Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Programa de Pós-Graduação em Engenharia Elétrica

Uma Abordagem para Suporte à Verificação Funcional no Nível de Sistema Aplicada a Circuitos Digitais que Empregam a Técnica *Power Gating*

Tese de Doutorado

George Sobral Silveira

Prof. Elmar Uwe Kurt Melcher

Orientador

Campina Grande, PB - Brasil

Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Programa de Pós-Graduação em Engenharia Elétrica

George Sobral Silveira

Tese de Doutorado submetida à Coordenação do Curso de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para obtenção do grau de Doutor em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Processamento da Informação

Prof. Elmar Uwe Kurt Melcher

Orientador

Campina Grande, PB - Brasil

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

S587a Silveira, George Sobral

Uma abordagem para suporte à verificação funcional no nível de sistema aplicada a circuitos digitais que empregam a técnica *Power Gating* / George Sobral Silveira . – Campina Grande, 2012.

118f.: il.

Tese (Doutorado em Engenharia Elétrica) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática

Orientador: Prof. Dr. Elmar Uwe Kurt Melcher.

Referências.

1. Verificação Funcional. 2. *Power Gating*. 3. ESL. 4. SystemC TLM. I. Título.

CDU 621.3 (043)

"Uma Abordagem para Suporte à Verificação Funcional no Nível de Sistema Aplicada a Circuitos Digitais que Empregam a Técnica *Power Gating*"

GEORGE SOBRAL SILVEIRA

TESE APROVADA EM 10/08/2012

ELMAR UWE KURT MELCHER, Dr., UFCG
Orientador(a)

ANTONIO MARCUS NOGUEIRA LIMA, Dr., UFCG
Examinador(a)

EDNA NATIVIDADE DA SILVA BARROS, Ph.D, UFPE
Examinador(a)

JOÃO MARQUES DE CARVALHO, Ph.D., UFCG
Examinador(a)

GUIDO COSTA SOUZA DE ARAÚJO, Dr., UNICAMP
Examinador(a)

JOSEANA MACEDO FECHINE RÈGIS DE ARAÚJO, D.Sc., UFCG
Examinador(a)

CAMPINA GRANDE - PB

Resumo

A indústria de semicondutores tem investido fortemente no desenvolvimento de sistemas complexos em um único chip, conhecidos como SoC (*System-on-Chip*). Com os diversos recursos adicionados ao SoC, ocorreu o aumento da complexidade no fluxo de desenvolvimento, principalmente no processo de verificação e um aumento do seu consumo energético. Entretanto, nos últimos anos, aumentou a preocupação com a energia consumida por dispositivos eletrônicos. Dentre as diversas técnicas utilizadas para reduzir o consumo de energia, *Power Gating* tem se destacado pela sua eficiência. Ultimamente, o processo de verificação dessa técnica vem sendo executado no nível de abstração RTL (*Register Transfer-Level*), com base nas tecnologias CPF (*Common Power Format*) e UPF (*Unified Power Format*). De acordo com a literatura, as tecnologias que oferecem suporte a CPF e UPF, e baseadas em simulações, limitam a verificação até o nível de abstração RTL. Nesse nível, a técnica de *Power Gating* proporciona um considerável aumento na complexidade do processo de verificação dos atuais SoC. Diante desse cenário, o objetivo deste trabalho consiste em uma abordagem metodológica para a verificação funcional no nível ESL (*Electronic System-Level*) e RTL de circuitos digitais que empregam a técnica de *Power Gating,* utilizando uma versão modificada do simulador OSCI (*Open SystemC Initiative*). Foram realizados quatro estudos de caso e os resultados demonstraram a eficácia da solução proposta.

Palavras-chave: Verificação Funcional, *Power Gating*, ESL, SystemC TLM.

Abstract

The semiconductor industry has strongly invested in the development of complex systems on a single chip, known as System-on-Chip (SoC), which are extensively used in portable devices. With the many features added to SoC, there has been an increase of complexity in the development flow, especially in the verification process, and an increase in SoC power consumption. However, in recent years, the concern about power consumption of electronic devices, has increased. Among the different techniques to reduce power consumption, Power Gating has been highlighted for its efficiency. Lately, the verification process of this technique has been executed in Register Transfer-Level (RTL) abstraction, based on Common Power Format (CPF) and Unified Power Format (UPF) . The simulators which support CPF and UPF limit the verification to RTL level or below. At this level, Power Gating accounts for a considerable increase in complexity of the SoC verification process. Given this scenario, the objective of this work consists of an approach to perform the functional verification of digital circuits containing the Power Gating technique at the Electronic System Level (ESL) and at the Register Transfer Level (RTL), using a modified Open SystemC Initiative (OSCI) simulator. Four case studies were performed and the results demonstrated the effectiveness of the proposed solution.

Keyowrds: Functional Verification, Power Gating, ESL, SystemC TLM.

"... tudo posso naquele que me fortalece. Nada e ninguém no mundo vai me fazer desistir..."

Pe. Fábio de Melo

"Não somos responsáveis apenas pelo que fazemos mas, também pelo que deixamos de fazer."

Moliére

Dedicatória

Dedico este trabalho aos meus pais Geraldo e Nevinha, aos meus irmãos (ã) Gilson, Júnior e Giovanna, as minhas cunhadas, sobrinhas e sobrinhos e, em especial, a minha noiva Ravely.

Agradecimentos

Agradeço, primeiramente, a Deus, pela minha vida e por mais esta conquista. Meus agradecimentos especiais aos meus pais, Geraldo e Nevinha a quem eu devo tudo, aos meus irmãos (ã) Gilson, Júnior e Giovanna, as minhas cunhadas, sobrinhas e sobrinhos. Quero agradecer com muito carinho e amor a minha noiva e futura esposa Ravely “Maguinha”, obrigado por estar ao meu lado todos os dias dessa jornada, pela sua enorme paciência durante essa caminhada.

Obrigado ao meu orientador professor Elmar, “Alemão”, pelo incentivo, dedicação, amizade e pelo envolvimento em todo o processo desta tese, e um sincero obrigado à professora Joseana, pela participação no meu doutorado.

Aos integrantes do CAFA-CLUB, Wilson, Henrique, Isaac, Zurita, Osman, Fagner, Matheus e aos demais integrantes e ex-integrantes do LAD, Alisson, Jorgeluis, Leandro, Fabrício, Sérgio, Helder, Maria e Dani, a todos vocês, muito obrigado pelas contribuições durante a execução do meu trabalho.

Quero agradecer, em especial, aos meus grandes amigos Tibério e Custódio, pela amizade e apoio em diversas fases da minha vida.

A Ângela, da COPELE, sempre disposta a ajudar e resolver todos os problemas que ocorreram durante o doutorado.

Agradeço à agência CAPES por financiar o desenvolvimento do trabalho.

As pessoas que não tiveram o nome citado aqui, mas que de alguma forma também contribuíram para esse trabalho, meu muito obrigado.

Obrigado!

Sumário

Lista de Abreviaturas

ASIC	*Application-Specific Integrated Circuits*
ATPG	*Automatic Test Pattern Generator*
BCD	*Binary Coded Decimal*
BVE-COVER	*Brazil-IP Verification Extension Coverage Library*
CI	*Circuito Integrado*
CMOS	*Complementary Metal Oxide Semiconductor*
CPF	*Common Power Format*
DSP	*Digital Signal Processor*
DUV	*Design Under Verification*
DVFS	*Dynamic Voltage and Frequency Scaling*
EDA	*Electronic Design Automation*
ESL	*Electronic System-Level*
FIFO	*First In First Out*
FPGA	*Field-Programmable Gate Array*
SC_FIFO_LP	*First In First Out - Low Power*
GND	*Ground*
GPU	*Graphics Processing Unit*
HDL	*Hardware Description Language*
IP	*Intellectual Property*
ITRS	*International Technology Roadmap for Semiconductors*
LPC	*Low Power Coalition*
MOS	*Metal Oxide Semiconductor*
NoC	*Network-on-Chip*
OSCI	*Open SystemC Initiative*
PCG	*Power Gating Controller*
PCM	*Power Controller Management*
PDML	*Power Definition Markup Language*
PGFB	*Power Gated Functional Block*

RTL	*Register Transfer-Level*
SC_LP	*SystemC - Low Power*
SC_LP_V2	*SystemC - Low Power-Version 2*
SC_LP_V3	*SystemC - Low Power-Version 3*
SCV	*SystemC Verification Library*
Si2	*Silicon Integration Initiative*
SPVR	*Speaker Verification*
SoC	*System-on-Chip*
TCL	*Tool Control Language*
TLM	*Transaction-Level Modeling*
TDP	*Thermal Design Power*
TTM	*Time-to-Market*
UPF	*Unified Power Format*
VHDL	*Very High Speed Integrated Circuits*
WSN	*Wireless Sensor Network*
EDR	*Enhanced Data Rate*

Lista de Figuras

Lista de Tabelas

Capítulo 1 - Introdução

Atualmente, os circuitos integrados (CI) ou *chips* são empregados em diversos segmentos, uma vez que são amplamente utilizados em produtos eletrônicos, telecomunicações, informática, indústria automotiva etc. Na última década, as indústrias de dispositivos eletrônicos estão investindo fortemente no desenvolvimento de sistemas complexos em um único *chip*, estes dispositivos são conhecidos por SoC (*System-on-Chip*) e, atualmente, são amplamente aplicados em dispositivos portáteis como, por exemplo, telefones celulares, câmeras fotográficas e filmadoras digitais.

Os SoC são constituídos fundamentalmente por blocos, que são núcleos de propriedade intelectual (IP). Os IP são projetados para executar tarefas específicas em um determinado sistema, podendo ser componentes programáveis (processadores) ou não, como por exemplo, dispositivos de entrada e saída. A comunicação entre os diferentes IP no SoC é realizada por estruturas de comunicação, tais como, barramentos compartilhados ou NoC (*Network-on-Chip*) (KEATING *et al.*, 2007).

Atualmente, existem CI com mais de 4 bilhões de transistores, por exemplo, a GPU (*Graphics Processing Unit*) AMD® Radeon HD 7970 (MORIMOTO, 2011). Com os diversos recursos adicionados em pequenos dispositivos eletrônicos, dois fatores devem ser considerados: (i) o aumento na complexidade no fluxo de desenvolvimento dos componentes semicondutores, principalmente no processo de verificação, (ii) o aumento do seu consumo energético, proporcionados pela elevada quantidade de funcionalidades implementadas. Por exemplo, a TDP (*Thermal Design Power)* de AMD® Radeon HD7970 é de 250 W. Devido a esses fatores, ocorreram profundas mudanças no processo de

desenvolvimento na indústria de semicondutores nos últimos anos e, principalmente, nas etapas de verificação (KEATING *et al.*, 2007).

A indústria tem investido no desenvolvimento de chips com baixo consumo de energia e vem se deparando com problemas provenientes da tecnologia nano-CMOS, principalmente em relação ao consumo de potência estática dos chips. Existe um conjunto de técnicas que podem ser aplicadas em diversas fases do fluxo de desenvolvimento com objetivo de reduzir o consumo, mas, a aplicação dessas técnicas não é totalmente automatizada e aumenta a complexidade no processo de verificação. Estes fatores restringem a aplicação de algumas dessas técnicas pela indústria.

1.1. Motivação e Contexto

A verificação é um processo utilizado para demonstrar que a funcionalidade do projeto é preservada na sua implementação, ou seja, assegura que o projeto atende aos requisitos funcionais definidos na sua especificação funcional (BERGERON, 2006). O esforço com o processo de verificação de um SoC pode ser responsável por 70% do total gasto no seu desenvolvimento, mesmo com as melhores metodologias e ferramentas de verificação (BERGERON, 2006).

O processo de verificação é executado sobre um modelo abstrato, em que a abstração é uma função inversa do nível de precisão. No presente trabalho, a precisão de um modelo indica a exatidão e corretude em descrever o comportamento e a funcionalidade do projeto proposto em detalhes, utilizando alguma linguagem HDL (*Hardware Description Language*). Diante da complexidade do SoC, é comum modelar o projeto no nível de sistema ESL (*Electronic System-Level*) utilizando o conjunto de técnicas TLM (*Transaction-Level Modeling*) e/ou no nível RTL (*Register Transfer-Level*) fazendo uso, por exemplo, da linguagem HDL SystemC. Com base nesses modelos, é possível integrar e verificar os diversos módulos funcionais (*Hardware/Software*) com altos níveis de abstração (BLACK *et al.*, 2010; GHENASSIA, 2005).

No nível de abstração RTL são caracterizados os detalhes das funcionalidades dos blocos e das interfaces de comunicação, definindo os pinos de cada barramento e a lógica relacionada a esses. Devido à quantidade de detalhes, o processo de simulação RTL é mais

lento do que ESL, elevando os custos e o tempo de desenvolvimento (BOMBIERI; FUMMI; GUARNIERI, 2011; LEE; CHANG, 2011). No nível de abstração ESL, os detalhes da comunicação entre os módulos são separados da implementação das unidades funcionais ou da arquitetura de comunicação. Os mecanismos de comunicação como barramentos ou FIFOs (*First In First Out*) são modelados como canais abstratos, em que a ênfase é dada na funcionalidade da transferência dos dados e não na implementação. Essa aproximação torna mais fácil para o projetista, no nível de sistema, a tarefa de realizar a exploração arquitetural do sistema.

Com relação à energia consumida pelos SoC, nas décadas passadas foi levantada a questão a respeito do crescente aumento do consumo de energia pelos chips (PEDRAM, 1996). Nos últimos anos, a importância sobre a restrição do consumo de energia dos dispositivos eletrônicos aumentou rapidamente, à medida que a tecnologia do processo de fabricação dos circuitos eletrônicos avançou e à medida que o número de transistores que compõe o dispositivo aumentou (KEATING et al., 2007; MATTOS; JR.; PILLA, 2009; MOHANTY et al., 2008). Existem diversas razões para reduzir a energia consumida pelos SoC, por exemplo, aumento do tempo de funcionamento com uma carga da bateria, redução do desgaste, redução de custos com resfriamento do dispositivo eletrônico e redução do impacto ambiental (*carbon footprint*) (JIANG et al., 2009; KAWA, 2008; MOHANTY et al., 2008; STAN et al., 2010).

O consumo total de energia de um circuito integrado é um somatório do consumo de energia dinâmica e estática (KEATING *et al.*, 2007). Técnicas para reduzir a potência dinâmica, energia consumida durante o chaveamento do transistor, se tornaram comum. Com tecnologias de processo CMOS inferiores a 90 nm, a potência estática[1], energia desperdiçada por causa das correntes de "fuga", se tornou relevante e, em muitos casos, passou a ser uma restrição dominante no projeto (IEEE-SA, 2009). Em trabalhos recentes, pesquisadores têm relatado que tecnologias de processos de fabricação menores que 90 nm possuem potência estática e dinâmica equivalentes (IEEE-SA, 2009; JADCHERLA *et al.*, 2009; KAWA, 2008) e, de acordo com as previsões da ITRS (*International Technology Roadmap for*

[1] *Leakage Power*

Semiconductors) com relação à potência dinâmica e estática, esta relação tende a se inverter nos próximos anos (ITRS, 2010).

As técnicas para redução do consumo de energia podem ser aplicadas durante o desenvolvimento de um SoC, desde a concepção e especificação do sistema até a fase da geração do *layout* (KEATING *et al.*, 2007). Dentre as principais técnicas aplicadas atualmente para desenvolver SoC *low power*, se destacam: *Clock Gating, Multi-*V_{th}*, Power Gating, Voltage Islands, Logic Restructuring* e DVFS *(Dynamic Voltage and Frequency Scaling).* Essas técnicas podem ser combinadas entre si e exigem funcionalidades adicionais como controladores de gerenciamento de energia, células de isolamento de domínios de energia e/ou registradores para retenção de valores lógicos (CHEN; LIN, 2008; HENZLER, 2007; JADCHERLA et al., 2009).

Este trabalho é baseado na técnica de *Power Gating*, que parte da seguinte premissa: se durante o funcionamento de um SoC, um determinado submódulo não está sendo necessário, então ele será desligado e, caso venha a ser necessário, ele será religado. Durante o período em que o submódulo permanece desligado, ocorrerá a redução do consumo de energia estática (KEATING *et al.*, 2007).

A escolha pela técnica de *Power Gating* foi baseada nos fatores descritos a seguir. De acordo com algumas pesquisas, essa técnica é a maneira mais eficiente para reduzir o consumo de energia estática nos dispositivos semicondutores (CHEN; LIN, 2008; SATHANUR; CALIMERA; et al., 2008; UPASANI et al., 2009), podendo alcançar 96% de redução nas correntes de fuga (LAKSHMI; VAYA; VENKATARAMANAN, 2011). Existem diversas pesquisas que estão sendo realizadas para maximizar sua eficiência (JUAN et al., 2010; LEE et al., 2009; MATSUTANI et al., 2010; SATHANUR et al., 2007; SATHANUR; BENINI; et al., 2008; SINGH et al., 2010; TADA; NOTANI; NUMA, 2006; YONG; UNG, 2010). De acordo com a previsão da ITRS, *Power Gating* é uma forte candidata a liderar o *ranking* entre as técnicas mais aplicadas para reduzir o consumo de energia dos dispositivos semicondutores nos próximos anos.

No fluxo de desenvolvimento de um SoC, nos últimos anos, o processo de verificação de *Power Gating* tem sido iniciado no nível de abstração RTL, utilizando principalmente as tecnologias baseadas em CPF (*Common Power Format*) e UPF (*Unified Power Format*). CPF e UPF são linguagens de especificações abertas, que capturam do projeto todas as

informações sobre os requisitos em relação ao consumo de energia e restrições funcionais em um único formato de arquivo portável e consistente (IEEE-SA, 2009; SI2, 2008). Essas linguagens apresentam uma solução para o problema das linguagens HDL com alto nível de abstração, pois, não possuem elementos sintáticos que descrevam os requisitos relacionados ao consumo de energia do projeto (KAPOOR *et al.*, 2009).

1.2. Definição do Problema e Relevância do Trabalho

No nível de abstração RTL, a técnica de *Power Gating* proporciona um considerável aumento na complexidade do processo de verificação do SoC por ser uma técnica invasiva (LIU *et al.*, 2010). A técnica afeta as interfaces de comunicação interna do módulo funcional *Power Gated* e adiciona atrasos significativos devido aos eventos de ativação e desativação, dificultando o processo de verificação do SoC (GIRARD; NICOLICI; WEN, 2009; JADCHERLA et al., 2009; KEATING et al., 2007; PFI, 2009). Para os projetos atuais de SoC, o processo de verificação no nível RTL é um desafio para os engenheiros.

De acordo com os trabalhos relacionados, vários projetos vêm aplicando a técnica de *Power Gating* em conjunto com outras técnicas, sendo, inicialmente, verificadas funcionalmente por meio do uso de CPF ou UPF no nível RTL (CHEN; LIN, 2009; GAMMIE et al., 2010; HOANG et al., 2011; HSIEH et al., 2008; JOSEPH; S.; SANKARAPANDIAMMAL, 2009; KISSLER et al., 2011; KUWAHARA, 2010; LAKSHMI; VAYA; VENKATARAMANAN, 2011). As tecnologias que oferecem suporte a CPF e UPF, que são baseadas em simulações, limitam a verificação até o nível de abstração RTL (PINTO, 2008).

De acordo com algumas pesquisas, iniciando o processo de verificação no nível ESL, o processo se tornará mais eficiente e ágil, pois, nesse nível, é mais fácil delimitar o espaço de verificação do projeto e trabalhar a microarquitetura do projeto, com o objetivo de facilitar a aplicação da técnica de *Power Gating* (PINTO, 2008; SHIBATA *et al.*, 2010). Destaca-se, também, que existem alguns trabalhos que apresentam a aplicação e verificação de algumas técnicas no nível ESL (COURT; KELLY, 2011; LEBRETON; VIVET, 2008).

Diante dos fatores: (i) aumento da complexidade introduzida por *Power Gating* ao processo de verificação, (ii) limitações tecnológicas que restringem a verificação até o nível RTL, (iii) necessidade de alcançar mais eficiência e reduzir os problemas no processo de

verificação, existe a necessidade de aplicar e verificar, inicialmente, a técnica de *Power Gating* em um nível de abstração maior do que RTL, no âmbito deste trabalho, ESL.

1.3. Objetivo Geral

O objetivo deste trabalho consiste em desenvolver uma abordagem para verificação funcional no nível ESL de circuitos integrados, que utilizem a técnica de *Power Gating*.

1.3.1. Objetivos Específicos

- Desenvolvimento de uma ferramenta de simulação para sistemas descritos no nível ESL, com base em SystemC-TLM, a ser utilizada na verificação funcional da técnica de *Power Gating* aplicada ao sistema;
- Da mesma forma que a ferramenta possibilita a simulação da técnica de *Power Gating* no nível ESL, essa deverá oferecer a funcionalidade no nível RTL, possibilitando a análise comparativa da verificação funcional ESL *vs*. RTL;
- Validação das técnicas desenvolvidas utilizando estudos de caso que permitam a obtenção de resultados pertinentes em relação a verificação funcional da técnica de Power Gating.

1.4. Metodologia

Para a realização do trabalho, foi realizado um levantamento do estado da arte na área. Foram obtidas informações sobre os processos de verificação de circuitos integrados digitais e das técnicas para redução do consumo de energia desses circuitos. Foram analisados as implicações e estratégias na aplicação e verificação das técnicas, com ênfase no processo de verificação funcional e na técnica de *Power Gating*.

Com base no estado da arte, para alcançar o objetivo deste trabalho, foi desenvolvida uma abordagem, utilizando um simulador funcional, para realizar a verificação funcional de *Power Gating*, nos níveis ESL e RTL. O simulador foi denominado de SystemC-*Low Power* (SC-LP). A estratégia utilizada é semelhante à desenvolvida para simular reconfiguração dinâmica e parcial (BRITO; MELCHER; ROSAS, 2006; BRITO et al., 2007). No contexto deste trabalho, tem-se uma abordagem *bottom-up*, que adiciona funções para ativar e desativar

submódulos pelo programador, simulando a desativação e reativação desses submódulos no sistema. Para auxiliar o simulador na simulação de projetos digitais contendo *Power Gating*, no nível ESL, foi desenvolvida uma nova classe SC_FIFO, denominada SC_FIFO_LP, na qual foi adicionado o conceito de tempo às transações. Esta característica é necessária, pois o comportamento de *Power Gating* é dependente de informações temporais.

Estudos de caso foram realizados, com base em projetos de circuitos digitais desenvolvidos no contexto do programa Brazil-IP[2]. Os resultados obtidos demonstram a eficácia da ferramenta desenvolvida, viabilizando o uso da abordagem no processo de verificação funcional de circuitos digitais contendo *Power Gating* nos níveis ESL e RTL.

1.5. Contribuições

Neste trabalho, foi apresentado um simulador SystemC, que foi modificado para suportar a simulação de projetos de circuitos digitais que contenham a técnica de *Power Gating,* descrito no nível ESL e/ou RTL, com base em SystemC. Também é apresentada uma abordagem que descreve os passos para aplicação e verificação da técnica de *Power Gating*, utilizando o simulador SC-LP. Diante dessas características, a principal contribuição ao estado da arte, é possibilitar a execução do processo de verificação funcional de circuitos digitais, que contenham a técnica de *Power Gating*, no nível ESL.

Outra contribuição importante é a possibilidade de explorar e analisar a arquitetura dos projetos de circuitos digitais, no nível ESL. Dessa forma, distribuindo as funcionalidades do sistema entre os submódulos funcionais, com o objetivo de maximizar a economia de energia, com base na aplicação de *Power Gating*. Além disso, os resultados obtidos na verificação funcional, no nível ESL, podem ser confrontados com os resultados obtidos no nível RTL. É importante salientar, que a verificação utilizando o simulador SC-LP, é independente da metodologia de verificação.

O desenvolvimento deste trabalho proporcionou a publicação de um artigo e de um capítulo de livro. O artigo foi publicado no ano de 2009, abordando a verificação funcional da técnica de *Power Gating*, no nível RTL, utilizando SystemC (SILVEIRA; BRITO; MELCHER,

[2] Consórcio para a Formação de Talentos Humanos na Concepção e Projeto de Sistemas Digitais e Propriedade Intelectual.

2009). O capítulo de livro foi publicado no ano de 2010, sobre reconfiguração dinâmica de sistemas, que aborda a correlação entre a reconfiguração dinâmica e a técnica de *Power Gating* (BRITO; SILVEIRA; UWE, 2010). Em 2012, foram submetidos dois artigos a periódicos, o primeiro artigo foi submetido ao periódico *International Journal of Reconfigurable Computing* (JIRC), abordando a verificação funcional da técnica de *Power Gating*, no nível RTL, utilizando SystemC. O segundo artigo foi submetido ao periódico da IEEE *Transactions on Very Large Scale Integration* (VLSI), abordando a verificação funcional da técnica de *Power Gating*, no nível ESL, utilizando SystemC-TLM.

1.6. Organização do Documento

Este documento está organizado em sete capítulos:

- No Capítulo 1, foi apresentada uma introdução ao problema e destacado o objetivo desta Tese;
- Nos Capítulos 2 e 3, são apresentados uma revisão da literatura e o estado da arte sobre verificação e gerenciamento do consumo de energia nos circuitos digitais;
- No Capítulo 4, são apresentados uma revisão da literatura e o estado da arte sobre a verificação de projetos de baixo de consumo de energia e trabalhos relacionados;
- No Capítulo 5, é descrito o simulador SystemC - *Low Power*, ressaltando o modelo de simulação baseado em eventos e seu processo de desenvolvimento. Também é apresentada uma abordagem metodológica para aplicação, simulação e verificação da técnica de *Power Gating* nos níveis ESL e RTL;
- No Capítulo 6, são apresentados os estudos de caso e análises dos resultados alcançados;
- No Capítulo 7, são apresentadas as considerações finais sobre este trabalho, caracterizando as contribuições para o estado da arte e sugestões para trabalhos futuros.

Capítulo 2 - Verificação

Neste capítulo, é apresentada, inicialmente, uma definição sobre os conceitos básicos da verificação de circuitos integrados, abrangendo os níveis de abstração em que pode ser executada. Em seguida, é abordado o conjunto de técnicas TLM utilizadas na modelagem do projeto no nível de sistema. Por fim, são descritas as principais tecnologias e métricas utilizadas atualmente no contexto da verificação.

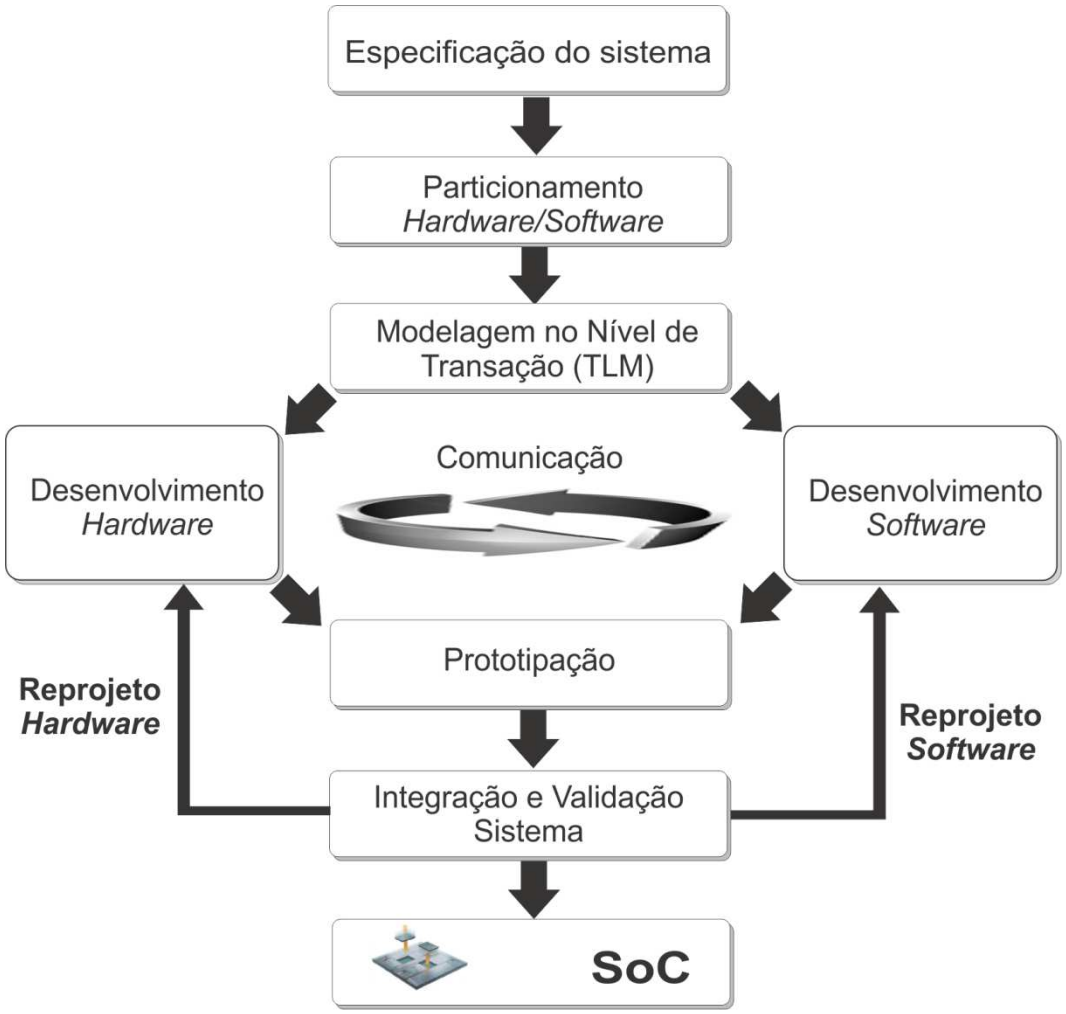
2.1. Fluxo de projeto de um SoC.

O projeto de um circuito integrado possui um fluxo de criação composto por diversas etapas. O fluxo de desenvolvimento é bastante rigoroso e segue metodologias consistentes para disponibilizar um projeto de circuito integrado para a produção final com alto nível de qualidade. Não se tem um consenso entre os trabalhos da área sobre os elementos conceituais de cada etapa. No entanto, para que o leitor possa se orientar em relação ao fluxo do projeto, as etapas podem ser descritas da seguinte forma: especificação do sistema, particionamento dos elementos em *Hardware* e *Software* (H/S), modelagem no nível de transação (TLM), desenvolvimento do (H/S), prototipação, integração e validação do sistema, e SoC (GHENASSIA, 2005). Na Figura 2.1, é apresentado o diagrama do fluxo de projeto do SoC.

O desenvolvimento é iniciado com a especificação do sistema, baseada nos requisitos do sistema, que são obtidos, principalmente, por meio de pesquisas junto ao cliente ou mercado consumidor, e dizem respeito às funcionalidades, tamanho do chip, consumo de energia, velocidade de processamento etc. Com base na especificação geral do sistema, ocorre a etapa de particionamento, em que o projeto é dividido em dois caminhos distintos,

Hardware e *Software.* Baseado no particionamento dos requisitos, os submódulos do sistema são modelados no nível de transação, a fim de serem implementados na etapa de desenvolvimento. Os submódulos implementados em (H/S) seguem para etapa de prototipação, que pode ser realizada, por exemplo, em FPGA (*Field-Programmable Gate Array)*. Na etapa seguinte, ocorre a integração do (H/S) e validação do sistema. Se forem detectados problemas em algum dos projetos, o processo será iterado, reprojeto, como indicado na Figura 2.1. Estes laços podem se repetir até que seja alcançado o resultado almejado.

Figura 2.1 - Fluxo de projeto de um SoC.

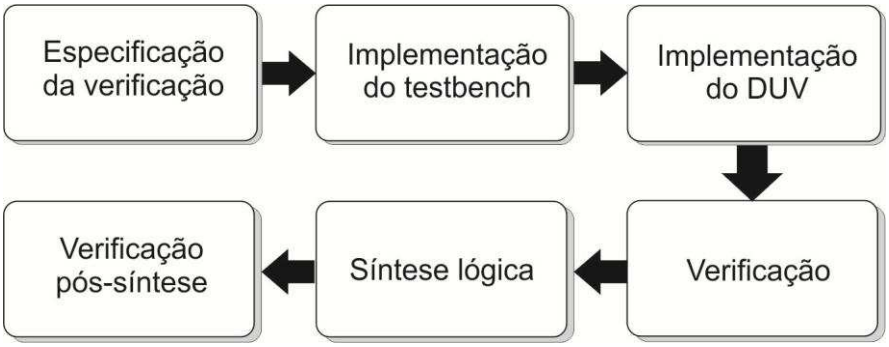


Fonte: Baseada em (GHENASSIA, 2005).

O processo de verificação é executado durante a etapa de desenvolvimento. Esta etapa é, normalmente, subdividida em 6 fases e, neste trabalho, foi baseada na metodologia

de verificação VeriSC (SILVA, 2007). Na Figura 2.2, é apresentado o fluxo contendo as fases durante o desenvolvimento do projeto.

Figura 2.2 - Fases do processo de desenvolvimento.



Descrição das fases:

- Especificação da verificação, a qual possui o objetivo de guiar o processo de verificação de um dispositivo. A especificação contém os aspectos que devem ser verificados no SoC. O documento é redigido em formato de texto e normalmente quem realiza essa especificação é o engenheiro de verificação, mas todos os participantes do projeto devem ser consultados;
- Implementação do *testbench* é a fase em que é construído o ambiente no qual o projeto do SoC (DUV) será inserido. Nesse ambiente, o DUV receberá estímulos. As respostas referentes aos estímulos serão comparadas com o resultado ideal gerado pelo modelo de referência. O *testbench* deve ser implementado preferencialmente em um nível alto de abstração denominado de nível de transação. Os engenheiros responsáveis pela implementação, preferencialmente, devem ser divididos em dois grupos; os que implementam o modelo de referência e os que implementam os módulos gerador de estímulos (*Source*) e verificador de respostas (*Checker*). Essa divisão é necessária para evitar a inserção de erros, devido à má interpretação da especificação da verificação;
- Implementação do DUV é a fase em que é descrito o sistema utilizando uma linguagem de *hardware*, tal como VHDL, Verilog, SystemC, entre outras. O DUV é o modelo que nas fases seguintes será sintetizado, por um procedimento em grande parte automatizado, para uma arquitetura alvo como, por exemplo, FPGA e/ou ASIC.

A equipe de engenheiros responsáveis pela implementação, preferencialmente, deve ser diferente das equipes que implementaram o *testbench*. Isto é necessário para evitar a propagação de erros, entre as fases, devido à má interpretação da especificação da verificação;

- Verificação, com base no *testbench*, é checado se as funcionalidades do DUV foram preservadas na sua implementação. Normalmente, é executado pelos engenheiros que implementaram o *testbench*;
- Síntese lógica é um processo que converte o código HDL descrito em um nível de abstração para um nível de abstração menor. Normalmente, essa etapa é automatizada e executada pela equipe de engenheiros que implementou o DUV;
- Verificação pós-síntese é essencial para garantir que os requisitos de tempo do dispositivo, como por exemplo, atrasos das portas lógicas, sejam atendido. Nessa fase, tanto aspectos de funcionalidade, quanto de tempo são levados em consideração durante a verificação. Normalmente, é executado pelos engenheiros que implementaram o *testbench*.

