PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS E PROCESSOS INDUSTRIAIS - MESTRADO
ÁREA DE CONCENTRAÇÃO EM CONTROLE E OTIMIZAÇÃO DE PROCESSOS INDUSTRIAIS

Lúcio Renê Prade

ESTUDO DE TÉCNICAS DE CODIFICAÇÃO VISANDO A OTIMIZAÇÃO DE CIRCUITOS DIGITAIS: UM ESTUDO DE CASO

Santa Cruz do Sul, junho de 2009

Lúcio Renê Prade

ESTUDO DE TÉCNICAS DE CODIFICAÇÃO VISANDO A OTIMIZAÇÃO DE CIRCUITOS DIGITAIS: UM ESTUDO DE CASO

Dissertação apresentada ao Programa de Pós-Graduação em Sistemas e Processos Industriais - Mestrado, Área de concentração em Controle e Otimização de Processos Industriais, Universidade de Santa Cruz do Sul - UNISC, como requisito parcial para obtenção do título de Mestre em Sistemas e Processos Industriais.

Orientador: Prof. Dr. Rafael Ramos dos Santos

Santa Cruz do Sul, junho de 2009

“O assunto mais importante do mundo pode ser
simplificado até ao ponto em que todos possam apreciá-lo
e compreendê-lo. Isso é - ou deveria ser - a mais elevada forma de arte.”
— SIR CHARLES CHAPLIN

AGRADECIMENTOS

Agradecer a todos que ajudaram a construir esta dissertação não é tarefa fácil. O maior perigo que se coloca para o agradecimento seletivo é não esquecer de mencionar nenhuma pessoa. Então, a meus amigos que, de uma forma ou de outra, contribuíram com sua amizade e com sugestões para a realização deste trabalho, gostaria de expressar minha profunda gratidão.

Não poderia deixar de iniciar as homenagens pela figura mais importante na realização deste trabalho, meu orientador Prof. Dr. Rafael Ramos dos Santos. A quem cabe grande parte do mérito deste trabalho, pela sua paciência, que não foi pouca, pelos conselhos, pelo rumo nos momentos de indecisão, pela sua disponibilidade, dedicação e acima de tudo pela amizade, muito obrigado !

A todo o pessoal do grupo de pesquisa GPSEM, em especial ao Vitor e o Torrel pela ajuda nas implementações e nos testes e ao Ricardo pelas ajudas com os servidores e as ferramentas. Ao pessoal do LSI-TEC Nordeste, que mesmo no pouco tempo de convívio me proporcionou experiencias novas na área de microeletrônica. Ao suporte do NSCAD pela ajuda com os *scripts* das ferramentas e as dúvidas. Ao Josias do Ceitec pelas orientações nas escolhas das ferramentas e tutoriais. E aos demais professores do mestrado por todo o incentivo, convívio e amizade nestes anos de curso. Muito obrigado a todos!

Não poderia deixar de citar meus pais, sem os quais não teria conseguido chegar até aqui, fosse pelo apoio financeiro ou pela preocupação com minha educação desde criança. E para finalizar a minha esposa Adriana, por me compreender, me incentivar, e pelas noites mal dormidas me ajudando a revisar o texto. A Rafa pelo incentivo, os desenhos e o carinho.

Minha esperança é que, compensando o tempo e esforço dispendidos, algumas das idéias apresentadas aqui venham por ajudar em trabalhos futuros e mesmo no dia-a-dia em projetos de ASIC.

RESUMO

Esta dissertação trata do estudo do projeto de um circuito integrado digital de comunicação de dados, mais especificamente um circuito para comunicação de dados em redes Ethernet. Durante o trabalho, realizou-se o estudo do fluxo e das ferramentas CAD utilizadas em cada etapa do projeto, detendo-se nas etapas de codificação em linguagem de descrição de hardware Verilog e nos resultados de síntese lógica. Estas duas etapas foram o foco da dissertação onde levantou-se na bibliografia e na prática, algumas técnicas a nível de codificação em linguagem de hardware para otimização do projeto nos quesitos área, potência dissipada e freqüência máxima de funcionamento. A aplicação de diferentes combinações dessas técnicas ao projeto do controlador Ethernet tido como estudo de caso, tiveram seus resultados de síntese lógica analisados. Estas análises foram feitas levando-se em consideração custos de implementação como tempo e complexidade, ganhos reais em número de portas lógicas, miliwatt e freqüência, além de levar em conta outras vantagens e desvantagens técnicas. Com isso determinou-se um conjunto de técnicas possíveis de se aplicar à codificação em linguagem de descrição de hardware do projeto de um circuito integrado digital, que pode trazer otimização e melhorias nos três quesitos já citados. Estas técnicas podem ser facilmente utilizadas e combinadas com outras em etapas como no roteamento, síntese física ou mesmo no processo de fabricação para o desenvolvimento de projetos com grandes restrições de área, potência e freqüência.

Palavras-chave: Projeto de Circuitos Integrados Digitais, Otimização, Técnicas de Codificação em Linguagem de Descrição de Hardware, Síntese Lógica.

ABSTRACT

This work tackles the study of a data communication integrated circuit, more specifically an integrated circuit for Ethernet data communication. Throughout the work, the design flow and commonly used CAD tools were studied focusing on the hardware description language Verilog and in special the logic synthesis results. Furthermore, the research of techniques for area, power consumption and frequency was conduced looking into the literature and conducing experiments to validate the results. The combination of different techniques to a case study allowed the analysis of the results and a comparison of them in terms of improvements obtained by each one. The analysis was made taking into account the implementation cost in terms of time length and complexity, gate count reduction, power and frequency on top of other pros and cons of each technique. Giving that, the result is a detailed comparison which allows one to choose and apply the techniques in order to obtain the desired product. The techniques can be easily employed and combined with other techniques in lower abstraction levels such as routing, physical synthesis and even in the fabrication process to achieve requirements in terms of area, power and frequency.

Keywords: *Digital Designs of Application Specific Integrated Circuit, Optimization, Hardware Description Language Encoding Techniques, Logical Synthesis.*

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ADK	*Academic Design Kit*
ADMS	Advanced Mentor Simulator
ASIC	Circuito Integrado de Aplicação Específica
CAD	*Computer-aided design*
CI	Circuito Integrado
CIF	Formato de base de dados utilizado pela industria de CIs.
CRC	*Cyclic Redundancy Check*
CSMA/CD	*Carrier Sense Multiple Access with Collision Detection*
DC OP	Análise de Potência Contínua Dissipada
DMA	*Direct Memory Access*
DRC	*Design Rule Check*
EDA	*Electronic design automation*
ERC	*Electrical Rule Check*
ESD	*Electrical Static Discharge*
FCS	*Frame Check Sequence*
FIFO	*First-In-First-Out*
FPGA	*Field-Programmable Gate Array*
FPLD	*Field Programmable Logic Device*
FSM	*Finite State Machine*
GDSII	Formato de base de dados utilizado pela industria de CIs.
GPL	GNU General Public License
HDL	*Hardware Description Language*
IEEE	Institute of Electrical and Electronic Engineers
IOB	*Input-Output Block*
IP Core	*Intellectual Property Core*
IP	*Intellectual Property*
ISO	International Standards Organization

LAN	*Local Area Network*
LE	*Logic Element*
LSB	*Least Significant Bit*
LVS	*Layout Versus Esquemático*
MAC	*Medium Access Control*
MCT	Ministério da Ciência e Tecnologia
MDC	Minimo Divisor Comum
MHz	Megahertz
MII	*Media Independent Interfece*
MUX	Multiplexador
OSI	Open Systems Interconnect
PHY	Canal de conexão da camadade enlace ao meio fisico
PLI	*Programming Language Interface*
PNM	Programa Nacional de Microeletrônica
RAM	*Random Access Memory*
RTL	*Register Transfer Level*
SMD	*Surface Mounted Device*
TSMC	Taiwan Semiconductor Manufacturing Company Limited
VHDL	*VHSIC Hardware Description Language*
VHSIC	*Very High Speed Integrated Circuit*
VLSI	*Very Large Scale Integration*

SUMÁRIO

INTRODUÇÃO

Nas últimas décadas, tecnologias de semicondutores foram responsáveis por enormes progressos tecnológicos no mundo, período em que a indústria de semicondutores cresceu a uma taxa média da ordem de 16%, contra aproximadamente 4% da economia em geral. Os avanços na área de microeletrônica permitiram agregação de valor em toda a cadeia produtiva de praticamente todos os segmentos industriais. Para um determinado país, o significado de possuir competência tecnológica e empresarial em projeto e fabricação de circuitos integrados, assim como em aplicações da microeletrônica em produtos, é sua inclusão no cenário mundial da microeletrônica e o conseqüente desenvolvimento tecnológico, econômico e social (TECNOLOGIA 2003).

No Brasil, foi lançado em 2001, pelo Ministério da Ciência e Tecnologia (MCT), o Programa Nacional de Microeletrônica (PNM) (TECNOLOGIA 2001) com o objetivo de estabelecer um conjunto de ações estratégicas para o desenvolvimento da microeletrônica no país. Em 2003 foram estabelecidas as Diretrizes de Política Industrial, Tecnológica e de Comércio Exterior que destacam semicondutores e software como sendo indústrias estratégicas e prioritárias para o desenvolvimento tecnológico nacional.

Motivado por estes incentivos, pela falta de profissionais especializados no projeto e design de circuitos integrados no Brasil, a instalação de novos centros de desenvolvimento e fabricação de circuitos integrado (BAMPI 2005), os constantes avanços tecnológicos e a crescente demanda de profissionais na pesquisa ligada a micro e nano eletrônica, optou-se por realizar o estudo e o projeto de um circuito integrado digital, mais especificamente na área de comunicação de dados, uma das áreas que mais vem crescendo nos últimos tempos mesmo com a desaceleração atual da economia (ECONÔMICO 2009). Além dos motivos citados acima, também serviu de motivação as constantes exigências do mercado por circuitos cada vez mais otimizados, sendo assim buscou-se entender e medir o impacto do estilo de codificação da linguagem de descrição de hardware Verilog no resultado de síntese do projeto, estudando técnicas e demonstrando seus resultados,

que poderão ser aplicadas para a otimização em novos projetos de circuitos integrados.

Escopo do Trabalho

Este trabalho apresenta os passos para a concepção de um circuito integrado digital. Iniciando pelo projeto (*design*) e verificação, onde foram aprofundados os estudos, o processo de fabricação, os testes pós fabricação e finalmente o encapsulamento. Para tanto foi realizado um estudo de caso que utilizará o MAC (*Media Access Control*) de uma rede Ethernet. A escolha deste padrão justifica-se devido ao fato de 90 % das redes utilizarem este mecanismo e ao fato de possíveis utilizações em outros projetos que necessitem interfaces de comunicação (TANENBAUM 2003), (IEEE 2002).

O trabalho utilizou o processo de desenvolvimento de circuitos integrados, acompanhando a metodologia utilizada a nível industrial, baseando-se nos principais fluxos e sugerindo melhorias em alguns pontos do processo do projeto mais especificamente no processo de codificação em linguagem HDL .

Objetivos

Objetivo Principal

Estudo da metodologia, das ferramentas de projeto e implementação de circuitos integrados utilizados atualmente pela indústria eletrônica; desenvolver o projeto de um circuito integrado como estudo de caso, passando pelos principais passos utilizados na indústria; conhecer por completo o processo; aprimorar fluxos de projeto e estudar técnicas de codificação em linguagem de descrição de hardware Verilog, que visam a otimização do processo de síntese lógica, medindo o impacto que estas técnicas causam no resultado da síntese.

Objetivos Secundários

- Estudo da linguagem de descrição de hardware Verilog a um nível mais avançado;
- Estudo do padrão Ethernet - IEEE 802.3;

- Domínio de ferramentas CAD para microeletrônica;
- Conhecimento do processo de fabricação de circuitos integrados

Organização

O trabalho foi dividido em cinco capítulos para uma melhor organização. No primeiro capítulo é feita uma revisão bibliográfica dos conceitos envolvidos no projeto de circuitos digitais, fluxos e informações peculiares sobre projeto.

No segundo capítulo são demonstradas algumas das principais técnicas de codificação de linguagem de descrição de hardware encontradas na bibliografia.

O terceiro capítulo caracteriza o estudo de caso, sobre o qual foi realizado a aplicação das técnicas de codificação.

No quarto capítulo são demonstrados e discutidos os resultados estimados para potência, área e restrições de tempo resultantes da aplicação das técnicas individualmente e agrupadas em combinações.

No quinto capítulo são apontadas as principais considerações e conclusões que puderam ser observadas nos resultados deste trabalho, bem como as possíveis linhas para dar continuidade e maior abrangência ao estudo.

Na seção de anexos estão ilustrados para um melhor entendimento, alguns códigos VHDL e Verilog que foram utilizados no decorrer do trabalho e também algumas publicações produzidas durante o desenvolvimento do mesmo.

1 FLUXO DE PROJETOS PARA CIRCUITOS DIGITAIS

No decorrer deste capítulo realizou-se uma revisão bibliográfica dos conceitos envolvidos no projeto de circuitos integrados digitais, fluxos e informações peculiares sobre o circuito a ser projetado. O termo fluxo é utilizado para circunscrever as várias fases do projeto de um circuito integrado. Faz-se necessário para dar início a um projeto, traçar as especificações das funções e das condições de operação do circuito.

A relevância desta etapa jamais deve ser subestimada, pois uma especificação mal realizada poderia implicar em uma reestruturação, com apreciável aumento do custo global do produto final (WEST e ESHRAGHIAN 1993).

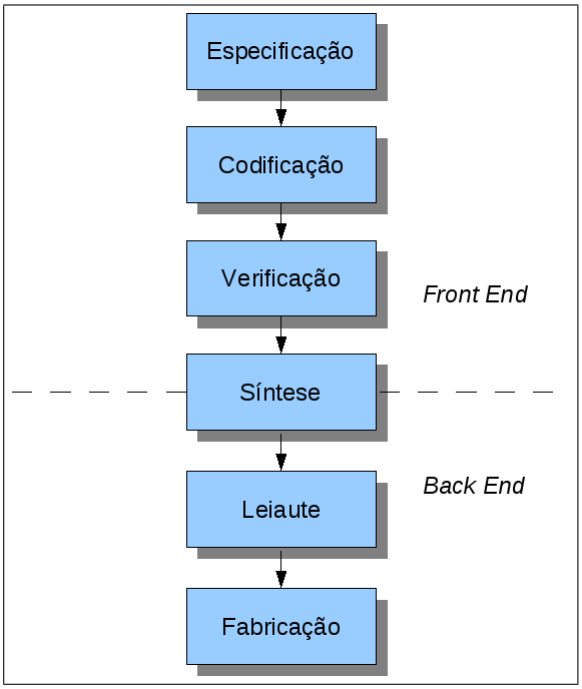

Figura 1: Ciclo de projeto de um circuito integrado digital (WANG 2006)

A ilustração do ciclo de projeto de forma simplificada vista na Figura 1, dá-se ao acolher um modelo seqüencial de projeto *top-down*, no qual cada etapa é completamente executada,

passando-se à etapa subseqüente. A terminologia traduzida foi conjugada de diversas fontes, não retrata exatamente nenhuma visão particular ou comercial.

Baseando-se numa dada especificação, o projetista forma uma idéia geral com relação a solução para o problema. Neste estágio um diagrama de blocos é gerado para representar o projeto, geralmente, ferramentas de CAD (Computer-aided design) não são necessárias nesta fase. Este estágio costuma ser definido com o conceito do projeto. Esta etapa é de vital importância para todo o projeto, consome muito tempo, e exige importantes decisões técnicas e gerenciais. O sistema deve ser especificado em suas interfaces, protocolos, opções de arquitetura, desempenho, além de outros fatores mercadológicos e gerenciais (WEST e ESHRAGHIAN 1993).

Tendo-se a idéia geral definida, os próximos passos para a continuação do projeto são:

- Projeto Funcional - Visa obter a descrição comportamental abstrata de um sistema que funcione de acordo com as especificações. Esta etapa pode considerar alguns aspectos estruturais ou físicos que influenciem o comportamento, principalmente através de estimativas, mas deve, preferencialmente, preocupar-se com a funcionalidade. Normalmente a especificação do sistema em linguagem humana será traduzida para descrições comportamentais de hardware como VHDL, Verilog ou SystemC, e diagramas de tempo (SEDRA e SMITH 2004).

- Projeto Arquitetural - Tendo-se o comportamento de todo o sistema, escolhe-se as opções arquiteturais para sua subdivisão, e também as de implementação interna de cada subdivisão. Aqui consideram-se estruturas como processadores, memórias, circuitos dedicados, e suas microarquiteturas internas como tipos de cache, uso de pipeline, etc. Um dos modelos mais gerais de arquitetura de um sistema é como um conjunto de máquinas que se comunicam. Cada uma é formada por uma parte operativa (*Data Path*) e uma parte de controle (PC), sendo a PC uma máquina de estados finitos (FSM) que gera sinais de controle para fluxo e processamento dos dados na parte operativa (SEDRA e SMITH 2004).

- Projeto Lógico - Nesta etapa refina-se o sistema estruturalmente. Módulos definidos na arquitetura são detalhados, e funções no domínio comportamental são traduzidas para o nível lógico estrutural. Usa-se portas lógicas, registradores, multiplexadores, pequenos macro módulos de biblioteca (parametrizáveis ou não), barramentos, etc. Estes elementos devem ser selecionados conforme sua disponibilidade na tecnologia de implementação escolhida,

pois não serão mais alterados. Atualmente a transformação do projeto arquitetural para o lógico é realizada através de ferramentas CAD específicas como Leonardo Spectrum da Mentor Graphics e RTL Compiler da empresa Cadence(WESTE 2005).

- Projeto de Leiaute - Também denominado de projeto físico, é o conjunto de passos necessários para, a partir da descrição estrutural do circuito (composto por blocos, portas lógicas ou transistores), sintetizar a descrição geométrica final das máscaras, incluindo o leiaute daqueles que já o tem definido. Nesta etapa é praticamente obrigatório o uso de programas para realização automática das tarefas desejadas, em conseqüência do elevado número de elementos (SEDRA e SMITH 2004).

