o V_o recolhido ao deslocar um novo VT_i foi capturado (é válido), é incluir a célula CB no vector a deslocar para o exterior, colocando-a antes de V_o (junto a TDO), ou depois, como se mostra na figura 5-20.

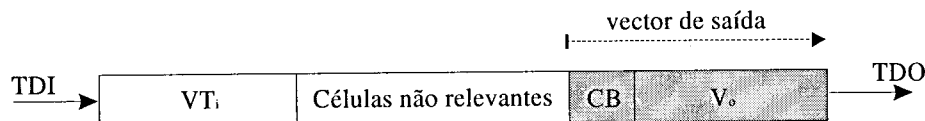


Figura 5-20. Deslocamento de CB juntamente com V_o .

Embora este tipo de procedimento tenha sido inicialmente considerado desaconselhável, a análise dinâmica justifica que, sendo a probabilidade de captura elevada, se torna agora vantajoso. Desta forma, o BS μ C recebe um V_o e juntamente com ele a indicação de ser ou não válido, o que resolve um dos problemas referidos, embora reste ainda encontrar um processo para aceder aos registos individualmente.

• A instrução MONIT

A instrução *SELRn* permite endereçar selectivamente um ou outro registo, particularidade que pode ter interesse no teste fora de serviço, mas em regime concorrente a sequência de acessos foi previamente definida, está optimizada e é fixa. Neste caso, a existência de um contador incrementado à passagem pelo estado Select-DR permite assegurar o necessário sequenciamento e evita aquela instrução. A instrução *SELRn* deixa assim de ser necessária, podendo ser

substituída por uma outra, que denominámos *MONIT* (Monitorização), também opcional e cuja função é análoga a *SEL_{Rn}*, mas que em lugar de aceitar como argumento o número do registo (*n*), activa o contador para multiplexagem dos registos, como se indica na figura 5-21.

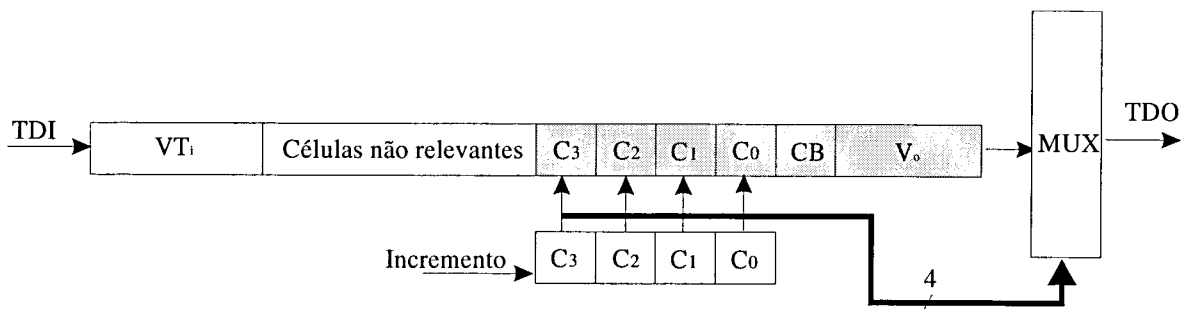


Figura 5-21. Contador de registos: deslocamento dos bits e selecção dos registos

Os bits C_i deste contador, carregados em células colocadas a seguir a CB, após serem lidas permitem ao BS μ C identificar univocamente o registo acedido. O valor do contador será neste caso capturado na passagem pelo estado Capture-DR, em células com topologia idêntica às de observação, cuja constituição se representa na figura 5-22.

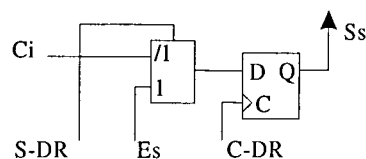


Figura 5-22. Constituição das células dedicadas à leitura do contador

Para evitar este conjunto de bits adicionais (por cada registo), existe a alternativa de implementar um contador análogo no BS μ C. No entanto, por razões de segurança, é necessário controlar o número do registo seleccionado nos dois contadores. Para este fim, o bit menos significativo do contador (C_0) será deslocado como se mostra na figura 5-23, o que permite ao BS μ C detectar imediatamente diferenças e reinicializar ambos os contadores.

Poderia igualmente ser utilizado um bit de paridade, em alternativa a C_0 , mas consideramos que a complexidade inerente não se justifica em virtude da frequente reinicialização dos contadores que propomos também. Neste sentido, para evitar erros cumulativos, os contadores serão reinicializados sempre que se completa um acesso a todos os registos (T_v), fazendo o controlador

do TAP passar pelo estado *Test-Logic/Reset*, ou *Run-test/Idle*, se este estado não for usado em regime normal, como referimos no início desta subsecção.

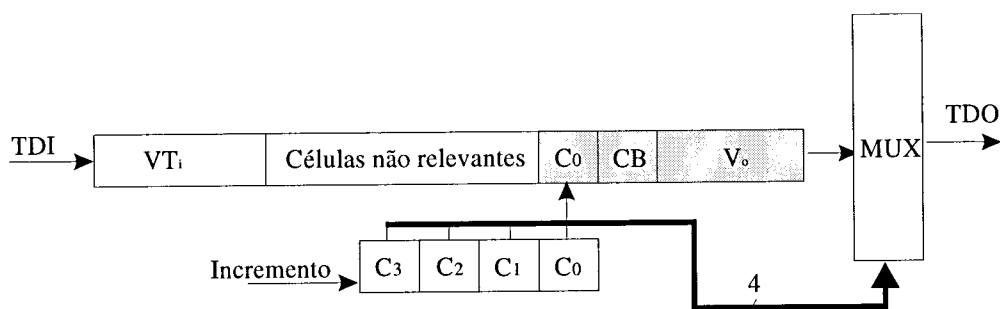


Figura 5-23. Deslocamento do bit menos significativo do contador

Em conclusão, a solução apresentada na figura 5-23 é a que apresenta menor sobrecarga, permitindo também otimizar a arquitectura CST para o menor intervalo de verificação dos vectores de teste nos registos, com as seguintes características:

- Reduz T_v a 50% do intervalo original, permitindo duplicar o número de vectores de teste verificados.
- Melhora a fiabilidade por evitar aceder ao registo de instrução (fase crítica na monitorização, como se referiu no início do capítulo).

A redução de T_v permite ainda, em alternativa, reduzir a frequência de TCK ou aumentar o número de registos que o BSμC pode verificar no mesmo intervalo de tempo.

5.4 Aplicabilidade da infra-estrutura CST

Nesta secção trataremos quatro aspectos da aplicabilidade da infra-estrutura CST, começando por calcular a sobrecarga que origina, analisando depois as relações de testabilidade entre registos (referidas na secção 5-1), a utilização da infra-estrutura CST na tolerância a falhas, e a concluir, a análise das limitações no que respeita aos circuitos sequenciais.

5.4.1 Sobrecarga da infra-estrutura CST

A sobrecarga da infra-estrutura CST, em relação à infra-estrutura BST, pode ser estimada, em termos médios, como sendo constituída por:

- Duas células adicionais por registo (CB e C₀)¹⁴.
- Circuito que gera o sinal D

Partindo do princípio que um registo terá normalmente mais de 20 células (*grosso modo*, oito de entrada, oito de saída e algumas acessórias)¹⁵, as células CB e C₀ representam uma sobrecarga inferior a 10% (2/20). A sobrecarga do circuito que gera o sinal D representa 1/(13+1) de cada célula de entrada, ou 7% por célula. Se estas representarem aproximadamente metade do total de células nos registos, a sobrecarga nestes será inferior a 4%. Temos assim que um registo CST representa uma sobrecarga não superior a $(10+4)\% = 14\%$ de um registo BST.

Por outro lado, em circuitos de média complexidade, o controlador do TAP representa cerca de metade do encargo da infra-estrutura BST e, nestas condições, a sobrecarga de teste implicada pela infra-estrutura CST baixa para um máximo de 7%, em relação ao conjunto de registos BST e controlador do TAP. Notemos ainda que o contador indicado na figura 5-23 (que só existe no caso de haver registos de utilizador), representa uma sobrecarga pequena no controlador do TAP (cerca de 20/500 portas lógicas), e que a lógica associada será partilhada com a que teria de existir para selecção dos registos, motivo porque não a consideramos aqui.

Finalmente, a infra-estrutura BST justifica-se essencialmente em componentes de média ou alta complexidade e, na generalidade das aplicações conhecidas, representa entre quatro e sete por cento do circuito de missão [Cro89, Kie98, Doo98, Zor98]. Podemos então estimar um acréscimo de sobrecarga, imputável à infra-estrutura CST, inferior a 1% num componente médio $(0,07 \cdot 0,07 = 0,0049 \approx 0,5\%)$, que não terá efeitos mensuráveis na respectiva fiabilidade.

5.4.2 Testabilidade entre registos

A infra-estrutura BST permite que, durante os testes fora de serviço, os registos sejam usados em determinados casos para fornecer os estímulos, enquanto outros registos capturam as respostas, ou para capturar as respostas aos estímulos injectados por outros registos. Esta versatilidade fica restringida no teste concorrente, em que as células dos registos só podem em cada instante detectar igualdades ou capturar respostas. Analisando as relações possíveis entre registos, conclui-se que:

¹⁴ - C₀ só existirá se forem implementados registos de utilizador no circuito de missão.

¹⁵ - Note-se que algumas tentativas de utilizar a infra-estrutura BST em serviço, sugerem que os registos deverão ter não mais de 40-50 células [Kie98].

- É permitido um grupo (de células) de detecção activar múltiplos grupos de captura, mesmo que em diferentes registos. Neste caso, cada grupo de detecção terá a sua célula CB, e será uma cópia deste CB que é capturada, numa célula idêntica à da figura 5-22, quando cada registo de captura é lido.
- É proibido um grupo de captura ser controlado por mais do que um grupo de detecção, uma vez que a multiplicidade de hipóteses resultante é de interpretação complexa e, em determinados casos, a resposta capturada pode não ter significado. Estas restrições deverão ser tidas em conta pelo projectista e pelas ferramentas que geram os vectores de teste.

Como se ilustra na figura 5-24, a infra-estrutura CST permite, em simultâneo, relações de cada um dos seguintes tipos (presumindo que A, e C são grupos apenas de detecção e que B e D são grupos apenas de captura):

- A-B, C-D.
- A-B, A-D.
- C-B, C-D.

As capturas resultantes de outras combinações, como por exemplo (A-B, C-B), são de interpretação complexa (em B), mesmo que sejam utilizadas duas células CB no grupo B.

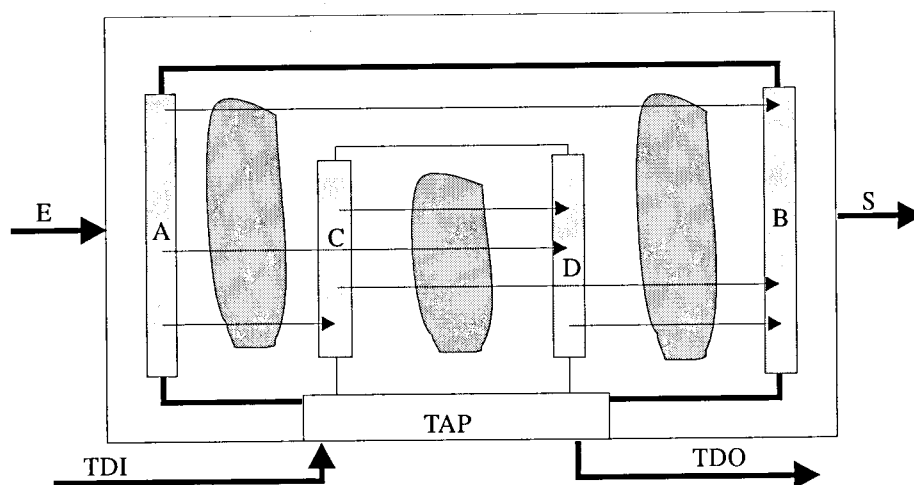


Figura 5-24. Relações de testabilidade inter-registos

Note-se, contudo, que até aqui temos considerado cada um dos grupos de células a funcionar em modo de detecção ou de captura, mas cada grupo de células pode ser utilizado para detecção ou captura em regime de alternância.

Esta variante é possível desde que sejam utilizadas as células de controlabilidade e observabilidade, e se controlem os sinais de Capture-DR nos grupos de captura, da forma descrita na secção 5.2. Por outro lado, nada impede que um registo seja fraccionado em múltiplos grupos de células, solução particularmente útil no registo periférico de componentes com elevado número de pinos, onde passam a ser permitidas múltiplas configurações, como se mostra na figura 5-25.

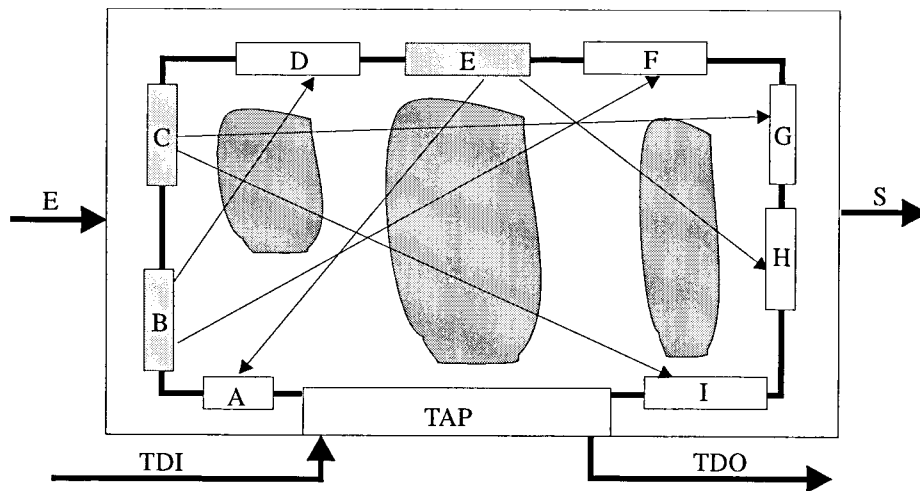


Figura 5-25. Definição de múltiplos grupos num mesmo registo

Note-se que, para acelerar o acesso ao registo representado na figura 5-25, os grupos de células de detecção (B, C, E) e captura (A, D, F, G, H, I) devem ser encadeados, em conformidade com o ponto 10.2.1.m da norma 1149.1a, o que nos conduz de novo às condições da secção 5.1.2, agora com cada grupo subdividido.

5.4.3 Aplicação à Tolerância a Falhas discreta

A infra-estrutura CST foi desenvolvida com o objectivo de monitorização, que melhora a confiabilidade, mas uma vez que não corrige as saídas, não permite tolerar falhas. Repare-se, no entanto, que as células de saída permitem na maioria dos casos também funções de controlabilidade, o que possibilita a *injecção* do conteúdo do andar de retenção. Aproveitando este facto, a infra-estrutura CST permite que, carregados VT_i e VT_o , a detecção de $V_i = VT_i$ provoque a *injecção* de VT_o nas saídas, ultrapassando a lógica interna, como indicamos na figura 5-26.

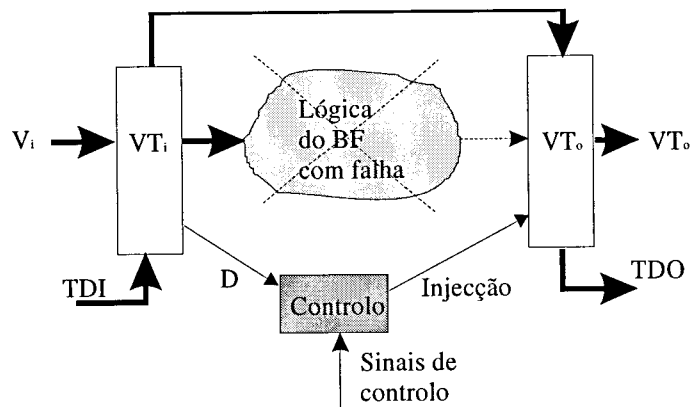


Figura 5-26. Injecção de vetores de teste sincronizada

Esta funcionalidade abre caminho para a inserção de características de tolerância a falhas, de que resulta a capacidade de injectar erros a montante do circuito ou de corrigir a saída de forma discreta, como se mostra na figura 5-27 e que descrevemos a seguir.

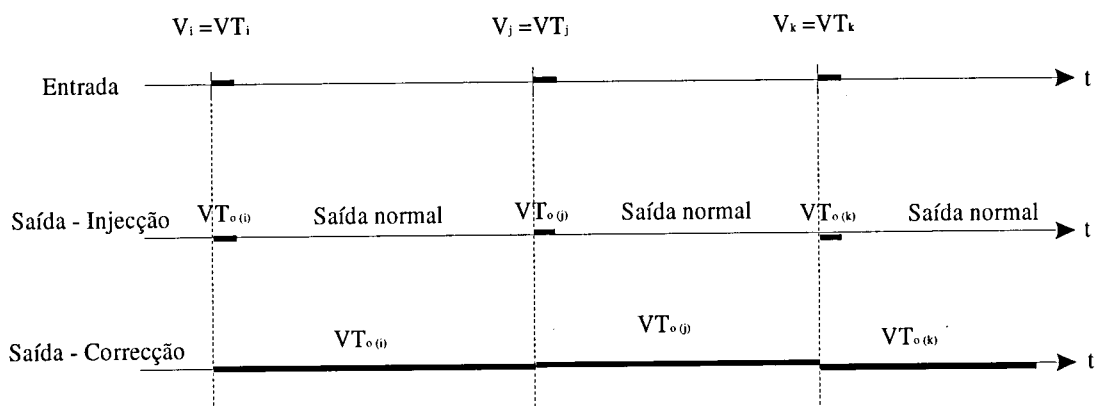


Figura 5-27. Diagrama temporal para injeção de erros e correcção discreta da saída

- **Injecção de erros na saída do circuito de missão**

Neste caso, a infra-estrutura CST permite injectar erros na saída, que simulam faltas no circuito interno, evitando injectores de faltas adicionais na depuração das características de tolerância a falhas. Este processo pode ser controlado pelo BSμC, com informação fornecida antecipadamente pelo controlador hierárquico central, tornando possível controlar a saída bit a bit, à frequência de trabalho do circuito de missão, sem o parar e sem necessidade de protótipos especiais.

Para controlar o sinal MODE nas células de saída, que permite injectar o conteúdo do andar de retenção, propomos uma instrução opcional com designação *INJECT*, que na solução mais simples utiliza um bit do registo de instrução, a que demos o nome de *Bit de Injecção (IB)*. O circuito da figura 5-28 mostra como, quando este bit estiver activo (“0”), o sinal Mode* acompanha o de detecção D.

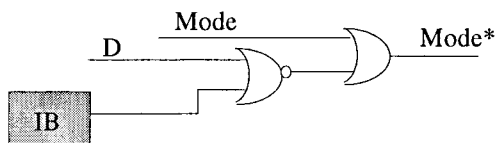


Figura 5-28. Controlo do sinal MODE na injeção de erros

Como se vê na figura 5-27, neste modo de funcionamento a saída só é afectada enquanto dura a igualdade na entrada, mantendo os valores normais durante a parte restante do funcionamento, uma vez que o circuito interno é suposto bom.

- **Substituição discreta de um bloco funcional com falha permanente**

A infra-estrutura CST permite também ultrapassar, de forma discreta, um circuito de missão com falha permanente. Ao contrário da injeção de erros, em que a saída só é alterada momentaneamente, a substituição discreta de um bloco funcional com falha implica reter cada VT₀ (correspondente a uma detecção na entrada) até que a saída seja actualizada por nova detecção, como foi representado na figura 5-27. Esta solução explora a existência dos andares de deslocamento e retenção das células de saída, de forma a que o sinal que actualiza o andar de retenção só seja activado em função do sinal de detecção D [San97c]. Uma instrução opcional, a que atribuímos a denominação *SURVIVE*, será enviada pelo BSuC para a infra-estrutura CST após detectar uma falha, activando o bit IB (admitindo que esta variante não se justifica em conjunto com a anterior) de forma a permitir que o sinal Update-DR* corresponda à inversão lógica de D, como se esquematiza na figura 5-29.

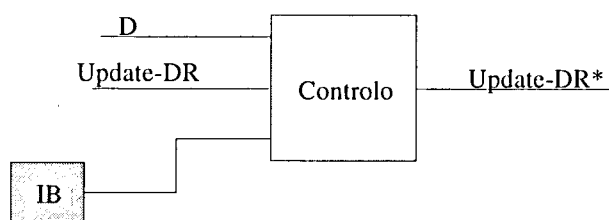


Figura 5-29. Controlo de Update-DR para substituição discreta do circuito de missão

Esta solução, embora simples de implementar, não assegura o confinamento do erro e, além disso, permite somente uma tolerância a falhas discreta e restrita aos vectores de teste memorizados. Como tal, não se justifica que seja analisada em mais pormenor, uma vez que a proposta que apresentaremos no capítulo seis é preferível em todos os aspectos de segurança, embora implicando uma sobrecarga maior.

5.4.4 O caso dos circuitos sequenciais

Nos circuitos sequenciais, para determinarmos se uma combinação de saída é ou não correcta, temos que conhecer as variáveis de estado, o que dificulta o modo de funcionamento da infra-estrutura CST descrita. No entanto, os elementos de memória (FF) de um circuito sequencial podem ser destacados da parte combinatória (modelo de Huffman), permitindo analisar separadamente ambas as partes, como se mostra na figura 5-30.

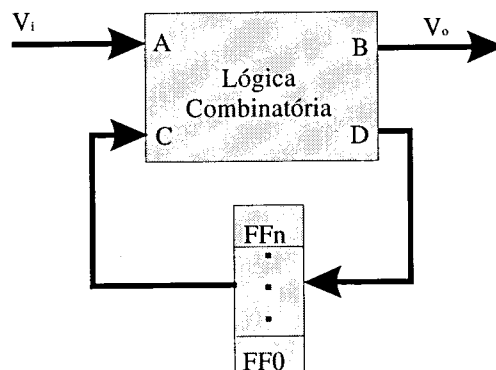


Figura 5-30. Os quatro grupos a monitorar no modelo de Huffman

Na generalidade das análises de circuitos sequenciais, em tolerância a falhas, considera-se que os elementos de memória não introduzem faltas, o que na prática implica criar FF *fiáveis* e pode ser conseguido pelos seguintes processos:

- *Transístores dos FF com maior área de silício*: estes transístores, para além de ficarem mais robustos face à degradação permanente, requerem também maior energia de comutação, o que melhora a imunidade a falhas temporárias [Abre97].
- *Elementos de memória com arquitectura TMR*: solução apontada como possível por alguns autores [Nic97], mas de que não conhecemos qualquer caso prático, é complexa e conduz a uma sobrecarga de confiabilidade e custo elevados.

- *Memórias Tolerantes a Perturbações* [Nic97]: trata-se essencialmente de um FF com arquitectura TMR, simplificada a nível de transístor. O custo é superior ao da alternativa com transístores robustos, mas a fiabilidade é também superior.

Qualquer destas variantes permite, de forma aproximada, considerar um modelo em que os FF são ideais. A infra-estrutura CST pode então ser utilizada para verificar os quatro grupos de sinais de entrada e saída da parte combinatória (A, B, C, D), mas as falhas temporárias não detectadas, quando ocorrem, podem aqui tornar-se indetectáveis, se os padrões da sequência errada puderem ser interpretados como correctos. Neste sentido, estes circuitos deverão ser projectados de forma a evitar que ocorram estados que permitam uma interpretação errada [Abr90]. No entanto, o tratamento dos circuitos sequenciais na tolerância a falhas é uma área em que o projectista continua a ter influência preponderante e em que são sempre possíveis diversas abordagens.

5.5 *Determinação da frequência mínima de TCK*

Para ilustrar a aplicação prática dos conceitos envolvidos na proposta apresentada neste capítulo, apresentaremos nesta secção diversos gráficos que facilitam a determinação da frequência mínima de TCK, para satisfazer as características da aplicação, considerando um exemplo para uma aplicação com as seguintes características:

- $T_L = 50ms$: intervalo de latência máximo admissível.
- $V = 200 VTs$: máximo número de vectores de teste para qualquer registo.
- $N = 12$: o BSuC monitoriza 12 registos.
- $n = 14$: média do número de bits a deslocar para os registos.
- $f = 20 MHz$: frequência de operação do circuito de missão.

Pretendemos para este caso determinar a frequência mínima de TCK. Embora este cálculo possa ser efectuado utilizando directamente as equações que indicaremos, o recurso a gráficos permite ter uma perspectiva mais abrangente da gama de valores envolvidos.

O primeiro passo consiste em determinarmos o intervalo de verificação, dado por $T_v = T_L/V$, através das curvas do gráfico representado na figura 5-31, que tem em abcissas o número máximo de vectores de teste (V). Obtemos assim $T_v = 250\mu s$, sendo este o intervalo de verificação máximo para se poder efectuar um acesso a todos os blocos funcionais do circuito.

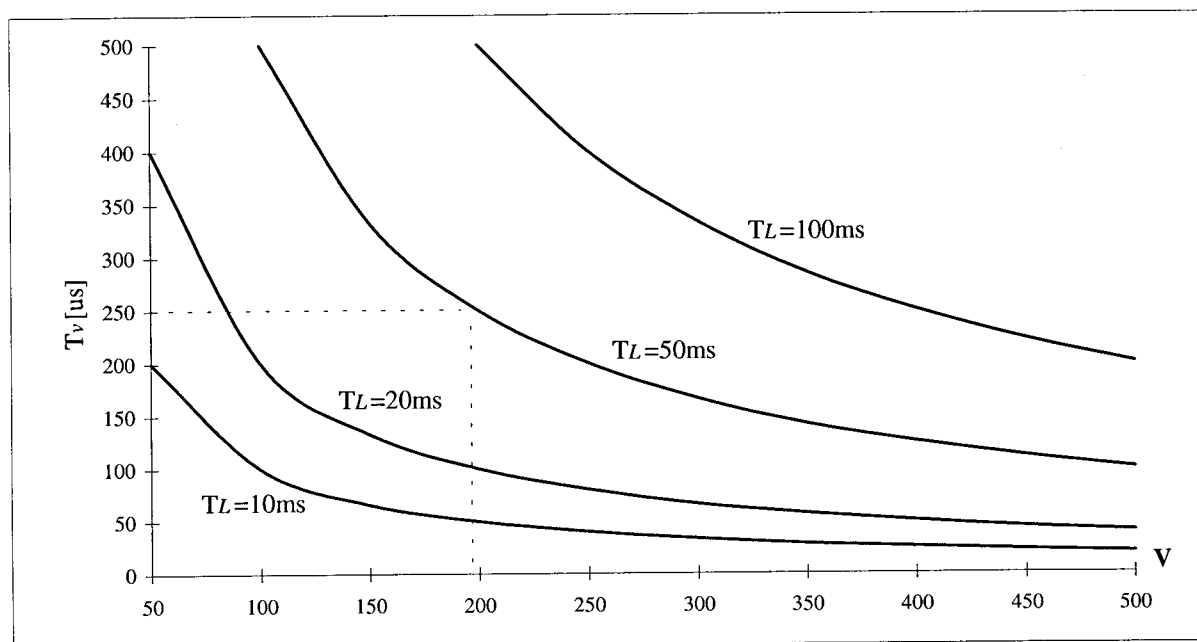


Figura 5-31. T_v como função de T_L e V

O segundo passo consiste em determinarmos $t_v = T_v/N$, através do gráfico apresentado na figura 5-32. Por consulta às curvas ilustradas, obtemos $t_v \approx 20$ μs, sendo este o intervalo de verificação máximo admissível por vector de teste e por registo.

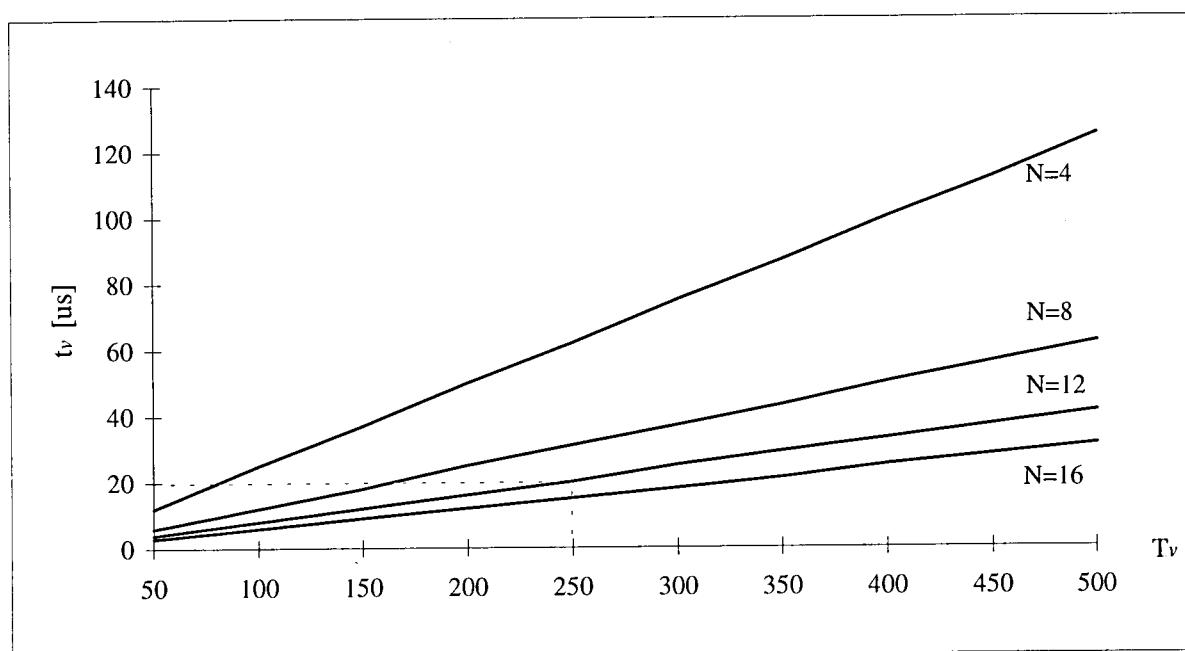


Figura 5-32. Intervalo t_v em função de T_v e N

O terceiro passo consiste em determinarmos a menor frequência de TCK que permite verificar os vectores de teste necessários. A partir da equação 5.4, obtemos $\phi = t_v/(29+n)$, e desta ainda a frequência de TCK como $1/\phi$. As curvas resultantes desta equação estão representadas no gráfico da figura 5-33, permitindo realçar a pequena influência que o valor de n tem na frequência de TCK. Para $t_v = 20 \mu s$ e $n = 14$, obtemos assim uma frequência $\phi > 2 \text{ MHz}$, para verificar 200 vectores de teste em cada um dos 12 blocos funcionais, num intervalo não superior a 50 ms.

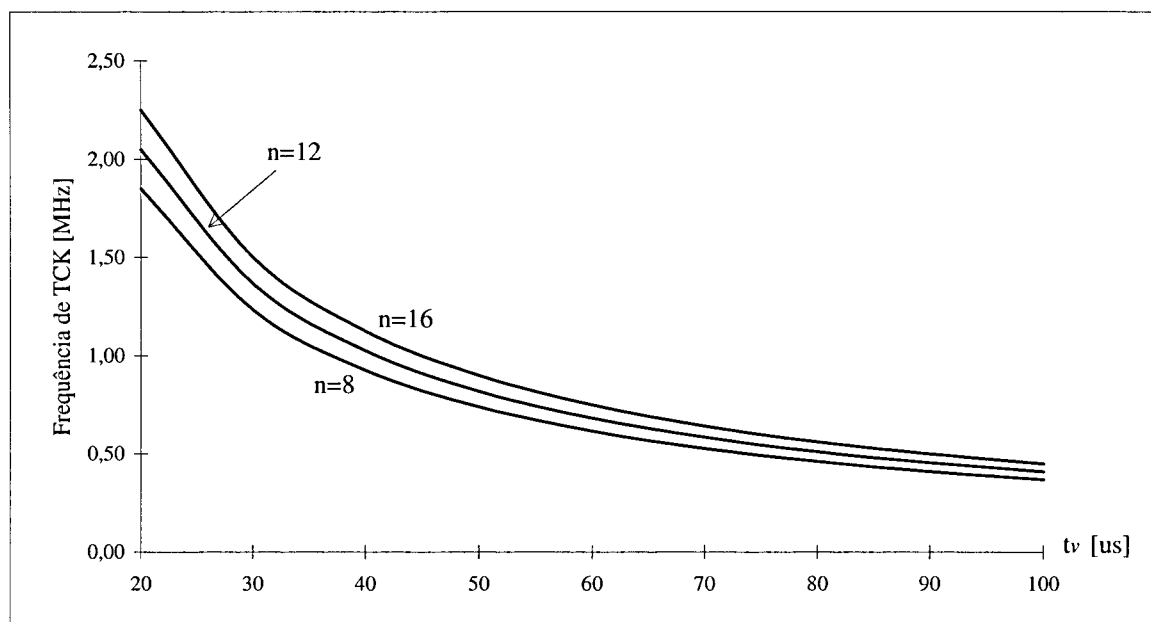


Figura 5-33. Frequência mínima de TCK para verificar os vectores necessários.

Devemos realçar o baixo valor da frequência de TCK, que constitui uma vantagem importante para a fiabilidade da infra-estrutura CST. Notemos ainda que este circuito permitiria monitorar até $14 \cdot 12 \cdot 2 = 336$ pontos de teste.

5.6 Introdução ao exemplo do anexo A

No Anexo A está documentado um exemplo em que escolhemos para bloco funcional o circuito de referência ISCAS'85 C17, esquematizado na figura 5-34. Atendendo a que o ênfase deve aqui ser colocado no processo de monitorização concorrente, escolhemos este circuito de missão, frequentemente utilizado por permitir uma visualização gráfica acessível, que facilita a inspecção de resultados e diagramas [Sav84, Brg85, Pom97, Flo98b]. O circuito C17 tem cinco entradas (E0-E4) e duas saídas (S0, S1), existindo 34 faltas *sempre-a* possíveis, que se reduzem a 22 após colapso [Brg84].

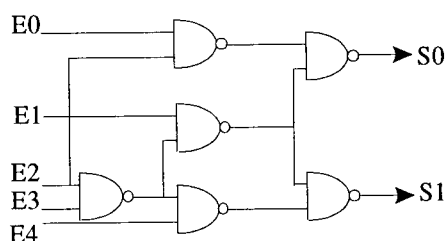


Figura 5-34. Esquema do circuito ISCAS'85 C17

O teste mais compacto conhecido para detecção de faltas neste circuito tem quatro vectores, indicados na Tabela 5-2 [Pom97].

Tabela 5-2. Vectores de detecção para o circuito ISCAS'85 C17

#VT	VT _i (E4...E0)	VT _o (S1, S0)
0	0 1 0 0 1	0 0
1	1 1 1 1 0	0 0
2	0 1 0 1 1	1 1
3	1 0 1 0 1	1 1

Os resultados de simulação, incluídos nos diagramas do anexo A, mostram como o processo de captura decorre no registo CST de forma assíncrona em relação à infra-estrutura BST original e permitem ilustrar em concreto o funcionamento da solução proposta ao longo deste capítulo.

5.7 Conclusão

A infra-estrutura CST permite a monitorização em regime concorrente de circuitos de missão que possam ser testados de forma modular, tanto combinatórios como sequenciais, se nestes os FF forem observáveis por registos de utilizador. Ao reutilizar a infra-estrutura BST e os vectores de teste previamente gerados, acrescentando a cada registo lógica adicional que lhe permite um funcionamento parcialmente autónomo dos registos em regime concorrente, proporciona-se ao utilizador uma técnica de monitorização transparente e com reduzido custo, que apresenta as seguintes vantagens principais:

- Não é intrusiva e não condiciona a topologia da lógica utilizada no projecto do circuito de missão [Abr90].
- Permite a máxima fiabilidade relativa do circuito de missão e uma forma de diversidade de projecto (parcial) fácil de implementar.
- Aproveita os vectores do teste de produção: os projectos baseados em bibliotecas permitem desenhar o circuito de missão a partir de blocos funcionais para os quais já são conhecidos os vectores de teste [Sil88], como no caso de núcleos IP.
- É possível utilizar qualquer dos dois tipos de células na infra-estrutura CST.

A principal limitação desta infra-estrutura, como forma de monitorização, é o intervalo de latência, que depende fundamentalmente do número de células no grupo de detecção.