2.2. Conceitos básicos de verificação

A verificação é um processo utilizado para demonstrar que a funcionalidade do projeto é preservada na sua implementação, ou seja, assegurar que o projeto atenda aos requisitos definidos na sua especificação funcional (BERGERON, 2006). Não é possível afirmar que um processo de verificação tenha sido concluído por completo, pois ele apenas é capaz de mostrar a presença de erros, e não é capaz de provar sua ausência. Com a quantidade de tempo suficiente, um erro poderá ser encontrado (WIEMANN, 2008). A questão é: o receio de que ainda exista erro não detectado é suficientemente grave para justificar o gasto com um esforço maior?

O esforço com o processo de verificação de um SoC pode ser responsável por 60-80% do esforço total gasto no desenvolvimento do projeto. Isto é devido ao aumento da complexidade no chip, devido às diversas funcionalidades implementadas. Mesmo com as melhorias nas metodologias e ferramentas de verificação, esse número se mantém nos últimos 10 anos. Várias questões surgem no processo de verificação, por exemplo, Quais

estratégias e opções de tecnologia devem ser usadas? Como planejar e minimizar o tempo de verificação (RASHINKAR; PATERSON; SINGH, 2002)? O processo de verificação é longo e de alto custo, no entanto indispensável.

2.3. Níveis de abstração

O processo de verificação é executado sobre um modelo abstrato em que a abstração é uma função inversa do nível de precisão do modelo. Neste trabalho, a precisão de um modelo indica a exatidão ou corretude em descrever o comportamento e a funcionalidade do projeto proposto em detalhes, utilizando alguma linguagem HDL. Assim, quanto maior for o nível de detalhes do projeto, menor será o nível de abstração. Existem dois fatores decisivos para determinar o grau de precisão do modelo independente da modelagem (GHENASSIA, 2005; KASUYA; TESFAYE, 2007):

- **Granularidade de comunicação dos dados**: Este critério reflete a granularidade dos dados pela estrutura de comunicação utilizada pelo modelo;
- **Precisão do tempo:** O tempo determina a fidelidade do modelo em relação ao comportamento temporal pretendido. Pode ser conceitualmente entendido como uma escala de dois extremos, ou seja, *Timed* e *Untimed*. Exemplos de precisões intermediárias são: precisão de ciclo de *clock* e atraso unitário de porta lógica.

O planejamento do processo de verificação deve ser iniciado simultaneamente com a criação da especificação do sistema proposto. A especificação irá conduzir as estratégias e aspectos da metodologia de verificação. A seguir, são relacionados os níveis de abstração em que o processo de verificação pode ser executado de acordo com as etapas do fluxo de desenvolvimento do SoC (BERGERON, 2006; RASHINKAR; PATERSON; SINGH, 2002).

- **Verificação no Nível de Sistema** - Na concepção do sistema, o comportamento é modelado de acordo com as especificações. O comportamento é verificado por meio de um *testbench*, ambiente de teste, que faz uma simulação comportamental. Esse ambiente pode ser criado usando a linguagem C, C++ ou alguma HDL (*Hardware Description Language*). A verificação no nível de sistema é geralmente baseada em transações e não aborda os aspectos dos detalhes da implementação.

- **Verificação de Hardware RTL e Software** - Neste nível, o código RTL e o *testbench* são obtidos a partir da concepção do sistema e são utilizados para verificar as funcionalidades do projeto descrito em RTL. A verificação do software é realizada com base nas especificações do sistema e os arquivos de testes são obtidos da equipe de software. Dependendo dos requisitos da verificação, a integração de hardware/software pode ser realizada utilizando protótipos, sistema de prototipagem rápida, emulação, ou co-verificação de HW/SW.
- **Verificação da Netlist** - Netlist é o projeto descrito no nível de portas lógicas, é o resultado do processo de síntese lógica do código RTL. Pode ser verificada por meio de ferramentas formais e/ou simulação, com base em um modelo de referência. A verificação formal garante que o código RTL e a Netlist são logicamente equivalentes. Os requisitos de tempo de atraso das portas são verificados neste nível.
- **Verificação Física** - Neste nível, a verificação tenta assegurar que não existam violações dos requisitos de geometria física na implementação do projeto. A verificação física inclui a checagem de regras, por exemplo*, layout versus* esquemático e análise dos efeitos de antena.

No nível de abstração RTL são caracterizados os detalhes das funcionalidades dos blocos e das interfaces de comunicação, definindo os pinos de cada barramento e da lógica relacionada a eles. Devido às características citadas, o processo de simulação RTL torna-se bastante lento, elevando os custos e o tempo de desenvolvimento. Diante desses problemas, diversas pesquisas vêm sendo realizados em busca de soluções. Uma linha de pesquisa tem se destacado, baseada na adição de um nível de abstração acima do RTL, conhecido por projeto ESL, este nível estende o fluxo de projeto para *system-to*-RTL. Um conjunto de técnicas denominado de TLM tem evoluído para auxiliar no processo de verificação em ESL (BLACK *et al.*, 2010; GHENASSIA, 2005).

A partir dos objetivos deste trabalho, na próxima seção serão apresentados os detalhes sobre o processo de verificação no nível de sistema, detalhando os conceitos, princípios e vantagens de TLM.

2.4. *Transaction-Level Modeling*

Em um sistema digital, os componentes funcionam de maneira concorrente e cada um é composto por um conjunto finito de estados. Os estados internos de cada componente são representados por um conjunto de variáveis e o comportamento é modelado por um conjunto de processos concorrentes que podem se comunicar entre si, e serem executados em paralelo.

Assim como os componentes de um SoC, em TLM, modelagem no nível de trasação, a arquitetura do sistema é baseada em um conjunto de módulos integrados que se comunicam por meio de transações que fluem por canais abstratos. Esta é a característica essencial da modelagem em TLM (BAILEY; MARTIN, 2010; GHENASSIA, 2005). Esses canais são descritos em linguagens de programação de alto nível, por exemplo, SystemC.

TLM define uma transação como uma transferência de dados e/ou sincronização entre dois módulos na ocorrência de eventos em um instante de tempo. Por exemplo, a comunicação poder ser qualquer estrutura de palavra ou bit entre interfaces de comunicação ou a transferência de uma imagem.

Os detalhes da comunicação entre os módulos são separados dos detalhes de implementação das unidades funcionais ou da arquitetura de comunicação. As interfaces TLM encapsulam os protocolos de comunicação, por exemplo, FIFO (BLACK *et al.*, 2010). Para estabelecer uma comunicação, o processo simplesmente precisa ter acesso às interfaces por meio das portas de entrada/saída do módulo. Potencialmente, os dados também podem ser transmitidos entre os módulos e o ambiente de teste.

A sincronização do sistema em TLM é uma ação explícita entre os módulos, que têm que coordenar ou gerenciar um comportamento distribuído por meio deles. Assim, a cooperação entre os diferentes módulos é vital para assegurar o comportamento determinístico de um SoC TLM. A sincronização do sistema pode ser modelada por meio de eventos específicos, por exemplo, sinais e interrupções.

2.4.1. Modelos de Abstração

Em TLM, a modelagem do projeto durante o fluxo de desenvolvimento se baseia nos seguintes fatores: granularidade de comunicação dos dados e precisão do tempo para decidir a precisão do modelo. Guiadas por esses fatores, existem duas classes fundamentais: *Timed-TLM* e *Untimed-TLM* (BLACK *et al.*, 2010; GHENASSIA, 2005).

- ***Untimed-TLM*** é um modelo arquitetural direcionado ao desenvolvimento de software e verificação funcional, em que a noção do tempo não é necessária para observar as funcionalidades desejadas. A alta velocidade da simulação é o objetivo deste modelo.
- ***Timed-TLM*** é um modelo microarquitetural contendo noções de tempo essenciais para as especificações de comportamento e comunicação. O foco é a precisão da simulação necessária para o desenvolvimento de software embarcado de tempo real e para análise arquitetural.

Timed-TLM e *Untimed-TLM* são modelos adaptados com finalidades distintas, em que a interface ou comunicação entre os módulos pode ser refinada independentemente da funcionalidade e/ou do algoritmo. O objetivo é criar uma plataforma única que simule modelos diferentes de acordo com as necessidades do usuário (BLACK *et al.*, 2010; GHENASSIA, 2005).

2.4.2. Vantagens de TLM

TLM provê uma comunicação confiável em alto nível de abstração, em que é possível confrontar aspectos cruciais dentro da complexidade e o tempo de desenvolvimento do SoC (SWAN, 2006). As vantagens são baseadas nos seguintes pontos:

- Implementação e teste do software no início do desenvolvimento;
- Realização da análise arquitetural do sistema;
- Verificação de hardware/software em conjunto;
- Redução do tempo de desenvolvimento.

As vantagens que foram descritas possibilitam a redução dos custos de desenvolvimento do SoC, pois tornam o processo de verificação mais eficiente, devido à diminuição da probabilidade de detectar problemas no processo de integração entre *hardware/software* do sistema e à diminuição do número de iterações no fluxo de desenvolvimento. Parte dessas vantagens é proporcionada pela criação de um canal de comunicação abstrato por onde é possível fluir a comunicação entre hardware e software do projeto descrito em TLM.

2.4.3. Metodologia de TLM

Com base nos conceitos e modelos de abstração de TLM, será apresentada a sua metodologia. Inicia-se com os métodos tradicionais aplicados para capturar os requisitos do cliente. De acordo com as informações levantadas, é gerado um modelo TLM de alto nível. Este modelo proporcionará um refinamento dos requisitos e ajudará na captura dos parâmetros críticos para o sistema.

Caso a linguagem do projeto seja adequada e as técnicas sejam aplicadas de forma consistente em todo o fluxo, então, TLM pode ser reutilizado e refinado em vários casos de uso: modelagem algorítmica; modelagem arquitetural; plataforma virtual de desenvolvimento de software; verificação e refinamento do hardware (BLACK *et al.*, 2010).

A modelagem algorítmica auxiliará na definição dos algoritmos específicos que serão utilizados pela aplicação, por exemplo, decodificador de vídeo, criptografia, e etc. A modelagem arquitetural abrange o particionamento do hardware e software, o desempenho dos barramentos de comunicação e requisitos sobre a gestão do consumo de energia. A plataforma virtual de desenvolvimento de software permite o desenvolvimento precoce do software do sistema. O refinamento de hardware objetiva criar uma especificação livre de ambiguidade e executável, para acelerar o desenvolvimento e o processo de verificação funcional.

2.5. Tecnologias de Verificação

Os erros lógicos nos dispositivos são causados pelas discrepâncias ocorridas entre o comportamento pretendido e o comportamento observado. Esses erros podem ocorrer

devido à especificação ambígua, interpretação errada da especificação ou devido a algum erro inserido durante a implementação do DUV (*Design Under Verification*). Está disponível uma grande variedade de tecnologias para verificação capaz de detectar os erros. Essas tecnologias podem ser agrupadas em duas classes: Dinâmicas e Estáticas. Para atingir os objetivos da verificação de um SOC, uma combinação dessas classes pode ser usada (BERGERON, 2006; RASHINKAR; PATERSON; SINGH, 2002).

2.5.1. Verificação Estática

Na classe de tecnologias de verificação estática, também chamada de verificação formal, não existe a necessidade de vetores de teste e ambientes de simulação. A verificação formal prova a inexistência de erros por meio de equações matemáticas e verificação de modelos (DRECHSLER, 2004). No entanto, esse processo é bastante complexo, limitando a sua aplicabilidade (DRECHSLER, 2004). Esta classe pode se dividir em duas categorias: verificação de modelos e verificação de equivalência.

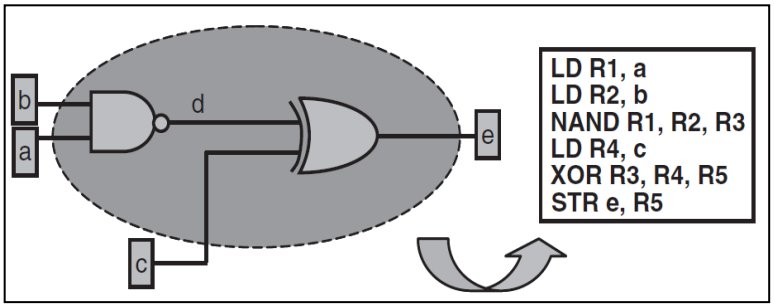
- A verificação de modelos demonstra que as propriedades definidas pelo usuário nunca podem ser violadas qualquer que seja a sequência de entradas, essa verificação é baseada em *assertions*.
- A verificação de equivalência por sua vez compara dois modelos para determinar se eles são logicamente equivalentes ou não, por exemplo, RTL *versus* Netlist.

2.5.2. Verificação Dinâmica

A verificação dinâmica, também denominada verificação funcional, é baseada em simulação. Esta é a tecnologia mais comum de verificação, pois não tem limitações de aplicação prática. Nela se cria um universo artificial que se assemelha ao mundo real. Nesse universo, várias características físicas são simplificadas ou até mesmo ignoradas para facilitar a tarefa de simulação. A velocidade de execução do simulador depende do tamanho do projeto e do nível de atividade dentro da simulação. No entanto, não há limitações na escala do projeto a ser verificado, desde que não haja restrições com relação ao tempo de simulação. Existem dois tipos básicos de simuladores: simuladores baseados em ciclos e simuladores orientados a eventos (FUJITA; GHOSH; PRASAD, 2008).

Simuladores baseados em ciclo assumem que o sinal ativo do relógio é o único evento que pode mudar o estado do circuito. Todas as entradas do circuito são sincronizadas com o relógio. Portanto, só pode simular projetos síncronos. Dessa forma, a atomicidade de tempo neste tipo de simulação é um ciclo de *clock*. Na Figura 2.3 as entradas e saídas são registradas. O simulador baseado em ciclo abstrai todos os detalhes (região sombreada da figura) na lógica entre registradores e os converte em equações funcionais.

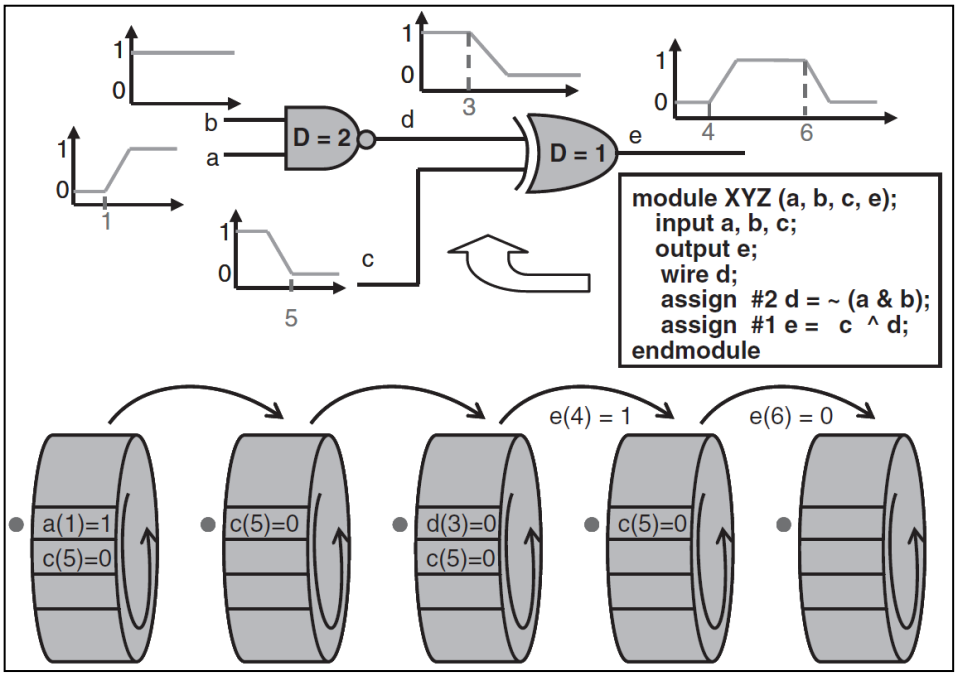
Figura 2.3 - Simulador baseado em ciclo.



Fonte: (FUJITA; GHOSH; PRASAD, 2008).

Um simulador orientado a eventos se mantém informado dos eventos ou mudanças de sinais que ocorrem no circuito ao longo do tempo, ele é ilustrado na Figura 2.4. e seu funcionamento é baseado na ocorrência do evento, processando um evento por vez.

Figura 2.4 - Simulador orientado a eventos.



Fonte: (FUJITA; GHOSH; PRASAD, 2008).

Os efeitos gerados pelos eventos são avaliados com base em uma roda do tempo em que os eventos são enfileirados, o efeito é propagado pelo projeto até que seja alcançado um estado de equilíbrio. Um elemento do projeto pode ser avaliado por várias vezes em um único ciclo, pois os eventos podem ser assíncronos, fornecendo, assim, um ambiente de simulação de alta precisão (FUJITA; GHOSH; PRASAD, 2008). Na Figura 2.4, é apresentado um circuito em Verilog RTL. No simulador, o evento na frente da fila na roda do tempo é indicado pelos pontos cinza. As formas de onda indicam as mudanças de sinais à medida que avança o tempo. Os valores de D dentro das portas lógicas representam os atrasos das portas, que são estimados neste nível.

O conceito de verificação funcional adotado neste trabalho é baseado nos livros de Bergeron (BERGERON, 2003, 2006), em que a verificação funcional é um processo que verifica, por meio de simulação, se o dispositivo implementado (DUV) está de acordo com as funcionalidades especificadas.

Para que a simulação seja possível, é necessário que haja um ambiente de verificação que possa receber o DUV, inserir estímulos e comparar suas respostas com as de um modelo ideal. Esse ambiente é denominado de *testbench*. O ponto de partida de um projeto de verificação consiste na existência de um modelo considerado ideal, uma especificação que deve ser respeitada durante todas as fases.

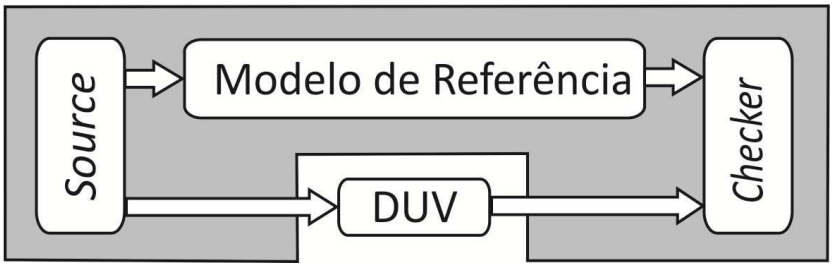
O modelo considerado ideal (Modelo de Referência) é simulado juntamente com o modelo do dispositivo, sendo verificado e seus resultados comparados. Quando o projeto é complexo, ao encontrar um erro, é necessário fazer uma investigação de todo o modelo para descobrir a causa. Um aspecto importante é decidir a granularidade do circuito a ser verificado, ou seja, definir o quanto que será subdividido. Durante a verificação de um projeto, o circuito é normalmente dividido em blocos funcionais para facilitar a sua verificação.

2.6. *Testbench*

Testbench é um ambiente de simulação que é baseado em um conjunto de modelos. Estes podem ser descritos em várias linguagens HDL e trabalham para alcançar o objetivo da verificação. Em alguns projetos, a mesma linguagem pode ser utilizada para descrever o

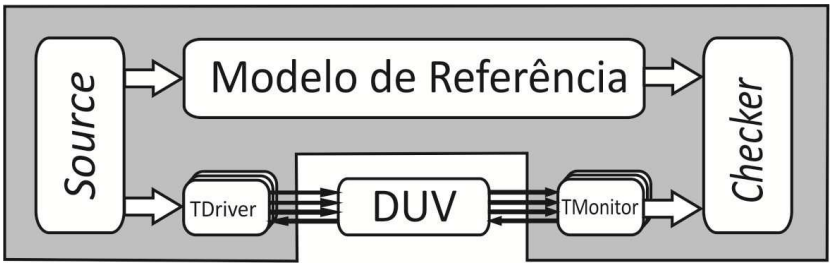
testbench e o sistema a ser verificado. A escolha da linguagem é influenciada pelas metodologias que devem ser aplicadas para realizar a verificação (BAILEY; MARTIN, 2010). Um bom *testbench* deve possuir as seguintes características básicas: ser dirigido por cobertura, possuir aleatoriedade direcionada, ser auto-verificável e baseado em transações (SILVA, 2007). A composição do *testbench* utilizada neste trabalho pode ser vista nas Figura 2.5 e Figura 2.6.

Figura 2.5 - Estrutura do *testbench* para DUV descrito em ESL.



Fonte: (SILVA, 2007).

Figura 2.6 - Estrutura do *testbench* para DUV descrito em RTL.



Fonte: (SILVA, 2007).

O *testbench* é a parte da figura que está dentro da área de cor cinza. Ele é composto dos seguintes blocos: *Source*, TDriver(s), TMonitor(es), Modelo de Referência e *Checker*. Cada interligação dos blocos é viabilizada por FIFO(s), representadas na figura por setas na cor branca. O DUV é inserido no *testbench* para ser estimulado e verificado pelo ambiente de simulação, comparando suas respostas com as de um modelo ideal. Caso o DUV seja descrito em ESL, ele será conectado ao *testbench* por FIFO(s) e se o DUV for descrito em RTL, ele será conectado por meios de interfaces formadas de sinais, representados na Figura 2.6 por setas na cor preta.

Em resumo, as estruturas do *testbench* são:

- O *Source,* é responsável por criar estímulos para a simulação, os quais devem ser cuidadosamente escolhidos para satisfazer os critérios de cobertura especificados. Todos os estímulos criados pelo *Source* são transações.
- O TDriver, é responsável por converter as transações recebidas pelo *Source* em sinais e submetê-los para o DUV, além de executar o protocolo de comunicação (*handshake*) com o DUV por meio de sinais.
- O TMonitor, executa o papel inverso do TDriver, convertendo todos os sinais de saída do DUV para transações e repassando as mesmas para o Checker, via FIFO. Além disso, executa um protocolo de comunicação com o DUV por meio de sinais.
- O *Checker*, é responsável por comparar as respostas resultantes do DUV e do Modelo de Referência para verificar se são equivalentes.
- O Modelo de Referência, contém a implementação ideal (especificada) do sistema. Portanto, ao receber estímulos, ele deve produzir respostas corretas.
- O DUV, é o projeto que será verificado e que deve ser implementado em TLM e/ou RTL. Caso seja modelo no nível RTL, é necessário um mecanismo para se comunicar com o *testbench* que trabalha em TLM. Essa comunicação se faz com a ajuda dos blocos TDriver(s) e TMonitor(es) que traduzem sinais para transações e vice-versa.
- As FIFO(s), exercem uma tarefa muito importante no *testbench*, pois elas são responsáveis por controlar o sequenciamento e o sincronismo dos dados que entram e saem delas.

Para o *testbench* da Figura 2.5, o ambiente de simulação funciona da seguinte forma, o *Source* irá gerar os estímulos, em formato de transações, que irão alimentar o Modelo de Referência e o DUV, para que sejam processados. O *Checker* irá comparar as transações, que chegam do DUV e do Modelo de Referência, através da comparação das variáveis, a fim de checar se são funcionalmente correspondentes.

Em outras palavras, para cada transação gerada pelo *Source*, o Modelo de Referência e o DUV irão realizar seus respectivos processamentos, em que, cada um, irá gerar um

resultado, baseados na mesma transação. Os resultados, serão entregues, em formato de transações, ao *Checker*. Este irá pegar as transações e checará se são iguais. O processo de verificação estará concluído quando for alcançado 100% de cobertura ou, caso seja detectado alguma disparidade entre as transações. Caso se detecte a falha no DUV, passa-se para à fase de depuração, a qual está fora do escopo deste trabalho.

2.7. Cobertura

É necessário definir as métricas para medir o progresso da verificação. A métrica de cobertura é um parâmetro pertencente ao ambiente de verificação (SILVA, 2007). São utilizados vários métodos para estimar a qualidade e progressão da verificação funcional. Alguns métodos são baseados na medição de aspectos de qualidade da simulação por meio de ferramentas, outros são baseados na metodologia utilizada no projeto. Devido às diferentes abordagens, é normalmente utilizada a combinação de várias técnicas para alcançar os objetivos. Os métodos mais utilizados podem ser agrupados em duas categorias (MEYER, 2004; PIZIALI, 2004):

- **Cobertura de código** - Na forma mais simples, são contadas as linhas de código do DUV que foram exercitadas durante a simulação.
- **Cobertura funcional** - Verifica se as funcionalidades especificadas do DUV foram exercitadas.

O progresso da verificação de um projeto pode ser medido, por exemplo, pelo número de características funcionais confirmadas corretamente (BERGERON et al., 2006).

2.7.1. Cobertura Funcional

Na verificação funcional, um grande problema é descobrir o momento de parar a simulação de forma que tenha sido coberto todo o plano de verificação. Esse problema é decorrente do fato de que a simulação em si não consegue afirmar se todas as funcionalidades implementadas foram verificadas.

A principal técnica para mostrar que uma simulação obteve sucesso é analisar a cobertura. Isso significa criar uma lista de ocorrências (funcionalidades especificadas a serem testadas) e observar se cada uma foi vista durante a simulação.

A Cobertura Funcional é uma técnica usada para medir o progresso da simulação e reportar quais funcionalidades não foram testadas, podendo ajudar no monitoramento da qualidade dos testes e direcioná-los para criar geradores de estímulos que cubram as áreas que não foram adequadamente testadas.

Os eventos são coletados diretamente da simulação durante a execução. Cada vez que um evento acontece, é feita uma atualização na lista de cobertura. No plano de verificação está definida a cobertura desejada de cada DUV a ser verificado. Quando a cobertura especificada foi atingida, a verificação daquele DUV pode ser considerada concluída.

2.8. Linguagens

Para os níveis de portas lógicas e RTL, o modelo computacional predominante para as linguagens de programação foram baseadas em modelos de eventos discretos, em que todos os elementos podem ser executados em paralelo. Com a migração para projetos em ESL, é necessária uma linguagem que seja capaz de lidar com modelos de tempo indeterminado ou parcialmente ordenados e que possam manipular dados em alto nível de abstração. Projetos ESL impõem um conjunto de exigências para a linguagem. Alguns desses requisitos são (BLACK *et al.*, 2010):

- Diferentes níveis de abstração;
- Uma linguagem aberta;
- Bom desempenho na simulação das características;
- Ferramentas de suporte;
- Suportar os conceitos de TLM.

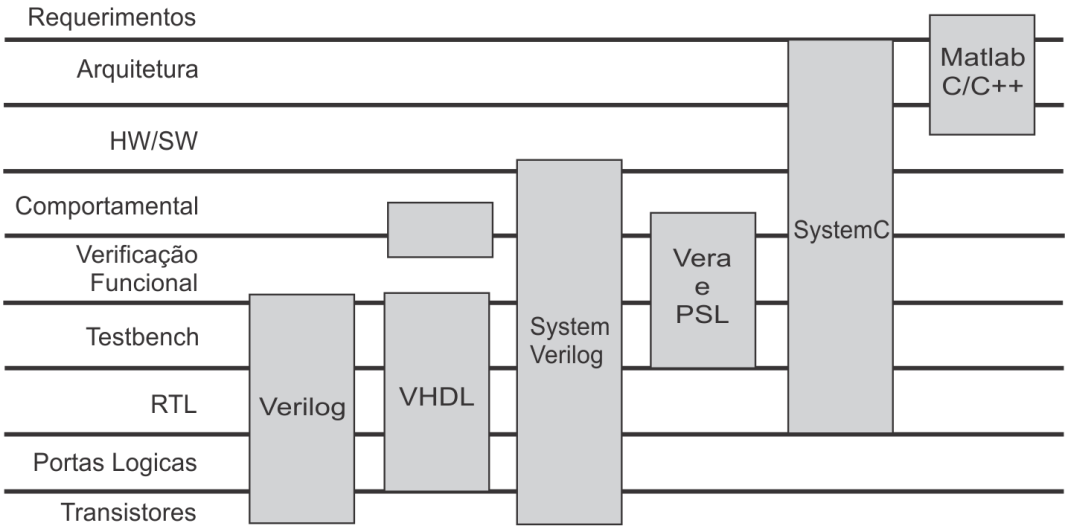
Uma linguagem bem projetada e utilizada por muitas pessoas não significa que foi bem sucedida. Um requisito para uma linguagem ser bem sucedida é o quanto que ela é adotada pela indústria. Várias linguagens têm surgido para suportar os vários aspectos da concepção de SoC.

A linguagem C/C++ é bastante utilizada para o desenvolvimento de software de sistemas embarcados. As linguagens de descrição de hardware, VHDL ou *VHSIC (Very High Speed Integrated Circuits)* Verilog e SystemVerilog, são utilizadas para simular e sintetizar

circuitos digitais. Vera *e* SystemVerilog são as linguagens escolhidas para a verificação funcional de ASICs (*Application-Specific Integrated Circuits*). SystemVerilog é uma nova linguagem, com menos de dez anos de história, que evoluiu a partir da linguagem Verilog para poder ser usada tanto para o projeto RTL como para a verificação funcional (BERGERON et al., 2006).

MATLAB e diversas outras ferramentas são amplamente utilizadas para capturar requisitos do sistema e o desenvolvimento de algoritmos para processamento de sinais. SystemC possui uma grande abrangência para os níveis de abstração mais altos dentro do fluxo de desenvolvimento de SoC, sendo uma opção de linguagem para iniciar a adoção de ESL e a modelagem TLM. Na Figura 2.7, são destacadas as aplicações de algumas linguagens para projeto de sistemas (BLACK *et al.*, 2010).

Figura 2.7 - Comparação de linguagens.



Fonte: Adaptada de (BLACK *et al.*, 2010).

SystemC foi a linguagem adotada neste trabalho, pois é possível, por meio dessa linguagem, descrever circuitos nos níveis ESL e RTL. Propriamente dito, SystemC não é uma linguagem, é uma biblioteca de classes de C++, e quando está acoplada à SCV (*SystemC Verification Library*) prover várias características relevantes para a concepção e modelagem do sistema. Além disso, proporciona uma linguagem comum com base em C/C++, para software e hardware. De acordo com a Figura 2.7, mesmo com a abrangência da aplicabilidade de SystemC, está longe de ser perfeita. Embora, inicialmente tenha sido bem

recebida pela comunidade acadêmica, a indústria não demonstrou a mesma receptividade (BLACK *et al.*, 2010).

A flexibilidade de SystemC vem de dois mecanismos. O primeiro é o uso de notificação e primitivas de espera, a segunda são as classes ***sc_interface***, ***sc_channel*** e ***sc_prim_channel*** (BLACK *et al.*, 2010). Com esses recursos, SystemC oferece suporte estático e dinâmico sobre o fluxo de dados bem como suporte a modelos de eventos discretos. Outro recurso ao seu favor é o suporte de múltiplos níveis de abstração dentro do domínio de eventos discretos (BAILEY; MARTIN, 2010).

SystemC foi padronizado em dezembro de 2006 pela IEEE, como o padrão 1666. O OSCI (*Open SystemC Initiative*) (ACELLERRA, 2012), desde o trabalho de pré-normalização, tem avançado no refinamento de SystemC. Atualmente, o principal esforço está na padronização de um conjunto de interfaces TLM.

A biblioteca TLM 2.0 é uma extensão de SystemC. Esta biblioteca adiciona novos recursos, semelhantes à semântica dos barramentos de comunicação. SystemC não define a semântica das comunicações entre os modelos, só fornece as primitivas essenciais que são necessárias para as comunicações. Hoje, está ocorrendo uma rápida adoção de TLM 2.0 pela indústria com apoio significativo de grandes desenvolvedores de EDA (*Electronic Design Automation*) (BAILEY; MARTIN, 2010).

2.9. Metodologia de Verificação

À medida que o tamanho e complexidade do projeto crescem, as equipes de desenvolvimento tentam abordar deficiências em seus processos de verificação com diferentes abordagens, tais como a verificação baseada em assertivas, transações, ou cobertura. Cada um desses métodos se concentra em uma ou duas questões individuais da verificação, como o tempo de depuração ou identificação de buracos na cobertura. Infelizmente, quando a verificação é baseada em um método ele resolve um problema, mas agrava outro (WILCOX, 2004).

As ferramentas de verificação associadas aos métodos não devem ser o centro da verificação. Essas ferramentas devem ser vistas como uma coleção de utilitários para ajudar a alcançar os objetivos da verificação. Primeiro, deve-se entender o que será realizado,

então, desenvolve-se uma metodologia utilizando as ferramentas adequadas no momento certo. Em vários projetos, algumas ferramentas podem ser muito utilizadas e, em outros, podem ser levemente utilizadas ou não.

Verificação baseada em alguma metodologia possui mais ênfase no planejamento e estratégia do que nas técnicas. A equipe alocada para verificação deve avaliar as necessidades do projeto e, em conjunto com o plano de verificação, desenvolver a melhor abordagem para atingir os objetivos. Após desenvolver uma metodologia, as ferramentas são reunidas para lidar com as metas do projeto. De acordo com Bergeron, uma metodologia deve ter quatro características básicas (BERGERON, 2006):

- **Ser dirigida por cobertura,** uma vez que é inviável simular todos os cenários possíveis de um DUV, devido à sua alta complexidade. Assim, é fundamental poder especificar quais os cenários que devem ser simulados de fato. A verificação funcional deve ser finalizada quando os cenários forem todos cobertos conforme o plano de verificação.
- **Suportar a geração aleatória de estímulos** na existência dos critérios de cobertura, as ferramentas devem gerar os estímulos necessários para satisfazer os critérios. O uso de funções pseudo-aleatórias possibilita estressar adequadamente o DUV, fazendo com que cenários mais improváveis e complexos sejam cobertos.
- **Ser auto-verificável**, pois a metodologia deve dar suporte à comparação automática dos resultados produzidos pelo DUV. No caso de divergência dos resultados, a metodologia deve reportar a discrepância.
- **Ser baseada em transações**, uma vez que é inviável trabalhar com ambientes totalmente descritos em RTL, devido ao alto nível de detalhes. Assim, o ambiente de verificação deve funcionar em um nível de abstração acima da abstração do DUV, chamado de nível de transação.

Este trabalho adotou a metodologia VeriSC (SILVA, 2007; SILVA *et al.*, 2005). Esta é composta de um fluxo de verificação que não se inicia pela implementação do DUV. Nesse fluxo, a implementação do *testbench* e do modelo de referência antecedem a do DUV. Para permitir que o *testbench* seja implementado antes do DUV, a metodologia possui um mecanismo para simular a presença do DUV com os próprios elementos do *testbench*, sem a

necessidade da geração de código adicional, que não seja reutilizável. O modelo de referência calcula todas as saídas esperadas do sistema com base nas entradas que são utilizadas durante a verificação funcional.

VeriSC é uma metodologia que adota o conceito de projetos com hierarquia, de forma que cada um possa ser dividido em partes a serem implementadas e verificadas. Ela pode ser aplicada para a verificação funcional de DUV digitais síncronos que utilizam um único sinal de relógio, podendo ser utilizada em circuito que possuam mais de um relógio, caso este possa ser dividido em vários subcircuitos com apenas um relógio (SILVA *et al.*, 2005). Essas características fazem com que a metodologia permita a criação de um *testbench* compatível com o estado da arte descrita na literatura e dos cenários de testes que possibilitem exercitar as funcionalidades especificadas em um plano de verificação.

Idealmente, o DUV e o modelo de referência devem ser implementados por equipes distintas de engenheiros. O objetivo é diminuir a probabilidade de que uma interpretação equivocada da especificação do sistema se propague para o DUV e o ambiente de verificação. Caso esse problema ocorra ao final do processo de verificação funcional, o DUV irá divergir da especificação do projeto.

