UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

LAURO SCHEFFER PURICELLI

IMPLEMENTAÇÃO E VALIDAÇÃO DE TESTES DE UM CIRCUITO INTEGRADO PARA VALIDAR UMA BIBLIOTECA DE CÉLULAS

Trabalho de Graduação.

Prof. Dr. Renato Perez Ribas
Orientador

Porto Alegre, julho de 2014.

AGRADECIMENTOS

Primeiramente aos meus pais, que não mediram esforços para que este momento chegasse. Meu pai que, desde cedo, me mostrou o mundo da eletrônica e da computação e me fez gostá-lo. Minha mãe que, com seus ombros de ferro, sempre batalhou a vida inteira para que a pobreza que ela um dia conheceu eu não a conhecesse.

Segundo, para o Estado brasileiro que me acompanha e investe em mim a longos 13 anos e que, em matéria de educação, faz muito com o pouco que tem.

Ao professor Ribas que soube me orientar pela caminhada da realização deste trabalho.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ASIC	*Application-Specific Integrated Circuit*
FPGA	*Field Programmable Gate Array*
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
UFRGS	Universidade Federal do Rio Grande do Sul
RTL	*Register Transfer Level*
STA	*Static Time Analysis*
VHDL	*Very High Speed Integrated Circuits Hardware Description Language*
ATE	*Automatic Test Equipment*
LEC	*Logical Equivalence Checking*
CMOS	*Complementary Metal-oxide Semiconductor*

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Com a crescente complexidade dos Circuitos Integrados (CIs), a metodologia de construção de um CI que utiliza uma biblioteca de células pré-fabricadas se tornou a mais utilizada pela indústria. Para tal, é indispensável a confiabilidade na biblioteca de células utilizada bem como nos dados que estas células dispõem. Pensando nestes desafios foi proposto um circuito com o objetivo de testar da forma mais completa que se possa uma biblioteca de células. Sendo assim, este trabalho tem por objetivo o estudo deste circuito, descrever os testes a serem realizados no ASIC e validá-los funcionalmente.

Palavras-Chave: validação, ASIC, biblioteca de células.

ABSTRACT

With the increasing complexity of the Integrated Circuits (ICs), the methodology that uses a library cell of pre-designed gates to build an IC became the preferred strategy in industry nowadays. In this way, one of the key points is the reliability of the library cells and its data used in the flow. Taking these challenges into account a circuit was proposed in order to completely test a library cells. Thus, this work aims to study this circuit, describe the tests to be performed in ASIC and functionally validate this tests through simulations.

Keywords: Validation, Library Cell, ASIC, Integrated Circuit

1 INTRODUÇÃO

Com o avanço da tecnologia CMOS (*complementary metal-oxyde semiconductor*) e sua popularização, há cada vez mais circuitos integrados (chips) nas mais diferentes atividades praticadas no dia a dia das pessoas. Desde o trabalho, em computadores dos mais diferentes tipos, ao lazer, atividades domésticas e físicas, enfim, nossa vida foi tomada por uma gama de dispositivos eletrônicos necessários às nossas vidas.

Com a evolução dos circuitos ao longo dos anos, que tem diminuido de tamanho e aumentado de complexidade, o desafio de se fabricar tais circuitos tem aumentado consideravelmente, impactando nas mais diferentes áreas do projeto de um Circuito Integrado (CI). Há assim a necessidade de utilizar ferramental e metodologias que tornem o projeto de um CI mais padronizado, ágil e flexível, principalmente nos circuitos digitais.

Para o projeto de um circuito integrado digital, podem ser utilizadas metodologias de projeto conhecidas como *full-custom* e *standard cell*. Na técnica de construção *full-custom*, cada transistor do circuito é "desenhado" individualmente. Sendo assim, cada parte pode ser inteiramente definida e otimizada para o circuito em questão. Porém, é uma técnica de projeto que demanda muito tempo e exige profissionais mais experientes e qualificados [Schuch, 2009].

1.1 Metodologia de Projeto "Standard Cell"

A maior parte dos circuitos integrados digitais de aplicação específica, que são conhecidos por ASICs, são projetados usando a metodologia *standard cell*. Nesta metodologia, um conjunto de células lógicas pré-projetadas, que está disponível em uma biblioteca, é utilizado pela equipe de projetistas no circuito que está sendo projetado.

Primeiramente, no fluxo de projeto '*Standard Cell'* é definido a arquitetura do sistema, a microarquitetura de cada bloco e as interconexões entre os blocos que compõe o circuito. Ao ser definido a arquitetura do circuito, estes blocos são codificados em linguagem de hardware como VHDL ou Verilog. Esta primeira etapa da construção de um CI independe da biblioteca de células a ser utilizada, podendo ser reutilizada em outros projetos.

Uma vez que se tenha as descrições dos blocos e o código destes, são realizadas simulações lógicas, também conhecido como verificação funcional, que tem por objetivo comparar o circuito desenvolvido com as suas especificações lógicas, não utilizando assim as especificações das células a serem utilizadas no circuito.

Ao se finalizar a descrição do circuito em linguagem de hardware e finalizar a verificação funcional deste, é realizado então o mapeamento para uma biblioteca de células escolhida. Esta é uma das tarefas realizadas na etapa de síntese lógica, e a saída

é um arquivo, p.e. *netlist.v*, também em linguagem de hardware contendo toda a lógica do circuito mapeada para a biblioteca de células alvo.

Com o circuito mapeado, são realizados diversas verificações como a análise estática no tempo (STA), consumo de potência, e comparação lógica (LEC – *Logical Equivalent Checking*) entre a descrição não-mapeada (HDL) e mapeada do circuito, para verificar a integridade lógica do circuito mapeado, utilizando-se as características lógicas, temporais e de consumo das células da biblioteca escolhida.

Por fim, é realizada a síntese física do circuito onde as células da biblioteca são mapeadas para silício de acordo com suas definições de área e conexões, que geralmente são disponíveis em arquivos LEF/DEF, com outras células. Também são utilizados os modelos temporais destas para garantir que o circuito funcione com uma frequencia dentro da especificação adotada.

1.2 Motivação

Com o uso das bibliotecas de células no desenvolvimento de circuitos digitais, é necessário que, antes do uso destas células, elas tenham sido previamente testadas exaustivamente e validadas/caracterizadas para que o projeto não seja penalizado por uma eventual célula lógica que esteja defeituosa ou parametrizada de forma errada. Deste modo, o objetivo do trabalho aqui descrito é realizar os testes, validar e provar que uma biblioteca de células funciona corretamente.

Para tal, dentre as várias abordagens existente na literatura, será utilizado o trabalho de [Ribas, 2010]. A partir de estudos teóricos foi proposto um chip para se testar uma biblioteca de células, que aqui é chamado de Libtest, e então foram criadas as ferramentas de síntese para ele, e, por último, foi enviado para fabricação.

1.3 Proposta

Deste modo, o presente trabalho tem por objetivo descrever e realizar o teste e a validação da implementação fabricada em silício. Será estudada a arquitetura do circuito proposto bem como a metodologia de testes de uma biblioteca de células proposta, sua implementação e então será proposto um conjunto de vetores de teste para validar o circuito Libtest.

1.4 Organização do texto

O trabalho etá dividido de tal forma: no capítulo 1 há uma breve introdução. No capítulo 2 são discutidas as questões referentes a uma biblioteca de células. No capítulo 3 é apresentada a arquitetura do circuito sobre o qual serão propostos os testes de validação. No capítulo quatro é detalhada a estratégia de validação do circuito apresentado no capítulo três. No capítulo 5 há a apresentação da arquitetura do teste proposto e as simulações. E por fim as conclusões.

2 BIBLIOTECA DE CÉLULAS

O projeto de um ASIC baseado em bibliotecas de células é o método mais utilizado pela indústria atualmente. Nesta metodologia é considerada o reuso de uma biblioteca de células para se construir circuitos digitais complexos. Com o uso desta técnica, baseada em células e portas lógicas pré-caracterizadas, o tamanho e a eficiência do ASIC final passam a ser delimitados pelo perfil da biblioteca escolhida. [Ribas, 2010]

Uma biblioteca de células consiste em um conjunto de células com diferentes características quanto ao número de entradas e saídas, velocidade, área e consumo. Logo, para se utilizar uma biblioteca de células em um projeto de circuito integrado, é necessário garatir que não só cada célula funcione individualmente, como também que os modelos e informações referentes a estas células estejam corretos, para uma correta avaliação dos projetistas no que se refere a área, atrasos e consumo do circuito que está sendo projetado.

Segundo [Lin, 1999] algumas características do que é considerado essencial em uma biblioteca de células são:

1) A funcionalidade de cada célula devem estar corretas nos modelos de síntese lógica e simulação.
2) A performance de tempo de cada célula descrita no *datasheet* deve ser suficientemente correta.
3) O desenho (*layout*) de cada célula não deve conter violações de regra de desenho.
4) As células podem ser melhor utilizadas por uma ferramenta de síntese.
5) As células devem permitir a otimização das etapas de *placement* e roteamento de um projeto maior.

2.1 Classificação das células e perfil da biblioteca

Ao se desenvolver uma biblioteca de células em determinado nó tecnológico, há a necessidade de se optar quais as características que esta biblioteca terá e que serão diferenciais em relação a outras possivelmente existentes. Quanto maior for o número de células da biblioteca, mais opções de otimização se terá na etapa de mapeamento tecnológico, mas mais complexa será esta etapa para a ferramenta de síntese escolhida.

Há três características interdependentes nas células de uma biblioteca, que tem de ser levadas em consideração ao se construir uma biblioteca, que são área, atraso do sinal e consumo. Quanto maior forem as células desenvolvidas, menor será o tempo de propagação e maior o seu consumo. Já aquelas células que são otimizadas para um

baixo consumo (*low-power*) sofrem alguma penalidade na performance (frequência) mas em compensação ocupam uma área menor.

Numa biblioteca digital, há prioritariamente, três tipos de células: inversores/buffers, combinacional e sequencial. Os inversores/buffer são os tipos mais simples, pois simplesmeste invertem o sinal de entrada (inversores) ou aumentam a capacidade de corrente do sinal. As células combinacionais são aquelas em que a saída depende somente da lógica implementada entre as entradas da célula, e as células sequenciais (*latchs* e *flip-flops*) são aquelas em que a(s) saída(s) dependem das entradas e do estado das saídas no tempo anterior.

Assim, o perfil da biblioteca depende das características destas células. Algumas abordagens, como [Ricci, 2007] defendem o uso de um número reduzido de células enquanto outras, como [Guan, 1996], visam um número grande de células na biblioteca para se ter maior flexibilidade na construção dos circuitos.

Algumas células presentes na maioria das bibliotecas são, na parte das sequenciais os flip-flops D com ativação na borda de subida ou de descida, com ou sem os sinais de set e reset, síncrono ou assíncrono. Na parte combinacional as células básicas são as portas AND, OR, NAND e NOR de 2, 3 ou 4 entradas e combinações destas como AOI (And-or inverter), bem como portas mais complexas , por exemplo XOR, XNOR. Um bom exemplo de biblioteca de células bem completa pode-se encontrar em "Digital Standard Cell Library SAED_EDK90_CORE" da Synopsys, cujo o datasheet se encontra disponível em

[http://web.engr.oregonstate.edu/~traylor/ece474/reading/SAED_Cell_Lib_Rev1_4_20_1.pdf].

2.2 Dados e informações de uma biblioteca de células

Ao se desenvolver uma biblioteca de células, que, como discutido anteriormente, será utilizada em diferentes projetos no futuro, é necessário disponibilizar para os projetistas que desenvolvem a aplicação os mais variados modelos, elétricos, de consumo, de atrasos e de área das células a serem utilizadas para que, no final do projeto que utiliza esta biblioteca de células, as medidas estimatidas sejam muito parecidas com os valores apresentados pelo circuito integrado em silício. Para tal, é necessário realizar a caracterização da biblioteca de células.

2.2.1 Caracterização da biblioteca

O processo de caracterização de uma biblioteca ou de células individuais envolve a extração dos seus parâmetros elétricos, temporais e de consumo. Sendo assim, é necessário utilizar modelos e convenções previamente estabelecidos e amplamente utilizados como por exemplo para a medição de atrasos o *Linear Delay Model* (LDM) e o *Non-Linear Delay Model* (NLDM). Abaixo, há a definição de alguns dados temporais que são de extrema importância na caracterização do circuito:

- **tempo de subida**: é o tempo que um sinal de saída leva para ir do nível lógico 0 para o nível lógico 1. Cada nível lógico, 0 e 1, está associoado a um percentual da tensão de Vdd que corresponde a cada nível, sendo normalmente de 10% para 0 e 90% para 1.