- Verificação do Leiaute *versus* Esquemático - Depois de se completar a descrição do leiaute, uma verificação leiaute *versus* esquemático (LVS) é realizada. Isto assegura que o leiaute está em conformidade com o esquemático. Nesta etapa do projeto, se necessário, são feitas diversas vezes a verificação LVS e após a verificação DRC (*Design Rule Check*), até que haja total conformidade com o esquemático (WANG 2006).

- Checagem de Regras de Projeto (*Design Rule Check* - DRC) - Este estágio é dependente da tecnologia de processo que se utiliza para fabricar o *chip*. A verificação das regras de projeto assegura que as regras utilizadas no processo de fabricação não são violadas. O leiaute deve ser desenhado baseado nas limitações da regra de projeto. Como já foi dito anteriormente, nesta etapa o projeto comuta entre leiaute, LVS e DRC, até que haja conformidade. Sendo verificadas regras de checagem elétrica(*Electrical Rule Check* - ERC), Leiaute, descargas elétricas estáticas (*Electrical Static Discharge* - ESD) e antena (HURST 1998).

- Preparação da base de dados, geração do CIF e GDSII - (formatos de arquivos contendo informações para a fabricação) - Este é o ultimo estágio antes do processo de fabricação. A maioria das foundryes aceita as submissões nestes formatos. Estes são arquivos que descrevem o layout do circuito integrado. Eles incorporam todos os detalhes necessários para a fabricação do *chip* (HURST 1998).

- Fabricação - O circuito é fabricado por longos processos físicos e químicos a partir do seu leiaute completo, ou usando uma forma simplificada de implementação, dentre as descritas adiante. A fabricação de protótipos, ou sua emulação, é importante para que o componente possa ser testado fisicamente e em conjunto com seu ambiente de operação antes da produção em massa (REIS 2003).

- Empacotamento e Teste - No passado o que era uma simples escolha de um empacotamento físico para o substrato de silício, hoje engloba opções e tarefas bastante complexas, como o uso de MCMs (Multi-Chip Modules), ou algoritmos para interconexão de pinos de um novo empacotamento. Finalmente, o teste para garantir o funcionamento de cada componente após sua fabricação é complexo em termos de hipóteses e cobertura de falhas, geração e aplicação de vetores de testes. Já é um padrão a existência de circuitos de auto teste dentro dos próprios componentes (BIST); ao final desta fase tem-se o circuito pronto para o uso (REIS 2003).

1.1 Linguagem de Descrição de Hardware Verilog

Verilog HDL (IEEE 2005), ou simplesmente Verilog, foi introduzida em 1984 pela Gateway Design Automation, proprietária desta linguagem de descrição de hardware e simulação. Em 1988 foi apresentada pela empresa Synopsys uma ferramenta Verilog baseada não mais apenas na simulação mas que também realizava síntese lógica desta linguagem. Em 1989 com a aquisição da Gateway pela Cadence Design System a popularidade da linguagem Verilog aumentou muito, passando a ser umas das mais utilizadas no projeto de circuitos integrados. A linguagem tem sua raiz sintática na linguagem C e permite que diferentes níveis de abstração sejam misturados num mesmo modelo (ASHENDEN 2002).

Deste modo, o projetista pode definir um modelo de hardware em termos de *switches*, *gates*, RTL ou código comportamental, necessitando dominar apenas uma linguagem para estímulos e para gerar projetos hierárquicos. As mais populares ferramentas de síntese suportam Verilog, tornando-a uma linguagem muito escolhida pelos projetistas. Todos os fabricantes disponibilizam bibliotecas em Verilog para simulação e síntese lógica, o uso de PLI (*Programming Language Interface*) é uma poderosa característica que permite ao usuário escrever em um código C comum e interagir com a estrutura de dados internos do Verilog. Projetistas podem personalizar seu simulador Verilog de acordo com suas necessidades com o uso de PLI (HENNESSY e PATTERSON 2005).

1.2 Ferramentas CAD para Micro e Nanoeletrônica

Do ponto de vista de padrões de fato, há um consenso em enquadrar as ferramentas de EDA nas seguintes etapas:

- Síntese de Alto Nível ou Comportamental - São ferramentas necessárias para gerar a partir de uma descrição comportamental em nível algorítmico uma descrição estrutural em nível de macro componentes. São tarefas típicas da síntese de alto nível a alocação de recursos e o escalonamento de operações. A parte de controle resultante pode ser especificada tanto no domínio comportamental quanto no domínio estrutural em nível lógico, visto que a tradução entre um e outro pode ser feita com facilidade (RABAEY 2003).

- Síntese Lógica - Ferramentas de síntese lógica tem como objetivo tomar estas descrições estruturais ou comportamentais nos níveis RTL ou lógicos, e gerar a estrutura completa do circuito que será implementado, segundo as primitivas estruturais disponíveis na forma de implementação escolhida. São tarefas típicas a decomposição lógica, a otimização, e o mapeamento tecnológico (HACHTEL e SOMENZI 2006).

- Síntese de Leiaute ou Síntese Física - São ferramentas necessárias para a geração do leiaute a partir da estrutura exata do circuito, formado tipicamente por portas lógicas e pela lista de interconexões. São tarefas típicas, detalhadas nos próximos itens,são também chamadas de ferramentas de Place & Route pois uma de suas funções é está, rotear e posicionar as células (RABAEY 2003).

1.3 Metodologia de Projeto

Há dois tipos básicos de metodologia de projeto: a *top-down* apresentada na Figura 2 e a *bottom-up* apresentada na Figura 3 . Na metodologia de projeto *top-down* define-se o bloco principal e depois se identificam os sub-blocos necessários para compor o bloco principal. A divisão dos blocos e sub-blocos é feita até o nível das células básicas. Na metodologia de projeto bottom-up é realizado o caminho inverso. Identificam-se as células básicas que estão disponíveis, que são organizadas formando um bloco maior. Este procedimento é realizado até chegar a um bloco principal.

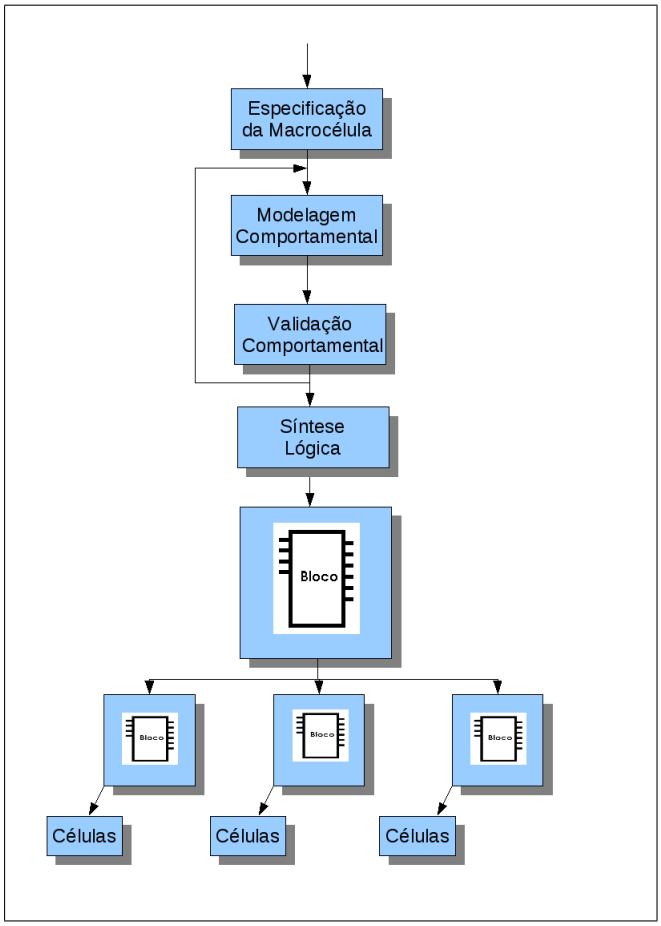

Figura 2: Metodologia de Projeto *Top down*

Tipicamente, uma combinação das duas metodologias é utilizada. O fluxo encontra um ponto intermediário, onde uma parte dos projetistas desenvolve uma biblioteca de células básicas e outra parte desenvolve o bloco principal (TOZETTO 2007).

Em Verilog existe o conceito de módulo, que é a construção de um bloco básico. Um módulo pode ser apenas um elemento ou uma coleção de elementos em menor nível com uma funcionalidade comum que pode ser usado em vários lugares no projeto. O módulo supre a necessidade funcional de um bloco de maior nível através de suas portas de interface (entradas e saídas), mas esconde sua implementação interna. Isto permite aos projetistas modificar módulos internos sem afetar o resto do projeto. Em Verilog um módulo é declarado por uma palavra-chave *module*. Uma palavra-chave correspondente *endmodule* deve aparecer no final da definição do módulo. Cada módulo deve ter um *module_name* que é identificado por um módulo e um *module_terminal_list* que descreve os terminais de entrada e saída do módulo. O trecho de Código 1 mostra a estrutura de um módulo tipico descrito em linguagem de hardware Verilog (IEEE 2005).

Verilog é uma linguagem estrutural e comportamental onde os módulos podem ser defi-

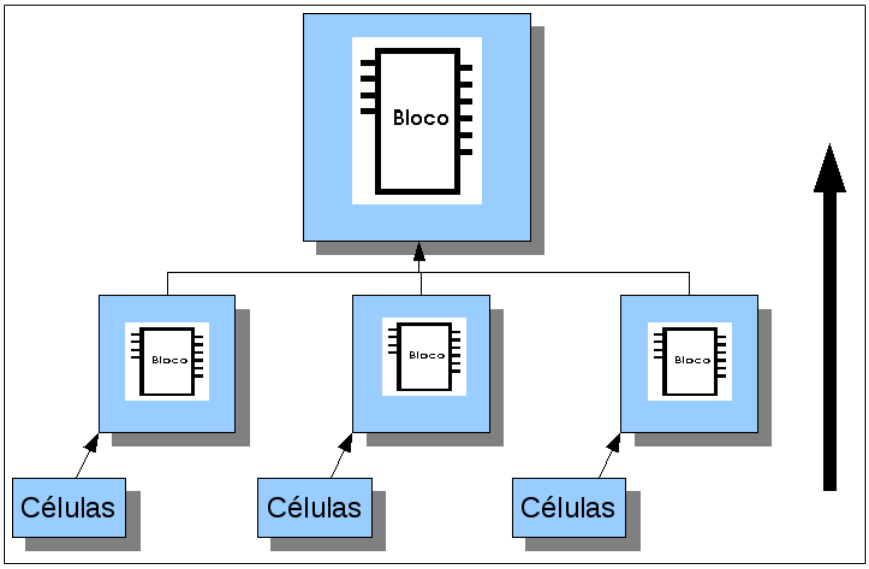

Figura 3: Metodologia de Projeto *Bottom up*

nidos em até quatro níveis de abstração, dependendo das necessidades do projeto. O módulo comporta-se da mesma forma com o ambiente externo, independentemente do nível de abstração que é descrito. Como a descrição interna não aparece para o ambiente externo, esta pode ser mudada sem qualquer alteração no ambiente. Os níveis de abstração são definidos como mostra a Figura 4.

- Comportamental ou algorítmico - Este é o mais alto nível de abstração disponibilizado por Verilog. Um módulo pode ser descrito em termos de seu algoritmo sem possuir detalhes de implementação de hardware. Projetar deste modo é muito parecido com programar em C;

- Fluxo de dados (*Register Transfer Level* - RTL)- Neste nível o módulo é projetado especificando-se o fluxo de dados. O projetista informa como os dados fluem entre os registradores de hardware e como os dados são processados no projeto;

- Portas (*gates*) - Estes módulos são projetados em termos de portas lógicas e interconexões entre suas portas. Projetar neste nível é similar a descrever um projeto em termos de um diagrama lógico de portas;

- Chaves(*switches*) - Este é o nível mais baixo de abstração em Verilog. Um módulo pode ser implementado em termos de chaves, nós e as interconexões entre eles. Projetar neste nível requer conhecimento detalhado da implementação de chaves através do uso de transistores.

```
1  //////////////////////////////////////////////////////////////////
2  ////  eth_register.v                                          ////
3  ////  This file is part of the Ethernet IP core project       ////
4  ////  http://www.opencores.org/projects/ethmac/               ////
5  //////////////////////////////////////////////////////////////////
6  `include "timescale.v"
7
8  module eth_register(DataIn, DataOut, Write, Clk, Reset, SyncReset);
9
10 parameter WIDTH = 8; // default parameter of the register width
11 parameter RESET_VALUE = 0;
12
13 input [WIDTH-1:0] DataIn;
14
15 input Write;
16 input Clk;
17 input Reset;
18 input SyncReset;
19
20 output [WIDTH-1:0] DataOut;
21 reg    [WIDTH-1:0] DataOut;
22
23 always @ (posedge Clk or posedge Reset)
24 begin
25   if(Reset)
26     DataOut<=#1 RESET_VALUE;
27   else
28   if(SyncReset)
29     DataOut<=#1 RESET_VALUE;
30   else
31   if(Write)                        // write
32     DataOut<=#1 DataIn;
33 end
34
35 endmodule   // Register
```

Código 1: Exemplo de um módulo em linguagem Verilog

Verilog permite ao projetista combinar todos os quatro níveis de abstração em um projeto. Normalmente, o nível mais alto de abstração é o mais flexível, e o projeto torna-se independente da tecnologia. É possível realizar uma inserção de módulos já criados, denominados instâncias. Quando o módulo é invocado, a linguagem Verilog cria um *template* em que cada objeto tem seu próprio nome, variáveis, parâmetros e interfaces de entrada e saída.

A funcionalidade do projeto pode ser testada pela aplicação de estímulos e verificação de resultados. O bloco que realiza esta tarefa é chamado bloco de estímulos, e pode ser escrito em Verilog. O bloco de estímulos é também comumente chamado de *testbench*. Dois estilos de aplicação de estímulos são possíveis. No primeiro estilo, o bloco de estímulo instância o bloco do projeto e controla os sinais diretamente no bloco do projeto. O segundo estilo é inserir o bloco de estímulos e de projeto em um módulo principal. Assim o bloco de estímulos interage com o bloco do projeto só através da interface. O módulo de estímulo controla os sinais de entrada do bloco do projeto e verifica o sinal de saída.

1.4 Síntese Lógica

Verilog é uma linguagem de descrição de alto nível para sistemas e projetos de circuitos. Suporta vários níveis de abstração e estilos de descrição, incluindo a descrição estrutural, fluxo

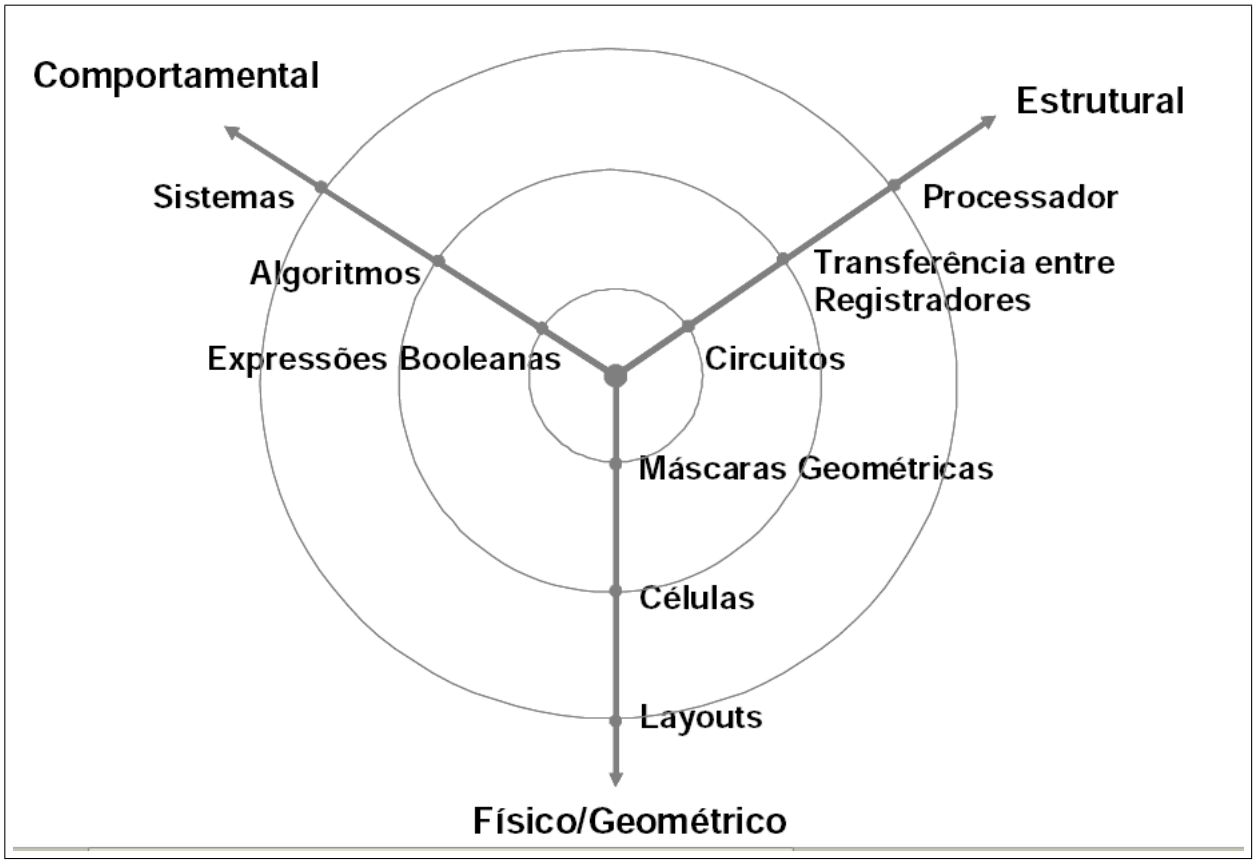

Figura 4: Diagrama de Gajsky - Níveis de abstração do projeto de hardware

de dados e comportamental. Permitem misturar vários níveis de entrada do projeto, sintetizando todos os níveis de abstração e minimizando a lógica necessária, resultando em um *netlist* final na tecnologia de fabricação escolhida. Todos as construções da linguagem são altamente simuláveis, mas não totalmente sintetizáveis, existem muitas construções na linguagens que não têm representação válida em circuitos digitais.