2.10. Considerações Finais do Capítulo

Neste capítulo, foram apresentadas as informações básicas para a compreensão do processo de verificação, e diante da complexidade do SoC, foi demonstrado que o nível de abstração ESL pode agilizar o seu processo de verificação. Para auxiliar nessa tarefa, foi apresentado o conjunto de técnicas TLM. Para diferentes equipes do projeto, TLM é utilizado como única referência e apresenta uma metodologia confiável para superar os pontos críticos do projeto e proporcionar a interação entre as equipes. Foram abordados os conceitos sobre a verificação funcional, os componentes necessários para sua realização e as métricas para medir seu progresso. Por fim, foi apresentado um comparativo das principais linguagens utilizadas no desenvolvimento de SoC e a definição dos conceitos básicos sobre metodologia de verificação.

Capítulo 3 - Gerenciamento do Consumo de Energia

O objetivo deste capítulo é auxiliar na compreensão sobre o consumo de energia em circuitos digitais, iniciando com a resposta à seguinte questão: "por que se deve reduzir o consumo de energia dos chips?". Logo em seguida, serão apresentados de forma sucinta os principais conceitos associados à tecnologia de circuitos integrados CMOS (*Complementary Metal Oxide Semiconductor*) estático, demonstrando como ocorre a dissipação de energia em circuitos digitais que utilizam essa tecnologia e a tendência da relação entre a energia dinâmica e estática. Por fim, serão demonstradas as principais técnicas de redução de consumo com ênfase na técnica de *Power Gating*.

3.1. Por que Reduzir o Consumo de Energia dos Chips?

Nas décadas passadas, foi levantada a questão sobre o crescente aumento do consumo de energia pelos *chips* (PEDRAM, 1996). Nos últimos anos, a importância sobre a restrição do consumo de energia dos dispositivos eletrônicos aumentou rapidamente à medida que a tecnologia do processo de fabricação dos circuitos eletrônicos avançou (KEATING et al., 2007; MATTOS; JR.; PILLA, 2009; MOHANTY et al., 2008). A seguir, serão apresentadas algumas razões pelas quais é necessário reduzir o consumo de energia (KAWA, 2008; MOHANTY *et al.*, 2008; STAN *et al.*, 2010), a destacar:

- Diminuição da densidade de potência no chip;
- Aumento da vida útil da bateria;
- Aumento da margem da relação sinal-ruído (SNR);

- Redução dos custos com resfriamento dos dispositivos;
- Redução dos custos de energia;
- Aumento da confiabilidade do sistema.

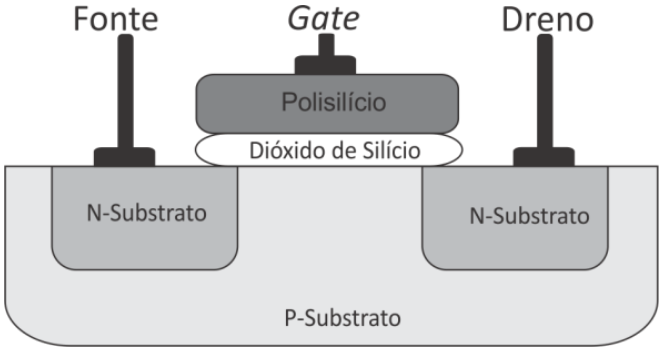
Os semicondutores são empregados em grande escala, desde os dispositivos móveis a supercomputadores. O consumo de energia tem sido alvo de otimização para a maioria dos projetos com tecnologia CMOS, pois, possui grande impacto sobre o desempenho do circuito, independente da aplicação (GAMMIE *et al.*, 2010; KEATING *et al.*, 2007).

Para o grupo de consumidores de dispositivos móveis, além das funcionalidades e desempenho satisfatórios, tais dispositivos devem evitar o rápido esgotamento das baterias. Para o segmento dos dispositivos eletrônicos que não dependem de baterias, existe uma considerável pressão para reduzir o consumo de energia mantendo o desempenho, por exemplo, em computadores pessoais e *Data Centers*, este é influenciado pelo alto custo gasto com o resfriamento dos dispositivos CMOS (JIANG et al., 2009; STAN et al., 2010).

3.2. Tecnologia CMOS

O termo CMOS possui mais de um significado, provocando confusões de comunicação e entendimento. Neste trabalho, as tecnologias ou processos de fabricação CMOS são aqueles que fornecem os dois tipos de transistores: PMOS e NMOS. Esta tecnologia é predominante no processo de desenvolvimento de SoC e ASIC. Uma porta lógica CMOS, corresponde a uma topologia de porta lógica formada por duas redes lógicas independentes e complementares (MATTOS; JR.; PILLA, 2009). Na Figura 3.1, é apresentada a estrutura física de um transistor NMOS (*Metal Oxide Semiconductor)*.

Figura 3.1 - Estrutura física do transistor NMOS.



O transistor MOS é produzido sobre uma base de substrato de silício, por exemplo, no transistor NMOS a região da base é um substrato de silício tipo "P". Duas regiões fortemente dopadas tipo "N" são adicionadas na base originando o Dreno e a Fonte. Uma camada muito fina de dióxido de silício isolante é depositada sobre a região do substrato entre o Dreno e a Fonte. Sobre a camada isolante é depositada uma camada de polisilício condutor que é denominada de *Gate*.

O funcionamento do transistor é baseado na aplicação de potencial elétrico no *Gate*. Para um transistor NMOS, caso não seja aplicada uma tensão no *Gate*, o dispositivo se comportará como um circuito aberto, resultado de uma alta impedância entre o Dreno e a Fonte. Caso seja aplicada uma tensão positiva no *Gate* e superior à tensão de limiar V_{th} (*threshold*), será induzida a criação de um canal condutor ligando a região do Dreno com a região da Fonte, se comportando como um circuito fechado.

Será visto nas próximas seções que o cenário descrito acima é mais complexo, particularmente para dispositivos nano-CMOS, devido a uma série de novos fenômenos característicos da tecnologia.

3.3. Dissipação de Potência

O consumo de energia elétrica total de um transistor MOS é uma função da atividade de transição, capacitância, tensão e da estrutura do transistor em si. Pode-se dividir o consumo em duas grandes categorias: estática e dinâmica. A potência total (P_t) é um somatório da potência dinâmica (P_d) e estática (P_e) (KEATING *et al.*, 2007).

$$P_t = P_d + P_e \,. \qquad \textbf{Eq. (1)}$$

3.3.1. Potência Dinâmica

A Potência dinâmica (P_d) é a soma de dois fatores: Potência de chaveamento (*Switching*) e curto-circuito (*Short-Circuit*), dada pela seguinte equação (CHABINI, 2007):

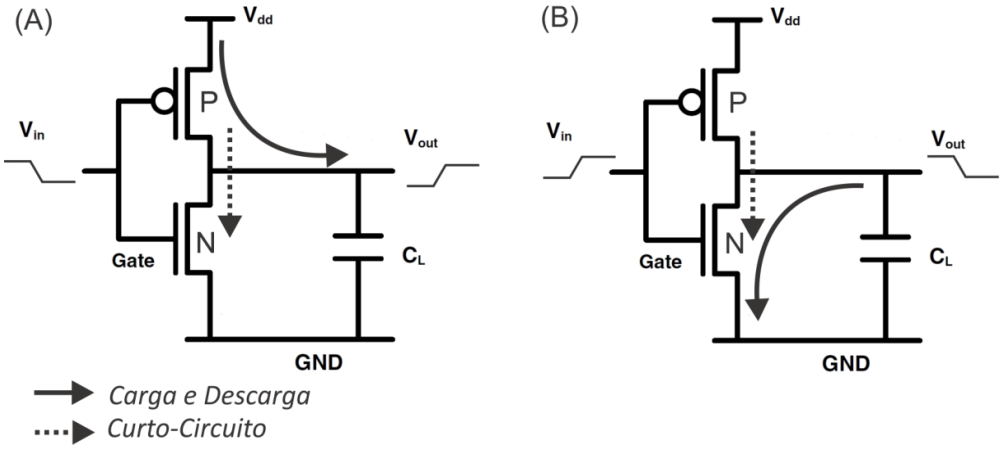
$$P_d \approx AFC_{ef}{V_{dd}}^2 \,. \qquad \textbf{Eq. (2)}$$

Em que: A = fator de atividade de chaveamento, F = frequência de *clock*, C_{ef} = capacitância efetiva, V_{dd} = tensão de alimentação.

A potência de chaveamento é dissipada durante a carga ou descarga das capacitâncias internas do transistor e das capacitâncias dos fios de interconexão. A energia de curto-circuito é a energia dissipada por um curto-circuito, que ocorre durante um curto período de tempo, na conexão entre a tensão de alimentação V_{dd} e GND (*Ground*) durante o chaveamento entre os estados do transistor (HENZLER, 2007).

Na Figura 3.2, é apresentada uma porta lógica inversora, com tecnologia CMOS. Essa porta lógica é composta por dois transistores: um tipo-P e um tipo-N. Na Figura 3.2(A), estão as representações das correntes de chaveamento (carga) e de curto-circuito. Na Figura 3.2(B), estão as representações das correntes de chaveamento (descarga) e de curto-circuito.

Figura 3.2 - Energia dinâmica em uma porta lógica inversora CMOS.

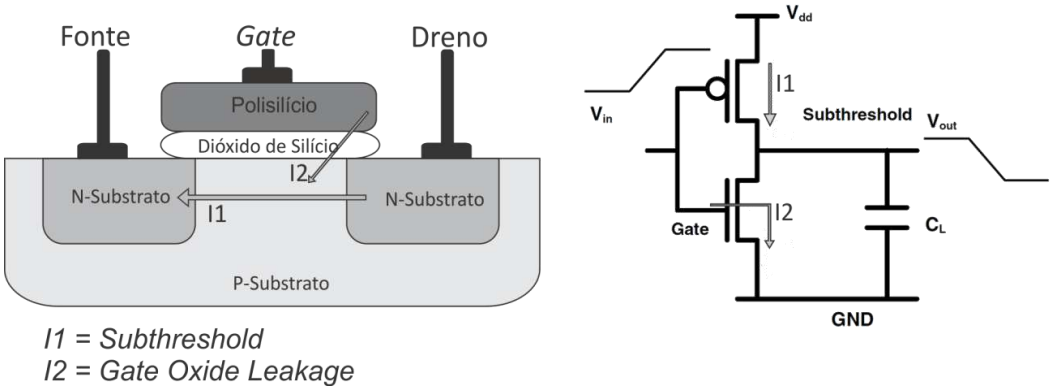


3.3.2. Potência Estática

A Potência estática corresponde à energia dissipada quando o circuito não está chaveando, ou seja, quando o circuito está inativo. Existem diversas origens para a dissipação da potência estática, a maior parte é resultante da “fuga" das correntes entre o Dreno e a Fonte (*Subthreshold*) e por meio do dióxido de silício (*Gate Oxide Leakage*). É importante ressaltar, que ao contrário da corrente de *Subthreshold*, que ocorre quando o dispositivo não está conduzindo, a corrente por meio do dióxido de silício está presente na existência de uma diferença de potencial. Em outras palavras, independente do estado do

transistor, esta corrente existirá (MATTOS; JR.; PILLA, 2009). Na Figura 3.3, são apresentadas as principais correntes de fuga em uma porta lógica inversora CMOS.

Figura 3.3 - Principais correntes de fuga em uma porta lógica inversora CMOS.



A potência estática (P_e) é dada pela seguinte equação (BUTZEN et al., 2010):

$$P_e = V_{dd} I_{leakage}\,, \qquad \textbf{Eq. (3)}$$

Em que: $I_{leakage}$ = somatório das correntes de *"fuga"* e V_{dd}= tensão de alimentação.

A corrente de fuga total pode ser aproximada pela soma das correntes de *Subthreshold* e *Gate Oxide Leakage.* Essas são representadas pelas seguintes equações (BUTZEN et al., 2010):

$$I_s = I_0 e^{(V_{gs} - V_{th}/nV_T)}\left[1 - e^{-(V_{ds}/V_T)}\right], \qquad \textbf{Eq. (4)}$$

$$I_g = WLA\left(\frac{V_{ox}}{t_{ox}}\right)^2 exp\left(\frac{-B\left(1-\left(1-(V_{ox}/\phi_{ox})\right)^{3/2}\right)}{V_{ox}/t_{ox}}\right). \qquad \textbf{Eq. (5)}$$

Em que: I_s = corrente de *Subthreshold*, I_g = corrente *Gate Oxide Leakage*, $I_0 = \left(W\mu_0 C_{ox} {V_T}^2 e^{1.8}\right)/L$, V_T = tensão térmica, V_{th} = tensão de limiar (*threshold*), V_{ds}= tensão *Drain-Source*, V_{gs}= tensão *Gate-Source*, W = largura do transistor, L = comprimento do transistor, C_{ox} = capacitância do óxido do *Gate*, μ_0= mobilidade de portadores de carga, n = *subthreshold swing coefficient,* V_{ox} = potencial por meio do óxido, t_{ox} = espessura do óxido, $A = q^3/16\pi^2 h\phi_{ox}$, $B = 4\pi\sqrt{2m_{ox}}\,{\phi_{ox}}^{3/2}/3hq$, m_{ox} = massa efetiva da partícula em tunelamento, ϕ_{ox} = barreira de potencial para o tunelamento, h = $1/2\pi$ vezes a constante de Planck, q= carga do elétron.

Ao longo da evolução da tecnologia CMOS, a tensão de V_{dd} vem sendo reduzida para diminuir a dissipação de potência dinâmica e estática. Mas, a tensão Threshold V_{th} também vem sendo reduzida para garantir a eficiência em termos de velocidade de comutação do transistor. Consequentemente, de acordo com as equações apresentadas, diminuindo a tensão V_{th}, as correntes de "fuga" crescem exponencialmente, tornando-se uma das principais preocupações nos projetos de CI em escala nanométrica (BUTZEN et al., 2008).

3.4. Tendência do Consumo em Tecnologias CMOS

Sistemas móveis dependentes de baterias, como telefones celulares e computadores portáteis, tornaram-se um segmento estratégico para a indústria eletrônica. Nos processos de fabricação com tamanho do *Gate* acima de 90 nm, o consumo de energia dinâmica é uma importante restrição de projeto (IEEE-SA, 2009). Técnicas para reduzir o consumo de energia dinâmica, energia consumida durante o chaveamento do transistor, se tornaram comuns. Foi relativamente simples a aplicação dessas técnicas ao fluxo do projeto, pois as técnicas são baseadas no gerenciamento do sinal de *clock* e normalmente são automatizadas por meio de ferramentas EDA (PFI, 2009).

Com tecnologias de processo inferior a 90 nm, o consumo de energia estática se tornou relevante e, em muitos casos, é uma restrição dominante no projeto. Devido às questões físicas dos transistores demonstrados na Seção 3.3, mesmo quando o circuito está em repouso, sem alternância dos estados internos dos transistores, ocorre um consumo considerável devido às correntes de fuga (IEEE-SA, 2009).

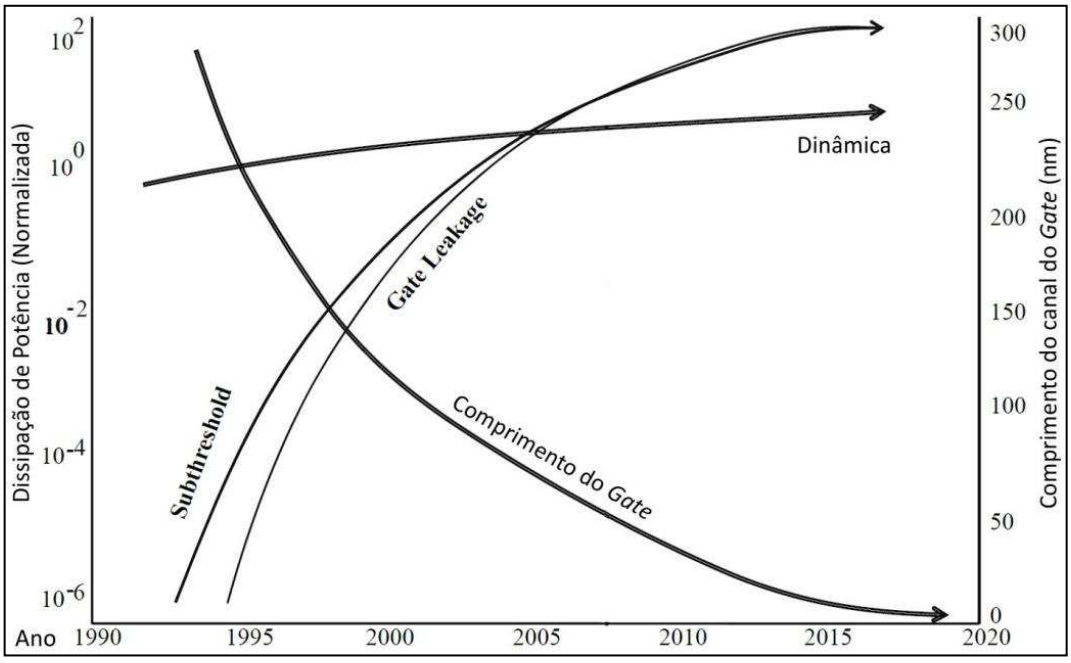
Devido às técnicas para reduzir o consumo de energia dinâmica e dos novos processos de fabricação dos transistores utilizados nos chips, a relação entre as energias, dinâmica *versus* estática, teve uma considerável diminuição. Em trabalhos recentes, pesquisadores têm relatado que tecnologias de processos de fabricação menores que 90 nm possuem consumo de energia estática próxima ao da dinâmica (IEEE-SA, 2009; JADCHERLA *et al.*, 2009; KAWA, 2008).

A principal fonte de informações sobre o estado atual da tecnologia de semicondutores em geral, em particular nano tecnologia CMOS, é o ITRS (*International Technology Roadmap for Semiconductors*) (ITRS, 2010). O ITRS é um consórcio internacional

composto por grandes fabricantes de semicondutores, fornecedores de equipamentos e instituições acadêmicas. Este consórcio tem por objetivo identificar os principais desafios tecnológicos da indústria de semicondutores para os próximos anos por meio de um relatório, contendo estimativas dos principais desafios.

Na Figura 3.4, é apresentado um gráfico em escala logarítmica com a previsão da relação entre o consumo de energia estática *versus* dinâmica, com base nos dados da ITRS (MOHANTY *et al.*, 2008). O gráfico apresenta as principais fontes de dissipação de energia em nano-transistores e circuitos integrados CMOS, a partir do qual é possível observar que em 2020 a indústria estará trabalhando com processo de nano-CMOS com *Gate* de 9 nm e que as correntes de *Subthreshold* e *Gate Leakage* hoje já são dominantes no somatório da energia total.

Figura 3.4 - Tendência do consumo de energia dos circuitos integrados CMOS.



Fonte: Figura adaptada de (MOHANTY *et al.*, 2008).

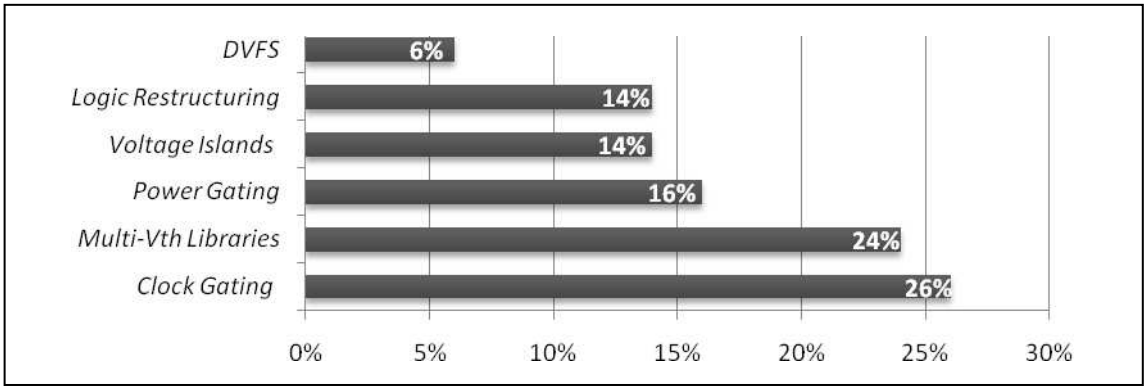
De acordo com as tendências da tecnologia CMOS, novas técnicas foram desenvolvidas para gerenciar o consumo de energia estática, por exemplo, técnicas de desligamento automático para um conjunto de elementos lógicos ou técnicas baseadas no limiar da tensão V_{th} do transistor. Para as técnicas baseadas no desligamento automático de um conjunto de elementos lógicos e de acordo com a Equação 3, com o $V_{dd} = 0$ a potência

estática dissipada pelo conjunto de elementos lógicos será igual a 0. Para as técnicas baseadas no limiar de tensão do transistor e de acordo com a equação 4, a corrente de *Subthreshold* é inversamente proporcional à tensão V_{th}. Assim, quanto maior for a tensão de V_{th}, menor será a corrente de *Subthreshold*, mas em consequência a velocidade de comutação entre os estados internos do transistor será menor (GIRARD; NICOLICI; WEN, 2009). Portanto, é necessário buscar um valor de V_{th} que reduza a dissipação de potência estática sem comprometer o desempenho do transistor.

3.5. Técnicas *Low Power*

Várias técnicas foram e estão sendo desenvolvidas para diminuir o consumo de energia dos circuitos integrados (KEATING *et al.*, 2007). Os projetistas possuem várias formas de gerenciar as técnicas para controlar o consumo de energia. Algumas são fáceis de implementar, outras possuem uma alta complexidade com relação à frequência de operação, área ocupada no chip (GAMMIE *et al.*, 2010; KAWA, 2008) e com relação ao trabalho de desenvolvimento para sua implementação no projeto do SoC. No ano de 2007, foi realizada uma pesquisa com o objetivo de estimar quais técnicas eram mais utilizadas pelos projetistas para gerenciar o consumo de energia. Foram consultadas 115 empresas, abrangendo os segmentos das telecomunicações, dispositivos eletrônicos portáveis e equipamentos de redes (SINGH, 2008). A Figura 3.5 contém as preferências das empresas pesquisadas.

Figura 3.5 - Popularidade das técnicas de gerenciamento de energia.



Fonte: Figura adaptada de (SINGH, 2008).

Dentre as principais técnicas aplicadas atualmente para desenvolver SoC de baixo consumo, destacam-se: *Clock Gating, Multi-V_{th}, Power Gating, Voltage Islands, Logic*

Restructuring e *Dynamic Voltage and Frequency Scaling* (DVFS)**.** Essas técnicas podem ser combinadas e exigem funcionalidades adicionais como controladores de gerenciamento de energia, células de isolamento de domínios de energia e/ou registradores para retenção de valores lógicos (CHEN; LIN, 2008; HENZLER, 2007; JADCHERLA et al., 2009):

- ***Clock Gating***: Técnica baseada na interrupção do sinal de *clock* de *flip-flops* cujo valor fica preservado naquele ciclo de *clock.*
- ***Voltage Islands***: Técnica aplicada com o objetivo de obter submódulos funcionais do circuito alimentados com diferentes níveis de tensão.
- ***Multi-V_{th}***: Técnica baseada na utilização de portas *standard cell* com diferentes níveis de V_{th} no mesmo circuito. As *standard cells* com baixo V_{th} possuirão menor atraso e alto consumo de energia estática, e as *standard cells* com alto V_{th} possuirão maior atraso e baixo consumo de energia estática.
- ***Power-Gating:*** Técnica baseada na suspensão da alimentação de submódulos funcionais do SoC enquanto o restante do SoC permanece em funcionamento. É uma técnica sofisticada, com implicações de tempo e alta complexidade de implementação. Essa técnica proporciona tipicamente uma redução no consumo de energia estática diferentimente de outras técnicas.
- ***Logic Restructuring:*** Técnica aplicada durante o processo de síntese utilizando como critério o nível de atividade nas entradas das portas lógicas. Algumas portas lógicas são remapeadas para uma combinação de portas lógicas equivalentes.
- ***DVFS***: Técnica que opera em diferentes blocos, variando o nível de tensão de alimentação e/ou frequência de módulos funcionais de acordo com as exigências de desempenho durante o funcionamento.

A escolha das técnicas para gerenciar o consumo de energia abrange uma variedade de possibilidades, impactando em benefícios e penalidades ao projeto. Naturalmente, o impacto das técnicas de gerenciamento de energia pode variar com base no projeto e na implementação dessas. As técnicas mais comuns são apresentadas na Tabela 3.1.

Tabela 3.1 - Comparação entre as técnicas de baixo consumo.

	Redução da Energia Dinâmica	**Redução na Energia Estática**	**Penalidade no Tempo**	**Penalidade na Área**	**Penalidade na Complexidade e *Time-to-Market* (TTM)**	**Impacto na Implementação**	**Impacto no Projeto**	**Impacto na Verificação**
DVFS	40–70%	2–3X	~0%, Adição de *level shifters*; escalonamento do sinal de *clock* devido à alteração da latência dinâmica.	< 10%, Adição de *level shifters* e unidade de gerenciamento de energia.	Alto tempo de projeto, tempo de resposta e TTM	Alto	Alto	Alto
Logic Restructuring	<5%	~0X	~0%	Baixo	Desprezível	Desprezível	Desprezível	Desprezível
Voltage Islands	40–50%	2X	~0%, Adição de *level shifters*; escalonamento do sinal de *clock* devido à alteração da latência.	<10%, Roteamento e interconexão do domínios de energia; adição de *level shifters.*	Alto tempo de projeto, tempo de resposta e TTM	Médio	Médio	Baixo
Power Gating	~0%	10–50X	4–8%, Adição das células de Isolamento, complexidade do tempo, atraso no tempo para ativação, picos de corrente.	5–15%, Adição de células de isolamento, retenção de estado, unidade de gerenciamento de energia e rede de chaves para alimentação.	Arquitetura no nível de sistema, suporte para controle de energia, verificação, síntese e implementação.	Médio - Alto	Alto	Alto
Multi-V_{th}	0%	2–3X	~0%	< 2%	Baixo	Baixo	Desprezível	Desprezível
Clock Gating	20%	~0X	~0%	< 2%	Desprezível	Baixo	Baixo	Desprezível

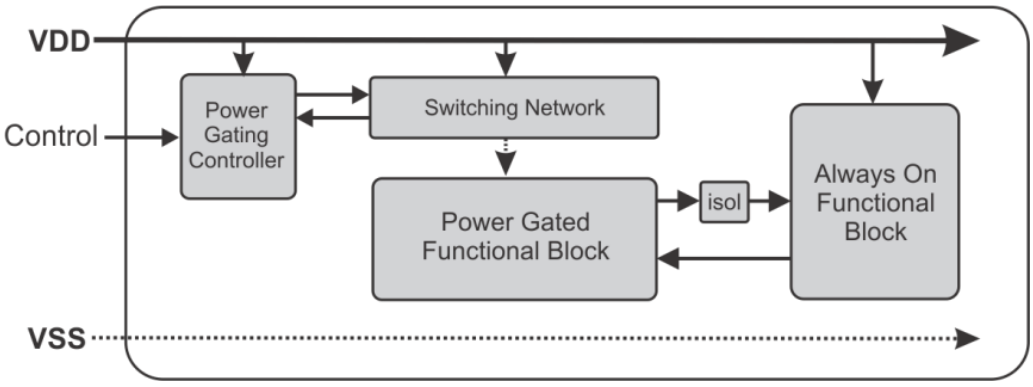
Fonte: Tabela adaptada (PFI, 2009).

3.6. Visão Geral de *Power Gating*

Este trabalho é baseado na técnica de *Power Gating*, por ser a técnica tipicamente mais eficiente na redução da potência estática, mas, ao mesmo tempo a técnica que representa o maior desafio para o projetista. Assim, nesta seção será descrito um breve resumo sobre a técnica. *Power Gating* é maneira mais eficiente de reduzir a energia estática (CHEN; LIN, 2008; UPASANI et al., 2009), Sathanur *et al* (SATHANUR; CALIMERA; et al., 2008) fazem uma análise da aplicação da técnica e demonstram a efetividade na redução do consumo de energia estática nos dispositivos semicondutores.

A técnica parte da seguinte situação: se um determinado submódulo funcional PGFB (*Power Gated Functional Block*) de um SoC não está sendo utilizado, então ele pode ser desligado e caso venha a ser necessário, ele será religado. A estratégia de *Power Gating* é baseada na adição de mecanismos para desligar PGFB quando não são necessários para o funcionamento do SoC. Em síntese, a técnica utiliza o conceito de domínio de energia para descrever blocos comutáveis (ligado ou desligado). Um domínio de energia é uma coleção de elementos do projeto que compartilham entre si a mesma fonte primária de energia (KEATING *et al.*, 2007). Na Figura 3.6, é apresentado o diagrama em blocos com a estrutura do SoC com *Power Gating*.

Figura 3.6 - Diagrama em blocos de um SoC com *Power Gating.*



Fonte: Figura baseada de (KEATING *et al.*, 2007).

Diferentemente de um bloco que está sempre em atividade, um submódulo PGFB é alimentado com VDD ou VSS a partir de um conjunto de chaves CMOS. Estas chaves são distribuídas ao redor do bloco funcional e seu controle é realizado por um bloco

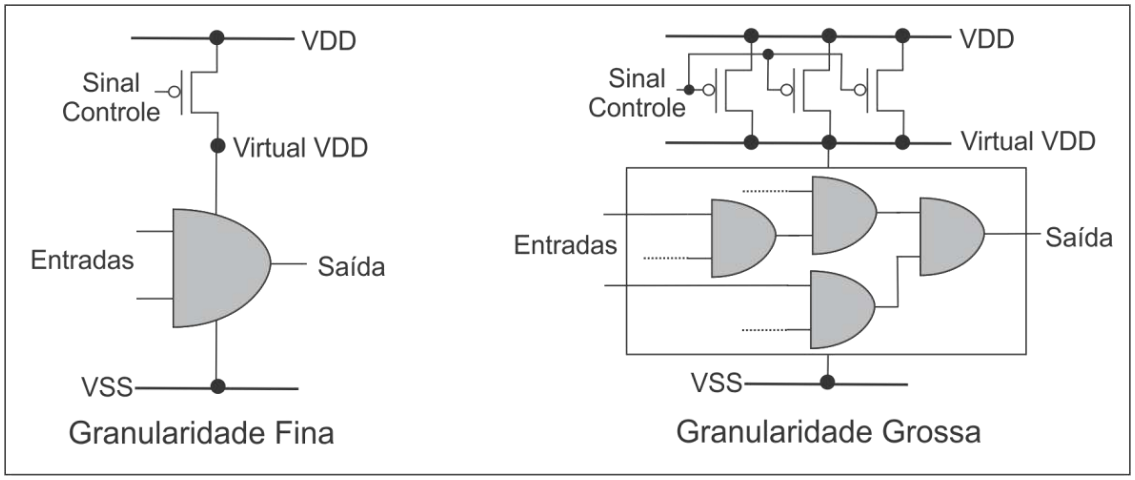
denominado PCG (*Power Gating Controller*). As chaves são transistores, conhecidos por transistores do "sono".

A técnica de *Power Gating* é invasiva, pois afeta as interfaces de comunicação interna do SoC e adiciona atrasos significativos devido aos eventos de ativação e desativação. Estes eventos devem ser realizados em momentos adequados para alcançar a economia de energia mantendo o desempenho especificado para a aplicação (KEATING *et al.*, 2007).

3.6.1. Granulariedade da Técnica

A técnica de *Power Gating* pode ser classificada em dois estilos, granularidade fina e grossa. Na Figura 3.7, são apresentados os estilos de granularidade fina e grossa.

Figura 3.7 - Estilos de granularidade de *Power Gating*.



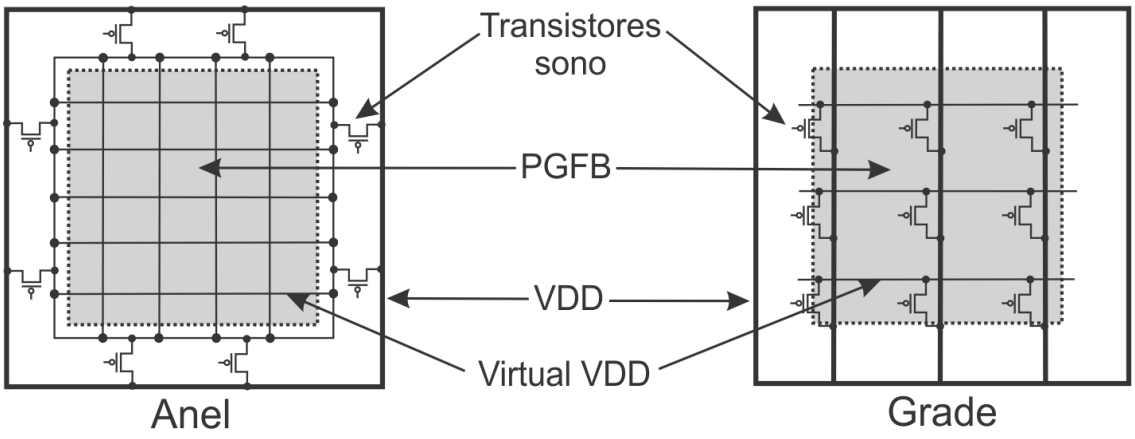
Na granulariedade fina, é inserido um transistor do sono em cada célula, estas células são conhecidas como MTCMOS (*Multi-Threshold* CMOS). Uma vez que o domínio de energia é dividido em várias regiões, os transistores do sono são controlados individualmente. As principais vantagens desse estilo são: diminuição do ruído, pequena latência do evento de ativação e a diminuição da corrente de pico. As desvantagens estão: no acréscimo de área devido à adição dos transistores do sono, este aumento pode ser de até 3 vezes e na distribuição e roteamentos dos transistores do sono pelo bloco funcional (KAWANO *et al.*, 2011; KEATING *et al.*, 2007).

No esquema de granularidade grossa, um grande bloco de circuito é alimentado por um único domínio de energia. São utilizados transistores de sono ligados em paralelo, entre o domínio de energia principal e o domínio virtual, os quais são comandados por um único sinal. As vantagens são, menor sensibilidade à variação PVT (processo, voltagem, temperatura) dos transistores do sono, menor queda de tensão nos fios de alimentação e menor consumo de área. As desvantagens são: necessidade de uma rede de energia complexa para alimentar o bloco funcional, maior ruído e latência no evento de ativação, necessidade de uma lógica especial e restrições no processo de síntese física (KEATING *et al.*, 2007). Na maioria dos projetos, um aumento da área por um fator de três teria um aumento de custo acima da especificação. Portanto, na maioria dos projetos tem sido utilizada a granularidade grossa (KEATING *et al.*, 2007).

3.6.2. Rede de Transistores

Power Gating sobrecarrega a área de silício com o acréscimo e o roteamento da rede de transistores que alimentam o domínio de energia do PGFB e introduz questões sobre a integridade da potência, com respeito a *IR drop*, queda na tensão ao longo do fio condutor que alimentam os transistores, a latência para restaurar a alimentação do PGFB e o pico de corrente após o evento de ativação. Existem algumas formas de distribuir os transistores para alimentar o bloco funcional, as principais são anel e grade, a escolha do estilo vai depender do projetista do *layout* (KEATING et al., 2007; SOFER; BERKOVITZ; NEIMAN, 2011). Na Figura 3.8, são apresentados os dois estilos.

Figura 3.8 - Estilos das redes de transistores.