- **tempo de descida**: é o tempo que um sinal de saída leva para ir do nível lógico 1 para o nível lógico 0.

- **tempo de propagação**: é o tempo que define a rapidez com que uma mudança em uma entrada provoca uma mudança na saída. Normalmente é medido o tempo entre a entrada ter 50% da sua tensão final até o sinal de saída ter 50% da tensão final da transação.

Para o cálculo da potência estática e dinâmica dissipada pelas células, há de se levar uma série de parâmetros no cálculo como os atrasos no circuito, sua temperatura e parâmetros de fabricação. Sendo assim, a especificação dos limites de operação do circuito em relação a temperatura, frequência e tensão tem de ser levados em conta.

A caracterização elétrica completa das portas lógicas deve ser feita levando-se em conta entradas com diferentes tempos de subida e de descida, capacitâncias de saída, *corners* do circuito (temperatura, tensão e variação de parâmetros de fabricação) e valores de dissipação de energia estática e dinâmica.

2.2.2 Arquivos de uma biblioteca de células

Estas características elétricas e de área das células são fundamentais para realização do mapeamento do circuito, análise estática de tempo (STA) e estimação de consumo. Estes dados são geralmente providos por geradores automáticos de bibliotecas e ferramentas de caracterização e divididos por célula. Deste modo, a biblioteca deve ser verificada e validada no ambiente EDA.

São fornecidos arquivos em linguagem de hardware (HDL) contendo as diretivas das células a serem utilizadas, para que se possa realizar o mapeamento tecnológico (RTL -> netlist). As informações sobre os atrasos e consumo das células normalmente são disponibilizadas em um arquivo Liberty com extensão .lib.

Uma vez que se tenha a lista das interconexões mapeadas para a tecnologia alvo, é necessário transformar estas informações em informações geométricas e físicas. No arquivo LEF (*Layout Exchance Format*) há as informações de geometria das células. Também pode conter algumas informações elétricas como resistência e *thickness* e regras de desenho. O arquivo DEF (*Design Exchange Format*) representa o desenho físico em um formato ASCII e contém informações de roteamento. Por último há também um arquivo GDS (Graphic Data System), que é um arquivo binário contendo as informações geométricas de cada célula.

3 CIRCUITO LIBTEST

O presente capítulo tem por objetivo detalhar e explicar o funcionamento e a arquitetura do CI que foi projetado com o objetivo de se testar uma biblioteca de células. O CI projetado foi baseado no artigo "*Contributions to the evaluation of ensembles of combinational logic gates"* [Ribas, 2010].

3.1 Teste e validação de uma biblioteca de células

A validação em silício de portas lógicas é uma etapa importante do processo de criação de uma biblioteca de células, visto que, do ponto de vista de quem a utilizará no futuro, é muito importante a garantia de que o conjunto de células disponibilizado funcione de acordo com o especificado. Nas tecnologias mais avançadas, o desenvolvimento de novas bibliotecas é feito juntamente com a co-otimização do processo de fabricação. Ou seja, a biblioteca é redesenhada e recaracterizada várias vezes durante o afinamento das regras de processo.

Deste modo, diversas estratégias de teste foram desenvolvidas, desde a criação de estruturas específicas, osciladores em anel, contadores e utilização de circuitos *benchmarks*.

- ***benchmarks***: as células disponíveis na bibioteca são mapeadas para um circuito que já foi previamente testado e validado. Logo, caso o CI não esteja funcionando do modo esperado, é possível deduzir que haja algum erro ou em alguma(s) célula(s) ou na caracterização desta(s). É muito útil também para determinados objetivos como a comparação entre duas bibliotecas diferentes que contenham por exemplo diferentes topologias de transistores ou células com diferentes capacidades de corrente de saída (*drive strhenghts*). Apesar de os *benchmarks* serem extremamente úteis e utilizados, pode acontecer de, ou uma ou mais células desta biblioteca não ser utilizada, visto que a síntese do circuito é feito com ferramentas automáticas de CAD de acordo com configurações e restrições pré-definidas, ou uma células ser instanciada, porém não ser estimulada por completo. Por exemplo, uma porta AND2 em que uma das entradas tenha o valor lógico sempre fixo em 1.

- **osciladores em anel**: são utilizados principalmente no teste de buffers e inversores, visto que, como só se tem uma entrada e uma saída, a medição dos atrasos fica facilitada. É um modo eficiente de se medir o atraso médio de uma célula, sem a influência de fatores como circuitos externos ou de ambiente. Possui sempre um número de estágios ímpar, para garantir a condição de oscilação. A frequência de oscilação depende da quantidade de inversores utilizados. Quanto maior for a quantidade de inversores utilizada, maior será a precisão das medidas realizadas [Ribas, 2006].

3.2 Libtest

Devido ao grande número de funções lógicas booleanas e diferentes *drive strenghts*, o mais amplo dos três grupos de células lógicas é o de células combinacioanis, sendo este o foco do estudo de caso.

O objetivo principal do desenvolvimento do Libtest foi criar uma metodologia para produzir automaticamente um circuito de teste para a completa validação lógica e elétrica de um conjunto de células lógicas. Ela apresenta prodecimentos de operação e monitoramento simples, sendo útil para avaliar as bibliotecas tanto no nível de EDA quanto de silício. No nível de EDA, a metodologia é útil para validar os arquivos da biblioteca no ambiente de desenvolvimento, comparando os dados obtidos com os modelos de atraso e consumo das células. O circuito resultante pode ainda servir como um *benchmark* para se comparar diferentes modelos de desenho e topologias de transistores para o mesmo conjunto de células. No nível de silício, o principal objetivo é prover uma solução com número limitado de *pads* de I/O e o mínimo uso de *Automatic Test Pattern Generation* (ATPG) reduzindo assim os custos de teste.

Para se testar um conjunto de células lógicas é necessário instanciar todas as portas em paralelo, usando entradas compartilhadas, visando a controlabilidade do teste, enquanto os sinais de saída podem ser multiplexados visando a redução do número de pinos de *I/O* sem perder a observabilidade do teste.

Foi levado em consideração os seguintes aspectos para que a abordagem de construção de um circuito para testar um conjunto de células se tornasse efetiva:

1) assegurar pelo menos uma instanciação de cada célula disponível na biblioteca em avaliação;
2) assegurar um teste funcional completo em pelo menos uma instância de cada célula distinta da biblioteca escolhida;
3) permitir a verificação da precisão das medidas de atraso e potência dos modelos de células utilizados no fluxo ASIC;
4) prover meios de teste de confiabilidade do circuito com efeitos de envelhecimento sem a utilização de equipamentos de teste automáticos, também conhecidos como ATE (*Automatic Test Equipment*).
5) ter um número considerável de instâncias comparado com o número de células a serem testadas.

Esta solução de teste mistura várias técnicas de *design* e de teste amplamente utilizadas para alcançar os objetivos conforme os cinco itens acima, visando a validação de um conjunto de células combinacionais do ponto de vista funcional e elétrico.

Blocos combinacionais específicos foram criados para garantir a cobertura de uma parte de células a serem validadas (item 2). Os pinos de saída recebem o mesmo vetor de bits dos pinos de entrada, permitindo que os blocos sejam conectados em longas cadeias em cascata. O uso de vários blocos combinacionais permite que sejam instanciadas todas as células (item 1). A arquitetura do circuito é composta por tais blocos, em configuração de anel, sincronizados por uma barreira de registradores. Os modos síncrono e assíncrono possuem diferentes recursos relacionados com os objetivos acima. A configuração em anel permite a verificaçao da precisão dos modelos de atraso e de consumo de energia das células, comparando simulações elétricas com

medidas experimentais (item 3). O oscilador BIST também está incluso no circuito para a verificação de um extenso número de diferentes caminhos lógicos. Juntamente com o modo assíncrono, ele permite um teste de média e longa duração. Em caso de eventuais erros, o diagnóstico é facilitado pelo uso de multiplexadores. Por fim, o númemo total de instâncias no circuito de teste não é tão grande quando comparado com o tamanho da maioria dos tradicionais *benchmarks* usados para este propósito (aspecto 5).

3.2.1 Blocos combinacionais

Foram propostos blocos combinacionais de modo a garantir a aplicação completa dos estímulos nos conjuntos de células lógicas a serem verificados. Para alcançar o objetivo proposto, cada bloco é composto de duas partes ou estágios.

No primeiro estágio, um subconjunto de células pertencentes a biblioteca que está sendo testada é colocado em paralelo, compondo um circuito que tem como saída deste estágio o vetor "W", que é resultado do comportamento funcional das células instanciadas.

O segundo estágio recebe os valores intermediários "W" e recupera a informação de entrada na saída "Out", de modo que $Input \equiv Out$. A Figura 3.1 mostra um exemplo de bloco combinacional onde o primeiro estágio tem uma profundidade lógica igual a 1.

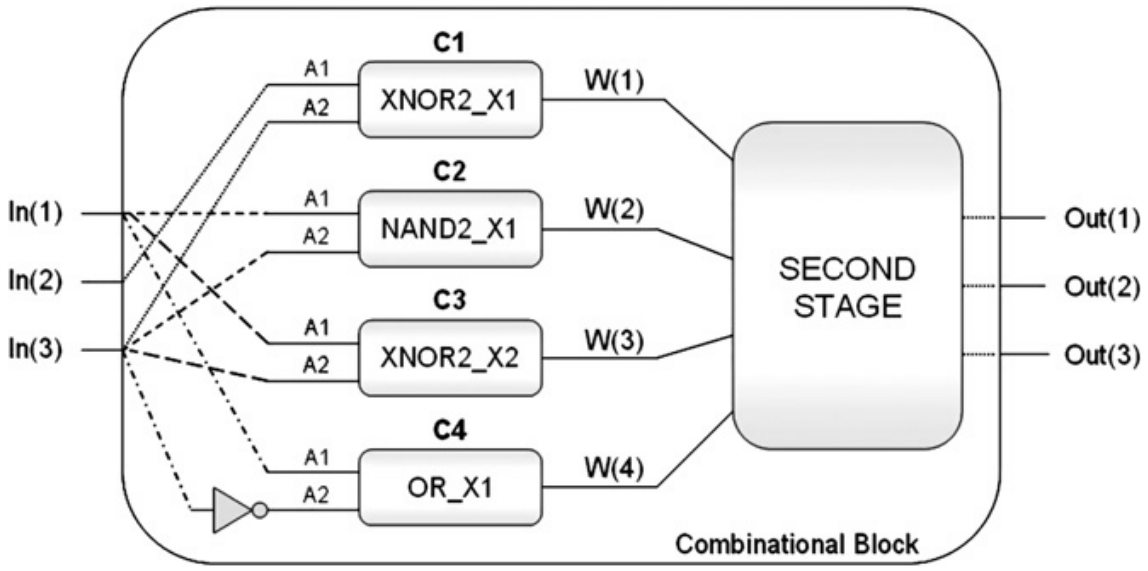


Figura 3.1. Exemplo de um bloco combinacional de três entradas.

Em resumo, o primeiro estágio produz as saídas intermediárias "W" e o segundo estágio recupera a informação de entrada. Utilizando esta estratégia, pode ser realizada a verificação funcional completa das células localizadas no primeiro estágio aplicando-se todas as possíveis combinações lógicas na entrada. Além disso, uma vez que a informação de entrada é reproduzida na saída, a verificação funcional se torna mais rápida e confiável ao se comparar os sinais de entrada e saída. Uma outra consideração é que esta característica permite o cascateamento dos blocos e a transmissão completa dos estímulos ao longo da cadeia. Quanto maior for o caminho percorrido maior será o atraso e por consequência melhor será a medição. Caminhos longos com maiores atrasos são preferíveis para medições temporais do circuito.

Abaixo, alguns requisitos do projeto desenvolvido:

a) toda a célula em teste será instanciada pelo menos uma vez no primeiro estágio do bloco combinacional visando garantir o estímulo completo de sua operação.

b) toda célula na biblioteca é considerada uma célula distinta. Por exemplo, duas células equivalentes logicamente mas com diferentes *drive strenghts* são tratadas separadamente.

c) todo bloco combinacional possui o mesmo número de entradas e saídas.

d) os vetores de entrada e saída tem o mesmo estado estável (*steady-state*).

e) a quantidade final de blocos combinacionais depende do total de células no conjunto a ser verificados, e ainda do número de células usados na composição de cada bloco do primeiro estágio.

f) o segundo estágio de cada bloco é sintetizado levando-se em conta o conjunto de células da mesma biblioteca utilizada.