No capítulo seguinte apresentaremos uma arquitectura que permite propriedades de tolerância a falhas, baseada na infra-estrutura CST, e destinada a circuitos dedicados a aplicações críticas.

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

Capítulo 6

Capítulo 7

Capítulo 8

Bibliografia

Anexos

Arquitectura XMR:

Redundância Modular Incompleta

6. **Arquitectura XMR: Redundância Modular Incompleta**

Como referimos no capítulo quatro, há hoje múltiplas aplicações na indústria automóvel, ferroviária, de equipamentos médicos, controlo de tráfico e outras, que necessitam de sistemas confiáveis de baixo custo. Na indústria automóvel, em particular, onde têm de ser conciliados a contenção de custo (poucas centenas de Euro são sensíveis no preço final) e elevados volumes de produção (milhões por ano), o conceito *X-by-wire*¹⁶, que os fabricantes pretendem implementar, levará dentro de poucos anos à existência num só veículo de mais de uma dezena de módulos de controlo com funções críticas, para as quais se espera um MTTF de 10 anos e períodos de funcionamento de 3 a 5 Horas por dia [Hei98].

Podemos prever que algumas aplicações menos críticas aceitarão pequenas latências de erro e poderão utilizar circuitos com monitorização do tipo CST, mas as falhas permanentes têm de ser toleradas nas funções mais críticas. No entanto, são raras as aplicações que exigem mascaramento de erro, motivando o interesse em alargar as capacidades da infra-estrutura CST a arquitecturas que permitam uma simbiose de baixo custo para o nível de confiabilidade 3 (*vide* tabela 4.1) e que sejam passíveis de integração num só componente.

Ao desenvolver a proposta apresentada no capítulo anterior, tivemos como principal objectivo aproveitar a infra-estrutura BST para monitorar o funcionamento, minimizando a sobrecarga no circuito integrado, e como tal a infra-estrutura CST terá sempre limitações dependentes da norma 1149.1. Com a motivação acima referida, o objectivo centra-se agora no projecto de componentes *tolerantes às falhas internas*, em particular componentes VLSI onde a sobrecarga de confiabilidade pode ser obtida com baixo custo, embora subsistam diversos problemas ainda sem solução, como referimos no capítulo quatro [San98d].

¹⁶ - O conceito *X-by-Wire* envolve a substituição de todas as ligações mecânicas accionadas pelo condutor, por actuadores controlados electronicamente. Para além de reduzir os custos de produção, facilita a interligação aos sistemas centralizados de gestão da segurança.

6.1 Sistemas duplex com CST

Um módulo CST permite detectar erros, mas não os confina. Esta característica pode, no entanto, ser obtida associando dois circuitos CST, para comparar em tempo real as saídas e pontos de teste idênticos nas duas réplicas em configuração duplex, como mostra a figura 6-1. Naturalmente, é suposto que a complexidade do BSμC e do multiplexer de saída, e a consequente taxa de falhas, sejam muito inferiores às dos circuitos de missão, cujas infraestruturas CST são alimentadas em paralelo por TDI, enquanto o BSμC recebe os dois TDO em entradas independentes, cuja diferença gera erro [San98a].

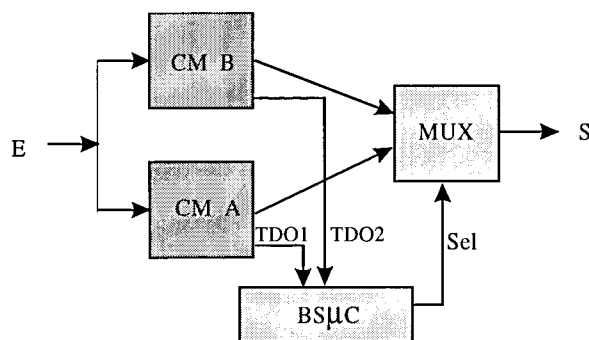


Figura 6-1. Arquitectura duplex com CST

Neste caso, os vectores de teste em memória servem apenas para detectar *fmc*, ou para localizar o circuito de missão avariado e seleccionar o correcto em caso de desacordo. Podemos também obter alguma diversidade de projecto com duas abordagens: *i*) implementar os circuitos de missão com arquitecturas diferenciadas (por exemplo, uma réplica em lógica complementar), mas esta solução requer dois projectos distintos, ou *ii*) recorrer à *diversidade temporal*, tendo em conta que as falhas transitórias são geralmente de curta duração, aumentando por essa forma a imunidade ao ruído [Ren98b]. Fisicamente, este tipo de diversidade pode ser implementado com um dos circuitos de missão atrasado de $\frac{1}{2}$ a vários períodos de relógio. No entanto, a diversidade de projecto origina sempre soluções complexas e parece improvável que venha a generalizar-se, sendo aliás evitada em diversos sistemas reais [Roq98].

Na configuração apresentada na figura 6.1, o bloco de multiplexagem deve poder encaminhar um ou outro circuito de missão para a saída do módulo. Para além disso, deve ainda confinar o erro e fornecer uma saída conhecida durante o intervalo de latência, funções que um simples multiplexer não permite. Há, pois, múltiplas vantagens em integrar a arquitectura apresentada num só CI, mas o trabalho do projectista tem diversas condicionantes, que passamos a analisar.

6.1.1 *Considerações sobre o mascaramento e detecção de erro*

O objectivo último de um circuito tolerante a falhas é corrigir o erro de imediato mas, como vimos, a redundância necessária ao mascaramento degrada o MTTF e o custo. Nas arquitecturas tradicionais, quando se procura minimizar a redundância física, diminui a quantidade de informação disponível e a correcção deixa de ser possível para redundâncias de ordem inferior a três. No entanto, o mascaramento raramente é necessário porque muitas aplicações têm alguma inércia, sendo suficiente confinar o erro e recuperar após um pequeno intervalo de latência. Podemos então estabelecer como primeiro objectivo um CI que assegure o nível de confiabilidade 3 (vide tabela 4-1):

- *Projectar CIs com arquitectura para confinar o erro e recuperar, que devem gerar saídas conhecidas (seguras) durante o intervalo de latência.*

Confinar o erro implica verificar todas as palavras de saída, o que requer uma quantidade de informação pelo menos dupla. Uma vez que os códigos detectores de erro são limitativos e que a infra-estrutura BST também não tem capacidade de suprir informação suficiente para tal fim, em regime concorrente, resta duplicar o circuito de missão. Por outro lado, excluídos os códigos correctores de erro pelas razões apontadas no capítulo três, a única alternativa para tolerar falhas permanentes é, pelo menos, duplicar o circuito de missão. Uma terceira condicionante são as limitações da replicação nos componentes VLSI, apontadas no capítulo quatro. Temos assim, e por caminhos diferentes, uma orientação da arquitectura desejável:

- *Um módulo com circuito de missão duplicado.*

Num módulo com dois circuitos de missão, em que a detecção de erro não é por codificação, existe um intervalo entre a *detecção* do erro e a *localização* do circuito de missão avariado, durante o qual não deve ser perdida a informação correspondente aos vectores de entrada que chegam entretanto ao circuito de missão. Esta limitação leva-nos ao último objectivo:

- *Manter os circuitos de missão em funcionamento durante o intervalo de latência.*

A generalidade das arquitecturas tolerantes a falhas considera faltas simples, uma vez que as faltas múltiplas conduziriam a soluções demasiado caras ou impraticáveis e, como vimos, durante a vida útil a falta simples é representativa. Nestas condições, o módulo com dois circuitos de missão pode ser aceite como seguro e tem uma sobrecarga pouco superior à da lógica auto-verificável (vide tabela 4.3), pelo que os MTTF são próximos. Para concluir, a duplicação reúne o maior número de vantagens, mas não tem informação suficiente para recuperar, assunto que analisaremos seguidamente.

6.1.2 Definição de arquitecturas com Redundância Fraccionária

O problema que temos, da secção anterior, é a inexistência de arquitecturas de tolerância a falhas, por replicação, intermédias. De acordo com a equação 3.7, a redundância de ordem três permite correcção e, abaixo desta, a redundância de ordem dois apenas pode detectar erros, sem haver gradação entre ambas. As soluções tradicionais para tolerar falhas a nível físico, partindo de dois circuitos de missão, são a codificação [Lub94] ou a inclusão de um terceiro circuito de missão (e temos de novo um TMR), mas o encargo de ambas é exagerado para integrar num só CI, com custo aceitável. Contudo, existe agora a possibilidade de fornecer informação adicional ao módulo através da infra-estrutura BST, embora as limitações que já lhe apontámos em regime concorrente façam com que esta informação seja insuficiente, para ser útil numa arquitectura tradicional. A procura de uma solução para este problema conduziu-nos à *teoria de informação aplicada ao teste* [Agr81, Set82]. É sabido que a informação na saída de um circuito depende das características do próprio circuito e da informação na entrada. Uma falta no circuito origina perda de informação (útil) na saída, excepto possivelmente se existir reconvergência interna. De acordo com a referida teoria, a probabilidade de um determinado teste detectar um erro (ou falta) no circuito da figura 6-2 é dada por:

$$P(T) = 1 - 2^{-(H_o \cdot T) / k}$$

onde:

- k é o número de nós (pontos de teste) em que se efectua a observação.
- P é a probabilidade de detecção nesses nós.
- H_o é a quantidade de informação, em bits/vector, nos nós analisados (saídas na figura 6-2).
- T é o número de vectores do teste.

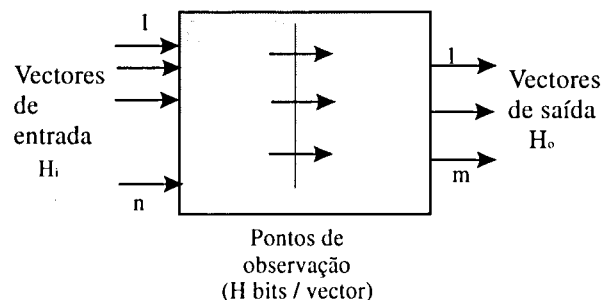


Figura 6-2. Quantidade de informação num circuito

Uma vez que, para um determinado modelo de faltas, o produto $H_0 \cdot T$ corresponde à cobertura de faltas do teste a montante dos nós analisados (as saídas no nosso caso), a probabilidade de detectar um erro é máxima quando a quantidade de informação na saída do circuito é maximizada [Agr81]. Este resultado traduz-se na impossibilidade de suprimir erros quando o “canal de informação” atinge a capacidade máxima de transmissão, o que leva à necessidade de codificação. Por outro lado, uma falta é indetectável se a quantidade de informação transmitida for nula, porque não há erros.

Outra conclusão que resulta desta teoria, é que um teste com capacidade de detecção maximizada, tem de ser gerado com base na descrição estrutural do circuito, o que coincide com o que referimos no capítulo dois, sobre a compactação e minimização do custo de aplicação do teste. Num teste compacto, os vectores de teste devem produzir a máxima sensibilização de percursos, para detectar o maior número de faltas, e como tal maximizar a informação necessária na saída, para detectar as faltas alvo.

Considerando agora que a probabilidade de detecção cresce com a quantidade de informação utilizada no teste, e que o intervalo de detecção está inversamente relacionado com aquela probabilidade, concluímos que uma maior quantidade de informação contida nos vectores de teste corresponde a uma desactivação mais rápida do circuito de missão em falta. Com base neste conceito, introduzimos o seguinte Lema:

- *Num circuito com monitorização concorrente, o intervalo de latência aceitável condiciona a quantidade de redundância necessária.*

A principal consequência deste lema é admitir a existência de arquitecturas com *redundância fraccionária*, que possibilitam a um módulo com dois circuitos de missão características tanto mais próximas das de um TMR, quanto maior a quantidade de informação adicional que lhe seja fornecida [San98b]. A figura 6-3 representa as linhas verticais correspondentes aos encargos mínimos teóricos em arquitecturas replicadas, para detecção (200%, para a duplicação) e correcção (300%, em conformidade com a equação 3-6). Além destas, estão as novas curvas que o lema permite antever como *função do inverso da quantidade de informação adicional disponível no módulo*, que no caso das arquitecturas convencionais tem um relação directa com o encargo, e que se complementa com dois casos de soluções conhecidas, o TMR (real, com encargo entre 320 e 400%) e o B²UBIST [Lub94] para o qual tomamos valores de base indicados em [Dua97]. Um circuito CST situa-se na zona **A** e pretendemos agora uma arquitectura na zona **B**, que permita confinar e corrigir o erro.

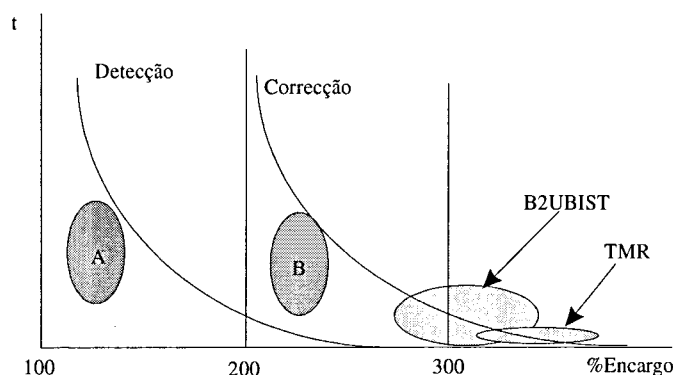


Figura 6-3. Intervalo de detecção e correcção *versus* encargo associado

6.2 A arquitectura XMR

A nova arquitectura que desenvolvemos e denominámos *Redundância Modular Incompleta*, ou simbolicamente **XMR** [San98c], é um subgrupo das arquitecturas NMR, constituída por um módulo com dois circuitos de missão idênticos, que partilham uma infra-estrutura CST comum, melhorada para funções de detecção de erro e localização do circuito de missão avariado, como mostra a figura 6-4.

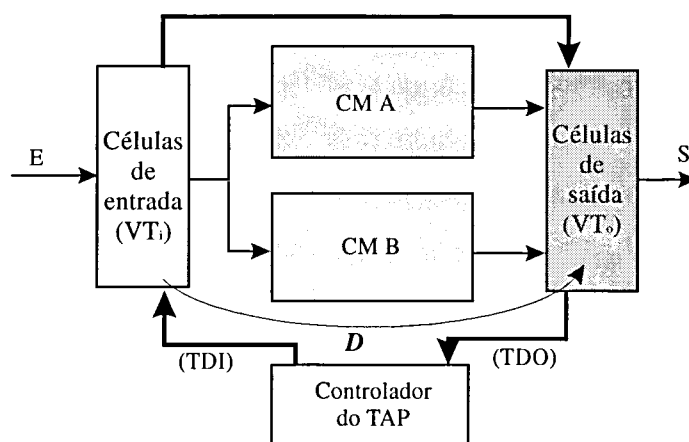


Figura 6-4. Arquitectura XMR

O grupo de células de entrada, com capacidade de detecção (o sinal D), é idêntico ao que integra a infra-estrutura CST. As células de saída, no entanto, foram redesenhadas para receber as saídas dos dois circuitos de missão e assegurar a comparação entre elas. Um desacordo destes sinais coloca de imediato as saídas do módulo num estado predefinido, até ser localizado e desactivado

o circuito de missão avariado. Este procedimento é efectuado com o auxílio da informação contida nos vectores de teste armazenados no BSμC, uma vez que na sequência de uma detecção na entrada, as células de saída contêm as seguintes três componentes de informação:

- As saídas dos circuitos de missão A e B.
- A resposta esperada: o VT_0 armazenado nas células do grupo de saída.

As duas primeiras facultam uma comparação contínua, enquanto a terceira é suficiente para votar e permite a esta arquitectura, essencialmente duplicada, um comportamento que podemos considerar como *TMR discreto*, ou XMR, em que $2 < X < 3$. Desta forma, exceptuando as *fmc* tal como nas arquitecturas TMR, o confinamento é autónomo por depender apenas dos circuitos de missão A e B, tendo a quantidade de informação acessível através da infra-estrutura BST influência somente no intervalo de recuperação e na detecção daquele tipo de faltas.

6.2.1 Princípio de funcionamento da arquitectura XMR

Seja Q a quantidade de informação disponível num circuito de missão e p a fracção de informação armazenada em memória e fornecida pela infra-estrutura BST. Sempre que ocorre uma detecção na entrada, um módulo XMR dispõe da quantidade de informação definida por $Q_M = Q \cdot (1 + 1 + p)$.

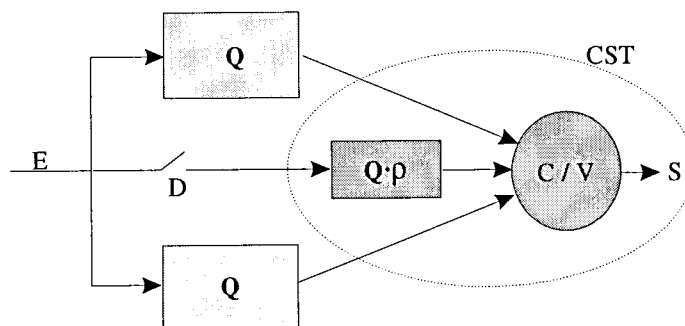


Figura 6-5. Princípio de funcionamento da arquitectura XMR

Nas arquitecturas NMR, N identifica o nível de redundância e cada uma das réplicas trata igual quantidade de informação. Na arquitectura XMR, não há correspondência entre o encargo físico e a quantidade de informação disponível, parte da qual vem do exterior, e X simboliza esta incógnita. No entanto, é-nos possível definir X de forma mais rigorosa, começando por notar que há dois valores diferentes para este parâmetro:

- X_S - coeficiente da *sobrecarga* que a infra-estrutura CST melhorada representa face ao circuito de missão. Deve ser minimizado e, como vimos no capítulo cinco, em regra é inferior a 10%.
- X_I - coeficiente de *informação*. Como vimos na secção 6.1.2, é numericamente igual à cobertura de faltas global dos vectores do teste, pelo que deve ser maximizado.

Tendo em conta que X_I deve ser maximizado e X_S minimizado, a consideração conjunta destes dois parâmetros permite-nos definir X pela equação:

$$X = 2 + (X_I - X_S) \quad X \in]2, 3[\quad (\text{Eq. 6-1})$$

Esta equação considera a informação nos dois módulos e tem como majorante 3, que equivale às características de uma arquitectura TMR tradicional. Num CI típico, podemos esperar que a infra-estrutura CST represente menos de 10% do CM e que a cobertura de faltas seja superior a 90%, pelo que X deverá rondar 2,8. A sobrecarga associada ao BS μ C, sendo um componente de produção em série e acumulando outras funções (vigilância e protocolos de comunicação), terá um custo negligenciável.

6.2.2 Modos de operação

Um componente XMR permite os seguintes modos de funcionamento:

- *Sem BS μ C* (funcionamento independente) - neste caso teremos $p=0$ e a quantidade de informação disponível no XMR corresponde somente a $Q_M = 2 \cdot Q$, suficiente para confinar erros num só circuito de missão, mas não para os corrigir.
- *Com BS μ C* - neste caso teremos três possibilidades de funcionamento:
 - *Modo Normal*, no qual $Q_M = Q \cdot (1 + p)$. Perante a ocorrência de uma falha, teremos confinamento imediato e será produzida uma saída *segura*, até que o módulo com erro seja desactivado.
 - *Modo CST*, no qual $Q_M = Q \cdot (1 + p)$. Este modo de funcionamento tem lugar a partir do modo normal, perante a ocorrência de uma falha e após ter sido desactivado o circuito com erro. Caso a falha tenha sido temporária, é possível o retorno ao modo normal.
 - *Modo de Sobrevivência*, no qual $Q_M = Q \cdot p$, e que decorre do modo CST, na situação de o segundo circuito de missão também falhar. Tal como foi descrito na secção 5.4, os vectores de teste serão carregados e VT₀ injectado, quando o sinal D é activado.

A saída *segura*, que referimos anteriormente, é fornecida durante o intervalo de latência e sempre que seja impossível votar, eventualmente porque o BS μ C falhou, mantendo uma palavra predefinida na saída. As características destes modos estão resumidas na tabela 6-1.

Tabela 6-1. Características dos Modos de Funcionamento

Modo	Q_M	Confinamento	Saída segura	Recuperação
Independente	$Q \cdot 2$	S	S	N
Normal	$Q \cdot (1 + 1 + p)$	S	S	S
CST	$Q \cdot (1 + p)$	Temporizado	Temporizado	Sobrevivência
Sobrevivência	$Q \cdot p$	N	* (ver texto)	N

No modo de sobrevivência, em que não existe outra informação de referência, a saída segura (*) só é possível se a infra-estrutura CST estiver funcional. Notemos finalmente que, na arquitectura XMR, a infra-estrutura BST é reutilizada para 4 funções distintas:

- Confinar erros.
- Prover uma palavra segura na saída.
- Fornecer informação para monitorar.
- Votar.

6.2.3 Características dinâmicas

Na arquitectura XMR o erro é confinado em resultado da comparação, sendo a principal restrição o *intervalo de recuperação*. Ao contrário da infra-estrutura CST, onde é necessário verificar o máximo de vectores de teste no intervalo de latência aceitável, aqui a primeira votação após confinamento permite desactivar o circuito de missão defeituoso.

O intervalo de recuperação médio, função das características do sinal de entrada, corresponde assim ao parâmetro T_d , determinado no capítulo anterior para a infra-estrutura CST. No entanto, enquanto na infra-estrutura CST pretendemos uma eficiência de captura (E_c) elevada, para maximizar o número de vectores de teste verificados, no funcionamento XMR o parâmetro E_c não é relevante e deve ser considerada a média estatística do intervalo de detecção, que conduz ao intervalo de recuperação, dado por:

$$T_d = 2^{n-1} \cdot \tau$$

O gráfico da figura 6-6 ilustra este intervalo em função do número de bits de entrada, n , para as frequências de 1 MHz, 10 MHz e 20 MHz.

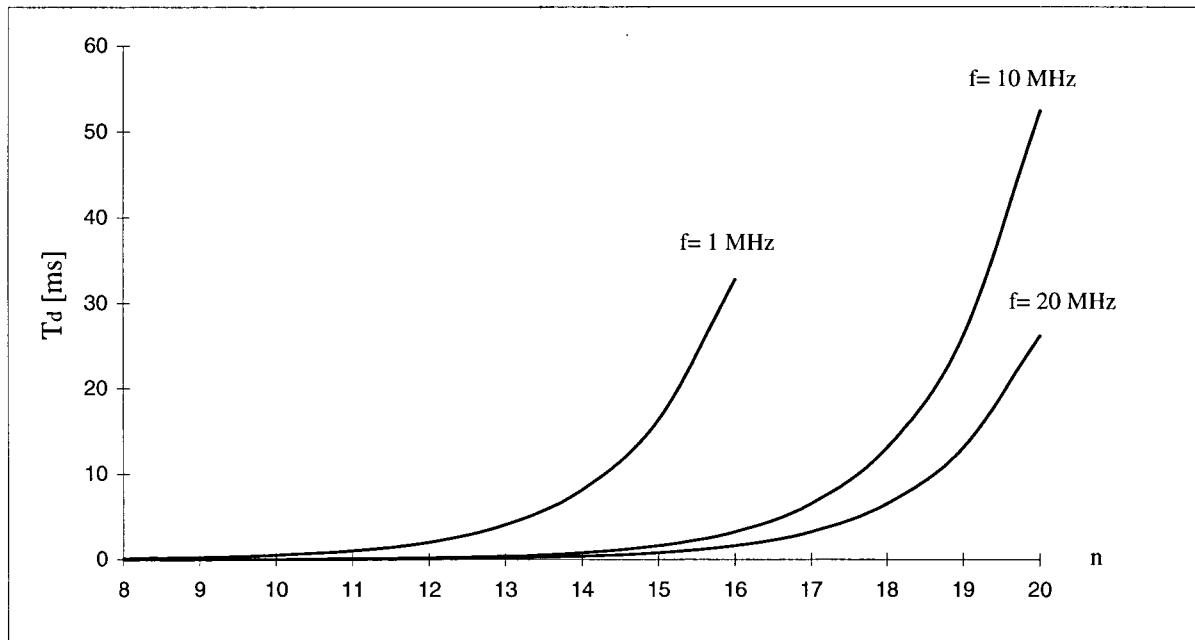


Figura 6-6. Tempo médio de recuperação num CI XMR

Podemos ver que para um intervalo de latência (admissível) de 50 ms, são permitidas entradas com (até) 16 bits, para frequências de 1 MHz, e que acima de 10 MHz os tempos de recuperação em 16 bits são inferiores a 5 ms, sendo possível monitorar até 20 bits dentro do referido intervalo de latência.

6.2.4 As células de decisão

As células de saída indicadas na figura 6-4 e representadas na figura 6-7, recebem os sinais A e B, dos circuitos de missão respectivos, e um bit de VT_o (C), deslocado e renovado nas condições descritas no capítulo anterior, que será considerado apenas quando o sinal de detecção (D) estiver activo. As saídas são o sinal Z, o sinal de erro (E) e o de selecção (S), gerados de acordo com duas acções independentes:

- Comparar os sinais A e B.
- Votar sempre que ocorre igualdade na entrada ($V_i = VT_i$).

A constituição desta célula é derivada da que implementa as funções de controlabilidade e observabilidade (*vide* figura 2-4), à qual foi acrescentada a lógica de decisão, formada pelo bloco que confina o erro e encaminha para X o sinal correcto (A/B). Esta célula tem ainda a capacidade de manter a saída Z num estado conhecido, enquanto durar a indecisão, e gera os sinais de erro e selecção, conjugando a informação contida nos sinais C e D.

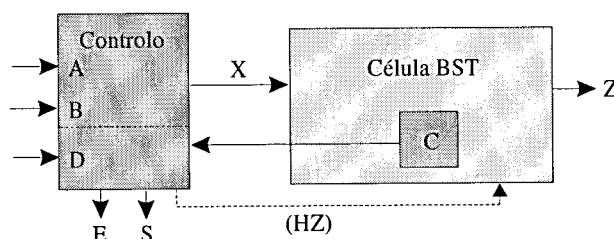


Figura 6-7. Esquema da célula de decisão

A *Comparação* entre A e B confina os erros com origem num circuito de missão e tem prioridade sobre as restantes acções, sendo independente da informação do BSμC. Um desacordo entre A e B activa o sinal E e coloca X num estado considerado seguro, predefinido para formar a palavra de saída desejada, que pode também ser de alta impedância, se a célula dispuser dessa característica (e que será controlada pelo sinal HZ representado).

A *Votação* só é possível quando o sinal D indica que existe uma igualdade na entrada, que deve ser confirmada na saída ($C=A=B$), tornando possível desactivar o circuito de missão avariado. Esta acção permite detectar dois tipos de erro:

- $A=B \neq C$ significa uma *fmc*, ou erro na infra-estrutura CST/XMR.
- Após um desacordo ($A \neq B$), permite localizar o circuito de missão errado e direccionar o correcto para a saída do módulo.

Finalmente, $A=B=C$ confirma a operação normal e, na hipótese de falta simples, corresponde a testar permanentemente a infra-estrutura CST/XMR, tal como são testados os circuitos de missão. É de realçar ainda que, durante o período de latência, ambos os circuitos de missão mantêm a operação normal, o que é particularmente importante para lógica sequencial, enquanto as saídas do módulo mantêm um estado seguro, entendido como uma combinação que conduza a lógica a jusante para estados seguros, ou não altere os actuais. Tomando “0” como estado seguro (pressuposto que se mantém no anexo B, onde exemplificamos a implementação da arquitectura

XMR), apresentamos na tabela 6-2 a relação lógica entre as entradas (A, B, C) e as saídas (X, E), que servirá de referência para definir estas células. O sinal S é auxiliar e será definido adiante. Note-se que os estados marcados com (*) são compatíveis com um TMR, mas a fiabilidade da informação veiculada pela infra-estrutura BST deve ser tida em conta nesta opção, um assunto a que voltaremos no próximo capítulo.

Tabela 6-2. Matriz de decisão

A	B	C	X	E
0	0	0	0	0
0	0	1	0	1*
0	1	0	0	1
0	1	1	1	1
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1*
1	1	1	1	0

Esta tabela é válida para uma célula, mas não pode ser generalizada directamente para uma saída multi-bit, constituída por duas ou mais células. Neste caso, logo que uma célula detectar erro, o confinamento deve ser estendido a todas as células, implicando que os sinais E e S gerados por cada célula de decisão sejam reunidos, memorizados convenientemente e redistribuídos, como se mostra na figura 6-8.

O bloco de *Retenção e Selecção* representado, recebe os sinais conjuntos E e S que, depois de memorizados em dois FF, dão origem aos sinais que denominámos F e T (na sequência alfabética correspondente), comuns ao controlo de todas as células de saída. Este bloco recebe ainda o sinal de detecção D, que tem funções de sincronismo para as tomadas de decisão. Notemos que este bloco é necessário desde que haja duas células na saída, mas independente do número de células. Para além disso, se o projectista tiver acesso à função AND em lógica cablada (*wired-and*), os sinais E e S podem ser gerados recorrendo a esta função, o que permite utilizar estas duas linhas comuns a todas as células de decisão.

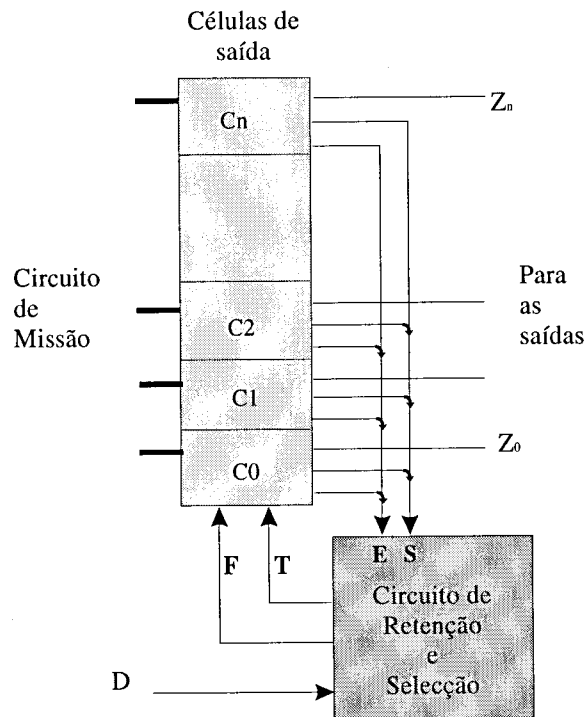


Figura 6-8. Localização do circuito de retenção e selecção

Com base nas considerações que conduziram à figura 6-8 (saídas multi-bit), o bloco de controlo representado na figura 6-7 foi redefinido com se mostra na figura 6-9, com um bloco que gera a saída de sinal $X = f(A, B, F, T)$ e outro que dá origem aos sinais de erro e selecção $(E, S) = f(A, B, C, D)$.

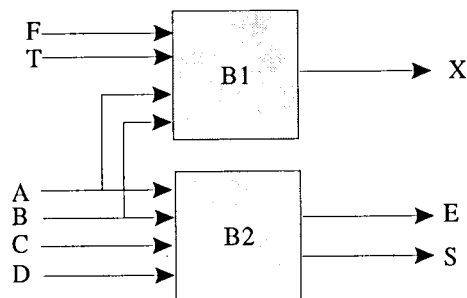


Figura 6-9. Redefinição do bloco de decisão nas células

As tabelas 6-3 a 6-5 representam as relações funcionais para estes circuitos, tomando como referência a tabela 6-2. Começaremos pelo circuito que gera os sinais E e S, passando em seguida ao circuito de retenção e selecção que gera F e T, para terminar no bloco que controla a

saída X, em função destes dois sinais. Vemos na tabela 6-3 (onde a parte superior, com $D=1$, corresponde à detecção de erro e a parte inferior, com $D=0$, define as condições de votação) que o sinal de erro é activado sempre que $A \neq B$ e que, durante a votação, o sinal de selecção indica se $C=A$ ($S=0$, erro em B) ou $C=B$ ($S=1$, erro em A).

Tabela 6-3. Tabela de verdade para o circuito que gera E e S

D	C	B	A	E	S
1	x	0	0	0	0
1	x	0	1	1	0
1	x	1	0	1	0
1	x	1	1	0	0
1	x	0	0	0	0
1	x	0	1	1	0
1	x	1	0	1	0
1	x	1	1	0	0
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	1	0
0	0	1	1	0	0
0	1	0	0	0	0
0	1	0	1	1	0
0	1	1	0	1	1
0	1	1	1	0	0

O circuito de retenção e selecção, indicado na figura 6-8, gera os sinais F e T a partir da informação existente nos sinais D, E e S, como se mostra na figura 6-10.

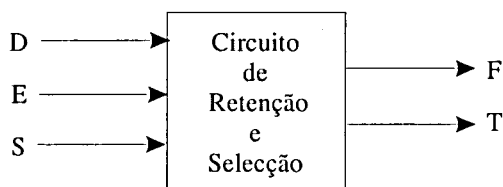


Figura 6-10. Diagrama geral do circuito de retenção e selecção

Os sinais F e T definem quatro estados, com o significado que lhes atribuímos na tabela 6-4, a que corresponde também o diagrama de estados, do circuito sequencial assíncrono que implementa o bloco de retenção e selecção, representado na figura 6-11. Partindo do estado normal (TF=00), a primeira detecção de erro activa F e leva o circuito para o estado de *confinamento* (TF=01). A partir deste estado, a votação activa o sinal T, enquanto F passa a direccionar o circuito correcto para a saída X. Por razões de segurança, não é permitido o retorno a qualquer estado, o que corresponderia a autorizar a reentrada em funcionamento de um circuito onde fora já detectado erro. A possibilidade de confinamento e votação em simultâneo foi excluída, uma vez que a probabilidade desta situação ocorrer é negligenciável e implica uma solução mais complexa, logo menos fiável.

Tabela 6-4. Informação associada aos sinais E e S

T	F	Significado
0	0	Normal, sem erro.
0	1	Detectado erro, espera votação.
1	0	X=A, erro no circuito de missão B.
1	1	X=B, erro no circuito de missão A.

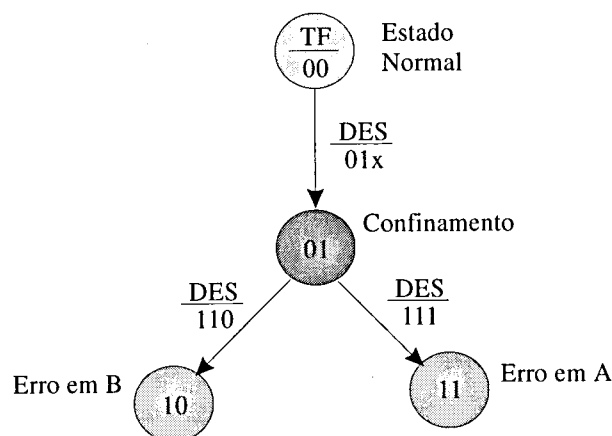


Figura 6-11. Sequência de estados do bloco de retenção

Uma questão adicional é a possibilidade de desactivar os sinais F e T, sempre que a votação encontra A=B=C, o que permitiria *reintegrar* o circuito de missão onde foi detectado erro. No entanto, esta combinação não prova que a falha tenha sido temporária e as suas implicações só podem ser avaliadas conhecendo melhor a fiabilidade da infra-estrutura BST neste regime.

Na tabela 6-5 está representada a tabela de verdade do circuito que gera o sinal X, considerando $X=0$ como o valor seguro.

Tabela 6-5. Tabela de verdade do circuito que gera o sinal X

T	F	B	A	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

As equações resultantes para os três sinais gerados pelo bloco de decisão nas células, são as indicadas a seguir:

$$X = B \cdot F \cdot T + A \cdot F \cdot T$$

$$E = A \oplus B$$

$$S = /A \cdot B \cdot C \cdot /D + A \cdot /B \cdot /C \cdot /D$$

Note-se que a equação do sinal X (a informação “útil”) conduz a um tempo de atraso equivalente a 2 portas lógicas, no percurso dos sinais A e B, que teria de existir também em qualquer circuito de confinamento exterior. No entanto, devido à existência do multiplexer de saída da célula, o atraso total pode ser otimizado para 3 portas lógicas, e não 4 como ocorreria se o sinal fosse tratado no exterior do componente. Não considerando simplificações posteriores, estas equações implicam a utilização de 7 portas lógicas, que representam uma sobrecarga equivalente a 54% (7/13) da célula original (numa implementação em FPGA).