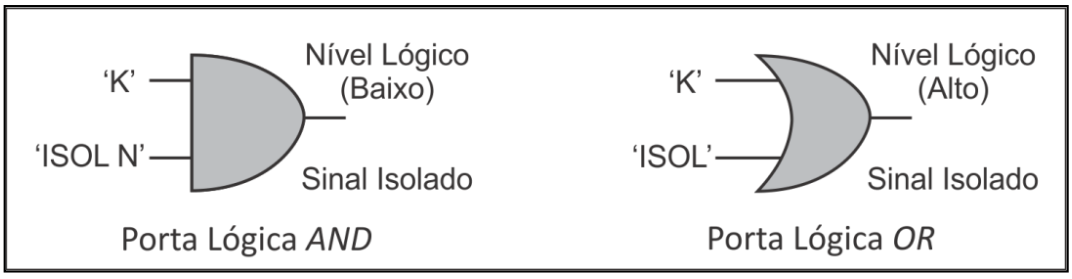


3.6.3. Isolamento do Sinal

Cada domínio de energia representa uma parte da área física do chip, e mesmo sendo independentemente comutável, essas áreas permanecem fisicamente conectadas enquanto estão ativadas ou desativadas. Entre os domínios de energia, durante o processo de ativação, pode ocorrer um pico de corrente. Este é devido ao atraso que o PGFB sofre para atingir a tensão adequada de funcionamento, resultando em um período significativo na faixa da tensão V_{th}. Portanto, é necessário um isolamento lógico entre os domínios afim de evitar a flutuação dos sinais e a propagação de sinais indesejados quando estiver desativado (KEATING *et al.*, 2007).

O isolamento lógico é realizado por meio de células de isolamento. Estas células são portas lógicas que determinam os valores de entrada e/ou saída quando o domínio de energia está desligado. Na Figura 3.6, o isolamento está representado pelo bloco "isol". O isolamento deve ser aplicado antes do desligamento da energia e permanecer até que o PGFB esteja plenamente em funcionamento. Na Figura 3.9, são apresentados dois tipos de células de isolamento, contituídas pelas portas lógicas AND e OR.

Figura 3.9 - Células de isolamento lógico.



O funcionamento das células de isolamento é baseado na função lógica da porta que a constitui. Para a porta AND, se o sinal de isolamento "ISOLN" estiver no nível lógico baixo, a saída irá para nível lógico baixo, caso "ISOLN " esteja no nível lógico alto, a saída será igual a "K". Para a porta OR, se o sinal de isolamento "ISOL" estiver no nível lógico alto, a saída irá para nível lógico alto, caso "ISOL " esteja no nível lógico baixo, a saída será igual a "K".

3.6.4. Retenção do Estado

Com o desligamento do PGFB, as informações internas sobre o estado são perdidas, podendo ser um inconveniente em determinadas aplicações. Após a reativação e retomada da atividade PGFB, este deve partir do estado inicial (*reset*), o que pode provocar um significativo consumo de energia e tempo. Mas, caso seja empregada uma estratégia de retenção do estado interno, o estado anterior ao desligamento pode ser restaurado após a reativação (HENZLER, 2007).

A estratégia de retenção depende das características do subsistema. Por exemplo, uma unidade digital de processamento de sinais que seja orientado ao fluxo de dados, pode ser capaz de reiniciar seu processamento com base em novos dados de entrada, neste caso, a retenção não é necessária. No entanto, um periférico ou uma cache de um processador, que geralmente tem o estado interno residual considerável, quando for reativado e para que retome seu funcionamento normal é possível que seja necessário restaurar seu estado anterior, acarretando em uma transferência de dados excessiva pelo barramento, para buscar as informações necessárias, provocando atraso e consumo de energia desnecessário, nesta caso, é indicado o uso da retenção (BAILEY et al., 2008; KEATING et al., 2007).

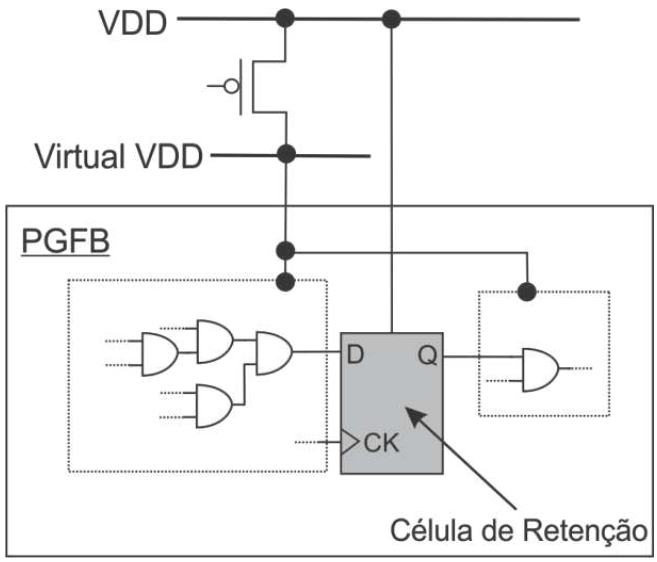
Uma das principais vantagens da retenção do estado do PGFB é a transparência da técnica para o projeto. Para manter a retenção transparente, os sinais de *clock* e *reset* do PGFB não podem estar ativos durante a retenção. As desvantagens são o aumento da área e a diminuição da economia de energia do SoC: (i) com relação à área, é necessário adicionar mais lógica ao projeto para proporcionar a retenção, o aumento é tipicamente de 20%, podendo chegar a 50% sobre a área total do bloco funcional; (ii) sobre o consumo de energia, para permitir a retenção, a lógica adicional que implementa a retenção, deve ser alimentada por um domínio de energia que esteja sempre energisado. Como a lógica da retenção pertence ao PGFB, durante a desativação, a economia de energia será menor (HENZLER, 2007; JADCHERLA *et al.*, 2009; KEATING *et al.*, 2007).

Sendo assim, uma das principais decisões sobre a arquitetura de *Power Gating* é definir quais estados, informações, devem ser armazenados durante o desligamento do bloco funcional, a fim de minimizar o impacto sobre o consumo de energia originado pela

lógica da retenção. Baseado nessas decisões, é aplicada a retenção parcial dos estados, em que as informações estratégicas do bloco funcional são retidas. Esse tipo de retenção adiciona desafios ao processo de verificação, pois o responsável pela verificação terá que lidar com a combinação de estados retidos e não retidos no PGFB e não permitir que os valores dos estados sejam corrompidos (KEATING *et al.*, 2007).

Existem vários métodos para salvar e restaurar um estado interno do PGFB, sendo baseados em *software*, *scan chains* e registradores. Independente do método, o objetivo é que a estratégia adotada para retenção do estado seja rápida e eficiente (JUAN et al., 2010; KEATING et al., 2007; LEE et al., 2009) fornecendo ao PGFB um método para retomar rapidamente o pleno funcionamento após ativação. Na Figura 3.10, é apresentado a estrutura de PGFB contendo uma célula de retenção, observe que a célula, *flip-flop*, é alimentada por outro domínio de energia, assim, quando o bloco funcional estiver desligado, o *flip-flop* continuará retendo o valor lógico.

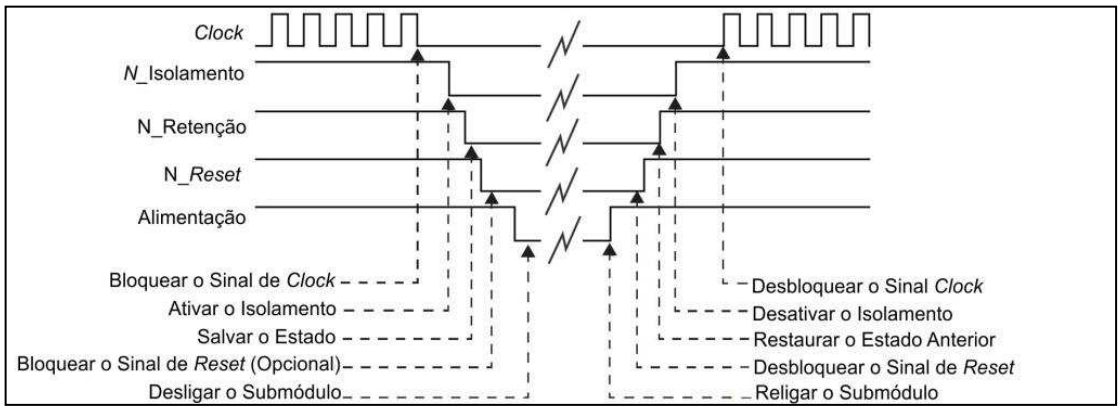
Figura 3.10 - Estrutura de PGFB com retenção.



3.6.5. Sequência da Comutação do PGFB

Na Figura 3.11, é apresentada a sequência do processo de ativação e desativação do PGFB no nível de sinais. Ordem dos eventos para desativar: suspender o sinal de *clock*; aplicar o isolamento lógico; reter o estado; desabilitar o sinal de *reset*; e por último suspender a alimentação do domínio de energia. Para reativar, a ordem será inversa.

Figura 3.11 - Sequência de ativação e desativação do PGFB.

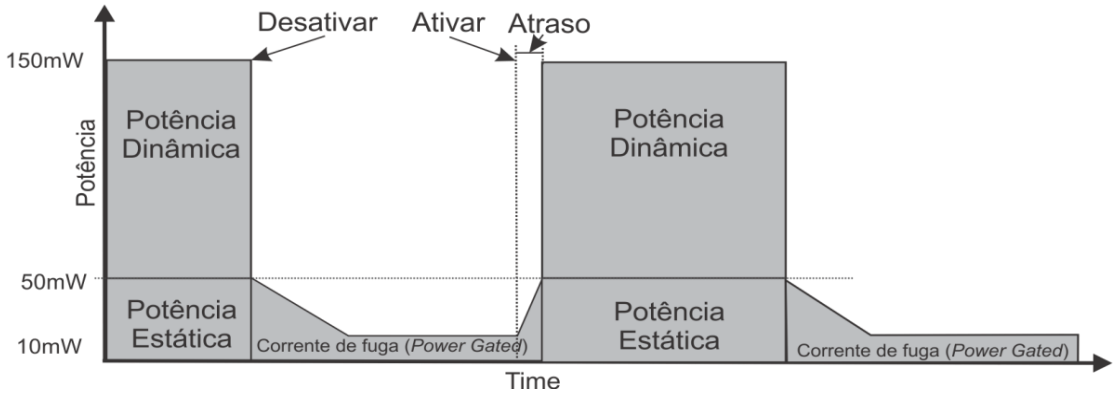


Fonte: Figura adaptada de (JADCHERLA *et al.*, 2009).

3.6.6. Comportamento do Consumo de Energia com *Power Gating*

Quando o evento de desativação ocorre, a economia de energia não é instantânea devido às questões térmicas geradas pela atividade executada antes da desativação e da natureza não ideal da tecnologia CMOS empregada na técnica *Power Gating* (GIRARD; NICOLICI; WEN, 2009)(KEATING *et al.*, 2007). No evento de ativação do bloco, é necessário aguardar uma quantidade de tempo para que a PGFB esteja energizado adequadamente para que possa retomar sua atividade (KEATING *et al.*, 2007). Essa quantidade de tempo é baseada no tempo que leva para carregar todas as capacitâncias do bloco funcional (XU; LI, 2011). Diante da natureza da técnica de *Power Gating* ser *Timed*, a modelagem da técnica no nível ESL está restrita ao nível de abstração, com informação temporal. Na Figura 3.12, é apresentado um exemplo do comportamento do consumo de energia em relação ao tempo de acordo com a atividade do PGFB.

Figura 3.12 - Perfil do consumo de energia de *Power Gating.*



Fonte: Figura adaptada de (KEATING *et al.*, 2007).

Com a popularização da técnica de *Power Gating* e o aumento do consumo de energia estática diante dos avanços da tecnologia CMOS, diversas pesquisas estão sendo realizadas com o objetivo de aprimorar a técnica (ROY; RANGANATHAN; KATKOORI, 2011; SATHANUR et al., 2007; SATHANUR; BENINI; et al., 2008; TADA; NOTANI; NUMA, 2006; YONG; UNG, 2010). Alguns trabalhos possuem um objetivo específico, por exemplo, diminuir o atraso que ocorre no processo durante a ativação (JUAN et al., 2010; LEE et al., 2009; MATSUTANI et al., 2010; SINGH et al., 2010), melhorar o processo de desativação e maximizar a economia de energia enquanto o PGFB está desativado (CHEN; LIN; TUNG, 2010; GUO et al., 2008; XU; JONE; VEMURI, 2008). Outros trabalhos buscam melhorar a forma como é alimentado o bloco funcional (KIM; KIM; CHOI, 2011; PAKBAZNIA; PEDRAM, 2012; SOFER; BERKOVITZ; NEIMAN, 2011; TAKAI; HASHIMOTO; ONOYE, 2011).

3.7. Considerações Finais do Capítulo

Neste capítulo, foram apresentadas as informações básicas sobre o consumo de energia em dispositivos semicondutores com tecnologia CMOS, demonstrando as principais fontes de dissipação de energia. Diante das necessidades do mercado consumidor, as restrições sobre o consumo de energia se tornaram uma das prioridades nos projetos de dispositivos semicondutores. De acordo com as previsões da ITRS, com relação à energia dinâmica e estática, consumo de potência dinâmica superior à estática, esta relação tende a se inverter e as técnicas que tem o foco na redução da energia estática serão prioritárias. A técnica de *Power Gating*, diante da sua eficiência atual e do número de pesquisas que estão sendo realizadas para maximizar sua eficiência, é uma forte candidata a liderar no futuro o *ranking* entre as técnicas mais aplicadas para reduzir o consumo de energia dos dispositivos semicondutores.

Capítulo 4 - Verificação de Projeto de Baixo Consumo

Nos capítulos anteriores, foram apresentados os conceitos básicos que são aplicados neste trabalho. Abordou-se a tecnologia de verificação dinâmica realizada no nível ESL e as origens do consumo de energia dos circuitos digitais, com ênfase na tecnologia CMOS e a técnica de *Power Gating*.

Neste capítulo, será apresentado o estado da arte no processo de verificação funcional de projetos que contenham técnicas de redução de consumo de energia. Serão abordadas questões metodológicas envolvendo os níveis de abstração e a tecnologias adotadas no processo de verificação funcional, com ênfase na técnica de *Power* Gating.

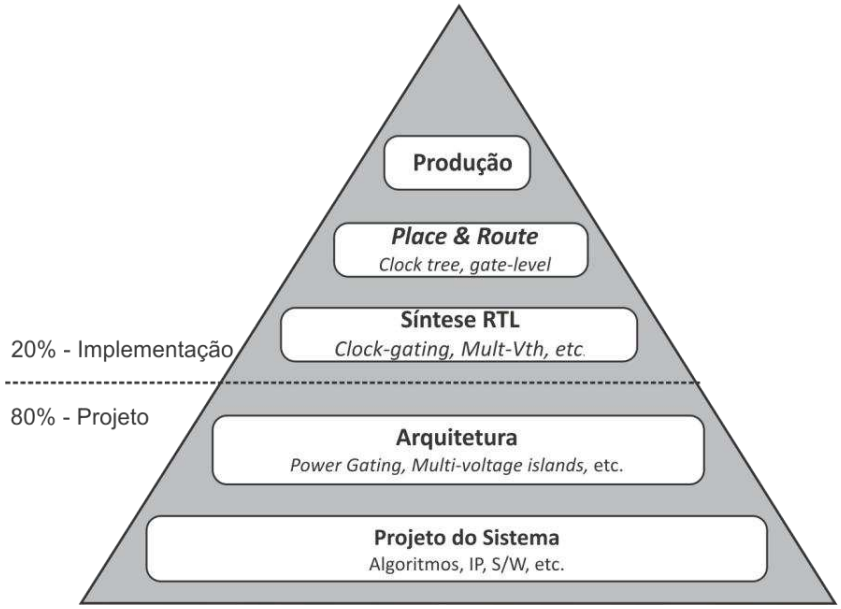
4.1. Introdução

Atualmente, as técnicas mais eficientes para reduzir o consumo de energia nos CI são baseados no desligamento e/ou redução do nível da tensão em domínios de energia, e pelo controle do sinal de *clock* em determinados módulos funcionais (CHEN; LIN, 2009; KAPOOR et al., 2009; KUWAHARA, 2010). É consenso na literatura que as decisões sobre as restrições do consumo de energia de um CI, quando aplicadas na fase inicial do fluxo, especialmente na concepção do projeto e da arquitetura do sistema, possuam um fator de impacto superior às decisões aplicadas na fase de implementação RTL (JADCHERLA et al., 2009; PFI, 2009; XU; YONG, 2009).

Para obter mais eficiência na redução do consumo, algumas técnicas devem ser aplicadas em momentos estratégicos no fluxo de desenvolvimento do CI para minimizar os problemas gerados por sua aplicação. Tais problemas podem ser críticos, resultando em

iterações no fluxo e modificações na arquitetura do CI. Na Figura 4.1, é apresentado um diagrama expressando a taxa de impacto das decisões e uma sugestão para o momento de aplicação de algumas técnicas de acordo com as etapas de desenvolvimento (MATHUR, 2008).

Figura 4.1 - Fator de impacto das decisões sobre a redução do consumo.



Fonte: Figura adaptada (MATHUR, 2008).

Com relação à técnica de *Power Gating*, o impacto na redução do consumo de energia e o sucesso da sua aplicação, serão superiores se a técnica for aplicada no início de desenvolvimento do CI, devido à técnica ser invasiva e dependente da arquitetura do CI.

Na primeira geração de projetos, os CI possuíam poucos domínios de energia, mas os projetos recentes podem alcançar 50 domínios, produzindo numerosos modos de alimentação do CI (CHEN; LIN, 2009). Isso conduz a um crescimento exponencial no número de transições *Power-Up* e *Power-Down* que devem ser verificadas funcionalmente antes da fabricação do silício.

Dentro do fluxo de desenvolvimento, o processo de verificação das técnicas para reduzir o consumo de energia pode ser realizado no menor nível de abstração, no nível físico. Nessa situação, falhas encontradas podem levar a um reprojeto desde o projeto arquitetural, causando aumento no custo e no tempo de desenvolvimento. A probabilidade de encontrar falhas decorrentes da má aplicação de algumas técnicas é elevada, devido à

complexidade e ao nível de detalhes do chip. Diante desse fato, os engenheiros de verificação preferem iniciar o processo em níveis de abstração mais elevados, por exemplo, *Netlist*, RTL ou ESL.

Diversos trabalhos relatam a execução do processo de verificação de técnicas para reduzir o consumo no nível RTL. Segundo Yeung *et al* (YEUNG; CHOI, 2009), o processo de verificação dinâmica, baseado em simulação, de projeto contendo técnicas para reduzir o consumo, pode ser complementado com base na verificação estática no nível RTL. Em seu trabalho, os autores correlacionam quais tecnologias de verificação estática devem ser aplicadas com base nos cenários apresentados no processo de verificação dinâmica.

No nível RTL, a técnica de *Power Gating* aumenta consideravelmente a complexidade do processo de verificação. A problemática em torno da verificação da técnica é decorrente da adição de novos problemas para verificar as estratégias de isolamento, PGC e a retenção do estado (ANDERSON, 2012). Além disso, o problema de verificar a unidade de PGC pode se agravar quando o número de módulos PGFB é grande, devido ao aumento exponencial das transições de estado em função dos domínios de energia (LIU *et al.*, 2010).

No nível ESL, é mais fácil delimitar o espaço de verificação, que normalmente é difícil de ser estimado (SHIBATA *et al.*, 2010). Segundo Pinto (PINTO, 2008), iniciar o processo de verificação de técnicas para redução de consumo de energia no nível ESL deve tornar o processo mais eficiente e ágil. Diante da natureza da técnica de *Power Gating* ser *Timed*, é importante observar que durante o processo de verificação nos altos níveis de abstração, o conceito de tempo deve ser preservado. Portanto, o processo de verificação da técnica no nível ESL está restrito ao nível de abstração, com informação temporal, *Timed*. Na verificação funcional, um modelo ESL *Untimed* e que não inclui ainda a implementação da técnica de *Power Gating*, pode ser comparado funcionalmente com um modelo ESL *Timed* no qual a técnica de *Power Gating* se encontra implementada.

4.2. Tecnologias para Verificação de Chips de Baixo Consumo

No decorrer da história, as indústrias de ferramentas EDA têm fornecido diversas ferramentas para apoiar o desenvolvimento de projetos de chip de baixo consumo de energia. Mas, os fluxos de projetos convencionais possuem falhas durante a incorporação de

técnicas avançadas para reduzir o consumo de energia (SAINI; SINGH, 2007). Consequentemente, várias equipes de projeto recorriam a metodologias *ad-hoc* e inflexíveis, resultando no aumento do risco de encontrar falhas no silício, problemas de desempenho e aumento do tempo para o produto ser lançado no mercado.

Atualmente, as tecnologias mais aplicadas e amplamente difundidas para realizar a verificação de projetos de baixo consumo são baseadas na especificação de PDML (*Power Definition Markup Language*), simulação *Power-Aware*, e análise formal. Essas tecnologias são os componentes fundamentais para várias metodologias de verificação de consumo de energia (BEMBARON; KAKKAR; MUKHERJEE, 2009; CRONE; CHIDOLUE, 2007; HAZRA et al., 2010):

- PDML provêem um modo para especificar a arquitetura de alimentação do projeto independentemente da descrição no nível RTL. A especificação de PDML inclui a conectividade da alimentação, controle do comportamento da desativação e interação entre os diferentes domínios de energia.
- *Power-Aware* é a simulação da inclusão das funcionalidades do mecanismo de gerenciamento do consumo, na qual o simulador de RTL lê e interpreta o PDML de forma que o comportamento *power-up* e *power-down* possa ser modelado.
- Análise formal é um processo de natureza exaustiva, baseado na checagem de modelos, podendo assim encontrar erros, que não foram exercitados por algum conjunto de testes.

Durante o fluxo de desenvolvimento RTL-*to*-GDSII, vários projetos de circuitos integrados vêm tendo sucesso na incorporação do gerenciamento do consumo de energia utilizando técnicas como *Clock-Gating, Power-Gating, Voltage Islands, DVFS* etc. A aplicação dessas técnicas não são totalmente automatizadas e a avaliação do *tradeoff* entre as técnicas não é fácil (KAPOOR *et al.*, 2009).

Nos últimos anos, os desenvolvedores de circuitos de baixo consumo de energia, diante dos incentivos da indústria de ferramentas EDA, vêm utilizando no processo de verificação as tecnologias baseadas em PDML e simulação *Power-Aware* (GAMMIE *et al.*, 2010; PFI, 2009). Na próxima seção, serão apresentados mais detalhes sobre essas tecnologias.

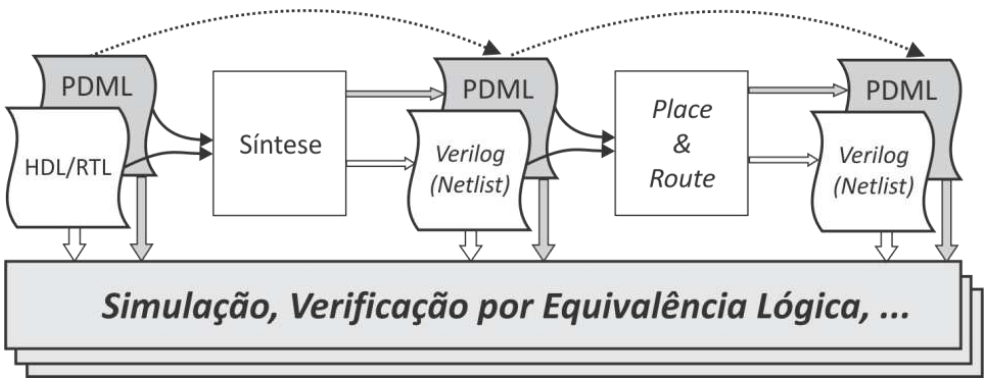
4.3. Verificação Baseada em PDML

Em 2006, um conjunto de indústrias EDA, desenvolveu uma norma que unificou múltiplos formatos proprietários em um único padrão, definindo um processo aberto e portável para especificação do consumo de energia em circuitos integrados. Assim, foi criado o UPF (UPF, 2010), posteriormente padronizado pela IEEE (IEEE-SA, 2009). Na mesma época, a empresa de ferramentas EDA CADENCE® desenvolveu uma norma semelhante à UPF, que foi denominada de CPF (SI2, 2008). Em 2007, a norma foi integrada à Si2 (*Silicon Integration Initiative*) LPC (*Low Power Coalition)* para ser gerenciada pela mesma. A Si2 é uma organização de empresas do setor de semicondutores, sistemas eletrônicos e indústria de ferramentas EDA (SI2, 2010).

UPF e CPF são linguagens de especificação aberta, que capturam do projeto todas as especificações sobre as intenções em relação ao consumo de energia, restrições e requisitos funcionais em um único formato de arquivo portável e consistente. Ambos os formatos são baseadas em TCL (*Tool Control Language*) e compartilham da mesma filosofia, mas, as sintaxes das linguagens diferem em 90% (ALLEN, 2008).

No processo de verificação baseado nos formatos CPF ou UPF, os arquivos RTL não são modificados com base nas intenções de consumo de energia. Assim, os arquivos com as especificações de requisitos funcionais do projeto ficam separados dos arquivos com a especificação de requisitos de consumo de energia. Na Figura 4.2, é apresentado o fluxo de desenvolvimento de CI utilizando PDML.

Figura 4.2 - Fluxo de desenvolvimento de projeto de baixo consumo.



Fonte: Figura baseada de (IEEE-SA, 2009).

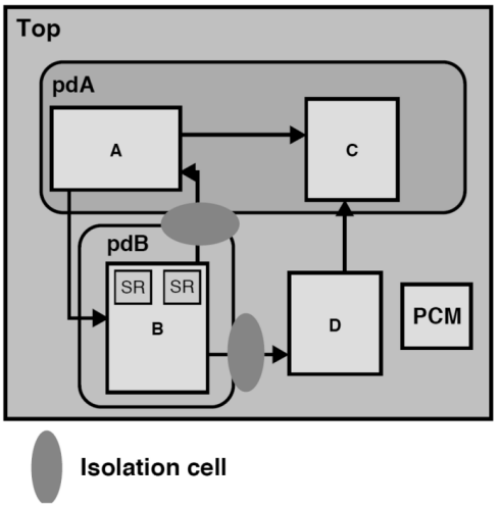
Essas características facilitam o reuso de código, pois o código RTL não precisa ser alterado, podendo ser instanciado várias vezes. Em cada caso, terá o comportamento em relação ao consumo de energia de acordo com o arquivo de especificação de requisitos correspondente. Os arquivos PDML gerados são utilizados em todas as fases do fluxo de desenvolvimento do CI.

As tecnologias de verificação citadas na subseção anterior são suportadas por ambos os formatos (IEEE-SA, 2009; SI2, 2008). As linguagens possuem bibliotecas e tecnologias específicas que são utilizadas para a síntese e implementação (LIU *et al.*, 2010). A principal diferença entre os dois formatos é que UPF não contém todos os comandos para definir os elementos de biblioteca como, por exemplo, *level-shifter* ou registradores de retenção (ALLEN, 2008). Mas, a biblioteca *Liberty library* (.lib) da Synopsys®, por exemplo, consegue capturar essas informações, cobrindo as deficiências de UPF (“Liberty Library Synopsys,” 2011).

4.3.1. Especificação dos Requisitos de Consumo de Energia com CPF

Para demonstrar como é realizada a especificação dos requisitos de consumo de energia, será utilizado um exemplo da aplicação da técnica de *Power Gating* usando CPF. Esse exemplo foi extraído do livro editado pela *Power Forward Initiative* (PFI, 2009). Na Figura 4.3, é apresentado o projeto em alto nível.

Figura 4.3 - Múltiplos Domínios e *Power Gating*.



Fonte: (PFI, 2009).

O projeto possuido dois domínios de energia específicos pdA, pdB, e o domínio de energia geral pdTop. Todas as instâncias que não são atribuídas a um dos domínios específicos são pertencentes ao domínio pdTop. No Código 4.1, é apresentado a sintaxe utilizando CPF.

Código 4.1 - Descrição de vários domínios de energia, retenção e isolamento usando CPF.

```
# Define o domínio de energia " Top"
set_design TOP

# Define o domínio de energia principal
create_power_domain –name pdTop –default

# Define o domínio de energia "A"
create_power_domain –name pdA –instances {uA uC} –shutoff_condition {!uPCM/pso[0]}

# Define o domínio de energia "B" quando PGFB está desativado
create_power_domain –name pdB –instances {uB} –shutoff_condition {!uPCM/pso[1]}

# Define a célula de isolamento, com a saída no nível lógico alto
set hiPin {uB/en1 uB/en2}
create_isolation_rule –name ir1 –from pdB –isolation_condition {uPCM/iso} \
–isolation_output high –pins $hiPin

# Define a célula de retenção do estado
set srpgList {uB/reg1 uB/reg2}
create_state_retention_rule –name sr1 –restore_edge {uPCM/restore[0]} -instances $srpgList
```

No exemplo, todos os sinais de controle são gerenciados por uma unidade PCM (*Power Control Management*). O domínio de energia pode ser desativado por sinais específicos por meio do comando ***shutoff_condition{}.*** O isolamento dos sinais e a retenção dos valores em *Power Gating* são realizados por meio dos comandos ***create_isolation_rule*** e ***create_state_retention_rule***. Em algumas situações o reuso de um IP pré-existente, pode vir associado com um arquivo CPF, que descreve os requisitos sobre a retenção do estado e isolamento. Nesse caso, não é necessário realizar uma nova descrição. O processo utilizando UPF é semelhante a esse (CROFT; BAILEY, 2007; LIU et al., 2010).

4.3.2. Verificação com Base em CPF

Após gerar os arquivos CPF, com os requisitos sobre consumo de energia do SoC descritos em RTL, o passo seguinte é verificar a sintaxe do conteúdo dos arquivos CPF. Esse

passo é realizado com tecnologia de verificação formal (PFI, 2009). O próximo passo é a verificação comportamental das funcionalidades do sistema RTL, sobreposto com o arquivo CPF, contendo a especificação dos requisitos do consumo de energia. Essa verificação é baseada em simulação RTL.

Com base no exemplo da subseção anterior, utilizando a técnica de *Power Gating,* será descrito o comportamento de um simulador com suporte a CPF (PFI, 2009) e também é válido para suporte a UPF (CRONE; CHIDOLUE, 2007). O sistema é simulado com o PGFB desligado para verificar a corretude funcional e a capacidade de se recuperar após a reativação do PGBF.

Os sinais de controle especificados no arquivo CPF para o isolamento e a retenção são gerados pelo controlador do consumo de energia. O comportamento do sistema com o PGFB desligado é acionado no simulador após os sinais de controle correspondentes serem ativados. Quando o PGFB está desligado, o simulador força todos os elementos internos do PGFB para valores desconhecidos "X". Pouco antes do desligamento, o sinal de isolamento é ativado e o simulador força todas as saídas do PGBF para valores “0”, “1”, ou alta impedância “Z” conforme especificados no arquivo CPF. Entre o isolamento e a desativação, o sinal de retenção é ativado pelo controlador de energia, fazendo com que o simulador armazene os valores atuais das variáveis especificadas no arquivo CPF.

No processo de ativação do PGFB, ocorre a sequência oposta dos eventos: a energia é restaurada, seguido pela restauração dos valores retidos nas células de retenção e a remoção do isolamento dos valores nas saídas. Um fato importante neste estágio é que não foi necessário modificar o arquivo RTL para suportar essas funções, a retenção do estado e o isolamento são realizados de forma transparente. Durante a fase de simulação do fluxo, os sinais de controle não estão conectados às unidades de domínios de energia, estas conexões serão implementadas pelas estruturas apropriadas (Seção 3.6) na fase de síntese física do projeto.

Dentre os trabalhos sobre processo de verificação utilizando a tecnologia PDML, Kapoor *et al.* (KAPOOR *et al.*, 2009) e Gammie *et al*. (GAMMIE *et al.*, 2010) fizeram uma análise geral das principais técnicas para redução de energia que são baseadas no gerenciamento da tensão, relatando as implicações sobre a verificação das técnicas.

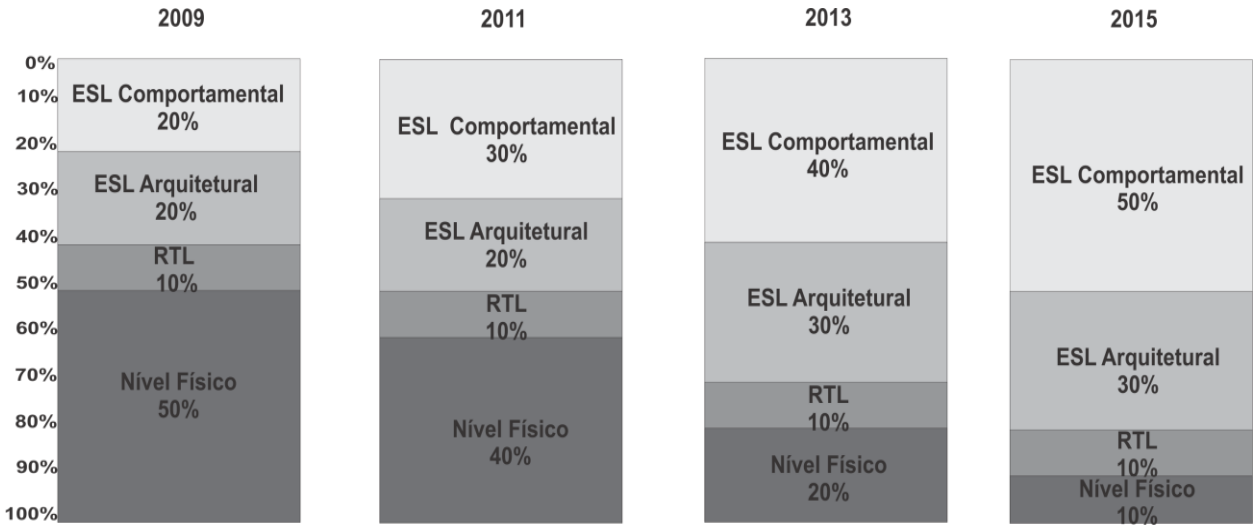
Segundo Kapoor *et al*., os formatos UPF e CPF apresentam uma solução para o problema das linguagens HDL, já que estas linguagens por si só não suportam a descrição da especificação dos requisitos do consumo de energia do projeto.

As tecnologias baseadas em simulações que oferecem suporte a UPF e CPF, limitam a verificação ao nível de abstração RTL (CADENCE, 2012; MENTOR GRAPHIC'S, 2012; PINTO, 2008; SYNOPSYS, 2012). No nível ESL, não são verificados os estados "X" (desconhecido) e "Z" (alta impedância) de sinais, pois vai contra à filosofia adotada pela própria abstração ESL, no que se refere à abstração dos detalhes da implementação do projeto. No Capítulo 5, será explicado como a ferramenta de simulação adaptada neste trabalho trata os sinais "X" e "Z" dentro do contexto do processo de retenção e isolamento da técnica de *Power Gating* para o nível ESL.

4.4. Tendência da Verificação das Técnicas para Reduzir o Consumo

Vários pesquisadores estão buscando novas formas para gerenciar o consumo de energia estática e dinâmica dos SoC, com base em características abstratas do seu funcionamento, com o objetivo de conseguir um equilíbrio entre as diversas técnicas aplicadas na redução do consumo de energia. Na Figura 4.4, é apresentada uma previsão da aplicação das decisões para reduzir o consumo de energia de acordo com o nível de abstração do projeto (ITRS, 2009).

Figura 4.4 - Previsão da participação dos níveis de abstração do projeto na redução do consumo de energia.



Fonte: Figura adaptada (ITRS, 2009).

Cada barra da Figura 4.4 está subdividida em 4 (quatro) níveis de abstração, cada nível contém a porcentagem das decisões sobre a redução do consumo de energia. É possível observar que, em um futuro próximo, as decisões tomadas em ESL serão responsáveis por 80% da redução do consumo de energia do SoC.

