Síntese de Circuitos Conformes com o Standard Boundary Scan

Ana Antunes, Meryem Marzouki, António Ferrari

***Resumo*- Este artigo apresenta uma ferramenta que permite a inclusão de facilidades de teste ao nível da descrição VHDL de um circuito integrado. A lógica de teste incluida respeita o standard Boundary Scan.**

***Abstract*- This paper presents a tool that allows the inclusion of the Boundary Scan test logic in a VHDL description of a given integrated circuit.**

I. INTRODUÇÃO

A síntese de alto nível tem ocupado um grande número de investigadores nos últimos anos. Como resultado do trabalho desenvolvido nessa área existem já sistemas de síntese automáticos que partem de descrições comportamentais dos circuitos.

O interesse deste tipo de sistemas é justificado uma vez que eles permitem uma diminuição significativa do tempo de concepção dos circuitos. Por outro lado a massificação da produção de circuitos integrados leva à crescente sensibilização dos responsáveis pela concepção para o desenvolvimento de circuitos tendo em vista o teste e a consequente inclusão dos elementos necessários à realização da estratégia de teste escolhida.

Neste momento os sistemas de síntese de alto nível disponíveis não tomam em conta o problema do teste dos circuitos.

É neste contexto que surge este trabalho que se propõe desenvolver uma estratégia para a realização de síntese de circuitos conformes com o standard Boundary Scan.

II. O STANDARD BOUNDARY SCAN

O standard *Boundary Scan*, IEEE 1149.1-1990 foi desenvolvido com o objectivo de facilitar o teste de circuitos integrados, placas e sistemas. Este standard fornece um conjunto de regras que permitem o tratamento do problema de teste de uma forma estruturada. O standard BS (*Boundary Scan*) é flexível, no sentido em que o responsável pela concepção pode criar modos de operação diferentes daqueles que existem já definidos de forma a poder realizar operações úteis para o teste de um dado circuito em particular.

Associada à definição da arquitectura BS existe uma linguagem de descrição que permite a especificação da lógica utilizada para o teste. Essa linguagem chama-se BSDL - Boundary Scan Description Language.

A arquitectura BS é constituida por vários elementos. Esses elementos são: pinos de teste, registos e um controlador. A figura 1 representa um diagrama de blocos da arquitectura *Boundary Scan*.

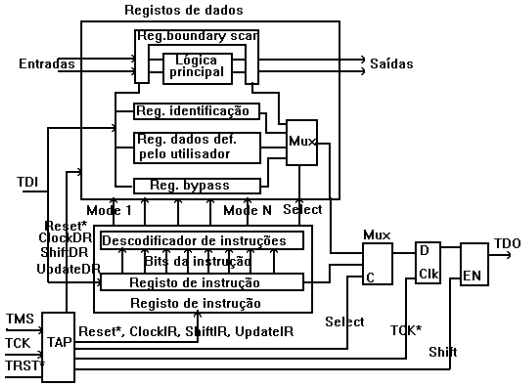

Figura 1- Diagrama de blocos da arquitectura *Boundary Scan*

A. Pinos de teste

Existem 4 pinos obrigatórios e um pino opcional. Estes pinos vão permitir aceder à lógica BS dentro do circuito para efectuar a sua programação e para observar os resultados dos testes aplicados. Os pinos obrigatórios são:

TCK - Test Clock (entrada - relógio específico para teste)

TMS - Test Mode Select (entrada - permite seleccionar o modo de teste)

TDI - Test Data In (entrada - dados de teste)

TDO - Test Data Out (saída - resultados do teste)

O quinto pino é opcional:

TRST* - Test Reset (entrada - reset assíncrono)

Os pinos TDI e TDO são respectivamente a entrada/saída da cadeia de *scan*.

B. Controlador TAP

O controlador TAP é uma máquina de estados finitos com 16 estados. É responsável pela interpretação dos sinais exteriores que controlam a lógica BS e gera os sinais que controlam toda a lógica BS.

O controlador TAP é controlado pelos sinais TCK, TMS e TRST* (se existir). Este elemento é totalmente descrito pelo standard *Boundary Scan* [1].

C. Registos

Existem dois tipos de registos: o registo de instrução e o registos de dado.