3.2.2 Arquitetura

Uma vez descrito as características necessárias dos blocos combinacionais, será explicada a arquitetura do circuito proposto. A Figura 3.2 mostra a sua arquitetura. Os blocos combinacionais (CB) são cascateados de modo que os dados no começo da cadeia 'In(n..1)' sejam reproduzidos nos nodos 'Out(n..1)' se nenhum erro ocorrer durante o teste das células.

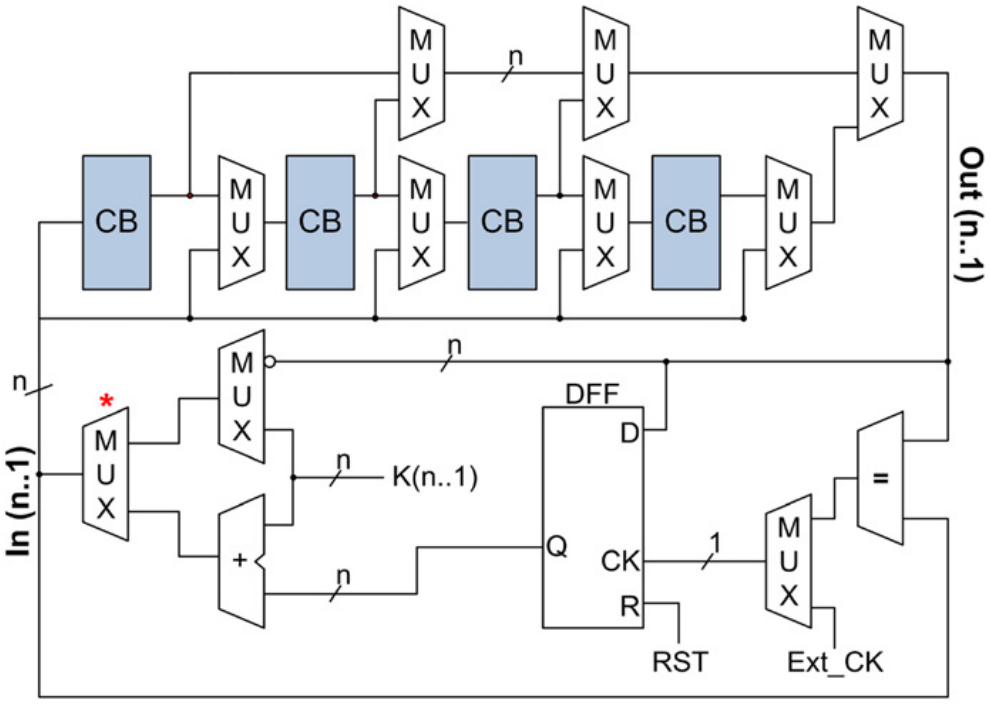


Figura 3.2. Arquitetura do circuito de teste

Para se ter uma sequência de vetores de teste com o mínimo de interferência externa, os sinais no final da cadeia são reconectados à entrada em uma configuração de anel. O Flip-Flop D (DFF) é necessário para sincronizar o somador cuja saída, a depender do valor de seleção do multiplexador (*), será utilizada na entrada dos blocos combinacionais, evitando assim condições de corrida no caminho da realimentação.

Pode ser utilizado como relógio do DFF tanto um relógio externo (Ext_CK), que é chamado de modo síncrono, como o valor da comparação entre a entrada e a saída dos blocos combinacionais, modo assíncrono.

O somador é utilizado para variar os dados de entrada da cadeia de blocos combinacionais (In) de modo que possa ser utilizada todas as combinações possíveis, permitindo assim variar os dados em k, testando as diferentes cargas e descargas de tensão entre as células.

Os multiplexadores da parte de cima do circuito são responsáveis pela escolha de por quais blocos combinacionais o sinal da entrada dos blocos combinacionais (In) será propagado até a saída destes (Out).

O multiplexador acima do somador é responsável pelo modo BIST. Este mux é o único a funcionar de um modo especial, visto que ele, a depender do valor de seleção pode inverter um dos n bits da saída Out, loop fechado, e manter os outros n-1 bits estáveis, em modo loop aberto.

O multiplexador representado com '*' define qual o modo de operação do circuito, se no modo BIST ou síncrono/assíncrono.

3.2.3 Modos de operação

Os modos de operação do circuito são síncrono, assíncrono, *loop* de 1 bit, modo *loop* aberto e diagnóstico. Serão explicados abaixo.

1) **Modo Síncrono**: O flip-flop é controlado por um relógio externo (Ext_CK). O somador é usado para incrementar o vetor de dados no anel, agindo como um contador síncrono '+k'. O correto comportamento da contagem indica o correto funcionamento dos blocos combinacionais e, consequentemente, do comportamento funcional de todo o conjunto de células que estão sendo verificados.
2) **Modo assíncrono**: No modo assíncrono, o sinal de relógio utilizado para ativar o flip-flop é fornecido pelo comparador que verifica se o vetor de entrada *In* já se propagou para a saída *Out*. Quando o mesmo valor estimulado na entrada chega à saída o comparador muda de '0' para '1' fornencendo o sinal de relógio para o FF. Os dados são então registrados no FF e passados para o somador. O somador incrementa o dado e aplica o novo vetor para a cadeia. Neste momento, como a entrada In não é mais igual a saída out o sinal do comparador volta ao valor 0 e assim permanece até que, novamente, o dado da entrada se propague até a saída. Neste modo de operação, só um sinal externo é necessário para começar a autocontagem. Se alguma célula estiver com defeito, o dado no final da cadeia será diferente do dado da entrada da cadeia e então o comparador ficará em 0, parando assim a execução automática da contagem.
3) **Modo loop fechado de um bit:** Este modo é fortemente inspirado no oscilador BIST (*Build-in Self Test*). O princípio básico é induzir a oscilação de um único bit em loop fechado enquanto os outros bits se mantém em loop aberto. No funcionamento normal do circuito, é esperado que o valor binário de uma entrada com índice *i* reapareça na saída de mesmo índice quando as duas são conectadas diretamente. No modo de loop, a saída de índice *i* é conectada a entrada através de um inversor, o que faz com que a cada ciclo de propagação se inverta o valor na saída.

A inversão do bit é necessário para garantir a condição de oscilação que foi implementada através do multiplexador acima do somador na Figura 3.2 e explicado na Figura 3.3. Através do decodificador, somente um bit do vetor Out[n...1] é utilizado no modo BIST e os outros são derivados da entrada K.

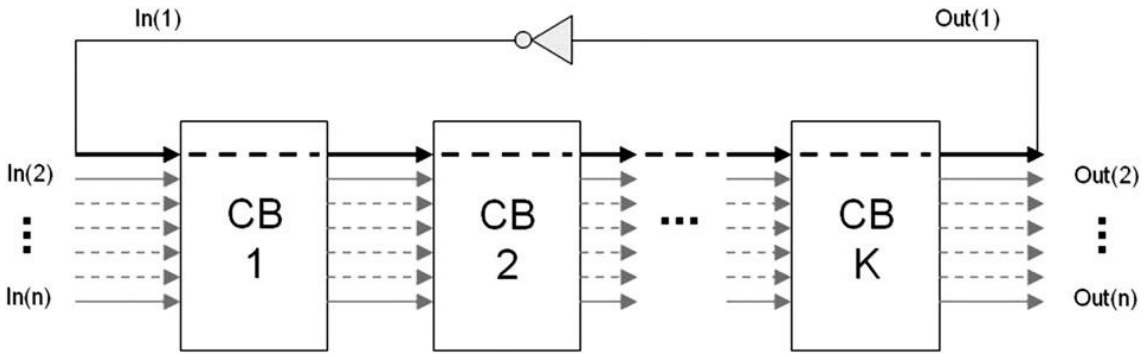


Figura 3.3. Ilustração do oscilador BIST.

4) **Modo Loop Aberto:** O sinal da entrada K passa diretamente para a saída Out, passando pela cadeia dos blocos combinacionais, sem passar pelo registrador.

5) **Modo Diagnóstico:** Os multiplexadores presentes na cadeia combinacional são utilizados para isolar blocos combinacionais caso seja necessário. Multiplexadores na entrada de cada bloco combinacional servem para selecionar os sinais vindos ou do bloco anterior ou do começo da cadeia, fazendo com que a saída dos blocos anteriores não interfira na entrada do bloco atual. De modo análogo, multiplexadores na saída dos blocos podem enviar os dados do meio da cadeia diretamente para sua saída. Assim, a cadeia pode ser reduzida para apenas um bloco, ou nenhum, sendo neste caso útil para verificar o bloco somador e os *flip-flops* sem a influência da cadeia de blocos combinacionais.

Para validar a implementação deste circuito de teste, tanto o verilog quanto o netlist foram verificados. Simulações lógicas e elétricas foram realizadas para os diferentes modos de operação do circuito descritos anteriormente. Diversas sequências de transições de sinais e configuração dos multiplexadores foram aplicadas de modo que pudessem ativar diferentes caminhos lógicos e realizar uma extensa verificação do circuito.

Do ponto de vista do comportamento funcional do circuito, um dos propósitos é prover um circuito de teste onde cada célula da biblioteca em construção seja completamente verificada. Em um primeiro momento, este teste pode ser tratado verificando se passou ou não, a não ser que haja algum erro específico em alguma célula.

Apesar de ser possível fazer esta verificação no modo síncrono, monitorando-se a entrada do somador '+K', o modo assíncrono é mais apropriado para o teste, visto que há uma lógica de verificação e de chaveamento próprios, sem precisar de interferência externa. Ou seja, se o circuito está rodando, então é garantida a correta funcionalidade de todas as células, sem a necessidade de verificar o vetor de saída 'Out(n..1)'.

O multiplexador utilizado para prover o modo de oscilação BIST pode ainda ser utilizado para interromper o loop de realimentação sem a necessidade de circuitos adicionais. Neste modo, chamado de 'cadeia aberta' permite o controle externo dos

estimulos que são enviados à cadeia de blocos combinacionais. Pode, assim, juntamente com os multiplexadores dos blocos combinacionais, auxiliar na identificação de blocos do circuito que estejam com defeito.

4 BRING-UP DO ESTUDO DE CASO

Uma vez abordados a importância das bibliotecas de células no desenvolvimento de um ASIC e explicado o objetivo, a arquitetura e o funcionamento do chip desenvolvido, o presente capítulo tem por objetivo descrever os principais testes funcionais que serão realizados no chip que tem por objetivo a validação e caracterização do conjunto de células lógicas de uma biblioteca de células. A biblioteca escolhida para os testes é uma biblioteca comercial, porém os testes aqui descritos podem ser adaptados para as mais diferentes bibliotecas de células disponíveis no mercado.

4.1 Especificações do ASIC

- Tensão de operação: 3.3V +-10%

- Maior número de entradas em uma célula: 6.

- Quantidade de blocos combinacionais em cascata: 20.

4.1.1 Pinos de entrada

Abaixo, será listado os pinos de entrada do ASIC a ser testado, bem como descrito suas características.

- clock

É o sinal do relógio externo que é fornecido ao circuito para a sincronização dos registradores (FF) no modo síncrono.

- clock_mux_sel

É o multiplexador que define qual sinal será utilizado para sincronizar os registradors (FF). Abaixo, os valores possíveis:

0: clock externo (modo síncrono)
1: resultado da comparação (modo assíncrono)

- mode_mux_sel

É o multiplexador que define se o ASIC estará no modo síncrono ou no modo OBIST/loop aberto. Abaixo, os possíveis valores para a entrada:

0: modo síncrono. A saída do multiplexador *mode_mux* recebe a saída do somador (*adder_output*).

1: modo OBIST/loop aberto. A saída do multiplexador recebe a saída do bloco bist_mux (*bist_mux_output*).

- bist_decoder_input[2:0]

O valor desta entrada define qual dos 6 bits de *internal_chain_output* será invertido. Os outros bits do sinal *bist_mux_output* serão derivados da entrada K. Abaixo, os valores possíveis para esta entrada:

0 a 5: o sinal *bist_mux_output* recebe o valor invertido do respectivo bit (de 0 a 5) do sinal *internal_chain_output* e os outros bits são derivados da entrada K.

6 ou 7: todos os bits de *bist_mux_output* são derivados da entrada K (modo loop aberto).

- mux_decoder_input[4:0]

Este sinal de entrada configura os multiplexadores da cadeia de blocos combinacionais (CBs), definindo assim por quais blocos combinacionais o sinal *internal_chain_input* passará até chegar a saída do bloco *main*_container. Abaixo, os possíveis valores da entrada:

0: o sinal não passará por nenhum CB. Através dos multiplexadores conetados aos blocos combinacionais o sinal de entrada passará diretamente para a saída dos CBs. Este modo também é conhecido como *short-circuit mode*.

1 a 20: o sinal passará somente pelo bloco combinacional correspondente, que pode variar do CB1 ao CB20, de acordo com o valor da entrada.