4.5. Trabalhos Relacionados

Nesta seção, serão tratados os trabalhos relacionados, no âmbito da verificação de circuitos integrados que contenham técnicas para reduzir o consumo de energia. O objetivo é mostrar o nível de abstração e as técnicas aplicadas no processo de verificação de cada trabalho. Não serão detalhadas as ferramentas e metodologias que foram utilizadas, e os objetivos de cada trabalho serão descritos de forma sucinta. Sendo assim, os documentos foram agrupados com relação ao nível de abstração e à técnicas utilizada.

Esses trabalhos foram encontrados nas bibliotecas digitais da IEEE Xplore®, ACM, Springerlink, portal de periódicos da CAPES, ferramenta de busca Google e Google Acadêmico e por meio de alguns autores. Nas buscas, foram utilizadas as palavras chaves (*verification, functional, low power, power gating, power gate*, *power shutoff, System-Level,* RTL, ESL e TLM) e suas combinações.

Kitahara *et al* (KITAHARA *et al.*, 2005) apresentaram em seu trabalho a aplicação da técnica de Multi-V_{th} em RTL, realizaram otimizações nas células Multi-V_{th} no nível de *layout*. Sengupta e Saleh (SENGUPTA; SALEH, 2008) aplicaram a técnica de *Voltage Islands*, diretamente no nível de *layout*. Em nenhum dos trabalhos foi informado como o processo de verificação foi executado, apenas relataram a técnica para reduzir o consumo e o nível de abstração que foi aplicado.

Nos trabalhos descritos por Jiang *et al* (JIANG; MAREK-SADOWSKA; NASSIF, 2005) e Sarkar *et al* (SARKAR; LIN; WANG, 2007), eles fizeram relatos sobre os benefícios e custos da aplicação da técnica de *Power Gating* em termos de consumo de energia, área e desempenho. Em seus experimentos, foi adotada a verificação funcional da técnica no nível *Netlist,* Jiang *et al* utilizaram a ferramenta HSPICE® da Synopsys®. Sarkar *et al* utilizaram a ferramenta SPICE e fizeram relatos sobre a complexidade e dificuldade na verificação. Eles propuseram uma metodologia para auxiliar nas tarefas de análise e verificação da técnica.

De acordo com a literatura, a complexidade relatada no processo de verificação é originada pelos detalhes introduzidos pelo nível de abstração *Netlist*.

Hillman (HILLMAN, 2005) descreveu em seu trabalho a aplicação da técnica de *Voltage Islands* no processador Xtensa para reduzir o consumo estático. Para verificar que o processador atendesse os requisitos de desempenho após aplicação da técnica, ele realizou o processo de verificação funcional no nível de *Netlist*. Não foram relatadas as implicações no processo de verificação e nem o ferramental utilizado.

Varma *et al* (VARMA *et al.*, 2008) descreveram em seu trabalho uma arquitetura de baixo consumo de energia para Handset LSI Medity^{TM} M2. Para reduzir o consumo de energia, utilizou as técnicas de Multi-V_{th}, DVSF e *Clock Gating.* Varma relatou que o processo de verificação ocorreu no nível *Netlist*. Nesse trabalho, demonstrou-se a combinação das técnicas para aumentar o ganho na economia de energia e criticou-se a falta de amadurecimento das ferramentas EDA na aplicação das técnicas para reduzir o consumo de energia.

Upasani *et al* (UPASANI *et al.*, 2009) descreveram uma estratégia para reduzir os conflitos de concorrência entre as técnicas de *Power Gating* e *Clock Gating* aplicadas no mesmo projeto. No estudo de caso apresentado, ele realizou a verificação funcional das técnicas no nível *Netlist* utilizando a ferramenta Modelsim da Mentor Graphics. Em ambos os trabalhos, não existem relatos sobre a complexidade da verificação das técnicas.

Dos trabalhos descritos até o momento, a maioria dos processos de verificação foi executada no nível de abstração *Netlist*, e existem poucos relatos sobre as implicações sobre a verificação das técnicas de baixo consumo. Nos próximos trabalhos, há uma tendência em iniciar o processo em níveis superiores de abstração utilizando as tecnologias predominantes.

Hsieh *et al* (HSIEH et al., 2008) descreveram em seu trabalho a aplicação das técnicas DVFS, *Power Gating* e Multi-V_{th} em um DSP (*Digital Signal Processor*). Joseph *et al* (JOSEPH; S.; SANKARAPANDIAMMAL, 2009) aplicaram a técnica de *Clock Gating* em um processador RISC de 5 estágios de paralelismo (*pipelining*). Hsieh *et al* e Joseph *et al* utilizaram o fluxo de desenvolvimento baseado em CPF e os processos de verificação funcional das técnicas foram

iniciados no nível RTL. Apenas Joseph *et al* especificaram a ferramenta utilizada na verificação, no caso, o simulador Modelsim da Mentor.

Eisner *et al* (EISNER; NAHIR; YORAV, 2009) relataram em seu trabalho o problema com o custo computacional para realizar a verificação funcional da técnica de *Power Gating* e apresentaram uma alternativa com base em métodos formais por equivalência para verificar a aplicação da técnica em RTL. A verificação formal foi aplicada exclusivamente ao PGFB do projeto, utilizando a ferramenta SixthSense da IBM®, em que eles compararam com o mesmo bloco funcional, sem conter a técnica de *Power Gating* implementada. Para o restante do projeto foi aplicada a verificação funcional. Os autores patentearam a técnica utilizada (EISNER; ZICHRON YAACOV; YORAV, 2011). Nesse trabalho não fica clara a complexidade da modelagem matemática das funcionalidades do projeto.

Viswanath *et al* (VISWANATH; VASUDEVAN; ABRAHAM, 2009) descrevem em seu trabalho uma ferramenta, por eles desenvolvida, que aplica automaticamente a técnica de *Clock Gating* ao projeto, por meio da reescrita de trechos do código Verilog RTL. Eles utilizaram um decodificador Viterbi no estudo de caso. Para garantir a corretude do projeto, ele realizou a verificação utilizando ferramentas baseadas em métodos formais por equivalência no nível RTL. Um ponto a se questionar sobre esse método de verificação, é a escalabilidade para aplicação em projeto de maior complexidade.

Chen *et al* (CHEN; LIN, 2009) apresentaram uma metodologia de planejamento das interfaces entre os domínios de energia, para *Power Gating* e DVFS, e propuseram um molde para as interfaces de controle dos circuitos de gerenciamento das técnicas. No estudo de caso, os autores utilizaram uma ferramenta, desenvolvida por eles, para verificar a integridade dos arquivos CPF ou UPF. A metodologia apresentada auxilia na solução de um dos problemas da aplicação das técnicas, o número excessivo de estados dos domínios de energia, que são introduzidas pelas técnicas ao sistema.

Kuwahara *et al* (KUWAHARA, 2010) relataram em seu trabalho a utilização das técnicas *Power Gating,* DFS *(Dynamic Frequency Selection), Clock Gating* e *Multi-*V_{th} em um SoC da Toshiba®, que consistia em uma interface de rede *wireless.* Uma característica importante deste trabalho são a modelagem e simulação do sistema no nível ESL, utilizando

Matlab® e a linguagem C para definir a arquitetura e os algoritmos do sistema. As técnicas de redução de consumo só foram verificadas no nível RTL utilizando UPF.

Gammie *et al* (GAMMIE *et al.*, 2010), fizeram uma análise sobre o desempenho das técnicas de gerenciamento de consumo de energia aplicado na *Texas Instruments* para dispositivos móveis. Os autores deram ênfase às técnicas *Power Gating,* DVFS *e* AVS *(Adaptive Voltage Scaling)* por serem as principais técnicas aplicadas em processadores multimídia da Texas. Com base em três processadores, descreveram como as técnicas são aplicadas utilizando a tecnologia SmartReflex e verificadas funcionalmente utilizando UPF.

Pasha *et al* (PASHA; DERRIEN; SENTIEYS, 2010), descreveram em seu trabalho a utilização da técnica de *Power Gating* aplicada aos nós de uma rede de sensores *Wireless.* Cada nó é baseado em um microcontrolador com tarefas específicas. A técnica é aplicada em cada microcontrolador no nível RTL, porém não relataram como foi realizado o processo de verificação. Kim *et al* (KIM; KIM; BAEK, 2010) descrevem o fluxo de desenvolvimento de SoC baseado em uma plataforma unificada de verificação. No estudo de caso, aplicaram a técnica de *Clock Gating* para reduzir o consumo de energia de um transmissor Bluetooth v2.0+EDR. Eles realizaram o processo de verificação funcional do SoC em RTL, mas não apresentaram detalhes sobre as ferramentas que dão o suporte a sua plataforma de verificação.

Kissler *et al* (KISSLER *et al.*, 2011) em seu trabalho apresentaram uma técnica para gerenciar diversos domínios de energia. No estudo de caso, aplicaram a técnica de *Power Gating* em um processador matricial 6x6. A ativação e desativação dos processadores internos ocorriam de acordo com a necessidade de processamento. A verificação do sistema se iniciou no nível RTL utilizando CPF.

Lakshmi *et al* (LAKSHMI; VAYA; VENKATARAMANAN, 2011) explicam em seu trabalho como aplicar a técnica de *Power Gating* utilizando CPF. Como exemplo, eles aplicaram a técnica no SoC de modulação de voz. Apresentaram a sintaxe básica do arquivo CPF para *Power Gating*, e realizaram a verificação funcional no nível RTL utilizando as ferramentas da Cadence®. No trabalho de Hoang *et al* (HOANG *et al.*, 2011), eles aplicaram a técnica de *Power Gating* em um processador FlexCore utilizando CPF. Os autores não relatam as

ferramentas que foram utilizadas e em qual nível foi iniciada a verificação funcional, por está utilizando CPF. Acredita-se que a verificação foi iniciada no nível RTL.

Kumar e Vasudevan (KUMAR; VASUDEVAN, 2012) descreveram em seu trabalho como realizar a verificação de *Power Gating* utilizando formalismo, baseado em estáticas, no nível RTL. A técnica foi aplicada em um processador OpenSPARC e utilizaram a ferramenta pCTL (*probabilistic Computational Tree Logic*) para formalizar o sistema e a ferramenta Ymer para verificar. Relataram que o tempo de simulação da verificação foi menos de um segundo, mas não fizeram relatos sobre o custo da formalização do projeto.

No trabalho de Lebreton e Vivet (LEBRETON; VIVET, 2008) foi modelada a arquitetura de um NoC com aplicação da técnica de DVFS e estimado o consumo no nível ESL. Para auxiliar a modelagem TLM-Timed e a verificação do sistema, eles desenvolveram um *framework* baseado em SystemC. Os autores não deixam claro se o núcleo do SystemC foi modificado para trabalhar em conjunto com *framework* desenvolvido. Este trabalho reforça a tendência de modelar a arquitetura e aplicar as técnicas para reduzir o consumo no nível ESL.

No trabalho de Court e Kelly (COURT; KELLY, 2011) eles aplicaram a técnica de *Power Gating* em uma simulação do processador Alpha utilizando o simulador SimpleScalar, um simulador de arquiteturas de processadores, também chamado simulador de conjunto de instruções. Para simular a técnica, fizeram modificações no SimpleScalar. Os autores não fizeram relatos se o sistema contendo o processador foi verificado. O nível de arquitetura de um processador está acima do nível RTL, mas diferente de ESL, é um nível de abstração com domínio de aplicação específico para processadores.

Nos últimos trabalhos apresentados, pode-se observar a utilização das tecnologias CPF e UPF no processo de verificação das técnicas para reduzir o consumo de energia. Existem relatos que iniciaram o processo de verificação no nível ESL, utilizando a verificação estática ou alguma tecnologia para casos específicos. Mas, nenhum desses trabalhos apresenta alguma solução para a aplicação e verificação funcional de projetos diversos contendo a técnica de *Power Gating,* no nível ESL.

Na Tabela 4.1 são apresentadas as principais características dos trabalhos apresentados, abrangendo o nível de abstração no qual foi executado o processo de verificação, a tecnologia utilizada e as técnicas aplicadas para redução de consumo de energia que foram aplicadas.

Tabela 4.1 - Principais características dos trabalhos relacionados

	Trabalho	Abstração Da Verificação	Tipo de Verificação	Tecnologia de Verificação	Técnica(s) Aplicada(s)
1.	(KITAHARA *et al.*, 2005)	*Layout*	-	-	*Multi-V_{th}*
2.	(SENGUPTA; SALEH, 2008)	*Layout*	-	-	*Voltage Islands*
3.	(JIANG; MAREK-SADOWSKA; NASSIF, 2005)	*Netlist*	Dinâmica	-	*Power Gating*
4.	(SARKAR; LIN; WANG, 2007)	*Netlist*	Dinâmica	-	*Power Gating*
5.	(HILLMAN, 2005)	*Netlist*	Dinâmica	-	*Voltage Islands*
6.	(VARMA *et al.*, 2008)	*Netlist*	-	-	*Multi-V_{th} , DVFS, Clock Gating*
7.	(UPASANI *et al.*, 2009)	*Netlist*	Dinâmica	-	*Power Gating, Clock Gating*
8.	(HSIEH et al., 2008)	RTL	Dinâmica	CPF	*Multi-V_{th} , DVFS e Power Gating*
9.	(JOSEPH; S.; SANKARAPANDIAMMAL, 2009)	RTL	Dinâmica	CPF	*Clock Gating*
10.	(EISNER; NAHIR; YORAV, 2009)	RTL	Estática	-	*Power Gating*
11.	(VISWANATH; VASUDEVAN; ABRAHAM, 2009)	RTL	Estática	-	*Clock Gating*

Continua na próxima página...

Tabela 4.1 - Continuação.

	Trabalho	Abstração Da Verificação	Tipo de Verificação	Tecnologia de Verificação	Técnica(s) Aplicada(s)
12.	(CHEN; LIN, 2009)	RTL	Estática	CPF	*Power Gating e* DVFS
13.	(KUWAHARA, 2010)	RTL	Dinâmica	UPF	*Power Gating,* DVFS*, Clock Gating e Multi-*V_{th}
14.	(GAMMIE *et al.*, 2010)	RTL	Dinâmica	UPF	DVFS*, Power Gating e* AVS
15.	(PASHA; DERRIEN; SENTIEYS, 2010)	RTL	-	-	*Power Gating*
16.	(KIM; KIM; BAEK, 2010)	RTL	Dinâmica	-	*Clock Gating*
17.	(KISSLER *et al.*, 2011)	RTL	Dinâmica	CPF	*Power Gating*
18.	(LAKSHMI; VAYA; VENKATARAMANAN, 2011)	RTL	Dinâmica	CPF	*Power Gating*
19.	(HOANG *et al.*, 2011)	RTL	Dinâmica	CPF	*Power Gating*
20.	(KUMAR; VASUDEVAN, 2012)	RTL	Estática	-	*Power Gating*
21.	(LEBRETON; VIVET, 2008)	ESL	Dinâmica	SC/TLM	DVFS
22.	(COURT; KELLY, 2011)	ESL	-	-	*Power Gating*

4.6. Considerações Finais do Capítulo

Neste capítulo, foi apresentado o estado da arte ano âmbito da verificação de CI contendo as técnicas para redução de consumo de energia. Foram levantadas questões metodológicas e técnicas envolvendo o processo de verificação de *Power Gating*. Para alcançar melhor eficiência mantendo o desempenho especificado e reduzir os problemas no processo de verificação, deve ser aplicado inicialmente no maior nível de abstração possível do projeto. Foi apresentado o esforço das indústrias de ferramentas EDA em padronizar as tecnologias de verificação baseadas em PDML e a tendência do fator de impacto sobre as decisões em relação às restrições de consumo de energia.

De acordo com os trabalhos relacionados, é possível observar que vários projetos têm aplicado a técnica de *Power Gating* em conjunto com outras técnicas, e que nos últimos anos, tais técnicas, vêm sendo verificadas funcionalmente por meio do uso de CPF ou UPF no nível RTL. Mas, apesar dos benefícios de CPF e UPF, elas limitam a verificação funcional ao nível RTL, que para os atuais projetos de CI, o processo de verificação representa um desafio para os engenheiros.

Diante do aumento da complexidade que *Power Gating* introduz ao processo de verificação, e das limitações tecnológicas, que restringem a verificação até o nível RTL, a possibilidade de aplicar e verificar a técnica de *Power Gating* no nível ESL se mostra relevante.

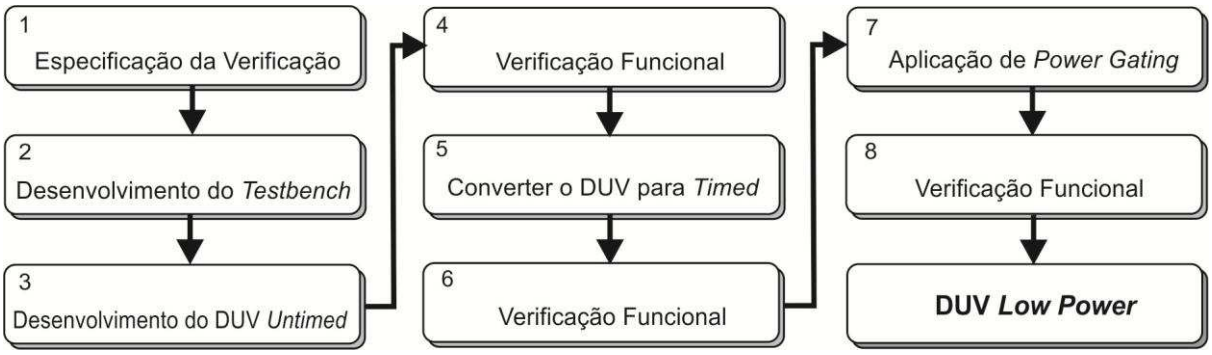
Capítulo 5 - Abordagem de Verificação Funcional

Neste capítulo, será apresentado uma abordagem metodológica de como aplicar e simular a técnica de *Power Gating.* Será demonstrado um simulador que é capaz de simular o comportamento da técnica de *Power Gating* aplicado em um circuito digital descrito em SystemC nos níveis RTL e/ou ESL, demonstrando as etapas do desenvolvimento do simulador de acordo com os níveis de abstração e seu princípio de funcionamento.

5.1. Uma Abordagem para Verificação Funcional da Técnica de *Power Gating.*

Para auxiliar no processo da verificação funcional de projeto de circuito integrado, contendo a técnica de *Power Gating*, será demonstrado uma abordagem metodológica, baseada na metodologia de verificação VeriSC. Essa abordagem é dividida em 8 etapas. Os engenheiros responsáveis pela execução de cada etapa, é baseada no que foi descrito na Seção 2.1, sobre as etapas do processo de verificação. Na Figura 5.1, são apresentadas as etapas para aplicar e verificar funcionalmente a técnica de *Power Gating* no nível ESL.

Figura 5.1 - Fluxo para verificação funcional de *Power Gating* no nível ESL.



Descrição das ações que devem ser executadas em cada etapa:

1. Especificação da Verificação - Descreva, em forma de texto, não ambíguo, todas as funcionalidades que devem ser verificadas no SoC. Esse texto guiará a execução das demais etapas;
2. Implementação do *testbench* - Com base na metodologia VeriSC, implemente o ambiente no qual o DUV será inserido. Esse ambiente deve ser descrito no nível ESL, e deve enviar estímulos ao DUV e receber as respostas para serem comparadas, com as respostas do modelo de referência. Os estímulos devem ser baseados na especificação da verificação, e o modelo de referência deve ser *Untimed*;
3. Implementação do DUV - Particione o DUV em submódulos funcionais, com base no modelo arquitetural do projeto. Os submódulos devem ser *Untimed* e se comunicarem por meio de transações, que percorrem um canal abstrato, por exemplo, FIFO. Nessa etapa, o objetivo é na funcionalidade de cada submódulo do DUV;
4. Verificação - Com base no *testbench* e no DUV, inicie a simulação e verifique se todas as funcionalidade do DUV, que foram especificadas na primeira etapa, são preservadas na implementação;
5. Converter o DUV para *Timed* - Após a verificação das funcionalidades, adicione as informações temporais aos submódulos do DUV. As informações em relação aos tempos, estão descritas no documento que contêm as especificações do sistema;
6. Verificação - Nesta etapa, execute novamente a simulação e verifique se todas as funcionalidades permanecem preservadas após a adição das informações temporais. Caso seja detectado algum erro durante a verificação, retorne à etapa anterior e analise os tempos que foram aplicados aos submódulos. Ao final desta etapa, o DUV está subdividido funcionalmente e modelado no nível de abstração adequado para receber a aplicação da técnica de *Power Gating;*
7. Aplicação de *Power Gating* - Selecione o submódulo que será o PGFB, substitua as SC_FIFO que conecta o PGFB, aos demais submódulos, por SC_FIFO_LP. Adicione

o submódulo PGC ao DUV. Conecte o PGC ao(s) submódulo(s) que contenham as informações necessárias para as tomadas de decisões de desativação e ativação do PGFB. O tempo de atraso, que é passado na função de ativação, é baseado na especificação dos transistores de "sono", que serão utilizados na fase de *layout* do projeto;

8. Verificação - Nesta etapa, execute novamente a simulação e verifique se todas as funcionalidades permanecem preservadas após a adição da técnica de *Power Gating*. Caso não sejam detectados erros durante a verificação, a técnica de *Power Gating* foi aplicada com sucesso ao projeto. Mas, caso seja detectado algum erro, durante a verificação, retorne à etapa anterior e analise a semântica da PGC. Com base nas informações do arquivo de *log*, observe se o PGFB está sendo adequadamente desativado e ativado.

É importante observar, que de acordo com as etapas da abordagem, a técnica de *Power Gating* é aplicada ao DUV, na 7ª etapa. Portanto, o DUV é *Timed* e suas funcionalidades foram verificadas com sucesso. Com base nessas informações, durante a 8ª etapa, caso o *testbench* detecte alguma falha, é provável que tenha sido originada pela desativação errônea do PGFB. Em outras palavras, a falha é devido ao fato do PGFB estar desligado ao invés de estar ligado. Essa informação é util para a equipe de depuração. Pois, restringe a área de busca da falha, ao submódulo PGC.

5.2. Simulação de *Power Gating* em SystemC RTL

Um módulo de SystemC é o menor contêiner de funcionalidades, contendo o estado, comportamento e estrutura, para ser conectado em uma hierárquia. Cada módulo pode ter um ou mais métodos, em que cada método é um processo. O simulador SystemC é baseado em um núcleo multitarefa não-preemptivo, ou seja, que não pode forçar um processo em execução a devolver o controle ao núcleo. O núcleo do simulador escalona os processos e chama cada um, conforme a necessidade.

O simulador SystemC possui dois tipos básicos de processos, SC_METHOD e SC_THREAD. Ambos são funções membro da classe SC_MODULE e somente o núcleo do simulador pode invocar cada um destes processos. Em um processo SC_METHOD, o tempo

não avança entre a invocação e retorno da função, este processo é não bloqueante e executado repetidamente. O SC_THREAD é um processo que é invocado apenas uma vez e possui a característica de ser bloqueante, o bloqueio é executado por meio do método ***wait()***, permitindo que outros processos sejam executados pelo núcleo.

Para realizar a verificação funcional de *Power Gating,* utilizando um simulador funcional SystemC, foi selecionada uma abordagem semelhante àquela desenvolvida para simular a reconfiguração dinâmica e parcial (BRITO; MELCHER; ROSAS, 2006; BRITO et al., 2007), uma abordagem *bottom-up*, que adiciona funções para ativar e desativar módulos pelo programador, simulando em tempo real a reconfiguração.

5.2.1. Simulador SystemC

SystemC não permite a desativação dos módulos durante a simulação, porém esta ferramenta é de código fonte aberto e pode-se alterar seu código fonte para esse fim. Na estratégia utilizada, adicionou-se duas novas funções especiais no núcleo do simulador SystemC para ligar (***sc_lp_turn_on***) e desligar (***sc_lp_turn_off***) módulos durante a simulação. A rotina ***sc_lp_add_constraint*** foi criada para armazenar os atributos dos módulos como, por exemplo, o tempo necessário para sua ativação. Após essas modificações, o simulador foi denominado de **SC-LP V1** (*SystemC - Low Power* Versão 1).

A retenção do estado interno do PGFB, nesta nova versão do simulador é total, ou seja, os valores de todas as variáveis do PGFB são retidos. O mecanismo de isolamento da saída do módulo PGFB mantém também o valor presente antes da desativação e as características da retenção e do isolamento são proporcionadas devido à persistência das variáveis no SC_MODULE. Na Tabela 5.1, estão contidas as declarações das funções no arquivo *sc_simcontext.h* do núcleo do SystemC.

Tabela 5.1 - Declaração das funções.

Nome das funções
extern void sc_lp_turn_on(std::string module_name);
extern void sc_lp_turn_off(std::string module_name);
extern void sc_lp_add_constraint(std::string module_name, sc_time wakedelay);

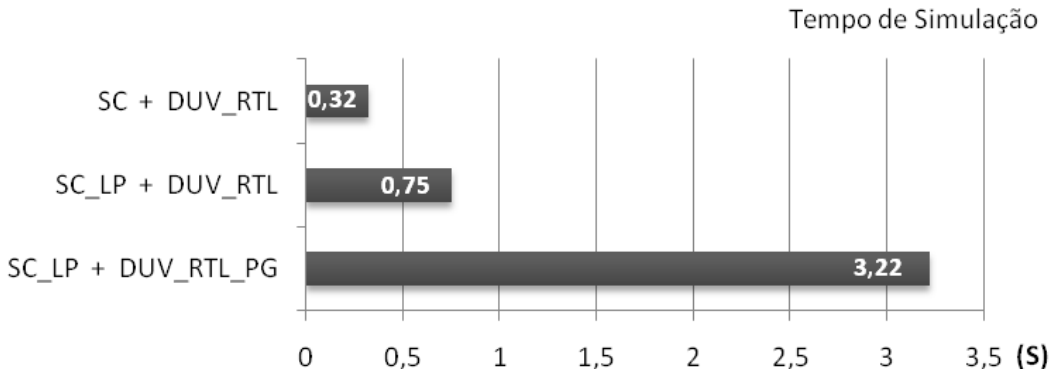
Uma lista encadeada foi usada para armazenar os nomes dos módulos, SC_METHOD que não devem ser executados. A rotina ***sc_lp_turn_off*** acrescenta o nome do módulo na lista, enquanto ***sc_lp_turn_on*** remove o módulo da lista, permitindo a sua execução. Outra lista é mantida para guardar as restrições do módulo (atraso para ativação), que será necessária quando a função ***sc_lp_add_constraint*** é chamada. Nesse caso, as restrições são adicionadas à lista e não podem ser removidas, apenas substituídas. A palavra-chave ***extern*** na declaração das funções, indica que a rotina pode ser chamada fora da classe *sc_context.h*. Em outras palavras, essa função pode ser chamada pelo código do usuário em simulações regulares.

O código RTL em SystemC é modelado utilizando SC_METHOD. O funcionamento do simulador é baseado nas informações contidas nas listas encadeadas. No núcleo do simulador, antes de executar um módulo do projeto, é realizado uma consulta à lista encadeada para saber se o nome do módulo está presente na lista, caso esteja, não será executado. Se o nome do módulo não constar na lista, será verificado se existe alguma restrição de tempo, atraso para ativação. Caso não exista, o módulo será executado.

5.2.2. Otimização do Simulador SystemC-LP para RTL

No trabalho de (BRITO; MELCHER; ROSAS, 2006; BRITO et al., 2007), foram apresentadas informações sobre a perda de desempenho do simulador. Diante desta informação, foram realizados alguns testes iniciais que confirmaram a perda de desempenho. Na Figura 5.2, é apresentado um gráfico com o desempenho dos simuladores.

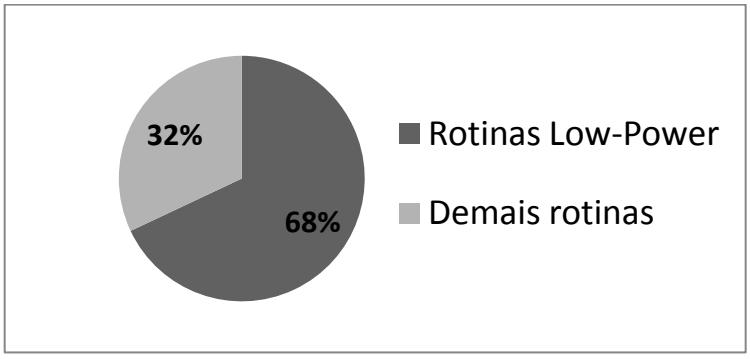
Figura 5.2 - Desempenho dos simuladores.



Na Figura 5.2, são apresentados os tempos de simulação, em que o mesmo projeto RTL, foi simulado utilizando o simulador SystemC original e o modificado. É possível observar, que o tempo de simulação utilizando SystemC-LP foi 135% superior a SystemC original. Dessa forma, considerando a simulação utilizando RTL com a técnica *Power Gating*, o desempenho do simulador foi bastante afetado, pois teve um aumento no tempo de simulação de aproximadamente 906%.

Diante dessas informações, o simulador SystemC-LP passou por um processo de otimização realizado com base no perfilamento do simulador em execução usando a ferramenta GProf (GPROF, 2010). O perfilamento do *software* permite a identificação de trechos de programas que são mais executados dentro do código fonte. Assim, é possível obter um perfil com a distribuição de tempo em cada método/rotina do programa. Além disso, é possível quantificar o número de chamadas a esses métodos/rotinas (GPROF, 2010; PERON, 2007). Na Figura 5.3, é apresentado um gráfico com o custo computacional das rotinas que são executadas durante a simulação, é possível observar que as rotinas *Low Power* representam 68% do custo computacional.

Figura 5.3 - Custo computacional do SC-LP_V1.



Os dados do perfilamento do simulador demonstraram um número excessivo de acessos às listas encadeadas que foram adicionadas ao núcleo do simulador SystemC. Diante dessas informações, foram realizadas análises assintóticas e semânticas das estruturas de dados usadas para implementar o núcleo do simulador. A análise assintótica é um método para descrever o comportamento do crescimento do tempo de execução do algoritmo quando aplicado a um volume muito grande de dados de entrada (CORMEN *et al.*, 2002).

5.2.3. Modificações do Núcleo SystemC-LP

Com base nos resultados do perfilamento, foi possível destacar as seguintes necessidades do simulador para suportar a técnica de *Power Gating*: (1) é necessário uma estrutura de dados para armazenar as informações sobre os módulos que estão desligados e do tempo de atraso necessário para retomar o funcionamento após a sua reativação e (2) a estrutura de dados deve fornecer acesso às informações em intervalo de tempo curto e constante.

Diante dos requisitos citados, a modificação do núcleo do simulador foi re-implementado utilizando a estrutura de dados **hash_map** para substituir as listas encadeadas. Cada elemento do **hash_map** representa um módulo, SC_METHOD, do projeto e é composto por uma estrutura de dados contendo duas variáveis (Booleana e SC_TIME). A variável booleana é responsável por identificar se o módulo está ligado ou não. A variável SC_TIME é responsável por armazenar o tempo de atraso necessário para reativar o módulo.

Os elementos do **hash_map** são acessados a partir de uma chave, consistindo do nome do módulo. As assinaturas das funções foram alteradas, ***sc_lp_add_constraint*** foi retirada e a sua função foi adicionada à rotina ***sc_lp_turn_on***. Na Tabela 5.2 a, são apresentadas as assinaturas das funções no arquivo ***sc_simcontext.h*** do núcleo do simulador, que foi denominado de SC-LP V2 (*SystemC - Low Power* Versão 2).

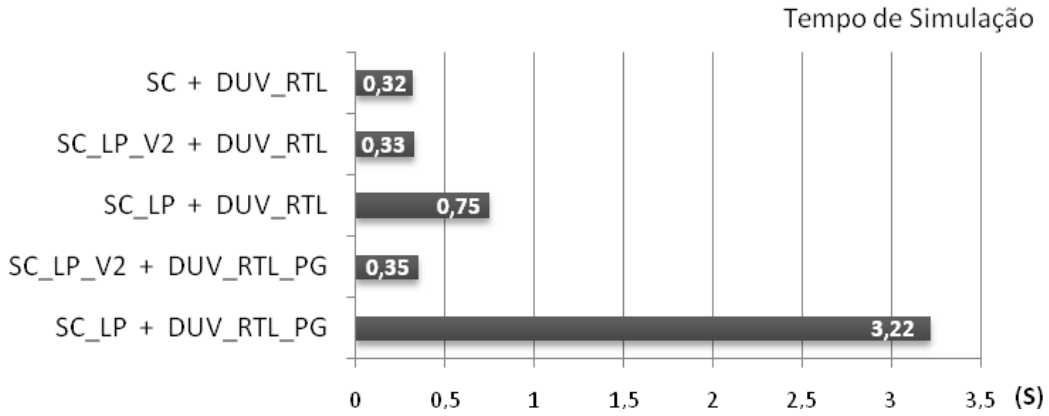
Tabela 5.2 - Declaração das novas funções.

Nome das funções
extern void sc_lp_turn_on_method (const char module_name, sc_time wakedelay);*
extern void sc_lp_turn_off_method (const char module_name);*

Após as modificações, a melhoria no desempenho do simulador foi expressiva. Na Figura 5.4, é apresentado um gráfico com a comparação dos tempos médios de simulação. Pode-se observar que as simulações do projeto DUV_RTL, utilizando o simulador SC-LP-V2 (*SystemC - Low Power V2)*, tem-se um aumento de 3,1% no tempo de simulação e as

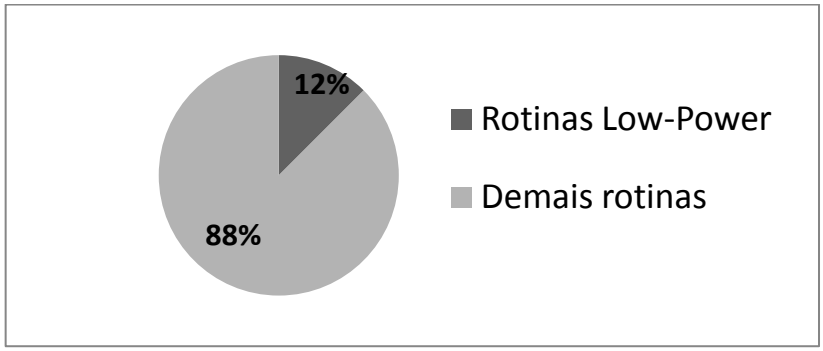
simulações do projeto com *Power Gating* DUV_RTL_PG proporcionaram um aumento de 9,4%, ambas em comparação à versão original do simulador OSCI SystemC 2.2.0.

Figura 5.4. Desempenho dos simuladores após otimização.



Comparando os dois simuladores SystemC-LP, os ganhos foram significativos. O simulador SC-LP-V2 conseguiu reduzir o tempo de simulação em 56% na execução do projeto sem *Power Gating* e 89% simulando com *Power Gating,* ao serem comparadas com o simulador SC_LP-V1. Esses valores foram obtidos com base no tempo médio de 60 simulações. Na Figura 5.5 , é apresentado um gráfico com o custo computacional das rotinas que são executadas durante a simulação utilizando o simulador SC-LP_V2, é possível observar que as rotinas *Low Power* só representam 12% do custo computacional.

Figura 5.5 - Custo computacional do SC-LP_V2



O ganho de desempenho foi alcançado a partir da eliminação dos custos da adição e remoção dos elementos das listas encadeadas e do aumento da velocidade de acesso às informações por meio do uso das funções *hash* e estrutura de dados *map*.