O registo de instrução é um registo obrigatório. É o conteúdo deste registo que define o modo de teste e qual o registo de dados que vai ser manipulado. Cada uma das células deste registo é constituida por um *flip-flop* para fazer o deslocamento dos dados e uma *latch* que contém a instrução actual. Este registo tem um tamanho mínimo de duas células.

O standard define 3 registos de dados dos quais um é opcional.

Os registos de dados obrigatórios são dois, o registo *boundary scan* e o registo *bypass*.

O registo *boundary scan* controla e observa as entradas e saídas da lógica principal. O standard define vários tipos de configurações para as células deste registo [1].

O registo *bypass* faz o curto-circuito da cadeia de *scan*. Este registo é utilizado sobretudo quando se pretende fazer o teste de um único circuito numa placa (teste interno).

O único registo opcional definido pelo standard é o registo de identificação. Este registo contém informação sobre a identidade do dispositivo sob teste.

Podem ainda criar-se novos registos de dados para utilizar em conjunto com novos modos de operação, caso seja necessário. As regras para a definição destes registos são fornecidas pelo standard.

D. Modos de operação

O standard define dois modos de operação que são escolhidos através de instruções pré-definidas.

No modo *non-invasive* os elementos da arquitectura especificados pelo standard são utilizados para comunicar de modo assíncrono com o mundo exterior. No entanto a lógica de teste não tem influência sobre o funcionamento da lógica principal do circuito. Pode utilizar-se este modo para carregar os vectores de teste e as instruções de teste desejadas e para observar os resultados do teste.

As instruções definidas pelo standard para este modo de operação são:

- BYPASS (obrigatória) - A função desta instrução é carregar o registo *bypass* entre os pinos TDI e TDO, fazendo o curto-circuito da cadeia de *scan*. A sequência binária com todos os bits a ‘1’ deve obrigatoriamente descodificar esta instrução, podendo no entanto existir outras sequências com a mesma função.

- SAMPLE/ PRELOAD (obrigatória) - Esta instrução tem duas funções, uma de amostragem durante a qual todos os *flip-flops* do registo *boundary scan* são carregados com os valores das entradas e saídas da lógica principal, e uma função de armazenamento que permite carregar nos *flip-flops* do registo *boundary scan* novos dados provenientes da entrada TDI. Estes dados são depois colocados nas *latches* do mesmo registo durante a passagem do controlador TAP por um estado subsequente.

- IDCODE (opcional) - Esta instrução vai actuar sobre o registo de identificação, carregando-o com o código de identificação do dispositivo sob teste, código esse que será depois deslocado para o exterior através da cadeia de *scan.* A codificação desta instrução não é definida pelo standard.

- USERCODE (opcional) - O objectivo desta instrução é alargar a instrução IDCODE ao caso (por exemplo) dos circuitos integrados programáveis, para os quais a instrução IDCODE não é suficiente para identificar o dispositivo e a sua programação. A codificação desta instrução não está definida no standard.

No modo *pin-permission* podem-se controlar os pinos de entrada e de saída do circuito. Este modo permite o teste da lógica principal do circuito assim como o isolamento dessa lógica em relação à actividade nos pinos a que está habitualmente ligada.

- EXTEST (obrigatória) - Com esta instrução podem-se amostrar as entradas e controlar as saídas da lógica principal do circuito. O standard BS define um código obrigatório para esta instrução podendo no entanto existir outros códigos para esta mesma instrução. O código obrigatório é a sequência binária com todos os bits a ‘0’. Esta instrução actua sobre o registo *boundary scan*.

- INTEST (opcional) - Esta instrução permite controlar as entradas e amostrar as saídas do circuito. O standard não define nenhum código para esta instrução. INTEST actua sobre o registo *boundary scan.*

- RUNBIST (opcional) - RUNBIST permite o acesso às facilidades BIST (Built-In Self Test) do circuito (se existirem) de uma forma estruturada. A codificação desta instrução não está definida no standard. O registo sobre o qual esta instrução vai actuar é definido pelo utilizador. Esse registo serve para recolher o resultado do teste BIST.

- HIGHZ (opcional) - Esta instrução coloca os pinos de saída e os pinos bidireccionais em alta impedância. A codificação não é definida pelo standard. Esta instrução coloca o registo *bypass* entre os pinos TDI e TDO.

- CLAMP (opcional) - CLAMP permite forçar um valor fixo nos pinos de saída do circuito. Os valores impostos provêm do registo *boundary scan*. A codificação desta instrução não é definida pelo standard. O registo *bypass* é colocado entre os pinos TDI e TDO quando esta instrução está activa.