6.3 Análise de fiabilidade da arquitectura XMR

A fiabilidade da arquitectura XMR depende de cinco componentes:

- Os dois circuitos de missão.
- A infra-estrutura CST/XMR no módulo (CI).
- O BS μ C.
- As pistas utilizadas na CCI pela infra-estrutura BST.

Do ponto de vista funcional, os dois circuitos de missão formam um paralelo e, contrariamente aos TMR, um módulo XMR pode funcionar com um só circuito de missão em modo CST, uma vez que assegura a monitorização, e que aceitaremos como solução de recurso. Este par de circuitos de missão está em série com o bloco integrado pelos três últimos componentes, que estão também em série na cadeia BST.

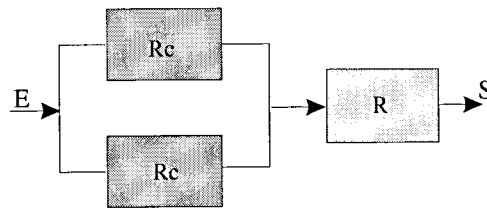


Figura 6-12. Disposição dos componentes na arquitectura XMR

Tal como para o TMR, associamos aos componentes utilizados na infra-estrutura CST/XMR os parâmetros λ (taxa de falhas) e R (fiabilidade), e a cada circuito de missão λ_c e R_c . Obtemos assim a fiabilidade da arquitectura XMR, dada por:

$$R_{XMR} = R \cdot [1 - (1 - R_c)^2] = R \cdot R_c \cdot (2 - R_c) \quad (\text{Eq. 6-2})$$

A fiabilidade R é determinada pela fiabilidade do BS μ C (R_B), pela fiabilidade das pistas (R_p) e pela da infra-estrutura CST/XMR dentro do CI (R_i), sendo calculada pela equação:

$$R = R_B \cdot R_p \cdot R_i = e^{-(\lambda_B + \lambda_p + \lambda_i)t} = e^{-\lambda \cdot t}$$

Embora não haja termos de referência para estes parâmetros, podemos considerar que:

- Dentro do circuito integrado, concluímos na secção 5.4 que a infra-estrutura CST representa geralmente menos de 8% do circuito de missão. Atendendo a que as células de decisão têm

uma sobrecarga próxima de 50% em relação às que lhe serviram de base, se admitirmos que estas células constituem, em média, menos de metade do registo e que o controlador do TAP representará cerca de metade do encargo da infra-estrutura, podemos estimar a sobrecarga de confiabilidade da infra-estrutura CST/XMR no CI como sendo não superior a $[(1+1,5)/2/2]8\% = 5\%$. Uma vez que esta lógica está na mesma pastilha do circuito de missão, a taxa de falhas respectiva será proporcional, o que nos permite considerar $\lambda_i < \lambda_c/10$.

- O BSμC necessário ao funcionamento da arquitectura XMR, onde todas as decisões são tomadas a nível das células, é basicamente um conversor paralelo-série, uma vez que a comparação das respostas serve somente para detectar *func*. Este tipo de circuito pode ser projectado confiável [Boh97] e, por outro lado, sabe-se que a memória ROM tem fiabilidade elevada [Che98]. No futuro, beneficiando de uma utilização alargada (e da depuração inerente), pode tornar-se um componente com elevado MTTF, como demonstram componentes actuais de complexidade idêntica [Cal98]. Adicionalmente, o BSμC testa-se permanentemente durante o funcionamento da infra-estrutura XMR, quando analisa o V_o recebido, e se falhar, um simples circuito mono-estável (*watchdog*) pode detectar esse estado e activar um BSμC redundante. Considerando ainda que as frequências de TCK são baixas, é lícito atribuir ao BSμC uma taxa de falhas muito inferior à da infra-estrutura: $\lambda_B \ll \lambda_i$.
- Quanto às pistas da infra-estrutura BST na CCI, na hipótese de falta simples, são testadas implicitamente em cada ciclo de acesso bem sucedido, o que permite detectar de imediato qualquer falha que nelas ocorra, sendo lícito aceitar também $\lambda_p \ll \lambda_i$.

Assim, e porque os dois últimos componentes não afectam a capacidade de confinamento da arquitectura XMR, temos $\lambda = \lambda_B + \lambda_p + \lambda_i \approx \lambda_i$, o que permite rescrever a equação 6-2, durante o período de vida útil, como:

$$R_{XMR} = e^{-(\lambda + \lambda_c) \cdot t} \cdot (2 \cdot e^{-\lambda_c \cdot t}) \quad (\text{Eq. 6-3})$$

A figura 6-13 compara a fiabilidade do circuito de missão com a das arquitecturas XMR e TMR, no qual assumimos que, para os tipos de circuito de missão envolvidos, o CVM tem também uma taxa de falhas $\lambda = 0,1 \cdot \lambda_c$. O eixo das abcissas representa unidades temporais relativas, considerando unitário o intervalo em que a fiabilidade do TMR iguala a do circuito de missão.

Este gráfico evidencia que a arquitectura XMR tem uma fiabilidade sempre superior ao TMR, em resultado de aceitarmos o funcionamento em modo CST quando falha um dos circuitos, o que alarga claramente o tempo de missão face ao circuito base.

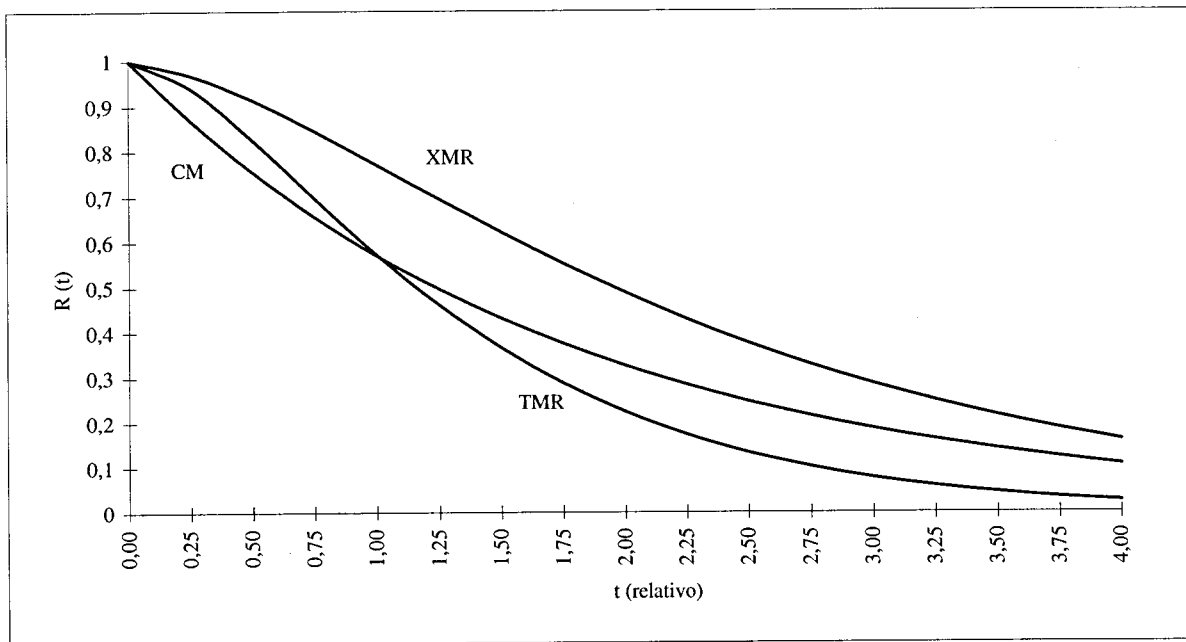


Figura 6-13. Fiabilidade do CM e das arquitecturas TMR e XMR

No sentido de ilustrar o benefício resultante do pequeno encargo desta arquitectura a nível de CI, o gráfico da figura 6-14 representa a fiabilidade *relativa* das arquitecturas TMR e XMR, face ao circuito de missão, considerando para ambas a relação entre a taxa de falhas $\lambda = 0,1 \cdot \lambda_c$ (curvas TMR.1 e XMR.1 respectivamente), bem como o comportamento da arquitectura XMR em dois casos menos favoráveis, traduzidos nas curvas:

- XMR.2: admite-se que a taxa de falhas da infra-estrutura CST/XMR dentro do componente se mantém, sendo igualado pela taxa de falhas conjunta do BS μ C e das pistas, o que corresponde a $(\lambda_B + \lambda_p) \approx \lambda_i$, de onde $\lambda = 2 \cdot \lambda_i \approx 0,2 \cdot \lambda_c$ (como $\lambda_i < 0,1 \cdot \lambda_c$).
- XMR.3: admite-se também que a taxa de falhas da infra-estrutura CST/XMR dentro do componente se mantém, e que tanto a do BS μ C como a das pistas a igualam, o que corresponde a $\lambda_B \approx \lambda_p \approx \lambda_i$, de onde $\lambda = 3 \cdot \lambda_i \approx 0,3 \cdot \lambda_c$.

É de realçar o alargamento do tempo de missão, quase quádruplo do da arquitectura TMR mesmo quando a taxa de falhas da infra-estrutura CST/XMR representa 30% da do circuito de missão. Finalmente, notemos que apesar do período de latência até recuperar, a arquitectura XMR é uma arquitectura *segura* mesmo em modo CST, uma vez que assegura a detecção de erro. Certamente que, em aplicações críticas, o funcionamento em modo CST deve ser entendido como uma forma de recurso, que não confina o erro.

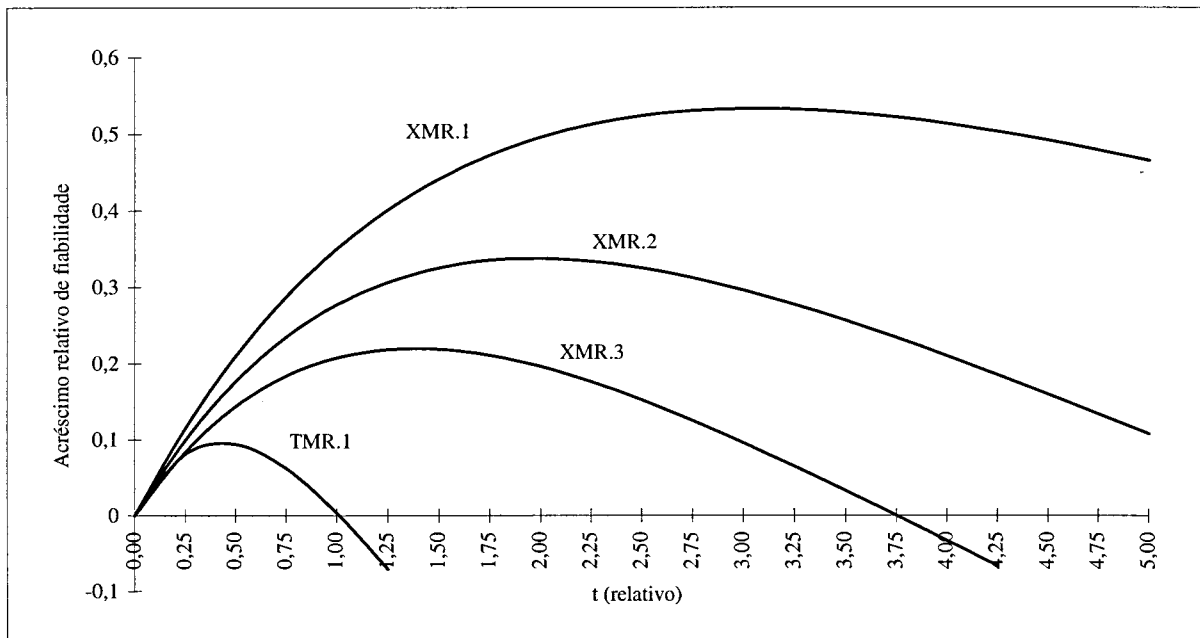


Figura 6-14. Acréscimo da fiabilidade das arquitecturas TMR e XMR, relativa ao CM

O anexo B contém o projecto de um circuito integrado com arquitectura XMR, onde o circuito ISCAS'85 C17 foi alterado de forma a permitir a injeção de faltas. Os diagramas de simulação incluídos comprovam o princípio de funcionamento descrito para esta arquitectura e permitem ilustrar a sua aplicação a um caso concreto, ainda que simples para permitir a inspecção do comportamento.

6.4 Extensão da arquitectura XMR à configuração 2XMR

Nas aplicações mais críticas, a aviação em especial, a generalidade dos sistemas de controlo são pelo menos triplicados e cada canal é geralmente implementado separadamente dos outros, por razões de segurança, como referimos no capítulo três. Um circuito integrado XMR pode substituir uma solução TMR em muitas aplicações, mas sendo um CI único, não possibilita a disposição independente das *réplicas*. Na realidade, a especificação da arquitectura XMR prevê um circuito integrado tolerante a falhas físicas, único, sendo o aspecto agora referido colateral, mas interessa-nos ainda assim explorar a utilização destes circuitos integrados nas aplicações indicadas.

A configuração 2XMR é composta por dois circuitos independentes, como se ilustra na figura 6-15, cada um formado por componentes XMR cujas saídas se reúnem num CI independente que contém um grupo de células de votação [San98e]. Esta configuração requer que as infraestruturas CST/XMR dos três componentes sejam alimentadas em paralelo, para que a votação ocorra em três pontos independentes: as saídas dos dois CIs XMR e a união das saídas. As características mais relevantes que apresenta a configuração 2XMR são:

- Dois canais independentes, cada um com comportamento XMR.
- Enquanto existirem dois circuitos de missão sem falhas (o que permite a avaria dos outros dois, independentemente do módulo onde ocorram), não é necessário confinamento, nem existe intervalo de latência na saída, dado que o componente de decisão final dispõe das informações de erro E1 e E2.
- Sobrecarga apenas 20% superior à de um TMR (cerca de $210 \cdot 2 / 350$).
- Duas cablagens, em lugar de três, o que tende a compensar a sobrecarga anterior.

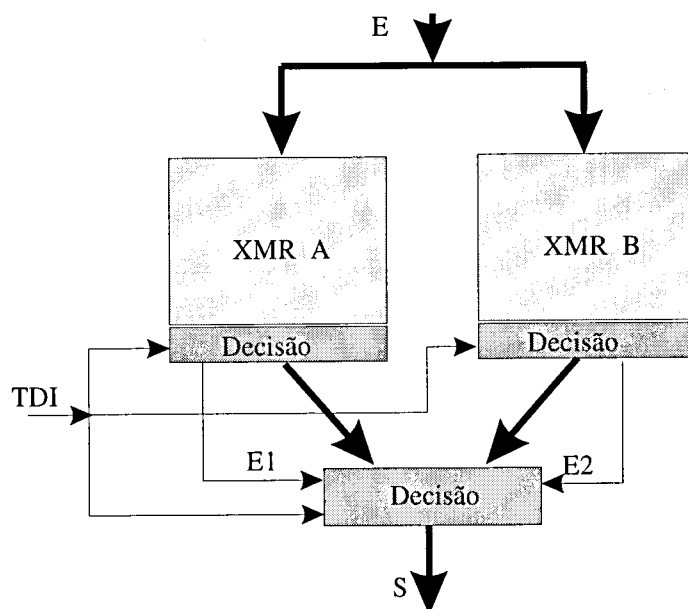


Figura 6-15. A configuração 2XMR

Note-se que a falha simultânea dos dois canais é uma possibilidade que pode ser eliminada na prática [Roq98], o que torna esta arquitectura promissora em aplicações críticas.

6.5 Conclusão

A par das aplicações tradicionais de tolerância a falhas, que justificam o custo do melhor projecto possível e todo o esforço em o depurar, temos um vasto leque de novas aplicações, que não permitem projectos de elevada redundância. As limitações vão desde o custo do sistema ao volume ocupado, passando pelo consumo de energia e nalguns casos pelas grandes quantidades de produção, incompatíveis com algumas arquitecturas tradicionais.

Estas aplicações necessitam de CIs que tolerem falhas permanentes e, na maioria dos casos, será suficiente colocar as saídas num estado reconhecido como seguro, durante o intervalo de latência. Do ponto de vista das saídas do módulo, confinar e recuperar serão aceitáveis se os circuitos por elas alimentados forem combinatórios. Dado estarmos a considerar o projecto de componentes VLSI, que englobam virtualmente toda a lógica de controlo, é de aceitar que as respectivas saídas alimentem actuadores, quase sempre combinatórios.

Convém ainda referir que a causa mais provável de *fmc* em funcionamento (temporárias) é o ruído induzido pela tensão de alimentação. Por esta razão, pensamos que no futuro os CIs vocacionados para esta arquitectura deverão possuir pinos de alimentação independentes para cada circuito de missão, o que não será problemático, dada a existência de múltiplos pinos de alimentação nos componentes actuais.

Por fim, é importante realçar que se aproveita o teste inerente ao funcionamento da infra-estrutura BST em regime concorrente, para verificar as células de decisão. Esta operação ocorre sempre que $A=B=C$ e envolve a lógica interna da infra-estrutura, assegurando-lhe o teste para faltas simples e evitando desenhar células em lógica auto-verificável, com vantagens na sua fiabilidade intrínseca (por serem mais simples).

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

Capítulo 6

Capítulo 7

Capítulo 8

Bibliografia

Anexos

Fiabilidade da Informação na Infra-estrutura IEEE 1149.1

7. Fiabilidade da Informação na Infra-estrutura IEEE 1149.1

À medida que a infra-estrutura BST é proposta para um número crescente de aplicações em serviço (calendarizado ou concorrente), a fiabilidade da informação que transmite tem de ser melhorada. Esta responsabilização da infra-estrutura BST é nova, porque a norma 1149.1 foi projectada para teste fora de serviço em que, detectado um erro, há tempo de repetir o teste para determinar onde ocorreu, se houver interesse nisso. Por esta razão, a infra-estrutura nunca teve mecanismos para garantir a segurança da informação veiculada, mas agora, utilizada em regime concorrente, este tipo de protecções é primordial. Neste capítulo começaremos por abordar este problema, integrando depois a solução que encontrámos no projecto de uma FPGA Altera 7128 com arquitectura XMR.

7.1 *A Fiabilidade de transmissão na infra-estrutura BST*

A infra-estrutura CST permite monitorar o circuito e possibilita uma tolerância a falhas discreta no modo de sobrevivência, duas situações que dependem da informação fornecida através da infra-estrutura 1149.1. Na arquitectura XMR, por seu lado, são tomadas decisões dentro do circuito integrado e a localização do circuito de missão errado depende também da informação que a infra-estrutura BST fornece, para além de que a decisão de restaurar o funcionamento do módulo é crítica. Adicionalmente, a detecção de *fmc* depende da mesma informação, pelo que temos múltiplas razões para procurar evitar que ocorra um erro (originado pela transmissão na infra-estrutura BST) que não é fatal (crítico), quando o sistema principal (formado pelos dois circuitos de missão) funciona correctamente [San99a].

Existem três motivos principais para prevermos a possibilidade de ocorrerem erros na utilização da infra-estrutura BST em modo concorrente:

- O funcionamento simultâneo do circuito de missão e da infra-estrutura CST pode induzir ruído mútuo.
- A tendência constante para o aumento das frequências.
- A redução da tensão de alimentação e das consequentes margens de ruído, que se tornarão particularmente críticas abaixo de 2,5V. *gerando ruído e falhas - caso de*

Qualquer destes motivos justifica a abordagem que nos propomos efectuar mas, no entanto, conhecemos apenas duas propostas dedicadas à fiabilidade da informação transmitida pela infra-estrutura BST 1149.1, já referidas no capítulo quatro. Em [McH90] é analisada a utilização de paridade, associada a sinais de interrupção, para garantir a fiabilidade da informação deslocada, principalmente de instruções, consideradas a fase mais crítica do processo de varrimento. Nesta abordagem, representada na figura 7-1, é utilizada uma nova linha associada a mecanismos de interrupção (TINT*), que requer pinos adicionais, limitativos de uma utilização generalizada.

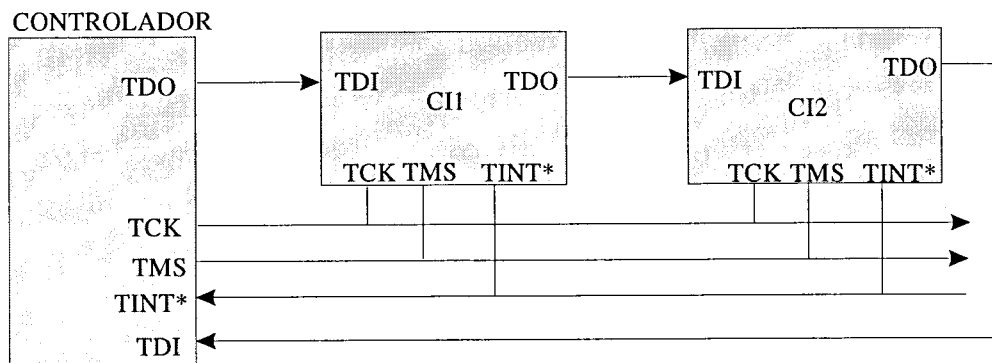


Figura 7-1. Utilização de interrupções

Esta abordagem foi posteriormente utilizada em alguns componentes [Tex99], onde é presumida a utilização de instruções com paridade sempre par e, se tal não ocorrer (se a instrução carregada tiver paridade ímpar), é activado um bit dedicado do registo de instrução (denominado IRERR). No entanto, o erro (se existiu) só será detectado no acesso seguinte ao registo de instrução, independentemente das consequências que possa entretanto gerar, o que é inaceitável em regime concorrente. No caso da infra-estrutura CST e da arquitectura XMR, uma vez que não são deslocadas instruções em regime normal, esta solução não serve os objectivos desejados.

Na outra proposta [DeH95], os autores questionam a fiabilidade de um só acesso físico aos diversos blocos funcionais, num contexto de tolerância a falhas, e sugerem uma matriz de ligações para a infra-estrutura BST, que permite aceder a cada bloco funcional através de mais do que um TAP e registo. Esta solução, que se representa na figura 7-2 para um acesso de redundância dois, melhora a fiabilidade da infra-estrutura mas não a da informação que é transmitida e consideramos a sobrecarga excessiva para as vantagens que permite no teste concorrente.

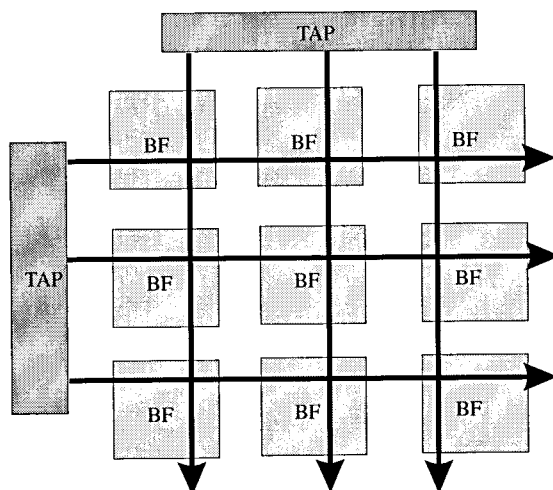


Figura 7-2. Matriz de acessos aos blocos funcionais

A análise que efectuámos levou-nos a concluir que há dois processos satisfatórios de aumentar a fiabilidade da transmissão na infra-estrutura 1149.1:

- Uma linha de controlo adicional.
- Codificar a informação transmitida.

A primeira opção foi abandonada para evitar pinos extra, em favor da codificação, para a qual há três factores a ter em conta:

- Complexidade do circuito de codificação e decodificação.
- Velocidade de transmissão (afectada pelo número de bits extra).
- Taxa de erros na transmissão.

Os dois primeiros factores apontam a *paridade* como o processo mais simples de implementar, calculável independentemente no emissor e no receptor, e permitindo confrontar resultados. Alternativas como contagem de transições, assinatura ou síndrome, implicam processos mais elaborados, tanto no circuito envolvido como na quantidade de informação a transmitir.

Quanto à taxa de erros, sabemos que a probabilidade de erros múltiplos no funcionamento normal é extremamente baixa, negligenciável mesmo na indústria aeroespacial [Roq98]. Nos casos que temos vindo a considerar, a probabilidade de uma falha dessas ocorrer em simultâneo com uma decisão na arquitectura XMR é, por maioria de razões, irrelevante, excepto se resultar de ruído na tensão de alimentação, mas este caso requer circuitos de monitorização específicos para V_{cc} , podendo ser evitado com alimentações independentes, como referimos no capítulo seis. É de notar que, tanto na infra-estrutura CST como na arquitectura XMR, após a inicialização, não há instruções deslocadas para o TAP durante o funcionamento das soluções propostas, interessando assegurar a fiabilidade da transmissão de *dados*. O mecanismo que propomos neste capítulo calcula a paridade no emissor e no destinatário, permitindo ao BSμC comparar os dois resultados após o último bit enviado, como se mostra na figura 7-3.

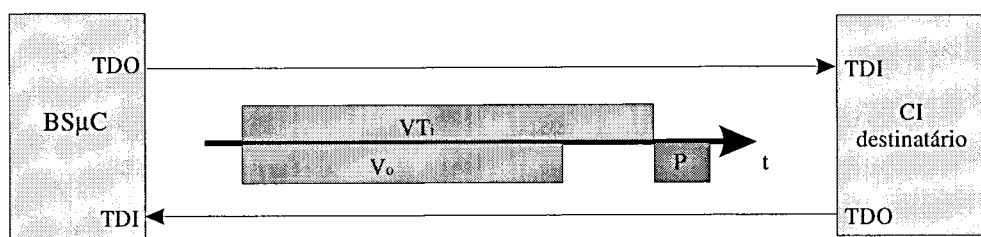


Figura 7-3. Sequência temporal de dados e do bit de paridade

7.1.1 Processos para determinar a Paridade

O processo de cálculo da paridade pode ser sequencial (P_s) ou paralelo (P_p), como indicamos na Figura 7.4. A paridade série é calculada permanentemente à medida que a informação é deslocada, uma solução em que o circuito de cálculo necessita apenas de três portas lógicas e de um FF, encargo semelhante ao de uma célula de observabilidade (*vide* figura 10.29 no anexo C). No segundo caso, a paridade é gerada a partir do conteúdo final das células de cada registo, o que permite obter a máxima fiabilidade, mas implica um circuito gerador de paridade para cada registo, à qual se acrescenta a sobrecarga das ligações inerentes a esta variante.

Embora o processo de cálculo sequencial detecte apenas os erros externos ao circuito integrado (a montante do pino TDI), pensamos ser esta a maior causa potencial de erros na transmissão, pelo que no seguimento deste trabalho será este o mecanismo que utilizaremos, em virtude da

sua simplicidade. Independentemente do processo utilizado para determinar a paridade, todos os outros resultados são idênticos. Notemos ainda, que em ambos os casos o resultado (P) será disponibilizado ao mesmo tempo, que corresponde à passagem pelo estado Exit-1 como veremos de seguida.

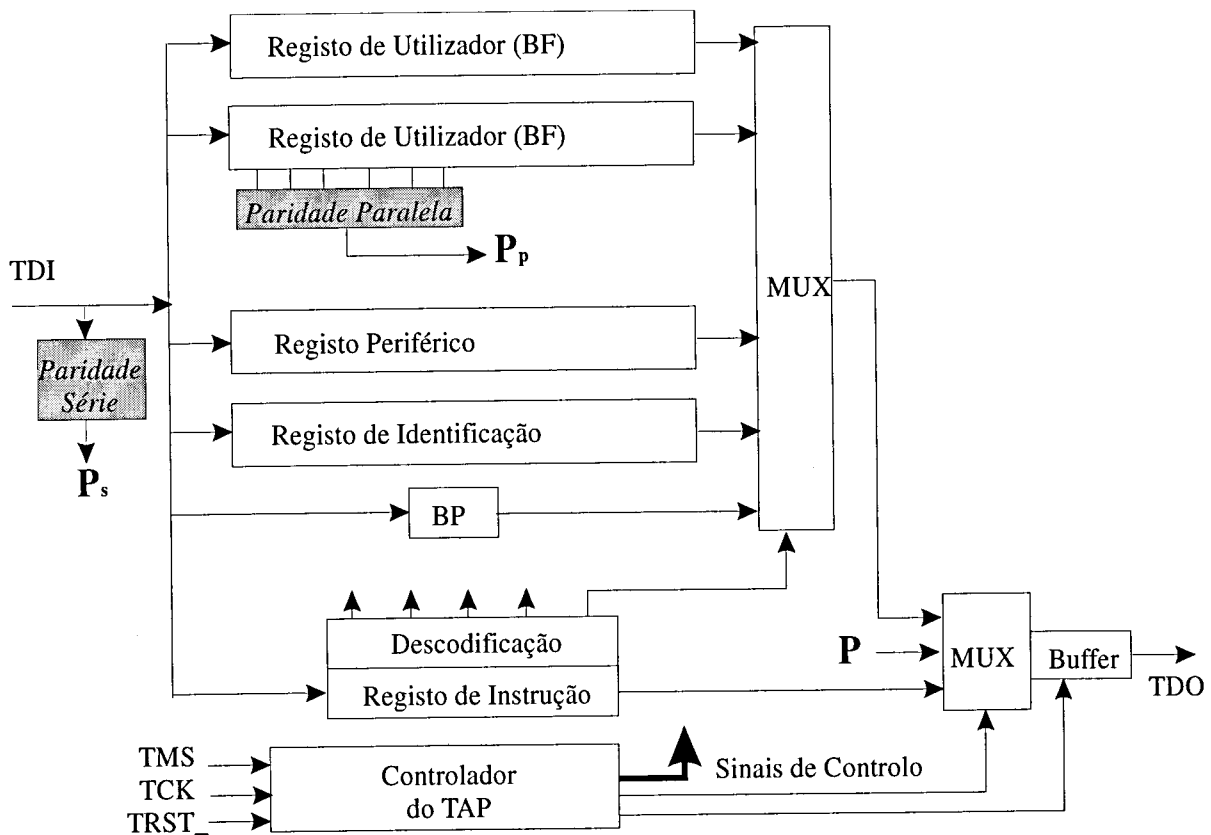


Figura 7-4. Geração de paridade em série e em paralelo

7.1.2 Análise do diagrama de estados do TAP

Dispondo da paridade da informação deslocada, tanto no BS μ C emissor como no circuito integrado receptor, resta compará-los para aferir da eventual ocorrência de erros na transmissão através da infra-estrutura BST. Ao analisar o diagrama de estados do controlador do TAP, reproduzido parcialmente na figura 7-5, vemos que a transmissão:

- É iniciada com a passagem do estado Capture para Shift.
- Termina com a passagem do estado Shift para Exit-1.
- Após a descida de TCK, no estado Exit-1, a saída TDO permanece em alta impedância.

A sequência descrita, válida tanto para os ciclos de dados como de instrução, permite fornecer o bit de paridade em TDO, à saída do estado Exit-1, acção que não é compatível com a norma, por não impor o estado de alta impedância em TDO, nestas circunstâncias. No entanto, como não é possível passar a outro registo sem progredir para o estado Update, em condições normais a alteração proposta não será detectável pelos circuitos compatíveis com a norma [San99b].

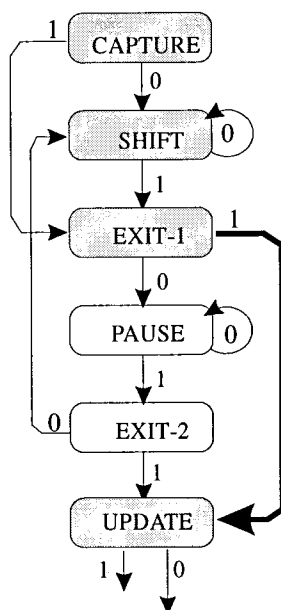


Figura 7-5. Controlo de paridade com transição de Exit-1 para Update

No caso apresentado acima, o controlador do TAP passa ao estado Update e actualiza a informação nas células do receptor, mesmo se estiver errada, o que implicará repetir de novo o ciclo. É no entanto possível evitar esta situação se o controlador do TAP for levado a passar pelos estado Pause e Exit-2, como se ilustra na figura 7-6.

Neste caso, TDO fornece também o bit de paridade no estado Exit-1, permitindo suspender o controlador no estado Pause já com TDO em alta impedância, como a norma prevê para este estado. O interesse desta variante é que, em caso de erro no deslocamento, o controlador pode regressar ao estado Shift para repetir o envio da informação, num processo recursivo que evita carregar informação errada no andar de retenção das células.

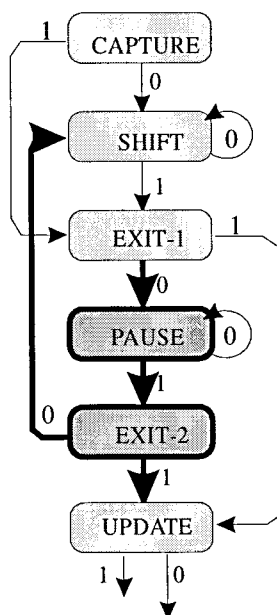


Figura 7-6. Controlo de paridade com transição via Pause e Exit-2

Notemos que em nenhuma das opções foram inseridos estados adicionais no controlador do TAP e só na segunda variante são acrescentados dois períodos de TCK [San99c]. Caberá ao projectista optar pela simplicidade da primeira solução ou pela maior segurança da segunda, que no entanto inclui dois ciclos TMS desnecessários se a informação estiver correcta, como é de esperar na maioria dos casos.

Finalmente, é importante realçar que a determinação de paridade foi apresentada para a informação que circula no sentido $BS\mu C \rightarrow CI$, por ser a mais crítica nas propostas CST e XMR, uma vez que as decisões ocorrem essencialmente no CI, mas pode ser também aplicada ao sinal de retorno $CI \rightarrow BS\mu C$, se este percurso for considerado mais crítico. É ainda possível disponibilizar os dois bits de paridade (um correspondente à informação que entrou por TDI e outro à que saiu por TDO, do ponto de vista de um CI), mas neste caso só o modo de controlo indicado na figura 7-6 permite deslocar o segundo bit de paridade, com segurança, no estado Pause.

Estamos finalmente em condições para apresentar uma versão integrada das soluções propostas, que o leitor pode encontrar no anexo C, simulada para implementação numa FPGA Altera EMP7128S-10. Recomendamos também a consulta dos anexos A e B, que contêm informação complementar, útil para uma melhor compreensão dos esquemas e diagramas apresentados naquele anexo.

7.2 *Conclusão*

Neste capítulo apresentámos uma solução para aumentar a fiabilidade da informação veiculada pela infra-estrutura BST, tanto na versão base como nas que são utilizadas para a infra-estrutura CST e na arquitectura XMR. Utilizando apenas as quatro linhas obrigatórias, esta solução permite ao BS μ C ler a paridade calculada no CI destinatário, durante o ciclo TCK subsequente ao último bit transmitido, sem afectar o funcionamento e de um modo transparente. A sobrecarga física inerente à proposta é diminuta, enquanto a sobrecarga dinâmica é:

i) Nula, no caso do controlo em conformidade com a figura 7-5.

ii) Dois períodos de TCK, no caso da opção de controlo correspondente à figura 7-6.

Pensamos que as vantagens resultantes de ter acesso ao bit de paridade, em conformidade com a proposta apresentada, ultrapassam largamente a incompatibilidade formal com a norma 1149.1.

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

Capítulo 6

Capítulo 7

Capítulo 8

Bibliografia

Anexos

Conclusão

8. Conclusão

Este trabalho teve como motivação a procura de soluções para os problemas com que o utilizador de componentes VLSI (e também de circuitos menos complexos) se depara actualmente, a nível de confiabilidade. Só recentemente os CIs atingiram um número de portas lógicas que permite a integração comercial de circuitos complexos, e mesmo de sistemas, num único integrado (SOC, *System on a Chip*). Como tal, temos actualmente condições de proporcionar sistemas de controlo a custos impensáveis há uma década, que permitem encontrar continuamente novas aplicações. Em termos médios, as exigências de confiabilidade a que estes sistemas estão sujeitos têm vindo a crescer, fruto de múltiplos factores, entre os quais destacamos dois, que consideramos particularmente relevantes:

- Aplicações que envolvem a segurança de pessoas.
- Redução das geometrias de integração, que afecta a fiabilidade dos componentes, actualmente próxima do limiar aceitável, mesmo em sistemas de utilização corrente.