5.2.4. Retenção e Isolamento

Conforme exposto no Capítulo 3, o comportamento desejável do simulador em relação à retenção dos valores das variáveis internas do PGFB durante a desativação poderia ocorrer de 2 formas: retenção total ou parcial. Para a célula de isolamento dos sinais, é desejável que sua saída possa ir para algum dos seguintes estados: (i) a saída mantém o valor de antes da desativação; (ii) a saída é igual ao estado lógico "0", "1", "Z" ou "X". Portanto, essas características, da retenção e do isolamento, foram adicionadas ao simulador.

O comportamento padrão do simulador em relação à retenção consiste em armazenar os valores das variáveis, realizando a retenção total. O padrão do mecanismo de isolamento da saída consiste em manter o valor presente antes da desativação. Para proporcionar a retenção parcial e poder alterar a saída da célula de isolamento, foi desenvolvida uma biblioteca chamada de PGLIB (*Power Gating - Library*), que deve ser adicionada ao código fonte do projeto. A Tabela 5.3 contém algumas declarações das funções dessa biblioteca.

Tabela 5.3 - Declaração das funções da PGLIB.

Nome das funções
extern void pglib_int(const char name, void * var, int value);*
extern void pglib_int_set(const char name);*
extern void pglib _sc_lv(const char name, void * var, sc_lv<64> value);*
extern void pglib_sc_lv_set(const char name);*

O princípio de funcionamento da biblioteca é simples, para cada tipo de dado são criados dois métodos, uma com a função de declarar a variável que terá seu valor alterado, e outra com a função de realizar a alteração do valor.

Por exemplo, o método ***pglib_int,*** tem a função de declarar a variável que terá seu estado alterado e os parâmetros passados nesse método são: nome para ser associado à

variável, um apontador para a variável que será alterada e o valor que será atribuído à variável no momento da desativação. O apontador da variável e o valor, são armazenador em duas estruturas de dados **hash_map** e o nome que é passado é utilizado como chave para acessar esses valores.

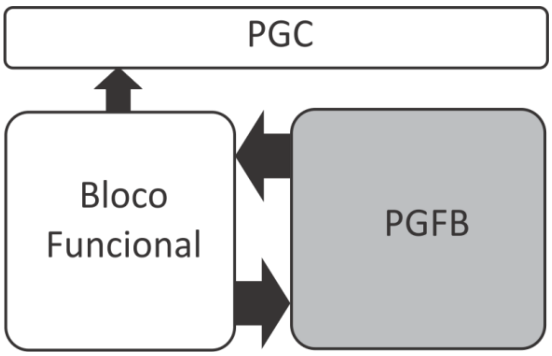
O método ***pglib_int_set*** tem a função de atribuir o valor predeterminado à variável. O parâmetro "nome" do método é utilizado para localizar a variável dentro da estrutura **hash_map** e alterar seu valor. Devido à estrutura do **hash_map** não permitir o armazenamento de tipos de variáveis distintas, não foi possível sobrecarregar a função. Mas, é permitido estender a biblioteca PGLIB para suportar tipo de variável composta. Na próxima subseção, será demonstrado um exemplo, por meio de código SystemC RTL, de como utilizar essa biblioteca.

5.2.5. Síntaxe da Aplicação da Técnica de *Power Gating* e PGLIB

Será demonstrado como aplicar a técnica *Power Gating* e a biblioteca PGLIB em um circuito digital. Para alcançar esse objetivo, são necessárias as seguintes premissas: (i) O projeto está subdividido em blocos funcionais; (ii) determinar o bloco que é passível de ser desativado, em intervalos de tempo, durante o funcionamento do sistema; (iii) adicionar ao sistema o PGC.

Será utilizado um exemplo, de um sistema fictício, composto por 2 módulos funcionais e o módulo PGC, para explicar a aplicação da técnica. A Figura 5.6 contém o diagrama do sistema, em que o módulo PGC recebe informações do bloco funcional, que serão utilizadas para determinar o momento de desativar e reativar o PGFB.

Figura 5.6 - Diagrama em bloco de um sistema fictício.



A seguir, serão apresentados, no Código 5.1, as estruturas dos blocos, descritos em SystemC RTL, que descrevem em termos gerais a estrutura em linhas de código do arquivo sistema.cpp.

Código 5.1 - Estrutura básica do sistema.cpp.

```
1.      #include "systemc.h"  // Adição da biblioteca SystemC
2.      #include "pglib.h"  // Adição da biblioteca PGLIB
3.
4.      #include "bloco_funcional.h"
5.      #include "pgfb.h"
6.      #include "pgc.h"
7.
8.      int sc_main (int argc, char *argv[])
9.      {
10.       // Declaração dos sinais de entrada e saída
11.
12.       // Instanciação dos módulos
13.       pgc  pgc_i("pgc_i");
14.       pgfb pgfb_i("pgfb_i");
15.       bloco_funcional  bf_i("bf_i");
16.
17.       // Conexões dos módulos
18.
19.       pgpr_sc_lv (  "pgfb_i.port_out", &pgfb_i.port_out , "0000" ); //Registro da variável
20.       pgpr_int ( "pgfb_i.cont" ,  &pgfb_i.cont , 0x0 ); //Registro da variável
21.
22.       sc_start();
23.       return 0;
24.     };
```

Nas linhas 1 e 2, são adicionadas as bibliotecas SystemC e PGLIB. As linhas 4, 5 e 6 adicionam os cabeçalhos dos módulos Bloco_Funcional, PGFB e PGC. Logo após, são declarados os sinais de entrada e saída dos módulos. Nas linhas 13, 14 e 15, são instanciados os módulos. Na linha 19, é chamado o método da PGLIB, ***pgpr_sc_lv,*** que registra a variável, do tipo sc_lv (*logic vector)* de SystemC, que terá seu valor alterado após o evento de desativação. Neste caso, são passados os seguintes parâmetros: ***"pgfb_i.port_out",*** uma *string* utilizada para acessar a variável por meio de PGLIB, ***&pgfb_i.port_out,*** o endereço da variável e "**0000**" o valor que será atribuído à variável após o evento de desativação de PGFB.

Na linha 20, é chamado o método ***pglib_int,*** que é semelhante a ***pgpr_sc_lv***, a diferença é que ele vai manipular uma variável do tipo inteira. Quando ocorrer o evento de desativação de PGFB, será atribuído à variável o valor "0x0".

O código 5.2 descreve, em termos gerais, a estrutura em linhas de código SystemC RTL do módulo "Bloco Funcional".

Código 5.2 - Estrutura do Bloco Funcional.

```
1.    SC_MODULE(Bloco_Funcional) {  // Declaração do módulo.
2.       // Declaração de variáveis.
3.       // Declaração das interfaces de entrada e saída.
4.       sc_out<bool > port_A;  // Interface de comunicação com PGC.
5.       sc_in  <sc_lv<4> > port_B;  // Interface de comunicação com PGFB.
6.       sc_out<sc_lv<4> > port_C;  // Interface de comunicação com PGFB.
7.
8.    SC_CTOR(Bloco_Funcional) {  SC_METHOD(BF);  sensitive_pos << clk; } // construtor
9.
10.   void BF(){
11.       if (reset) {
12.          // Atribuições as variáveis no estado inicial do sistema.
13.          port_A = false;  // Informação necessária para o PGC.
14.       }else {
15.          // Linha de código RTL.
16.          port_A = true;  // Informação necessária para o PGC.
17.          // Linha de código RTL.
18.          // Linha de código RTL.
19.          port_A = false;  // Informação necessária para o PGC
20.          // Linha de código RTL.
21.          // Linha de código RTL.
22.   } // BF
23.   }// SC_MODULE
```

Na linha 1 é declarado o módulo seguido das declarações das variáveis e da interface de comunicação. Na linha 4, 5 e 6, são declaradas as interfaces de comunicação com os blocos PGC e PGFB. Na linha 8, é declarado o construtor do módulo. Da linha 10 até 22, são descritas as linhas funcionais do bloco. Nas linhas 13, 16 e 19, são realizadas atribuições na porta de comunicação com bloco PGC. O posicionamento dessas linhas vai depender da codificação de cada sistema. Neste exemplo, no estado inicial "*reset*", é atribuído o valor lógico "***false***" à porta de saída, ou seja, no início do funcionamento do sistema é enviado o

valor falso para PGC. Dependendo da execução do código RTL, será atribuído o valor lógico "***true***" ou "***false***" na porta de saída, para ser enviado ao bloco PGC. A informação enviada para PGC pode ser um valor primitivo, que é o caso do exemplo ou uma estrutura de dados complexa. Tudo depende de qual tipo de informação o PGC necessita para a tomada de decisão.

O código 5.3 descreve, em termos gerais, a estrutura em linhas de código SystemC RTL do bloco "PGFB". A estrutura é semelhante ao módulo "Bloco Funcional".

Código 5.3 - Estrutura do PGFB.

```
1.      SC_MODULE(PGFB) {  // Declaração do módulo.
2.          // Declaração de variáveis.
3.          int cont;  // Declaração da variável que não será retida.
4.          int aux;   // Declaração da variável
5.          // Declaração das interfaces de entrada e saída.
6.          sc_in   <sc_lv<4> > port_in;    // Interface de comunicação com Bloco Funcional.
7.          sc_out<sc_lv<4> > port_out;  // Interface de comunicação com Bloco Funcional.
8.
9.      SC_CTOR(PGFB) {  SC_METHOD(BF);  sensitive_pos << clk; } // construtor
10.
11.     void BF(){
12.         if (reset) {
13.             // Atribuição as variáveis no estado inicial do sistema.
14.             cont = 0;  // Atribuição de  valor.
15.         }else {
16.            // Linha de código RTL.
17.            // Linha de código RTL.
18.            cont = cont + 1;  // Atribuição de um valor a teste.
19.            // Linha de código RTL.
20.     } // BF
21.     }// SC_MODULE
```

Na linha 3 é declarada uma variável do tipo inteira, que não terá seu valor retido no momento da desativação. Na linha 4 é declarada uma variável do tipo inteira que terá seu valor retido. Nas linhas 6 e 7, são declaradas as portas de comunicação com o "Bloco Funcional". Na linha 14, será atribuído o valor "0" à variável **cont**, ação que ocorrerá no início da execução do sistema, porém, dependendo do código RTL, a variável **cont** será incrementada.

O código 5.4 descreve, em termos gerais, a estrutura em linhas de código SystemC RTL do bloco "PGC". A estrutura é semelhante ao módulo "Bloco Funcional".

Código 5.4 - Estrutura do bloco PGC.

```
1.  SC_MODULE( Bloco_PGC ) {
2.      // Declaração de variáveis.
3.      // Declaração das interfaces de entrada e saída.
4.          sc_in<bool > port_in;  // Interface de comunicação com bloco funcional.
5.
6.  SC_CTOR(Bloco_PGC) {   // Construtor
7.      SC_METHOD(PGC);  sensitive_pos << clk;
8.  } // SC_CTOR
9.
10. void PGC(){
11.     if (reset) {
12.         // Atribuição as variáveis no estado inicial do sistema.
13.     }else {
14.       // Linha de código RTL.
15.       // Linha de código RTL.
16.       if (!port_in) {
17.           sc_lp_turn_off_method("nome_PGFB"); //  desativação
18.           pglib_sc_lv_set("pgfb_i.port_out"); // Não retenção dos valores
19.           pglib_int_set("pgfb_i.cont"); // Não retenção dos valores
20.       }
21.       // Linha de código RTL.
22.       // Linha de código RTL.
23.       if (port_in) {  sc_lp_turn_on_method("nome_PGFB", sc_time(1000,SC_NS));} //  ativação
24.       // Linha de código RTL.
25.       // Linha de código RTL.
26. } // PGC
27. }// SC_MODULE
```

No código 5.4, dependendo da execução do código RTL e da informação recebida pela "port_in", caso seja "***false"***, serão chamados os métodos ***sc_lp_turn_off_method("nome_PGFB")*** contendo o nome do PGFB que deve ser desativado e os métodos ***(pglib_sc_lv_set e pglib_int_set)*** contendo o nome das variáveis do PGFB que devem ter seus valores alterados. Neste caso, independente do valor que esteja na variável **cont**, o valor será substituído por "**0**" e independente do valor da **port_out** do PGFB, ele será substituído pelo valor "**0000**".

Caso a informação recebida pela "port_in", seja ***"true"***, será chamado o método ***sc_lp_turn_on_method("nome_PGFB", sc_time(1000,SC_NS))*** contendo o nome do PGFB que deve ser ativado e o tempo de 1000 ns que o bloco deverá aguardar para poder entrar

em funcionamento. Neste caso, após ativação, os valores da variável "**cont**" e da "**port_out**" serão respectivamente "**0**" e "**0000**". As demais variáveis e portas do PGFB tiveram seus estados restaurados após ativação. Dessa forma, é possível simular o comportamento da célula de isolamento, que colocou sua saída no nível lógico baixo. A variável **"aux"** descrita no Código 5.3 terá seu valor retido, ao contrário da variável **"cont"**, quando o PGFB for desativado, simulando assim a retenção parcial das variáveis do PGFB.

Dependendo da complexidade do sistema, pode existir mais de uma chamada de cada método ***sc_lp_turn_off_method*** e ***sc_lp_turn_on_method,*** distribuídas pelo código do PGC. Cada chamada dependerá das informações recebidas do sistema. Essas informações podem ser originadas de um ou mais blocos funcionais. No exemplo utilizado, foi adicionado apenas um PGFB, mais é possível aplicar a técnica em vários blocos funcionais, pois os blocos são identificados pelos seus respectivos nomes.

5.3. Simulação de *Power Gating* no nível ESL

Uma característica fundamental de um circuito digital no nível ESL é a modelagem de sua arquitetura baseada em um conjunto de módulos integrados, se comunicando por meio de transações que fluem por canais abstratos (BAILEY; MARTIN, 2010; GHENASSIA, 2005). Em SystemC, os módulos podem ser codificados utilizando SC_THREAD e/ou SC_METHOD e os canais abstratos podem ser codificados utilizando SC_FIFO.

O processo SC_METHOD possui a característica de ser não bloqueante, portanto, não é permitido o uso de chamadas a métodos bloqueantes, como por exemplo, SC_FIFO. Esta estrutura de dados é bloqueante, pois as funções ***read()*** e ***write()*** invocam o método ***wait()*** implicitamente. Os processos SC_METHOD podem usar canais abstratos, mas são obrigados a utilizar métodos não-bloqueantes (BLACK *et al.*, 2010).

Para os modelos abstratos, a codificação utilizando SC_THREAD é mais fácil e conveniente. Os *threads* são habilitados a usar as chamadas de bloqueio, simplificando a utilização dos canais abstratos de comunicação, SC_FIFO (BAILEY; MARTIN, 2010; BLACK et al., 2010). Portanto, para realizar a simulação de *Power Gating* em ESL, foram adicionadas novas funcionalidades ao núcleo do simulador SystemC-LP V2 para lidar com módulos SC_THREAD e foi criada uma nova estrutura de dados FIFO, para contemplar informações

temporais. Essa nova estrutura de dados foi declarada como SC_FIFO_LP (*SC_FIFO-Low Power*). As modificações aplicadas no simulador foram baseadas no comportamento do funcionamento do PGFB e podem ser resumidas da seguinte forma:

- Quando PGFB está ativado, a saída do submódulo é uma função das transações de entrada e do seu estado interno;
- Quando PGFB está desativado, não são produzidas transações na sua saída;
- Quando PGFB está desativado, transações que chegam na entrada são descartadas;
- Quando PGFB está no intervalo entre o evento de ativação e a retomada do processamento, o submódulo deve aguardar uma quantidade de tempo para que possa voltar a realizar sua função.

5.3.1. Novas Modificações no Núcleo

O núcleo do simulador SystemC é não-preemptivo. Portanto, foi adicionada uma nova estrutura de dados **hash_map** ao núcleo do simulador para dar suporte aos processos SC_THREAD e auxiliar no bloqueio das *threads*. Cada elemento desse **hash_map** representa um módulo, SC_THREAD, do projeto e é composto por uma estrutura de dados contendo duas variáveis (Booleana e SC_TIME). A variável booleana é responsável por identificar se o módulo está ligado ou não, e a variável SC_TIME é responsável por armazenar o tempo de atraso necessário para reativar o módulo. Os elementos são acessados a partir de uma chave, consistindo do nome do módulo. A Tabela 5.4 apresenta as assinaturas das funções no arquivo ***sc_simcontext.h*** do núcleo do simulador. O novo simulador foi denominado de SC-LP_V3 (SystemC - *Low Power* Versão 3).

Tabela 5.4 - Declaração das novas funções para nível ESL.

Nome das funções
extern void sc_lp_turn_on_thread (const char module_name, sc_time wakedelay);* *extern void sc_lp_turn_off_thread (const char* module_name);*

A palavra-chave ***extern*** foi adicionada na declaração das funções. Dessa forma, a rotina pode ser chamada fora da classe ***sc_context.h***. Em outras palavras, essa função pode ser chamada pelo código do usuário em simulações regulares. As características do processo de retenção do estado interno do PGFB, nesta versão do simulador, permanecem iguais à versão anterior do simulador, ou seja, é possível realizar a retenção total e parcial.

Com relação ao mecanismo de isolamento da saída do módulo PGFB, o procedimento é o mesmo para a versão anterior do simulador. Como mencionado na Seção 5.2.4 , diante do nível de abstração de ESL, não faz sentido aplicar os valores "X" e "Z" a alguma variável ou porta de comunicação neste nível.

Além do suporte ao módulo SC_THREAD, foi adicionado ao simulador a capacidade de calcular o tempo que cada módulo permanece ativado. Essa informação pode ser utilizada na estimação de consumo de energia com o auxílio de ferramentas de terceiros e também pode ser utilizada para conseguir o melhor ajuste no PGC, dos eventos de ativação e desativação, a fim de maximizar o tempo que o PGFB fica desativado sem prejudicar o desempenho do sistema.

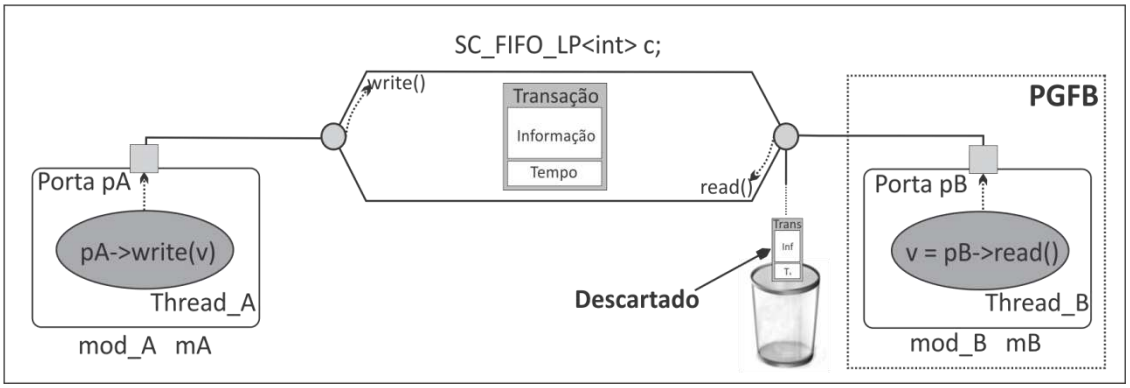
5.3.2. Criação da Classe SC_FIFO_LP

Em uma estrutura de dados FIFO da classe SC_FIFO de SystemC, as transações que são adicionadas na fila, são retiradas por ordem de chegada. Existe o conceito de ordem entre as transações que estão contidas na SC_FIFO. Mas, não é possível determinar o instante de tempo, em que cada transação foi inserida na SC_FIFO (BLACK *et al.*, 2010), tornando essas transações *Untimed.*

Devido à natureza de *Power Gating* ser *Timed*, foi criada uma nova classe denominada **SC_FIFO_LP**, na qual foi adicionado o conceito de tempo à classe SC_FIFO, em que cada transação que percorre uma SC_FIFO_LP contém a informação do instante de tempo que foi gerada. Essa estrutura de dados representa um canal de comunicação direto, com profundidade igual a um, permitindo somente o recebimento ou envio de uma transação. As informações sobre o estado do PGFB, que são necessárias para o funcionamento adequado das rotinas ***write()*** e ***read()*** da classe SC_FIFO_LP, são extraídas da nova estrutura de **hash_map** que foi adicionada ao núcleo desta nova versão do simulador.

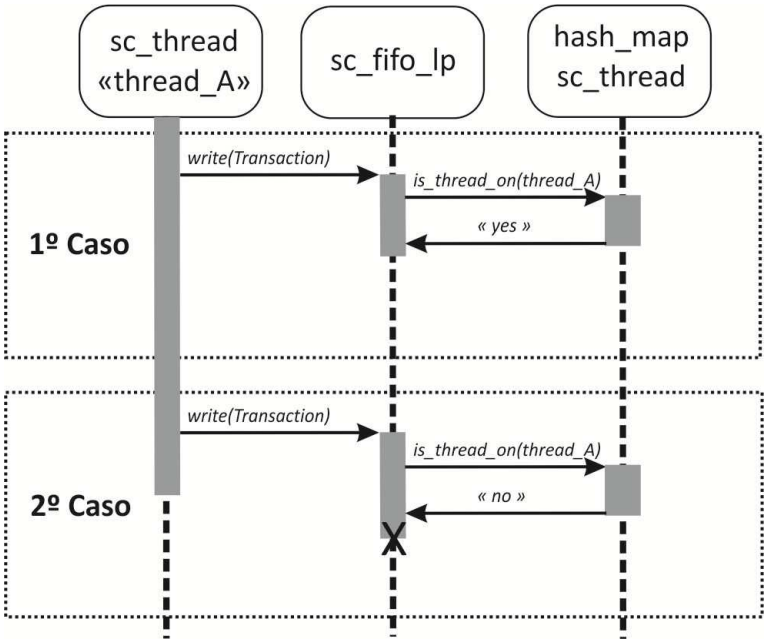
Essa estrutura é semelhante a que foi descrita na Seção 5.1.3, a diferença é que cada elemento da nova estrutura do **hash_map** representa um módulo SC_THREAD. O núcleo do simulador contém duas estruturas de **hash_map**, uma para referenciar SC_METHOD e outra para referenciar SC_THREAD. Na Figura 5.7, é apresentada a estrutura de comunicação entre dois módulos SC_THREAD, que se comunicam utilizando um SC_FIFO_LP.

Figura 5.7 - Comunicação entre Threads utilizando SC_FIFO_LP.



A semântica das rotinas ***write()*** e ***read()*** da classe **SC_FIFO_LP,** possui o seguinte comportamento. Para a rotina ***write(),*** a escrita na SC_FIFO_LP só ocorre se o módulo, que está escrevendo na FIFO, estiver ativado. Na Figura 5.8, é apresentado um diagrama de sequência, contendo os eventos que ocorrem no núcleo do simulador.

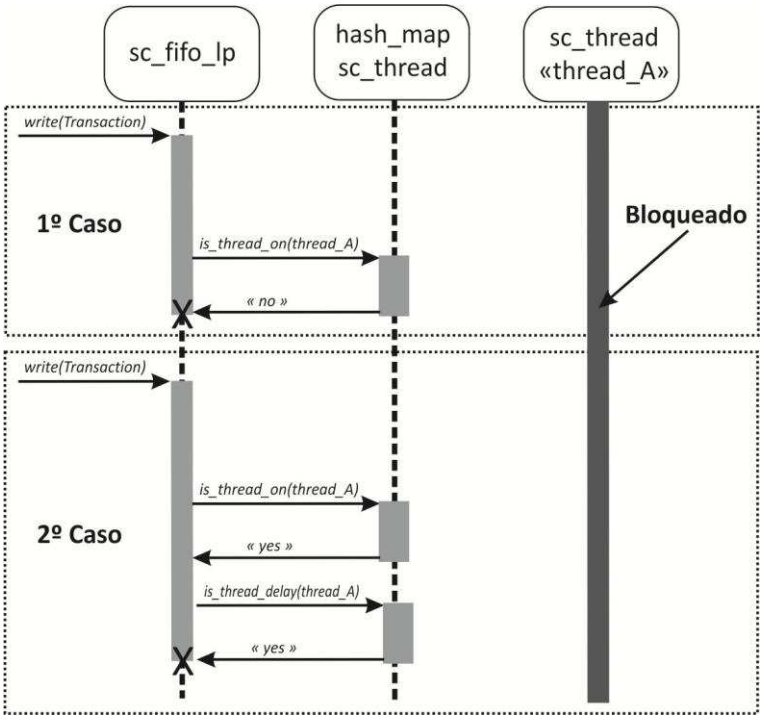
Figura 5.8 - Sequências de eventos para rotina *write*().



Quando o SC_THREAD invoca a rotina ***write(),*** internamente, a SC_FIFO_LP realiza uma consulta ao **hash_map**, que contém a relação com os SC_THREAD que foram instanciados, a fim de descobrir se o *thread* (submódulo) está ativado ou desativado. Observe na Figura 5.8, que no 1º caso, a transação é escrita com sucesso, pois o *thread* está ativada. No 2º caso, a transação é descartada.

Para a rotina ***read()***, só será permitido ler as transações pelo PGFB se as transações foram geradas enquanto o PGFB estiver ativado, ou seja, caso o PGFB esteja desativado ou esteja aguardando o tempo de ativação, qualquer transação que foi gerada durante este intervalo e for endereçada ao PGFB será descartada pela SC_FIFO_LP. Se antes do desligamento existia uma transação para ser lida e o PGFB não a leu, essa transação será descartada caso ele venha a ser desligado. Na Figura 5.9, é apresentado um diagrama de sequência, contendo os eventos que ocorrem no núcleo do simulador.

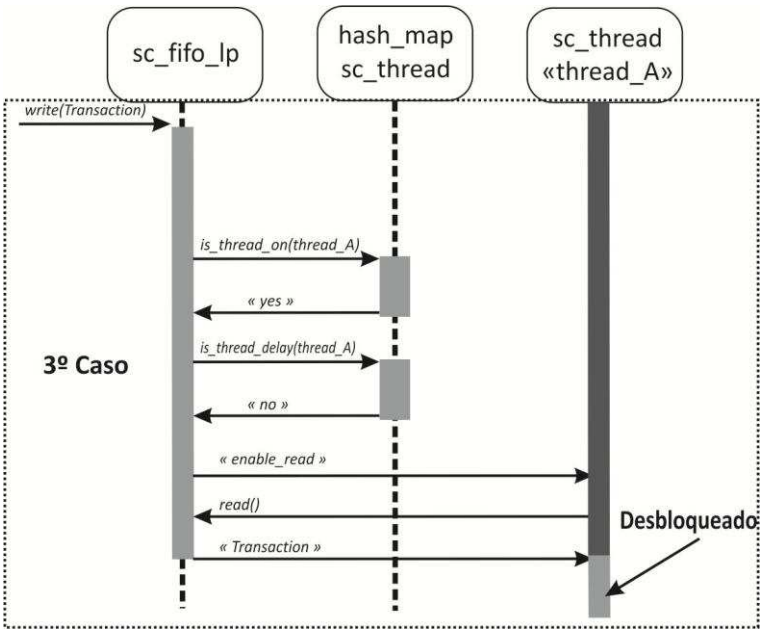
Figura 5.9 - Sequências de eventos para rotina *read*(), para SC_THREAD desativado.



De acordo com a Figura 5.9, após a escrita de uma transação na SC_FIFO_LP, internamente, ocorre uma consulta ao **hash_map** que contém a relação dos SC_THREAD que foram instanciados, a fim de saber se o *thread* (submódulo) estar ativado ou desativado, e se ele estar aguardando uma quantidade de tempo para voltar ao funcionamento. Observe na Figura 5.9, no 1º caso o SC_THREAD estava desativado, consequentemente a transação foi

descartada. No 2º caso, o SC_THREAD estava ativado, mas, aguardava o tempo para voltar a sua atividade, ocasionado o descarte da transação. Na Figura 5.10 , é apresentado um diagrama de sequência, contendo os eventos que ocorrem no núcleo do simulador, quando o SCTHREAD estar ativado. No 3º caso, o SC_THREAD realiza a leitura da transação com sucesso e volta a ficar desbloqueado.

Figura 5.10 - Sequências de eventos para rotina *read*(), para SC_THREAD ativado.



5.3.3. Princípio de Funcionamento SC_LP_V3

O funcionamento do simulador para a técnica de *Power Gating* no nível ESL é baseado no bloqueio da *thread,* devido às chamadas implícitas do método ***wait()***, pelas funções ***read()*** e ***write()*** da SC_FIFO_LP. Por exemplo, a função ***read()*** de uma instância da SC_FIFO_LP<T>, chamará o método ***wait()*** quando a FIFO estiver vazia, provocando o bloqueio da *thread*. A simulação do comportamento de *Power Gating* é baseada no bloqueio.

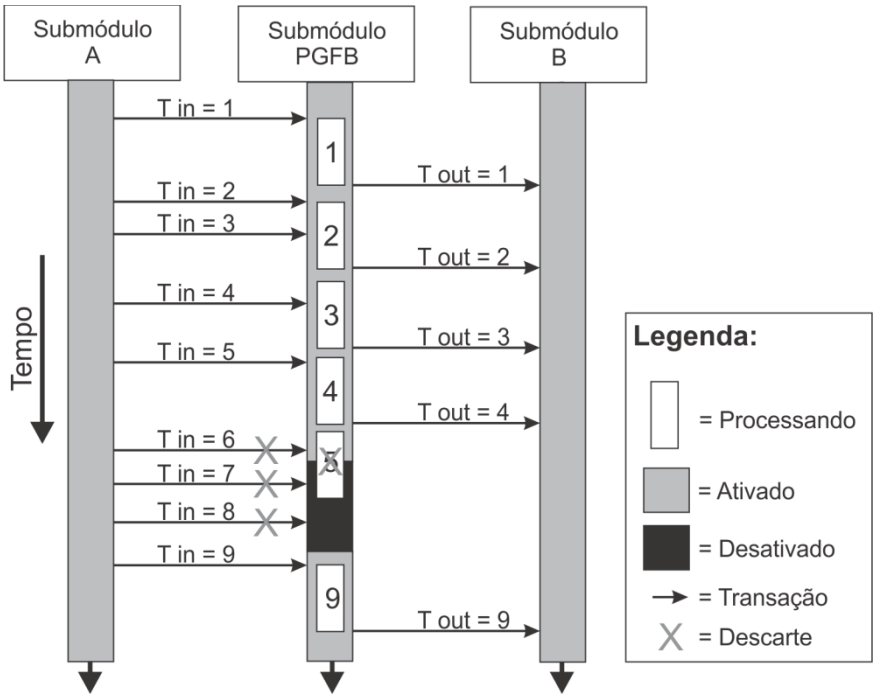
Com base no segundo parâmetro da declaração da SC_FIFO_LP, antes de realizar a função de ***read()*** ou ***write(),*** é realizada uma consulta à estrutura de dados **hash_map** que contém os nomes dos módulos SC_THREAD. Essa consulta tem como objetivos identificar se o módulo está ligado ou desligado e os instantes de tempo em que ocorreram os eventos de desativação e ativação. Com base nessas informações são realizados os descartes das

transações inválidas. A simulação do comportamento da técnica de *Power Gating* pelo simulador SC_LP_V3 será descrito a seguir.

PGFB ativado: Na ocorrência de uma transação na FIFO, destinada ao PGFB, será extraída a transação utilizando a função ***read()*** e processada normalmente. Em seguida, o resultado do processamento será transmitido pela FIFO de saída utilizando a função ***write()***. Caso ocorra o desligamento do módulo durante o processamento, a transação com o resultado será descartada da FIFO. Este descarte simula o comportamento da perda do processamento ocasionado pelo módulo estar desligado.

PGFB desativado: Caso sejam geradas transações destinadas ao PGFB enquanto estiver desativado ou em processo de ativação, essas transações serão descartadas da FIFO. Devido ao bloqueio proporcionado pela função ***read()***, as linhas de código que a procedem não serão executadas. Em resumo, se não existirem transações válidas, que foram geradas enquanto o PGFB estava em pleno funcionamento, o *thread* ficará bloqueado, simulando desta forma a falta de atividade do módulo. Para facilitar o entendimento do funcionamento da SC_FIFO_LP, a Figura 5.11 contém um diagrama com a comunicação entre 3 submódulos em função do tempo.

Figura 5.11 - Diagrama de tempo de 3 submódulos se comunicando.



A Figura 5.11 possui três submódulos, o submódulo **"A"** envia transações para o PGFB, este envia transações para **"B"**. A faixa de cor cinza indica que o submódulo está ativado, a faixa na cor preta indica que está desativado. Cada retângulo na cor branca indica que o PGFB está realizando um processamento.

O PGFB recebe a transação **Tin=1** do submódulo **"A"**, realiza seu processamento e envia o resultado **Tout=1** para o submódulo **"B"**, o mesmo ocorre com a transação **Tin=2**. A transação **Tin=3** ocorre porque a transação anterior foi lida, mas, o processamento de **Tin=3**, só ocorrerá após o processamento da **Tin=2**. Durante o processamento de **Tin=5** ocorre o desligamento do PGFB, este evento irá provocar a perda do processamento e não ocorrerá o envio de **Tout=5**.

A transação **Tin=6**, mesmo chegando antes do desligamento será descartada, pois o PGFB não realizou a leitura devido ao processamento de **Tin=5**. As transações **Tin=7** e **Tin=8** serão descartadas, pois foram geradas enquanto o PGFB estava desativado.

5.3.4. Sintaxe da Aplicação da Técnica de *Power Gating* no nível ESL

A aplicação de *Power Gating* no nível ESL é bastante semelhante à RTL, a diferença reside na codificação nos canais de comunicação e nas chamadas dos métodos para ativar e desativar o PGFB. Para simular adequadamente o comportamento da técnica no nível ESL, é necessário seguir duas regras: (i) utilizar SC_FIFO_LP em todos os canais que se comunicam com o PGFB; (ii) respeitar uma estrutura específica de codificação do módulo. A estrutura de codificação do PGFB é apresentada no Código 5.5.

Código 5.5 - Estrutura do PGFB no nível ESL.

```
1.      SC_MODULE(PGFB) {  // Declaração do módulo.
2.          // Declaração de variáveis.
3.          int trans_in_1
4.          int trans_out_1
5.          // Declaração das interfaces de entrada e saída.
6.          sc_fifo_in     int   port_in_1;   // Interface de comunicação com Bloco Funcional.
7.          sc_fifo_out    int   port_out_1;  // Interface de comunicação com Bloco Funcional.
8.
9.      SC_CTOR(PGFB) {  SC_THREAD(bf); } // construtor
10.
11.     void bf(){
```

```
12.             while (1) {
13.                 trans_in_1  = port_in_1.read();  // Leitura da transação e ponto de bloqueio
14.                 // Linha de código ESL.
15.                 // Linha de código ESL.
16.                 // Linha de código ESL.
17.                 // Linha de código ESL.
18.                 port_out_1.write(trans_out_1);
19.             } //WHILE
20.         } // BF
21.         }// SC_THREAD
```

Essa estrutura do PGFB no nível ESL (Código 5.5) é bastante semelhante ao código 5.3, mas, existe uma diferença crucial. A linha de código, onde ocorre a função de leitura da transação da FIFO deve preceder todas as linhas de código ESL, que representam a funcionalidade do módulo. Essa função é representada pela linha de número 13 "***trans_in_1 = port_in_1.read();***".