O standard permite ao utilizador a definição de outras instruções.

III. ARQUITECTURA BOUNDARY SCAN PROPOSTA

A ferramenta desenvolvida propõe duas opções uma procede só à inclusão da lógica BS obrigatória, a outra, mais flexível, permite ao utilizador a escolha dos registos e instruções a incluir (incluindo sempre a lógica BS obrigatória).

A. Arquitectura obrigatória

A arquitectura que designaremos por obrigatória é constituída pelo conjunto das estruturas de teste obrigatórias que definem a arquitectura BS mínima.
Os elementos desta arquitectura são:

- os pinos de teste TCK, TMS, TDI e TDO.
- o controlador TAP.
- registo *bypass*.
- registo *boundary scan* com um número de células igual ao número de pinos de entrada / saída da descrição inicial.
- registo de instrução com 2 bits, que permite a codificação das 3 instruções obrigatórias (EXTEST, SAMPLE/ PRELOAD, BYPASS).

Os códigos atribuidos a cada uma destas instruções são:

- EXTEST - ‘00’ (obrigatório).
- SAMPLE/ PRELOAD - ‘10’
- BYPASS - ‘11’ (obrigatório) e ‘01’ (porque de acordo com o standard BS todos os códigos não utilizados devem descodificar BYPASS).

B. Arquitectura opcional

A arquitectura que designaremos por opcional permite ao utilizador escolher outras instruções e outros registos bem como a inclusão do pino TRST* para além da lógica de base (lógica obrigatória).
As instruções propostas são as 6 instruções opcionais definidas pelo standard e um número máximo de 7 instruções que podem ser definidas pelo utilizador às quais vai ser atribuído um registo de dados novo.
As 6 instruções opcionais já definidas são:

- INTEST - teste interno do circuito.
- RUNBIST - acciona o teste integrado BIST.
- IDCODE - fornece o código de identificação do circuito.
- USERCODE - identifica a programação do circuito.
- CLAMP - força os pinos de saída do circuito a um determinado valor pré-definido.
- HIGHZ - coloca as saídas do circuito no estado de alta impedância.

Os códigos destas instruções não são definidos pelo standard por isso foi fixado o código para cada uma delas. A codificação destas instruções é apresentada no ponto III.A.
O registo de instrução tem 4 bits de modo a poder codificar as 16 instruções propostas: 3 obrigatórias, 6 opcionais definidas pelo standard e no máximo 7 definidas pelo utilizador.
Os registos de dados são associados às instruções que os manipulam: o registo de identificação é criado quando a instrução IDCODE é escolhida e a cada instrução definida pelo utilizador é associado um novo registo de dados.
O registo de identificação tem um tamanho de 32 bits definido pelo standard e o código de identificação do circuito é especificado pelo utilizador.
O tamanho dos registos de dados associados às novas instruções é definido pelo utilizador.
À instrução RUNBIST é associado o registo *boundary scan*.

IV. DESCRIÇÃO DOS ELEMENTOS DE BASE DA ARQUITECTURA

Neste ponto apresentam-se as características fundamentais das células de base que foram descritas em VHDL, nível RTL (*Register Tranfer Logic*).

A. Controlador TAP

Este elemento é totalmente descrito pelo standard BS. O controlador tem 16 estados e as transições de estado fazem-se no flanco ascendente do relógio de teste TCK. Alguns destes estados actuam sobre o registo de instrução, outros sobre o registo de dados que é referido pela instrução corrente. Para mais pormenores sobre a função de cada estado e a descrição dos sinais de saída do controlador consultar [1].

B. Célula boundary scan

Esta célula é composta de 2 multiplexers e uma *latch*. O multiplexer de entrada selecciona entre os dados provenientes da entrada de *shift* (SI) ou os dados do pino ao qual a célula está ligada. O multiplexer de saída selecciona entre os dados provenientes do pino (PI) ou os

dados contidos na *latch* Upd. Na figura 2 pode ver-se a estrutura desta célula.