Por seu lado, o teste calendarizado, tradicionalmente utilizado em aplicações críticas, tem limitações que são particularmente sensíveis nos componentes VLSI, devido à tipologia de falhas que podem afectar estes componentes e à limitada acessibilidade que apresentam. Esta situação gera um interesse acrescido pela monitorização concorrente e, no nosso caso, pela possibilidade de aproveitar a infra-estrutura BST como suporte a este tipo de processos.

Começamos por notar que esta infra-estrutura é uma ferramenta cada vez mais universal, utilizada em diferentes técnicas de teste vectorial, sobretudo acima de uma determinada complexidade. São raros os novos componentes VLSI que não dispõe de infra-estrutura 1149.1, a que se junta o facto de a crescente reutilização de módulos e de núcleos IP implicar a utilização de registos de varrimento internos. Adicionalmente, na generalidade das aplicações

mais relevantes que requerem confiabilidade, mesmo onde a infra-estrutura BST não é utilizada em serviço, é necessário recorrer a um BSμC residente, seja para controlar os testes iniciais ou os calendarizados, ou para a conversão entre o barramento a nível de sistema (1149.5 ou outro) e a infra-estrutura 1149.1.

Partindo desta base, a evolução para a infra-estrutura CST representa uma sobrecarga muito pequena, que podemos estimar inferior a 1% num componente médio, e como tal sem repercussões apreciáveis no custo dos componentes (onde esteja prevista a infra-estrutura BST), nem no seu tempo de vida útil. O utilizador pode assim providenciar uma monitorização transparente, a um custo muito baixo, útil em todas as aplicações que não exijam confinamento de erro.

A arquitectura XMR resulta de uma simbiose entre uma arquitectura de redundância-2 e a infra-estrutura CST, que lhe permite ultrapassar a principal limitação: a falta de informação para decisões que permitam a recuperação. Confrontada com arquitecturas tradicionais, a arquitectura XMR permite o menor encargo num componente, sendo válida em todas as aplicações que não impliquem mascarar o erro. Finalmente, o controlo de paridade da informação veiculada pela infra-estrutura BST, permite um acréscimo importante de confiabilidade, a custo simbólico, mas essencial nas aplicações preconizadas.

8.1 *Metodologia de implementação das propostas apresentadas*

Nesta secção pretendemos descrever a metodologia necessária para implementar as propostas apresentadas neste documento, que começa pelo projecto do circuito de missão e cujas etapas se esquematizam na figura 8-1. A fase de projecto dá origem a um ficheiro esquemático, do tipo gráfico ou contendo uma descrição em alto nível, de que são exemplos o VHDL [Ash90, Ska95] e o AHDL [Sal97, Alt99]. Após a necessária depuração, por simulação do modelo, deverá ser gerado um ficheiro compatível com a ferramenta de análise de testabilidade e geração do teste, eventualmente em formato EDIF ou VHDL [Par90]. Para além de produzir um relatório das características de testabilidade do projecto, esta ferramenta gera ainda os vectores de teste e identifica os nós (pontos de teste) onde será necessário inserir células BST, um processo a completar por ferramentas específicas, que produzem geralmente um ficheiro em linguagens descritivas, como BSDL (*Boundary Scan Description Language*). No caso do *MAX+PLUS II*, que usamos para validar as propostas apresentadas nesta dissertação, todos os passos são

transparentes para utilizador, desde o projecto à programação, com excepção da análise de testabilidade.

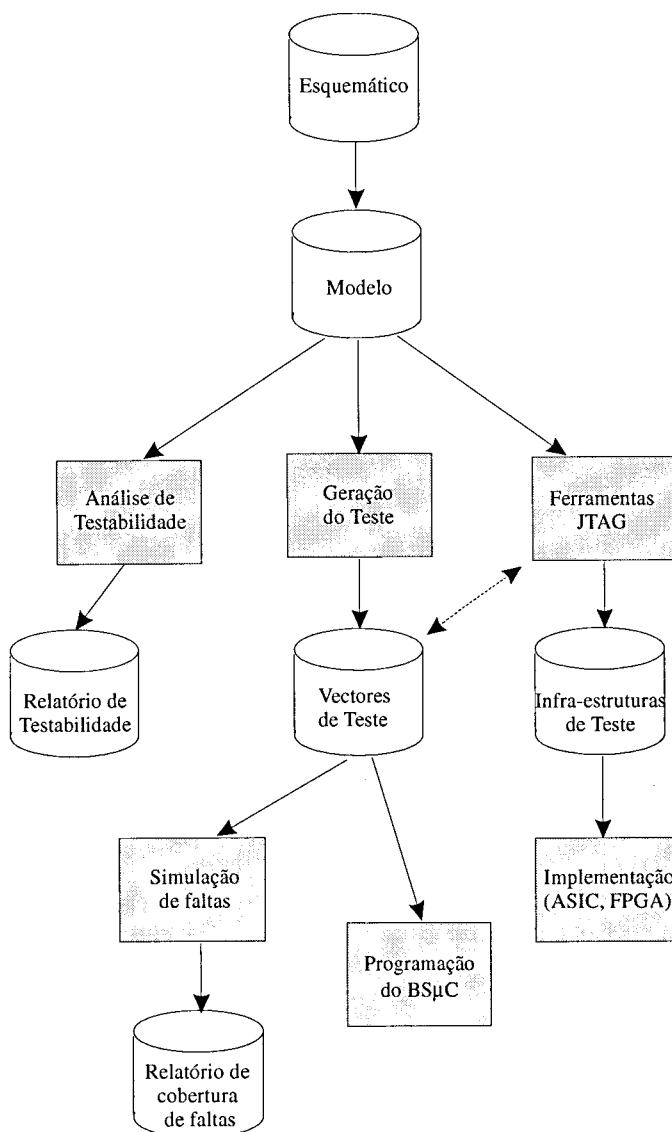


Figura 8-1. Etapas da metodologia de implementação

Identificados convenientemente os registos BST/CST que são utilizados, o conjunto de vetores de teste deve ser programado no BSμC. Na verificação experimental que efectuámos no âmbito deste trabalho, esta etapa teve lugar através da programação de um PIC16C84, a que recorremos para emular o BSμC. Enquanto não forem desenvolvidos os BSμC a que faremos referência na próxima secção, o projectista pode em alternativa utilizar os controladores referidos em [Fer92b, Mont98] ou controladores comerciais dos principais fabricantes (Texas Instruments e Siemens, por exemplo), podendo também recorrer a aplicações executáveis em PC [Fer95].

8.2 Comentários sobre o BS μ C necessário para monitorização

O BS μ C definido em [Fer92b] pode ser utilizado no controlo da infra-estrutura CST, mas é possível simplificá-lo (com vantagem na respectiva fiabilidade) para satisfazer somente as seguintes características:

- PINOS

- Obrigatórios: 8 - GND, V_{cc} , CLK, RST, TCK, TDI, TDO, TMS.
- Opcionais: até 15 - número de sinais TMS adicionais a controlar independentemente, em função do número de componentes no sistema, que supomos ser no máximo de 16.

- MEMÓRIA

- RAM - 32 bytes (2 bytes por registo e até 16 registos)
- ROM - 2 a 16 Kbytes, considerando um máximo de 250 vectores de teste por registo e vectores de 16 bits no máximo, correspondendo a dois bytes por cada um dos dois vectores de teste (VT_i e VT_o). Daqui resulta $250 \cdot 2 \cdot 16 \approx 16$ Kbytes.

- INSTRUÇÕES

Entre as instruções utilizadas pelo controlador que temos vindo a referir, as que são necessárias na monitorização são:

- TMS 0/1: esta instrução básica poderá ser substituída por outras de mais alto nível, que simplifiquem a programação. Dois exemplos serão as macro-instruções *GOTOBF n* , que partindo do estado Run-Test/Idle leva o controlador do TAP até ao estado Capture-DR do BF especificado, bem como *END* para terminar o ciclo a partir do estado EXIT-1.
- SELTMS n : esta instrução selecciona o sinal TMS activo, se existir mais do que um.
- LD CNT: instrução que carrega o contador interno com o número de bits a deslocar. Esta instrução é dispensável se o contador deslocar sempre um número fixo de bits, programável por saltos (16, 24 ou 32), podendo também ser automatizada com base em apontadores para cada registo, programados na ROM.
- CMPTB: compara V_o com VT_o na tabela em ROM, usando V_i como endereço. É semelhante à instrução NSHFCP definida em [Fer92b] e que desloca um vector e compara a resposta obtida, mas não requer vector de máscara.

As restantes instruções e características do controlador descrito em [Fer92b] são dispensáveis no funcionamento que propusemos, pelo que o BS μ C pode ser optimizado (tarefa que temos em

vista num futuro próximo), vindo a tornar-se barato e confiável. A família de microcontroladores Microchip PIC16Cxx é um exemplo de complexidade muito semelhante ao BSμC necessário para apoiar a infra-estrutura CST, com um custo inferior a 2 Euro em versão OTP, que devido à simplicidade inerente a uma arquitectura RISC (16 instruções apenas) permite componentes fiáveis, actualmente a ser utilizados em aplicações desenvolvidas para a indústria espacial [Cal98].

Notemos ainda que o BSμC pode ser integrado no CI que supervisiona, tal como em [Pet97], mas do ponto de vista da fiabilidade é preferível que seja externo. A memória ROM, aceite como fiável na indústria espacial [Che98], pode ainda ser testada ciclicamente por controlo de soma.

8.3 *Direcções futuras*

As propostas apresentadas neste documento abrem múltiplas áreas que é necessário investigar e, no futuro imediato, temos em vista as seguintes:

- Desenvolvimento (ou adaptação) de ferramentas integradas, tanto para geração de testes como para a inserção de registos CST, optimizadas para a especificidade do funcionamento concorrente.
- Projecto do BSμC, nas duas vertentes que indicámos: uma orientada para os requisitos do CST e outra, ainda mais simples, destinada a apoiar apenas os componentes com arquitectura XMR.
- A arquitectura XMR pode ser adaptada aos circuitos sequenciais através da monitorização das entradas e saídas do circuito, como descrito. Os elementos de memória, por seu lado, farão parte de um registo separado, com células BST redesenhadas para conter dois FF no andar de retenção, permanentemente comparados. Desta forma, este registo só será acedido quando haja desacordo, o que acelera o tratamento dos circuitos sequenciais, e permite utilizar um processo de votação idêntico ao descrito no capítulo seis.
- A infra-estrutura CST pode ser alterada para procurar múltiplas combinações de VT_i em simultâneo [San97c]. A questão que se coloca consiste em saber se existe um vector de teste que, com um determinado número das suas combinações (16, 32, 64, 128 ou mais), permita obter uma cobertura de faltas especificada? Temos um interesse

particular nesta questão, uma vez que a existência de uma resposta positiva conduzirá a implicações relevantes nas propostas apresentadas, nomeadamente à possibilidade de prescindir do BS μ C e assegurar o auto-teste concorrente em circuitos isolados, e possivelmente mesmo a componentes XMR autónomos.

Surge assim uma nova área de monitorização, que merece ser explorada e justifica o investimento em novas direcções de investigação, para diminuir as restrições que condicionam este processo, essencialmente pelo facto de ser não intrusivo. Pela importância crescente que as questões de confiabilidade têm vindo a assumir no domínio dos sistemas baseados em microelectrónica, estamos certos que estas e outras direcções de trabalho serão progressivamente concretizadas, no futuro a curto prazo.

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

Capítulo 6

Capítulo 7

Capítulo 8

Bibliografía

Anexos

9. Referências citadas no texto

- A**
- Abr 84 M. Abramovici, "Critical Path Tracing: An alternative to Fault Simulation", *IEEE Design and Test of Computers*, Vol-1, N-1, pp. 83-93, 1984.
- Abr 85 M. Abramovici, P. R. Menon, "A Practical Approach to Fault Simulation and Test Generation for Bridging Faults", *IEEE Transactions on Computers*, Vol-C-34, N-7, pp. 658-663, 1985.
- Abr 90 M. Abramovici, M. A. Breuer, A. D. Friedman, *Digital Systems Testing and Testable Design*, Computer Science Press, Oxford, 1990, ISBN 0-7167-8179-4.
- Abr 98 M. Abramovici, S. Wijessuriya, C. Hamilton, C. Stroud, "Built-In Self-Test for Field Programmable Gate Array Interconnect", *Compendium of Papers, 4th IEEE International On-Line Testing Workshop*, pp. 37-41, 1998.
- Abra 98 J. Abraham, J. Maier, H-D Kochs, M. Nicolaidis, M. Renovel, "From Dependable Computing Systems to Computing for Integrated Dependable Systems", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 296-301, 1998.
- Abre 97 M. d'Abreu, A. Stokes, L. Sassoon, "Defect Analysis - Impact on Yield Improvement and to the Design of Manufacturable ICs", *Compendium of Papers, IEEE European Test Workshop*, S- 4, 1997.
- Abre 98 M. d'Abreu, P. Isakanian, S. Hunjan, "Understanding of the Fabrication Process - Key to Design and Test of Mixed Signal Integrated Circuits", *Compendium of Papers, IEEE European Test Workshop*, pp. 156-159, 1998.
- Abs 96 *Automatic Boundary Scan Insertion Tool*, disponível no endereço: <http://euclid.ascinc.com/products/absit>
- Add 93 *Addendum 1993*, disponível no endereço: <ftp://ftp.mcnc.org/pub/benchmarks/iscas89/addendum93/testpatterns/>
- Agr 78 V. D. Agrawal, "When to Use Random Testing", *IEEE Transactions on Computers*, Vol-C-27, N-11, pp. 1054-1055, 1978.

- Agr 81 V. D. Agrawal, "An Information Theoretic Approach to Digital Testing", *IEEE Transactions on Computers*, Vol-C-30, N-8, pp. 582-587, 1981.
- Agr 88 V. D. Agrawal, S. C. Seth, *Test Generation for VLSI Chips: Tutorial*, IEEE Computer Society, 1988, ISBN 0-8186-8786-X.
- All 98 *Allegro-Studio*, disponível no endereço: http://www.cadence.com/software/LVD/Allegro-Studio_oct98.htm
- Alt 99 Altera Corporation, disponível no endereço: <http://www.altera.com>
- Alv 97 J. C. Alves, A. Puga, L. Corte-Real, J. S. Matos, "FPGA Implementation of a Vector Processor for the Estimation of High-Order Moments", *Proceedings of the Design of Circuits and Integrated Systems Conference*, pp. 759-763, 1997.
- Amb 98 *Builgates*, disponível no endereço: <http://www.ambit.com/products/>
- And 81 T. Anderson, P. A. Lee, *Fault Tolerant Principles and Practice*, Prentice/Hall International, 1981, ISBN 0-13-308254-7.
- Arm 93 J. R. Armstrong, "Hierarchical Test Generation: Where We Are, and Where We Should Be Going", *IEEE Euro Design and Automation Conference*, pp. 434-439, 1993.
- Asa 96 H. Al-Asaad, J. P. Hayes, B. T. Murray, "Design of Scaleable Hardware Test Generators for On-Line BIST", *Compendium of Papers, 2nd IEEE International On-Line Testing Workshop*, pp. 164-167, 1996.
- Ash 90 P. J. Ashenden, *The VHDL Cookbook*, Department of Computer Science, University of Adelaide, South Australia, 1990.
- Avi 75 A. Avizienis, "Architecture of Fault-Tolerant Computing", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 3-16, 1975.
- Avi 78 A. Avizienis, "Fault-Tolerance: the survival attribute of digital systems", *Proceedings of the IEEE*, Vol-66, N-10, pp. 1109-1125, 1978.
- Avi 87 A. Avizienis, H. Kopetz, J. C. Laprie, *The Evolution of Fault-Tolerant Computing*, Springer-Verlag, Wien, 1987, ISBN 0-387-81941-X.
- Avi 98 A. Avizienis, Y. He, "The Taxonomy of Design Faults in COTS Microprocessors", *IEEE Fault Tolerant Computing Symposium, Digest of FastAbstracts*, pp. 52-53, 1998.

B

- Bal 91 L. Balado, J. Figueras, A. Rubio, "Self Checking Circuits in Presence of Bridging Faults. Possibilities of Current Sensing in XOR Parity Trees", *Proceedings of the 2nd IEEE European Test Conference*, p. 523, 1991.

- Bar 87 P. H. Bardell, W. H. McAnney, J. Savir, *Built-In Test for VLSI, Pseudo-random Techniques*, John Wiley & Sons, 1987, ISBN 0-471-62463-2.
- Bar 90 P. H. Bardell, "Design Considerations for Parallel Pseudorandom Pattern Generators", *Journal of Electronic Testing: Theory and Applications*, N-1, pp. 73-87, 1990.
- Bar 92 A. E. Barbour, "Solutions to the Minimization Problem of Fault-Tolerant Circuits", *IEEE Transactions on Computers*, Vol-41, N-4, pp. 429-443, 1992.
- Ben 91 R. G. Bennetts, A. Osseyran, "IEEE Std. 1149.1-1990 on BS: History, Literature Survey, and Current Status", *Journal of Electronic Testing: Theory and Applications*, N-2, pp. 11-25, 1991.
- Ben 99 *Benchmarks ISCAS 1999*, disponível no endereço: <http://www.cbl.nscu.edu/>
- Ber 96 H. Berviller, A. Dandache, F. Monteiro, B. Lepley, "Using FPGAs for the Implementation of Fail-Safe Interfaces", *Compendium of Papers, 2nd IEEE International On-Line Testing Workshop*, pp. 53-55, 1996.
- Bha 89 D. Bhattacharya, B. T. Murray, J. P. Hayes, "High-Level Test Generation for VLSI", *IEEE Computer*, pp. 16-24, 1989.
- Bha 91 D. Bhavsar, "A Proposal to Extend the IEEE Normalized 1149.1 TAP to System Backplanes", *Proceedings of the 2nd IEEE European Test Conference*, p. 531, 1991.
- Bog 95 A. Bogliolo, M. Damiani, "Synthesis of Multilevel Fault-Tolerant Combinational Circuits", *IEEE Euro Design and Automation Conference*, pp. 80-85, 1995.
- Boh 97 E. Bohl, R. Stephan, W. Glauert, "The Architecture of the Fail-Stop Controller AE11", *Compendium of Papers, 3rd IEEE International On-Line Testing Workshop*, pp. 47-52, 1997.
- Bom 98 M. Bombana, C. Cavalloro, F. Ferrandi, F. Fummi, D. Sciuto, "Implicit Testability Techniques for VHDL Based ASIC Design", *Compendium of Papers, IEEE European Test Workshop*, pp. 133-134, 1998.
- Brg 84 F. Brglez, "On Testability Analysis of Combinational Networks", *Proceedings of the International Symposium on Circuits and Systems*, pp. 221-225, 1984.
- Brg 85 F. Brglez, H. Fujiwara, "A Neutral List of 10 Combinational Benchmark Circuits and a Target Translator in FORTRAN", *Proceedings of the International Symposium on Circuits and Systems*, 1985. Disponível também no endereço: <ftp://ftp.mcnc.org/pub/benchmarks/iscas85/>

- Brg 89 F. Brglez, D. Bryan, K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits", *Proceedings of the International Symposium on Circuits and Systems*, pp. 1929-1934, 1989. Disponível também no endereço: <ftp://ftp.mcnc.org/pub/benchmarks/iscas89/>
- Bro 91 J. Brown, "Implementing a Parallel/Serial Converter for BST", *Proceedings of the 2nd IEEE European Test Conference*, p. 530, 1991.
- Bur 47 A. W. Burks, "Electronic Computing Circuits of the ENIAC", *Proceedings of the I. R. E.*, Vol-35, N-8, pp. 756-767, 1947. Reeditado em *Proceedings of the IEEE* Vol-85, N-7, pp. 1169-1173, 1997.

C

- Cad 98 *Affirma VHDL*, disponível no endereço: <http://www.cadence.com/software/LVD/affirmaNCVHDL>
- Cal 98 D. W. Caldwell, D. A. Rennels, "FTSM: A Fault-Tolerant Spaceborne Microcontroller", *IEEE Fault Tolerant Computing Symposium, Digest of Fast-Abstracts*, pp. 3-4, 1998.
- Cali 98 T. Calin, L. Anghel, M. Nicolaidis, "Asynchronous Current Monitors for Fault Detection in Deep Submicron CMOS", *Compendium of Papers, 4th IEEE International On-Line Testing Workshop*, pp. 56-60, 1998.
- Car 87 W. C. Carter, "Experiences in Fault Tolerant Computing, 1947-1971", em A. Avizienis, H. Kopetz, J. C. Laprie (editores), *The Evolution of F-T Computing*, Springer Verlag Editors, pp. 1-36, 1987.
- Cel 96 F. Celeiro, L. Dias, J. Ferreira, M. B. Santos, J. P. Teixeira, "Defect-Oriented IC Test and Diagnosis Using VHDL Fault Simulation", *Proceedings of the IEEE International Test Conference*, pp. 620-628, 1996.
- Cha 86 S. Chang, "Structured Functional Level Test Generation Using BDDs", *Proceedings of the IEEE International Test Conference*, pp. 97-104, 1986.
- Cha 94 S. Chau, "Fault Injection Boundary Scan Design for Verification of F-T Systems", *Proceedings of the IEEE International Test Conference*, pp. 677-682, 1994.
- Cha 97 T. J. Chakraborty, "On-line Test Method Using Boundary-Scan", *Compendium of Papers, 3rd IEEE International On-Line Testing Workshop*, pp. 156-159, 1997.
- Che 89 K-T Cheng, V. D. Agrawal, "An Economical Scan Design for Sequential Logic Test Generation", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 28-35, 1989.

- Che 90 W-T Cheng, M-L Yu, "Differential Fault Simulation for Sequential Circuits", *Journal of Electronic Testing: Theory and Applications*, N-1, pp. 7-13, 1990.
- Che 93 K-T Cheng, "Redundancy Removal for Sequential Circuits Without Reset States", *IEEE Transactions on Computer-Aided Design of ICs and Systems*, pp. 13-24, Vol-12, N-1, 1993.
- Che 98 P. Cheynet, R. Velasco, R. Ecoffet, S. Buchner, "Flight Results Analysis of Digital Experiment Devoted to Satellite Image Processing by Means of Neural Nets", *Compendium of Papers, 4th IEEE International On-Line Testing Workshop*, pp. 23-27, 1998.
- Chen 93 Y-Y Cheng, S. J. Updhyaya, "Yield Analysis of Reconfigurable Array Processors Based on Multiple-Level Redundancy", *IEEE Transactions on Computers*, pp. 1136-1141, Vol-C-42, N-9, 1993.
- Chi 97 S. Chiusano, F. Corno, P. Prinetto, M. S. Reorda, G. Squillero, "C2BIST, A New BIST Architecture", *Compendium of Papers, IEEE European Test Workshop*, S- 5, 1997.
- Chi 98 A. V. Chichkov, C. B. Almeida, "A Tool for Mapping a C Description into a Hardware Configuration using VHDL at RTL", *Proceedings of the Design of Circuits and Integrated Systems Conference*, pp. 502-506, 1998.
- Cic 91 B. Ciciani, "Redundancy Effect on Yield of Binary Tree RAMs", *Journal of Electronic Testing: Theory and Applications*, N-2, pp. 293-306, 1991.
- Cle 98 F. Clermidy, T. Collette, M. Nicolaidis, "On-Line periodic Test Strategy dedicated to Parallel Computers", *Compendium of Papers, 4th IEEE International On-Line Testing Workshop*, pp. 197-201, 1998.
- Cor 97 F. Corno, P. Prinetto, M. Rebaudengo, M. S. Reorda, G. Squillero, "Is 9-Valued Logic Really Necessary for Sequential ATPG?", *Compendium of Papers, IEEE European Test Workshop*, PS-2-2, 1997.
- Cou 81 B. Courtois, "Failure Mechanisms, Fault Hypothesis and Analytical Testing of LSI-NMOS Circuits", *IEEE VLSI Conference*, pp. 341-350, 1981.
- Cro 89 A. L. Crouch, C. Pyron, "Impact of JTAG/1149.1 Testability on Reliability", *Government Microcircuit Applications Conference*, pp. 83-90, 1989.
- Cyp 99 Cypress Corporation, disponível no endereço: <http://www.cypress.com>

D

- Dae 91 W. Daehn, "Fault Simulation Using Small Fault Samples", *Journal of Electronic Testing: Theory and Applications*, N-2, pp. 191-203, 1991.

- Dag 98 J. M. Daga, M. Rezzoug, D. Auvergne, "Extension of Comprehensive Submicron Delay Modeling to Generalized Loads", *Proceedings of the Design of Circuits and Integrated Systems Conference*, pp. 556-561, 1998.
- Dav 98 R. David, *Random Testing of Digital Circuits*, Marcel Dekker, Inc., N. Y., 1998, ISBN 0-8247-0182-8.
- DeH 95 A. DeHon, "Scan-Based Testability for Fault-Tolerant Architectures", M. I. T. Transit Project, Transit Note #60. Disponível no endereço: <http://www.ai.mit.edu/projects/transit/tn60>
- Der 91 B. I. Dervisoglu, "Features of a Scan and Clock Resource Chip for Providing Access to Board-Level Test Functions", *Journal of Electronic Testing: Theory and Applications*, N-2, pp. 107-115, 1991.
- Dev 90 S. Devadas, H-HT Ma, A. R. Newton, "Redundancies and Don't Cares in Sequential Logic Synthesis", *Journal of Electronic Testing: Theory and Applications*, N-1, pp. 15-30, 1990.
- Dia 98 O. P. Dias, I. Teixeira, J. P. Teixeira, "Metrics for Quality Assessment of Testable HW/SW Systems Architectures", *Compendium of Papers, IEEE European Test Workshop*, pp. 205-209, 1998.
- Doo 98 R. Doorneweert (JTAG Technologies) "Automatic BST Insertion in ASICs", *Compendium of Papers, IEEE European Test Workshop, Industrial Presentations*, p. 129, 1998.
- Dua 97 R. O. Duarte, I. A. Noufal, M. Nicolaidis, "A CAD Framework for Efficient Self-Checking Data Path Design", *Compendium of Papers, 3rd IEEE International On-Line Testing Workshop*, pp. 28-35, 1997.

F

- Fav 91 M. Favalli, P. Olivo, B. Riccò, F. Somenzi, "Fault Simulation for General FC MOS ICs", *Journal of Electronic Testing: Theory and Applications*, N-2, pp. 181-190, 1991.
- Fer 92a J. M. M. Ferreira, *O Teste de Cartas de Circuito Impresso com BST: Arquitetura de um Controlador Residente e Geração Automática do Programa de Teste*, Tese de Doutorado, Faculdade de Engenharia da Universidade do Porto, Abril de 1992.
- Fer 92b J.M. Ferreira, F.S. Pinto, J.S. Matos, "A Modular Architecture for Board-Level BIST of BS Boards", *Proceedings of the IEEE EuroASIC Conference*, pp.184-188, 1992.
- Fer 95 L. H. Ferreira, "Implementação de dois TAP's à custa da porta paralela (Centronics) dum PC", *Publicação interna da Faculdade de Engenharia da Universidade do Porto*, 1995.

- Fer 98 A. Ferré, J. Figueras, "Estimating Defect-Free I_{DDQ} from Noisy Data", *Compendium of Papers, IEEE European Test Workshop*, pp. 199-200, 1998.
- Flo 98a P. F. Flores, H. C. Neto, K. Chacrabarty, J. P. M. Silva, "A Model and Algorithm for Computing Minimum-Size Test Patterns", *Compendium of Papers, IEEE European Test Workshop*, pp. 147-148, 1998.
- Flo 98b P. F. Flores, H. C. Neto, J. P. M. Silva, "An Exact Solution to the Minimum-Size Test Pattern Problem", *Proceedings of the IEEE International Conference on CAD*, 1998.
- Fri 73 A. D. Friedman, "Easily Testable Iterative Systems", *IEEE Transactions on Computers*, Vol-C-22, N-12, pp. 1061-1064, 1973.
- Fri 74 A. D. Friedman, "Diagnosis of Short-Circuit Faults in Combinational Logic Circuits", *IEEE Transactions on Computers*, Vol-C-23, N-7, pp. 746-752, 1974.
- Fuj 83 H. Fujiwara, T. Shimono, "On the Acceleration of Test Generation Algorithms", *IEEE Transactions on Computers*, Vol-C-32, N-12, pp. 1137-1144, 1983.
- Fuj 85 H. Fujiwara, *Logic Testing and Design for Testability*, The MIT Press, Cambridge, MA, 1985, ISBN 0-262-06096-5.

G

- Gar 95 H. de Garis, "Evolvable Hardware Report", disponível em <http://www.cwi.nl/~ehrenbur/EvolvableHardware/WorkshopReport.html>
- Gho 91 S. Ghosh, T. J. Chakraborty, "On Behavior Fault Modeling for Digital Designs", *Journal of Electronic Testing: Theory and Applications*, N-2, pp. 135-151, 1991.
- Glo 93 C. Gloster, "Dynamic Scan Testing: A New Paradigm", disponível no endereço: <http://www.cbl.nscu.edu/publications/1993-Thesis-PhD-Gloster>
- Glo 95 C. Gloster, F. Brglez, "Partial Scan Selection for User-Specific Fault Coverage", disponível no endereço: <http://www.cbl.nscu.edu/publications/1995-EuroDAC-Gloster>
- Goe 81 P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Circuits", *IEEE Transactions on Computers*, Vol-C-30, N-3, pp. 215-222, 1981.
- Gol 79 L. H. Goldstein, "Controllability/Observability Analysis of Digital Circuits", *IEEE Transactions On Circuits and Systems*, Vol-CAS-26, N-9, pp. 685-693, 1979.
- Goo 90 A. J. Goor, O. Jansen, "Self Test for the Intel 8085", *IEEE Microprocessing and Microprogramming* 29, pp. 165-175, 1990.

- Gra 98 E. Gramatova, A. Somorovska, J. Gaspar, H. Manhaeve, "Test Pattern Generation Systems for I_{DDQ} -Voltage Test Experiments", *Compendium of Papers, IEEE European Test Workshop*, pp. 193-194, 1998.
- Gun 91 H. H. S. Gundlach, B. Koch, K-D. Muller-Glaser, "On the Selection of a Partial Scan Path with Respect to Target Faults", *Proceedings of the 2nd IEEE European Test Conference*, p. 482, 1991.

H

- Ham 98a I. Hamzaoglu, J. H. Patel, "Compact Two-Pattern Test Set Generation for Combinational and Full Scan Circuits", *Proceedings of the IEEE International Test Conference*, 1998.
- Ham 98b I. Hamzaoglu, J. H. Patel, "Test Set Compaction Algorithms for Combinational Circuits", *Proceedings of the IEEE International Conference on CAD*, pp. 283-289, 1998.
- Haw 98 C. Hawkins, A. Keshavarzi, K. Roy, "High Performance CMOS IC Challenges in I_{DDQ} Testing", *Compendium of Papers, IEEE European Test Workshop*, pp. 167-170, 1998.
- Hay 85 J. P. Hayes, "Fault Modeling: Design and Test Tutorial", *IEEE Design and Test of Computers*, Vol-2, N-4, pp. 88-95, 1985.
- Hay 87 J. P. Hayes, "An Introduction to Switch-Level Modeling", *IEEE Design and Test of Computers*, Vol-4, N-4, pp. 18-25, 1987.
- Hei 98 G. Heiner, T. Thurner, "Time-Triggered Architecture for Safety-Related Distributed Real-Time Systems in Transportation Systems", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 402-407, 1998.
- Hel 95 S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, B. Courtois, "Built-In Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers", *IEEE Transactions on Computers*, Vol-44, N 2, pp. 223-232, 1995.
- Her 98 A. Hertwig, H-J Wunderlich, "Low Power Serial Built-In Self-Test", *Compendium of Papers, IEEE European Test Workshop*, pp. 49-53, 1998.
- Hir 91 K. Hirabayashi, "Self-Checking CMOS Circuits Using Pass-Transistor Logic", *Journal of Electronic Testing: Theory and Applications*, N-2, pp. 205-208, 1991.
- Hop 78 A. L. Hopkins, T. Smith, J. H. Lala, "FTMP-A Highly Reliable F-T Multiprocessor for Aircraft", *Proceedings of the IEEE*, pp. 1221-1239, 1978.
- Hsi 99 M. S. Hsiao, E. M. Rudnick, J. H. Patel, "Fast Static Compaction Algorithm for Sequential Circuit Test Vectors", *IEEE Transactions on Computers*, Vol-48, N-3, pp. 311-322, 1999.

I

- IEEE 93 IEEE Standard 1149.1a, *IEEE Standard Test Access Port and Boundary-Scan Architecture*, IEEE Inc, NY, 1993.
- IEEE 95 IEEE Standard 1149.5, *Standard for Module Test and Maintenance (MTM) Bus Protocol*, IEEE Inc, TTTC, NY, 1995.
- IEEE 99a IEEE Standard 1149.4 Mixed Signal Working Group, *P1149.4 Standard for a Mixed-Signal Test Bus*, Draft 25, IEEE Inc, TTTC, NY, Fevereiro 1999.
- IEEE 99b IEEE Standard P1500, *Standards for Embedded Core Test*, disponível no endereço: <http://grouper.ieee.org/groups/1500/>
- IFI 99 IFIP workgroup, disponível no endereço: <http://www.dependability.org>
- Ism 90 A. A. Ismael, M. A. Breuer, "The Probability of Error Detection in Sequential Circuits Using Random Test Vectors", *Journal of Electronic Testing: Theory and Applications*, N-1, pp. 245-256, 1990.
- Iva 88 A. Ivanov, V. K. Agarwal, "An Iterative Technique for Calculating the Aliasing Probability of Linear Feedback Signature Registers", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 70-75, 1988.
- Iye 95 R. K. Iyer, K. S. Trivedi, J. Goldberg, "Introduction to Special Issue on F-T Computing", *IEEE Transactions on Computers*, Vol-44, N-2, pp. 165-167, 1995.

J

- Jai 85 K. Jain e V. D. Agrawal, "Statistical Fault Analysis", *IEEE Design and Test of Computers*, Vol-2, N-1, pp. 38-44, 1985.
- Jan 85 I. Jansch, B. Courtois, "Strongly Language Disjoint Checkers", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 390-395, 1985.
- Jha 93 N. K. Jha, "Fault Detection in CVS Parity Trees with Application to Strongly Self-Checking Parity Two-Rail Checkers", *IEEE Transactions on Computers*, Vol-42, N-2, 1993.
- Joh 89 B. W. Johnson, *Design and Analysis of Fault Tolerant Digital Systems*, Addison-Wesley Publishing Company, 1989, ISBN 0-201-07570-9.
- Jon 91a F. Jong, J. S. Matos, J. M. Ferreira, "Boundary Scan Test, Test Methodology, and Fault Modeling", *Journal of Electronic Testing: Theory and Applications*, N-2, pp. 77-88, 1991.
- Jon 91b F. Jong, F. Heyden, "Testing the Integrity of the Boundary Scan Test Infrastructure", *Proceedings of the IEEE International Test Conference*, pp. 106-112, 1991.

JTA 98 JTAG Technologies, disponível no endereço: <http://www.jtag.com>

K

- Kar 93 M. G. Karpovsky, S. M. Chaudhry, "Design of Self-Diagnostic Boards by Multiple Signature Analysis", *IEEE Transactions on Computers*, Vol-42, N-9, pp. 1035-1044, 1993.
- Karr 93 R. Karri, A. Orailoglu, "High-Level Synthesis of Fault-Secure Microarchitectures" *30th Association of Computing Machinery/IEEE Design Automation Conference*, pp. 429-433, 1993.
- Kei 97 M. Keim, B. Becker, "Nearly Exact Signal Probabilities for Synchronous Sequential Circuits", *Compendium of Papers, IEEE European Test Workshop, PS-2.1*, 1997.
- Kei 98 M. Keim, N. Dreschsler, R. Dreschsler, B. Becker, "Combining GAs and Symbolic Methods for High Quality Tests of Sequential Circuits", *Compendium of Papers, IEEE European Test Workshop*, pp. 141-142, 1998.
- Kie 98 G. Kiefer, H-J Wunderlich, "Deterministic BIST with Multiple Scan Chains", *Compendium of Papers, IEEE European Test Workshop*, pp. 39-43, 1998.
- Kim 82 C. Kime, K. K. Saluja, "Test Scheduling In Testable VLSI Circuits", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 406-411, 1982.
- Kim 91 K. Kim, J. G. Trong, D. S. Ha, "BIDES: A BIST Design Expert System", *Journal of Electronic Testing: Theory and Applications*, N-2, pp. 165-179, 1991.
- Kim 98a H. Kim, J. P. Hayes, "Realization-Independent Testing of IP-Based Systems", *IEEE Fault Tolerant Computing Symposium, Digest of FastAbstracts*, pp. 58-59, 1998.
- Kim 98b H. Kim, A. White, K. Shim, "Reliability Modeling of Real-Time Systems", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 304-313, 1998.
- Koe 79 B. Koenemann, J. Mucha, G. Zwiehoff, "Built-in Logic Block Observation Techniques", *IEEE Transactions on Computers*, pp. 37-41, 1979.
- Koe 98 B. Koenemann, "Creature From the Deep Sub-Micron Lagoon", *Compendium of Papers, IEEE European Test Workshop*, p. 1, 1998.
- Kor 90 I. Koren, A. D. Singh, "Fault-Tolerance in VLSI Circuits", *IEEE Computer*, pp. 73-82, 1990.
- Kun 97 C. Kuntzsch, F. Mayer, K. Ronge, "A Novel Approach for an On-Line Self-Test Architecture using ASIC Circuits in a Multi-Channel System", *Compendium of Papers, 3rd IEEE International On-Line Testing Workshop*, pp. 165-168, 1997.