Um projeto descrito em Verilog, depois de simulado pode ser sintetizado e otimizado, sendo possível ainda misturar descrições HDL com esquemáticos. A hierarquia do projeto e o modo com que este é particionado influenciam no resultado da síntese. O particionamento do projeto pode ser baseado na funcionalidade, mas em alguns casos pode ser feito apenas com propósitos de síntese. Os objetivos de particionamento para síntese são produzir o melhor resultado, melhorar a velocidade, otimizar o tempo e simplificar o processo de síntese. Para tanto, há uma série de medidas que podem ser tomadas no sentido de otimização da síntese, por exemplo, o tamanho dos blocos não deve exceder 5000 portas, os tempos de execução dos algoritmos de síntese não são lineares, portanto, se um bloco a ser sintetizado é grande demais, sua execução pode demorar um longo tempo (BROWN e ZVONKO 2003).

Separar sub projetos com objetivos diferentes em módulos e registrar todas as entradas e saídas de um bloco são uma boa prática para simplificar o processo de síntese, lógicas com características similares devem ser agrupadas para melhorar a qualidade da síntese (CILETTI 2002).

1.5 Simulação

O propósito principal da simulação funcionar é o de ajudar na verificação da exatidão de um projeto ASIC. Por definição, simulação digital é um processo de exercitar o modelo teórico de um projeto como uma função do tempo, por alguns estímulos na entrada. O modelo teórico consiste na descrição de como os componentes no projeto são conectados, chamado de conectividade, e de modelos de simulação que descrevem a funcionalidade de cada um dos componentes. A seqüência de entrada é referida como estímulos ou vetores. A conectividade é obtida como resultado da captura de esquemáticos ou do processo de síntese. A simulação lógica é usada para verificar se o projeto se encontra dentro das especificações pretendidas. A habilidade do projetista em usar o simulador determina o grau de confiança no projeto, aumentando o nível de qualidade na produção de ASIC.

1.5.1 Cobertura de Código na Simulação

Simular certamente é uma técnica chave na verificação de um circuito. O maior problema com este tipo de verificação consiste no fato de que só se pode garantir correto funcionamento do que se simulou.

A cobertura do código HDL é a medida de alguns parâmetros que conseguem identificar funcionalidades que não foram testadas, indicando a qualidade da regressão atual.

As ferramentas comerciais que estão disponíveis pelas três grandes empresas oferecem o cálculo das métricas de cobertura de código RTL. Usa-se essa cobertura como uma indicação da qualidade da verificação. É importante que esta métrica não seja aplicada a um teste (simulação) isolada. Esta informação não seria de muita valia. A idéia é verificar a cobertura de toda a regressão (conjunto de simulações) que estamos aplicando ao projeto para garantir que este está correto.

Para isto as ferramentas em geral oferecem uma função de *merge* (junção), que a partir dos dados de cobertura dos vários testes é capaz de oferecer a métrica do conjunto completo.

Encontra-se dificuldade para mensurar objetivamente o que significa conseguir 100% de cobertura no código. Esta informação é certamente valiosa de forma a depurar a bateria de testes, mas conseguir 100% de cobertura não garante que a verificação está completa.

1.6 Síntese Física

A organização geométrica que viabiliza ou provoca o uso de uma determinada metodologia é denominada estilo de leiaute. Os estilos e os conceitos comumente utilizados em metodologias de Projeto VLSI podem ser vistos na Figura 5.

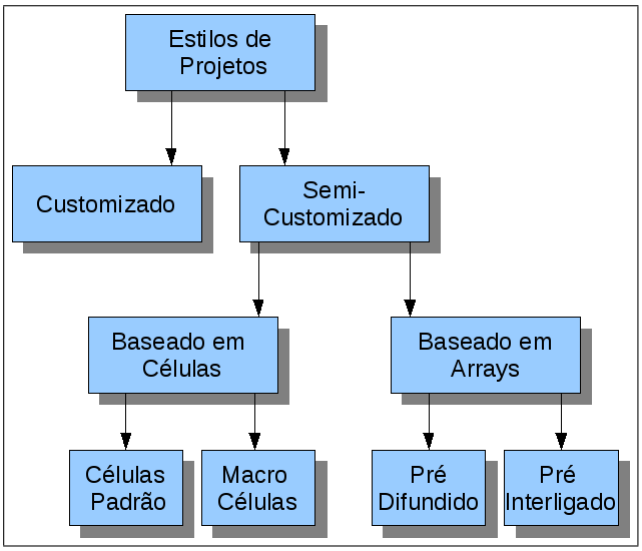

Figura 5: Estilos de projeto em microeletrônica.

No Projeto totalmente customizado (*full-custom*) se tem a liberdade para definir cada detalhe do leiaute dos transistores e conexões do circuito de maneira própria, sem o uso de padrões. Realizada por experientes projetistas elétricos e de leiaute, está técnica é empregada em circuitos ou partes de alto desempenho, como em microprocessadores de uso genérico. Em geral é utilizado para projetos analógicos, onde é difícil ter-se estruturas padronizadas.

Este estilo só pode ser implementado pela programação de todas as máscaras, razão pela qual aquela forma de programação também é denominada *full-custom*. Demonstra-se com isso a importância da síntese física, assim como a síntese lógica abordada amplamente nesse trabalho e da pesquisa por melhores metodologias e ferramentas automáticas em ambos os níveis de síntese.

Em estilos baseado em células (*cell based*), o circuito é feito com o uso de blocos projetados anteriormente que recebem o nome de células, macro células, módulos, etc. Este estilo surgiu como forma de reaproveitar o que já foi projetado. Usam-se os termos *general cell*, *macro-cell* ou *building blocks* quando estes blocos tem tamanhos e formas arbitrárias. Apesar de apresentar maior complexidade em combinar elétrica e geometricamente diferentes projetos dentro de um único, seu uso é ainda muito importante atualmente, sendo um meio-termo entre *full-custom* e *standard cell*. Como exemplo, o leiaute de um microprocessador inteiro pode ser embutido dentro de um outro projeto para fazer determinada parte do processamento (REIS 2003).

O estilo de células padrão (*standard cell*) usa uma estrutura de bandas de células de mesma altura, tomadas de uma biblioteca pré projetada e caracterizada em silício, e um conjunto de regiões para roteamento entre estas bandas. A metodologia associada permite um projeto rápido e seguro, onde a automação física é simplificada e o desempenho é garantido pelos elementos pré caracterizados. É um dos métodos preferidos para a implementação de lógica aleatória, aquela parte dos projetos que não possuem nenhuma estrutura regular. A Figura 6 demonstra estes estilos utilizados nos testes e análises para este estudo, aplicado no processo de síntese automática através de ferramentas CAD e bibliotecas de tecnologias de fabricação de ASIC contendo células pré-projetadas e testadas, disponibilizadas pelos fabricantes (REIS 2003).

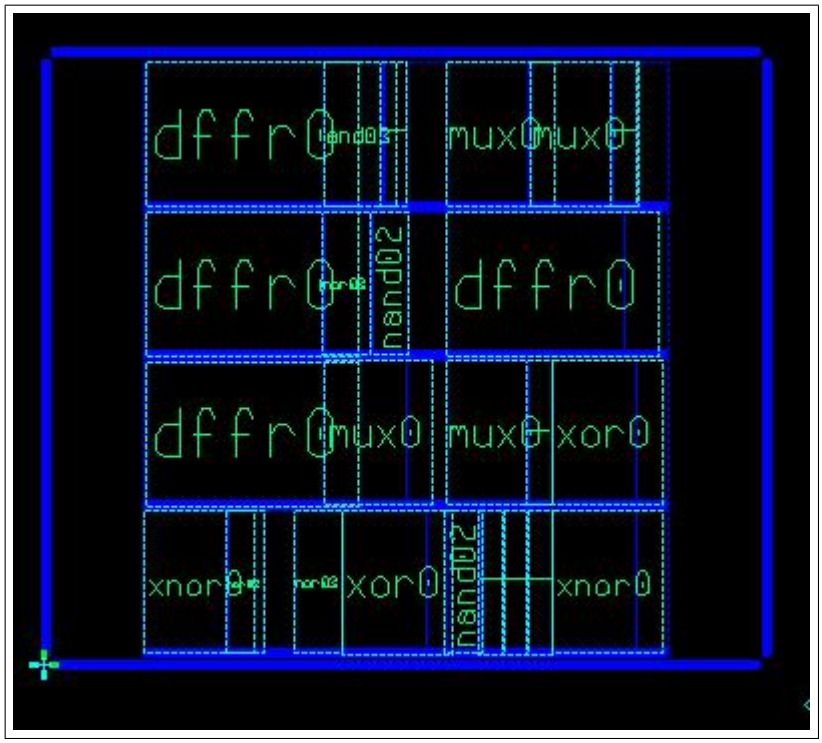

Figura 6: Projeto utilizando Standard Cells.

Projetos digitais em geral utilizam células conhecidas, portas and, or, nor, xor, flipflops, contadores, somadores, etc. A idéia de se projetar estas células e reusá-las parece certamente uma idéia óbvia, entretanto, mesmo para uma idéia aparentemente não original existem várias coisas que precisam ser definidas de modo que este tipo de abordagem funcione. Este tipo de modelo envolve a *foundry* que vai fabricar, o provedor de ferramentas EDA, o centro de design que vai desenhar o *chip*.

2 TÉCNICAS DE OTIMIZAÇÃO

Muitas são as técnicas e os processos utilizados nas várias etapas do projeto e fabricação de um circuito integrado para uso específico (ASIC). Todas estas técnicas buscam diminuir o custo de produção diminuindo a área de silício ocupada pelo circuito, ou aumentar o desempenho, incrementando a freqüência de trabalho e diminuindo o consumo de energia. Tenta-se otimizar ao máximo os circuitos projetados, para atender aos requisitos cada vez mais restritos de consumo, desempenho e durabilidade, impostos pelo concorrente mercado de semicondutores (CADENCE 2007).

Como mencionado anteriormente neste trabalho, empregou-se algumas técnicas de codificação em linguagem de descrição de hardware Verilog, que propõe a otimização do circuito. Técnicas estas que são empregadas na fase de codificação do projeto e visam melhorar os resultados da síntese Lógica, onde a linguagem HDL é transformada em portas lógicas.

As ferramentas de síntese automática como Leonardo Spectrum da fabricante Mentor Graphics e Encounter RTL Compiler da fabricante Cadence Inc., que foram utilizadas para a realização dos experimentos, já realizam diversas otimizações no momento do processo de síntese. Estas otimizações ocorrem na sua maioria nos trechos de código HDL que implementam lógica combinacional, por ser mais fácil implementar algoritmos de otimização para este tipo de lógica (MENTOR GRAPHICS 2008), (CADENCE 2008).

A otimização algorítmica empregada pelas ferramentas de síntese automática nem sempre pode otimizar ao máximo um projeto, baseando-se nisso algumas destas técnicas de codificação que serão estudadas, tentam aumentar o controle sobre a lógica gerada pelas ferramentas de síntese automáticas, fazendo com que as mesmas gerem estruturas já estudadas e simplificadas, com isso proporcionando um circuito mais otimizado.

Um ponto importante relatado é a questão das ferramentas de síntese automática permitirem elevar o nível de abstração através de descrições de alto nível do hardware, não necessitando-se realizar a descrição individual de cada porta lógica ou até mesmo transistor. Algumas das técnicas propostas, vão em oposição a esta filosofia das ferramentas, onde se utiliza um nível mais baixo de abstração na linguagem de descrição de hardware para conseguir um processo de síntese mais controlado e se obter o resultado desejado, sem inferências pouco otimizadas que podem ser geradas pelas ferramentas.

Provavelmente a existência de um ponto de equilíbrio ótimo entre o nível de abstração da descrição de hardware, otimização da síntese e custo da implementação deve ser conseguido. Para tanto será necessário o estudo e a análise destas técnicas propostas, discutindo-se a situação especifica onde as mesmas devem ser aplicadas.

2.1 Estudo Preliminar

Na literatura, encontrou-se um estudo comparando dois níveis de abstração da codificação de uma linguagem HDL com o resultado de área da síntese lógica utilizando-se ferramentas da fabricante Synopsys. Weijun Zhang (VAHID e GIVARGIS 2002) *in* (ZHANG 2001), comparou várias codificações de níveis comportamentais com suas equivalentes codificadas em nível RTL, utilizando a linguagem de descrição de hardware VHDL (IEEE 2004).

O estudo de Zhang obteve resultados positivos quando da codificação em um nível menos abstrato como é mostrado no gráfico a seguir, que apresenta na coluna Total, a área (*gates*) consumida pelos dois níveis de codificação, comportamental e RTL. Pode-se notar uma redução muito grande no número total de portas lógicas do nível comportamental para o nível RTL. Isso demonstra que nem sempre as ferramentas de síntese automática otimizam da melhor forma possível uma descrição em HDL. O gráfico também apresenta o número de portas lógicas utilizadas por blocos seqüenciais e blocos combinacionais. Nota-se uma inversão nos valores, para a codificação comportamental, o número de portas lógicas utilizadas em blocos seqüenciais é maior, já para a codificação RTL o número de portas lógicas utilizadas em codificação combinacional é maior. Isso demonstra que a síntese automática de descrições comportamentais inferência na sua maioria circuitos seqüenciais do tipo máquinas de estado. Já na codificação RTL onde os blocos gerados pela síntese automática possuem um controle mais rígido, pode-se forçar a ferramenta

a utilizar circuitos específicos e melhorar os resultados como demonstra o gráfico da Figura 7. Ainda no gráfico de Zhang observamos nos quatro pares de colunas mais a direita, os números de portas de entrada e saída, conexões, células padrões e referências de potencial foram utilizadas para as duas implementações.

Utilizando-se a mesma linha de raciocínio, realizou-se um estudo com a linguagem Verilog e algumas das principais técnicas de otimização aplicáveis a descrição HDL que foram encontradas na literatura. Buscou-se com isso medir a eficiência destas técnicas e também a relação custo benefício comparando os ganhos finais em área, potência dissipada e *timing* relacionada com o tempo gasto (complexidade) para implementá-las. Diferentemente do estudo de Zhang, não utilizou-se modelos essencialmente comportamentais, todas as técnicas foram implementadas sobre modelos mistos, os quais são normalmente empregados no desenvolvimento de projetos deste tipo.

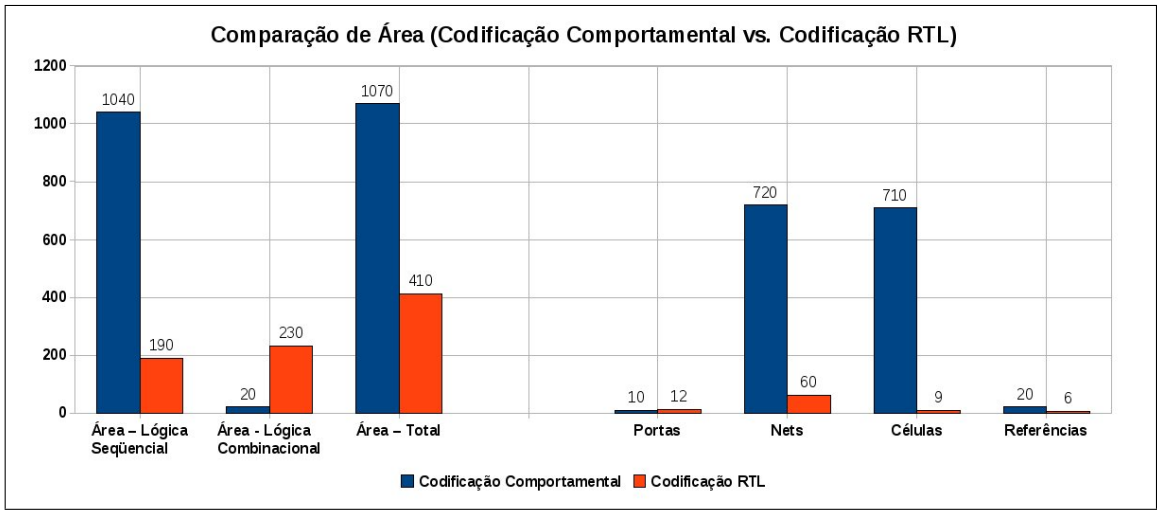

Figura 7: Estimativas de área levantadas por (ZHANG 2001).(Eixo Y = Portas Lógicas)

Para exemplificar melhor a diferença dos níveis de abstração comportamental e RTL utilizados por Zhang em seu estudo, o exemplo do circuito lógico que realiza o cálculo do máximo divisor comum está disposto ao final do trabalho, sendo o Anexo A, a codificação utilizando abstração comportamental e o Anexo B, a codificação utilizando a abstração RTL.

O resultado da síntese em número de portas lógicas foi apresentado no gráfico da Figura 7, onde pode-se observar no eixo Y o número de portas lógicas resultantes da síntese para os casos comportamental e RTL, comparando individualmente as portas lógicas gastas em cada parte do circuito.

2.2 Técnicas Levantadas na Literatura

A seguir serão descritas algumas das principais técnicas de otimização pesquisadas na literatura e que são utilizadas no projeto de um ASIC. Tais técnicas se restringem a nível de descrição de hardware através de linguagem HDL, mais especificamente a linguagem Verilog. O estudo destas técnicas e o impacto de sua implementação através da linguagem Verilog na síntese lógica como já foi dito é, um dos principais objetivos deste trabalho.

Após levantamento em literatura das técnicas, classificou-se as mesmas em três grandes grupos:Técnicas de otimização de tempo, potência e área. Sendo as técnicas de otimização de tempo aquelas que buscam diminuir o atraso do caminho critico do circuito, proporcionando ao mesmo trabalhar a uma freqüência maior. As técnicas de potência, são aquelas que buscam diminuir a potência total dissipada, sendo aqui empregadas técnicas que reduzem a potência dinâmica e a potência estática e as técnicas que buscam reduzir a área ocupada em silício, que nesta etapa de codificação HDL, implica basicamente na redução de portas lógicas.