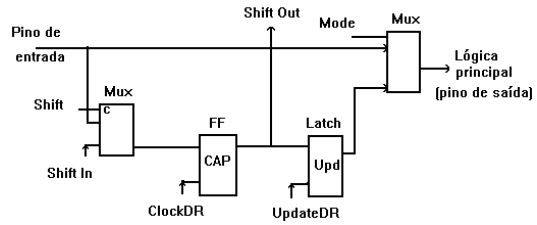

Figura 2 - Célula de base do registo *boundary scan*

C.Célula do registo de instrução

A célula do registo de instrução é composta por um multiplexer de entrada, um *flip-flop* e uma *latch* com um sinal de *preset* que serve para carregar as instruções IDCODE ou BYPASS durante o estado de *reset* do controlador. Ver figura 3.

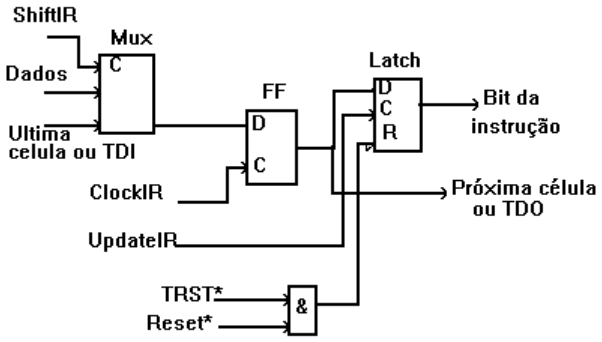

Figura 3 - Célula de base do registo de instrução

D.Descodificador do registo de instrução

Existem duas descrições diferentes deste elemento: uma fixa que descreve a descodificação das instruções obrigatórias, e outra gerada automaticamente de acordo com as instruções escolhidas pelo utilizador. A codificação das 16 instruções propostas é apresentada de seguida:

- EXTEST - ‘0000’ (obrigatório)
- IDCODE - ‘0001’
- USERCODE - ‘0011’
- INTEST - ‘0100’
- RUNBIST - ‘0101’
- HIGHZ - ‘0110’
- CLA.MP - ‘0111’
- I1 - ‘1000’
- I2 - ‘1001’
- I3 - ‘1010’
- I4 - ‘1011’
- I5 - ‘1100’
- I6 - ‘1101’
- I7 - ‘1110’
- BYPASS - ‘1111’ (obrigatório)

A descodificação das instruções da arquitectura obrigatória é feita de acordo com os códigos referidos no ponto II- *A*.

E. Célula do registo de identificação

Esta célula apresenta uma estrutura parecida com a do registo de instrução sendo no entanto mais simples pois não necessita do sinal de *preset* nem da *latch* Upd.

Esta célula foi também utilizada para os registos de dados associados às instruções definidas pelo utilizador.

F. Multiplexer de saída dos registos

Este multiplexer foi descrito sob a forma de um processo VHDL controlado por um dos sinais de saída do controlador TAP que informa qual dos registos deve ser colocado à saída.

A cada instante e de acordo com a instrução corrente, este multiplexer vai ligar a saída do registo conveniente à entrada do pino TDI.

G. Autorização do pino TDI

Este sinal só é activado quando a cadeia de *scan* está em modo *shift*. Este sinal foi descrito sob a forma de um processo VHDL.

H. Codificador do sinal mode

O sinal *mode* está presente em todas as células do registo BS e controla o multiplexer de saída dessas células.

A codificação da saída deste módulo depende da instrução corrente.

V. ESTRUTURA DA FERRAMENTA QUE GERA A DESCRIÇÃO VHDL FINAL

O programa desenvolvido foi descrito em linguagem C. Este programa aceita como entrada um ficheiro que contém a descrição VHDL do circuito inicial. A estrutura do programa desenvolvido é apresentada na figura 4.

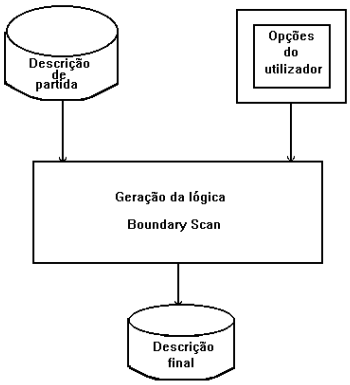

Figura 4 - Diagrama de blocos do programa desenvolvido

A primeira parte deste programa pede ao utilizador as suas opções para a arquitectura *boundary scan* a implementar e analisa a descrição do circuito de partida. A outra parte toma em conta as opções do utilizador previamente escolhidas e gera a descrição final de acordo com essas opções.