- Kuz 96 R. Kuznar, "Partitioning and Optimization of Digital Integrated Circuits", disponível no endereço: <http://www.cbl.nscu.edu/publications/1996-Thesis-PhD-Kuznar>

L

- Lal 85 P. K. Lala, *Fault Tolerant & Fault Testable Hardware Design*, Prentice/Hall International, 1985, ISBN 0-13-308248-2.
- Lal 94 J. H. Lala, R. E. Harper, "Architectural Principles for Safety-Critical Real-Time Applications", *Proceedings of the IEEE*, Vol-82, N-1, pp. 25-40, 1994.
- Lan 92 J. A. C. Landeck, *Um Controlador Robusto para Processos Industriais*, Tese de Mestrado, Universidade de Coimbra, 1992.
- Lap 85 J-C. Laprie, "Dependable Computing and Fault Tolerance: Concepts and Terminology", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 2-11, 1985.
- Leo 98 Leonardo package, disponível no endereço: <http://www.exemplar.com>
- Lev 82 Y. Levendel, P. R. Menon, "Test Generation Algorithms for Computer Hardware Description Languages", *IEEE Transactions on Computers*, Vol-C-31, N-7, pp. 577-588, 1982.
- Lev 94 S-T. Levi, A. K. Agrawala, *Fault Tolerant System Design*, McGraw-Hill, Inc, 1994, ISBN 0-07-037515-1.
- Lia 96 Y. Liao, D. Walker, "Fault Coverage Analysis for Physically Based CMOS Bridging Faults at Different Power Supply Voltages", *Proceedings of the IEEE International Test Conference*, pp. 767-775, 1996.
- Lie 91 J-C Lien, M. A. Breuer, "An Optimal Scheduling Algorithm for Testing Interconnects Using BS", *Journal of Electronic Testing: Theory and Applications*, N-2, pp. 117-130, 1991.
- Lit 91 I. P. Litikov, "Ring-Like Testing of Digital Circuits", *Journal of Electronic Testing: Theory and Applications*, N-1, pp. 301-304, 1991.
- Liu 94 J. W. Liu, W-K. Shih, K-J. Lin, R. Bettati, J-Y. Chung, "Imprecise Computations", *Proceedings of the IEEE*, pp. 83-94, 1994.
- Lo 88 J-C Lo, S. Thanawastien, "The Design of TSC Berger Code Checkers Based on Code Partitioning", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 226-231, 1988.
- Lo 92 J-C Lo, J. C. Daly, M. Nicolaidis, "Design of Static CMOS Self-Checking Circuits using Built-In Current Sensing", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 104-110, 1992.

- Lof 96 K. Lofstrom, "Early Capture for Boundary Scan Timing Measurements", *Proceedings of the IEEE International Test Conference*, pp. 417-422, 1996.
- Lub 92 M. Lubaszewski, B. Courtois, "On the Design of Self-Checking Boundary Scannable Boards", *Proceedings of the International IEEE Test Conference*, pp. 372-381, 1992.
- Lub 94 M. Lubaszewski, *Le Test Unifié de Cartes Appliqué à la Conception de Systemes Fiabiles*, Thèse Docteur Microelectronique, TIMA, INP de Grenoble, Junho de 1994.

M

- Mad 94 H. Madeira, J. G. Silva, "Experimental Evaluation of the Fail-Silent Behavior in Computers Without Error Masking", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 350-359, 1994.
- Mai 90 J. Maierhofer, "Hierarchical Self-test Concept Based on the JTAG Standard", *Proceedings of the IEEE International Test Conference*, pp. 127-133, 1990.
- Mar 91 D. M. Marcynuk, D. M. Miller, "The OR-k Method for On-Line Checking of Programmable Logic Arrays", *Journal of Electronic Testing: Theory and Applications*, N-3, pp. 53-65, 1991.
- Mar 97 W. Marouf, M. Marzouki, "Specification and Early Evaluation of System Testability Through Object Oriented Modeling", *Compendium of Papers, IEEE European Test Workshop, PS-1. 1*, 1997.
- Mari 91 E. J. Marinissen, R. Decker, "Minimization of Test Control Blocks", *Proceedings of the 2nd IEEE European Test Conference*, p. 427, 1991.
- Mart 97 C. Marta, V. Szekely, M. Rencz, B. Courtois, "Self-Checking Current Output and BSc-connected Frequency Output Temperature Sensor for DfTT", *Compendium of Papers, 3rd International On-Line Testing Workshop*, pp. 75-78, 1997.
- Mat 98 J. S. Matos, J. M. Silva, "Mixed-Signal Board level DFT Techniques Using P1149.4", *Proceedings of the International Conference on Electronics, Circuits and Systems*, pp. 441-446, 1998.
- Mau 90 C. Maunder, R. E. Tulloss (editores), *The Test Access Port and Boundary-Scan Architecture*, IEEE, 1990, ISBN 0-8186-9070-4.
- Mau 91 C. M. Maunder, R. E. Tulloss, "An Introduction to the BS Standard: ANSI/IEEE Std 1149.1", *Journal of Electronic Testing: Theory and Applications*, N-2, pp. 27-42, 1991.
- Max 87 R. A. Maxion, D. P. Siewiorek and S. Elkind, "Techniques and Architectures for F-T Computing", *Annals of Revue of Computer Science*, N-2, pp. 469-520, 1987.

- McC 81 E. J. McCluskey, S. Bozorgui-Nesbat, "Design for Autonomous Test", *IEEE Transactions on Computers*, Vol-C-30, N-11, pp. 866-875, 1981.
- McC 84 E. J. McCkluskey, "Verification Testing: A Pseudo Exhaustive Technique", *IEEE Transactions on Computers*, Vol-C-33, N-6, pp. 541-547, 1984.
- McH 90 P. F. McHugh e L. Whetsel, "Adding Parity and Interrupts to IEEE Std. 1149.1", em C. M. Maunder, R. E. Tulloss (editores), *The Test Access Port and Boundary-Scan Architecture*, IEEE Society Press, pp. 205-213, 1990, ISBN 0-8186-9070-4.
- Med 98 J. V-Medina, M. Lubaszewski, M. Nicolaidis, "On-Line Testing of Opamp Circuits Using Built-In Detector Observer", *Compendium of Papers, 4th IEEE International On-Line Testing Workshop*, pp. 187-191, 1998.
- Mee 97 M Meerwein, T, Lindenkreus, W. Glauert, "A Dynamic I_{DDQ} Monitor Cell with Sel-Test for the Fail-Stop Controller AE11", *Compendium of Papers, 3rd IEEE International On-Line Testing Workshop*, pp. 58-62, 1997.
- Men 98 Monet package, disponível no endereço: <http://www.mentorgraphics/monet/>
- Mer 97 M. Mercer, R. Kapur, T. Williams, "Failure Prediction Quality for Voltage versus I_{DDQ} testing", *Compendium of Papers, IEEE European Test Workshop, S-4.3*, 1997.
- Met 97 C. Metra, M. Favalli, B. Ricco, "Novel Single Output 1-out-of-3 Code Checker", *Compendium of Papers, 3rd IEEE International On-Line Testing Workshop*, pp. 228-232, 1997.
- Mil 65 Norma MIL-HDBK-217, disponível no endereço: http://www.diamond.spawar.navy.mil/specs/mil_std/MIL_HDBK_217.html
- Mil 98 E. Millan, L. Entrena, J. Espejo, S. Chiusano, F. Corno, "Using BDD's to Improve Structural Optimization for Sequential Circuits", *Proceedings of the Design of Circuits and Integrated Systems Conference*, pp. 284-289, 1998.
- Miu 98 Y. Miura, H. Yamazaki, "A Built-in Sensor Using Multiple Power Supplies", *Compendium of Papers, IEEE European Test Workshop*, pp. 176-180, 1998.
- Mon 98 F. Monteiro, A. Dandache, H. Berviller, B. Lepley, "A Secure Interface Based on BIST and Hybrid Coding Techniques", *Compendium of Papers, 4th IEEE International On-Line Testing Workshop*, pp. 142-145, 1998.
- Mont 98 P. M. C. C. Monteiro, *Uma Biblioteca VHDL para Controladores BST*, Tese de Mestrado, Faculdade de Engenharia da Universidade do Porto, Setembro, 1998.
- Mos 98 V. Moshanin, V. Ocheretnij, A. Dmitriev, "The Impact of Logic Optimization on Concurrent Error Detection", *Compendium of Papers, 4th IEEE International On-Line Testing Workshop*, pp. 81-85, 1998.

- Mur 88 B. T. Murray, J. P. Hayes, "Hierarchical Test Generation Using Precomputed Tests for Modules", *Proceedings of the IEEE International Test Conference*, pp. 221-229, 1988.
- Mur 91 B. T. Murray, J. P. Hayes, "Test Propagation Through Modules and Circuits", *Proceedings of the IEEE International Test Conference*, pp. 748-757, 1991.
- Mut 76 P. Muth, "A Nine-Valued Circuit Model for Test Generation", *IEEE Transactions on Computers*, Vol-C-25, N-6, pp. 630-636, 1976.

N

- Nar 93 S. Narayanan, R. Gupta, M. A. Breuer, "Optimal Configuring of Multiple Scan Chains", *IEEE Transactions on Computers*, Vol-42, N-9, pp. 1121-1131, 1993.
- NAS 97 *Mars Pathfinder Cruise Data Replay*, disponível no endereço: <http://mpfwww.nasa.gov/mpf/realtime/nf-cruisepictures.html>
- Nel 87 V. P. Nelson, B. D. Carroll, *Fault-Tolerant Computing: Tutorial*, IEEE Computer Society, 1987, ISBN 0-8186-0677-0.
- Nic 88 M. Nicolaidis, "A Unified Built-In-Self-Test Scheme: UBIST", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 157-163, 1988.
- Nic 89 M. Nicolaidis, S. Noraz, B. Courtois, "A Generalized Theory of Fail-Safe Systems", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 398-406, 1989.
- Nic 91 M. Nicolaidis, "Shorts in Self-Checking Circuits", *IEEE Journal of Electronic Testing: Theory and Applications*, N-1, pp. 257-273, 1991.
- Nic 97 M. Nicolaidis, T. Calin, "A Theory of Perturbation Tolerant Asynchronous FSMs and its Application on the Design of Perturbation Tolerant Memories", *Compendium of Papers, IEEE European Test Workshop*, S-5.3, 1997.
- Nic 98a M. Nicolaidis, "Fail-Safe Interfaces for VLSI: Theoretical Foundations and Implementation", *IEEE Transactions on Computers*, Vol-47, N-1, pp. 62-77, 1998.
- Nic 98b M. Nicolaidis, "Scaling Deeper to Submicron: On-Line Testing to the Rescue", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 299-301, 1998.
- Nic 98c M. Nicolaidis, Y. Zorian, D. K. Pradhan, editors, *On-Line Testing for VLSI*, Kluwer Academic Publishers, 1998, ISBN 0-7923-8132-7.

O

- Ora 96 A. Orailoglu, R. Karri, "Automatic Synthesis of Self-Recovering VLSI Systems", *IEEE Transactions on Computers*, Vol-45, N-2, pp. 131-142, 1996.

P

- Par 75 K. P. Parker, E. J. McCluskey, "Probabilistic Treatment of General Combinational Networks", *IEEE Transactions on Computers*, Vol-C-24, N-6, pp. 668-670, 1975.
- Par 90 K. P. Parker, *Integrating Design and Test Using CAE Tools for ATE Programming*, IEEE Computer Society Press, 1990, ISBN 0-8186-8788-6.
- Par 92 K. P. Parker, *The Boundary-Scan Handbook*, Kluwer Academic Press, 1992, ISBN 0-7923-9270-1.
- Pet 97 K. Petersen, B. Magnhagen, "A DFT Architecture for Total BIST of Mega-gate Designs", *Compendium of Papers, IEEE European Test Workshop*, PS-3.4, 1997.
- Phi 96 K. W. Philp, *The Analysis and Synthesis of Dependable Digital Systems*, Ph. D. Thesis, Robert Gordon University, Escócia, 1996.
- PLD 98 *Turbo-Tester*, disponível no endereço: <http://www.pld.ttu.ee>
- Pom 91 I. Pomeranz, L. Reddy, S. M. Reddy, "COMPACTEST: a Method to Generate Compact Test Sets for Combinational Circuits", *Proceedings of the IEEE International Test Conference*, pp. 194-203, 1991.
- Pom 92 I. Pomeranz, S. M. Reddy, "A Divide-and-Conquer Approach to TG for Large Synchronous Circuits", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 230-237, 1992.
- Pom 93 I. Pomeranz, S. M. Reddy, "Classification of Faults in Synchronous Sequential Circuits", *IEEE Transactions on Computers*, Vol-42, N 9, pp. 1066-1077, 1993.
- Pom 97 I. Pomeranz, S. M. Reddy, "On Dictionary-Based Fault Location in Digital Logic Circuits", *IEEE Transactions on Computers*, Vol-46, N-1, pp. 48-59, 1997.
- Pra 91 D. K. Pradhan, N. Srinivas, J. Swaminathan, "A Methodology for Partial Scan Design", *Proceedings of the 2nd IEEE European Test Conference*, pp. 263-269, 1991.
- Pra 94 D. K. Pradhan, N. H. Vaidya, "Roll-Forward Checkpointing Scheme: A Novel Fault-Tolerant Architecture", *IEEE Transactions on Computers*, Vol-43, N-10, pp. 1163-1174, 1994.

R

- Reb 98 M. Rebaudengo, M. S. Reorda, "An Evaluation of Software Redundancy Techniques for Hardware Errors Detection", *Compendium of Papers, 4th IEEE International On-Line Testing Workshop*, pp. 202-206, 1998.

- Red 85 M. Reddy, S. Reddy, P. Agrawal, "Transistor Level TG for MOS Circuits", *Proceedings of the 22nd ACM/IEEE Design Automation Conference*, pp. 825-828, 1985.
- Ren 98a M. Renovell, J. Portal, J. Figueras, Y. Zorian, "SRAM-Based FPGAs: Testing the RAM Mode of the LUT/RAM Modules", *Compendium of Papers, IEEE European Test Workshop*, pp. 5-9, 1998.
- Ren 98b M. Renovell, F. Azais, Y. Bertrand, "Analysing Relationship between Defect and Fault Model", *Compendium of Papers, IEEE European Test Workshop*, pp. 151-155, 1998.
- Riu 97 J. Rius, J. Figueras, "Circuit Delay Degradation due to Sensors for I_{DDQ} Testing", *Compendium of Papers, 3rd IEEE International On-Line Testing Workshop*, pp. 116-120, 1997.
- Roq 98 R. Roques, A. Correge, C. Boleat "Fault-Tolerant Computer for the Automated Transfer Vehicle", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 412-419, 1998.
- Rot 66 J. Roth, W. Bouricius, "Diagnosis of Automata Failures: A Calculus and a Method", *IBM Journal of R&D*, Vol-10, N-4, pp. 278-291, 1966.
- Rus 96 R. J. Russell, "A Method of Extending an 1149.1 Bus for Mixed-Signal Testing", *Proceedings of the IEEE International Test Conference*, pp. 410-416, 1996.

S

- Sal 88 K. K. Saluja, R. Sharma, C. R. Kime, "A Concurrent Testing Technique for Digital Circuits", *IEEE Transactions on CAD*, V. 7, N-12, pp. 1250-1260, 1988.
- Sal 97 Z. Salcic, A. Smailagic, *Digital Systems Design and Prototyping Using Field Programmable Logic*, Kluwer Academic Publishers, 1997, ISBN 0-7923-9935-8.
- San 97a J. M. V. Santos, J. M. M. Ferreira, "POST: Pseudo On-Line BS Failure Detection", *Compendium of Papers, IEEE European Test Workshop*, PS-3.5, 1997.
- San 97b J. M. V. Santos, J. M. M. Ferreira, "Failure Detection and Boundary Scan: a Pseudo On-Line Approach (POST)", *Compendium of Papers, 3rd IEEE International On-Line Testing Workshop*, pp. 160-164, 1997.
- San 97c J. M. V. Santos, J. M. M. Ferreira, "The POST Approach to On-Line Failure Detection Based on BST", *Proceedings of the Design of Circuits and Integrated Systems Conference*, pp. 27-32, 1997.

- San 97d J. M. V. Santos, "F-T: How Interesting in Industrial Equipment", *Proceedings of the IEEE International Symposium on Industrial Electronics*, Student Forum, pp. 7-11, 1997.
- San 98a J. M. V. Santos, J. M. M. Ferreira, "On Improving BST for Concurrent Functional Verification", *Compendium of Papers, IEEE European Test Workshop*, pp. 28-29, 1998.
- San 98b J. M. V. Santos, J. M. M. Ferreira, "Dependable Circuits Based on the XMR Architecture", *IEEE Fault Tolerant Computing Symposium, Digest of FastAbstracts*, pp. 11-12, 1998.
- San 98c J. M. V. Santos, J. M. M. Ferreira, "XMR: The Incomplete TMR Solution", *Compendium of Papers, 4th IEEE International On-Line Testing Workshop*, pp. 131-135, 1998.
- San 98d J. M. V. Santos, J. M. M. Ferreira, "Fault Tolerance: New Trends for Digital Circuits", *Proceedings of the 5th IEEE International Conference on Electronics Circuits and Systems*, Vol-3, pp. 237-240, 1998.
- San 98e J. M. V. Santos, J. M. M. Ferreira, "Joining Boundary-Scan to NMR Architectures: The XMR Architecture", *Proceedings of the Design of Circuits and Integrated Systems Conference*, pp. 630-634, 1998.
- San 99a J. M. V. Santos, J. M. M. Ferreira, "Adding Fast Parity Check to BST for Concurrent Monitoring", *Compendium of Papers, IEEE European Test Workshop*, P.S-4, 1999.
- San 99b J. M. V. Santos, J. M. M. Ferreira, "On Improving BST Reliability for On-Line Scan", *Compendium of Papers, IEEE International On-Line Testing Workshop*, pp.97-100, 1999.
- San 99c J. M. V. Santos, J. M. M. Ferreira, "On-Line Parity Control Enhances 1149.1 BST Reliability", *Proceedings of the Design of Circuits and Integrated Systems Conference*, 1999.
- Sant 97 M. Santos, F. Gonçalves, I. Teixeira, J. P. Teixeira, "Realistic Fault Modeling of Spot Defects in CMOS Digital ICs", *Compendium of Papers, IEEE European Test Workshop*, S-2.1, 1997.
- Sav 83 J. Savir, "Good Controllability and Observability do not Guarantee Good Testability", *IEEE Transactions on Computers*, Vol-C-32, N-12, pp. 1198-1200, 1983.
- Sav 84 J. Savir, G. S. Ditlow, P. H. Bardell, "Random Pattern Testability", *IEEE Transactions on Computers*, Vol-C-33, N-1, pp. 79-90, 1984.

- Sax 92 N. R. Saxena, P. Franco, E. J. McCluskey, "Simple Bounds on Serial Signature Analysis Aliasing for Random Testing", *IEEE Transactions on Computers*, Vol-41, N-5, pp. 638-647, 1992.
- Sch 72 D. R. Schertz, G. Metze, "A New Representation for Faults in Combinational Circuits", *IEEE Transactions on Computers*, Vol-C-21, N-8, pp. 858-866, 1972.
- Sch 92 J. M. Schoen, *Performance and Fault Modeling with VHDL*, Prentice Hall, NJ, 1992, ISBN 0-13-658816-6.
- Sed 94 R. M. Sedmak, "Boundary-Scan: Beyond Production Test", *IEEE Transactions on Computers*, V43, N-4, pp. 415-420, 1994.
- Sen 92 E. Sentovich, K. Singh, L. Lavagno, C. Moon, "SIS: a System for Sequential Systems Synthesis", *Publicação Interna da Universidade da Califórnia*, 1992.
- Set 82 S. Seth, V. D. Agrawal, "Statistical Design Verification", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 393-399, 1982.
- Set 85a S. Seth, V. D. Agrawal, "Cutting Chip-testing Costs", *IEEE Spectrum*, pp. 38-45, 1985.
- Set 85b S. Seth, L. Pan, V. D. Agrawal, "Predict-Probabilistic Estimation of Digital Circuit Testability", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 220-225, 1985.
- Sha 88 R. Sharma, K. K. Saluja, "An Implementation and Analysis of a Concurrent BIST Technique", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 164-169, 1988.
- She 85 J. Shen, W. Maly, F. Ferguson, "Inductive Fault Analysis of MOS Integrated Circuits", *IEEE Design and Test of Computers*, Vol-2, N-12, pp. 13-26, 1985.
- She 90 Y.-N. Shen, F. Lombardi, "Yield Enhancement and Manufacturing Throughput of Redundant Memories by Reparability/Unreparability Detection", *Journal of Electronic Testing: Theory and Applications*, N-1, pp. 43-57, 1990.
- Shr 81a T. Shridar, J. P. Hayes, "A Functional Approach to Testing Bit-Sliced Microprocessors", *IEEE Transactions on Computers*, Vol-C-30, N-8, pp. 536-571, 1981.
- Shr 81b T. Shridar, J. P. Hayes, "Design of Easily Testable Bit-Sliced Systems", *IEEE Transactions on Computers*, Vol-C-30, N-11, pp. 842-851, 1981.
- SIA 97 *Semiconductor Industry Association Roadmap*, disponível no endereço: <http://www.semichip.org>
- Sie 90 D. P. Siewiorek, M. Hsao, D. Rennels, J. Gray, T. Williams, "Ultradependable Architectures", *Annals of Revue of Computer Science*, N-4, pp. 503-515, 1990.

- Sie 91 D. P. Siewiorek, "Architecture of Fault-Tolerant Computers: An Historical Perspective", *Proceedings of the IEEE*, Vol-79, N-12, pp. 1710-1734, 1991.
- Sil 88 G. Silberman, H. Spillinger, "Riddle: A Foundation for Test Generation on a High Level Design Description", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 76-81, 1988.
- Sil 98 José A. P. M. Silva, *Teste de Circuitos Analógicos e Mistos por Correlação da Corrente de Alimentação e da Tensão de Saída*, Tese de Doutorado, Faculdade de Engenharia da Universidade do Porto, Maio 1998.
- Silv 98 L. M. Silva, J. G. Silva, "Avoiding Checkpointing Contamination in Parallel Systems", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 364-369, 1998.
- Silva 98 J. G. Silva, P. Prata, M. Rela, H. Madeira, "Practical Issues in the Use of ABFT and a New Failure Model", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 26-35, 1998.
- Sim 94 W. R. Simpson, J. W. Sheppard, *Systems Test and Diagnosis*, Kluwer Academic Publishers, 1994, ISBN 0-7923-9475-5.
- Sit 93 R. K. Sitaraman, N. K. Jha, "Optimal Design of Checks for Error Detection and Location in Fault-Tolerant Multiprocessor Systems", *IEEE Transactions on Computers*, Vol-42, N-7, pp. 780-789, 1993.
- Ska 95 K. Skahill, *VHDL for Programmable Logic*, Cypress Semiconductor Corporation, 1995.
- Skl 76 J. R. Sklaroff, "Shuttle Computers", *IBM Journal of Research and Development*, pp. 20-28, Janeiro 1976.
- Smi 78 J. E. Smith, G. Metze, "Strongly Fault Secure Logic Networks", *IEEE Transactions on Computers*, Vol-C-27, N-6, pp. 491-499, 1978.
- Sos 98 J. Sosnowski, J. Nowicki, "Experiments with On-Chip Monitoring in Pentium Processors", *IEEE Fault Tolerant Computing Symposium, Digest of FastAbstracts*, pp. 54-55, 1998.
- Spa 98 L. Spainhower, T. A. Gregg, "G4: A Fault-Tolerant CMOS Mainframe", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 432-440, 1998.
- Ste 91 D. Sterba, A. Halliday, D. McClean, "ATPG and Diagnostics for Boards Implementing Boundary Scan", *Journal of Electronic Testing: Theory and Applications*, N-2, pp. 89-98, 1991.
- Ste 96 E. Steck, K. Rokhsaz, S-P. Shine, "A Linear and Neural Network Feedback for Flight Control Decoupling", *IEEE Control Systems*, Vol-16, N-4, 1996.



- Sto 96 N. Storey, *Safety-Critical Computer Systems*, Addison-Wesley Longman, 1996, ISBN 0-201-42787-7.
- Str 93 C. E. Stroud, A. E. Barbour, "Testability and Test Generation for Majority Voting Fault-Tolerant Circuits", *Journal of Electronic Testing: Theory and Applications*, N-4, pp. 201-214, 1993.
- Str 97 C. E. Stroud, E. Lee, M. Abramovici, "BIST-Based Diagnostics of FPGA Logic Blocks", *Proceedings of the IEEE International Test Conference*, pp. 539-547, 1997.
- Sze 96 V. Szekely, Rencz, J. M. Karam, M. Lubaszewski, B. Courtois, "Thermal Monitoring of Self-Checking Systems", *Compendium of Papers, 2nd IEEE International On-Line Testing Workshop*, pp. 6-13, 1996.
- Sze 98 V. Szekely, M. Rencz, Cs. Marta, B. Courtois, "Various Aspects of the Design for Thermal Testability (DfTT)", *Compendium of Papers, IEEE European Test Workshop*, pp. 57-61, 1998.

T

- Tav 95 J. Tavares, *Geração de Programas de Teste para CCI com BST*, Tese de Mestrado, Faculdade de Engenharia da Universidade do Porto, 1995.
- Tei 91 J. P. Teixeira, I. Teixeira, C. Almeida, F. Gonçalves, J. Gonçalves, "A Methodology for Testability Enhancement at Layout Level", *Journal of Electronic Testing: Theory and Applications*, N-1, pp. 287-299, 1991.
- Tem 97 G. Tempesti, D. Mange, A. Stauffer, "A Self-Repairing FPGA Architecture Inspired by Biology", *Compendium of Papers, 3rd IEEE International On-Line Testing Workshop*, pp. 191-195, 1997.
- Tex 93 Texas Instruments, *FPGA Applications Handbook*, 1993.
- Tex 95a "Using DFT in ASICs", TI IEEE 1149.1 Testability Primer, capítulo 4, 1995, disponível no endereço: <http://www.ti.com/sc/docs/jtag/jtag2.htm>
- Tex 95b "IEEE 1149.1 Applications", TI IEEE 1149.1 Testability Primer, capítulo 7, 1995, disponível no endereço: <http://www.ti.com/sc/docs/jtag/jtag2.htm>
- Tex 99 Componentes JTAG da Texas Instruments, disponível no endereço: <http://www.ti.com/sc/docs/jtag>
- Thi 92 C. Thibeault, Y. Savaria, J-L. Houle, "Test Quality of Hierarchical Defect-Tolerant Integrated Circuits", *Journal of Electronic Testing: Theory and Applications*, N-3, Kluwer Academic Publishers, pp. 93-102, 1992.
- Thi 93 C. Thibeault, Y. Savaria, J-L Houle, "A Fast Method to Evaluate the Optimum Number of Spares in Defect-Tolerant Integrated Circuits", *IEEE Transactions on Computers*, Vol-43, N-6, pp. 475-484, 1993.

- Toy 78 W. Toy, "Fault-Tolerant Design of Local ESS Processors", *Proceedings of the IEEE*, pp. 1126-1145, 1988.

U

- Urb 98 G. Urban, H-J. Kolinowitz, J. Peleska, "A Survival Avionics System for Space Applications", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 372-379, 1998.

V

- Vef 91 H. Vefling, M. Viktil, "Test of Analog Components in a Digital Environment", *Proceedings of the 2nd IEEE European Test Conference*, p. 522, 1991.
- Vel 98 R. Velazco, P. Cheynet, A. Bofill, R. Ecoffet, "Thesic: A Tested Suitable for the Qualification of Integrated Circuits Devoted to Operate in Harsh Environment", *Compendium of Papers, IEEE European Test Workshop*, pp. 89-90, 1998.
- Vid 93 L. P. M. Vidal, *MAN VI. 00, Manutenção Condicional*, Tese de CESE, Instituto Superior de Engenharia do Porto, Julho de 1993.

W

- Wad 78 R. L. Wadsack, "Fault Modeling and Logic Simulation of CMOS and MOS Integrated Circuits", *Bell System Technical Journal*, Vol-57, pp. 1449-1474, 1978.
- Wag 91 K. D. Wagner, T. W. Williams, "Enhancing Board Functional Self-Test by Concurrent Sampling", *Proceedings of the IEEE International Test Conference*, pp. 633-640, 1991.
- Wei 92 C. B. Weinstock W. L. Heimerdinger, "The State of the Practice in Fault Tolerant Systems", *IEEE Fault Tolerant Computing Symposium, Digest of Papers*, pp. 2-5, 1992.
- Wen 78 J. Wensley, L. Laport, J. Goldberg, M. Green, K. Levitt, "SIFT: Design and Analysis of a F-T Computer for Aircraft Control", *Proceedings of the IEEE*, pp. 1240-1255, 1978.
- Whe 90 L. Whetsel, "Event Qualification: a Gateway to At-Speed Testing", *Proceedings of the IEEE International Test Conference*, pp. 135-141, 1990.
- Whe 92 L. Whetsel, "A Proposed Method of Accessing 1149.1 in a Backplane Environment", *Proceedings of the IEEE International Test Conference*, pp. 206-216, 1992.
- Whe 93 L. Whetsel, "Hierarchically Accessing 1149.1 Applications in a System Environment", *Proceedings of the IEEE International Test Conference*, pp. 517-526, 1993.

- Whe 96 L. Whetsel, "Proposal to Simplify Development of Mixed-Signal Test Standard", *Proceedings of the IEEE International Test Conference*, pp. 400-409, 1996.
- Wil 81 T. W. Williams, N. C. Brown, "Defect Level as a Function of Fault Coverage", *IEEE Transactions on Computers*, Vol-C-30, N-12, pp. 987-988, 1981.
- Wil 83 T. Williams, K. P. Parker, "Design for Testability - A Survey", *Proceedings of the IEEE*, Vol-71, N-1, pp. 98-112, 1983.

X

- Xia 92 W. Xiaoqing, K. Kinoshita, "Testable Design of Logic Circuits under Highly Observable Conditions", *IEEE Transactions on Computers*, Vol-41, N-5, pp. 654-659, 1992.
- Xil 99 Xilinx Corporation, disponível no endereço: <http://www.xilinx.com>
- Xu 85 S. Xu, S. Y. H. Su, "Detecting I/O and Internal Feedback Bridging Faults", *IEEE Transactions on Computers*, Vol-C-34, N-6, pp. 553-557, 1985.

Y

- Yan 98 T. Yang, Z. Peng, "Incremental Testability Analysis for Partial Scan Selection and Design Transformations", *Compendium of Papers, IEEE European Test Workshop*, pp. 107-112, 1998.

Z

- Zem 96 A. Zemva, "Testing and Optimization of Digital Integrated Circuits", disponível no endereço: <http://www.cbl.nscu.edu/publications/1996-Th-PhD-Zemva>
- Zor 90 Y. Zorian, V. K. Agarwal, "Optimizing Error Masking in BIST by Output Data Modification", *Journal of Electronic Testing: Theory and Applications*, N-1, pp. 57-71, 1990.
- Zor 92 Y. Zorian, A. Ivanov, "An Effective BIST Scheme for ROM's", *IEEE Transactions on Computers*, Vol-41, N-5, pp. 646-653, 1992.
- Zor 98 Y. Zorian, "Tools for Testing Systems on Silicon", *Compendium of Papers, IEEE European Test Workshop*, p. 79, 1998.

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

Capítulo 6

Capítulo 7

Capítulo 8

Bibliografía

Anexos

10. Anexos

Nestes anexos são apresentados os elementos de projecto de uma FPGA com as três propostas apresentadas, bem como os diagramas de simulação que ilustram e confirmam os princípios de funcionamento descritos. Atendendo ao pequeno número de pontos de teste do circuito ISCAS'85 C17, não é necessária a existência de registos de utilizador internos, pelo que temos apenas o registo periférico¹⁷. No entanto, deve notar-se que a definição de registos internos é no essencial um problema do teste de produção, que nem a infra-estrutura CST nem a arquitectura XMR afectam. Note-se ainda que, neste caso também, terá sempre de existir uma instrução análoga a SELRn, para seleccionar o registo desejado. Os esquemas incluídos nos anexos, são apresentados hierarquicamente, com detalhe crescente, a partir do esquema geral do CI.

¹⁷ - No caso de o circuito de missão justificar a definição de blocos funcionais, os sinais do controlador do TAP terão de ser controlados adequadamente para cada registo seleccionado, um processo que pode ser implementado multiplexando directamente estes sinais, ou utilizando (nas células) FF com sinal de inibição (*enable*).

10.1 Anexo A: O circuito integrado com infra-estrutura CST

A figura 10-1 representa o esquema global do CI onde foi implementada a infra-estrutura CST.

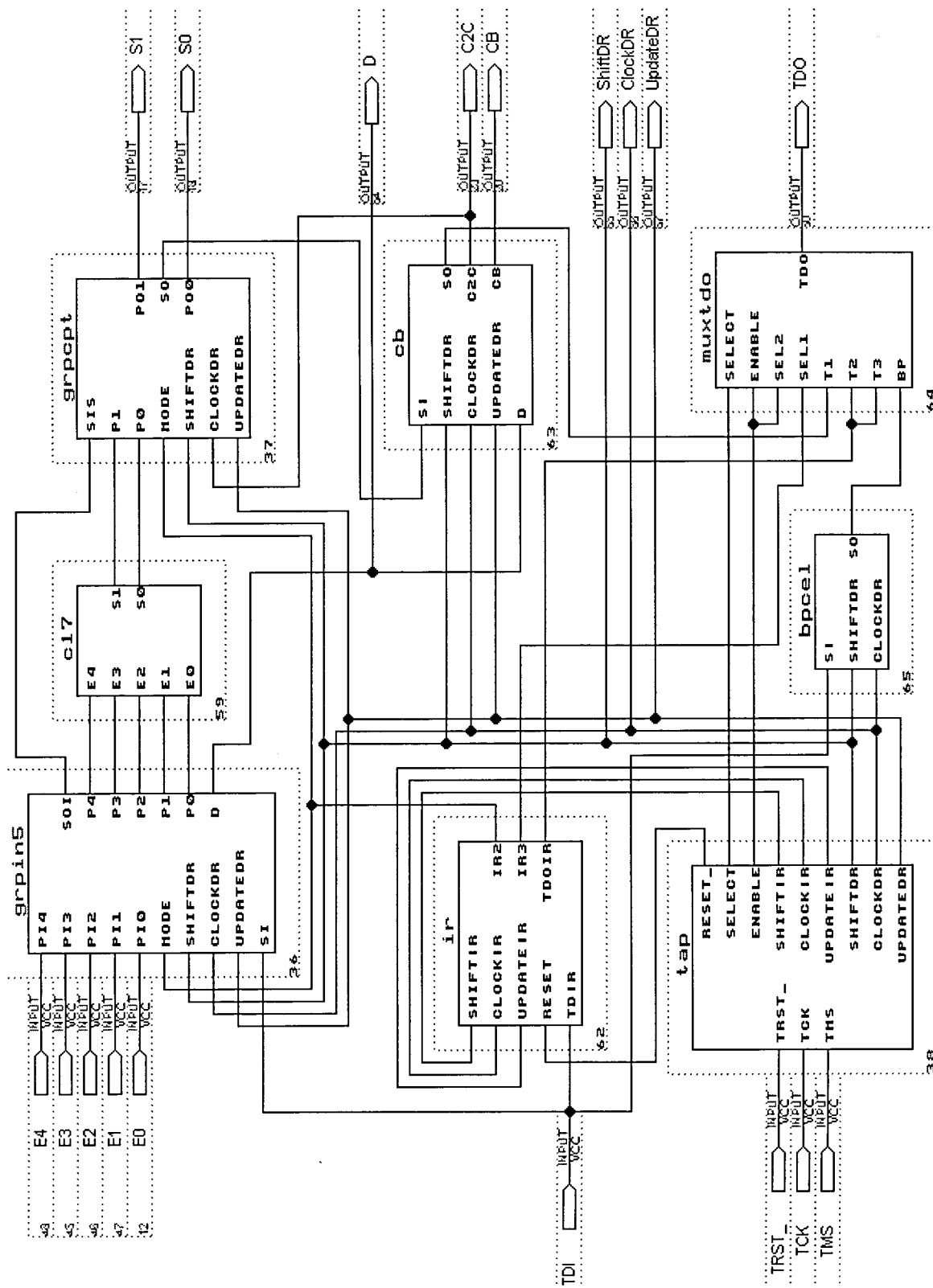


Figura 10-1. Esquema do CI com CST

No esquema da figura anterior, podemos reconhecer os blocos correspondentes ao controlador do TAP, o do registo de instrução (IR) e o de Bypass (BPCEL), o multiplexer de saída para TDO (MUXTDO), o grupo de células de entrada e detecção (GRPIN5), o grupo de células de saída e captura (GRPCPT), o circuito C17 e a célula com a lógica que controla a captura (CB), colocada junto a TDO com o objectivo de facilitar a interligação dos sinais.