21 a 31: com um destes valores de entrada, o sinal passará por todos os CBs do bloco *main_container*.

- sel_chain_input

Este sinal de entrada é o sinal de seleção do multiplexador *chain_input_mux* que define qual sinal será enviado à saída "*chain_input*". Abaixo, os possíveis valores:

0: *internal_chain_input* (entrada da cadeia dos CBs)

1: *adder_output* (saída do somador)

- sel_chain_output

Este sinal de entrada é o sinal de seleção do multiplexador *chain_output_mux* que define qual sinal será enviado à saída "*chain_output*". Abaixo, os possíveis valores:

0: *internal_chain_output* (saída da cadeia de CBs)

1: *adder_input* (saída do registrador)

4.1.2 Pinos de saída

- chain_input[5:0]

Este sinal de saída pode conter os sinais *internal_chain_input*, entrada da cadeia dos CBs, ou *adder_output*, saída do somador, dependendo do sinal de seleção do multiplexador *chain_input_mux*.

- chain_output[5:0]

Este sinal de saída pode conter os sinais *internal_chain_output*, saída da cadeia dos CBs, ou *adder_input*, saída do registrador (FF), dependendo do sinal de seleção do multiplexador *chain_output_mux*.

- comparison_result

Esta saída é o sinal de relógio para o registrador (FF) no modo assincrono. Caso a entrada dos blocos combinacionais (*internal_chain*_input) seja igual a saída (*internal_chain_output*) então *comparison_result* é igual a 1. Caso contrário, zero.

A Figura 4.1 mostra o esquemático dos pinos do ASIC desenvolvido.

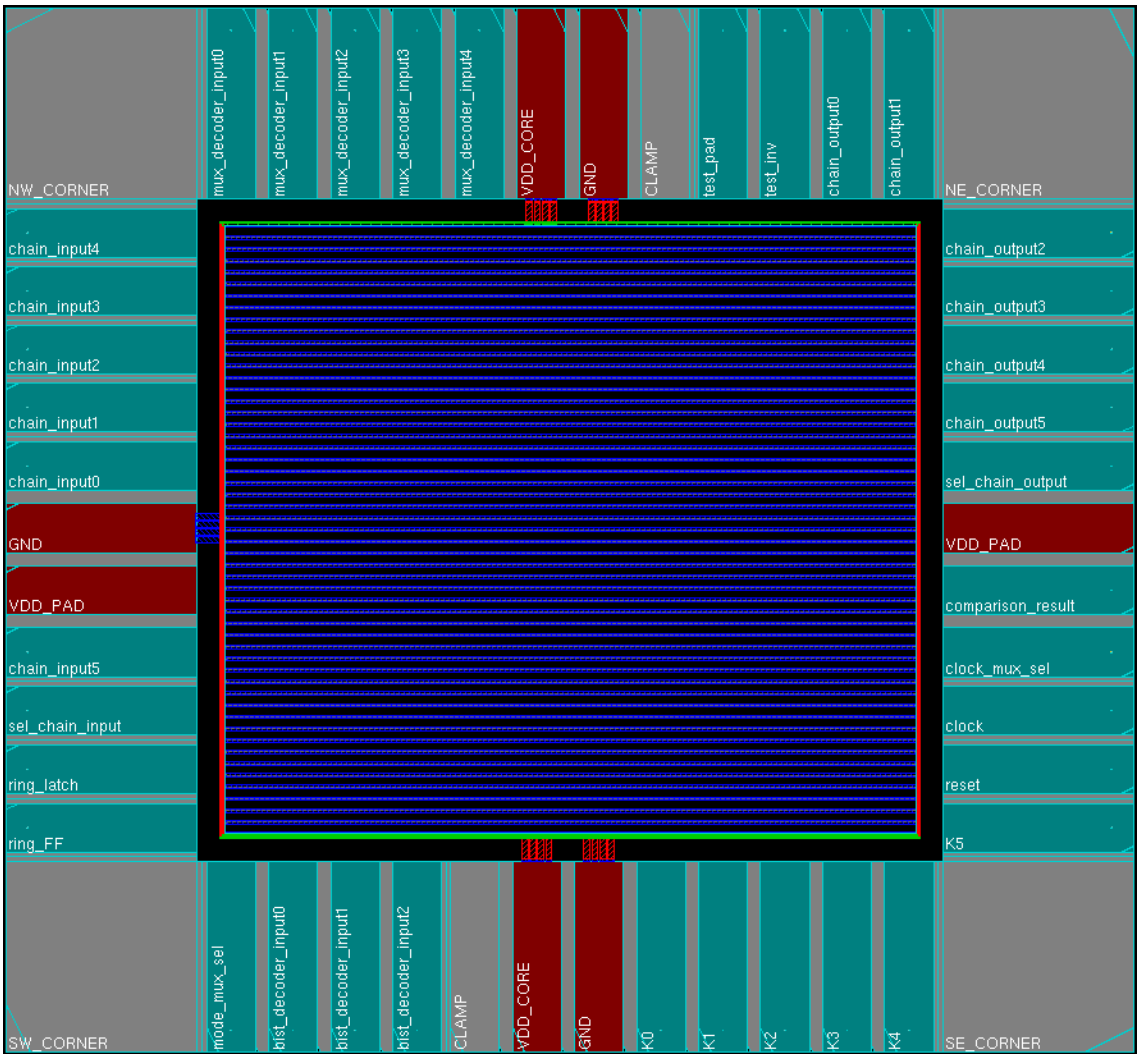


Figura 4.1. Esquemático dos pinos do ASIC.

4.2 Descrição dos testes

A seguir será explicado a forma de teste do ASIC, bem como das estruturas do circuito (multiplexadores, somador, registrador, inversores e blocos combinacionais).

Primeiramente será realizada uma inspeção visual no chip, com microscópio eletrônico auxiliado de um oscilpscópio, visando garantir que todos os pads estão conectados e os sinais de alimentação funcionando corretamente.

Há quatro pinos de alimentação do circuito. Dois pinos VDD_CORE, que alimentam o circuito e dois pinos VDD_PAD que alimentam o anél de pads em volta do circuito. Abaixo, serão descritos os testes realizados no circuito.

4.2.1 Teste do anel de pads

Para garantir o correto funcionamento e alimentação do anel de pads, na parte norte do circuito há dois pinos: *test_pad* e *test_inv*. Estes sinais são derivados do sinal de *reset* da seguinte forma:

Test_pad = reset

Test_inv = ~reset

Para os testes funcionais, primeiramente serão testados todos os blocos do chip exceto o bloco principal que contém os blocos combinacionais (*main_container*). Isto se deve ao fato de que, para testá-lo, é necessário garantir o correto funcionamento do resto dos blocos do ASIC desenvolvido. Uma vez que o registrador (FF), e os diversos multiplexadores foram testados, tendo-se obtido êxito, se vai para a segunda parte do teste que é o teste do modo síncrono com as diferentes configurações dos multiplexadores do bloco *main_container* geradas pelo bloco *mux_decoder*. Após, serão testados os modos assíncrono e OBIST do circuito.

A Figura 4.2 mostra a arquitetura do chip em teste. Note que o multiplexador com o sinal de asterísco (*) é o multiplexador responsável pela inversão de um único bit do sinal de saída dos blocos combinacionais (*internal_chain_output*), tendo assim o funcionamento diferente dos outros multiplexadores do circuito.

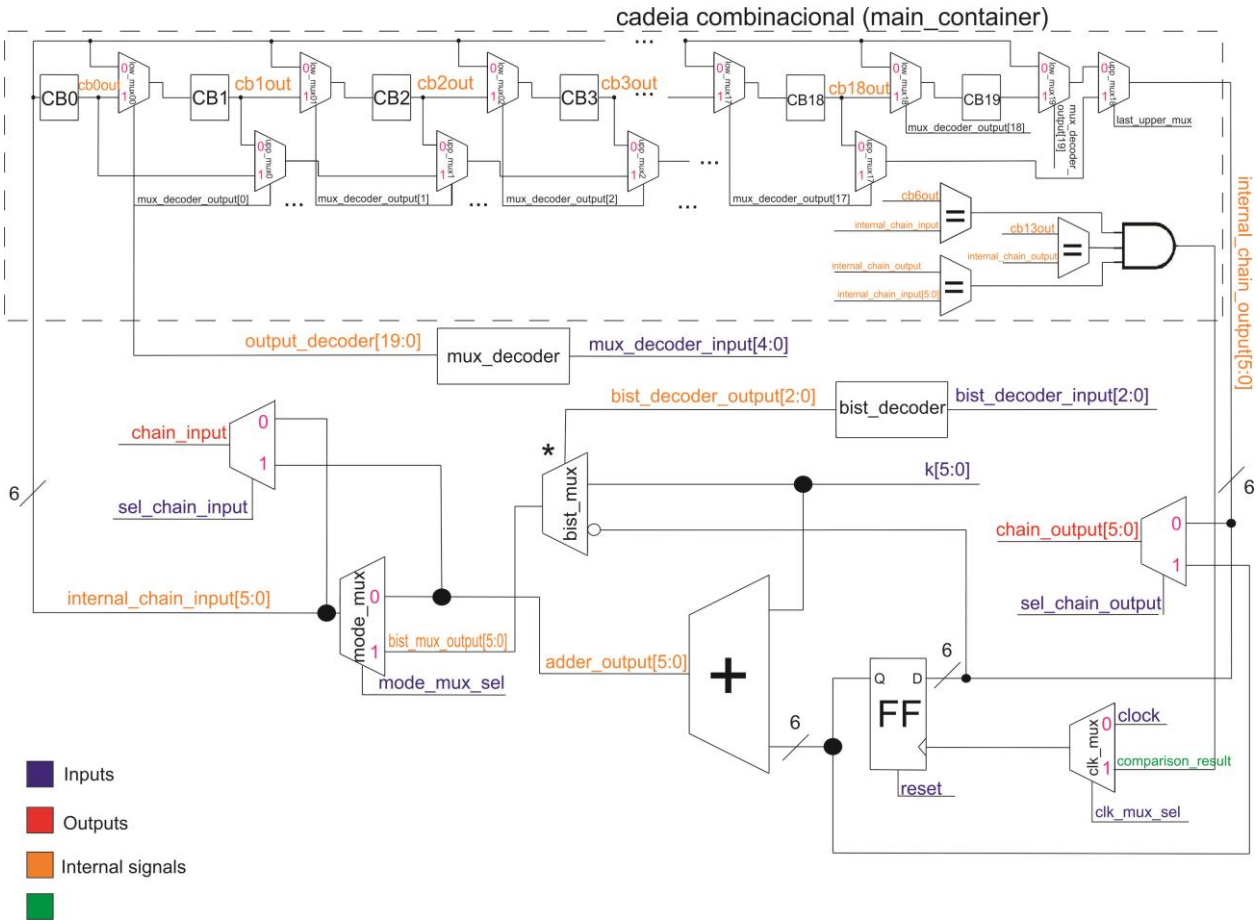


Figura 4.2. Arquitetura do circuito em teste.

Nesta primeira parte do teste funcional, como se quer testar somente os blocos especificados nos testes, os dados de entrada do bloco main_container passarão direto para a saída do mesmo, sem passar por nenhum bloco combinacional (CB). Para tal, a entrada mux_decoder_input = 0x00.

4.2.2 Teste do multiplexador *chain_input_mux* (1)

O presente teste tem por objetivo testar parte do multiplexador *chain_input_mux*. A Figura 4.3 mostra, em verde, o caminho percorrido desde a entrada K no circuito para realizar o teste.