Caso o PGFB receba transações de mais de uma FIFO, as chamadas das funções ***read()*** devem ser consecutivas, para que seja possível bloquear o SC_THREAD adequadamente. Na existência de duas funções ***read()*** não consecutivas, caso o PGFB receba uma transação válida, por meio da primeira função ***read()***, o simulador irá executar a próxima instrução, o que descaracteriza o desativamento do PGFB. Mas, se a próxima instrução for uma função ***read()***, o PGFB permanecerá bloqueado até que receba uma transação válida. Por exemplo, caso um PGFB realize uma função que dependa de 2 transações, e que sejam originadas de submódulos distintos, o PGFB só deve executar a operação (instruções) após receber ambas as transações.

O Código 5.6, descreve em termos gerais a estrutura em linhas de código SystemC TLM do arquivo sistema.cpp. Nas linhas 1 e 2, são adicionados as bibliotecas SystemC e PGLIB, as linhas 4, 5 e 6 adicionam os cabeçalhos dos módulos Bloco_Funcional, PGFB e PGC. A instanciação dos módulos é realizada nas linhas 11, 12 e 13. Logo após, são declarados as SC_FIFO_LP dos módulos nas linhas 16 e 17. Em seguida, são realizadas as conexões entres as instâncias dos módulos e as FIFO.

Código 5.6 - Estrutura básica do sistema.cpp.

```
1.    #include "systemc.h"  // Adição da biblioteca SystemC
2.    #include "pglib.h"  // Adição da biblioteca PGLIB
3.
4.    #include "bloco_funcional.h"
5.    #include "pgfb.h"
6.    #include "pgc.h"
7.
8.    int sc_main (int argc, char *argv[])
9.    {
10.     // Instanciação dos módulos
11.     pgc  pgc_i("pgc_i");
12.     pgfb pgfb_i("pgfb_i");
13.     bloco_funcional  bf_i("bf_i");
14.
15.     // Declaração das FIFOs
16.     sc_fifo_lp< int >  bf_to_pgfb ("bf_to_pgfb", "bf_i.bf ", "pgfb_i.bf", logfile);
17.     sc_fifo_lp< int *> pgfb_to_bf ("pgfb_to_bf", "pgfb_i.bf", "bf_i.bf", logfile);
18.
19.     // Conexões dos módulos
20.     bf_i.port_out_1(bf_to_pgfb);
21.     pgfb_i.port_in_1 (bf_to_pgfb);
22.     pgfb_i.port_out_1 (pgfb_to_bf);
23.     bf_i.port_in_1(pgfb_to_bf);
24.
25.     sc_start();
26.     return 0;}
```

A declaração da SC_FIFO_LP possui 4 parâmetros. Será utilizado a linha 16 para exemplificar a descrição: 1º) ***"bf_to_pgfb",*** nome da SC_FIFO_LP, para ser utilizado no arquivo de log; 2º) "bf_i.bf" nome do módulo que vai escrever as transações da FIFO, 3º) ***"pgfb_i.bf"***, nome do módulo que vai ler as transações da FIFO; 4º) ***"logfile"***, nome do arquivo onde serão registrados as transações. Na codificação dos demais módulos, deve ser alterada a chamada do método ***sc_lp_turn_on_method*** pelo método ***sc_lp_turn_on_thread***.

5.4. Considerações Finais do Capítulo

Neste capítulo foi apresentado um simulador OSCI SystemC, que foi modificado para suportar a simulação de projetos que contenham a técnica de *Power Gating* descrita no nível RTL e/ou ESL com base na linguagem SystemC. Foram apresentadas algumas características do simulador, por exemplo, flexibilidade de codificação e suas restrições.

Com base nas características funcionais do simulador e as necessidades de simular o comportamento de *Power Gating*, foram adicionadas novas funções ao núcleo do simulador e desenvolvida uma biblioteca para auxiliar na implementação das técnicas para retenção de valores e no isolamento do PGFB. Foram apresentados exemplos dos princípios e regras que devem ser seguidos para a aplicação das novas funções e a descrição do funcionamento do novo simulador, que foi denominado SystemC - *Low Power* V3. Por fim, foi apresentada uma abordagem para auxiliar na aplicação e verificação da técnica de *Power Gating*.

Capítulo 6 - Apresentação e Análises dos Resultados

Neste capítulo, serão apresentados os resultados referentes à simulação e verificação funcional da técnica de *Power Gating* utilizando SystemC RTL e TLM. Quatro estudos de caso que foram realizados: (i) verificação funcional da técnica de *Power Gating* utilizando SystemC RTL; (ii) verificação funcional da técnica de *Power Gating no nível ESL*, utilizando o projeto do estudo de caso anterior; (iii) verificação funcional de *Power Gating* no nível ESL, utilizando SystemC, aplicado ao verificador de identidade vocal; (iv) verificação funcional de *Power Gating* no nível ESL e RTL, utilizando SystemC, aplicada a um decodificador vídeo MPEG-4. Os resultados estão agrupados de acordo com as etapas em que foram executadas.

6.1. Metodologia e Verificação Funcional

O simulador SystemC, descrito no Capítulo 5, oferece suporte à simulação da técnica de *Power Gating* aplicado em projetos de circuito digitais nos níveis RTL e ESL. Essa ferramenta é apenas um simulador de circuitos digitais. Para realizar a verificação funcional de circuitos digitais, foram utilizadas as bibliotecas SCV (SystemC *Verification*) (ACELLERRA, 2012) e BVE-COVER (*Brazil-IP Verification Extension*) (SILVA, 2007).

A biblioteca SCV é um conjunto de APIs (*Application Programming Interface*), utilizadas no processo de verificação com SystemC. Ela fornece, por exemplo, a geração de valores com base em restrições, gravação de transação, etc. As APIs de SCV são implementadas em outros simuladores SystemC. A biblioteca BVE-COVER (*Brazil-IP Verification Extension*) tem o objetivo de encontrar "buracos" de cobertura e mostrar o progresso da simulação.

Nos processos de verificação funcional com cobertura nos 4 estudos de caso, foi adotada a metodologia VeriSC, que suporta o conceito de hierarquia de projetos, de modo que cada um pudesse ser dividido em partes, a fim de serem implementados e verificados. Os componentes dos *testbench* foram implementados de acordo com a metodologia, utilizando a linguagem SystemC. É importante ressaltar que é possível utilizar qualquer metodologia de verificação, pois o simulador é independente da metodologia.

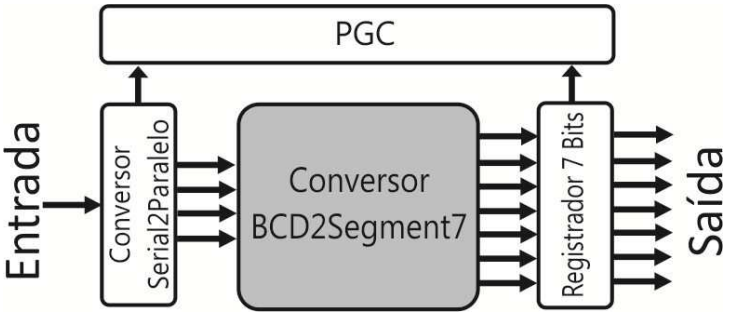
Em todas as simulações foi utilizado o mesmo computador, contendo a seguinte configuração: processador Intel® Core 2 Duo T5800, 3 GB de memória principal PC2700, HD Sata2, sistema operacional CentOS 4.8, GCC 3.2.3, SystemC 2.2.0 e SCV 1.0p2.

6.2. Estudo de Caso 1 - Decodificador BCD para *display* de sete segmentos, nível RTL

6.2.1. Descrição do Projeto

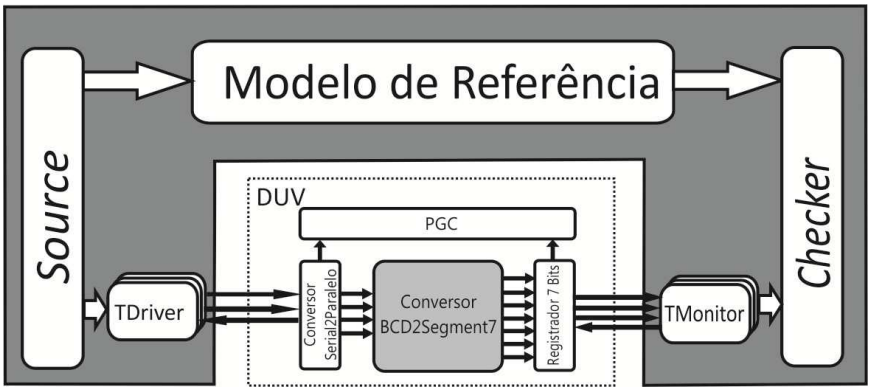
Este primeiro estudo de caso tem como objetivo demonstrar a verificação funcional de um circuito digital, relativamente simples, contendo a técnica de *Power Gating* descrito em SystemC RTL. Foi desenvolvido um projeto de um decodificador com a seguinte especificação: o circuito irá receber serialmente palavras de 4 bits no formato BCD (*Binary Coded Decimal*) e as converterá para o formato de display de segmentos. Depois da conversão do dado, este é disponibilizado na saída do circuito. Na Figura 6.1, é apresentado o diagrama em blocos do projeto, com o PGC, em que o bloco em tom de cinza foi escolhido para ser o PGFB.

Figura 6.1 - Diagrama em blocos do circuito.



Na Figura 6.2, é apresentado o diagrama do DUV inserido no *testbench*. Os blocos que estão dentro do retângulo pontilhado representam o DUV. O submódulo na cor cinza, do DUV, representa o PGFB.

Figura 6.2 - Diagrama em blocos do DUV inserido no *testbench*.



6.2.2. Metodologia

O projeto do decodificador foi implementado de quatro formas diferentes, descritas a seguir.

- **REFMOD_Untimed_TLM**: Foi descrito em SystemC, no nível de abstração ESL, utilizando a semântica de Untimed-TLM, com o objetivo de ser o modelo de referência do *testbench*. A implementação foi realizada da seguinte forma: os dois conversores Serial2Paralelo e BCD2Segment7 foram implementados, sendo que, a saída do conversor Serial2Paralelo é conectada diretamente à entrada do conversor BCD2Segment7, este conectado ao Registrador. O comportamento do circuito pode ser descrito da seguinte forma: Enquanto o conversor Serial2Paralelo paraleliza os bits que vão sendo lidos, o conversor BCD2Segment7 prossegue convertendo os valores que são fornecidos pelo conversor Serial2Paralelo, em outras palavras, a cada mudança de estado do conversor Serial2Paralelo decorrente da leitura dos bits ocorrerá uma mudança de estado do BCD2Segment7. Após a leitura completa da palavra de 4 bits pelo conversor Serial2Paralelo, o circuito Registrador irá armazenar a saída do BCD2Segment7 e disponibiliza o resultado final em sua saída.

- **DUV_RTL:** Foi descrito em SystemC RTL. A forma de implementar e o comportamento do circuito são idênticos ao REFMOD_Untimed_TLM.
- **DUV_RTL_PG:** O projeto foi implementado utilizando a técnica de *Power Gating* e foi descrito em SystemC RTL. A implementação é semelhante ao DUV_RTL. O comportamento desta versão difere do REFMOD_Untimed_TLM no seguinte ponto: enquanto o conversor Serial2Paralelo paraleliza a palavra de 4 bits, o conversor BCD2Segment7 é desativado e seu estado fica retido. Depois da leitura completa da palavra de 4 bits pelo conversor Serial2Paralelo, o conversor BCD2Segment7 é reativado. Deste ponto em diante, o conversor BCD2Segment7 aguardará um tempo de 2 μs que corresponde ao atraso entre o momento de ativação e o retorno de sua plena atividade. Durante este tempo, o estado anterior é restaurado. Após decorrer os 2 μs, o conversor realiza sua função e no passo seguinte o valor é armazenado pelo Registrador e disponibilizado na saída do circuito. Logo em seguida, o submódulo BCD2Segment7 é desativado novamente.

As simulações foram executadas em 3 etapas, cada simulação foi executada até alcançar 100% de cobertura funcional. O REFMOD_Untimed_TLM foi utilizado em todas as etapas como o modelo de referência. As etapas são descritas a seguir.

I. **SystemC + DUV_RTL**: Nesta etapa foram utilizados o simulador OSCI SystemC original versão 2.2.0 e a implementação do DUV_RTL.

II. **SystemC–LP + DUV_RTL:** Nesta etapa foram utilizados o simulador SystemC-LP_V3 com suporte a *Power Gating* e a implementação do DUV_RTL.

III. **SystemC–LP + DUV_RTL_PG:** Nesta etapa foram utilizados o simulador SystemC-LP_V3 com suporte a *Power Gating* e a implementação do DUV_RTL_PG.

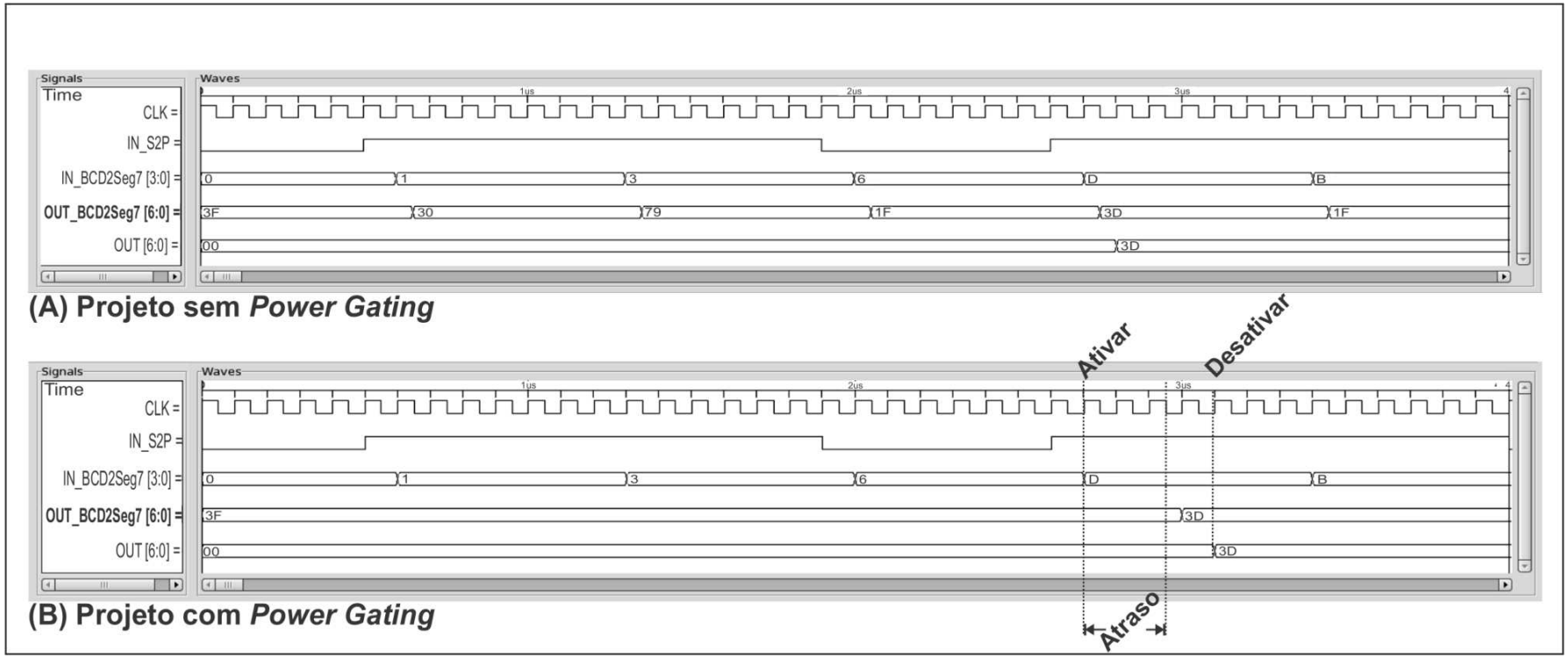
6.2.3. Resultados e Análises

Ao final da segunda etapa de simulação, o DUV_RTL foi verificado com sucesso utilizando as duas versões do simulador SystemC, tanto com a versão original quanto com a

versão modificada. Portanto, demonstra que as novas funções para manipular SC_METHOD, adicionadas ao núcleo do simulador, não interferiram nas demais funções do simulador.

Foram adicionadas 20 linhas de código extra para implementar as funções de PCG que consiste de uma máquina de estado que chama os novos métodos do SystemC. As linhas adicionadas representam 12% do total de linhas de código do DUV. Durante a simulação, pôde-se verificar os princípios de operação da técnica. Na Figura 6.3, são apresentada as formas de onda do funcionamento do circuito.

Figura 6.3 - Formas de Ondas.



Pode-se observar a partir da Figura 6.3(A) as mudanças de estado do conversor BCD2Segment7, enquanto o conversor Serial2paralelo realiza a paralelização da palavra de 4 bits. As variáveis IN_BCD2Seg7 e OUT_BCD2Seg7 são respectivamente a entrada e a saída do módulo PGFB. Na Figura 6.3(B), é possível visualizar que, durante a paralelização da palavra, o módulo BCD2Segment7 manteve o estado da saída constante devido à retenção de seu estado que ocorreu antes do desligamento. Ao final da leitura da palavra, o conversor BCD2Segment7 volta a atividade para realizar sua função e fornecer o valor na saída.

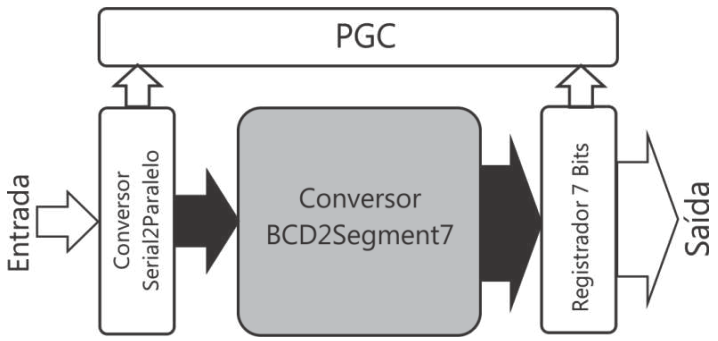
Outro ponto que pode ser observado a partir da Figura 6.3(B) é o atraso entre o momento da ativação e o tempo de retomada da atividade. Para este projeto, foi utilizado o atraso de 2µs. A ativação do conversor BCD2Segment7 ocorre logo após a paralelização dos 4 bits e a retomada das funções após 2 µs. O conversor BCD2Segment7 é sensível à transição negativa do sinal de *clock*. Diante deste fato, o atraso total é de 2,5 µs. Depois de repassar o novo resultado para a saída do circuito, o conversor BCD2Segment7 volta a ser desativado. Foi gasto com a simulação, o tempo de 0,53 ms para alcançar 100% de cobertura funcional.

6.3. Estudo de Caso 2 - Decodificador BCD para display de sete segmentos, nível ESL

6.3.1. Descrição do Projeto

Neste estudo de caso, o objetivo é realizar a verificação funcional de *Power Gating* em ESL. Na Figura 6.4, é apresentado o diagrama em blocos do projeto com o PGC, as setas na cor preta são FIFO-LP.

Figura 6.4 - Diagrama em blocos do circuito com PGC.



6.3.2. Metodologia

Foi utilizado o mesmo projeto do estudo de caso anterior com algumas alterações. Foram reimplementados dois módulos, que são descritos a seguir.

- **DUV_*Timed*_TLM**: Foi descrito em SystemC, no nível de abstração ESL, utilizando a semântica de *Timed*-TLM, com o objetivo de explorar arquiteturalmente o projeto. A implementação é semelhante ao REFMOD_*Untimed*_TLM, diferenciando-se na

conexão dos submódulos Serial2Paralelo, BCD2Segment7 e do Registrador, que é realizada por meio de uma estrutura de dados FIFO (*First In First Out*). Esta implementação possui informações temporais, cada submódulo consome 5µs para realizar sua função.

- **DUV_Timed_TLM_PG**: O projeto foi implementado utilizando a técnica de *Power Gating* e foi descrito em SystemC no nível de abstração ESL, utilizando a semântica de *Timed*-TLM. A forma da implementação é semelhante ao DUV_*Timed*_TLM, diferenciando-se na substituição da tradicional SC_FIFO pela SC_FIFO_LP (First In First Out - Low Power) que conecta o submódulo BCD2Segment7. Foi adicionado o submódulo PGC, com a função de chamar os novos métodos do SystemC-LP. O PGC recebe informações dos submódulos Serial2Paralelo e Registrador por meio de FIFO-LP. O comportamento desta versão difere do REFMOD_Untimed_TLM no seguinte ponto: enquanto o conversor Serial2Paralelo paraleliza a palavra de 4 bits, o conversor BCD2Segment7 é desativado e seu estado fica retido. Depois da leitura completa da palavra de 4 bits pelo conversor Serial2Paralelo, o conversor BCD2Segment7 é reativado. Deste ponto em diante, o conversor BCD2Segment7 aguardará um tempo de 2µs, que corresponde ao atraso entre o momento de ativação e o retorno de sua plena atividade, durante este tempo o estado anterior é restaurado. Após decorrer os 2 µs, o conversor realiza sua função e no passo seguinte o valor é armazenado pelo Registrador e disponibilizado na saída do circuito. Logo em seguida, o submódulo BCD2Segment7 é desativado novamente.

As simulações foram executadas em 3 etapas. Cada simulação foi executada até alcançar 100% de cobertura funcional. O REFMOD_Untimed_TLM foi utilizado em todas as etapas como modelo de referência. As etapas são descritas a seguir.

1. **SystemC + DUV_Timed_TLM**: Nesta etapa foram utilizados o simulador original OSCI SystemC V2.2.0 e a implementação do DUV_*Timed*_TLM.
2. **SystemC-LP + DUV_Timed_TLM**: Nesta etapa foram utilizados o simulador SystemC com suporte a *Power Gating* para o nível ESL e a implementação do DUV_*Timed*_TLM.

3. **SystemC-LP + DUV_Timed_TLM_PG**: Nesta etapa foram utilizados o simulador SystemC com suporte a *Power Gating* para o nível ESL, e a implementação DUV_*Timed*_TLM_PG contendo a técnica de *Power Gating* no nível ESL.

6.3.3. Resultados e Análises

Ao final da 2ª etapa de simulações, o DUV_Timed_TLM foi verificado com sucesso utilizando as duas versões do simulador SystemC, tanto com a versão original quanto com a versão modificada. Portanto, as novas funções para manipular SC_THREADS, não interferiram nas demais funções do simulador. Apôs as simulações da 3ª etapa pode-se destacar a efetiva simulação do comportamento da técnica de *Power Gating*, pois foi possível simular DUV_TLM_PG utilizando a nova versão do simulador SystemC-LP_V3. Na Figura 6.5, é apresentada parte do *log* da simulação da verificação funcional do projeto, em que o comportamento da técnica de *Power Gating* pode ser verificado por meio das informações dos eventos. Estes ocorreram dentro do intervalo de tempo para o sistema disponibilizar na saída o primeiro resultado da conversão.

Figura 6.5 - *Log* da verificação funcional do sistema.

```
                SystemC 2.2.0 --- Jun 29 2012 08:52:28
            Copyright (c) 1996-2006 by all Contributors
                        ALL RIGHTS RESERVED
        (Modified to support Power Gate by George S. Silveira - V 3)
********************************************************************************************
  Time  |                    Process                                    |  Operation
********************************************************************************************
  0   s      TURN OFF: BCD_7segment,  Sleep Time: 0 s,   Time_ON = 0 s
  1 us       RUN  Serial2Paralelo
  6 us       SC_FIFO_LP : sp_to_bcd7segment  Value: bcd=(1 )            Write
  6 us       SC_FIFO_LP : sp_to_bcd7segment  Value: bcd=(1 )            Delete
  6 us       RUN  Serial2Paralelo
 11 us       SC_FIFO_LP : sp_to_bcd7segment  Value: bcd=(3 )            Write
 11 us       SC_FIFO_LP : sp_to_bcd7segment  Value: bcd=(3 )            Delete
 11 us       RUN  Serial2Paralelo
 16 us       SC_FIFO_LP : sp_to_bcd7segment  Value: bcd=(6 )            Write
 16 us       SC_FIFO_LP : sp_to_bcd7segment  Value: bcd=(6 )            Delete
 16 us       RUN  Serial2Paralelo
 16 us      TURN ON: BCD_7segment,   Wake Time: 18 us
 21 us       SC_FIFO_LP : sp_to_bcd7segment  Value: bcd=(13 )           Write
 21 us       SC_FIFO_LP : sp_to_bcd7segment  Value: bcd=(13 )           Read
 21 us       RUN  BCD2Segment7
 24 us       SC_FIFO_LP : bcd7segment_to_reg  Value: segment7=(61 )   Write
 24 us       SC_FIFO_LP : bcd7segment_to_reg  Value: segment7=(61 )   Read
 24 us       RUN  Registrador
 25 us      TURN OFF: BCD_7segment,  Sleep Time: 25 us,   Time_ON = 7 us
```

A seguir, são descritos os principais eventos do log.

- No início da execução do sistema, o submódulo BCD2Segment7 é desativado;
- Em 1 µs, o submódulo Serial2Paralelo inicia seu processamento, lê uma transação originada do Source, processa a transação adequadamente e escreve a transação (resultado) na SC_FIFO_LP:sp_to_bcd7segment no instante de tempo 6µs;
- As transações escritas pelo submódulo Serial2Paralelo, nos tempos (6, 11 e 16) µs, na SC_FIFO_LP:sp_to_bcd7segment, que são geradas enquanto o submódulo BCD2Segment7 está desativado, são descartadas na saída da SC_FIFO_LP;
- No instante de 16 µs, após a 3ª escrita na SC_FIFO_LP:sp_to_bcd7segment, pelo submódulo Serial2Paralelo, é gerado o evento de ativação do submódulo BCD2Segment7, com um tempo de espera para ativação de 1µs.
- No instante de 18µs o submódulo está ativado e qualquer transação escrita deste instante em diante, na SC_FIFO_LP:sp_to_bcd7segment, será processada;
- No instante de 21µs, após a 4ª escrita, na SC_FIFO_LP:sp_to_bcd7segment, pelo submódulo Serial2Paralelo, e como o submódulo BCD2Segment7 está ativado, este realiza a leitura da transação na SC_FIFO_LP:sp_to_bcd7segment, executa o processamento e escreve a transação (resultado) na SC_FIFO-LP:bcd7segment_to_reg no instante de tempo 24µs. Neste mesmo instante, o submódulo Registrador lê a transação da SC_FIFO_LP:bcd7segment_to_reg;
- No instante de 25 µs, o submódulo BCD2Segment7 volta a ser desativado.

Diante da descrição do *log* da simulação do sistema, pode-se observar, com base nos eventos, os princípios de operação da técnica de *Power Gating*. Uma informação relevante que pode ser extraída do *log,* **"Time_ON = 7 µs",** é o tempo que o submódulo BCD2Segment7 permanece ativado, o que corresponde a 28% do tempo total. Esse valor pode ser utilizado como parâmetro para estimar a economia de energia com utilização da técnica de *Power Gating*. Foi gasto com a simulação, o tempo de 0,56 ms para alcançar 100% de cobertura funcional.

6.4. Estudo de Caso 3 - SPVR, no nível ESL

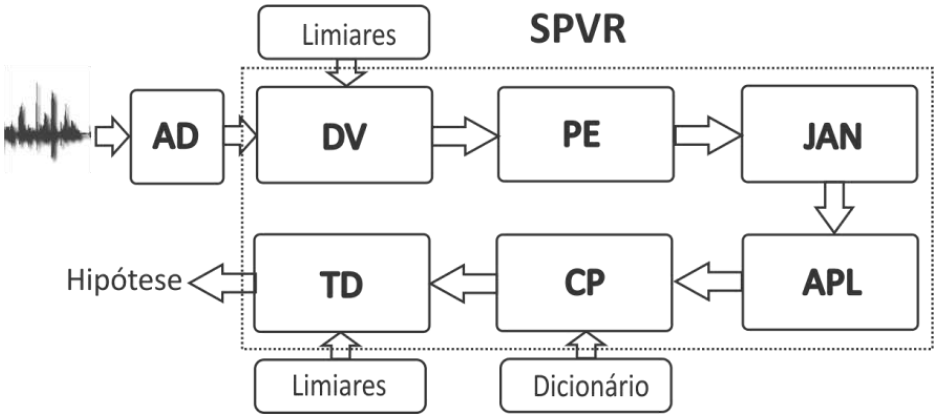
6.4.1. Descrição do Projeto

Este estudo de caso é baseado no projeto SPVR (*Speaker Verification*), desenvolvido no Laboratório de Arquiteturas Dedicadas (LAD), dentro do programa Brazil-IP, que permite verificar a identidade vocal de um locutor. O SPVR foi implementado em silício, com 55 mm^2 de área, tecnologia CMOS de 0.35 µm com quatro níveis de metal e frequência de operação de 781 kHz.

O SPVR compara as características do sinal de áudio de uma locução com um dicionário que contém as características vocais do locutor, o qual se deseja verificar. O sistema fornece uma hipótese, cujo valor pode ser Aceito, Rejeitado ou Desconhecido. O dicionário é produzido *off-line* na fase de treinamento do sistema (FECHINE et al., 2010; MELCHER et al., 2011).

O sistema foi dividido em seis módulos funcionais: Detector de Voz (DV), Pré-Ênfase (PE), Janelamento (JAN), Análise por Predição Linear (APL), Comparação de Padrões (CP) e Tomador de Decisão (TD). Todas as funcionalidades dos blocos foram implementadas com máquinas de estados finitos, com o objetivo de reduzir o consumo de energia (FECHINE *et al.*, 2010). Na Figura 6.6, é apresentado o diagrama em blocos do SPVR.

Figura 6.6 - Diagrama em bloco do SPVR



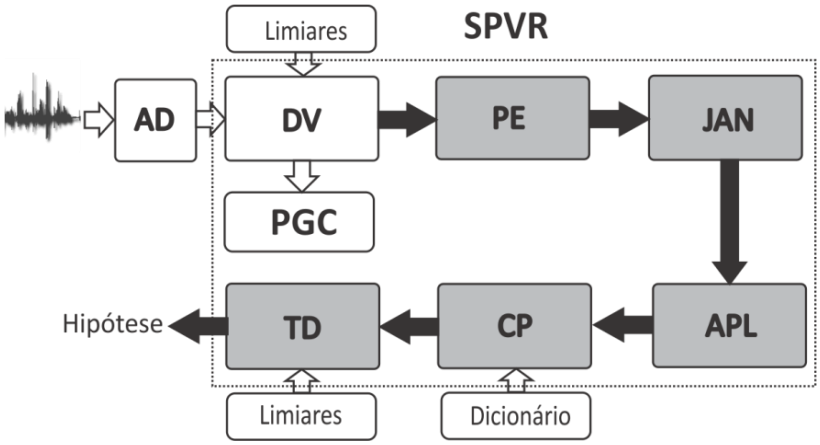
Fonte: (FECHINE *et al.*, 2010).

6.4.2. Metodologia

Na fase de concepção e modelagem arquitetural do sitema, foi utilizado um código escrito na liguagem C++, como modelo de referência no nível ESL. Com base neste código e utilizando as técnicas de SystemC-TLM, o sistema foi modelado em blocos e conectados por meio de FIFO. Devido à inexistência de código RTL descrito em SystemC, não foi possível realizar a verificação do SPVR no nível RTL.

Foram adicionadas informações temporais aos blocos do DUV e inserido o bloco PGC ao sistema. O bloco DV é responsável por detectar sequências de amostras de voz válidas e encaminhar essas sequências para os demais blocos, enquanto o DV não detectar uma amostra válida, os demais módulos permaneceram ociosos. Devido à essa característica do sistema, a técnica de *Power Gating* foi aplicada aos submódulos PE, JAN, APL, CP e TD. Na Figura 6.7, é apresentado o diagrama do SPVR com o módulo PGC.

Figura 6.7 - Diagrama do SVPR com PGC.



Fonte: Adaptada de (FECHINE *et al.*, 2010).

Todos os PGFB foram conectados aos demais por meio de SC_FIFO-LP, representadas na Figura 6.7 por setas pretas. As informações necessárias para o funcionamento do PGC são extraídas do bloco DV. Em síntese, caso seja detectada uma sequência de amostras de voz válida, pelo submódulo DV, o PGC irá ativar os PGFB, caso não, ele manterá todos desativados.

As simulações foram agrupadas em 3 etapas em que cada uma é constituída de 4 simulações, utilizando 2 arquivos de áudio de locutores distintos e seus respectivos

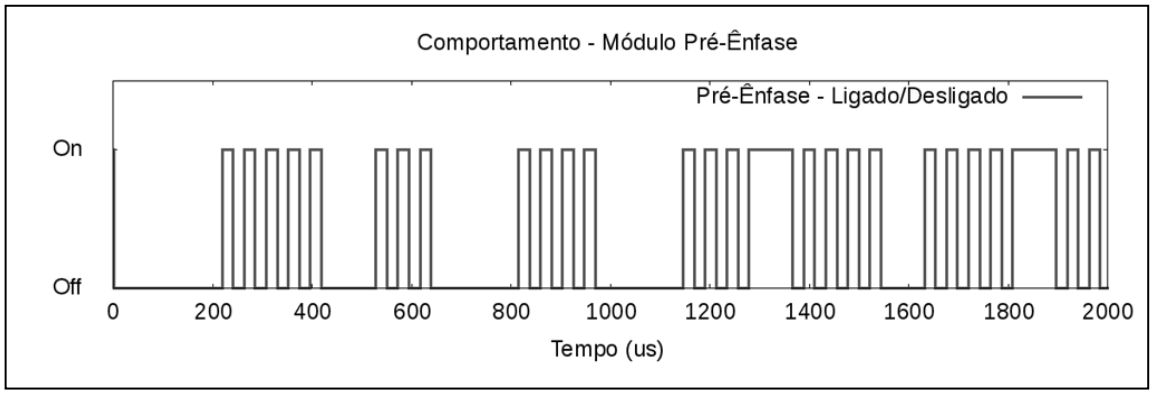
dicionários. Os arquivos de áudio estão no formato WAV (*Waveform Audio File Format*), com aproximadamente 30 segundos de duração e frequência de amostragem de 11.025 Hz. Em cada simulação, foi utilizado um arquivo de áudio com um dicionário, obtido a partir da Quantização Vetorial, representativo de cada locutor. As etapas são descritas a seguir.

1. **SystemC + SPVR_Untimed_TLM**: Nesta etapa, foram utilizados o simulador original OSCI SystemC V2.2.0 e a implementação do SPVR_*Untimed*_TLM.
2. **SystemC-LP + SPVR_Untimed_TLM**: Nesta etapa, foram utilizados o simulador SystemC com suporte a *Power Gating* para o nível ESL e a implementação do SPVR_*Untimed*_TLM.
3. **SystemC-LP + SPVR_Timed_TLM_PG**: Nesta etapa, foram utilizados o simulador SystemC com suporte a *Power Gating* para o nível ESL, e a implementação SPVR_*Timed*_TLM_PG contendo a técnica de *Power Gating* no nível ESL.

6.4.3. Resultados e Análises

Em todas as simulações, os resultados previstos foram alcançados. Sempre que foram utilizados o arquivo de áudio e o dicionário do mesmo locutor, a hipótese do sistema foi "aceito", utilizando o dicionário de outro locutor, a hipótese foi de "rejeitado". Na Figura 6.8, é apresentado o diagrama do comportamento do submódulo PE de uma das simulações. Neste diagrama, é possível observar os períodos que o PE permanece ativado e desativado em relação ao tempo. De acordo com o arquivo de *log*, o submódulo PE permaneceu aproximadamente 50% do tempo simulado no estado de desligado. Foi gasto o tempo médio de 7.8 s com cada simulação.