2.2.1 Técnicas de Codificação Visando Otimizações de Tempo (*timing*)

Dentre as otimizações que melhoram o *timing* de um determinado circuito destacam-se a eliminação de níveis lógicos e a replicação de caminhos, devido a sua baixa complexidade de implementação e portanto melhor custo benefício. Ambas buscam diminuir o tempo de propagação de um determinado sinal, fazendo assim com que o circuito possa trabalhar com um relógio de sincronismo que utilize uma freqüência mais elevada e conseqüentemente realizar um número maior de operações lógicas e aritméticas em uma unidade de tempo.

2.2.1.1 Eliminação de Níveis Lógicos *(Flattening)*

Sabendo-se que cada porta lógica resultante da síntese da descrição RTL de um circuito possui um atraso específico que é adicionado na propagação do sinal, busca-se reduzir ao máximo este atraso paralelizando-se ou eliminando-se níveis de lógica intermediários.

Este atraso é dado por várias características tais como: tempo de chaveamento, resistividade do material e temperatura de operação da tecnologia de transistores utilizados na fabrica-

ção. Sendo assim a tecnologia utilizada também interfere nos resultados, mas este estudo não abordará tal escolha, pois o foco são técnicas de codificação em linguagem HDL, otimizando o projeto desde as primeiras fases, não nas fases do *back-end*

No exemplo da Figura 8 pode se verificar uma associação lógica implementada de maneira normal, onde as operações são realizadas serialmente. Resultando assim um tempo de propagação total do sinal oriundo da soma dos atrasos individuais de cada um dos níveis de lógica (UYEMURA 2002).

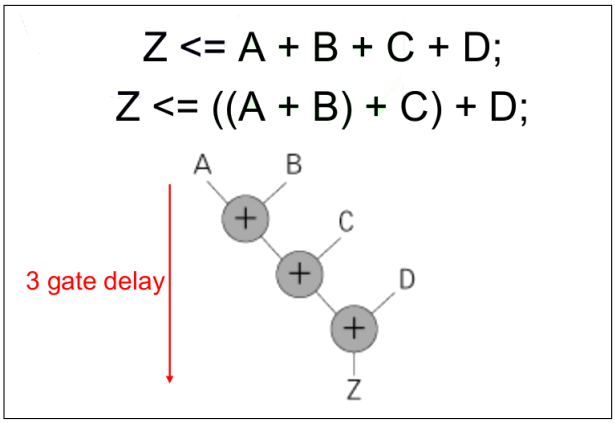

Figura 8: Serialização de níveis lógicos (UYEMURA 2002)

No exemplo da Figura 9 pode-se verificar uma associação lógica de maneira que os níveis que não possuem dependências são paralelizados, Resultando assim um tempo de propagação total do sinal menor que no exemplo da Figura 8, pois a propagação do sinal se dá apenas por dois níveis agora.

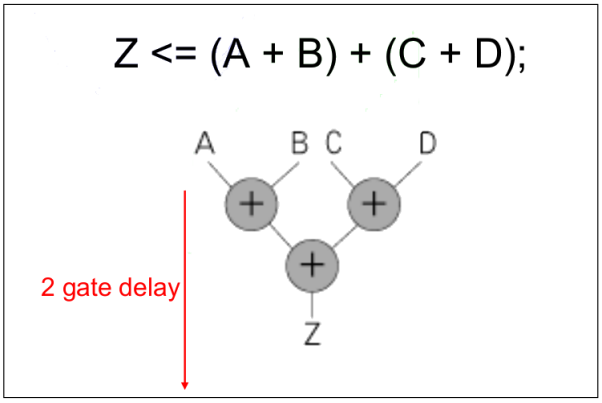

Figura 9: Paralelização de níveis lógicos que não possuem dependências (UYEMURA 2002)

Teoricamente esta técnica de otimização a nível de codificação RTL deverá gerar resultados positivos, sua aplicação é de fácil implementação e não se restringe unicamente a codificação Verilog. Após a análise da aplicação desta técnica ao estudo de caso no capítulo seguinte, saber-

se-á qual seu impacto na síntese lógica e como as ferramentas de síntese automática interpretam e otimizam esta maneira de codificação.

2.2.1.2 Replicação de Caminhos *(Paths)*

Nesta técnica faz-se uma análise do resultado da síntese lógica, buscando-se o chamado caminho crítico do bloco, ou até mesmo do projeto todo, determinando a máxima freqüência que o circuito pode trabalhar. Estes caminhos são aqueles que possuem o maior atraso dos sinais dentro do projeto limitando a freqüência (DAHAN 2007).

Na Figura 10 o *flip-flop* escurecido esta conectado a três outras redes no *chip*, redes estas distribuídas em locais diferentes do *chip*; devido ao posicionamento destas redes não é possível que o *flip-flop* fique o mais perto possível de todos os *flops* das redes que amostram o sinal, sendo assim a ferramenta de síntese tem como compromisso colocar o *flop* a uma distância mais ou menos equivalente de onde o sinal será amostrado pelas redes equilibrando o atraso. Como ele não fica o mais próximo possível de nenhuma das três redes todos os caminhos tem um atraso maior do que se os flops de geração do sinal e o de amostragem ficassem lado a lado. Na Figura 10 o caminho entre o *flip-flop* de entrada do circuito que passa por uma nuvem de lógica combinacional e é amostrada pelo *flop* escurecido, existe uma sobra de tempo bastante grande , pois em geral lógicas combinacionais são mais rápidas que lógicas seqüencias. Isso significa que o sinal que é amostrado pelo flop escurecido poderia ser amostrado cerca de 150 unidades de tempo antes do que é amostrado atualmente , ou então o caminho do *flop* de entrada até o *flop* escurecido poderia ter um atraso maior sem que o tempo total do circuito fosse afetado (DAHAN 2007).

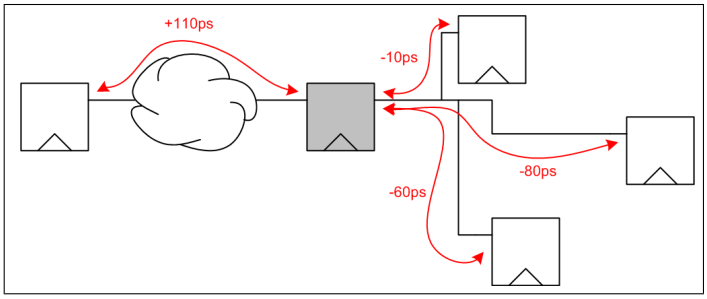

Figura 10: Circuito sem replicação de caminhos (DAHAN 2007).

Replicando-se o *flip-flop* escurecido como apresentada na Figura 11, a ferramenta de síntese pode instanciá-lo o mais próximo possível do *flip-flop* da rede onde será amostrado, redu-

zindo ao máximo este atraso. Esta replicação implicará em atrasos diferentes entre o *flop* de entrada e a amostragem do *flop* escurecido, mas estes atrasos geralmente são menor do que o atraso aplicado pela ferramenta de síntese utilizada para equilibrar as distâncias na Figura 10, quando ainda não existia a replicação do caminho.

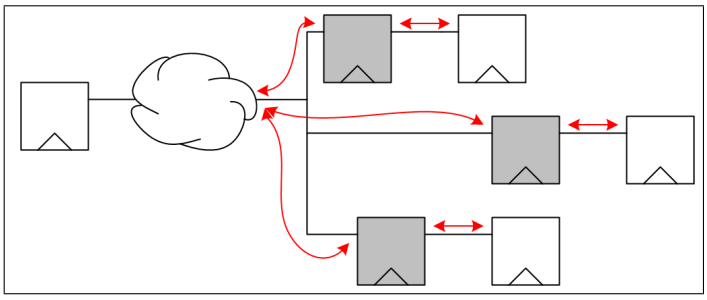

Figura 11: Caminhos replicados, diminuindo assim o *fan-out* e também o delay do caminho crítico (DAHAN 2007).

Consegue-se melhores resultados ainda, quando replica-se não só a lógica seqüencial, mas toda a lógica, como mostrado na Figura 12 . Assim cada lógica pode perfazer um caminho mais curto (menor atraso) até a rede onde o sinal deve ser amostrado, podendo empregar um clock maior para servir como referência, o que resulta em maior processamento na mesma unidade de tempo.

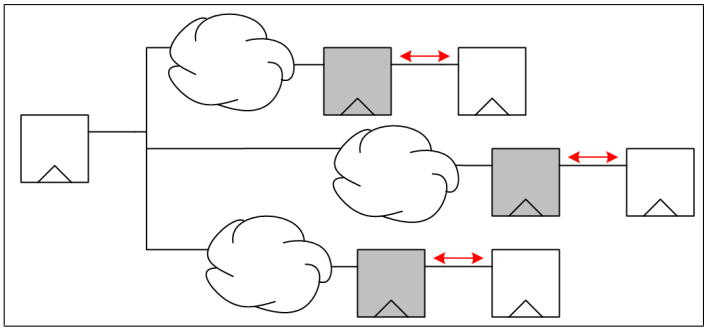

Figura 12: Caminhos totalmente replicados, incluindo a lógica combinacional, diminuindo assim atraso do circuito (DAHAN 2007).

A aplicação desta técnica também tem uma implicação no processo de síntese física, com a replicação o fan-out de cada caminho diminui, permitindo utilizar-se portas lógicas da biblioteca de células padrão confeccionadas por transistores de menos porte, pois não necessitam suportar correntes mais elevadas como no caso de um único caminho sendo amostrado por três redes (DAHAN 2007).

Esta técnica, possui *trade-off* em relação a área e potência causado pelo fato da inserção de maior número de portas lógicas para duplicar o caminho deverá ser empregada com cuidado

pois poderá gerar um aumento significativo de área .

2.2.2 Técnicas de Codificação Visando Otimizações de Potência Dissipada

Três componentes influenciam na potência total dissipada em um circuito integrado. A primeira é a corrente de fuga existente na junção, uma característica específica dos materiais utilizados na fabricação. A segunda é o potência dissipada estaticamente, resultante do processo de polarização dos transistores que também é bastante influenciada pelo material utilizado,assim como a corrente de fuga e por variáveis ambientais como temperatura, campo magnético entre outras. A terceira componente é a potência dissipada dinamicamente, resultante do processo de chaveamento ou troca de estado dos transistores.

Por se tratar de uma parte significativa, podendo chegar em média a 70% do valor total de potência dissipada , segundo (WEST e HARRIS 2005) este trabalho estará focado em técnicas de redução de potencia dinâmica e estática, até mesmo porque a nível de descrição de hardware não existem otimizações possíveis para corrente de fuga, que são características do processo de fabricação empregado na confecção do circuito.

2.2.2.1 Inversão de Barramento (Bus inversion)

Esta técnica já é muito empregada para o envio de dados entre *chips* diferentes e por barramentos de placas. Passou a ser utilizadas internamente no *chip* devido a ser facilmente implementada e gerar uma redução eficiente de potência pois reduz o número de chaveamentos dos transistores avaliando o dado a ser transmitido. Com essa redução de chaveamentos temos uma redução da potência dinâmica.

Na Figura 13 é possível ver o funcionamento da técnica de inversão de dados no barramento que reduz o chaveamento dos transistores. A coluna da esquerda representa a seqüencia de transmissão dos bits de modo normal, onde os bits em vermelho representam os bits transicionados de um estado para outro (cada linha representa um estado de transmissão), a coluna da direita representa a seqüencia de transmissão dos bits com a técnica de inversão de barramento, onde se acrescenta um bit a mais na palavra de dados transmitida para controle do modo de re-

presentação, da mesma forma os bits em vermelho representam bits transicionados de um estado de transmissão para o outro.

A função do bit de controle adicionado é indicar quando o dado representado no barramento é invertido, caso o próximo dado a ser transmitido possui mais de 50 % de seus bits invertidos em relação ao dado anterior, ele é representado de modo invertido no barramento, reduzindo os chaveamentos dos transistores como pode se observar pelos total de transições em cada um dos casos (WEST e HARRIS 2005).

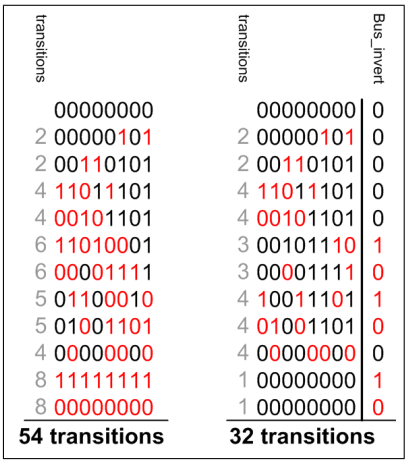

Figura 13: Técnica de inversão de barramento - número de transições

Um ponto importante a ser levado em conta é a necessidade do desenvolvimento de uma lógica de controle para a inversão do barramento e a inserção de armazenamentos para realizar a comparação dos estados, o que pode aumentar a complexidade da lógica e o número de portas utilizadas.

2.2.2.2 Representação de números binários (Binary Number Representation)

Esta técnica consiste no simples fato de analisar a melhor maneira de representar um número binário no projeto, dependendo da faixa numérica que se quer representar.

A representação em complemento de dois é a mais usada na prática. Embora a questão da inversão de maior número de bits em representações que variem de números negativos a números positivos perto do zero como pode ser visto na Figura 14 mais do que em sinal magnitude. Um ponto positivo para a representação em complemento de dois é o da lógica aritmética para somas e subtrações ser mais simplificada do que na representação sinal magnitude.

Existe um *trade-off* neste caso que precisa ser considerado caso a caso, pesando-se transições na representação dos bits e aumento de portas lógicas, devido ao fato de se necessitar de lógicas mais complexa para realizar somas e subtrações na representação sinal magnitude (MICHELI 2004) (DAHAN 2007).

Complemento de 2	Sinal Magnitude
0 – 00000000	**0** – 00000000
-1 – 11111111	**-1** – 10000001

Figura 14: Representação numérica

2.2.2.3 Habilitar/ Desabilitar Lógica (Disabling / Enabling Logic Clouds)

Esta técnica consiste em implementar mecanismos que desabilitem partes do circuito que não estejam sendo usadas, geralmente desabilitando o *clock* do bloco ou lógica. Também conhecida como *clock gating*, sua implementação deve ser empregada com cuidado, pois precisa de mecanismos auxiliares que garantam o sincronismo do *clock* no momento da habilitação do circuito (MILLS 2007).

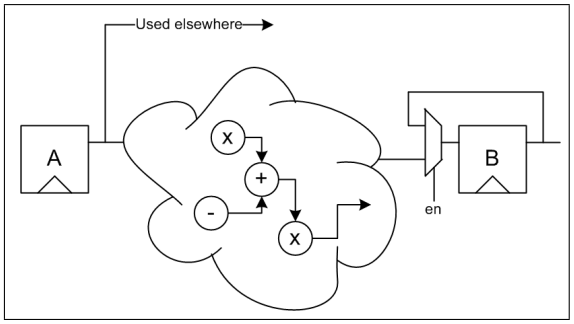

Figura 15: Habilitação de lógica

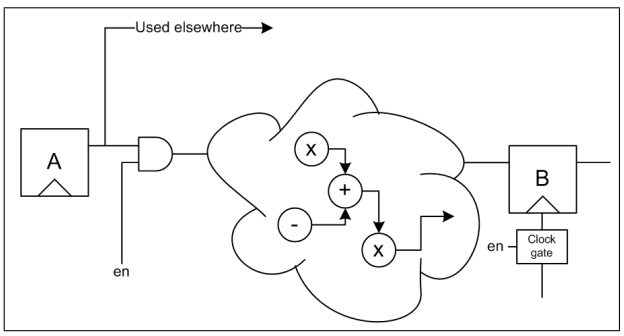

Figura 16: Chaveamento do clock

As Figuras 15 e 16 demonstram dois casos de aplicação de habilitação / desabilitação de lógica para reduzir chaveamentos desnecessários.

2.2.2.4 Lógica com alta taxa de comutação (*High Activity Nets*)

Esta técnica consiste em separar sinais que tenham um alto nível de transição usados em expressões lógicas, ocasionando a cada transição o chaveamento de todos os demais transistores envolvidos na expressão.

Estes casos típicos ocorrem em expressões lógicas que têm sinais atualizados esporadicamente juntamente com outros atualizados por exemplo, a cada comutação do relógio, onde a variação de um dos dados implica em toda uma modificação dos resultados intermediários da lógica até ser propagada para o resultado final (DAHAN 2007).

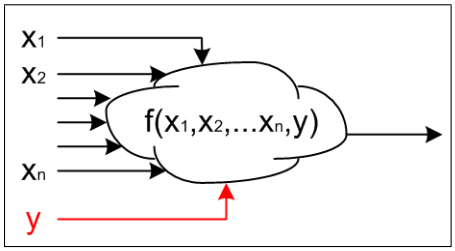

Figura 17: Sinais com muitas transições misturados a logica normal

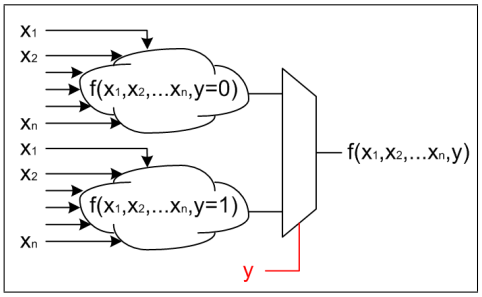

Figura 18: Sinal com muitas transições separado do resto da lógica

As Figuras 17 e 18 mostram como a separação do sinal pode ser realizada. São replicadas as nuvens de lógica com o sinal que possui alta transição de estado, no caso o sinal Y, fixado para cada uma das nuvens em um valor diferente. Estas nuvens de lógica são conectadas a um multiplexador que selecionará para a saída o valor desejado conforme o sinal Y. Note que a cada transição do sinal Y apenas o chaveamento do mux é necessário, e os resultados da nuvem lógica não são alterados. Com isso ocorre uma diminuição dos chaveamentos, e conseqüente diminuição da potência dinâmica, porém cabe salientar a existência de um aumento de portas lógicas que deve ser levado em consideração no momento de aplicar a técnica.