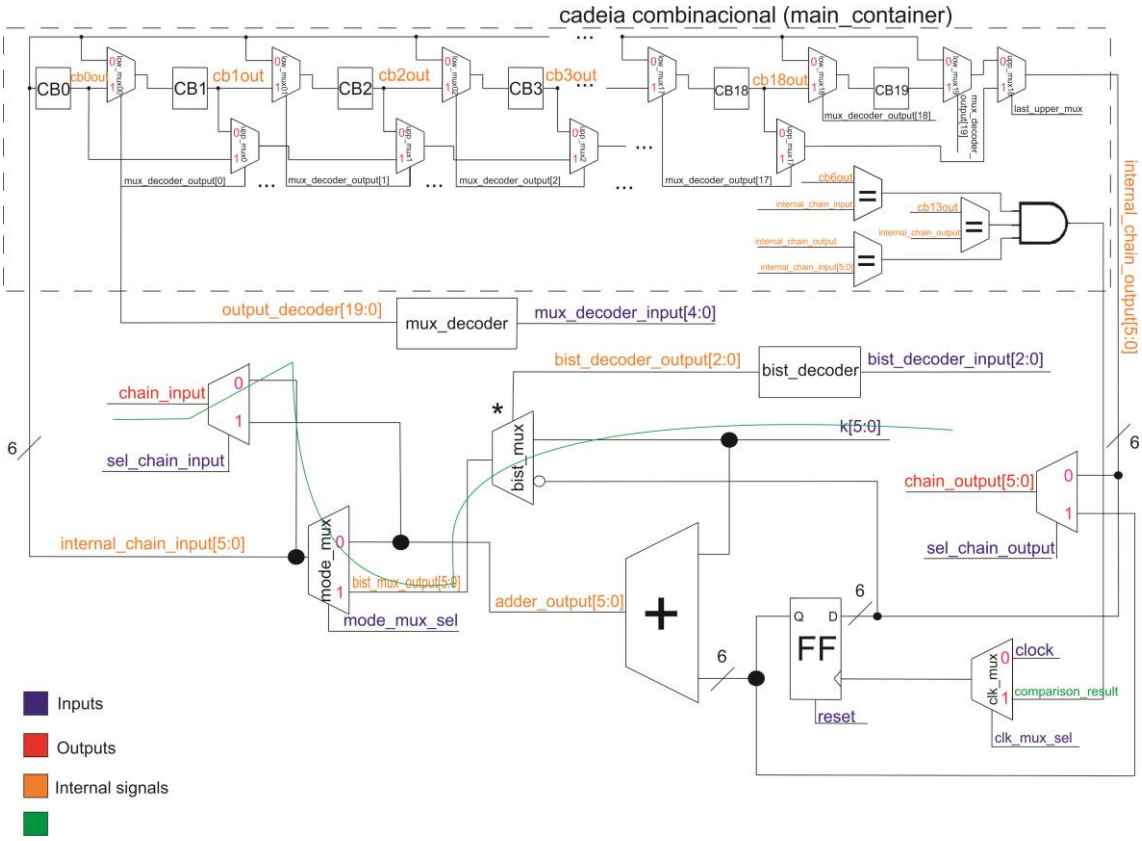


Figura 4.3. Caminho percorrido pelo teste 1.

Abaixo, as tabelas referentes ao teste.

Tabela 4.1. Teste do multiplexador *chain_input_mux*.

Sinais de entrada fixos	Valor Fixado
clk_mux_sel	0
Reset	1
mode_mux_sel	1
bist_decoder_input	6
mux_decoder_input	0x00
sel_chain_output	1
sel_chain_input	**0**
Sinal de entrada modificados	Valores modificados
clock	_|‾‾|...|‾‾|...|‾‾...
K	0 a 63
Sinal de saída verificado	Valor Esperado
chain_input	K

4.2.3 Teste do registrador (FF) com reset igual a 1

O presente teste tem por objetivo testar o funcionamento do flip-flop e a saída do multiplexador *chain_output* com o sinal de reset igual a 1.

Tabela 4.2. Teste do registrador com reset = 1.

Sinais de entrada fixos	Valor Fixado
clk_mux_sel	0
reset	1
mode_mux_sel	1
bist_decoder_input	6
mux_decoder_input	0x00
sel_chain_output	**1**
sel_chain_input	0
Sinal de entrada modificados	Valores modificados
clock	_|‾‾|...|‾‾|...|‾‾...
K	0 a 63
Sinal de saída verificado	Valor Esperado
chain_output	0

4.2.4 Teste do registrador (FF) com reset igual a 0

O presente teste tem por objetivo testar o flip-flop e a saída do multiplexador *chain_output* com o sinal de reset igual a 0.

Tabela 4.3. Teste do registrador com reset = 0.

Sinais de entrada fixos	Valor Fixado
clk_mux_sel	0
reset	**0**
mode_mux_sel	1
bist_decoder_input	6
mux_decoder_input	0x00
sel_chain_output	1
sel_chain_input	0
Sinal de entrada modificados	Valores modificados
clock	_|‾‾|...|‾‾|...|‾‾...
K	0 a 63
Sinal de saída verificado	Valor Esperado
chain_output	K

4.2.5 Teste do somador com reset igual a 1

O presente teste tem por objetivo testar o somador com uma das entradas igual a 0.

Tabela 4.4. Teste do somador com reset = 1.

Sinais de entrada fixos	Valor Fixado
clk_mux_sel	0
reset	1
mode_mux_sel	**0**
bist_decoder_input	6
mux_decoder_input	0x00
sel_chain_output	1
Sinal de entrada modificados	Valores modificados
clock	_|‾‾|...|‾‾|...|‾‾...
sel_chain_input	0 -> 1 -> 0
K	0 a 63
Sinal de saída verificado	Valor Esperado
chain_input	K

4.2.6 Teste do multiplexador *chain_output_mux*

O presente teste tem por objetivo testar correto funcionamento do multiplexador *chain_output_mux*.

Tabela 4.5. Teste do multiplexador *chain_output_mux*.

Sinais de entrada fixos	Valor Fixado
clk_mux_sel	0
reset	1
mode_mux_sel	**1**
bist_decoder_input	6
mux_decoder_input	0x00
sel_chain_input	1
Sinal de entrada modificados	Valores modificados
clock	_|‾‾|...|‾‾|...|‾‾...
K	0 a 63
sel_chain_output	**0 -> 1 -> 0**
Sinal de saída verificado	Valor Esperado
chain_output	K -> 0 -> K

4.2.7 Teste do bloco main_container (modo síncrono e diagnóstico)

O presente teste tem por objetivo testar o funcionamento dos blocos combinacionais bem como dos multiplexadores associados a estes. Para tal, começando pelo modo *short-circuit*, onde o sinal passa da entrada da cadeia para o final sem passar por nenhum dos blocos combinacionais, e depois passando, um a um, por eles, até que no último teste o sinal passe por todos os blocos combinacionais da cadeia.

Tabela 4.6. Teste do modo síncrono e diagnóstico.

Sinais de entrada fixos	Valor Fixado
clk_mux_sel	0
reset	0
mode_mux_sel	0
bist_decoder_input	6
sel_chain_input	0
sel_chain_output	0
Sinal de entrada modificados	Valores modificados
clock	_|‾‾|...|‾‾|...|‾‾...
K	0 a 63
mux_decoder_input	0 a 21
Sinal de saída verificado	Valor Esperado
chain_output	Chain_input

4.2.8 Teste modo assíncrono

O presente teste tem por objetivo testar o funcionamento do circuito no modo assíncrono. Este pode ser realizado de forma similar ao teste anterior, só modificando o sinal da entrada *clk_mux_sel* para que o relógio que sincroniza os FFs seja o comparador que sai do bloco *main_container*, *comparison_result*, e não um relógio externo.

Tabela 4.7. Teste do modo assíncrono.

Sinais de entrada fixos	Valor Fixado
clk_mux_sel	1
reset	0
mode_mux_sel	0
bist_decoder_input	6
sel_chain_input	0
sel_chain_output	0
Sinal de entrada modificados	**Valores modificados**
clock	_|‾‾|...|‾‾|...|‾‾...
K	0 a 63
mux_decoder_input	0 a 21
Sinal de saída verificado	**Valor Esperado**
chain_output	Chain_input

4.2.9 Teste modo loop aberto

Neste teste, o sinal da entrada K se propaga através do multipexador *bist_mux*, porém sem a inversão de qualquer um dos bits, sendo propagado diretamente para as saídas *chain_input* e *chain_output* passando pelo multiplexador *mode_mux*. Abaixo, as tabelas referente ao teste realizado.

Tabela 4.8. Teste do modo loop aberto.

Sinais de entrada fixos	Valor Fixado
clk_mux_sel	0
reset	1
mode_mux_sel	1
clock	0
sel_chain_input	0
sel_chain_output	0
bist_decoder_input	7
Sinal de entrada modificados	**Valores modificados**
K	0 a 63
mux_decoder_input	0 a 21
Sinal de saída verificado	**Valor Esperado**
chain_input	K
chain_output	K

4.2.10 Teste modo OBIST

Neste teste é verificado o funcionamento do circuito no modo OBIST, onde somente um dos bits da saída dos blocos combinacionais tem invertido seu valor lógico e os outros permanecem com o valor anterior. Para realizar a medição do sinal do bit invertido e verificar que ele de fato se inverteu será utilizado um osciloscópio digital, já que é de fácil medição com o aparelho. Os demais sinais serão verificados como anteriormente.

Logo, o teste do modo obist se divide e duas partes, uma feita com osciloscópio para verificar a frequência do bit oscilando e outra para verificar que os bits restantes de *internal_chain_input* permanecem iguais aos da entrada K. Abaixo, as tabelas referentes ao teste descrito.

Tabela 4.9. Teste do modo loop OBIST.

Sinais de entrada fixos	Valor Fixado
clk_mux_sel	0
reset	0
mode_mux_sel	1
clock	0
sel_chain_input	0
sel_chain_output	0
Sinal de entrada modificados	Valores modificados
bist_decoder_input	0 a 5
K	0 a 63
mux_decoder_input	0 a 21
Sinal de saída verificado	Valor Esperado
chain_output	chain_input && mask**

Para este teste do modo OBIST, as saídas *chain_input* e *chain_output* são iguais exceto pelo bit que está sendo invertido.

5 VALIDAÇÃO DA ESTRATÉGIA DE TESTES

No capítulo anterior foi traçada a estratégia de testes das estruturas do chip que queremos testar. Este capítulo tem por objetivo validar a estratégia de testes proposta. Foi criado um arquivo em linguagem de hardware, verilog, utilizado nas simulações, para guiar a validação.

Quando o chip estiver disponível, este será conectado com um FPGA, de modo que os estímulos de entrada do chip sejam geradas a partir da lógica de teste implementada no FPGA, ou seja, sejam as saídas do teste gerado. Já as portas de saída do chip serão entradas no circuito implementado no FPGA, para que possam ser medidas e verificadas de acordo com a lógica implementada em cada teste.

5.1 Arquitetura do circuito validador

O circuito utilizado para validar a estratégia dos testes possui uma arquitetura cujo bloco principal, *test_top*, possui uma máquina de estados que gerencia o andamento dos testes dos componentes do circuito Libtest, e os diferentes modos de operação: síncrono, assíncrono, loop aberto e diagnóstico.

Os testes descritos nos subcapítulos 3.2.2 à 3.2.6 são inteiramente realizados do estado 0 ao estado 11 da máquina de estádos ‘fsm_state’. Como para estes testes ou não se utiliza o somador ou se utiliza com uma das entradas igual a zero, não é necessário gerar valores para a entrada K que diferem de um incremento.

Para os modos síncrono, assíncrono e diagnóstico também é utilizado o bloco k_counter.

5.1.1 Bloco k_counter

Um dos principais ganhos da arquitetura do chip a ser testado é a possibilidade de variar os dados da entrada dos blocos combinacionais em +K unidades, em vez de somente +1, provendo diferentes ordens de cargas e descarga nas células que deseja-se testar.

Sendo assim, foi criado o bloco que gera valores para a entrada K do circuito Libtest de modo a se prover todas as combinações possíveis. O sinal de relógio deste bloco é derivado de uma lógica que o possibilita ser utilizado tanto para o modo síncrono quanto para o assíncrono.

```
assign k_counter_clk = (fsm_state >= 12 && fsm_state <= 15) ?
                                clock : ((fsm_state > 15 )? comparison_result:0);
```

Ha dois modos possíveis para a contagem de k.:

1) K é um número ímpar: este modo é o mais simples, visto que é só contar o número de vezes do total de transições possíveis (2^6) para testar todas as combinações possíveis para este valor ímpar de K. Isto se deve ao fato de que, um número K ímpar e outro valor 2^N são primos entre si, o que significa que, somando de K em K, só se repetirá um valor já utilizado anteriormente quando o teste completar as 2^N transições.

2) $K = n.2^N$: sendo n um número ímpar e K par, obviamente. Para estes números o teste é realizado em N etapas para não haver repetição de valores. Exemplo: para K = 2, A sequência prevista começando em 0 seria: (0, 2, 4, 6, 8, ..., 2042, 2044, 2046, 0, 1, 3, 5, 7,..., 2045, 2047, 1).

A Figura 5.1 mostra a simulação realizada no modo síncrono no momento que o valor da entrada K é modificado de 1 para 2 e a Figura 5.2 mostra a simulação quando K é igual a 2 e está sendo modificado da metade par para a metade ímpar do teste. Note que os valores das saídas *chain_input* e *chain_output* são iguais, significando assim o correto funcionamento dos blocos combinacionais.