A interface deste programa é feita através de um conjunto de menus interligados que propõem as várias opções possíveis. A estrutura dos menus é apresentada na figura 5.

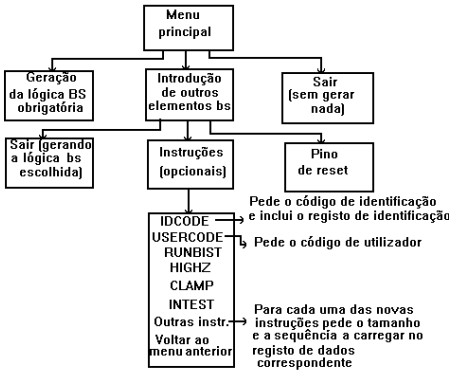

Figura 5 - Estrutura dos menus do programa

A geração da descrição VHDL final é feita por partes: primeiro é feita uma análise do ficheiro de partida e os dados referentes aos pinos de entrada e saída são copiados para uma estrutura de dados interna, depois o ficheiro de saída com a descrição VHDL do circuito e das facilidades *boundary scan* é gerado. Parte desse ficheiro é copiado do ficheiro original (a parte referente à descrição da lógica do circuito principal) e outra parte é inserida de acordo com a lógica *boundary scan* a incluir. O resultado de saída é colocado num ficheiro denominado *my_ic_BS*.

VI. COMPARAÇÃO DA FERRAMENTA DESENVOLVIDA COM A BIBLIOTECA DE TESTE SYNOPSYS

A ferramenta Synopsys permite fazer simulação lógica e síntese de descrições VHDL. Possui ainda comandos que permitem a inclusão de lógica *boundary scan* ao nível das portas lógicas. Associada a esta ferramenta existe uma biblioteca de células de teste e o utilizador pode escolher a arquitectura *boundary scan* que quer implementar através dos vários comandos disponíveis.

A ferramenta de síntese de circuitos conformes ao standard *Boundary Scan* apresenta uma filosofia diferente. A diferença situa-se no nível da descrição das células de base, que neste caso são descritas ao nível RTL (Register Transfer Logic) e que são sintetizadas ao mesmo tempo que o circuito de partida. Por seu lado a ferramenta Synopsys insere as células da arquitectura BS depois de ter feito a síntese do circuito inicial ou seja ao nível das portas lógicas. Para uma melhor compreensão das diferenças entre estes dois processos sugere-se a consulta da figura 6.

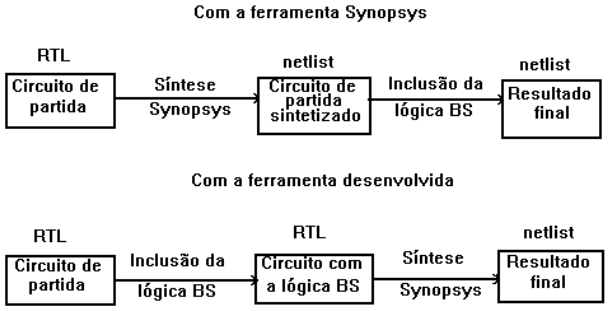

Figura 6 - Método utilizado para fazer a comparação

A. Apresentação dos resultados

Torna-se interessante a comparação em termos de área entre a lógica inserida através dos dois processos descritos.

As únicas comparações possíveis relacionam-se com algumas das células de base da arquitectura e com a área do circuito final (lógica principal + lógica de teste), isto porque só se pode aceder a algumas das células da biblioteca de teste Synopsys.

A tabela seguinte apresenta os valores (absolutos) da área das células comparadas e do circuito final.

	Método proposto	Synopsys
TAP	272.698	139.746
BS (1 célula)	39.092	36.256
ID (32 bits)	283.328	263.109
Circuito final	1168.689	846.136

Tabela 1- Comparação de área (valores absolutos) em portas equivalentes

A mesma tabela com os valores percentuais em relação à área das células da biblioteca de teste Synopsys.

	Método proposto
TAP	+ 95%
BS (1 célula)	+ 7.8%
ID (32 bits)	+ 7.7%
Circuito final	+ 38%

Tabela 2- Diferença de área em percentagem em relação aos valores obtidos com a ferramenta Synopsys

B. Análise dos resultados

Da análise das tabelas apresentadas conclui-se que a área das células que são descritas ao nível RTL e depois sintetizadas é maior que a área das células da biblioteca Synopsys. Os resultados não são surpreendentes se tivermos em conta o facto dos mecanismos de síntese de circuitos RTL obedecem a critérios gerais baseados nas primitivas da linguagem utilizadas e o facto das células da biblioteca Synopsys estarem descritas ao nível portas lógicas e dessas mesmas células estarem optimizadas.