2.2.3 Técnicas de Codificação Visando Otimização de Área

Basicamente para redução de área de silício é necessária a diminuição de portas lógicas, através de simplificações de expressões, minimização de estados em máquinas de estado, e eliminação de lógica replicada. Relata-se aqui duas técnicas que consistem basicamente na eliminação de portas desnecessárias para a lógica, mantendo-se o seu comportamento funcional original.

2.2.3.1 Extração MUX desnecessários

Na Figura 19 é possível verificar a eliminação de mux gerados desnecessariamente pela codificação RTL normal. Codificando-se de maneira a utilizar as primitivas lógicas é possível forçar a ferramenta de síntese lógica a substituir o multiplexador apenas por portas lógicas *and* e *or*.

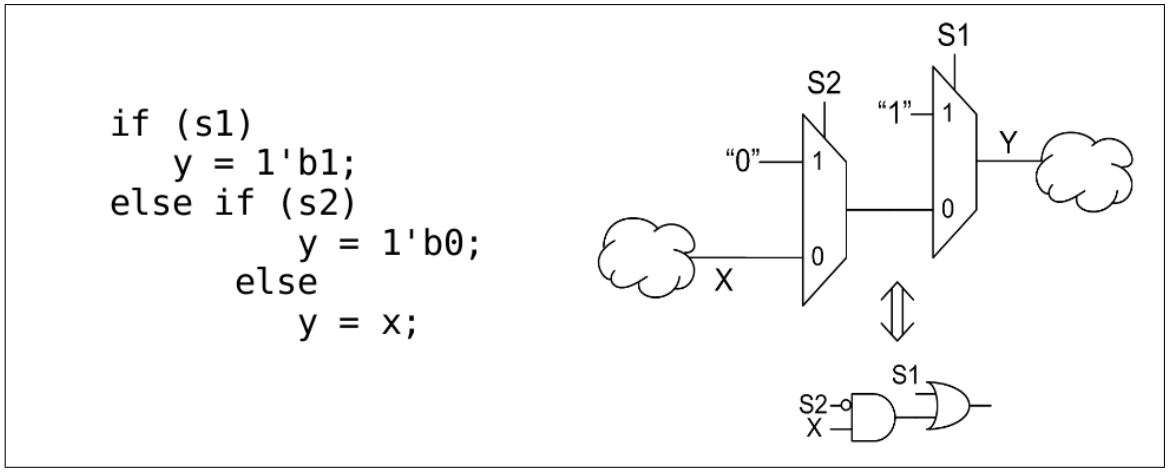

Figura 19: Exemplo de extração de um mux desnecessário

Esta técnica pode ser empregada em casos onde a codificação do mux tem um dos sinais de entrada fixo. Na Figura 20 são apresentados alguns exemplos de equivalências de codificação utilizando-se primitivas lógicas ao invés da codificação normal Verilog.

Esta técnica como pode-se observar, produz uma redução bastante grande de portas lógicas, uma vez que substitui um muliplexador composto por várias portas lógicas, por apenas uma ou duas primitivas lógicas (CILETTI 2002).

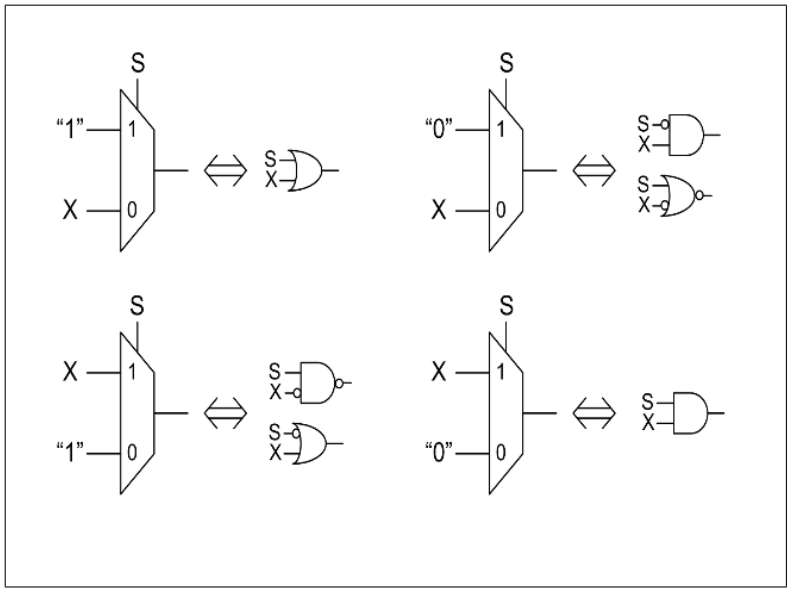

Figura 20: Exemplos de equivalências utilizando-se primitivas lógicas para a codificação

2.2.3.2 Simplificações Lógicas

A simplificação lógica de expressões na codificação RTL reduz o número de portas lógicas resultantes, o que garante uma menor área utilizada pelo projeto.

A complexidade do circuito lógico e da expressão lógica que o circuito representa estão diretamente ligadas. Embora a tabela da verdade que representa uma determinada função seja única, devido as diferentes possibilidades de simplificações a serem utilizadas, a expressão lógica resultante pode ser escrita de diferentes formas.

A utilização da simplificação algébrica para minimização de funções lógicas não segue regras claras e seqüenciais para a correta manipulação algébrica, assim muitas vezes simplificações realizadas pelas ferramentas nem sempre são as mais otimizadas em determinados casos. Esta técnica é um procedimento fortemente dependente da experiência e do adestramento do projetista, que neste caso deve saber identificar e aplicar corretamente o processo de simplificação lógica no circuito.

Dentre alguns métodos de simplificação lógica pode-se citar os postulados e teoremas como o de Demorgan, métodos como o de Karnaugh e de Quine-McCluskey além das lei de associação e distribuição de termos e eliminação de lógica desnecessária (GAJSKI 1997). Para a correta aplicação destes métodos de simplificação e minimização lógica é necessário ao projetista conhecer individualmente cada um deles e saber de suas restrições de aplicação, bem como um

possível aumento de atraso no circuito devido ao fato de se encadear um maior nível de lógica a expressão, mesmo diminuindo o número de portas lógicas empregas.

Como exemplo de extração de lógica desnecessária pode-se observar as Figuras 21 e 22. Na Figura 21 a lógica que realiza a operação é inserida internamente ao *looping*, gerando uma replicação dos circuitos que irão realizar a adição. Está implementação disponibiliza os resultados de todas as adições ao mesmo tempo, o que muitas vezes não é necessário, pois apenas um resultado é utilizado a cada ciclo.

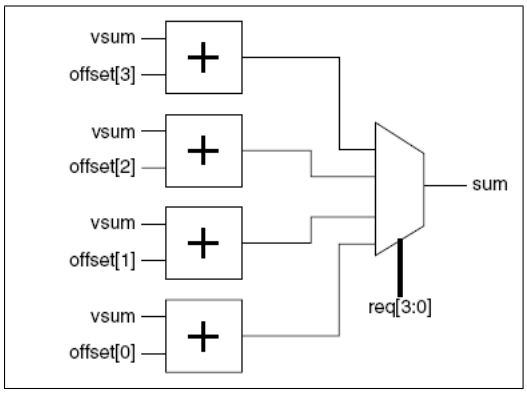

Figura 21: Implementação com lógica redundante

Se esta lógica de adição não fizer parte do caminho crítico do projeto, ela poderá ser retirada do *looping* e implementada uma única vez, como pode ser observado na Figura 22, permitindo-se o compartilhamento do circuito resultante e reduzindo assim área e conseqüentemente potência.

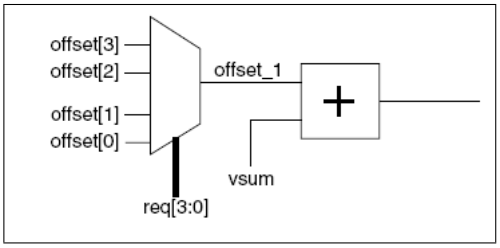

Figura 22: Exemplo de redução de lógica redundante

2.3 Considerações a Respeito das Técnicas de Codificação

As ferramentas de síntese automática, utilizam otimização no seu processo de tradução, possuindo modelos subjacentes que podem ser bastante complexos . No entanto, o projetista não precisa dominar todos os conceitos associados a estes modelos para explorar os recursos da ferramenta, visto que seu papel é parametrizar a ferramenta para que esta gere descrições corretas. Ele apenas precisa dominar a influência desta parametrização no desempenho dos modelos

de síntese. A aplicação destas técnicas à descrição HDL, nada mais é do que a parametrização mais rígida da ferramenta de síntese automática, visando que a mesma gere resultados mais otimizados.

Contudo, resultados diferentes podem ser esperados de ferramentas de fabricantes distintos, os modelos de síntese utilizados provavelmente não serão os mesmos. Sendo assim os resultados da aplicação da mesma técnica pode ser melhor em uma ferramenta e não tão bom assim em outra.

Outro ponto importante a ser considerado é que a síntese lógica para projetos com células padrão, que é o caso deste estudo, já é dependente de tecnologia de fabricação, ou seja, já depende das bibliotecas que descrevem as portas lógicas, e demais componentes possíveis de se utilizar naquela *foundry* e apresentam valores característicos diferentes para cada tecnologia.

3 ESTUDO DE CASO - IP CORE MAC ETHERNET

Neste capítulo procurou-se abordar o padrão Ethernet/IEEE 802.3, o qual foi tomado como estudo de caso deste trabalho. Iniciou-se por uma introdução dos principais conceitos envolvidos, características técnicas e peculiaridades do padrão e suas evoluções. Após descreveu-se *IP Core* que implementa tal padrão, seu funcionamento e o processo de simulação do *IP Core* através de *testbenchs*.

3.1 A Tecnologia Ethernet

Diante da necessidade de diminuir custos, aumentar a confiabilidade, disponibilizar o compartilhamento de recursos físicos como discos, impressoras e outros periféricos assim como o compartilhamento de informações em banco de dados e outros programas, surgiram as redes de computadores.

Nos primórdios as tecnologias de redes eram todas proprietárias, não seguindo padronização, isso dificultava a integração de redes de diferentes fabricantes, o progresso das pesquisas e a venda de equipamentos.

A proposta original da tecnologia Ethernet foi publicada no ano 1976 por Metcalfe e Boggs, a mesma detalhava uma rede onde todas as estações compartilhavam do mesmo meio de transmissão, um cabo coaxial, utilizando a configuração de barramento,atingindo taxa de transmissão de 2,94 Mbps (METCALFE e D.BOGGS 1976).

Posteriormente, no ano 1980, esta tecnologia foi padronizada pela primeira vez pelo consórcio de empresas DEC-Intel-Xerox com o nome de padrão DIX Ethernet. Finalmente, no ano 1985, sob a direção da IEEE (Institute of Electrical and Electronics Engineers) *Standards Asso-*

ciation se definiu o padrão Ethernet oficial para redes locais, identificando-o com o código IEEE 802.3.

Desde a sua regulamentação pelo IEEE suas especificações foram totalmente disponibilizadas, com esta abertura, combinada com a facilidade na utilização e com sua robustez, o resultado foi largo emprego desta tecnologia (IEEE 2002).

O padrão IEEE 802.3 ocupa o equivalente aos níveis 1 e 2 do modelo de referência para interconexão de sistemas abertos, em inglês, *Reference Model for Open Systems Interconnection* ou RM-OSI. O nível 2 do RM-OSI, também conhecido como nível de enlace de dados, é representado pelo subnível de controle de acesso ao meio ou MAC, em inglês, *Media Access Control* e pelo subnível de controle de enlace lógico, em inglês, *Logical Link Control* ou LLC, como se observa na Figura 23 (IEEE 2002).

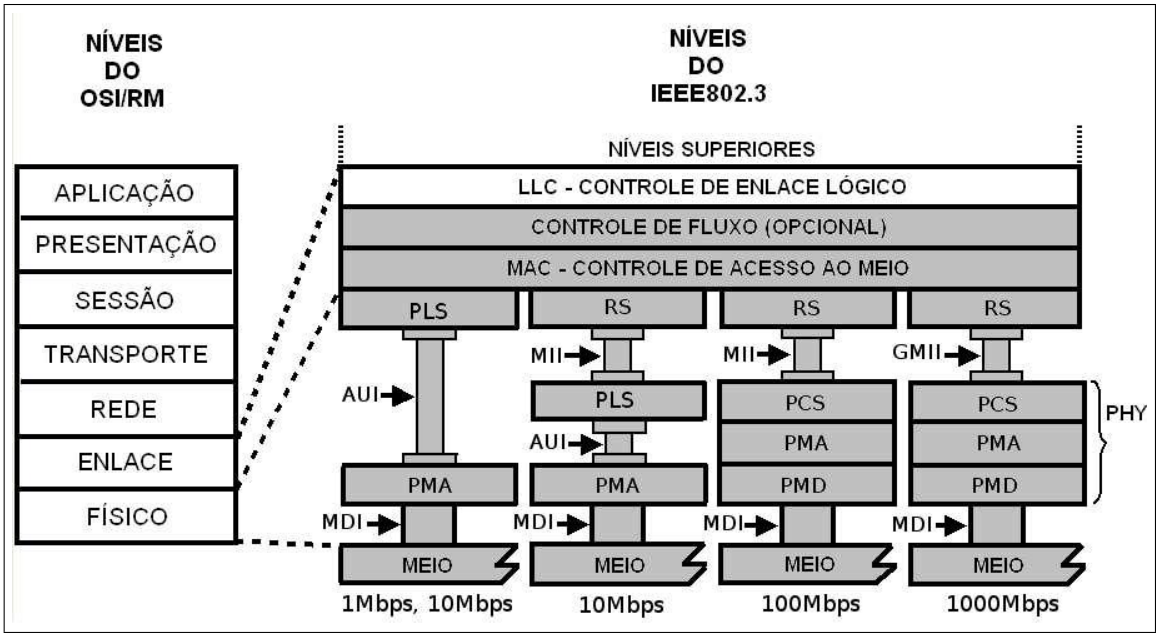

Figura 23: Relação entre o padrão Ethernet/IEEE 802.3 e o modelo RM/OSI (HORNA et al. 2007).

A tecnologia Ethernet, basicamente, consiste de três elementos: o meio físico, as regras de controle de acesso ao meio e o quadro de dados. O subnível MAC, onde estão implementadas as regras de acesso ao meio e a montagem/ desmontagem do quadro suporta dois modos de operação: o modo Half-duplex onde a transmissão de dados sobre um meio físico compartilhado e o modo Full-duplex que é utilizado para transmissão de dados em ligações ponto-a-ponto (TANENBAUM 2003).

O quadro Ethernet utilizado para trafegar dados através do meio possui campos de tamanho fixo, com exceção do campo de dados que pode variar entre 64 e 1518 bytes. A sinalização de início e sincronização do quadro é dada pelos campos de preâmbulo e SFD (Delimitador de Inicio de Frame, em ingles, *Start Frame Delimiter*). Já campo FCS(Checagem de Seqüência do Frame, em inglês, *Fram Check Sequence*) é um valor de 4 bytes resultante da técnica CRC-32 (32bit de Checagem de Redundância Ciclica, em inglês, *32 bits - Cyclic Redundancy Check*), utilizada para verificação de erros no quadro recebido. Na Figura 24 é possível observar a estrutura do quadro padrão 802.3, onde além dos campos descritos anteriormente também fazem parte os endereços MAC de origem e destino do dado.

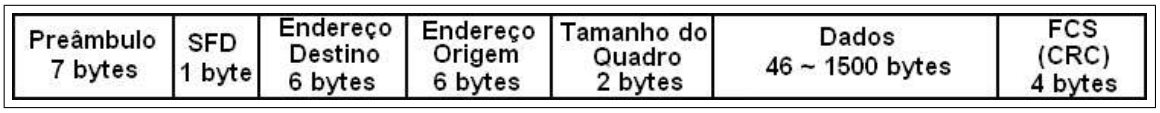

Figura 24: Formação do quadro Ethernet, bits de preâmbulo, endereços MAC, delimitadores, dados e checagem do quadro.

3.2 Modo *Half-Duplex* e Protocol CSMA/CD

O modo de transmissão em half-duplex requer que apenas uma estação transmita enquanto que todas as outras aguardam em silêncio. Esta é uma característica básica de um meio físico compartilhado. O controle deste processo fica a cargo do protocolo de acesso CSMA/CD *Carrier Sense Multiple Access with Collision Detection* que é especificado na cláusula 4 do padrão IEEE 802.3 (IEEE 2002).

Qualquer estação pode transmitir quando percebe o meio livre, pode ocorrer que duas ou mais estações tentem transmitir simultaneamente, nesse caso, ocorre uma colisão e os pacotes são corrompidos. Quando a colisão é detectada, a estação tenta retransmitir o pacote após um intervalo de tempo aleatório, isto implica que o CSMA/CD pode estar em três estados: transmitindo, disputando ou inativo. A Figura 25 a mostra o comportamento do CSMA/CD no tempo. Pode-se observar os estados descritos anteriormente, a espera exponencial no caso de colisões e o reinício da transmissão quando se detecta o meio livre.