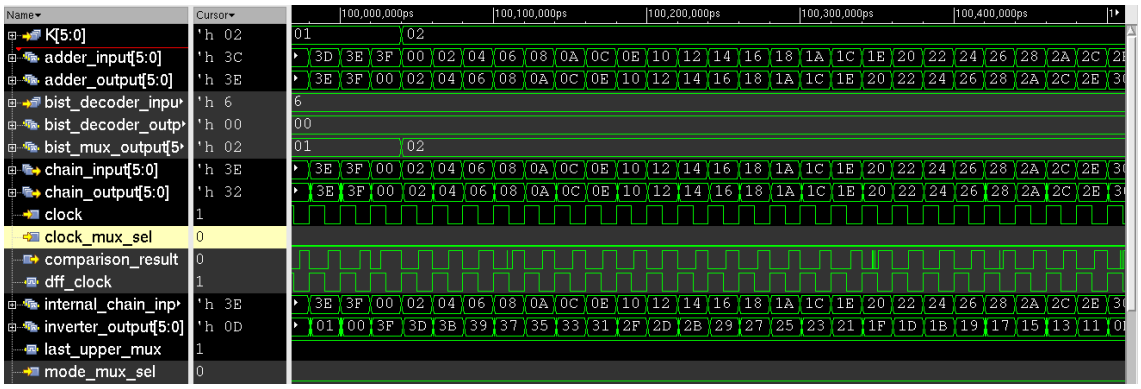


Figura 5.1. Modo síncrono com K variando de 1 para 2.

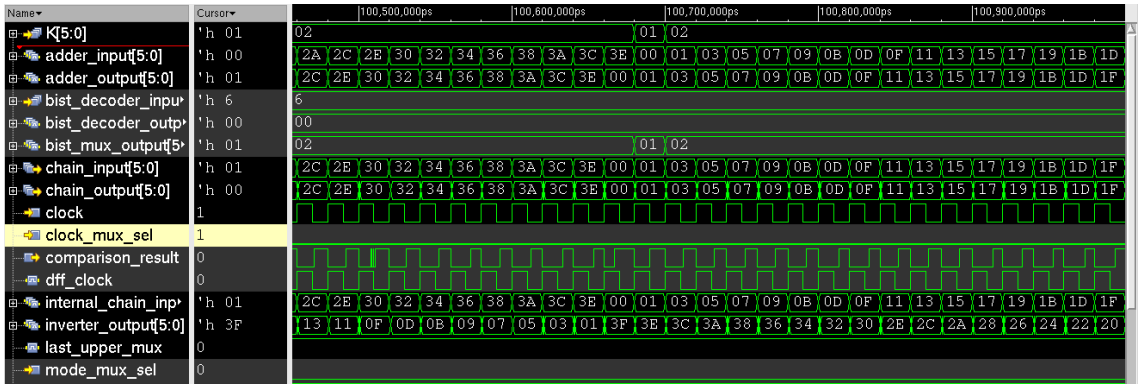


Figura 5.2. Modo síncrono com K igual a 2.

Ao realizar o teste no modo assíncrono, há de se tomar o cuidado de que o sinal de relógio do registrador do Libtest é o sinal de comparação entre a entrada e a saída dos blocos combinacionais, que é gerado pelo próprio circuito. Logo, há de se ter o cuidado para estimulá-lo de forma que o circuito consiga proseguir rodando "por conta própria". A Figura 5.1 mostra a simulação em forma de onda do começo de um teste no modo assíncrono. É possível notar que, a flag *comparison result* está em 1 durante um longo período de tempo e para forçar sua ida para '0', visando começar uma sequencia do teste, muda-se o valor da entrada dos blocos combinacionais *chain_input*. Quando o

sinal se propagar pela cadeia de blocos combinacionais, o valor de *comparison_result* voltará a ser 1 iniciando assim a contagem "automática". Em todos os testes realizados, caso seja descoberto algum erro funcional do chip nos testes, ou seja, algum bloco não está funcionando como deveria, uma flag de erro *error_flag* vai para o nível lógico alto e lá permanece até o fim do teste.

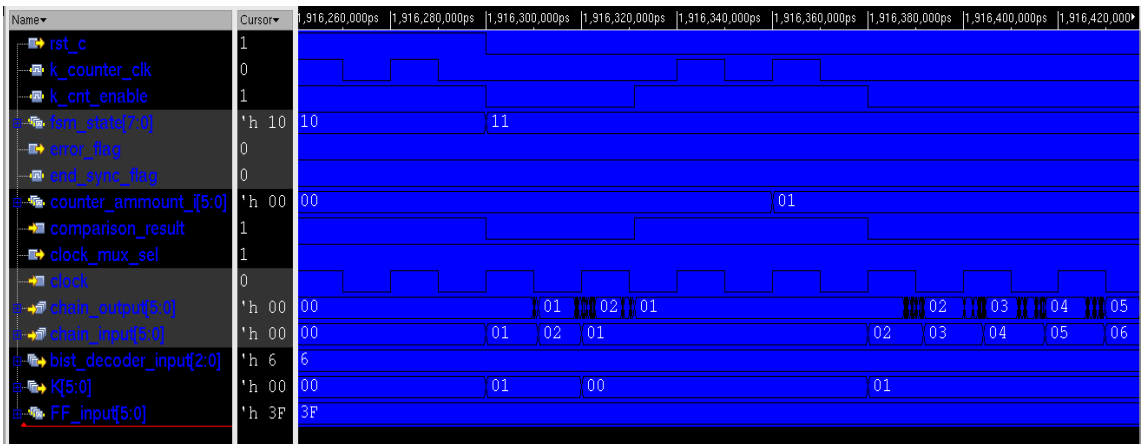


Figura 5.3. Formas de onda do início do teste no modo assíncrono.

6 CONCLUSÕES

Podemos concluir que foi proposta e validada, através de simulações, um conjunto de testes para se testar um ASIC que tem por objetivo testar uma biblioteca de células.

Uma vez que se tenha o chip em mãos, para verificar seu comportamento funcional é só aplicar os estímulos descritos nos variados testes possíveis e avaliar a resposta a estes estímulos, ou seja, os pinos de saída do chip.

Os testes aqui descritos podem ser adaptados para se testar outras bibliotecas de células que se deseje, e que utilizem a mesma arquitetura de teste como a descrita no capítulo 3, se fazendo pequenos ajustes como por exemplo do número das entradas e saídas do circuito.

REFERÊNCIAS

Schuch, Nívea. **Desafios e avanços em Computação: Inovações e Tecnologia**, UFPEL, 2009.

R.-B. Lin, I.S.-H. Chou and C.-M. Tsai, **''Benchmark circuits improve the quality of a standard cell library''**, In: Asia and South Pacific Design Automation Conference (ASP–DAC), 1999, pp. 173.

B. Guan and C. Sechen, **''Large standard cell libraries and their impact on layout area and circuit performance''**. In: International Conference in Circuit Design (ICCD), 1996, pp. 378–83.

A. Ricci, I. De Munari and P. Ciampolini, **''An evolutionary approach for standard-cell library reduction''**. In: Great Lakes Symposium on VLSI (GLSVLSI), 2007, pp. 305–10.

RIBAS, R. P. , Lubaszewski, M. **"Contributions to the evaluation of ensembles of combinational logic gates"**, Microelectronics Journal 42, (2011) 371-381, 2010

Silva, Carlos. Ribas, R.. Reis, Andre "Test Structures to evaluate a cell library", SFORUM, 2006

APÊNDICE A <CODIFICAÇO DO DA SIMULAÇÃO DOS TESTES EM VERILOG>

```
module test_top
(
        clock,
        reset,
        K,
        clock_mux_sel,
        mode_mux_sel,
        bist_decoder_input,
        mux_decoder_input,
        sel_chain_input,
        sel_chain_output,
        chain_input,
        chain_output,
        comparison_result,
        //ring_FF,
        //ring_latch
);

parameter DATA_WIDTH = 5;
parameter BIST_DECODER_INPUT_WIDTH = 2;
parameter BIST_DECODER_OUTPUT_WIDTH = 5;
parameter MUX_DECODER_INPUT_WIDTH = 4;
parameter MUX_DECODER_OUTPUT_WIDTH = 19;

input clk_c;
```

```
//ASIC OUTPUT

input[DATA_WIDTH:0] chain_input;
input[DATA_WIDTH:0] chain_output;
input comparison_result;

//ASIC INPUT
output rst_c;   //ASIC reset signal
output[DATA_WIDTH:0] K;
output clock_mux_sel;
output mode_mux_sel;
output[BIST_DECODER_INPUT_WIDTH:0] bist_decoder_input;
output[MUX_DECODER_INPUT_WIDTH:0] mux_decoder_input;
output sel_chain_input;
output sel_chain_output;

output reg error_flag;
reg end_test;
reg [7:0] fsm_state;
reg [7:0] counter;
wire [5:0] counter_ammount_i;
wire k_counter_clk;
wire k_cnt_enable;
wire end_sync_flag;

k_counter_main k_counter_main(
        .clk(k_counter_clk),
        rst(rst_c),
        end_test_flag(end_sync_flag),
        counter_ammount_i(counter_ammount_i)
);

  assign k_counter_clk = (fsm_state >= 12 && fsm_state <= 15) ? clock : ((fsm_state >
15 )? comparison_result:0);
```

```
always @ (posedge ext_clock or posedge rst) begin
        if(rst) begin
                end_test <= 0;
                counter <= 0;
                fsm_state <= 0;
                rst_c <= 1;
                K <= 0;
                clock_mux_sel<= 0;
                mode_mux_sel <= 0;
                bist_decoder_input <= 6;
                mux_decoder_input <= 0;
                sel_chain_input <= 0;
                sel_chain_output <= 1;
                fsm_state <= 0;
        end
        else begin
                fsm_state <= fsm_state;
                case (fsm_state)
                        0: begin        //      1.2.2   Teste   do   multiplexador
chain_input_mux (1)
                                rst_c <= 1;
                                K <= 0;
                                clock_mux_sel<= 0;
                                mode_mux_sel <= 1;
                                bist_decoder_input <= 6;
                                mux_decoder_input <= 0;
                                sel_chain_input <= 0;
                                sel_chain_output <= 0;
                                fsm_state <= fsm_state + 1;

                        end
                        1: begin
                                K <= K + 1;
                                if (chain_input != K) begin
                                        error_flag <= 1;
```

```
                end
                else begin
                        if (K == 63) begin
                                fsm_state <= fsm_state + 1;
                        end
                        else begin
                                fsm_state <= fsm_state;
                        end
                end
        end

        2: begin        //1.2.3 Teste do registrador (FF) com reset igual a 1
                rst_c <= 1;
                K <= 0;
                clock_mux_sel<= 0;
                mode_mux_sel <= 1;
                bist_decoder_input <= 6;
                mux_decoder_input <= 0;
                sel_chain_input <= 0;
                sel_chain_output <= 1;
                fsm_state <= fsm_state + 1;
        end
        3: begin
                K <= K + 1;
                if (chain_output != 0) begin
                        error_flag <= 1;
                end
                else begin
                        if (K == 63) begin
                                fsm_state <= fsm_state + 1;
                        end
                        else begin
                                fsm_state <= fsm_state;
                        end
```

```
            end
        end
        4: begin        //1.2.4 Teste do registrador (FF) com reset igual a 0
            rst_c <= 1;
            K <= 0;
            clock_mux_sel<= 0;
            mode_mux_sel <= 1;
            bist_decoder_input <= 6;
            mux_decoder_input <= 0;
            sel_chain_input <= 0;
            sel_chain_output <= 1;
            fsm_state <= fsm_state + 1;

        end
        5: begin
            rst_c <= 0;
            K <= K + 1;
            if (chain_output != K) begin
                error_flag <= 1;
            end
            else begin
                if (K == 63) begin
                    fsm_state <= fsm_state + 1;
                end
                else begin
                    fsm_state <= fsm_state;
                end
            end
        end
        6: begin        //1.2.5 Teste do somador com reset igual a 1 (modo direto)
            rst_c <= 1;
            K <= 0;
            clock_mux_sel<= 0;
```

```
                mode_mux_sel <= 0;
                bist_decoder_input <= 6;
                mux_decoder_input <= 0;
                sel_chain_input <= 0;
                sel_chain_output <= 1;
                fsm_state <= fsm_state + 1;

            end
            7: begin
                K <= K + 1;
                if (chain_input != K) begin
                    error_flag <= 1;
                end
                else begin
                    if (K == 63) begin
                        sel_chain_input <= 1;
                        fsm_state <= fsm_state + 1;
                    end
                    else begin
                        fsm_state <= fsm_state;
                    end
                end
            end
            8: begin        //1.2.5 Teste do somador com reset igual a 1
(passando pelo multiplexador MODE_MUX)
                K <= K + 1;
                if (chain_input != K) begin
                    error_flag <= 1;
                end
                else begin
                    if (K == 63) begin

                        fsm_state <= fsm_state + 1;
                    end
                    else begin
```

```
                                        fsm_state <= fsm_state;
                                    end
                                end
                            end
                            9: begin        //1.2.6 Teste do multiplexador chain_output_mux
                                rst_c <= 1;
                                K <= 0;
                                clock_mux_sel<= 0;
                                mode_mux_sel <= 1;
                                bist_decoder_input <= 6;
                                mux_decoder_input <= 0;
                                sel_chain_input <= 1;
                                sel_chain_output <= 0;
                                fsm_state <= fsm_state + 1;

                            end
                            10: begin
                                K <= K + 1;
                                if (chain_output != K) begin
                                    error_flag <= 1;
                                end
                                else begin
                                    if (K == 63) begin

                                        sel_chain_output <= 1;
    //modifica o sinal de seleção do multiplexador output_mux
                                        fsm_state <= fsm_state + 1;
                                    end
                                    else begin
                                        fsm_state <= fsm_state;
                                    end
                                end

                            end
                            11: begin
```

```
                K <= K + 1;

                if (chain_output != 0) begin
                        error_flag <= 1;
                end
                else begin
                        if (K == 63) begin

                                fsm_state <= fsm_state + 1;
                        end
                        else begin
                                fsm_state <= fsm_state;
                        end
                end
        end
        12: begin       //1.2.7 do modo síncrono e diagnostico. Longo reset com 100 ciclos de clock.
                rst_c <= 1;
                K <= 0;
                clock_mux_sel<= 0;
                mode_mux_sel <= 0;
                bist_decoder_input <= 6;
                mux_decoder_input <= mux_decoder_input;
                sel_chain_input <= 0;
                sel_chain_output <= 0;

                if(counter < 100) begin
                        fsm_state <= fsm_state;
                        counter <= counter + 1;
                end
                else begin
                        counter <= 0;
                        fsm_state <= fsm_state + 1;
                end
        end
        13: begin
```

```
                    k <= counter_ammount_i;
                    if(end_sync_flag == 1) begin
                        mux_decoder_input <= mux_decoder_input
+ 1;
                        fsm_state <= fsm_state + 1;
                    end
                end
                14: begin
                    if(mux_decoder_input == 22) begin
                        fsm_state <= fsm_state + 1;
                    end
                    else begin
                        fsm_state <= fsm_state - 2;
                    end

                end
                15: begin       //1.2.8 Modo Assíncrono e diagnostico.
Longo reset com 100 ciclos de clock.
                    rst_c <= 1;
                    K <= 0;
                    clock_mux_sel<= 1;  //sinal de que é modo
assíncrono
                    mode_mux_sel <= 0;
                    bist_decoder_input <= 6;
                    mux_decoder_input <= 21;
                    sel_chain_input <= 0;
                    sel_chain_output <= 0;

                    if(counter < 100) begin
                        fsm_state <= fsm_state;
                        counter <= counter + 1;
                    end
                    else begin
                        counter <= 0;
                        fsm_state <= fsm_state + 1;
                    end
```

```
                        end
                        16: begin
                                k <= counter_ammount_i;
                                if(end_sync_flag == 1) begin
                                        mux_decoder_input <= mux_decoder_input
+ 1;
                                        fsm_state <= fsm_state + 1;
                                end
                        end
                        17: begin
                                if(mux_decoder_input == 22) begin
                                        fsm_state <= fsm_state + 1;
                                end
                                else begin
                                        fsm_state <= fsm_state - 2;
                                end
                        18: begin
                                fsm_state <= fsm_state;
                                end_test <= 1;
                        end
                        default: begin
                        end
                endcase
        end
    end
endmodule
```