Figura 6.8 - Comportamento do submódulo PE em função do tempo.



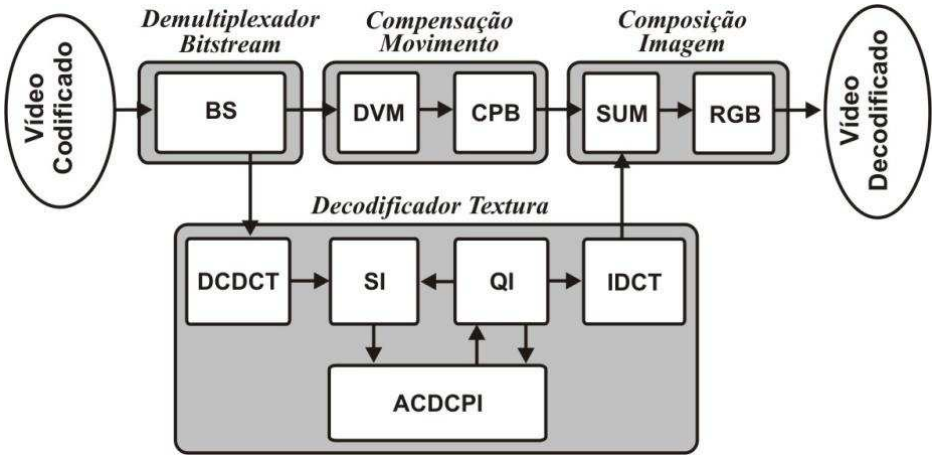
Ao final da 2ª etapa de simulações, o SPVR_*Untimed*_TLM foi verificado com sucesso utilizando as duas versões do simulador SystemC, tanto com a versão original quanto com a versão modificada. Este resultado confirma a transparência das modificações aplicadas ao simulador. Após a simulação da 3ª etapa, diante do tamanho e complexidade do *log* da simulação, foi implementado um *script* para filtrar as informações relevantes ao bloco PE e traçado um diagrama do seu comportamento em função do tempo, com base nos dados.

6.5. Estudo de Caso 4 - Decodificador MPEG-4, nos níveis ESL e RTL

6.5.1. Descrição do Projeto

Este estudo de caso foi baseado no decodificador MPEG-4 *Simple Profile Level* 0, com aproximadamente 21.000 linha de código RTL. Este decodificador foi implementado em silício, com 22,7mm^2 de área, tecnologia CMOS 0.35µm com quatro níveis de metal, frequência de operação de 25MHz para decodificar 30 quadros por segundo (ROCHA *et al.*, 2006). Na Figura 6.9, é apresentado o diagrama em blocos do decodificador MPEG-4.

Figura 6.9 - Esquemático do decodificador MPEG-4.



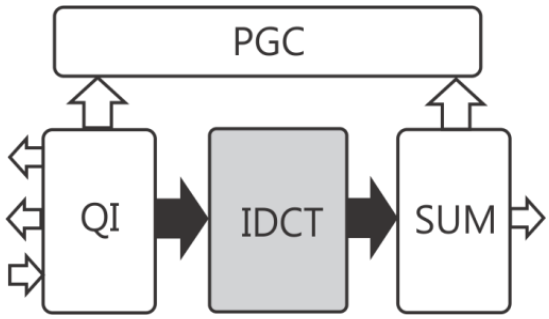
O decodificador foi subdividido em 10 módulos funcionais: *Bitstream* (BS), Decodificação de Vetor de Movimento (DVM), Copiador de Blocos de Predição (CPB), Somador (SUM), Conversor YCbCr para RGB (RGB), Decodificação de Coeficientes DCT (DCDCT), *Scan* Inverso (SI), Quantização Inversa (QI), AC e DC Predição para Intra (ACDCPI), Inversa da DCT (IDCT). Esses módulos foram agrupados em 4 grupos: demultiplexação do *bitstream*, compensação de movimento, decodificação da textura e composição da imagem.

Foi utilizado o software XVID (TEAM X, 2003), como modelo de referência, descrito na linguagem C.

6.5.2. Metodologia

A técnica de *Power Gating* foi aplicada ao submódulo IDCT nos níveis RTL e ESL. Este submódulo foi escolhido por apresentar durante o seu funcionamento, períodos de tempo sem atividade e por ser o módulo que necessita mais área de células lógicas no chip. Foi adicionado o submódulo PGC no decodificador, descrito em SystemC *Timed*-TLM e RTL, com a função de chamar os novos métodos do SystemC-LP. Na Figura 6.10, é apresentada uma parte do diagrama do MPEG-4 com o módulo PGC. Este módulo recebe informações dos submódulos QI e SUM por meio de SC_FIFO-LP. As FIFO's que conectam os submódulos QI com IDCT e IDCT com SUM, foram substituídas pela FIFO-LP. O PGC funciona da seguinte forma: ele irá ativar a IDCT sempre que o QI tiver transações para serem enviadas à IDCT e que o SUM estiver apto a receber transações da IDCT. Caso contrário a IDCT permanecerá desativada.

Figura 6.10 - Conexões do submódulo IDCT e PGC.



As simulações foram executadas em 6 etapas. Cada simulação foi executada até alcançar 100% de cobertura funcional conforme especificado no Plano de Verificação do projeto (ROCHA *et al.*, 2006). As etapas são descritas a seguir:

1. **SystemC + MPEG-4_Untimed_TLM**: Nesta etapa, foram utilizados o simulador original OSCI SystemC V2.2.0 e a implementação do MPEG-4_*Untimed*_TLM;
2. **SystemC-LP + MPEG-4_Untimed_TLM**: Nesta etapa, foram utilizados o simulador SystemC com suporte a *Power Gating* para o nível ESL e a implementação do MPEG-4_*Untimed*_TLM;

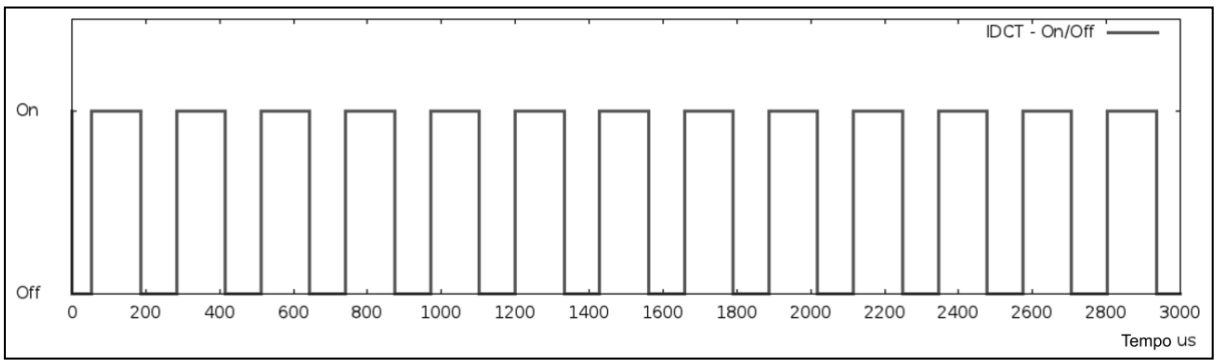
3. **SystemC-LP + MPEG-4_Timed_TLM_PG**: Nesta etapa, foram utilizados o simulador SystemC com suporte a *Power Gating* para o nível ESL, e a implementação MPEG-4_*Timed*_TLM_PG contendo a técnica de *Power Gating* no nível ESL;
4. **SystemC + MPEG-4_RTL**: Nesta etapa, foram utilizados o simulador original OSCI SystemC V2.2.0 e a implementação do MPEG-4_RTL;
5. **SystemC-LP + MPEG-4_RTL**: Nesta etapa foram utilizados o simulador SC-LP_V3 e a implementação do MPEG-4_RTL;
6. **SystemC-LP + MPEG-4_RTL_PG**: Nesta etapa foram utilizados o simulador SC-LP_V3, e a implementação MPEG-4_RTL_PG contendo a técnica de *Power Gating* no nível ESL.

6.5.3. Resultados e Análises

A transparência das novas funções do simulador foi reafirmada ao final da 2ª etapa de simulações. O MPEG_*Untimed*_TLM foi verificado com sucesso utilizando as duas versões do simulador SystemC, tanto com a versão original quanto com a versão modificada.

Após a simulação da 3ª etapa, diante do tamanho e complexidade do *log* da simulação, foi implementado um *script* para filtrar as informações relevantes à IDCT e traçar um diagrama do seu comportamento em função do tempo, com base nos dados. Na Figura 6.11 é apresentado o diagrama do comportamento da IDCT.

Figura 6.11 - Comportamento do submódulo IDCT em função do tempo.



A partir do diagrama, é possível observar os períodos em que a IDCT permanece ativada e desativada em relação ao tempo. Com base no arquivo de *log* da simulação, o submódulo IDCT permaneceu desativado por aproximadamente 41% do tempo simulado.

De acordo com as etapas 4ª e 5ª, o decodificador MPEG-4 descrito no nível RTL foi verificado com sucesso utilizando os simuladores SC e SC-LP_V3. A implementação do MPEG-4 contendo a técnica de *Power Gating* no nível RTL também foi verificada com sucesso utilizando o simulador SC-LP_V3 na 6ª etapa.

Ao avaliar o desempenho do simulador, foram encontrados resultados satisfatórios. Na Tabela 6.1, são apresentados os valores com as médias dos tempos das simulações e são correlacionados os simuladores SystemC original e SC-LP_V3, com as implementações do decodificador MPEG-4.

Tabela 6.1 - Desempenho dos simuladores.

Nº	Simulador	DUV	Tempo (s)
1	SC	MPEG-4_*Untimed*_TLM	6,86
2	SC-LP_V3	MPEG-4_*Untimed*_TLM	6,87
3	SC-LP_V3	MPEG-4_*Timed*_TLM_PG	7,26
4	SC	MPEG-4_RTL	466
5	SC-LP_V3	MPEG-4_RTL	503
6	SC-LP_V3	MPEG-4_RTL_PG	496

É possível observar a partir da Tabela 6.1 que, para as verificações do MPEG-4_*Untimed*_TLM, usando o simulador SystemC original e SC-LP_V3, o simulador apresentou um aumento de 0,2% no tempo de simulação e simulando MPEG_*Timed*_TLM_PG houve um aumento de 5,5%. A perda de desempenho do simulador no nível ESL é aceitável em relação ao tempo de simulação no nível RTL.

No nível RTL, comparando as verificações **4ª** e **5ª**, o simulador teve uma perda de desempenho de aproximadamente de 8%. Comparando as verificações **5ª** e **6ª**, é possível observar que a verificação **6ª** ocorreu em um tempo menor do que a **5ª**, isto é devido à não execução do módulo IDCT quando está desativado. Diante deste fato, dependendo da forma que *Power Gating* for aplicado ao sistema, é possível que o simulador SC_LP-V3 tenha o desempenho equivalente ou até melhor comparado ao simulador original no nível RTL.

6.6. Discussões Finais do Capítulo

A técnica de *Power Gating* é bastante aplicada em projetos, e diante da tendência da tecnologia CMOS, *Power Gating* é uma forte candidata a ser aplicada em futuros projetos utilizando tecnologia nano-CMOS. Os estudos de caso apresentados demonstram a efetiva aplicação, simulação e verificação da técnica de *Power Gating* nos níveis ESL e RTL utilizando SystemC. Esses resultados também demonstram que, utilizando o simulador SC_LP-V3, é possível confrontar os modelos nos níveis ESL e RTL, com o objetivo de comparar se os modelos são logicamente equivalentes.

Capítulo 7 - Considerações Finais

Dispositivos eletrônicos estão cada vez mais complexos e restrições sobre o consumo de energia são impostos pelo mercado consumidor. A indústria tem investido no desenvolvimento de chips com baixo consumo de energia e vem se deparando com problemas com a tecnologia nano-CMOS, principalmente com relação ao consumo de potência estática dos chips.

Existe um consenso na literatura sobre as vantagens de iniciar a verificação funcional no nível ESL. Com relação ao consumo de energia, existe um conjunto de técnicas que podem ser aplicadas em diversas fases do fluxo de desenvolvimento com objetivo de reduzir o consumo, por exemplo, a técnica de *Power Gating*. Esta técnica é eficiente na redução do consumo de energia total e sua aplicação é invasiva. Este fato restringe a sua adoção pela indústria, pois o processo de verificação funcional da técnica está atualmente restrito ao nível RTL.

7.1. Contribuições

Neste trabalho foi apresentado um simulador SystemC, que foi modificado para suportar a simulação de projetos que contém a técnica de *Power Gating,* descrito no nível ESL e RTL, com base em SystemC. Diante dessa característica, é possível realizar a verificação funcional de um IP contendo a técnica de *Power Gating* no início do fluxo de desenvolvimento, no nível ESL, utilizando SystemC TLM-*Timed* e posteriormente, confrontar com os resultados no nível RTL.

Outra contribuição deste trabalho reside na etapa do projeto arquitetural, pois é possível realizar a análise arquitetural do projeto contendo a técnica de *Power Gating* e estudar seu impacto na sua aplicação ao sistema. Pois, é possível distribuir as funcionalidades do sistema entre os módulos funcionais, da melhor forma possível, com o objetivo de maximizar a economia de energia, por meio da técnica de *Power Gating*. O tempo gasto com as simulações dos projetos no nível ESL foram inferiores ao tempo gasto com o mesmo projeto em RTL. O acréscimo de tempo nas simulações devido ao uso das novas funcionalidades do simulador nos níveis ESL e RTL, pode ser considerado aceitável diante dos benefícios alcançados.

7.2. Sugestões para Trabalhos Futuros

- Estudar a viabilidade de adaptar o simulador SystemC-LP-V3, para que, em conjunto com a biblioteca AMS (*Analog/Mixed-Signal*) (ACELLERRA, 2012), seja possível simular projetos analógicos e sinal-misto contendo a técnica de *Power Gating*.

- Adicionar ao simulador SystemC-LP-V3 uma forma de estimar o consumo de energia no nível ESL.

- Adicionar novas funcionalidades ao simulador para suportar a simulação de outras técnicas para redução do consumo de energia.

Referências Bibliográficas

ACELLERRA. **Open SystemC Initiative**. Disponível em: <www.accellera.org>. Acesso em: 17 abr. 2012.

ALLEN, D. **Power Formats: You Can Have It Your Way**. Disponível em: <http://electronicdesign.com/content/topic/power-formats-you-can-have-it-your-way18420/catpath/eda>. Acesso em: 17 abr. 2012.

ANDERSON, T. L. **SoC low-power verification requires a full-chip solution**. Disponível em: <http://www.eetimes.com/electronics-blogs/other/4370947/SoC-low-power-verification-requires-a-full-chip-solution>. Acesso em: 23 abr. 2012.

BAILEY, B.; MARTIN, G. **ESL Models and their Application: Electronic System Level Design and Verification in Practice**. Boston, MA: Springer US, 2010.

BAILEY, S. et al. **To Retain or Not to Retain: How do I Verify the State Elements of my Low Power Design?**DVCon '08. **Anais**...2008

BEMBARON, F.; KAKKAR, S.; MUKHERJEE, R. **Low Power Verification Methodology Using UPF**. San Jose, CA: [s.n.]. p. 228-233

BERGERON, J. **Writing testbenches: functional verification of HDL models Second Edition**. Second Ed. ed. [S.l.] Kluwer Academic, 2003.

BERGERON, J. **Writing Testbenches using System Verilog**. Boston, MA: Springer US, 2006.

BERGERON, J. et al. **Verification methodology manual for SystemVerilog**. New York, NY, USA: Springer, 2006.

BLACK, D. C. et al. **SystemC: From the Ground Up**. Second Edi ed. Boston, MA: Springer US, 2010.

BOMBIERI, N.; FUMMI, F.; GUARNIERI, V. **Accelerating RTL Fault Simulation through RTL-to-TLM Abstraction**2011 Sixteenth IEEE European Test Symposium. **Anais**...IEEE, maio. 2011

BRITO, A. V. et al. **Modelling and Simulation of Dynamic and Partially Reconfigurable Systems using SystemC**IEEE Computer Society Annual Symposium on VLSI (ISVLSI '07). **Anais**...IEEE, mar. 2007

BRITO, A. V.; MELCHER, E. U. K.; ROSAS, W. **An open-source tool for simulation of partially reconfigurable systems using SystemC**IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures (ISVLSI'06). **Anais**...IEEE, 2006

BRITO, A. V.; SILVEIRA, G. S.; UWE, E. M. K. A Methodology for Modelling and Simulation of Dynamic and Partially Reconfigurable Systems. In: **Dynamic Modelling**. Croatia: INTECH, 2010. p. 29-48.

BUTZEN, P. F. et al. **Subthreshold and Gate Leakage Estimation in Complex Gates**International Workshop on Logic & Synthesis. **Anais**...Lake Tahoe, CA, USA: 2008

BUTZEN, P. F. et al. Standby power consumption estimation by interacting leakage current mechanisms in nanoscaled CMOS digital circuits. **Microelectronics Journal**, v. 41, n. 4, p. 247-255, abr. 2010.

CADENCE. **Cadence - Low-Power Solution**. Disponível em: <http://www.cadence.com/solutions/lp/pages/default.aspx>. Acesso em: 25 abr. 2012.

CHABINI, N. **A Heuristic for Reducing Dynamic Power Dissipation in Clocked Sequential Designs**17th International Workshop on Power and Timing Modeling, Optimization and Simulation, PATMOS 2007. **Anais**...Springer-Verlag Berlin Heidelberg, 2007

CHEN, S.; LIN, J. **Experiences of low power design implementation and verification**Proceedings of the 2008 Asia and South Pacific Design Automation Conference. **Anais**...IEEE, jan. 2008

CHEN, S.; LIN, J. **Implementation and verification practices of DVFS and power gating**2009 International Symposium on VLSI Design, Automation and Test. **Anais**...IEEE, abr. 2009

CHEN, S.; LIN, R.; TUNG, H. **Power gating design for standard-cell-like structured ASICs**Design, Automation & Test. **Anais**...2010

CORMEN, T. H. et al. **Algoritmos: Teoria e Prática**. 2ª. ed. Rio de Janeiro: Campus, 2002.

COURT, C.; KELLY, P. Loop-Directed Mothballing: Power Gating Execution Units Using Runtime Loop Analysis. **IEEE Micro**, v. 31, n. 6, p. 29-38, nov. 2011.

CROFT, M.; BAILEY, S. **Is Your Low Power Design Switched On?**2007 International Symposium on System-on-Chip. **Anais**...IEEE, nov. 2007

CRONE, A.; CHIDOLUE, G. **Functional Verification of Low Power Designs at RTL** (N. A. L. Svensson, Ed.)17th International Workshop on Power and Timing Modeling, Optimization

and Simulation, PATMOS 2007. **Anais**...Gothenburg, Sweden: Springer Berlin Heidelberg NewYork, 2007

DRECHSLER, R. **Advanced formal verification**. New York, New York, USA: Kluwer Academic Publishers, 2004.

EISNER, C.; NAHIR, A.; YORAV, K. Functional verification of power gated designs by compositional reasoning. **Formal Methods in System Design**, v. 35, n. 1, p. 40-55, jul. 2009.

EISNER, C. R.; ZICHRON YAACOV; YORAV, K. F. **Functional verification of power gated designs by compositional reasoningComputer Aided Verification**, 2011. Disponível em: <http://www.springerlink.com/index/5816425924173876.pdf>. Acesso em: 26 ago. 2012

FECHINE, J. et al. **SPVR: An IP core for Real-Time Speaker Verification**IP-SOC Conference. **Anais**...Grenoble, France.: 2010

FUJITA, M.; GHOSH, I.; PRASAD, M. **Verification techniques for system-level design**. New York, New York, USA: Elsevier, 2008.

GAMMIE, G. et al. SmartReflex Power and Performance Management Technologies for 90 nm, 65 nm, and 45 nm Mobile Application Processors. **Proceedings of the IEEE**, v. 98, n. 2, p. 144-159, fev. 2010.

GHENASSIA, F. **Transaction-level modeling with Systemc: TLM concepts and applications for embedded systems**. Netherlands: Springer, 2005.

GIRARD, P.; NICOLICI, N.; WEN, X. **Power-Aware Testing and Test Strategies for Low Power Devices**. Boston, MA: Springer Verlag, 2009.

GPROF. **Gprof**. Disponível em: <http://www.gnu.org/software/binutils>. Acesso em: 17 nov. 2010.

GUO, L. et al. A novel performance driven power gating based on distributed sleep transistor network. **Proceedings of the 18th ACM Great Lakes symposium on VLSI - GLSVLSI '08**, p. 255, 2008.

HAZRA, A. et al. **Leveraging UPF-extracted assertions for modeling and formal verification of architectural power intent**Proceedings of the 47th Design Automation Conference on - DAC '10. **Anais**...New York, New York, USA: ACM Press, 2010

HENZLER, S. **Power Management of Digital Circuits in Deep Sub-Micron CMOS Technologies**. Munchen, Germany: Springer, 2007.

HILLMAN, D. **Using Mobilize Power Management IP for Dynamic & Static Power Reduction in SoC at 130 nm**Design, Automation and Test in Europe. **Anais**...IEEE, 2005

HOANG, T. T. et al. **Power gating multiplier of embedded processor datapath**2011 7th Conference on Ph.D. Research in Microelectronics and Electronics. **Anais**...IEEE, jul. 2011

HSIEH, T.-W. et al. **Energy-effective design & implementation of an embedded VLIW DSP**2008 International SoC Design Conference. **Anais**...IEEE, nov. 2008

IEEE-SA. **IEEE Std 1801-2009: IEEE Standard for Design and Verification of Low Power Integrated Circuits**. New York, NY, USA: The Institute of Electrical and Electronics Engineers, Inc., 2009.

ITRS. **International Technology Roadmap for Semiconductors 2009 Edition: DesingDesign**, 2009.

ITRS. **International Technology Roadmap for Semiconductors**. Disponível em: <http://public.itrs.net>. Acesso em: 27 out. 2010.

JADCHERLA, S. et al. **Verification Methodology Manual for Low Power**. Mountain View, CA, USA: Synopsys, 2009.

JIANG, C. et al. Dynamic voltage/frequency scaling for power reduction in data centers: Enough or not? **2009 ISECS International Colloquium on Computing, Communication, Control, and Management**, p. 428-431, ago. 2009.

JIANG, H.; MAREK-SADOWSKA, M.; NASSIF, S. R. **Benefits and costs of power-gating technique**2005 International Conference on Computer Design. **Anais**...IEEE Comput. Soc, 2005

JOSEPH, N.; S., S.; SANKARAPANDIAMMAL, K. FPGA Based Implementation of High Performance Architectural Level Low Power 32-bit RISC Core. **2009 International Conference on Advances in Recent Technologies in Communication and Computing**, p. 53-57, out. 2009.

JUAN, D.-C. et al. An Efficient Wake-Up Strategy Considering Spurious Glitches Phenomenon for Power Gating Designs. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, v. 18, n. 2, p. 246-255, fev. 2010.

KAPOOR, B. et al. **Impact of SoC power management techniques on verification and testing**2009 10th International Symposium on Quality of Electronic Design. **Anais**...IEEE, mar. 2009

KASUYA, A.; TESFAYE, T. Verification Methodologies in a TLM-to-RTL Design Flow. **2007 44th ACM/IEEE Design Automation Conference**, p. 199-204, jun. 2007.

KAWA, J. Low Power and Power Management for CMOS—An EDA Perspective. **IEEE Transactions on Electron Devices**, v. 55, n. 1, p. 186-196, jan. 2008.

KAWANO, T. et al. **Adjacent-State monitoring based fine-grained power-gating scheme for a low-power asynchronous pipelined system**2011 IEEE International Symposium of Circuits and Systems (ISCAS). **Anais**...IEEE, maio. 2011

KEATING, M. et al. **Low Power Methodology Manual: For System-on-Chip Design**. New York, NY: Springer, 2007.

KIM, J.; KIM, S.; BAEK, K.-H. A Low Power SOC Architecture for the V2.0+EDR Bluetooth Using a Unified Verification Platform. **IEICE Transactions on Information and Systems**, v. E93-D, n. 9, p. 2500-2508, 2010.

KIM, K. K.; KIM, Y.; CHOI, K. Hybrid CMOS and CNFET Power Gating in Ultralow Voltage Design. **IEEE Transactions on Nanotechnology**, v. 10, n. 6, p. 1439-1448, nov. 2011.

KISSLER, D. et al. Scalable Many-Domain Power Gating in Coarse-Grained Reconfigurable Processor Arrays. **IEEE Embedded Systems Letters**, v. 3, n. 2, p. 58-61, jun. 2011.

KITAHARA, T. et al. **Area-Efficient Selective Multi-Threshold CMOS Design Methodology for Standby Leakage Power Reduction**Design, Automation and Test in Europe. **Anais**...IEEE, 2005

KUMAR, J. A.; VASUDEVAN, S. Verifying dynamic power management schemes using statistical model checking. **17th Asia and South Pacific Design Automation Conference**, p. 579-584, jan. 2012.

KUWAHARA, M. **Design and verification methods of toshiba's wireless LAN baseband SoC**2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC). **Anais**...IEEE, jan. 2010

LAKSHMI, M. S.; VAYA, P.; VENKATARAMANAN, S. **Power management in SoC using CPF**2011 3rd International Conference on Electronics Computer Technology. **Anais**...IEEE, abr. 2011

LEBRETON, H.; VIVET, P. **Power Modeling in SystemC at Transaction Level, Application to a DVFS Architecture**2008 IEEE Computer Society Annual Symposium on VLSI. **Anais**...IEEE, 2008

LEE, K.; CHANG, C. **EPIDETOX: an ESL platform for integrated circuit design and tool exploration**Proceedings of the 9th International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS). **Anais**...2011

LEE, M.-C. et al. **An efficient wakeup scheduling considering resource constraint for sensor-based power gating designs**Proceedings of the 2009 International Conference on Computer-Aided Design - ICCAD '09. **Anais**...New York, New York, USA: ACM Press, 2009

LIU, J. et al. **Low Power Verification with UPF: Principle and Practice**Proceedings of Design and Verification Conference (DvCon 2010). **Anais**...San Jose, CA: 2010

Liberty Library Synopsys. Disponível em: <http://www.synopsys.com/>. Acesso em: 6 fev. 2011.

MATHUR, A. **Sequential Transformations for Low Power and CPF**. Disponível em: <http://www.si2.org/events_dir/2008/lpc2008/calypto.pdf>. Acesso em: 6 fev. 2011.

MATSUTANI, H. et al. **Ultra Fine-Grained Run-Time Power Gating of On-chip Routers for CMPs**2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip. **Anais**...IEEE, maio. 2010

MATTOS, J.; JR., L. R.; PILLA, M. **Desafios e Avanços em Computação - o estado da arte**. Pelotas, RS: Ed. da Universidade Federal de Pelotas, 2009. p. 260

MELCHER, E. et al. **ASIC Implementation of a Speech Detector IP-Core for Real-Time Speaker Verification**IP-SOC Conference. **Anais**...Grenoble, France: 2011

MENTOR GRAPHIC'S. **Mentor Graphics Low Power Solutions**. Disponível em: <http://www.mentor.com/solutions/low-power/functional-verification>. Acesso em: 25 abr. 2012.

MEYER, A. **Principles of functional verification**. USA: Elsevier, 2004.

MOHANTY, S. P. et al. **Low-Power High-Level Synthesis for Nanoscale CMOS Circuits**. Boston, MA: Springer US, 2008.

MORIMOTO, C. **Radeon HD 7970: 28 nm, 4.3 bilhões de transístores e nova arquitetura**. Disponível em: <http://www.hardware.com.br/artigos/radeonhd7970/>. Acesso em: 1 jul. 2012.

PAKBAZNIA, E.; PEDRAM, M. Design of a Tri-Modal Multi-Threshold CMOS Switch With Application to Data Retentive Power Gating. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, v. 20, n. 2, p. 380-385, fev. 2012.

PASHA, M. A.; DERRIEN, S.; SENTIEYS, O. **System Level Synthesis for Ultra Low-Power Wireless Sensor Nodes**2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools. **Anais**...IEEE, set. 2010

PEDRAM, M. Power minimization in IC design: principles and applications. **ACM Transactions on Design Automation of Electronic Systems**, v. 1, n. 1, p. 3-56, jan. 1996.

PERON, R. D. V. **Otimização de código fonte C para o processador embarcado Nios II**. São Paulo, SP: Universidade de São Paulo, 2007.

PFI, P. F. I. **A Practical Guide to Low-Power Design: User Experience with CPF**. California, USA: Power Forward Initiative, 2009.

PINTO, R. R. M. **Power Management Verification – an evolving discipline**Ninth International Workshop on Microprocessor Test and Verification. **Anais**...2008

PIZIALI, A. **Functional verification coverage measurement and analysis**. Dordrecht: Kluwer Academic, 2004.

RASHINKAR, P.; PATERSON, P.; SINGH, L. **System-on-a-chip verification: methodology and techniques**. New York, NY, USA: Kluwer Academic, 2002.

ROCHA, A. et al. **Silicon validated ip cores designed by the brazil-ip network**IP/SOC. **Anais**...2006

ROY, S.; RANGANATHAN, N.; KATKOORI, S. State-Retentive Power Gating of Register Files in Multicore Processors Featuring Multithreaded In-Order Cores. **IEEE Transactions on Computers**, v. 60, n. 11, p. 1547-1560, nov. 2011.

SAINI, R.; SINGH, V. **Low power verification flow to ease the pain in implementing MTCMOS based MSMV wireless design**CDNLive ! **Anais**...2007

SARKAR, A.; LIN, S.; WANG, K. **A methodology for analysis and verification of power gated circuits with correlated results**Proceedings of the 2007 international symposium on Low power electronics and design - ISLPED '07. **Anais**...New York, New York, USA: ACM Press, 2007

SATHANUR, A.; CALIMERA, A. et al. **On quantifying the figures of merit of power-gating for leakage power minimization in nanometer CMOS circuits**2008 IEEE International Symposium on Circuits and Systems. **Anais**...IEEE, maio. 2008

SATHANUR, A. et al. **Timing-driven row-based power gating**Proceedings of the 2007 international symposium on Low power electronics and design - ISLPED '07. **Anais**...New York, New York, USA: ACM Press, 2007

SATHANUR, A.; BENINI, L. et al. Multiple power-gating domain (multi-VGND) architecture for improved leakage power reduction. **Proceeding of the thirteenth international symposium on Low power electronics and design - ISLPED '08**, p. 51, 2008.

SENGUPTA, D.; SALEH, R. **Application-driven floorplan-aware voltage island design**Proceedings of the 45th annual conference on Design automation - DAC '08. **Anais**...New York, New York, USA: ACM Press, 2008

SHIBATA, S. et al. Efficient Design Space Exploration at System Level with Automatic Profiler Instrumentation. **IPSJ Transactions on System LSI Design Methodology**, v. 3, p. 179-193, 2010.

SI2. **Si2 Common Power Format Specification ™**. 1.1. ed. Austin: Silicon Integration Initiative, Inc., 2008.

SI2. **Silicon Integration Initiative**. Disponível em: <http://www.si2.org/>. Acesso em: 9 nov. 2010.

SILVA, K. R. G. DA et al. A methodology aimed at better integration of functional verification and RTL design. **Design Automation for Embedded Systems**, v. 10, n. 4, p. 285-298, dez. 2005.

SILVA, K. R. G. DA. **Uma Metodologia de Verificação Funcional para Circuitos Digitais**. Campina Grande, PB: Universidade Federal de Campina Grande, 2007.

SILVEIRA, G. S.; BRITO, A. V.; MELCHER, E. U. K. **Functional verification of power gate design in SystemC RTL**Proceedings of the 22nd Annual Symposium on Integrated Circuits and System Design Chip on the Dunes - SBCCI '09. **Anais**...New York, New York, USA: ACM Press, 2009

SINGH, A. **Understanding RTL power reduction techniques**. Disponível em: <http://www.scdsource.com/article.php?id=50>. Acesso em: 25 set. 2008.

SINGH, R. et al. **A three-step power-gating turn-on technique for controlling ground bounce noise**Proceedings of the 16th ACM/IEEE international symposium on Low power electronics and design - ISLPED '10. **Anais**...New York, New York, USA: ACM Press, 2010

SOFER, S.; BERKOVITZ, A.; NEIMAN, V. **High Coverage Power Integrity Verification in PSO Domains Employing Distributed PSO Switches**2011 12th International Workshop on Microprocessor Test and Verification. **Anais**...IEEE, dez. 2011

STAN, M. R. et al. Interaction of scaling trends in processor architecture and cooling. **2010 26th Annual IEEE Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM)**, p. 198-204, fev. 2010.

SWAN, S. **SystemC transaction level models and RTL verification**2006 43rd ACM/IEEE Design Automation Conference. **Anais**...IEEE, 2006

SYNOPSYS. **Synopsys - Eclypse™ Low Power Solution**. Disponível em: <http://www.synopsys.com/Solutions/EndSolutions/EclypseSolutions/Pages/default.aspx>. Acesso em: 25 abr. 2012.

TADA, A.; NOTANI, H.; NUMA, M. A novel power gating scheme with charge recycling. **IEICE Electronics Express**, v. 3, n. 12, p. 281-286, 2006.

TAKAI, Y.; HASHIMOTO, M.; ONOYE, T. **Power gating implementation for noise mitigation with body-tied triple-well structure**2011 IEEE Custom Integrated Circuits Conference (CICC). **Anais**...IEEE, set. 2011

TEAM X. **Xvid api 2.1 reference (for 0.9.x Series).** Disponível em: <http://www.xvid.org>. Acesso em: 2 ago. 2010.

UPASANI, G. et al. **Reducing Timing Overhead in Simultaneously Clock-Gated and Power-Gated Designs by Placement-Aware Clustering**19th International Workshop on Power and Timing Modeling, Optimization and Simulation, PATMOS 2009. **Anais**...Delft, The Netherlands: 2009

UPF. **Unified Power Format**. Disponível em: <http://www.unifiedpowerformat.com>. Acesso em: 3 nov. 2010.

VARMA, S. et al. **Low power architecture and design techniques for mobile handset LSI Medity™ M2**. [S.l.] IEEE, 2008. v. 48p. 748-753

VISWANATH, V.; VASUDEVAN, S.; ABRAHAM, J. A. Dedicated Rewriting: Automatic Verification of Low Power Transformations in RTL. **2009 22nd International Conference on VLSI Design**, p. 77-82, jan. 2009.

WIEMANN, A. **Standardized functional verification**. San Carlos, CA, USA: Springer Verlag, 2008.

WILCOX, P. **Professional verification: A Guide to Advanced Functional Verification**. New York, New York, USA: Kluwer Academic Publishers, 2004.

XU, H.; JONE, W.-B.; VEMURI, R. **Accurate Energy Breakeven Time Estimation for Run-time Power Gating**2008 IEEE/ACM International Conference on Computer-Aided Design. **Anais**...IEEE, nov. 2008

XU, T.; LI, P. **Decoupling for power gating: sources of power noise and design strategies**Proceedings of the 48th Design Automation. **Anais**...2011

XU, X.; YONG, H. **Method to Design Low-Power Embedded Processor for Wireless Endoscope**2009 3rd International Conference on Bioinformatics and Biomedical Engineering. **Anais**...IEEE, jun. 2009

YEUNG, P.; CHOI, S. **Advanced Static Verification for SoC Designs**SoC Design Conference (ISOCC), 2009. **Anais**...2009

YONG, L. K.; UNG, C. K. **Power density aware power gate placement optimization scheme**2nd Asia Symposium on Quality Electronic Design (ASQED). **Anais**...IEEE, ago. 2010