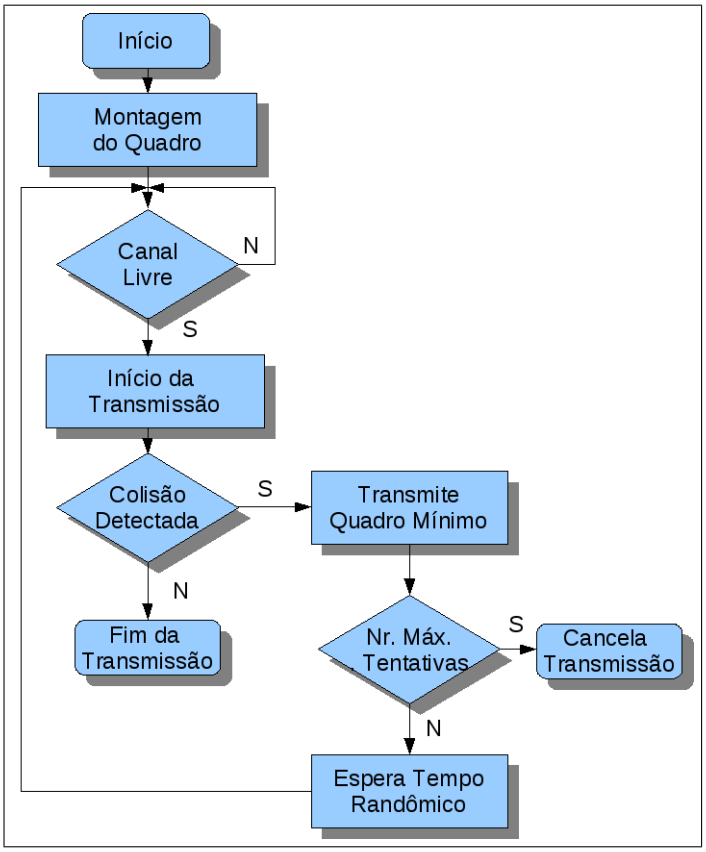

Figura 25: Funcionamento do algoritmo CSMA/CD implementado no MAC

3.3 Modo *Full-Duplex* e o Controle de Fluxo

Com a evolução das redes de computadores e a necessidade de velocidades maiores, novas características foram introduzidas no suplemento do padrão IEEE 802.3x, no ano 1997, uma delas foi o modo de operação *Full-duplex* que desativa o protocolo CSMA/CD e permite que seja feita a comunicação bidirecional simultânea entre dois dispositivos.

Neste mesmo suplemento do padrão IEEE 802.3, incluiu-se um mecanismo opcional para o controle de fluxo sobre conexões *full-duplex* denominado de *MAC Control*. Este mecanismo é especificado na cláusula 31 do padrão IEEE 802.3, sendo sua principal função causar uma pausa na transmissão de um dispositivo que está fornecendo quadros a uma velocidade além da suportada pela capacidade do buffer de armazenamento do dispositivo receptor. Esta pausa é obtida através do envio e da detecção de um quadro especial do tipo *Pause Frame* que transporta um parâmetro denominado de tempo de pausa, indicando o tempo em unidades de *slot* em que a transmissão deve ser interrompida (IEEE 2002).

3.4 Interface Independente do Meio (*Media Independent Interface*)

A função desta interface que está especificada na cláusula 22 do padrão IEEE 802.3, é permitir que qualquer um dos meios físicos padronizados para o trafego de dados do padrão Ethernet possa comunicar-se com o MAC, utilizando velocidades de transferência de dados de 10 Mbps e 100 Mbps. Ela também é responsável pela comunicação de controle e gerenciamento de dispositivos do nível físico de 10/100 Mbps, em inglês, *Physical Layer Devices* ou PHY. Esta parte da interface recebe o nome de Interface de Gerenciamento de Media Independente, em inglês, *Media Independent Interface Management* ou MIIM e permite acessar às informações de estado, erros e configurações específicas do PHY (IEEE 2002).

3.5 *IP Core* MAC Ethernet

Utilizou-se como base para o estudo um *IP Core* desenvolvido em linguagem de descrição de hardware Verilog, que implementa em hardware o subnível MAC de uma rede Ethernet/IEEE 802.3. Este IP Core está originalmente disponível no repositório de códigos de descrição de hardware chamado Opencores.Org (OPENCORES.ORG 2009),foi desenvolvido por Igor Mohor e esta sob licença GLP, podendo ser estudado, modificado e redistribuído (MOHOR 2002).

Este implementa todas as funcionalidades previstas no padrão IEEE 802.3 suportando modos de transmissão *half* e *full-duplex* de 10Mbs e 100Mbs, com suporte a frames de pausa e interface Wishbone para interconexão com o barramento de dados (MOHOR 2002).

Na Figura 26 pode-se observar a estrutura dos módulos implementados no *IP Core* MAC, basicamente constituído por seis submódulos que desempenham funções específicas dentro do projeto (MOHOR 2002).

- eth_registers - Módulo onde estão armazenados todos os registradores de configuração e leitura do *IP-core*. Este módulo por ser bastante estático e de implementação repetitiva (apenas alocação de *flip-flops*) não teve técnicas aplicadas a ele, estimando-se apenas os valores da implementação original (MOHOR 2002).
- eth_macstatus - Responsável por reportar erros e *status* de pronto para o processador através dos registradores e interrupções (MOHOR 2002).

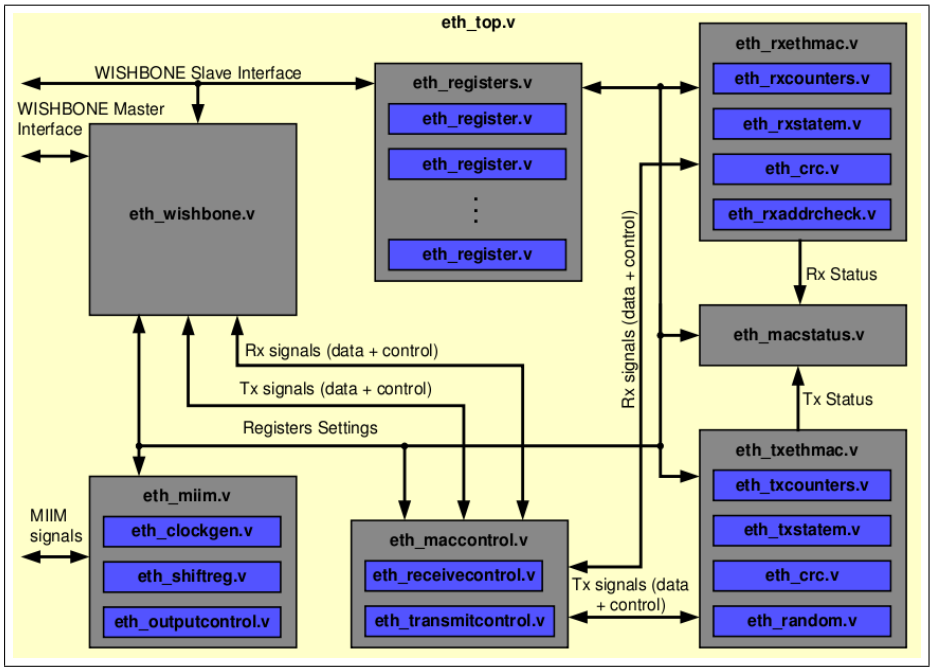

Figura 26: Diagrama de blocos do MAC Opencores (MOHOR 2002)

- eth_maccontol - Gera e detecta quadros de Pause. Utiliza lógica de handshaking com os módulos de transmissão e recepção do core, devido às diferenças de domínio de relógio (MOHOR 2002).

- eth_txethmac - Gera os campos de preâmbulo, SFD e FCS; Gera automaticamente o Inter-Frame Gap (IFG); Complementa automaticamente o tamanho mínimo do quadro transmitido (padding); Detecta colisões e gera o sinal de Jam. Implementa o algoritmo de *Truncated binary exponential backoff* e o processo de deferência. Possui funções adicionais fora do padrão como, habilitar a transmissão de pacotes tipo Jumbo(pacotes de tamanho superior a 1500 bytes (MOHOR 2002).

- eth_rxethmac - Remove os campos de preâmbulo, SFD e FCS; Detecta automaticamente quadros com número não inteiro de octetos ou com quantidade de octetos não compatíveis segundo o padrão IEEE 802.3; Implementa o mecanismo CRC-32 para detecção de erros no quadro recebido; Verifica endereços *Broadcast* e *Multicast*. Também contempla funções adicionais fora do padrão como, o suporte ao modo promíscuo (recepção de pacotes com qualquer endereço MAC) e recepção de pacotes tipo Jumbo de até 64KBytes (MOHOR 2002).

- eth_miim - É um co-processador independente encarregado do gerenciamento do PHY através de um canal bidirecional-serial chamado MDIO (*management data input/output*) e

um sinal de relógio de 2,5MHz chamado MDC (*management data clock*) (MOHOR 2002).

Este *IP Core* possui módulos codificados em Verilog, com um nível de abstração intermediária, o que permitiu a aplicação das diversas técnicas de codificação que visam a otimização do circuito estudadas no capítulo anterior.

Após a aplicação das técnicas a cada um dos módulos foi realizada a simulação funcional para garantir que as novas implementações mantiveram as funcionalidades originais.

3.6 Simulação, *Testbench* e Cobertura de Verificação de Código

O *IP Core* MAC Ethernet, possui juntamente com a descrição do hardware, códigos Verilog que implementam um *testbench* e módulos *stub*(módulos que implementam as funcionalidades necessárias para simular módulos auxiliares, mas não são sintetizáveis) como o PHY e a memória (MOHOR 2002).

Para a utilização do *testbench* algumas modificações foram realizadas a fim de simplificar a simulação, retirando alguns casos específicos de injeção de erros durante os testes. Mesmo com as simplificações, observou-se os testes mínimos necessários para comprovar o funcionamento do módulo de acordo com as cláusulas 31 - *Flow Control Test Suite* (DANCE et al. 2006) e 4 *Media Access Control (MAC) Test Suite* (BRAGA et al. 2006) do *Test Suites* proposto pelo *InterOperability Laboratory - University of New Hampshire*.

Outro parâmetro utilizado para mensurar a eficiência do testbench na simulação foi a cobertura de código na verificação. Para tanto foi utilizado a ferramenta Questa da Mentor Graphics (MENTOR GRAPHICS 2008). Esta ferramenta gera um relatório detalhado de qual a porcentagem do código atingida pelos estímulos gerados no testbench. Obteve-se um alto grau de cobertura de todo o código. A tabela 1 mostra a porcentagem de cobertura de código gerada pelo *tesbench* especificamente para o módulo eth_miim, que é responsável pela comunicação com o PHY e a configuração do mesmo.

Como já foi dito anteriormente após a aplicação de cada uma das técnicas estudadas necessitava-se realizar a verificação do *IP Core* para assegurar que todas as funcionalidades

Arquivos	Estados Ativos	Estados Atingidos	Cobertura %
eth_miim.v	83	83	100
eth_clockgen.v	10	10	100
eth_outputcontrol.v	15	15	100
eth_shiftreg.v	12	12	100
Arquivos	**Desvios Ativos**	**Desvios Atingidos**	**Cobertura %**
eth_miim.v	46	45	97.8
eth_clockgen.v	10	10	100
eth_outputcontrol.v	8	8	100
eth_shiftreg.v	16	16	100
Arquivos	**Condições Ativas**	**Condições Atingidas**	**Cobertura %**
eth_miim.v	18	18	100
eth_clockgen.v	0	0	100
eth_outputcontrol.v	0	0	100
eth_shiftreg.v	0	0	100

Tabela 1: Tabela de estimativas de cobertura do código para o bloco eth_miim gerada pela ferramenta Questa (MENTOR GRAPHICS 2008)

.

foram mantidas com a aplicação das modificações. Essa verificação foi realizada através de simulação funcional, utilizando-se o software Modelsim (MENTOR GRAPHICS 2008) e os estímulos do *testbench* analisado anteriormente.

4 RESULTADOS

Neste capítulo serão analisados os resultados obtidos empregando-se das técnicas de codificação estudadas anteriormente. Cada um dos módulos que fazem parte do *IP Core* MAC, teve seu código de descrição de hardware original analisado visualmente e a cada trecho de código identificou-se as técnicas de codificação possíveis de se aplicar.

As técnicas foram aplicadas visando um único objetivo de otimização. Primeiramente foram codificadas versões com técnicas de melhora do *timing*, de potência e de área, obtendo-se assim como resultado três novas implementações para cada módulo.

Para estas três novas implementações, realizou-se um levantamento da freqüência máxima que o circuito trabalha, número de portas lógicas ocupadas por cada uma das implementações e a estimativa de potência dissipada.

A potência dissipada para cada uma das versões do circuito foi estimada com a ferramenta ADMS da Mentor Graphics, utilizando como estimulo o *testbench* original que esta implementado e disponível junto com o código HDL do *IP Core* MAC, e também com a Ferramenta RTL Compiler da Cadence.

Já o número de gates e a freqüência máxima de funcionamento foi estimada pela ferramenta Leonardo Spectrum da Mentor Graphics e RTL Compiler da Cadence.

Como pode-se observar nas Tabelas 2,3 e 4 as colunas que demonstram os resultados da aplicação das técnicas de codificação visando melhorias individuais para timing, área e potência tiveram uma melhora nos resultados de síntese ainda que pequenas, comparados com os resultados obtidos na implementação original, passou-se para uma segunda etapa da análise dos resultados.

Após a implementação inicial dos três módulos com um único conjunto de técnicas de otimização, foram codificados mais três versões agora combinando-se simultaneamente as técnicas duas a duas, dando origem a mais três versões, uma otimizada para área e tempo, outra otimizada para potência e tempo e uma terceira otimizada para potência e área. Pode-se observar que em alguns casos as implementações se sobrepõem no caso de técnicas de potência e área, e praticamente se anulam em algumas técnicas de otimização de tempo e área ou tempo e potência.

Como última etapa, codificou-se os módulos aplicando-se ao mesmo tempo os três conjuntos de técnicas para realizar uma avaliação dos resultados da aplicação conjunta.

Foram utilizadas três ferramentas para realizar as estimativas. Primeiramente utilizou-se a ferramenta Leonardo Spectrum da Mentor Graphics com a biblioteca de tecnologia TSMC 0.35 microns para realizar o levantamento do número de portas lógicas (área) e o levantamento da freqüência máxima de funcionamento (requisito de tempo). Já para estimativas de potência utilizou-se a ferramenta de simulação mista da Mentor Graphics chamada ADMS e realizou-se a análise DC OP (Análise de potência dissipada em corrente contínua da ferramenta ADMS da fabricante Mentor Graphics).

Também utilizou-se a ferramenta RTL Compiler da empresa Cadence para realizar estimativas a fim de validar os testes de aplicação das técnicas com duas ferramentas diferentes. A tecnologia a qual teve-se acesso para ser utilizada com as ferramentas deste fabricante foi a gscl 0.45 microns proveniente de um *design Kit* acadêmico desenvolvido pela NC State University (CADENCE 2008).

Ambos os designs kits, ADK fornecido pela Mentor Graphics e o NSCU-CDK fornecido pela Cadence / NC State University são de uso estritamente acadêmicos, não sendo empregados em projetos reais, mas totalmente aplicáveis ao estudo em questão

Após o levantamento dos resultados os mesmos foram tabulados em três tabelas, uma tabela com os resultados de potência dissipada, para todos os conjuntos de técnicas aplicados, outra para os resultados de área utilizada e uma outra para a freqüência máxima de funcionamento do circuito. Este processo foi repetido para as duas ferramentas, Leonado Spectrum e ADMS da Mentor Graphics e RTL Compiler da Cadence.

4.1 Resultados Obtidos Empregando-se as Técnicas de Codificação Estudadas e Ferramentas Mentor Graphics

As Tabelas 2, 3 e 4, apresentam respectivamente os resultados da análise de potência (mW), de área utilizada (gates) e requisitos de tempo (MHz) para os módulos na sua codificação original, aplicando-se técnicas de otimização de área (A), potência(P) e tempo(T) bem como as possíveis combinações destes conjuntos de técnicas aplicadas simultaneamente aos módulos.

Nas tabelas são fornecidos os resultados para cada módulo do projeto com a aplicação das técnicas individualmente (somente técnicas de otimização para área, *timing* ou potência) e após em combinações (múltiplos conjuntos de técnicas ao mesmo tempo). Subentende-se individualmente como a aplicação do conjunto de técnicas estudadas especificamente para um propósito específico, tempo, área, ou potência. Já na aplicação das técnicas em combinações, subentende-se a aplicação de dois ou mais conjuntos de técnicas estudadas simultaneamente ao mesmo módulo com o objetivo de estudar a inter-relação das mesmas. A Tabela 2 apresenta os resultados da análise de potência sendo a unidade miliwatt (mw)

—	Original	P	T	A	P e T	P e A	T e A	P, T e A
Mac_control	888	879	889	880	878	877	888	886
Mac_status	653	650	655	649	654	650	652	652
Mac_tx	924	920	929	919	925	921	923	923
Mac_rx	1016	1008	1016	1009	1015	1007	1013	1015
Miim	460	449	462	451	459	450	459	458
Mac_registers	2029	2029	2029	2029	2029	2029	2029	2029
total	5970	5935	5980	5937	5960	5934	5964	5963

Tabela 2: Tabela de Estimativa de Potência em mW utilizando as Ferramentas Mentor Graphics

O mesmo processo de análise empregando-se a aplicação das técnicas individualmente e em combinações foi realizado para a obtenção da Tabela 3, tendo agora como objetivo levantar o número de portas utilizados em cada módulo do projeto com a aplicação das técnicas. A tabela expressa os valores de área em número de portas lógicas (`gates`).

Já a Tabela 4 demonstra os resultados obtidos na análise dos requisitos de tempo dos módulos, subentende-se por requisito de tempo (*timing*) o atraso que um sinal pode possuir desde sua entrada no projeto até atingir seu destino, sendo utilizado essa grandeza para calcular a freqüência máxima a que o circuito pode operar em condições normais.