ANEXO A

Este anexo corresponde ao artigo desenvolvido para o TG1 deste projeto de graduação.

Implementação e validação de Validação Pós Silício de um Circuito Integrado

Lauro Scheffer Puricelli[1], Renato Ribas[2]

[1]Centro de Excelência em Eletrônica Avançada (Ceitec)

CEP – 91.550-000 – Porto Alegre – RS – Brasil

[2]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

[1]lauro.puricelli@ceitec-sa.com, [2]rpribas@inf.ufrgs.br

Abstract. *This paper has the objective to propose a post-silicon validation of IC (integrated circuit) based on FPGA solution. The techniques and tools used in the industry will be studied and then a bring-up solution for the case study will be implemented before its production on a commercial scale.*

Resumo. *Este artigo tem o objetivo de propor uma solução de validação pós-silício de circuitos integrados (CIs) baseada em uma implementação de testes reconfiguráveis em FPGA. Para tal, serão estudadas e utilizadas técnicas e ferramentas usadas pela indústria de semicondutores com o objetivo de validar como produto um chip antes de sua produção em escala comercial.*

1.Introdução

Com o avanço da tecnologia CMOS (complementary metal-oxyde semiconductor) e sua popularização, há cada vez mais circuitos integrados (chips) nas mais diferentes atividades praticadas no dia a dia das pessoas. Desde o trabalho, em computadores dos mais diferentes tipos, ao lazer, atividades doméstica, física, enfim, nossa vida foi tomada por uma gama de dispositivos eletrônicos necessários à nossas vidas.

Com a Lei de Moore, segundo a qual para uma mesma área de silício a quantidade de transistores dobra a cada 18 meses[1], que conforme exemplo da Figura 1 vem sendo comprovada com o passar dos anos, apesar de já se prever o seu esgotamento por questões físicas, os chips desenvolvidos ao longo dos anos têm se tornado cada vez mais complexos.

Este aumento de complexidade, que tem relação direta com as sucessivas diminuições no tamanho dos transistores, tem trazido muitos desafios para as mais diferentes áreas que abarcam a concepção de um Circuito Integrado: desde a criação de novas e mais sofisticadas ferramentas de CAD, definições arquiteturais, testes de pré e pós fabricação, implementação física, fabricação e seus mais variados e complexos processos físico-químicos, enfim uma gama extensa de atividades e recursos que tem por objetivo final transformar as camadas de silício em um produto de aplicação prática e com um mercado que viabilize seu desenvolvimento e sua fabricação em escala comercial.

Figura 1. Lei de Moore

[http://www.cmg.org/measureit/issues/mit41/m_41_2.html]

Das atividades acima, os testes realizados durante os vários estágios de desenvolvimento de um chip são de extrema importância, e realizados nos mais diferentes níveis de abstração. Verificação funcional, prototipação em FPGA, testes de wafers e validação pós-silício são alguns exemplos. No caso da validação pós-silício, com seu custo e sua complexidade crescendo mais do que os de *design*, chegando a até 40% dos custos total do produto [2], a área tem ganhado grande relevância e interesse nos últimos anos.
Deste modo, o objetivo do trabalho é fazer um estudo comparativo entre os mais diferentes tipos de teste na fase de pós-fabricação de um CI, bem como realizar um estudo de caso para colocar em prática este estudo. Ambas as partes serão melhores detalhadas nos capítulos a seguir.

O presente artigo se divide da seguinte forma. No primeiro capítulo há uma breve introdução. No segundo são abordadas as diferentes etapas de um fluxo ASIC padrão bem como os tipos de testes necessários no desenvolvimento de um chip. O terceiro

capítulo trata sobre as principais motivações para o presente trabalho e o quarto sobre a proposta do mesmo. A metodologia que será seguida é descrita no quinto capítulo e o circuito de estudo de caso é explicado no sexto. O sétimo capítulo mostra o cronograma e as etapas de implementação e no oitavo finalmente temos a conclusão.

2.Embasamento técnico

Para a concepção de um circuito integrado, há várias etapas a serem vencidas, desde a definição dos requisitos do produto, definição da arquitetura, descrição do RTL (register transfer level), verificação formal e funcional, etapa de *backend*, entre outros, também conhecido como fluxo ASIC. A Figura 2 mostra o fluxo ASIC padrão.

Durante o fluxo ASIC, há várias etapas de verificação e validação com diferentes níveis de abstração. A verificação funcional serve para validar os diferentes blocos no nível do RTL, sendo feitos *testbenchs* de estímulo para se fazer a análise comportamental do circuito. A análise estática de tempo é utilizada para verificar os requerimentos temporais do circuito desenvolvido, levando em consideração as informações de atraso em nível de portas lógicas, *setup time, hold time*, tempo de subida e de descida bem como o roteamento utilizado pelos sinais. Análise de potência (p*ower a*nalisys) prevê o consumo de um circuito circuito ou blocos isolados do mesmo em determinadas condições de operação. Há verificações e simulações também no nível físico e de laioute do chip.

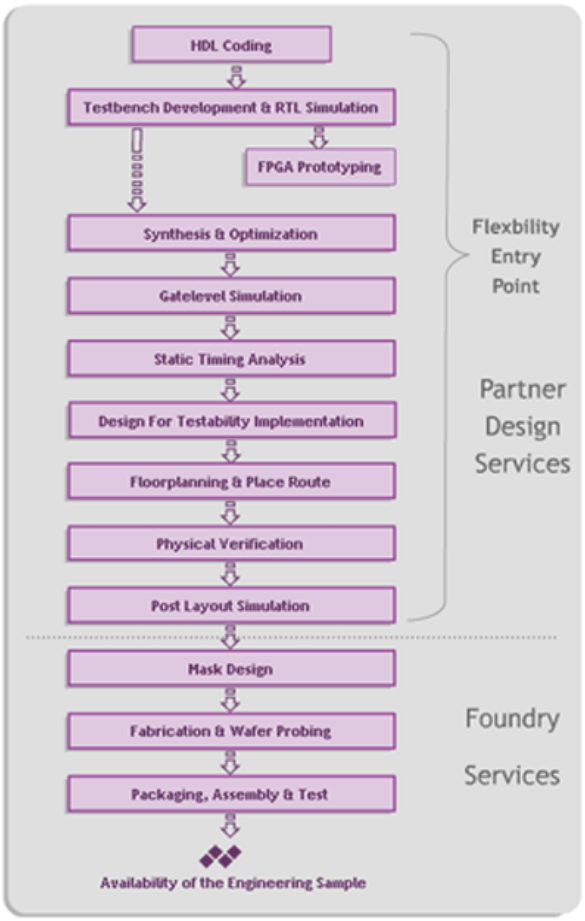


Figura 2. Fluxo ASIC [http://pinechips.com/asic_1.html]

Após a validação final do chip, conhecida como *tape-out*, este é enviado para ser fabricado, e então, retorna a empresa para ser validado como produto. Esta etapa é conhecida como *bring-up* ou validação pós-silício (*post-silicon validation)*. São feitos todos os testes para validar o CI comercial, utilizando ferramentas como equipamentos de teste automático (ATE), *Logic Analyzer*, FPGAs ou microcontroladores para comunicação com o CI, medições elétricas, analisador de espectro eletromagnético, entre outras, que variam de acordo com a especificação do CI e o objetivo do teste concebido.

Conforme dito acima, com a importância crescente na área de testes, uma vasta área passou a ser pesquisada e desenvolvida. Técnicas de codificação para auxiliar nos testes (DFT – *Design for Testability*), uso de geração automática de padrões de teste (ATPG) e circuitos que se auto testam (BIST - *Built-in self-test*) são algumas técnicas que são utilizadas atualmente na área visando o teste, a reusabilidade dos testes para diferente circuitos e também a diminuição dos custos, humanos e em aquisição de equipamentos para realizá-los.

3.Motivação

Os desafios da validação pós-silício estão cada vez maiores, devido ao alto nível de integração dos circuitos e sua alta complexidade, sendo assim praticamente impossível a detecção e correção de todos os erros antes da fase de tape-out. O efeito combinado desses fatores, com um *time-to-market* cada vez mais curto, tem exercido forte pressão na etapa de *bring-up* que, via de regra, é a última etapa antes da fabricação em escala comercial de um CI caso este não apresente nenhuma falha durante esta fase. Sendo assim, é necessário que esta etapa seja cada vez mais seja rápida, barata e aperfeiçoada [3].

Uma das abordagens utilizadas para testes pós-silício é o uso de FPGAs para realizar a comunicação com o CI. Esta abordagem é muito eficaz devido a sua flexibilidade e reconfigurabilidade durante execução (*on-the-fly*) [4] e também seu baixo custo quando comparada com o uso de equipamentos automáticos de teste (ATEs), por exemplo. Outro diferencial é que, enquanto a maioria dos ATEs são para teste específico de um único CI, o teste baseado em FPGAs pode ser utilizados em diferentes chips, modificando sua codificação e as configurações de I/O, ou utilizando arquiteturas reconfiguráveis.