Na figura 10-2, que representa o nível de topo do circuito, os sinais C2C e CB foram trazidos para o exterior a fim de permitir analisar a informação que lhes está associada, não sendo no entanto necessários no circuito definitivo.

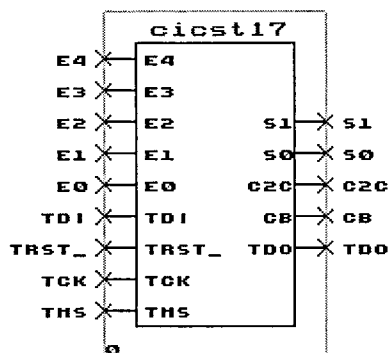


Figura 10-2. Terminais do CI com CST

A figura 10-3 representa o circuito ISCAS'85 C17, que foi utilizado como circuito de missão.

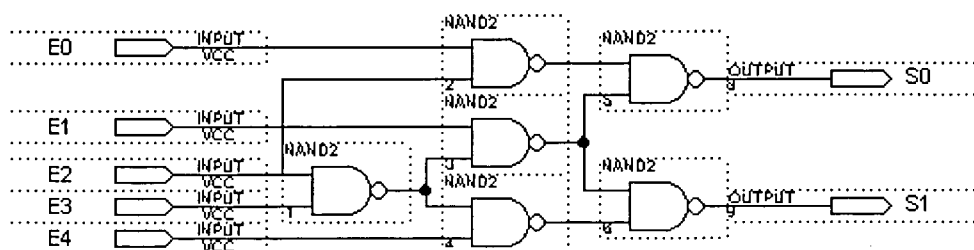


Figura 10-3. Esquema do circuito ISCAS'85 C17

10.1.1 Esquemas do grupo de entrada (GRPIN5)

O esquemas das figuras 10-4 e 10-5 representam, respectivamente, o grupo de células de entrada, em número igual ao das entradas do circuito de missão, e a constituição das células de detecção descritas no capítulo cinco, com destaque para porta lógica que gera o sinal *d*(etecção).

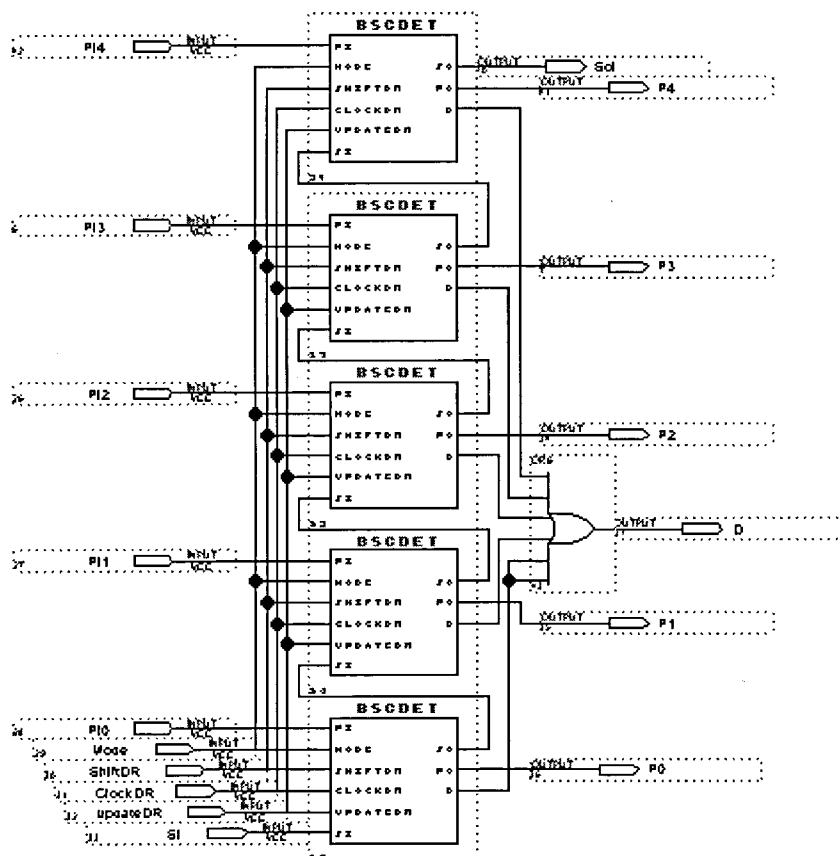


Figura 10-4. Grupo de células de entrada e detecção (GRPIN5)

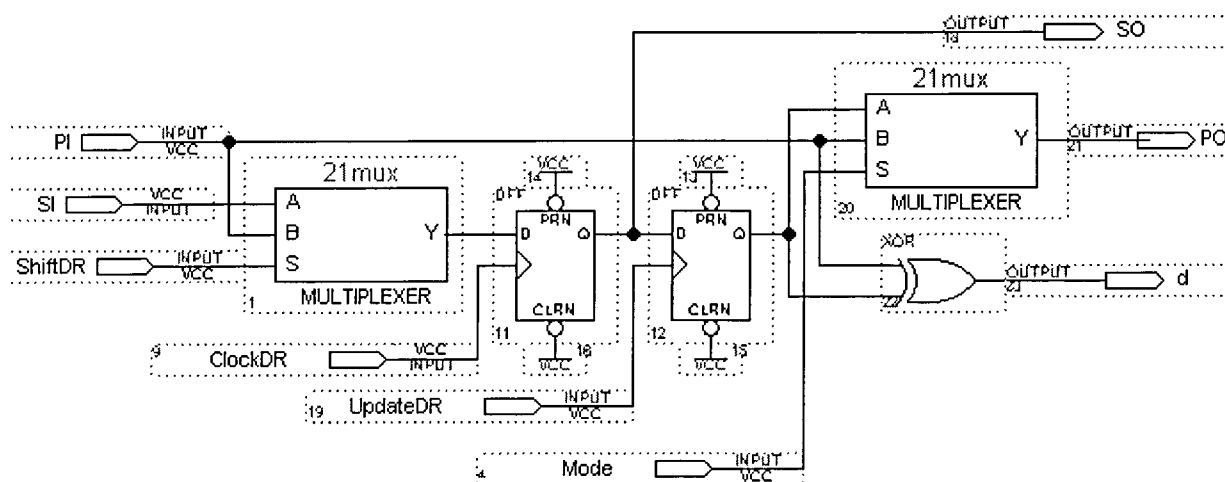


Figura 10-5. Esquema da célula de detecção

10.1.2 Esquemas do grupo de saída (GRPCPT)

O grupo de captura representado na figura 10-6 tem duas células, cuja constituição (normalizada) se documenta na figura 10-7.

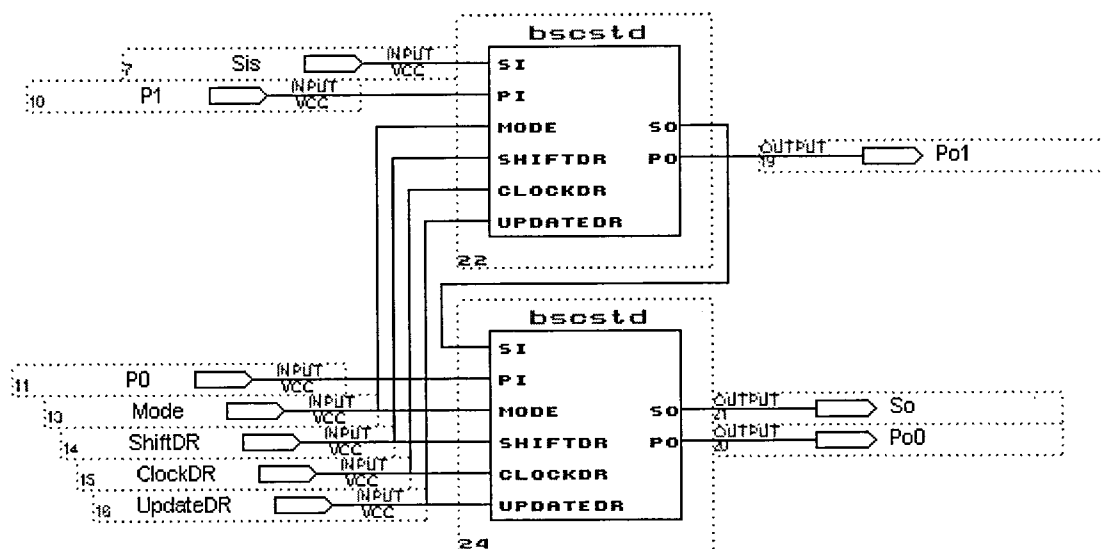


Figura 10-6. Grupo de células de saída e captura (GRPCPT)

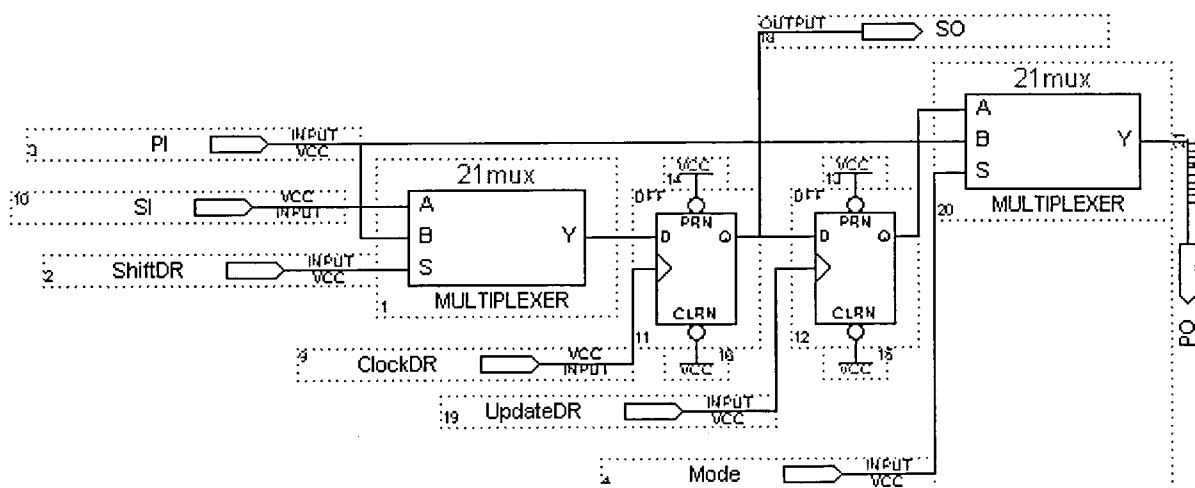


Figura 10-7. Célula de saída e captura

10.1.3 Esquema do circuito de controlo de captura (CB)

Nesta secção apresentamos o bloco CB, descrito na secção 5.2. Note-se contudo que existem duas variantes para a captura: se o circuito de missão tem um tempo de propagação superior ao do sinal C2C, será utilizado o circuito CB indicado no capítulo cinco (figura. 5-14); no caso contrário (tempo de propagação inferior a C2C), serão incluídas as duas portas lógicas indicadas na zona central da figura 10-8 (inversor e OR), que não alteram o princípio de funcionamento apresentado no capítulo cinco e apenas antecipam o instante em que a captura ocorre. No primeiro caso, a captura ocorre no fim da detecção (D), e no segundo, ocorre no início, atrasada de um tempo de propagação superior ao do circuito de missão.

No capítulo 5 descrevemos somente o caso mais geral, onde o circuito de missão terá tempo de propagação superior ao do sinal C2C, mas uma vez que na implementação aqui efectuada, este sinal tem um tempo de propagação correspondente a 5 portas lógicas, enquanto o circuito de missão C17 tem somente de 3, a captura efectua-se no início da detecção, como se pode verificar no sinal C2C da figura 10-14.

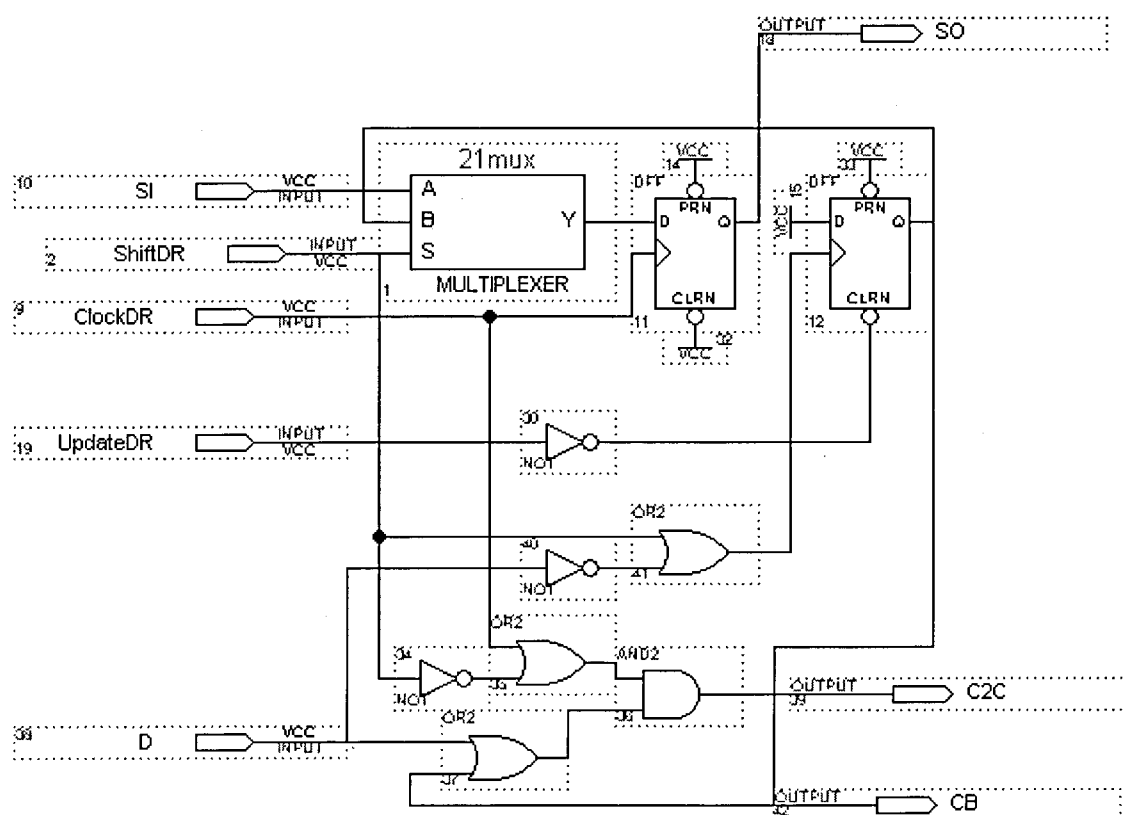


Figura 10-8. Circuito de controlo de captura (CB)

10.1.4 Multiplexer para TDO (MUXTDO)

Na figura 10-9 vemos representado o multiplexer para TDO, onde o sinal de maior prioridade (Sel2) está ligado conforme se indica na figura 10-1, reservando as entradas menos significativas para os sinais de controlo e as mais significativas para os registos de dados e o de instrução.

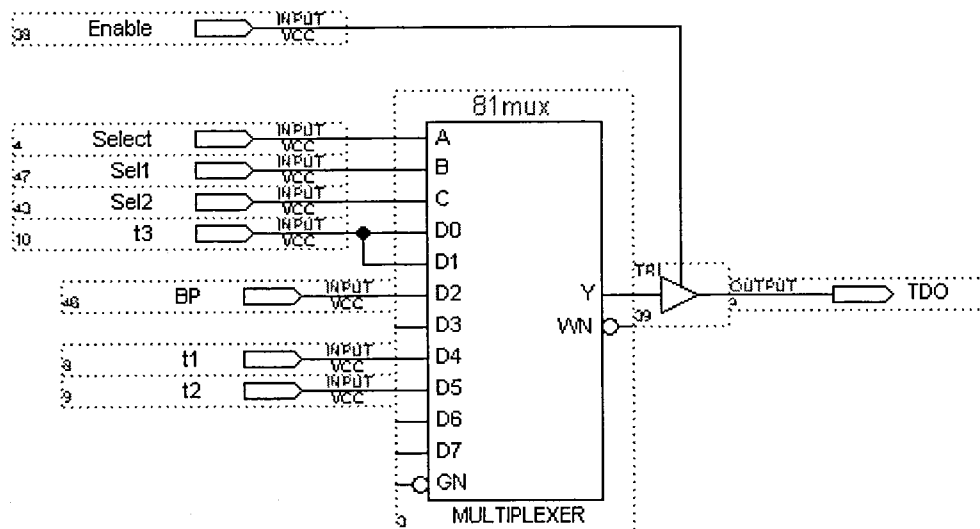


Figura 10-9. Circuito de multiplexagem para TDO (MUXTDO)

O registo de Instrução e o de Bypass, sendo independentes da proposta, não foram incluídos neste documento. Podem no entanto, ser consultados no CD-ROM que o acompanha (ficheiros: \CSTTAPCNTMR.gdf e BP.gdf).

10.1.5 Esquemas e diagrama do controlador do TAP

O controlador do TAP é o que se apresenta na figura 10-10 e coincide com o exemplo apresentado na norma 1149.1, sem qualquer alteração. É composto pelas duas partes indicadas nas figura 10-12 e 10-13, a que corresponde o diagrama temporal apresentado na figura 10-11.

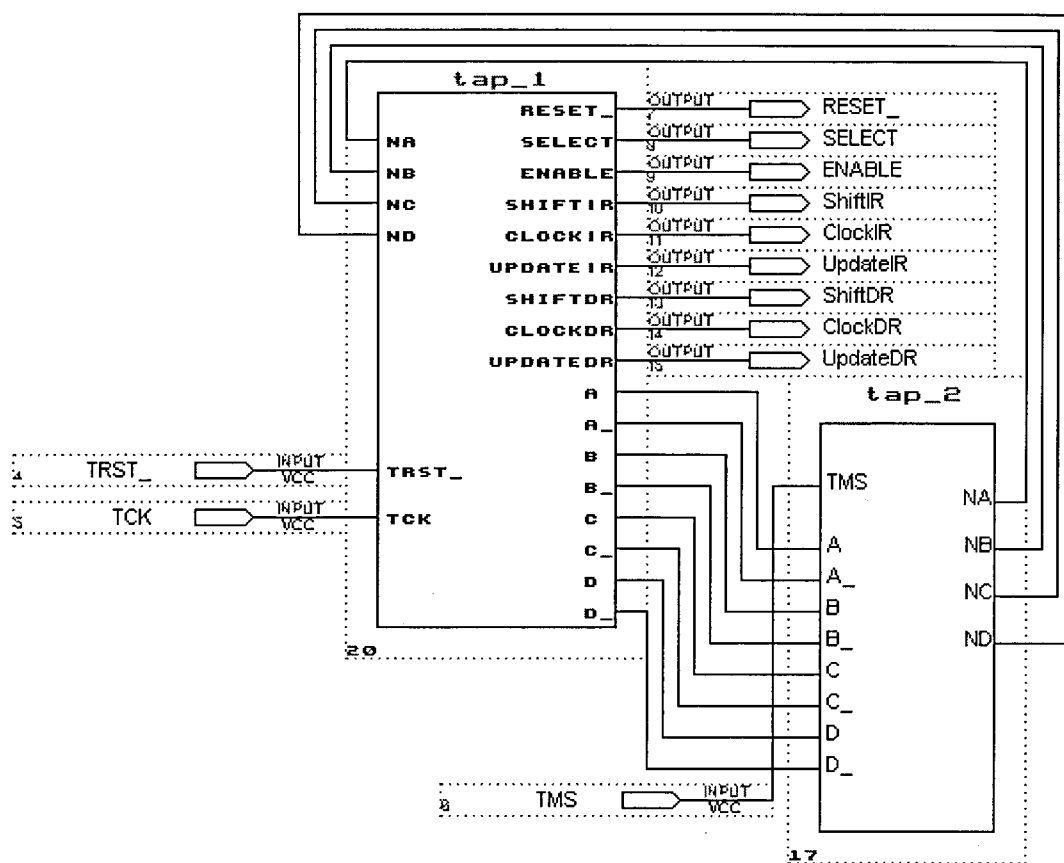


Figura 10-10. Controlador do TAP IEEE-P1149.1 (TAP)

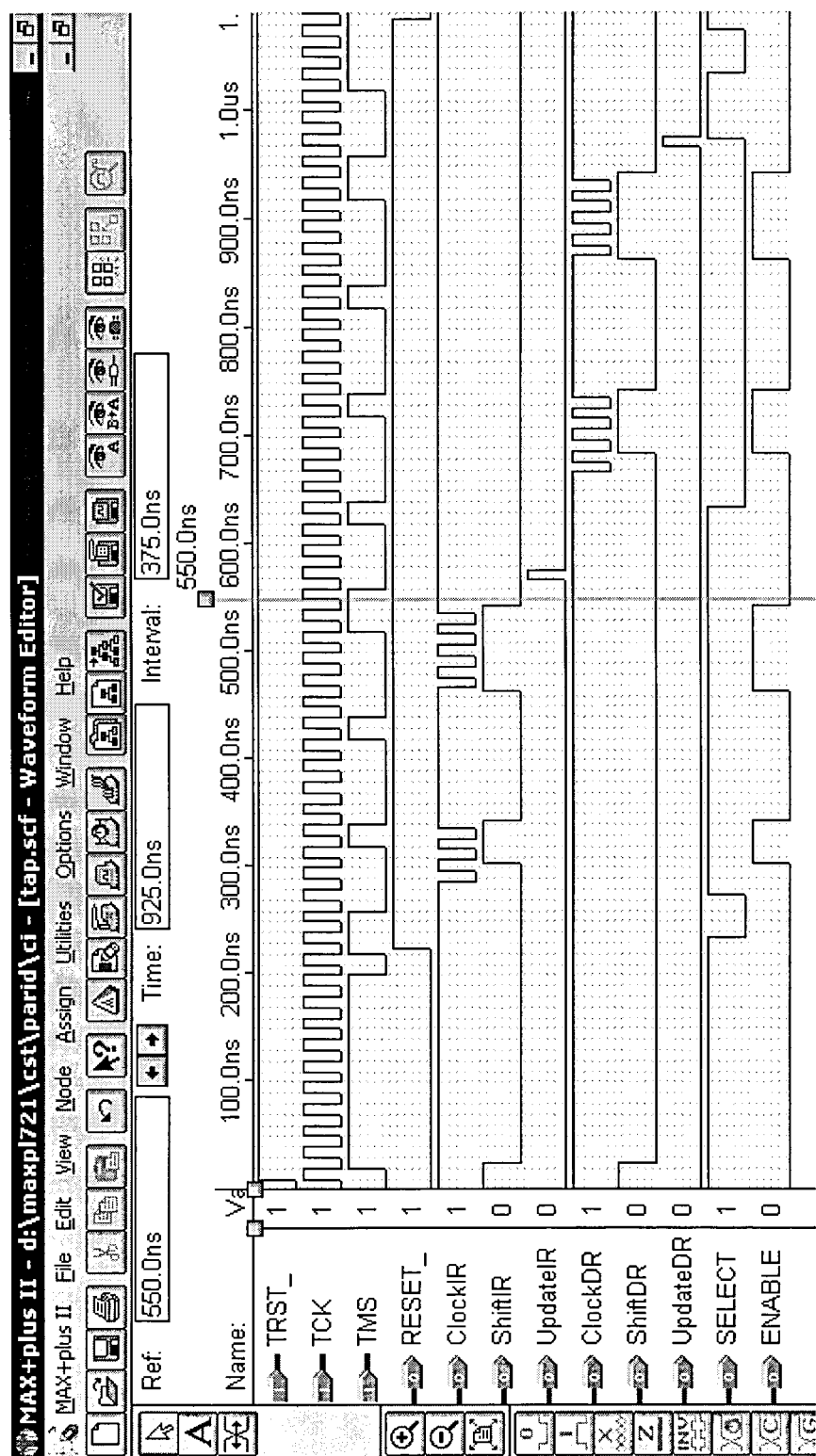


Figura 10-11. Diagramas temporais de funcionamento do controlador do TAP 1149.1

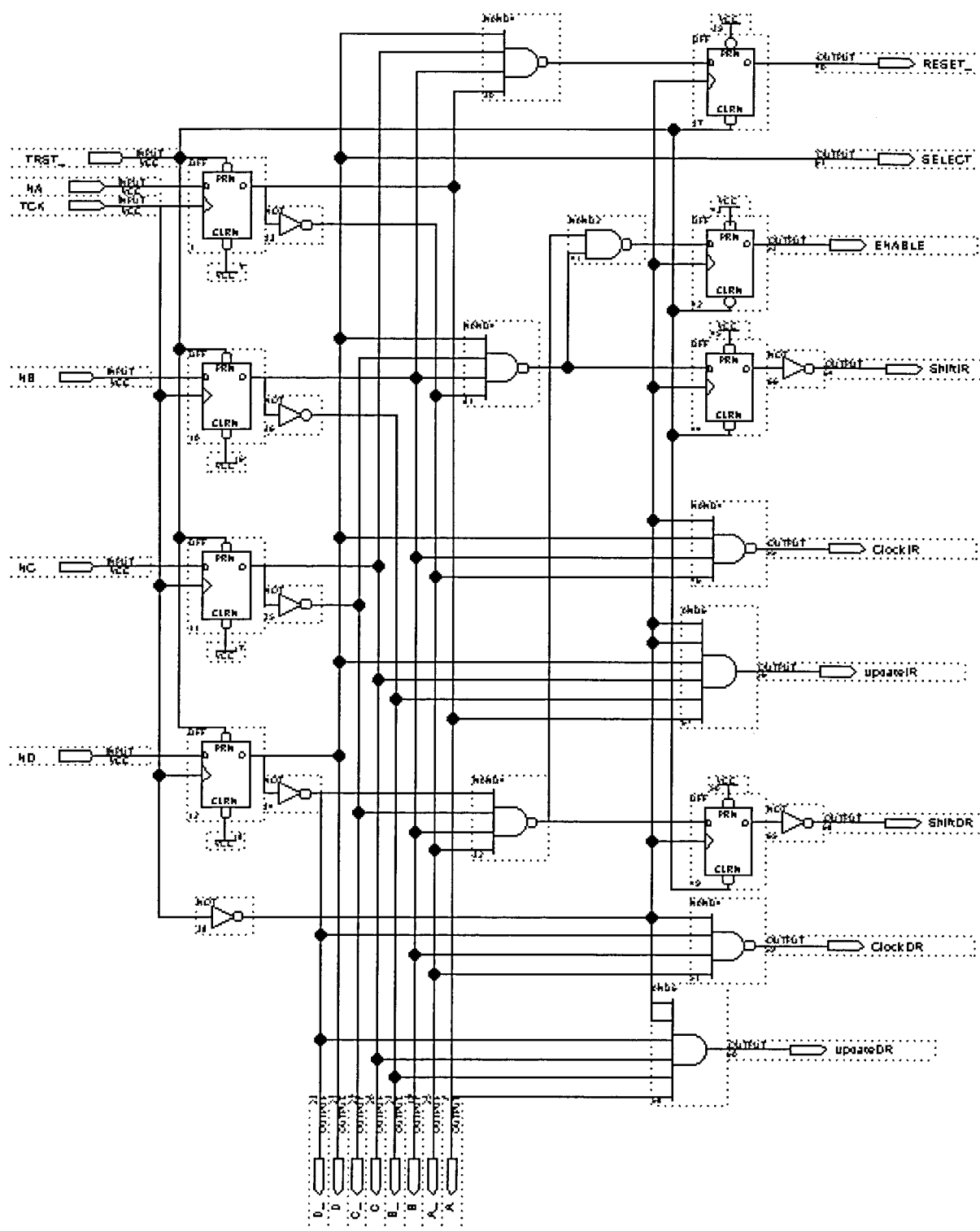


Figura 10-12. Parte 1 do controlador do TAP

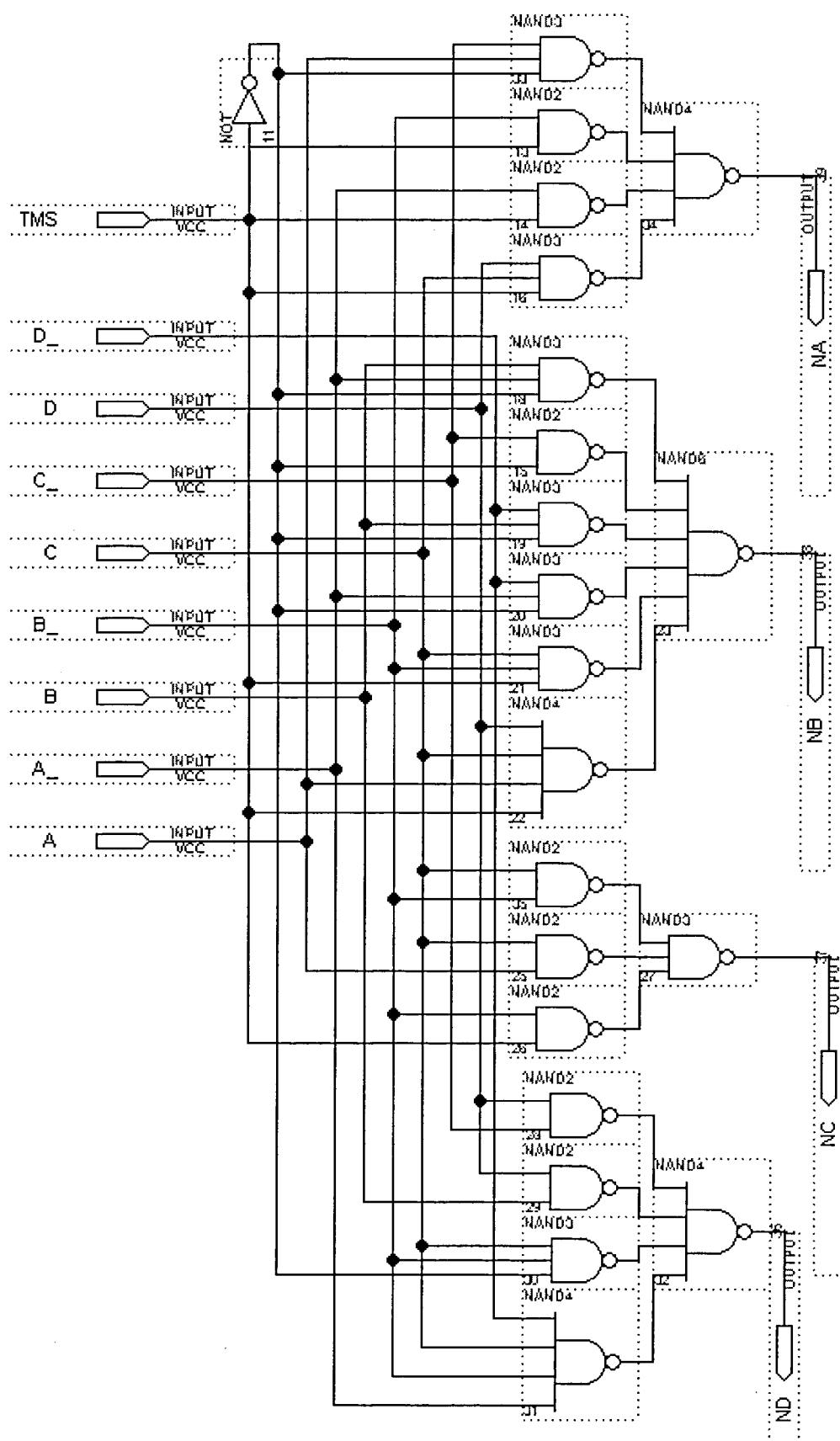


Figura 10-13. Parte 2 do controlador do TAP

10.1.6 *Diagrama temporal dos ciclos de detecção e captura*

A sobrecarga da infra-estrutura CST em cada registo é constituída pelas portas lógicas de detecção e pelo circuito que controla a captura, CB. O respectivo comportamento, integrado com o controlador do TAP, está representado na figura 10-14, onde destacamos:

- 500 ns - Início de varrimento do vector $VT_3 = 10101$.
- 1,1 μs - CB é activado pela transição no sinal Update-DR.
- 1,4 μs - Ocorre a detecção ($D=0$) e a captura é activada por C2C. CB é desactivado.
- 1,9 μs - Passagem pelo estado Capture-DR e inibição da passagem do primeiro impulso de Clock-DR para C2C.
- 2,0 μs - Início de varrimento do vector $VT_1 = 11110$. O primeiro bit deslocado por TDO indica que ocorreu captura, sendo seguido pelos dois “1” capturados nas saídas (S_0, S_1).
- 2,6 μs - CB é activado na passagem por Update-DR.
- 2,95 μs - Ocorre a detecção que desactiva CB e origina a captura de V_o .
- 3,4 μs - Passagem pelo estado Capture-DR.
- 3,5 μs - Início de novo varrimento, com a indicação de que ocorreu uma detecção, a que se seguem os dois “0” capturados em (S_0, S_1).
- 4,8 μs - Novo varrimento, com indicação de que não houve detecção.