—	Original	P	T	A	P e T	P e A	T e A	P, T e A
Mac_control	1291	1286	1297	1285	1292	1280	1290	1287
Mac_status	287	281	292	282	289	280	286	289
Mac_tx	1653	1650	1658	1651	1658	1651	1652	1652
Mac_rx	1496	1489	1504	1490	1501	1487	1495	1493
Miim	718	711	724	711	722	711	715	718
Mac_registers	2758	2758	2758	2758	2758	2758	2758	2758
total	8203	8175	8233	8177	8220	8167	8196	8197

Tabela 3: Tabela de Estimativa de Área em *gates* utilizando as Ferramentas Mentor Graphics

Pode-se observar que os resultados estimados para freqüência máxima de trabalho não foram muito afetados com a aplicação das técnicas. Apenas no caso específico de implementação das técnicas de tempo, onde obteve-se variações mínimas como mostra o quadro abaixo

—	Original	T	P	A	P e T	P e A	T e A	P, T e A
Mac_control Mrx_clk	194.3	199.2	194.3	194.3	194.3	194.3	194.3	194.3
Mac_control Mtx_clk	248.7	253.5	248.7	248.7	248.7	248.7	248.7	248.7
Mac_status Mrx_clk	191.4	193.8	191.4	191.4	191.4	191.4	191.4	191.4
Mac_status Mtx_clk	786.9	789.9	786.9	786.9	786.9	786.9	786.9	786.9
Mac_tx	121.9	121.9	121.9	121.9	121.9	121.9	121.9	121.9
Mac_rx	170.3	175.0	170.3	170.3	170.3	170.3	170.3	170.3
Miim	206.0	212.0	206.0	206.0	206.0	206.0	206.0	206.0
Top Level	108.4	108.4	108.4	108.4	108.4	108.4	108.4	108.4

Tabela 4: Tabela de Estimativa de Tempo em MHz utilizando as Ferramentas Mentor Graphics

Pode-se observar na tabela de requisitos de tempo a linha *Top-Level* que não apresentou variação alguma. Ela representa o *clock* máximo de funcionamento do projeto completo. Assim sendo pode-se verificar que nem sempre a melhora de requisitos de *timing* de um módulo necessariamente vá melhorar o projeto como um todo, pois é necessário nivelar o requisito pelo pior resultado para que todos possam ser atendidos.

4.2 Resultados Obtidos empregando-se as Técnicas de Codificação Estudadas e Ferramentas Cadence

Empregando-se as mesmas técnicas utilizadas na análise anterior, apenas utilizando ferramentas da empresa Cadence Inc. obteve-se as tabelas a seguir .

As tabelas 5, 6 e 7 apresentam respectivamente os resultados da análise de potência (mw),

área utilizada (gates) e tempo (MHz) para os módulos na sua codificação original, aplicando-se técnicas de otimização de área (A), potência (P) e tempo (T) bem como as possíveis combinações destes conjuntos de técnicas aplicadas simultaneamente aos módulos. A Tabela 5 apresenta os resultados da análise de potência sendo a unidade miliwatt (mW).

—	Original	P	T	A	P e T	P e A	T e A	P, T e A
Mac_control	52,8	50,49	56,21	50,55	51,99	50,45	52,06	54,47
Mac_status	92,64	91,02	95,1	91,03	92,53	91,02	92,47	91,83
Mac_tx	61,06	59,64	63,25	59,53	62,1	59,5	62,08	61
Mac_rx	54,73	52,75	58,36	52,64	56,37	52,63	54,82	53,77
Miim	38,93	35,89	40,01	35,68	38,93	35,4	39,01	39,99
Mac_registers	154,64	154,64	154,64	154,64	154,64	154,64	154,64	154,64
total	454,8	444,43	467,57	444,07	456,56	443,64	455,08	455,7

Tabela 5: Tabela de Estimativa de Potência em mW das Ferramentas Cadence

Como anteriormente descrito, o mesmo processo de análise empregando-se a aplicação das técnicas individualmente e em combinações foi realizado para a obtenção da Tabela 6, tendo agora como objetivo levantar o número de portas utilizados em cada módulo do projeto com a aplicação das técnicas.

—	Original	P	T	A	P e T	P e A	T e A	P, T e A
Mac_control	1452	1446	1467	1444	1455	1444	1455	1448
Mac_status	306	299	317	297	310	298	311	302
Mac_tx	1741	1736	1759	1735	1748	1724	1749	1740
Mac_rx	1618	1609	1626	1601	1625	1609	1624	1615
Miim	806	792	821	793	813	791	822	801
Mac_registers	2386	2386	2386	2386	2386	2386	2386	2386
total	8309	8268	8376	8256	8337	8252	8347	8292

Tabela 6: Tabela de Estimativa de Área em *gates* das Ferramentas Cadence

Para o levantamento dos requisitos de tempo utilizando a ferramenta RTL Compiler da Cadence, os resultados são dados por caminhos (*paths*) e não por domínios de *clock* como na ferramenta da Mentor Graphics, sendo assim, utilizou-se o pior resultado de um *path* que pertence ao mesmo domínio de *clock* para calcular a freqüência máxima do circuito. A tabela 7 mostra os resultados obtidos.

—	Original	T	P	A	P e T	P e A	T e A	P, T e A
Mac_control Mrx_clk	201.5	201.5	201.5	201.5	201.5	201.5	201.5	201.5
Mac_control Mtx_clk	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0
Mac_status Mrx_clk	204.3	204.3	204.3	204.3	204.3	204.3	204.3	204.3
Mac_status Mtx_clk	799.8	799.8	799.8	799.8	799.8	799.8	799.8	799.8
Mac_tx	130.8	130.8	130.8	130.8	130.8	130.8	130.8	130.8
Mac_rx	175.0	175.0	175.0	175.0	175.0	175.0	175.0	175.0
Miim	209.1	209.1	209.1	209.1	209.1	209.1	209.1	209.1
Top Level	115.0	115.0	115.0	115.0	115.0	115.0	115.0	115.0

Tabela 7: Tabela de Estimativa de Tempo em MHz utilizando as Ferramentas Cadence

4.3 Comparação dos Resultados Obtidos com Ferramentas Mentor Graphics e Cadence

Como foram utilizadas duas tecnologias diferentes na realização as estimativas, para efetivar-se uma comparação foi necessário utilizar valores percentuais normalizados, usando como padrão o valor estimado obtido na implementação original (Valor de referência) .

4.3.1 Resultados de Área

A Tabela 8 apresenta os resultados percentuais positivos em caso de aumento de área e negativo no caso de diminuição da área. Ela expressa os valores de aumento ou diminuição de área percentualmente (*gates*) para os resultados obtidos utilizando-se a ferramenta Leonardo Spectrum. Para uma melhor visualização e análise comparativa também é apresentado na Figura 27 o gráfico resultante dos dados tabulados.

—	Original	P	T	A	P e T	P e A	T e A	P, T e A
Mac_control	0	-0,39	0,46	-0,47	0,07	-0,86	-0,08	-0,31
Mac_status	0	-3,1	1,74	-1,75	0,69	-2,44	-0,45	0,69
Mac_tx	0	-0,19	0,3	-0,13	0,32	-0,13	0,07	0,07
Mac_rx	0	-0,44	0,53	-0,41	0,33	-0,61	-0,07	-0,21
Miim	0	-0,98	0,83	-0,98	0,55	-0,98	-0,42	0
Mac_registers	0	0	0	0	0	0	0	0
total	0	-0,45	0,32	0,32	00,2	-0,44	-0,09	-0,08

Tabela 8: Tabela de Estimativa de Área em Porcentual das Ferramentas Mentor Graphics

A tabela 9 expressa os valores de aumento ou diminuição de área percentualmente (*gates*) para os resultados obtidos utilizando-se a ferramenta RTL Compiler.Para uma melhor visualiza-

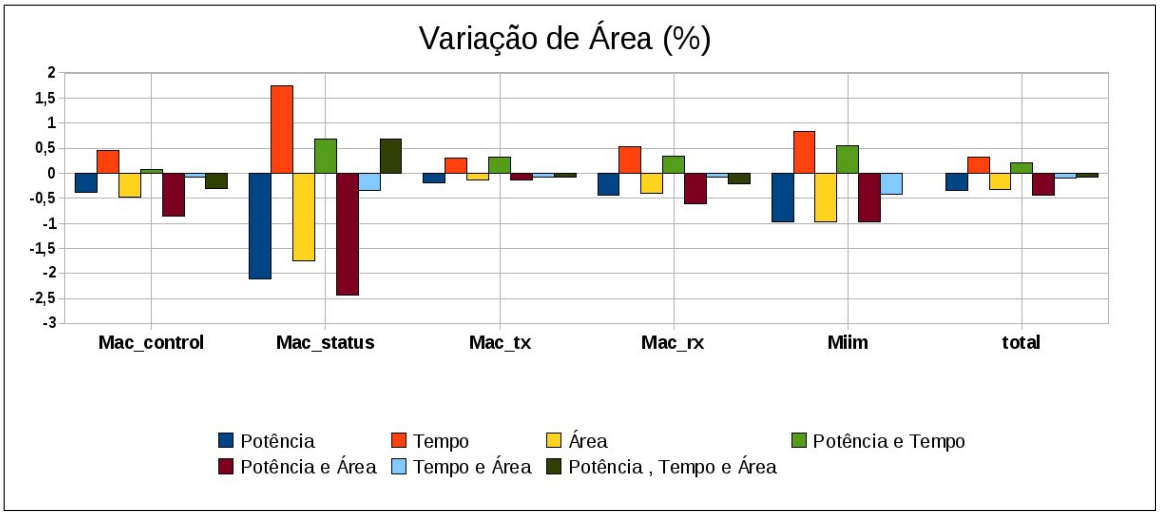

Figura 27: Gráfico de variação de área em percentual com Ferramenta Mentor Graphics

ção e análise comparativa também é apresentado no gráfico da Figura 27.

—	Original	P	T	A	P e T	P e A	T e A	P, T e A
Mac_control	0	-0,42	1,03	-0,66	00,2	-0,66	0,2	-0,28
Mac_status	0	-2,29	3,59	-2,95	1,3	-2,62	1,63	-1,24
Mac_tx	0	-0,39	1,03	-0,45	0,4	-0,98	0,45	-0,06
Mac_rx	0	-0,66	0,49	-1,06	0,43	-0,67	0,37	-0,21
Miim	0	-1,74	1,86	-1,62	0,86	-1,77	1,98	-0,77
Mac_registers	0	0	0	0	0	0	0	0
total	0	-0,5	0,8	-0,74	0,33	-0,79	0,45	-0,21

Tabela 9: Tabela de Estimativa de Área em Porcentual das Ferramentas Cadence

4.3.2 Resultados de Potência

A tabela 10 expressa os valores de aumento ou diminuição de potência percentualmente (mw), para os resultados obtidos utilizando-se a ferramenta RTL Compiler.Para uma melhor visualização e análise comparativa também é apresentado no gráfico da Figura 29.

A tabela 11 expressa os valores de aumento ou diminuição de potência percentualmente (mw), para os resultados obtidos utilizando-se a ferramenta Leonardo Spectrum. Para uma melhor visualização e análise comparativa resultante dos valores de potência também é apresentado no gráfico da Figura 30.

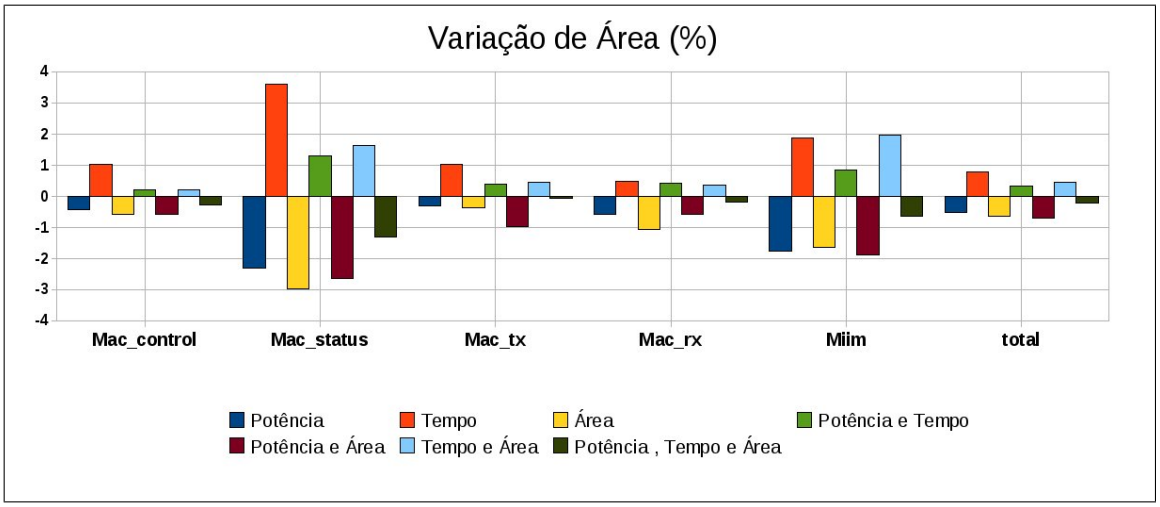

Figura 28: Estimativas de variação de área em percentual com Ferramenta Cadence

—	Original	P	T	A	P e T	P e A	T e A	P, T e A
Mac_control	0	-4,45	6,45	-4,23	-1,64	-4,45	-1,41	3,16
Mac_status	0	-8,98	-4,9	-8,97	-7,47	-8,98	-7,53	-7,47
Mac_tx	0	-2,43	3,58	-2,51	1,7	-2,66	1,67	-0,1
Mac_rx	0	-3,62	6,63	-3,65	2,99	-2,01	0,16	-1,76
Miim	0	-7,81	2,77	-8,45	0	-9,07	0,2	2,72
Mac_registers	0	0	0	0	0	0	0	0
total	0	-2,29	2,8	-2,46	0,38	-2,46	0,06	0,19

Tabela 10: Tabela de Estimativa de Potência em Porcentagem das Ferramentas Cadence (%)

4.4 Discussão dos Resultados

Analisando as tabelas apresentadas na seção anterior e observando o gráfico da Figura 31 onde é demonstrado a variação total expressa em porcentagem de potência e área para o emprego de cada conjunto de técnicas, é possível notar que existem características semelhantes para as técnicas empregadas mesmo utilizando-se tecnologias completamente diferentes.

Nota-se também que para o módulo mac_registers não existe variação dos valores de potência, tempo e área, isso devido ao fato de não se aplicar nenhuma das técnicas estudadas a tal módulo. Este módulo é composto basicamente de registradores instanciados, não cabendo a aplicação de nenhuma das técnicas utilizadas. Para este tipo de módulo, otimizações podem ser feitas no momento de especificação do projeto (número de registradores e tecnologia empregada), mas como o escopo do trabalho trata das técnicas de codificação visando otimizações na síntese, não se abordará otimizações na especificação de módulos.

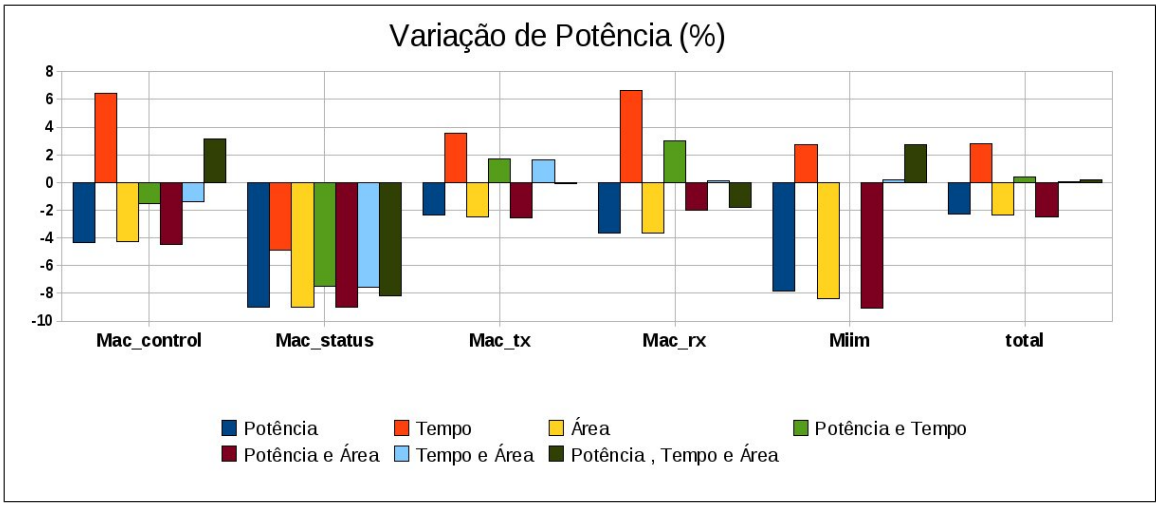

Figura 29: Estimativas de Potência com Ferramenta Cadence (%)

—	Original	P	T	A	P e T	P e A	T e A	P, T e A
Mac_control	0	-1,02	0,11	-0,91	-1,23	-1,24	0	-0,33
Mac_status	0	-1,46	0,3	-0,62	0,15	-0,66	-0,16	-0,16
Mac_tx	0	-0,44	0,54	-0,65	0,1	-0,33	-0,11	-0,11
Mac_rx	0	-0,79	0	-0,69	-0,01	-0,89	-0,3	-0,01
Miim	0	-2,4	0,43	-1,96	-0,22	- 2,18	-0,22	-0,44
Mac_registers	0	0	0	0	0	0	0	0
total	0	-0,59	0,16	-0,63	-0,17	-0,61	-0,11	-0,11

Tabela 11: Tabela de Estimativa de Potência em Porcentagem das Ferramentas Mentor (%)

4.4.1 Técnicas Aplicadas no módulo *maccontrol*

Este módulo é composto por três arquivos: (eth_maccontrol, eth_receivecontrol e eth_transmitcontrol) é o módulo que faz o controle dos demais módulos para transmissão e recepção de pacotes quando em modo *full duplex*; composto basicamente por multiplexadores síncronos e assíncronos, máquinas de estado e lógica combinacional.

Analisando-se a codificação original do módulo, foi possível aplicar a técnica de simplificação lógica e extração de multiplexadores desnecessários para a redução de área ocupada. No Código fonte 1 pode-se visualizar um trecho com a codificação original do *IP Core* utilizado, já no código Fonte 2 pode-se visualizar o código HDL com a aplicação da técnica de redução de área conhecida como eliminação de MUX. Geralmente ferramentas de síntese lógica interpretam seqüencias de *if* e *else* aninhados como multiplexadores com prioridade, eliminando-se uma destas checagens condicionais, elimina-se também um multiplexador do circuito, mantendo-se a sua funcionalidade original.