Apenas o controlador TAP apresenta uma diferença de área muito grande comparativamente às diferenças encontradas para as outras células. O controlador TAP é a célula mais complexa entre as células comparadas, por isso é compreensível que o processo de síntese tenha uma influencia maior (em termos de área) sobre esta célula que sobre as outras. Alem disso, o controlador TAP é um elemento obrigatório na arquitectura BS que tem que ser incluído em todos os circuitos com facilidades BS, por isso o esforço de optimização por parte dos criadores das células Synopsys sobre o controlador TAP deve ter sido maior em relação às outras células. Como este controlador é uma máquina de estados bem conhecida (descrita totalmente pelo standard BS) pode-se criar uma versão desta célula optimizada, para cada tecnologia, descrita ao nível das portas lógicas e pedir ao utilizador que escolha entre uma destas versões optimizadas e a versão a sintetizar.

A área final do circuito com a lógica BS sintetizada é 38% maior que a área do circuito gerado pelo Synopsys, o que é explicável em função dos factores já apontados. (Deve-se salientar que ao nível das células de base só foi possível comparar uma pequena parte de entre elas). Este valor de 38% foi obtido para um circuito principal constituído por um conjunto de 4 flip-flops tipo D para o qual foi escolhida a inclusão da arquitectura BS obrigatória. Neste caso a área da lógica BS em relação à área da lógica principal é muito grande , por isso podemos apontar o valor encontrado (38%) como sendo o valor máximo da diferença entre as áreas finais dos circuitos gerados pelos 2 processos comparados. No caso de circuitos muito grandes, o peso em termos de área, da lógica BS é menor e por isso a diferença percentual entre a área desse circuito gerado pelos dois processos deverá ser sempre menor.

O referido aumento da área final do circuito quando sintetizado em conjunto com a lógica de teste é o preço a pagar pela independência que se consegue ter em relação à tecnologia. Mesmo assim podem-se fazer optimizações no método proposto de modo a tentar diminuir o aumento de superfície inerente ao processo de síntese.

VI. CONCLUSÕES

No decorrer deste trabalho desenvolveu-se uma estratégia para fazer síntese de circuitos conformes ao standard *Boundary Scan*. Desenvolveu-se ainda uma biblioteca de células de base (para teste *Boundary Scan*) descritas ao nível RTL, que podem ser sintetizadas com o auxílio de uma ferramenta de síntese VHDL.

Tendo em vista uma melhoria da ferramenta de síntese de circuitos conformes ao standard *Boundary Scan* podem descrever-se vários tipos de células BS para colocar na biblioteca (nível RTL) e depois pedir ao utilizador que escolha o tipo de célula conveniente para a sua aplicação, deste modo o utilizador pode personalizar a lógica BS incluída no seu circuito. Quanto ao controlador TAP podem-se fazer as alterações propostas no ponto VI.B.

A ferramenta desenvolvida apresenta a vantagem de ser independente da tecnologia (uma única descrição das células de base independente da biblioteca sobre a qual vai ser sintetizado o circuito) mas há um compromisso que é importante pesar. Neste caso o que se ganha em independência (em relação à tecnologia) e em tempo de concepção paga-se em área.

AGRADECIMENTOS

Agradeço ao laboratório TIMA (Grenoble, França) a disponibilização dos meios necessários à realização deste trabalho e aos elementos do grupo Design of Complex Systems todo o apoio concedido.

REFERÊNCIAS

[1] IEEE Standard 1149.1-1990: IEEE Standard Test Access Port and Boundary Scan Architecture, 1990

[2] Supplement (B) to Standard Test Access Port and Boundary Scan Architecture, IEEE Std 1149.1-1990, 1994

[3] Kenneth Parker, “The Boundary Scan Handbook”, 1992

[4] M. Marzouki, V. Castro Alves and A. Antunes Ribeiro, “Requirements and General Framework for an Efficient Synthesis for Testability Methodology”, 2nd International Test Synthesis Whorkshop-Santa Barbara (CA), 1995