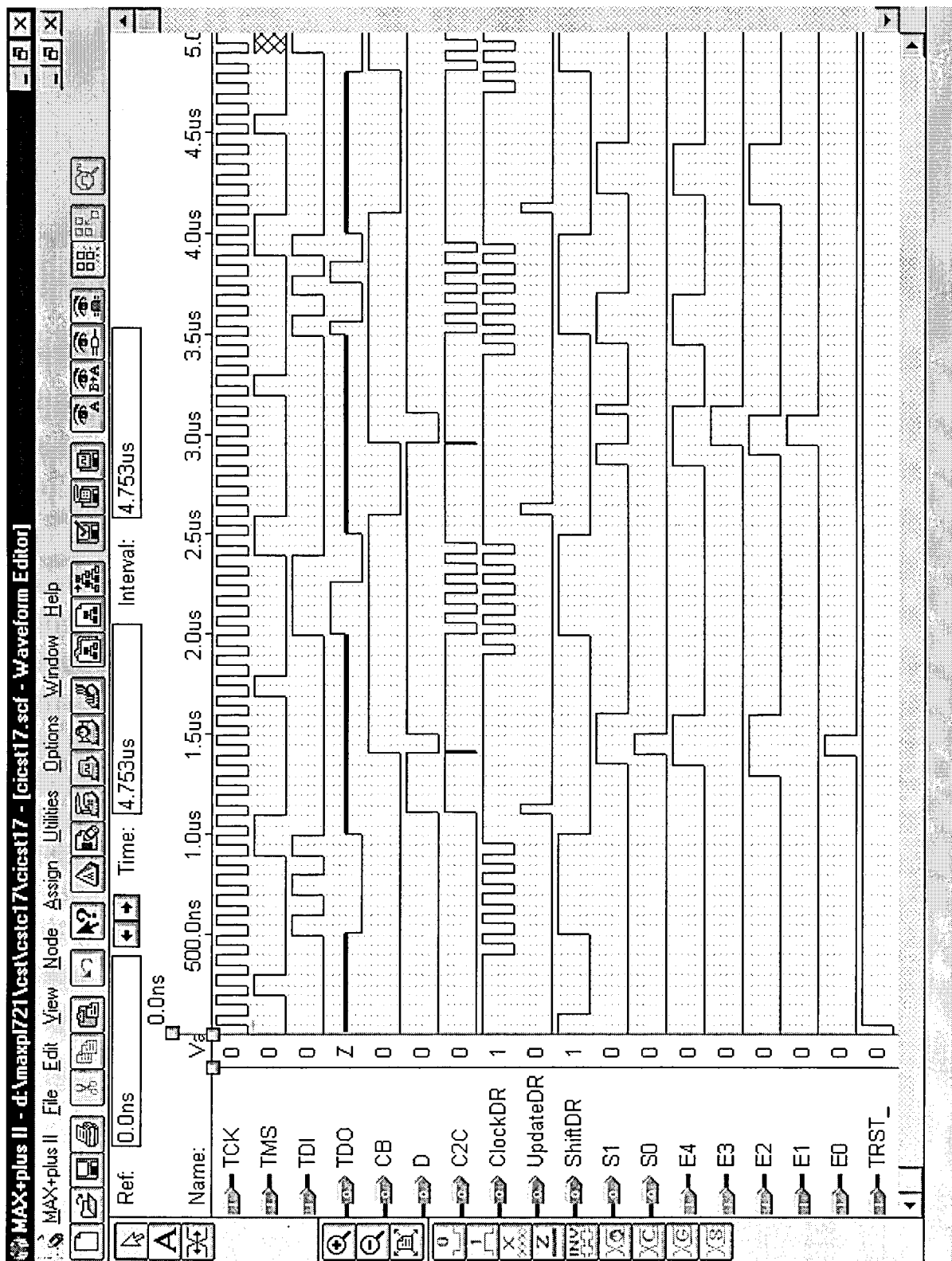


Figura 10-14. Diagrama de detecção e captura do circuito com infra-estrutura CST

Salienta-se neste diagrama a existência do circuito (CUTXMR) constituído por dois circuitos de missão C17 e com duas entradas para injeção de faltas (*fa*, *fb*), o novo grupo de saída e votação (GRPVOT) e ainda o bloco de retenção e direccionamento das células de saída (LGCRET). A figura 10-16 representa os pinos deste componente, onde os sinais CB, C2C, S, E, F e T foram trazidos para o exterior somente para análise.

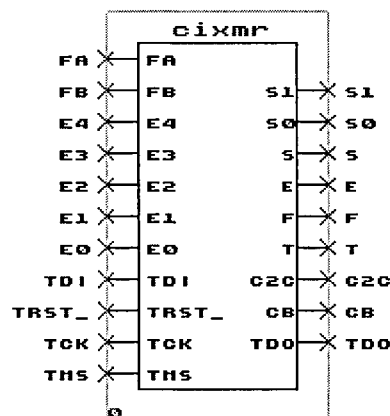


Figura 10-16. Entradas e saídas do CI com arquitectura XMR

O circuito de missão C17 foi duplicado como se mostra na figura 10-17, tendo ainda sido alterado para permitir a injeção de faltas idênticas em cada uma das réplicas A e B, *fa* e *fb* respectivamente, como se documenta na figura 10-18.

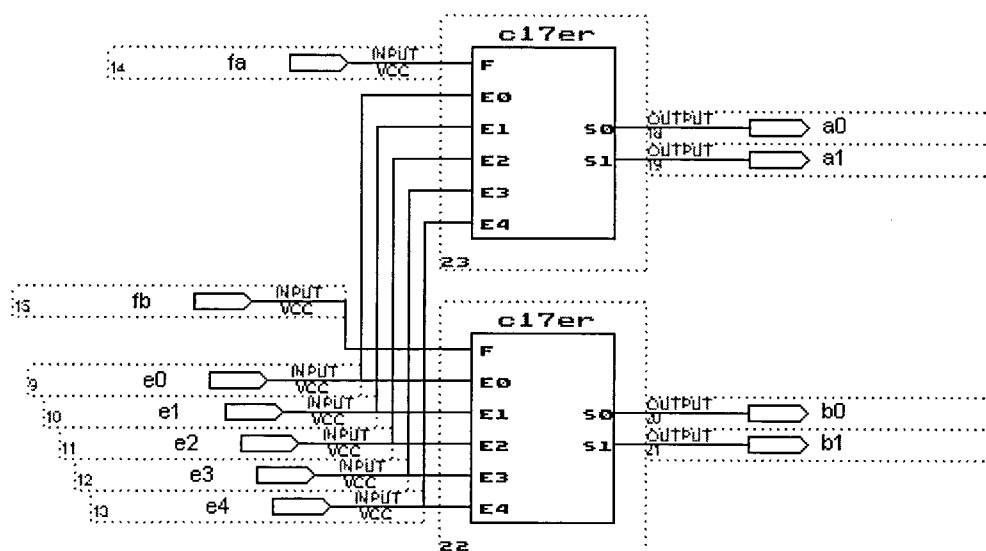


Figura 10-17. Duplo circuito de missão, com injeção de faltas (CUTXMR)

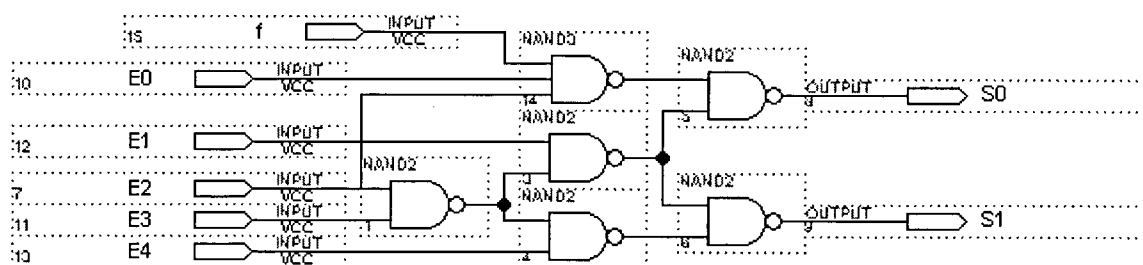


Figura 10-18. Circuito C17 com entrada para injeção de uma falta

Nota: Inicialmente, utilizámos uma versão do circuito C17 com 3 faltas injectáveis em cada circuito de missão, o que no entanto conduz a um número de diagramas excessivo para ser apresentado neste documento. Os esquemas respectivos podem contudo ser consultados no CD-ROM que acompanha este documento (ficheiro C17f3.gdf).

10.2.1 Esquemas dos blocos de saída e decisão

O circuito representado na figura 10-19 mostra as duas células de decisão, bem como a geração dos sinais E e S referidos na secção 6.2.

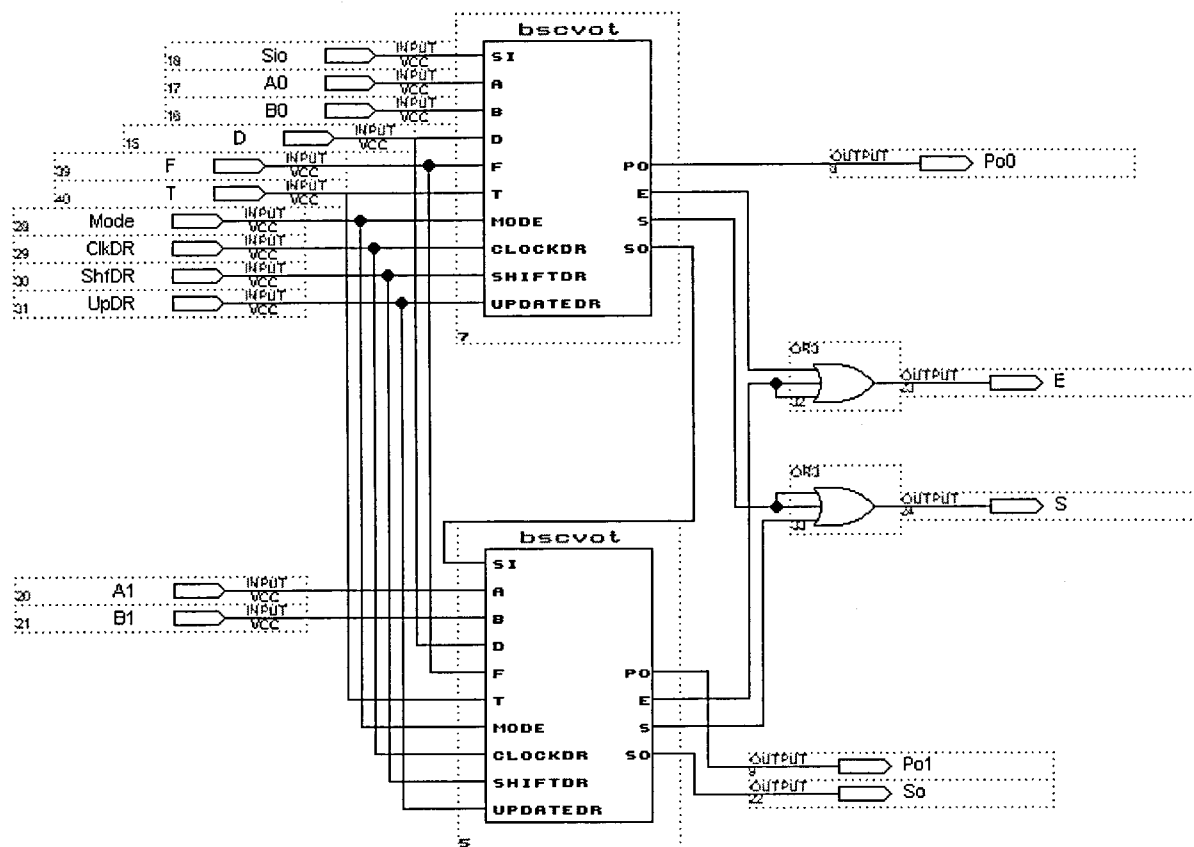


Figura 10-19 . Grupo de Saída e Decisão (GRPVOT)

A figura 10-20 representa a constituição da célula de decisão, com destaque para os blocos de decisão e votação (LGCVT), onde se vêem os sinais E, S, F e T. A parte restante da célula é a que está definida na norma, sem qualquer alteração.

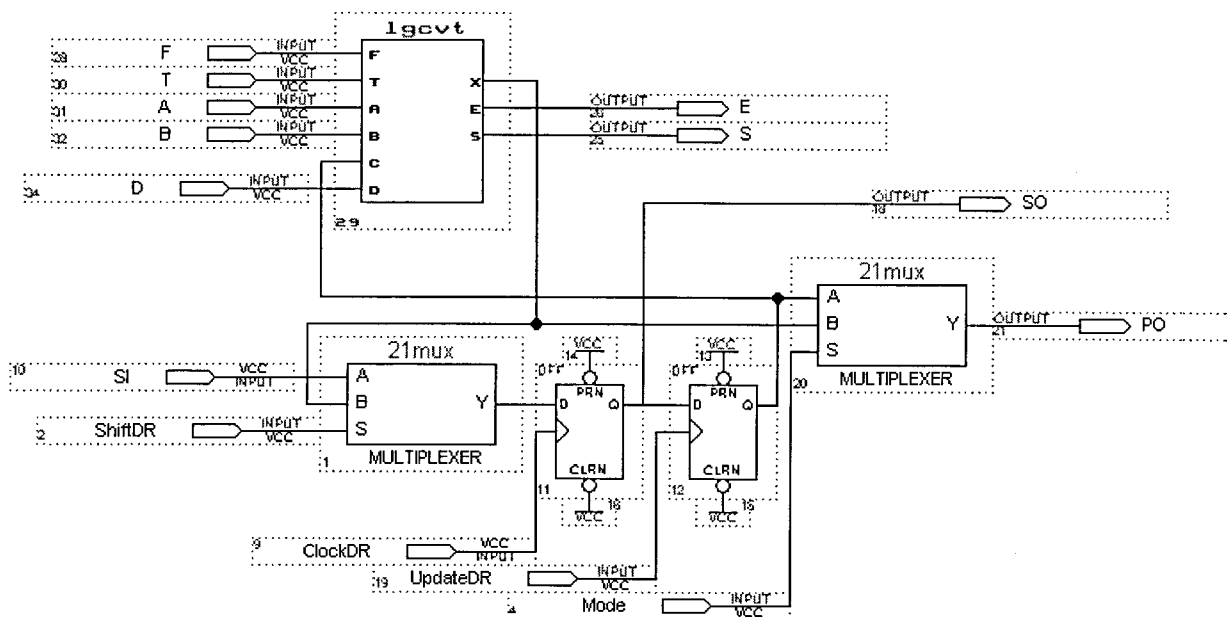


Figura 10-20. Célula de Decisão

O bloco lógico de votação (LGCVT) é constituído pelos dois blocos indicados na figura 10-21, a que correspondem as equações já indicadas na secção 6.2:

$$X = B \cdot F / T + A \cdot F / T$$

$$E = A \oplus B$$

$$S = /A \cdot B \cdot C / D + A \cdot B / C \cdot D$$

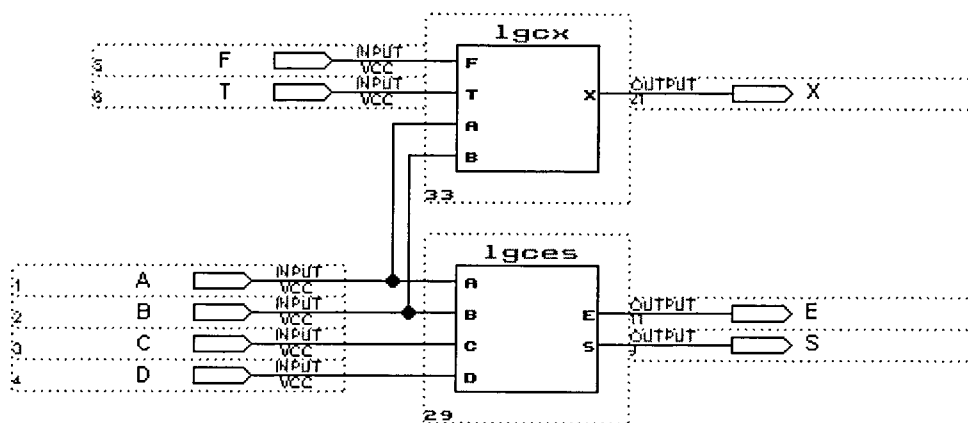


Figura 10-21 . Lógica de Decisão

O circuito de retenção e selecção, que permite direccionar para a saída do módulo a saída do circuito de missão correcto, está representado na figura 10-22, tendo como saídas os sinais F e T, que memorizam as condições de erro, conforme descrito no capítulo seis.

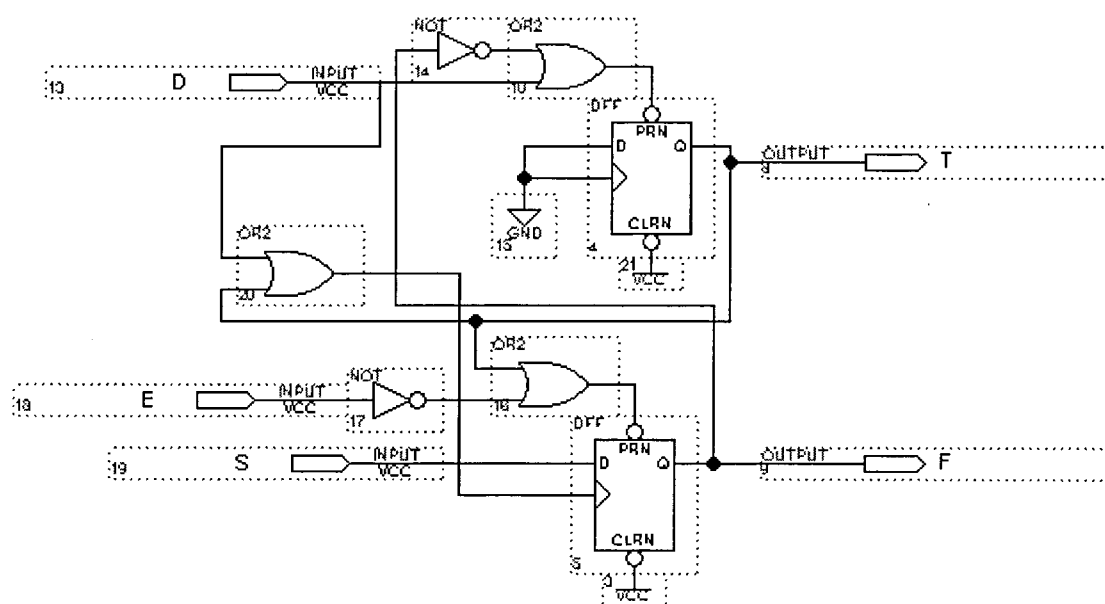


Figura 10-22. Circuito de retenção e direccionamento da saída (LGCRET)

10.2.2 Diagramas temporais

Os três diagramas seguintes representam o comportamento do CI XMR sem faltas e com cada falta injectada, separadamente, em cada um dos circuitos A e B. Com o objectivo de condensar os diagramas temporais, alternámos uma sequência dos quatro vectores de teste (VT_{0-3}) com outros sinais aleatórios. A primeira destas sequências serve de referência ao restante comportamento do circuito. Para os três casos referidos inicialmente, destacamos em seguida os aspectos mais relevantes, notando que os instantes temporais indicados são os de referência, independentemente dos atrasos da lógica (nestes diagramas, os sinais E, F e T são representados pelas correspondentes letras minúsculas).

1 - Comportamento sem faltas (vide figura 10-23):

- 150 ns - Primeira sequência de vectores VT_{0-3} , para referência.
- 500 ns - Início de um ciclo de dados. Repete-se a sequência de vectores de teste.
- 800 ns - Inicia-se o varrimento do vector $VT_3 = 11.10101$ ($VT_0 + VT_1$). O primeiro bit deslocado por TDO indica que não ocorreu captura prévia.
- 1,5 μ s - Fim do varrimento. TDO regressa a alta impedância.
- 1,6 μ s - CB é activado à subida de Update-DR.

- 1,8 μ s - O sinal C2C indica detecção na entrada (com $VT_3=10101$) e activa a captura. CB é desactivado.
- 2,1 μ s - Nova sequência de vectores de teste. Existe detecção mas não captura, uma vez que CB está desactivado.
- 2,5 μ s - Início de novo ciclo de dados.
- 2,8 μ s - Início de varrimento do vector $VT_1 = 00.11110$. O primeiro bit deslocado por TDO indica que ocorreu captura, sendo seguido pelos “1” capturados em (S_0, S_1).
- 3,5 μ s - Fim de ciclo, e TDO retorna a alta impedância.

2 - *Comportamento em presença da falta fa (diferenças na figura 10-24):*

- 400 ns - É injectada a falta no circuito de missão A.
- 600 ns - As saídas (S_0, S_1) denotam um comportamento diferente do de referência. O erro é detectado (sinal e) e retido pelo sinal f .
- 850 ns - Nova indicação de erro em e .
- 1,2 μ s - Repete-se a indicação de erro. As saídas (S_0, S_1) mantêm o estado seguro, “0”, até ser possível votar.
- 1,8 μ s - Ocorre a detecção (C2C) na entrada e votação: o sinal t passa a “1” e retém a informação de falta, enquanto f é fixado a “1”, encaminhando o circuito de missão B para a saída. O circuito passa a apresentar um comportamento correcto, apesar de continuar a existir erro (sinal e).

3 - *O comportamento em presença da falta fb é, em tudo, semelhante ao que existe com fa (figura 10-25):*

- 400 ns - É injectada a falta no circuito de missão B.
- 600 ns - As saídas (S_0, S_1) denotam um comportamento diferente do de referência. O erro é detectado e retido pelo sinal f .
- 850 ns - Nova indicação de erro.
- 1,2 μ s - Repete-se a indicação de erro. As saídas (S_0, S_1) mantêm o estado seguro, “0”, até surgir a votação.
- 1,8 μ s - Ocorre a detecção na entrada e votação: o sinal t passa a “0” e retém a informação de falta, enquanto f é fixado a “0”, encaminhando o circuito de missão A para a saída. O circuito passa a apresentar um comportamento correcto.

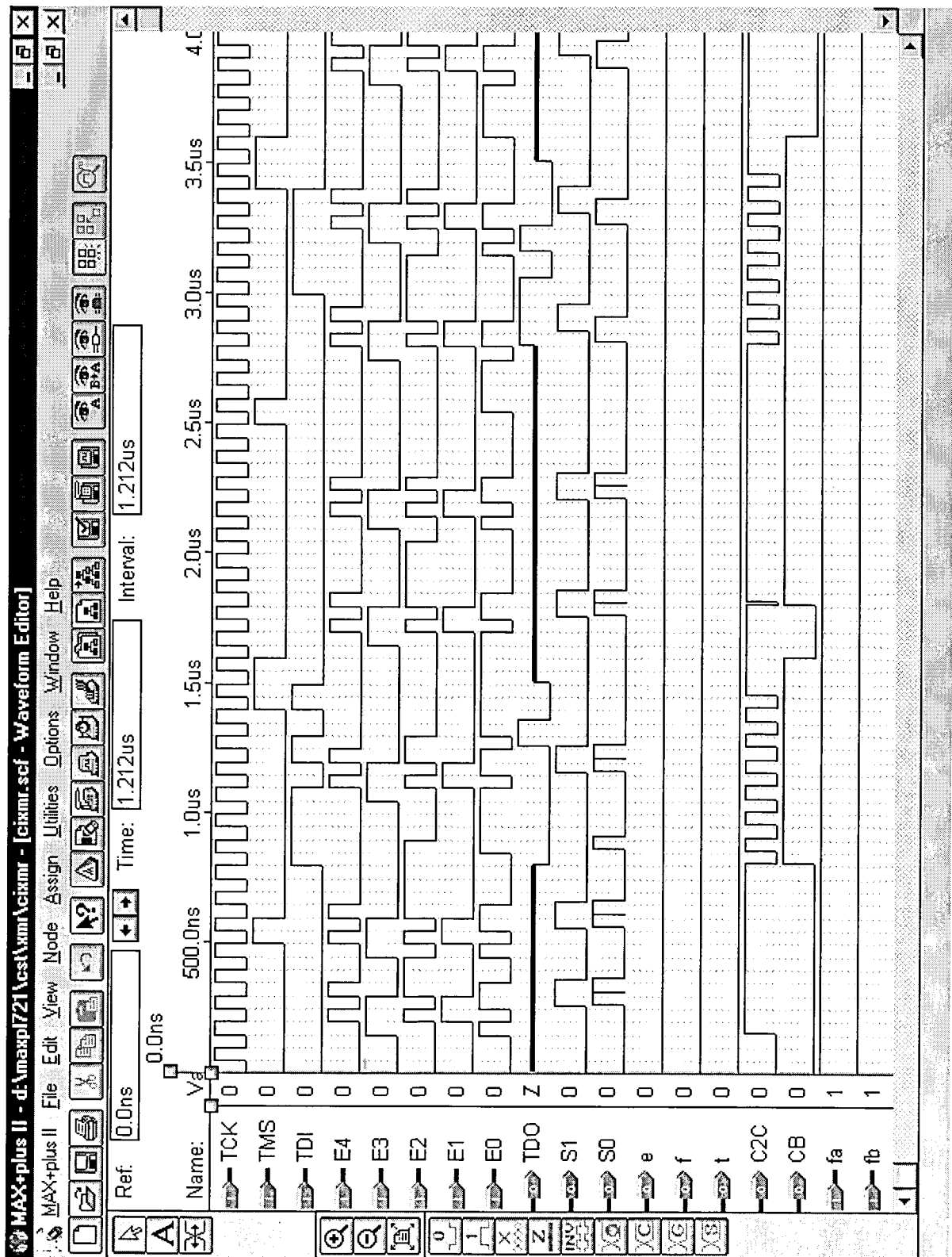


Figura 10-23. Comportamento sem faltas

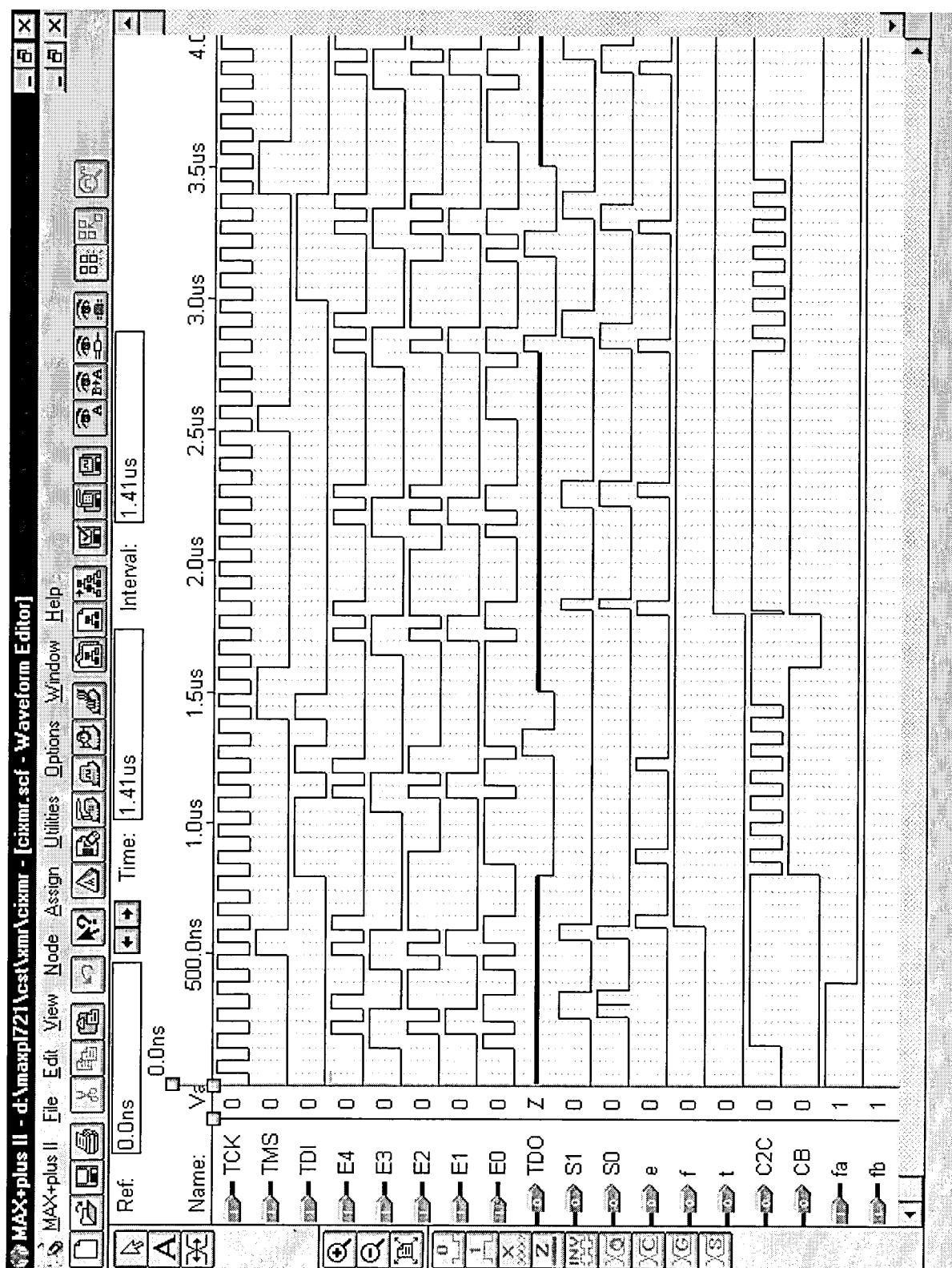


Figura 10-24. Comportamento com a falta *fa*

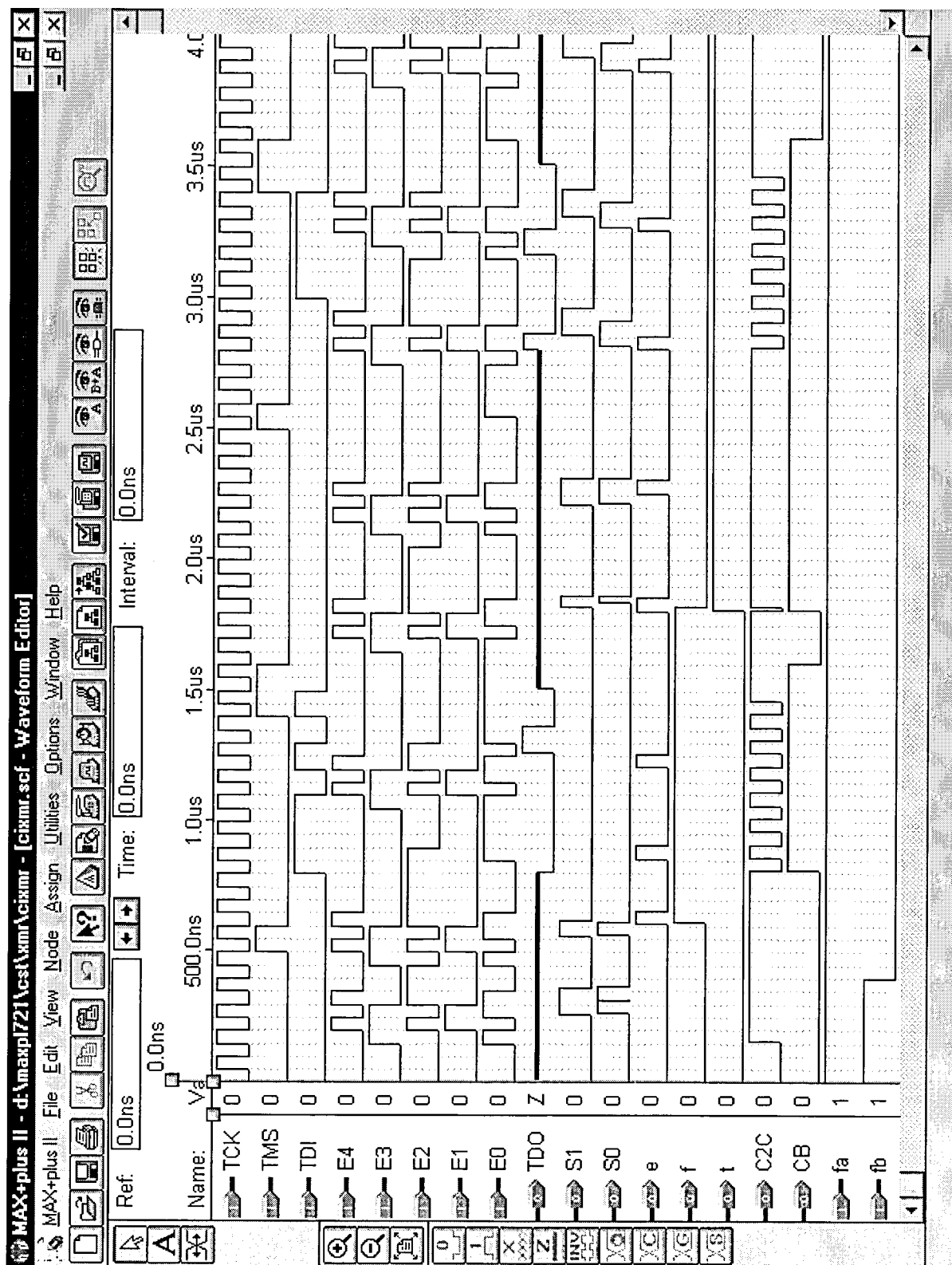


Figura 10-25. Comportamento com a falta *fb*

No esquema da figura anterior, que engloba a infra-estrutura CST, a arquitectura XMR e o controlo de Paridade na infra-estrutura BST, para além dos blocos caracterizados nos anexos A e B, foram acrescentados apenas dois blocos:

- PARDSR - gerador de paridade Série.
- ENABLP - o circuito que prolonga o sinal de *Enable*, permitindo manter TDO activo no estado em que o bit P é fornecido.

A figura 10-27 identifica os pinos do circuito integrado que permitem injectar faltas nos dois circuitos de missão, *fa* e *fb*, enquanto a figura 10-28 indica somente os pinos necessários no circuito final, sem os pinos de ensaio.

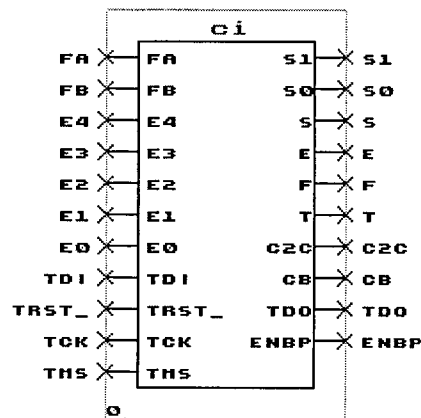


Figura 10-27. Pinos do CI utilizado na simulação

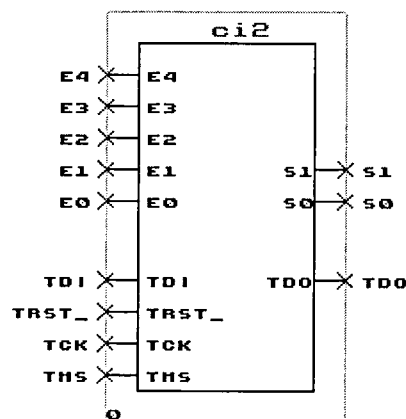


Figura 10-28. Pinos necessários no CI final

10.3.1 Esquemas dos blocos de paridade

A figura 10-29 representa o circuito que gera o sinal de paridade série (PARDSR), enquanto no diagrama da figura 10-30 se mostra como este circuito determina a paridade da sequência enviada, sendo reinicializado no estado Update.

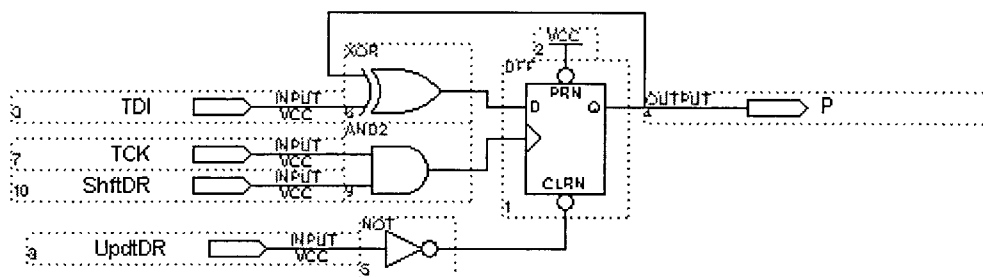


Figura 10-29. Circuito de geração de paridade série (PARDSR)

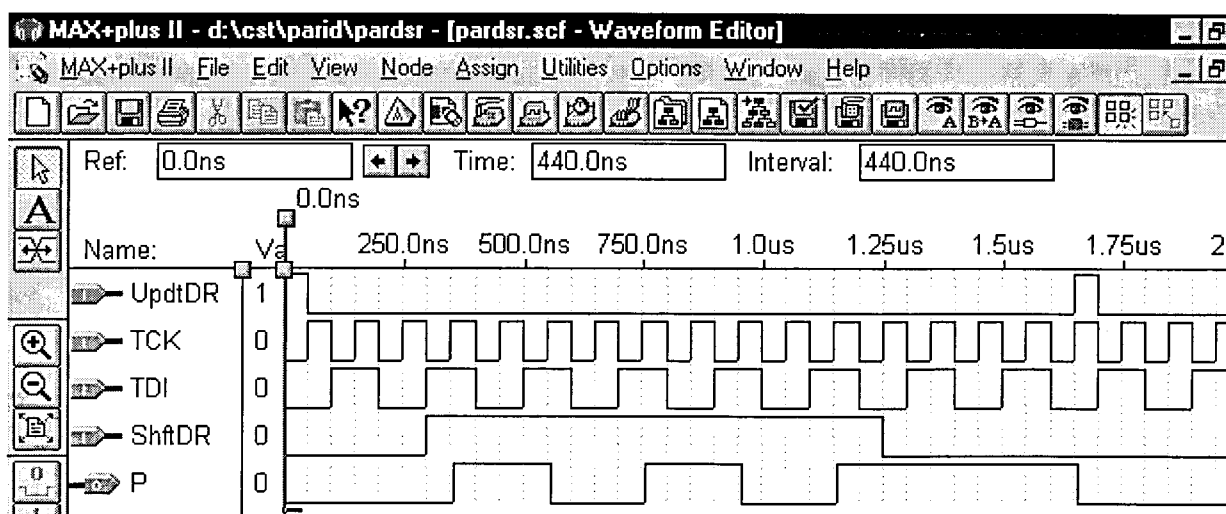


Figura 10-30. Ciclo temporal correspondente

A figura 10-31 representa o circuito ENABLP, que tem por função manter a saída TDO activa durante o estado EXIT1 (DR ou IR), permitindo fornecer o bit de paridade, nas condições descritas no capítulo sete.