4.Proposta

O objetivo do trabalho é propor uma série de testes na etapa de validação pós-silício. Serão abordados os diferentes tipos de solução que podem ser desenvolvidas, bem como suas vantagens e desvantagens no que se refere aos custos, reusabilidade e reconfigurabilidade da solução de acordo com os tipos de circuitos a serem testados, tipos de teste desejado, entre outros.

Conforme o capítulo anterior, o uso de FPGAs vem sendo cada vez mais utilizado pela indústria na etapa de validação pós-silício. Para tal, é de clara importância o conhecimento das principais arquiteturas e soluções de FPGAs comercialmente disponíveis. Logo, primeiramente será feito um estudo das diferentes arquiteturas de FPGAs comercialmente disponíveis dos principais fabricantes mundiais.

Na segunda etapa do projeto, um estudo detalhado sobre os diferentes tipos de teste, desafios e práticas que pode se ter para a realização de uma validação pós-silício será

efetuado. Juntamente a esta etapa, será realizado o estudo do circuito descrito no capítulo seis e então, posteriormente, será proposta uma solução de bring-up para este.

5.Metodologia

O presente trabalho foi dividido em várias etapas, de modo que cada uma seja mais específica possível, visando à organização do trabalho como um todo e ao acompanhamento detalhado das etapas a serem percorridas.

Primeiramente será feito um estudo sobre as arquiteturas FPGA. Esta etapa é importante visto que há uma grande variedade delas no mercado, com diferentes características dentre as quais podemos citar frequência de clock, tamanho e tipos de memória suportada (SRAM, flash, etc.), quantidada de células lógicas disponíveis, presença ou não de microprocessadores embarcados, diferença de periféricos embarcados nas placas, entre outros.

Tendo concluído parte desta etapa começara o estudo do chip especificado no sexto capítulo. Nesta etapa serão estudados o chip desenvolvido, sua arquitetura digital, especificações, sinais de entrada, saída e alimentação entre outros. É crucial para o trabalho o entendimento do chip visto que, somente assim é possível saber o que é preciso testar e como testá-lo.

Depois de ser feito o estudo dos diferentes modelos de FPGA no mercado e do chip referente ao estudo de caso, será elaborada uma estratégia de testes para este. Primeiro irá será feita uma pesquisa das estratégias utilizadas na indústria de semicondutores para se testar os chips. Em paralelo, será feito um minucioso levantamento do que precisa ser testado no chip do estudo de caso para que este seja validado.

Sabendo-se quais as necessidades de teste para a validação do CI como produto, será escolhida a plataforma FPGA em que se farão os testes, sendo levado em conta o estudo das arquiteturas. A próxima etapa então será codificar em linguagem de hardware o programa do FPGA para os testes.

A próxima etapa será a construção da plataforma em que o chip será testado, composta por este o FPGA e uma placa de circuito impresso para conectar os diversos sinais dos aparelhos interconectados.

Será feito um relatório completo das atividades desenvolvidas e também de como testar o CI. Caso o chip venha da fábrica a tempo, será realizado o teste.

6.Estudo de caso

O projeto de um ASIC baseado em bibliotecas de células é o método mais utilizado pela indústria atualmente. Nesta metodologia é considerada o reuso de uma biblioteca de células para se construir circuitos digitais complexos. Com o uso desta técnica, baseada em células e portas lógicas pré-caracterizadas, o tamanho e a eficiência do ASIC final passam a ser delimitados pelo perfil da biblioteca escolhida. [5]

Na abordagem que se utiliza biblioteca de células, é necessária a caracterização elétrica completa das portas lógicas antes de utilizá-las no fluxo ASIC.

A melhor forma para se projetar um circuito de teste para um conjunto de células lógicas é instanciar todas as portas em paralelo, usando entradas compartilhadas enquanto a sinais de saída podem ser multiplexados visando a redução do número de pinos de entrada e saída, visando a controlabilidade do teste.

Uma abordagem efetiva de construção de um circuito para testar um conjunto de células deve levar em consideração os seguintes aspectos:

1) assegurar pelo menos uma instanciação de cada célula disponível na biblioteca em avaliação;

2) asseguram um teste funcional completo em pelo menos uma instância de cada célula distinta da biblioteca escolhida;

3) permitir a verificação da precisão das medidas de atraso e potência dos modelos de células aplicados ao ASIC;
4) prover meios de teste de confiabilidade do circuito com efeitos de envelhecimento sem a utilização de equipamentos de teste automáticos.
5) ter um número considerável de instâncias comparado com o número de células a serem testadas.

Os blocos combinacionais aqui propostos são a chave para a efetividade dos testes. Eles são construídos de modo a garantir a aplicação completa dos estímulos nos conjuntos de células lógicas a serem testadas.

Para alcançar o objetivo proposto, cada bloco é composto de duas partes ou estágios, conforme a Figura 3.

Figura 3. Exemplo de um bloco combinacional de três entradas. [Ribas]

No primeiro estágio, um subconjunto de células em teste são colocadas em paralelo, compondo um circuito com profundidade lógica singular, tendo como saída deste estágio o vetor "W", que é resultado do comportamento funcional das células instanciadas. O segundo estágio recebe os valores intermediários "W" e produz o vetor de saída "Out", que é equivalente ao de entrada "In", de modo que .

Em resumo, o primeiro estágio produz as saídas intermediárias "W" e o segundo estágio recupera a informação de entrada. Utilizando esta estratégia, pode ser realizada a verificação funcional completa das células localizadas no primeiro estágio aplicando-se todas as possíveis combinações lógicas. Além disso, uma vez que a informação de entrada é reproduzida na saída, a verificação funcional se torna mais rápida e confiável ao se comparar os sinais de entrada e saída. Uma outra consideração é que com esta característica permite o cascateamento dos blocos e a transmissão completa dos estímulos ao longo da cadeia. Caminhos longos com maiores atrasos são preferíveis para medições temporais do circuito.

Uma vez mostrada as principais características da estrutura dos blocos que se pretende construir, alguns requisitos de *design* necessários serão resumidos a seguir:

a) toda a célula em teste será instanciada pelo menos uma vez no primeiro estágio do bloco combinacional visando garantir o estímulo completo de sua operação.

b) toda célula na biblioteca é considerada uma célula distinta. Por exemplo, duas células equivalentes logicamente mas com diferentes características são tratadas separadamente.

c) todo bloco combinacional tem o mesmo número de entradas e saídas.

d) os vetores de entrada e saída tem o mesmo estado estável (*steady-state*).

e) a quantidade final de blocos combinacionais depende do total de células no conjunto a ser verificados, e ainda do número de células usados na composição de cada bloco do primeiro estágio.

f) o segundo estágio de cada bloco é sintetizado levando-se em conta o conjunto de células da mesma biblioteca utilizada.

De acordo com as características dos blocos combinacionais descritos acima, a Figura 4 mostra a arquitetura do circuito de teste proposta. Os blocos combinacionais são cascateados de modo que os dados no começo da cadeia 'In(n..1)' sejam reproduzidos nos nodos 'Out(n..1)' se nenhum erro acontecer durante o teste das células.

Figura 4. Arquitetura do diagrama de blocos do circuito de teste [Ribas]

Para se ter uma sequencia de vetores de teste com o mínimo de interferência externa, os sinais no final da cadeia são reconectados à entrada em uma configuração de anel. Há multiplexadores nas cadeias combinacionais permitindo utilizar somente alguns blocos. Os Flip-flops D servem para evitar condições de corrida no caminho de realimentação. O somador é utilizado para variar os dados de entrada, permitindo assim variar os dados em k unidades em vez de somente 1, sendo muito útil para testas diferentes tipos de carga e descarga de voltagem dentro e entre as células. Há ainda um comparador e multiplexadores para configurar os diferentes modos de operação.

Os quatro modos de operação do circuito são: síncrono, assíncrono, *loop* de 1 bit e diagnóstico. Por limitação de espaço, será explicado somente o modo síncrono.

Modo Síncrono: A barreira de registradores é controlada por um relógio externo (Ext_CK). O somador é usado para incrementar o vetor de dados no anel, agindo como um contador síncrono '+k'. O correto comportamento da contagem indica o correto funcionamento dos blocos combinacionais e, consequentemente, do comportamento funcional de todo o conjunto de células que estão sendo verificados. Figura 5 mostra um exemplo do comportamento do circuito em modo síncrono e assíncrono.

Figura 5. Comportamento no modo síncrono (60 a 100 ns) e assíncrono (100 a 140 ns) [Ribas]

7.Cronograma e etapas de implementação

O cronograma de execução das tarefas referentes ao projeto foi previsto para seis meses. Não há dependências entre tarefas realizadas em paralelo. é apresentado abaixo:

Etapa 1: Estudo das arquiteturas FPGA

Etapa 2: Estudo do CI que será testado.

Etapa 3: Pesquisa de métodos de testes (validação pós-silício) utilizados pela indústria.

Etapa 4: Levantamento de necessidades de teste no CI do estudo de caso.

Etapa 5: Escolha da plataforma FPGA para realização dos testes.

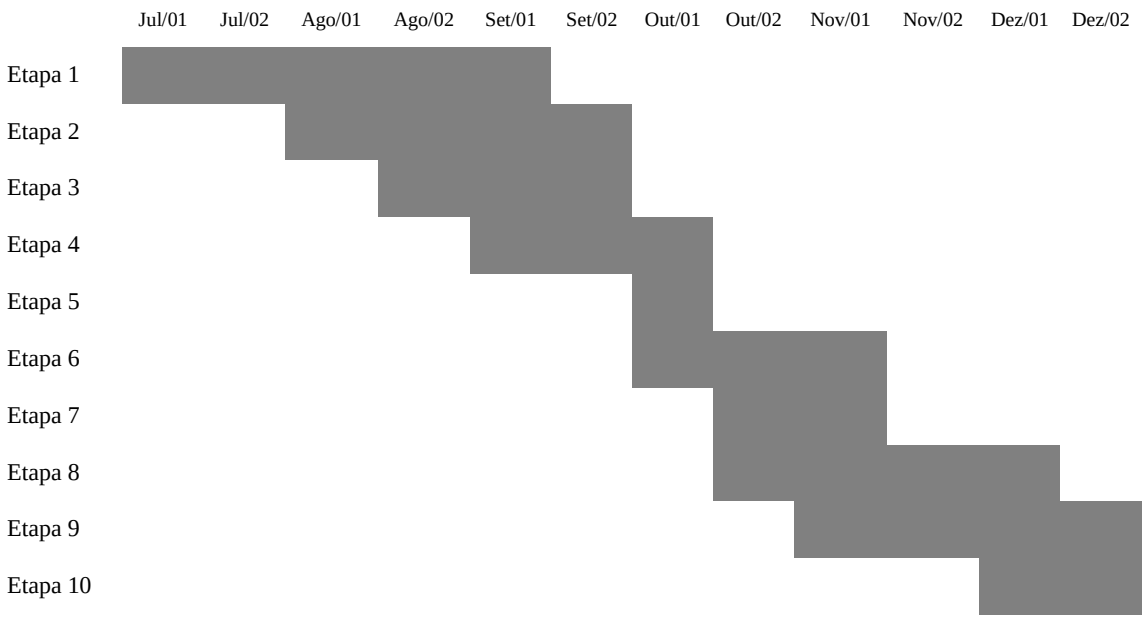
Etapa 6: Codificação do FPGA para testar o CI.

Etapa 7: Desenvolvimento da plataforma de testes do CI.

Etapa 8: Escrita do relatório das atividades desenvolvidas.

Etapa 9: Escrita do relatório do bring-up.

Etapa 10: Teste do chip.



8.Conclusão

A partir deste artigo, referente ao Trabalho de Graduação 1, foi possível obter a base necessária para a formulação do estudo e da implementação prática da fase de *bring-up* do chip do estudo de caso.

Ao final do trabalho desenvolvido, é desejado que se tenha projetado uma plataforma de testes do chip especificado, composta por um FPGA previamente escolhida, o CI e a circuitaria necessária para ligá-los, bem como fontes, osciloscópios para medidas, etc. Toda a documentação será feita, detalhando o processo de teste de modo que qualquer pessoa possa fazê-los utilizando o documento.

References

Moore's Law: More or Less <http://www.cmg.org/measureit/issues/mit41/m_41_2.html>

[2] Wang, Laung-Terng (2006). VLSI test principles and architectures : design for testability. 1st ed.

[3] Jagannath Keshava, Nagib Hakim, Chinna Prudvi (2010). “Post-silicon Validation Challenges: How EDA and Academia Can Help”.

[4] Debabrata Ghose (2012). “FPGA-based Solution for Automation of Post-Silicon Validation”.

[5] Ribas, R. P. , Lubaszewski, M. “Contributions to the evaluation of ensembles of combinational logic gates”, Microelectronics Journal 42, (2011) 371-381.