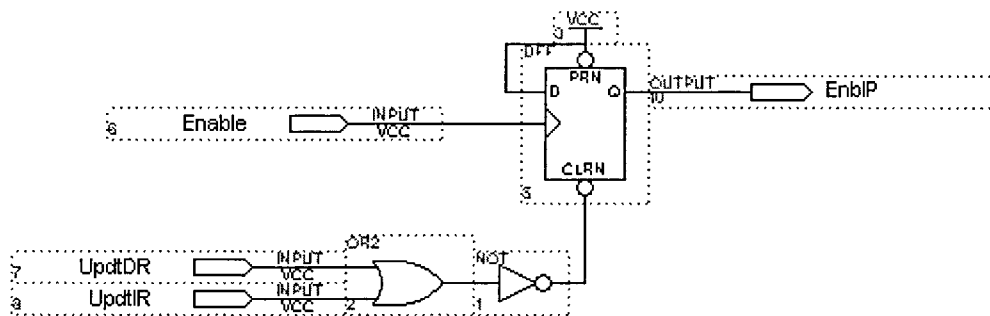


Figura 10-31. Circuito que prolonga o sinal ENABLE (ENABLP)

10.3.2 Diagramas temporais

Os três diagramas seguintes representam o comportamento do circuito integrado XMR com paridade, sem faltas e com uma falta injectada, separadamente, nos circuitos A e B. Tal como no anexo B, utilizámos uma série de vectores que inclui os 4 vectores de teste referidos no capítulo 5, repetida para facilitar a visualização, e que alterna com sinais aleatórios. Destacamos para os três casos, os aspectos mais relevantes.

1 - Sem faltas (alterações mais significativas):

- 150 ns - Primeira sequência de vectores VT_{0-3} , para referência.
- 500 ns - Início de varrimento do vector $VT_3 = 11.10101$ ($VT_0 + VT_1$). O primeiro bit deslocado por TDO indica que não ocorreu captura prévia.
- 1,5 μs - Fim de varrimento e indicação de Paridade ímpar (correspondente a cinco “1”).
- 1,6 μs - Sinal *enbp* (abreviatura de ENABLP) é desactivado e TDO retorna a alta impedância.
- 1,8 μs - Ocorre a detecção e a captura é activada por C2C. CB é desactivado e *enbp* activado.
- 2,8 μs - Início de varrimento do vector $VT_1 = 00.11110$. O primeiro bit deslocado por TDO indica que ocorreu captura.
- 3,5 μs - Indicação de paridade par (correspondente a quatro “1”).

Notemos, finalmente, nos diagramas das figuras 10-32, 33 e 34, que independentemente da existência de faltas no circuito de missão, o sinal de paridade continua a permitir controlar a informação deslocada pelo BS μ C, como é esperado.

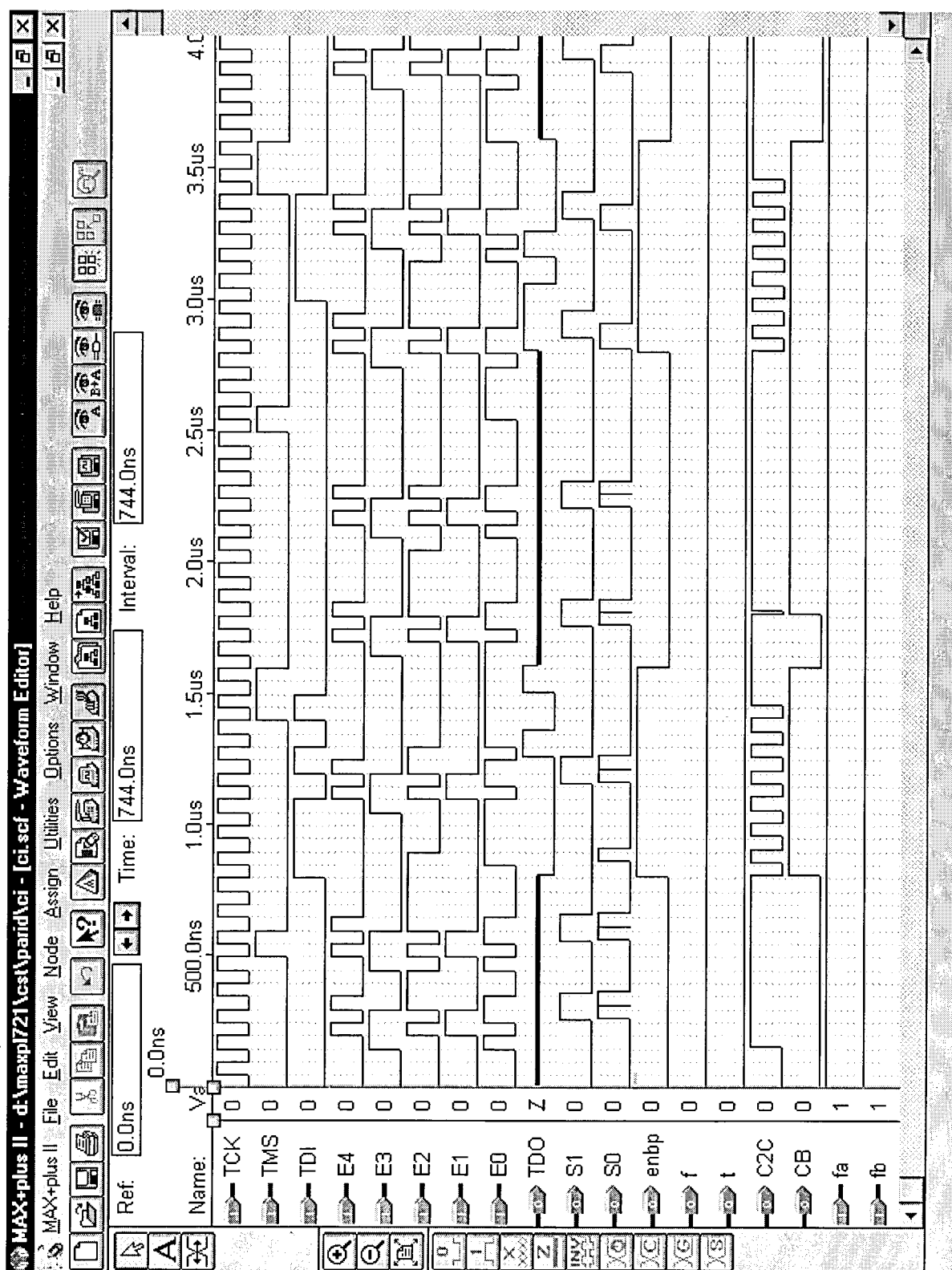


Figura 10-32. Diagrama temporal do CI sem faltas

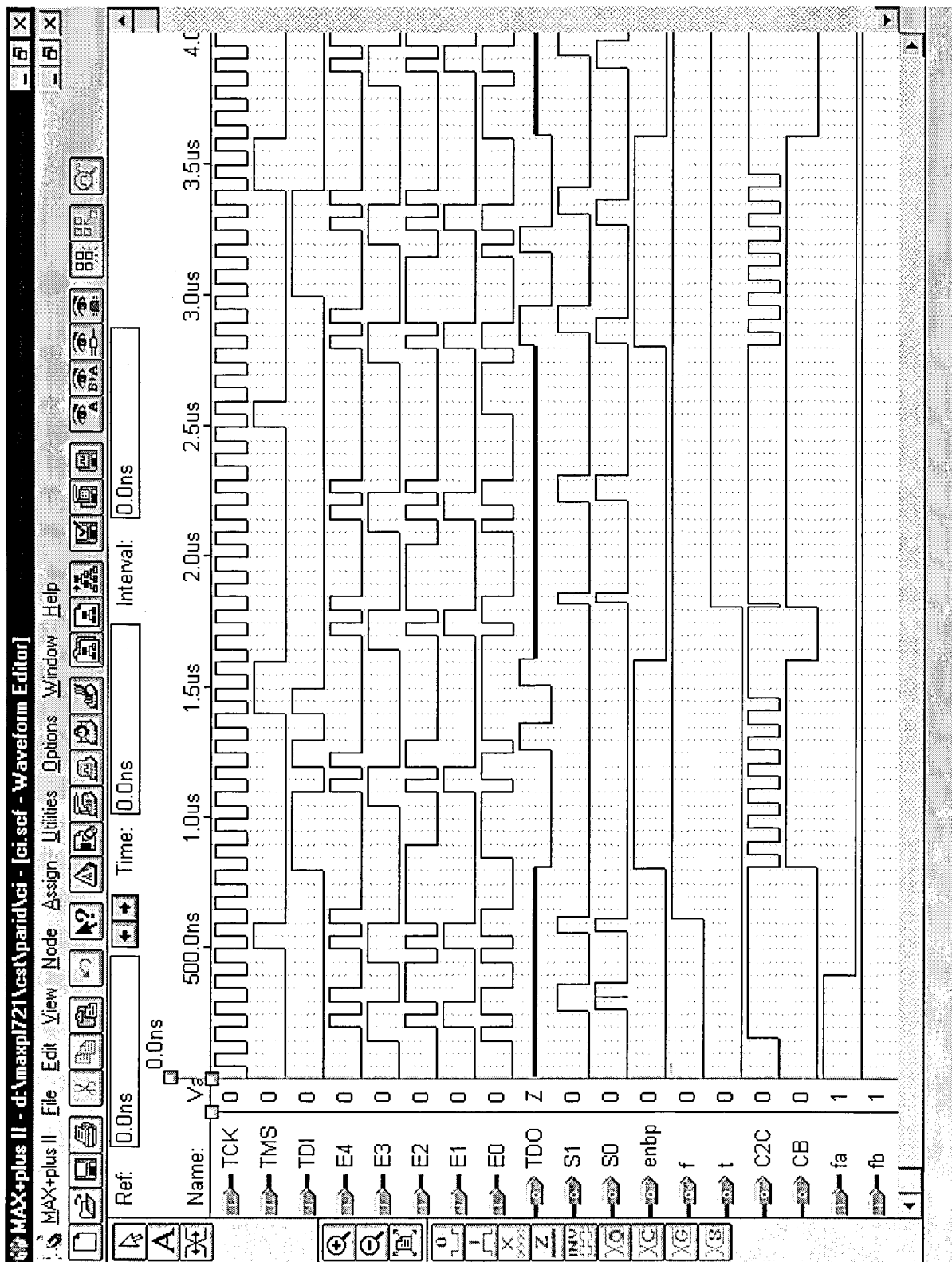


Figura 10-33. Diagrama temporal do CI com a falta *fa*

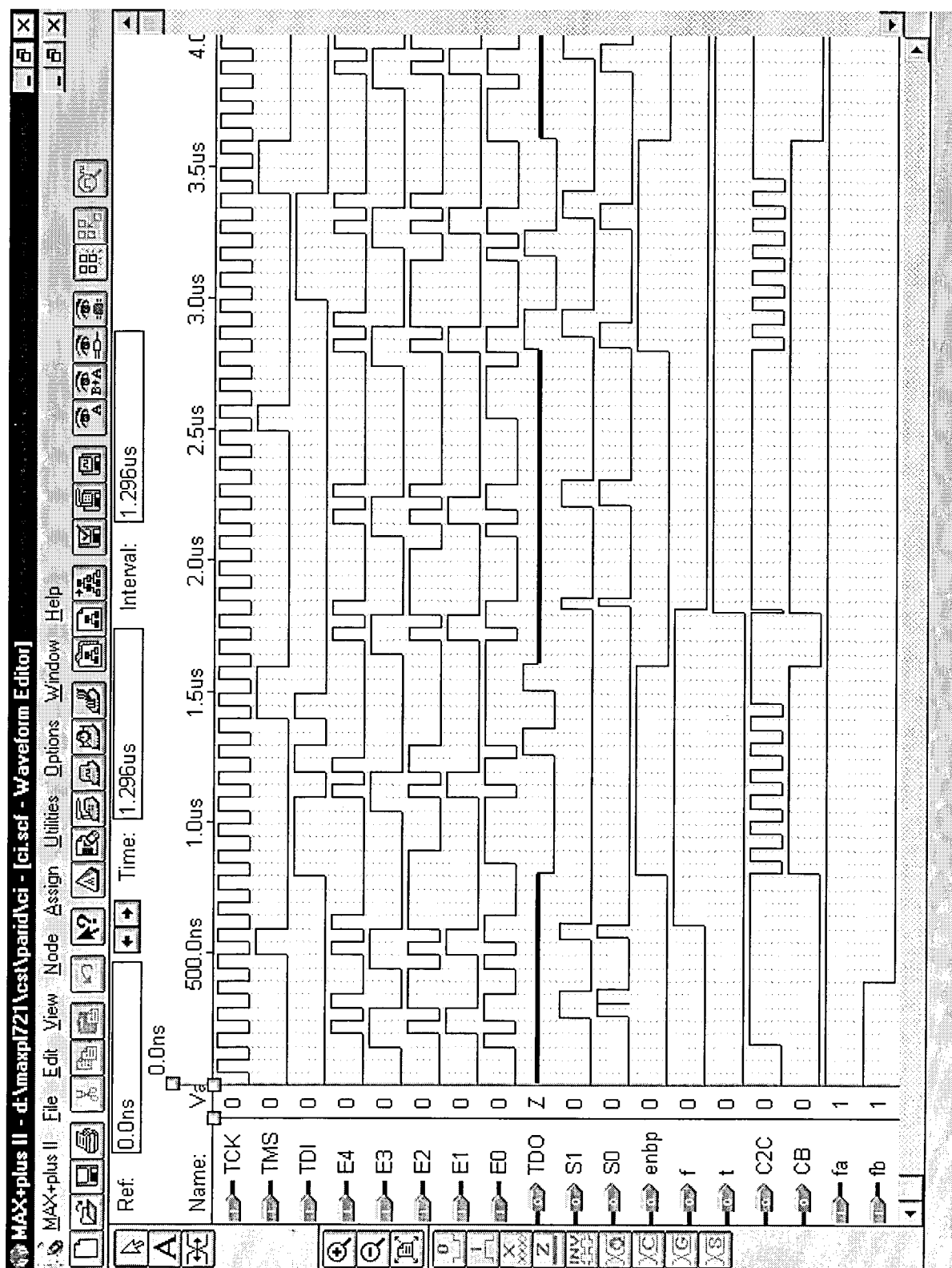


Figura 10-34. Diagrama temporal do CI com a falta *fb*

10.3.3 Relatório de projecto gerado pelo compilador do MAX+PLUS II

Este relatório é aqui apresentado de forma sintética, no sentido de facultar ao leitor mais interessado alguns pormenores da implementação em FPGA, nomeadamente a percentagem de ocupação e a forma como são gerados os sinais relevantes. No CD-ROM, existem os restantes relatórios com ênfase para o que descreve o comportamento temporal.

Project Information

*** FILE HIERARCHY ***

d:\cst\parid\ci.rpt

Project Information

MAX+plus II Compiler Report File
Version 7.21 Student Edition
Compiled: 06/23/99 12:47:13

Copyright (C) 1991-1997 Altera Corporation

Any megafunction design, and related net list (encrypted or decrypted), support information, device programming or simulation file, and any other associated documentation or information provided by Altera or a partner under the Megafunction Partnership Program may be used only to program Altera's Megafunction Partnership Program may be used only to program FPD devices (but not masked PLD devices) from Altera. Any other use of such megafunction design, net list, support information, device programming or simulation file, or any other related documentation or information is prohibited for any other purpose, including, but not limited to, modification, reverse engineering, de-compiling, or use with any other silicon devices, unless such use is explicitly licensed under a separate agreement with Altera or a megafunction partner. Title to all intellectual property, including patents, copyrights, trademarks, trade secrets, and other rights, embodied in any such megafunction design, net list, support information, device programming or simulation file, or any other related documentation or information provided by Altera or a partner, remains with Altera, the megafunction partner, or their respective licensors. No other licenses, including any licenses needed under any third party's intellectual property, are provided herein.

***** Project compilation was successful

*** DEVICE SUMMARY ***

Chip/ P/F	Device	Input Pins	Output Pins	Bidir Pins	Shareable LCS	Expanders	% Utilized
74F00	EPW7128SLC84-7	11	10	0	55	37	42 %
74F00	User Pins:	11	10	0			

d:\cst\parid\ci.rot

Project Information

** AUTO GLOBAL SIGNALS **

INFO: Signal 'TCK' chosen for auto global Clock

[illegible]

Device-Specific Information:

d:\cst\parid\ci.rpt

```

** OUTPUTS **

```

Pin	LAB	Primitive	Code	Total	Shared	p/a	INP	OUT	FBK	Name
52	LC	E	FF	5	4	0	4	9	1	CB
73	15	H	OUTUT	5	4	1	6	15	0	C2C
64	99	G	OUTUT	2	0	0	6	5	1	e
45	67	E	FF	2	0	0	0	1	4	entp
70	109	G	FF	2	0	0	0	1	4	f
54	183	F	OUTUT	7	5	1	7	6	0	S0
71	183	F	OUTUT	7	5	1	7	6	0	S0
74	177	H	OUTUT	2	0	0	0	3	0	S0
58	91	F	FF	10	5	0	5	6	1	lgcret
44	65	E	TRI	0	0	0	0	5	0	TIO

Code:

```
s = Synthesized pin or logic cell
t = Turbo logic cell
+ = Synchronous flipflop
/ = Slow slew-rate output
! = NOT gate push-back
r = Fitter-inserted logic cell
```

Device-Specific Information:

d:\cst\parid\ci.rpt

*** BURIED LOGIC ***

Pin	LC	LAB	Primitive	Code	Shareable		Fan-In		Fan-Out		Name			
					Total	Shared	n/a	INF	FBK	OUT		FBK		
--	116	H	DFFE	t	1	1	0	1	6	1	0	[cb:70][SO		
([cb:70][11])	--	--	--	--	--	--	--	--	--	--	--	--		
([cb:70][11])	--	122	H	SOFT	s	t	1	0	1	5	6	1	2	[cb:70][C20-1
([cb:70][36-1])	--	--	--	--	--	--	--	--	--	--	--	--	--	--
([cb:70][36-1])	--	120	H	SOFT	s	t	1	0	1	5	6	1	2	[cb:70][C20-2
([cb:70][36-2])	--	--	--	--	--	--	--	--	--	--	--	--	--	--
([cb:70][36-2])	--	118	H	SOFT	s	t	1	0	1	5	6	1	2	[cb:70][C20-3
([cb:70][36-3])	--	--	--	--	--	--	--	--	--	--	--	--	--	--
([cb:70][36-3])	--	127	H	SOFT	s	t	1	0	1	5	6	1	2	[cb:70][C20-4
([cb:70][36-4])	--	--	--	--	--	--	--	--	--	--	--	--	--	--
([cb:70][36-4])	--	119	H	SOFT	s	t	1	0	1	5	6	1	2	[cb:70][C20-5
([cb:70][36-5])	--	--	--	--	--	--	--	--	--	--	--	--	--	--
([cb:70][36-5])	--	87	F	SOFT	s	t	1	0	1	3	4	1	0	[cb:70][41-1
([cb:70][36-5])	--	78	E	DFFE	t	1	1	0	2	5	0	2		
[grpin5:36]bscset:17[SO	([grpin5:36]bscset:17)[11]	DFFE	t	0	0	0	0	1	5	6	16			
[grpin5:36]bscset:17[12	DFFE	t	0	0	0	0	0	1	5	6	16			
[grpin5:36]bscset:17[12	DFFE	t	0	0	0	0	0	1	5	6	16			
[grpin5:36]bscset:23[SO	([grpin5:36]bscset:23)[11]	DFFE	t	1	1	0	2	5	0	2				
[grpin5:36]bscset:23[SO	([grpin5:36]bscset:23)[11]	DFFE	t	0	0	0	0	1	5	6	18			
[grpin5:36]bscset:23[12	DFFE	t	1	1	0	2	5	0	2					
[grpin5:36]bscset:23[12	DFFE	t	1	1	0	2	5	0	2					
[grpin5:36]bscset:24[SO	([grpin5:36]bscset:24)[11]	DFFE	t	1	1	0	2	5	0	2				
[grpin5:36]bscset:24[12	DFFE	t	0	0	0	0	0	1	5	5	17			
[grpin5:36]bscset:24[12	DFFE	t	0	0	0	0	0	1	5	5	17			
[grpin5:36]bscset:24[12	DFFE	t	0	0	0	0	0	1	5	5	17			
[grpin5:36]bscset:25[SO	([grpin5:36]bscset:25)[11]	DFFE	t	1	1	0	3	4	0	2				
[grpin5:36]bscset:25[SO	([grpin5:36]bscset:25)[11]	DFFE	t	0	0	0	0	1	5	5	15			
[grpin5:36]bscset:25[12	DFFE	t	0	0	0	0	0	1	5	5	15			

Copy:

```
s = Synthesized pin or logic cell
t = Turbo logic cell
+ = Synchronous flipflop
/ = Slow slew-rate output
! = NOT gate push-back
r = Fitter-inserted logic cell
```

[illegible]

Device-Specific Information:
ci
d:\cst\parid\ci.rpt

** LOGIC CELL INTERCONNECTIONS **

Logic Array Block 'E':

```

Logic cells placed in LAB 'E'
LC80 CB
LC67 entp
LC78 grpins:36|bscdet:17|SO
LC68 grpins:36|bscdet:23|:12
LC69 tap:38|tap:1:20|:1
LC66 tap:38|tap:1:20|B
LC70 tap:38|tap:1:20|C
LC73 tap:38|tap:1:20|SELECT
LC65 TDO

Other LABs fed by signals
that feed LAB 'E'
A B C D E F G H
LC68 -> * * * * *
LC66 -> * * * * *
LC69 -> * * * * *
LC70 -> * * * * *
LC73 -> * * * * *

Logic cells that feed LAB 'E':
A B C D E F G H
LC68 -> * * * * *
LC66 -> * * * * *
LC69 -> * * * * *
LC70 -> * * * * *
LC73 -> * * * * *

Pin
10 -> * * * * *
11 -> * * * * *
12 -> * * * * *
13 -> * * * * *
14 -> * * * * *
15 -> * * * * *
16 -> * * * * *
17 -> * * * * *
18 -> * * * * *
19 -> * * * * *
20 -> * * * * *
21 -> * * * * *
22 -> * * * * *
23 -> * * * * *
24 -> * * * * *
25 -> * * * * *
26 -> * * * * *
27 -> * * * * *
28 -> * * * * *
29 -> * * * * *
30 -> * * * * *
31 -> * * * * *
32 -> * * * * *
33 -> * * * * *
34 -> * * * * *
35 -> * * * * *
36 -> * * * * *
37 -> * * * * *
38 -> * * * * *
39 -> * * * * *
40 -> * * * * *
41 -> * * * * *
42 -> * * * * *
43 -> * * * * *
44 -> * * * * *
45 -> * * * * *
46 -> * * * * *
47 -> * * * * *
48 -> * * * * *
49 -> * * * * *

```

* = The logic cell or pin is an input to the logic cell (or LAB) through the PIA.
- = The logic cell or pin is not an input to the logic cell (or LAB).

Device-Specific Information:
ci
d:\cst\parid\ci.rpt

** LOGIC CELL INTERCONNECTIONS **

Logic Array Block 'F':

```

Logic cells placed in LAB 'F'
LC87 cb:70|:41:1
LC88 grpins:36|bscdet:23|SO
LC94 grpins:36|bscdet:24|SO
LC89 grpins:36|bscdet:24|:12
LC93 grpins:36|bscdet:25|SO
LC82 grpins:36|bscdet:25|:12
LC85 ir:39|ircell:39|SO
LC96 ir:39|ircell:40|SO
LC86 ir:39|ircell:41|SO
LC90 ir:39|ircell:41|Po
LC81 ir:39|ircell:42|SO
LC84 lgrdet:63|:20-1
LC92 pdrdet:67|P
LC83 s
LC91 t

Other LABs fed by signals
that feed LAB 'F'
A B C D E F G H
LC94 -> * * * * *
LC89 -> * * * * *
LC93 -> * * * * *
LC82 -> * * * * *
LC96 -> * * * * *
LC86 -> * * * * *
LC90 -> * * * * *
LC81 -> * * * * *
LC84 -> * * * * *
LC92 -> * * * * *
LC83 -> * * * * *
LC91 -> * * * * *

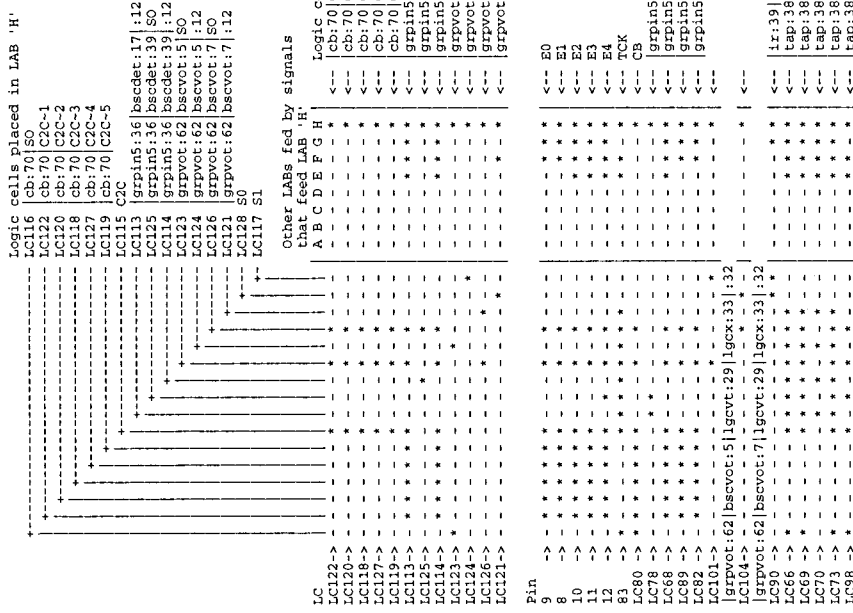
Logic cells that feed LAB 'F':
A B C D E F G H
LC87 -> * * * * *
LC88 -> * * * * *
LC94 -> * * * * *
LC89 -> * * * * *
LC93 -> * * * * *
LC82 -> * * * * *
LC85 -> * * * * *
LC96 -> * * * * *
LC86 -> * * * * *
LC90 -> * * * * *
LC81 -> * * * * *
LC84 -> * * * * *
LC92 -> * * * * *
LC83 -> * * * * *
LC91 -> * * * * *

Pin
9 -> * * * * *
10 -> * * * * *
11 -> * * * * *
12 -> * * * * *
13 -> * * * * *
14 -> * * * * *
15 -> * * * * *
16 -> * * * * *
17 -> * * * * *
18 -> * * * * *
19 -> * * * * *
20 -> * * * * *
21 -> * * * * *
22 -> * * * * *
23 -> * * * * *
24 -> * * * * *
25 -> * * * * *
26 -> * * * * *
27 -> * * * * *
28 -> * * * * *
29 -> * * * * *
30 -> * * * * *
31 -> * * * * *
32 -> * * * * *
33 -> * * * * *
34 -> * * * * *
35 -> * * * * *
36 -> * * * * *
37 -> * * * * *
38 -> * * * * *
39 -> * * * * *
40 -> * * * * *
41 -> * * * * *
42 -> * * * * *
43 -> * * * * *
44 -> * * * * *
45 -> * * * * *
46 -> * * * * *
47 -> * * * * *
48 -> * * * * *
49 -> * * * * *

```

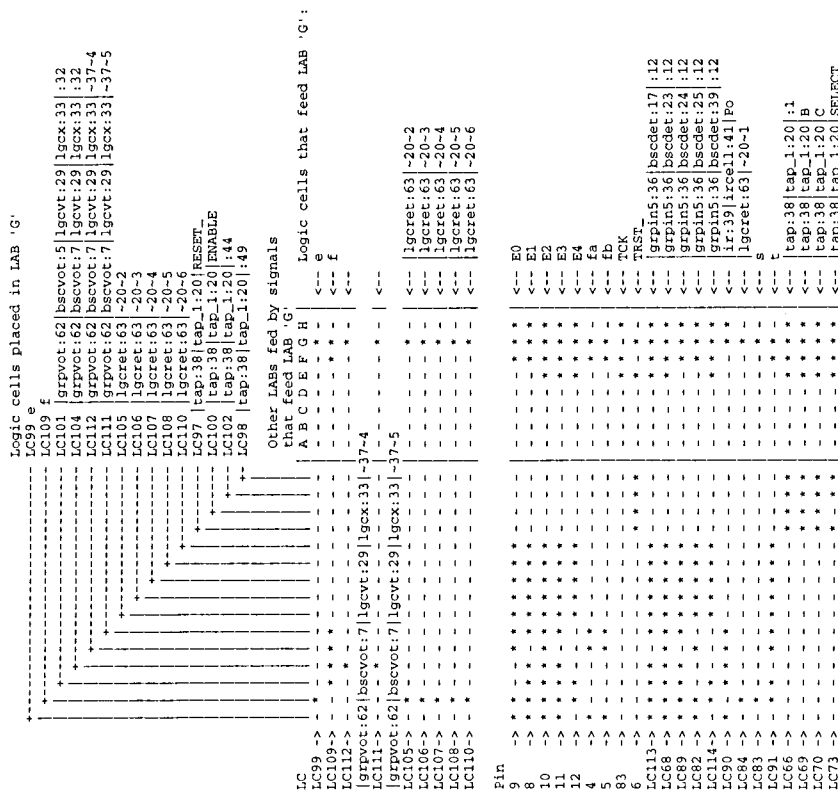
* = The logic cell or pin is an input to the logic cell (or LAB) through the PIA.
- = The logic cell or pin is not an input to the logic cell (or LAB).

Device-Specific Information:
ci
d:\est\parid\ci.rpt
** LOGIC CELL INTERCONNECTIONS **
Logic Array Block 'H':



* = The logic cell or pin is an input to the logic cell (or LAB) through the PIA.
- = The logic cell or pin is not an input to the logic cell (or LAB).

Device-Specific Information:
ci
d:\est\parid\ci.rpt
** LOGIC CELL INTERCONNECTIONS **
Logic Array Block 'G':



* = The logic cell or pin is an input to the logic cell (or LAB) through the PIA.
- = The logic cell or pin is not an input to the logic cell (or LAB).

d:\cst\parid\ci.rpt

*** EQUATIONS ***

```
: INPUT;  
E0 : INPUT;  
E1 : INPUT;  
E2 : INPUT;  
E3 : INPUT;  
E4 : INPUT;  
fa : INPUT;  
fb : INPUT;  
fck : INPUT;  
fdi : INPUT;  
tms : INPUT;  
rst : INPUT;
```

```
-- Node name is 'CB' = |cb:70|CB'
```

```

-- Equation name is 'CB', type is output
CB = DFFE( GND $ VCC, _EQ01, !_EQ02, VCC, VCC );

```

```
-EX001 = !LC097 & _X001 & _X002 & _X003 & _X004 & _X005;  
_X001 = EXP(E2) & LC068 & LC098;  
_X002 = EXP(E3) & LC098 & LC113;  
_X003 = EXP(E3) & LC098 & LC113;  
_X004 = EXP(E4) & LC098 & LC114;  
_X005 = EXP(E4) & LC098 & LC114;  
-EX002 = !LC066 & !LC069 & LC070 & !LC073 & !TCCK;
```

```
-- Node name is 'C2C'
```

```

- Equation name is 'C2C', location is LC115, type is output.

```

[illegible]

- Node name is 'e'

```
- Equation name is 'e', location is LC099, type is output.
```

```

e      _EQU05 = % GND;
      _EQU05 = fa & !fb & !LC068 & !C082 & !C090 & !X010;
      # fa & fb & !C068 & !C082 & !C090 & !X010
      # E0 & E2 & fa & !fb & !C090 & !X011
      # E0 & E2 & !fa & fb & !C090 & !X011;
      _X010 = EXP( !_C089 & !_LC113);
      _X011 = EXP( !E1 & !E3);

```

```
- Node name is 'enbp' = '|enablp:68|Enb1P|
```

```
- Equation name is 'enbp', type is output
enbp = DFFE( GND $ VCC, _LC100, !_EQ006, VCC, VCC);
_EQ006 = _LC066 & !_LC069 & _LC070 & !TC;
```

```
- Node name is 'f' = 'lgcret:63|f'
```

```

Equation name is 'F', type is output
f = DEFF(1, $ VCC F0007 VCC

```

```

-EQ007 = !_LC084 & !_LC105 & !_LC106 & !_LC107 & !_LC108 & !_LC110 & !_X012 &
        !_X013;

```

[illegible]

[illegible]

```
-- Node name is '[tap:38|tap:1:20]:44'
-- Equation name is '_LC102_', type is buried
_LC102 = DFFE( _EQ071 $ VCC, GLOBAL(TCK), TRST_, VCC, VCC);
_EQ071 = !_LC066 & !_LC069 & !_LC070 & !_LC073;

-- Node name is '[tap:38|tap:1:20]:49'
-- Equation name is '_LC098_', type is buried
_LC098 = DFFE( _EQ072 $ VCC, GLOBAL(TCK), TRST_, VCC, VCC);
_EQ072 = !_LC066 & !_LC069 & !_LC070 & !_LC073;

-- Shareable expanders that are duplicated in multiple LABS:
-- _X025 occurs in LABS ----EF-H

Project Information
** COMPILATION SETTINGS & TIMES **
Processing Menu Commands
-----
Design Doctor
Logic Synthesis:
Synthesis Type Used
= off
= Standard
= NORMAL
Default synthesis style
Logic option settings in 'NORMAL' style for 'MAX7000S' family
DECOMPOSE GATES
DUPLICATE LOGIC EXTRACTION
MINIMIZATION
MULTI LEVEL FACTORING
NOT_GATE_PUSH_BACK
PARALLEL EXPANDERS
REDUCE LOGIC
REFACTORIZATION
REGISTER OPTIMIZATION
RESYNTHESIZE_NETWORK
SLOW_SLEW_RATE
SOFT_BUFFER_INSERTION
SUBFACTOR_EXTRACTION
TURBO_BIT
XOR_SYNTHESIS
UNKNOWN_SIGNAL_BUFFERS
USE_APL_FOR_AND_OPERATORS
Other logic synthesis settings:
Automatic Global Clock
Automatic Global Clear
Automatic Global Preset
Automatic Global Output Enable
Automatic I/O Cell Registers
Automatic Register Packing
Automatic Open-Brain Pins
Automatic Implement in EAB
One-Hot State Machine Encoding
Optimize
Default timing Specifications: None
Cut All Bidir Feedback Timing Paths
Cut All Clear & Preset timing Paths
Ignore Timing Assignments
Functional SNF Extractor
```

```

Linked SNF Extractor      = off
Timing SNF Extractor     = on
Optimize Timing SNF      = off
Generate AHDL TDO File   = off
Filter Settings          = NORMAL
Smart Recompile          = off
Total Recompile          = off

```

```

Interface Menu Commands
-----

```

```

EDiF Netlist Writer      = off
Verilog Netlist Writer   = off
VHDL Netlist Writer      = off

```

```

Compilation Times
-----
Compiler Netlist Extractor  00:00:04
Database Builder            00:00:01
Logic Synthesizer           00:00:06
Partitioner                  00:00:01
Filter                       00:00:01
Timing SNF Extractor        00:00:04
Assembler                   00:00:02
Total Time                   00:00:00
-----
00:00:18

```

```

Memory Allocated
-----
Peak memory allocated during compilation = 3.278K

```