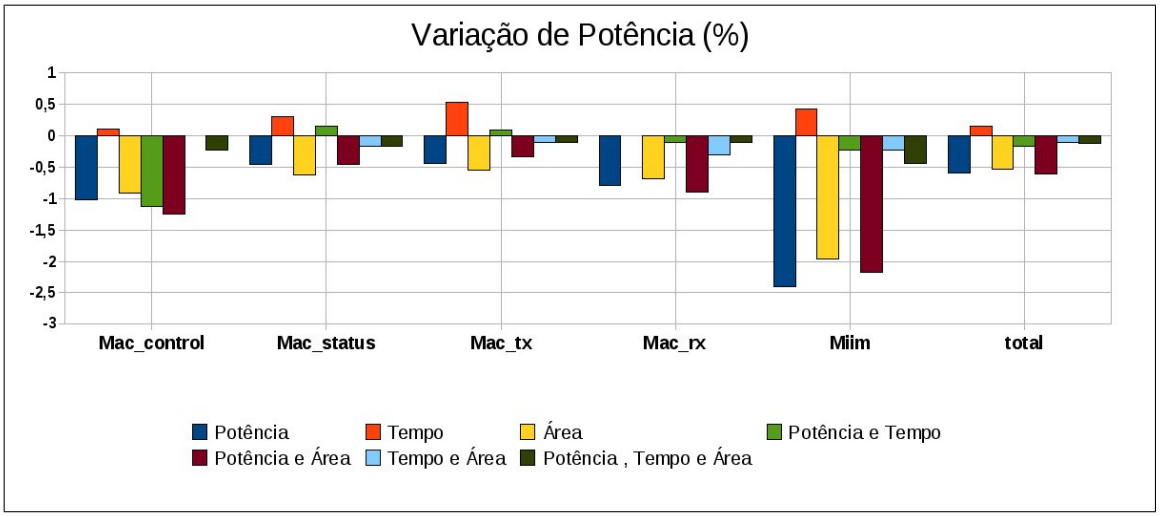

Figura 30: Estimativas de Potência com Ferramenta Mentor (%)

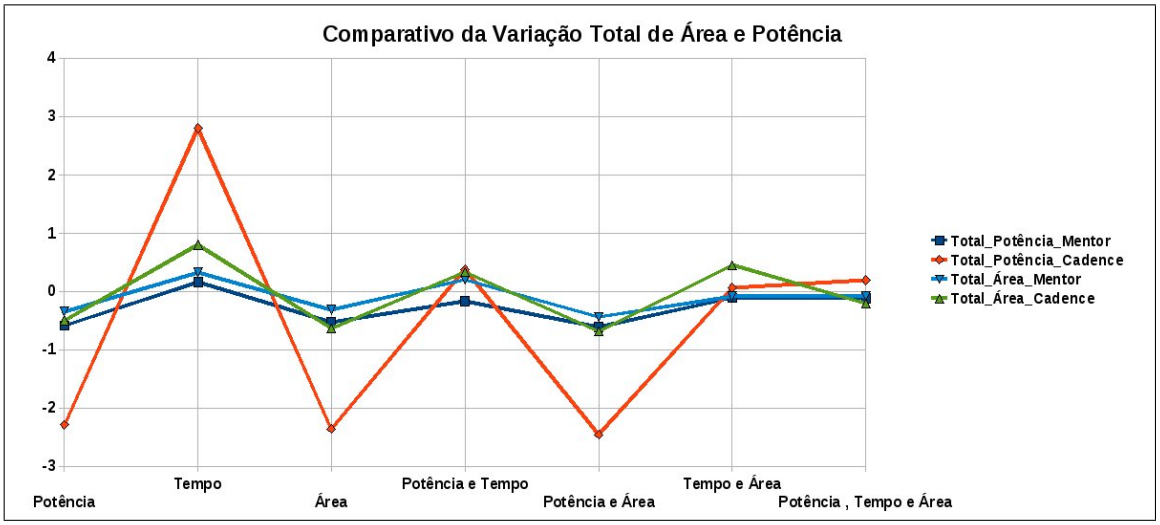

Figura 31: Comparativo dos Totais de variação de área e potência (%)

Como resultado da síntese lógica da codificação original obteve-se uma redução pequena na área e também na potência total dissipada, já os otimizações de tempo não surtiram os resultados esperados. Constatou-se que existiram melhorias de tempo em partes do circuito, mas o tempo total não foi modificado devido ao fato de muitas vezes estas melhorias não sobrecairem no caminho crítico do circuito.

4.4.2 Técnicas Aplicadas no módulo *macstatus*

Este módulo é composto por um único arquivo (eth_macstatus), é o módulo responsável pelo controle de status do mac ethernet. Ao analisar-se a codificação original do módulo, foi possível aplicar a técnica de simplificação lógica mais especificamente a redução de MUX para

```
1
2                   ...
3
4   assign ResetByteCnt = RxEndFrm;
5   assign IncrementByteCnt = RxValid & DetectionWindow & ~ByteCntEq18 & (~DlyCrcEn | DlyCrcEn & DlyCrcCnt[2]);
6
7
8   // Byte counter
9   always @ (posedge MRxClk or posedge RxReset)
10  begin
11    if(RxReset)
12      ByteCnt[4:0] <= #Tp 5'h0;
13    else
14    if(ResetByteCnt)
15      ByteCnt[4:0] <= #Tp 5'h0;
16    else
17    if(IncrementByteCnt)
18      ByteCnt[4:0] <= #Tp ByteCnt[4:0] + 1'b1;
19  end
20
21                  ...
```

Código 1: Trecho com codificação original do módulo

```
1
2                   ...
3
4   assign ResetByteCnt = RxEndFrm;
5   assign IncrementByteCnt = RxValid & DetectionWindow & ~ByteCntEq18 & (~DlyCrcEn | DlyCrcEn & DlyCrcCnt[2]);
6
7
8   // Byte counter
9   always @ (posedge MRxClk or posedge RxReset)
10  begin
11    if(RxReset |ResetByteCnt)
12      ByteCnt[4:0] <= #Tp 5'h0;    ///// EliminaÃ§Ã£o de um mux desnecessÃ¡rio
13    else
14    if(IncrementByteCnt)
15      ByteCnt[4:0] <= #Tp ByteCnt[4:0] + 1'b1;
16  end
17
18                  ...
```

Código 2: Trecho com codificação aplicando-se técnica de redução de área

a redução de área ocupada, a técnica de redução de níveis lógicos para *timing* e o chaveamento de *clock* para a redução de potência dinâmica dissipada.

Como resultado da síntese lógica da codificação original obtivemos a área acumulada de 287 instancias de células padrão da tecnologia de fabricação tsmc 0.35 microns. O Relatório da ferramenta Leonardo Spectrum apresentado abaixo detalha as celulas empregadas na síntese lógica antes e depois da aplicação das técnica simplificação lógica e extração de MUX para otimização de área

4.4.3 Técnicas Aplicadas no módulo miim

Este módulo é composto por três arquivos, (eth_shiftregister, eth_outputcontrol e eth_clockgen) é o módulo que faz o controle e configuração do chip PHY que faz a comunicação física com o meio de transmissão.

```
ImplemetaÃ§Ã£o Original                                    ImplementaÃ§Ã£o Otimizada

******************************************************** ************************************************************

Cell: eth_macstatus    View: INTERFACE    Library: work Cell: eth_macstatus    View: INTERFACE    Library: work

******************************************************** ************************************************************


 Cell       Library  References     Total Area          References     Total Area

and03       tsmc035_typ     1 x      2      2 gates          1 x         2      2 gates
ao32        tsmc035_typ     1 x      2      2 gates          1 x         2      2 gates
aoi21       tsmc035_typ     2 x      1      2 gates          2 x         1      2 gates
aoi22       tsmc035_typ     1 x      1      1 gates          1 x         1      1 gates
dffr        tsmc035_typ    17 x      6     95 gates         15 x         6     95 gates
dffs_ni     tsmc035_typ     1 x      6      6 gates          1 x         6      6 gates
inv01       tsmc035_typ    11 x      1      8 gates         11 x         1      8 gates
mux21_ni    tsmc035_typ    37 x      2     67 gates         37 x         2     67 gates
nand02      tsmc035_typ     2 x      1      2 gates          2 x         1      2 gates
nand03      tsmc035_typ     1 x      1      1 gates          1 x         1      1 gates
nand04      tsmc035_typ     2 x      1      3 gates          2 x         1      3 gates
nor02_2x    tsmc035_typ     3 x      1      3 gates          3 x         1      3 gates
nor02ii     tsmc035_typ     4 x      1      5 gates          4 x         1      5 gates
nor03_2x    tsmc035_typ     5 x      1      6 gates          5 x         1      6 gates
nor04       tsmc035_typ     1 x      1      1 gates          1 x         1      1 gates
oai21       tsmc035_typ     6 x      1      7 gates          6 x         1      7 gates
oai32       tsmc035_typ     1 x      2      2 gates          1 x         2      2 gates
oai33       tsmc035_typ     1 x      2      2 gates          1 x         2      2 gates
or03        tsmc035_typ     1 x      2      2 gates          1 x         2      2 gates
xnor2       tsmc035_typ    35 x      2     67 gates         32 x         2     67 gates
xor2        tsmc035_typ     1 x      2      2 gates          1 x         2      2 gates

 Number of ports :                   113              Number of ports :                   113
 Number of nets :                    231              Number of nets :                    231
 Number of instances :               134              Number of instances :               134
 Number of references to this view :   0              Number of references to this view :   0

Total accumulated area :                             Total accumulated area :
 Number of gates :                   287              Number of gates :                   282
 Number of accumulated instances :   134              Number of accumulated instances :   129
```

Código 3: Relatório da Ferramenta Leonardo Spectrum especificando as células padrões empregadas na síntese da codificação original e com aplicação da otimização de área

Analisando-se a codificação original do módulo, foi possível aplicar a técnica de simplificação lógica para a redução de área ocupada, a técnica de redução de níveis lógicos para *timing* e o chaveamento de *clock* e separação de *nets* com alta taxa de comutação para a redução de potência dinâmica dissipada.

4.4.4 Técnicas Aplicadas no módulo *registers*

Este módulo é composto por um arquivo (eth_registers) que faz o controle dos demais módulos, ele é composto basicamente por registradores, multiplexadores síncronos e assíncronos e lógica combinacional.

Analisando-se a codificação original do módulo, não foi possível aplicar técnicas de simplificação lógica devido ao fato do módulo ser composto basicamente por *flip-flops* instanciados, sendo um módulo muito estático e quase sem possibilidades de aplicação de técnicas de otimização a nível de descrição de hardware.

4.4.5 Técnicas Aplicadas no módulo *rxethmac*

Este módulo é composto por quatro arquivos, é responsável por faz a recepção de pacotes vindos do meio físico, é composto basicamente por multiplexadores síncronos e assíncronos, máquinas de estado e lógica combinacional.

Analisando-se a codificação original do módulo, foi possível aplicar a técnica de simplificação lógica mais especificamente a redução de MUX para a redução de área ocupada, a técnica de redução de níveis lógicos para *timing* e o chaveamento de *clock* para a redução de potência dinâmica dissipada.

4.4.6 Técnicas Aplicadas no módulo *txethmac*

Este módulo é composto por quatro arquivos, é o módulo que faz a transmissão de pacotes, é composto basicamente por multiplexadores síncronos e assíncronos,registradores, *fifos* máquinas de estado e lógica combinacional.

Analisando-se a codificação original do módulo, foi possível aplicar a técnica de simplificação lógica, mais especificamente a redução de MUX para a redução de área ocupada, a técnica de redução de níveis lógicos para *timing* e o chaveamento de *clock* para a redução de potência dinâmica dissipada.

4.4.7 Aplicação Simultânea das Técnicas ao Estudo de Caso

Pode-se observar nas tabelas e gráficos demonstrados neste capítulo que a aplicação dos três conjuntos de técnicas simultaneamente não surtiu um resultado satisfatório, demonstrando que as ferramentas utilizadas conseguem otimizar buscando o melhor resultado, balanceando os quesitos tempo, área e potência.

Na aplicação de técnicas individuais pode-se observar uma melhora nos resultados principalmente em relação a área. A melhor combinação visualizada é a de técnicas de redução de área e redução de potência, levando algumas vezes a uma redução de área maior do que se obteve somente com as técnicas de redução de área.

Já ao aplicar-se uma técnica de redução de tempo, que pelos resultados vistos não incrementam em muito a freqüência do circuito,a área geralmente aumenta, pois mais portas lógicas são necessárias para realizar a mesma função em um tempo menor.

CONCLUSÃO

Ao final deste estudo onde foi realizada a busca e classificação, estudo e implementação de algumas técnicas e estilos de codificação aplicados a linguagem de descrição de hardware Verilog. Buscou-se com estas técnicas de codificação sempre uma otimização e melhoria dos resultados de síntese lógica, tanto para a redução da área de silício utilizada (número de portas) e para a potência dissipada, como para os requisitos de tempo (freqüência máxima de trabalho) dos circuitos; pode-se verificar algumas observações importantes deixadas como contribuição do estudo realizado.

Estas observações são válidas especificamente para o estudo de caso aqui realizado. Não podendo-se garantir, apenas supor, que os resultados obtidos aqui serão similares em outros cenários; para tanto seria necessário um estudo de maior porte onde diversos projetos tivessem seus códigos analisados, e a eles fossem aplicadas as técnicas e os resultados inquiridos.

A maioria das técnicas demonstradas nos capítulos anteriores são de fácil aplicação se forem aplicadas a uma codificação RTL estrutural ou funcional, já sua aplicação a codificações comportamentais torna-se difícil, pois não se tem um controle tão preciso da síntese lógica.

Atenção especial deve-se dispensar para técnicas que utilizam habilitação e desabilitação de partes ou blocos do circuito projetado. É necessário a implementação de mecanismos que garantam o sincronismo com o *clock* no momento de habilitação/desabilitação para prevenir metaestabilidade nos circuitos.

Basicamente os três conjuntos de técnicas de otimização estudadas tentam fazer redução de gates e chaveamentos para diminuir a potência dissipada e área de silício utilizada, e reduzir através de simplificações ou divisão do circuito o número de níveis lógicos (portas lógicas encadeadas) e conseqüentemente os atrasos que os sinais elétricos sofrem para ter um ganho na freqüência máxima de trabalho do circuito .

Como já foi dito anteriormente, as técnicas de otimização para redução de potência e área, muitas vezes tem seus resultados sobrepostos ou somados, pois trabalham basicamente com o mesmo princípio de diminuição de portas lógicas para a diminuição de área e potência estática e diminuição de chaveamentos para a diminuição do potência dinâmica. Já as técnicas de melhoramento dos requisitos de tempo dos circuitos são inversamente proporcionais as técnicas de potência e tempo. Para realizar a diminuição de níveis lógicos geralmente é necessária a adição de mais lógica para a divisão do caminho dos sinais, o que gera quase sempre um maior consumo e aumento de área de silício.

As ferramentas de síntese testadas: Leonardo Spectrum da empresa Mentor Graphics e RTL Compiler da Cadence, demonstraram um elevado índice de otimizações automáticas no processo de síntese dos circuitos testados, principalmente por conter lógica combinacional, onde os algoritmos de simplificação das ferramenta possuem uma evolução maior. Mesmo com esse elevado índice de otimização das ferramentas há outro fator importante, trata-se do código escolhido que já apresentava uma codificação otimizada; observou-se que a aplicação das técnicas surtiu alguns resultados positivos; ainda que pequenos, mas se forem considerados em um projeto de maior porte e visando atingir requisitos de potência e área mínimos, demonstram que as técnicas de otimização estudadas são viáveis de aplicação,principalmente as de otimização área e potência em conjunto, já que os resultados não foram significativos implementado-se o conjunto das três técnicas simultaneamente na codificação e na implementação pois, quase nunca alteraram-se os requisitos de tempo do circuito.

Por fim é possível afirmar que ainda hoje com ferramentas de síntese automática cada vez mais inteligentes, quanto mais baixo o nível de abstração da codificação, maior o controle da síntese lógica e uma melhor otimização é conseguida.

Trabalhos Futuros

Faz-se uma sugestão para a continuação desta linha de estudo, envolvendo linguagem de descrição de hardware (HDL) e o impacto de seus estilos de codificação na síntese lógica, buscando-se sempre uma diminuição de área no arranjo dos transistores, diminuição da potência dissipada e um aumento da freqüência máxima de trabalho permitida pelo caminho crítico do circuito (CADENCE 2004).

Como foi visto nos resultados obtidos e demonstrados nas seções anteriores, a aplicação das técnicas de codificação selecionadas e classificadas no trabalho, causou uma pequena melhora no consumo de potência e área, em média menos de 1%, o que pode não ser muito significativo para um projeto pequeno como o estudo de caso analisado, mas para projetos maiores acredita-se que a aplicação destas mesmas técnicas surtirá um resultado ainda melhor.

Para tornar mais completo este estudo propõe-se testar o poder de síntese das ferramentas de outros fabricantes, diferentes dos utilizados neste estudo tais como Synopsys©e outras.

Seguindo a mesma linha da pesquisa, um outro estudo a ser realizado é a codificação, utilizando mais alto nível de abstração na descrição HDL, modelando comportamentalmente o circuito e não funcionalmente (*flops, mux* ...). Realizou-se um experimento inicial com o módulo MII responsável pela interface de configuração do PHY da rede.Codificou-se de maneira comportamental e não estrutural como a implementação original do módulo, utilizando-se apenas primitivas *if else*, *while* e *for*, não sendo instanciado componentes previamente codificados como multiplexadores e *flip-flops*. Realizou-se a simulação do projeto com a substituição do módulo MII original pelo codificado comportamentalmente e verificou-se sua funcionalidade igual a esperada.Os resultados preliminares da síntese lógica demonstraram uma diferença irrelevante (menores que 0.5%) tanto na modificação da área utilizada, potência dissipada, quanto no requisito de tempo.

Mas apenas com este teste preliminar não é possível chegar e nenhuma conclusão; ainda sendo necessário realizar um estudo mais completo para comprovar as reais implicações da codificação em um nível de abstração mais alto na síntese. Só assim será possível comprovar contribuição como a dada aqui neste estudo sobre a aplicação de técnicas de codificação simples na codificação HDL.

Este estudo, como já foi visto durante o decorrer do texto, foi realizado em linguagem Verilog; também é possível ampliar o estudo utilizando-se outras linguagem de descrição de *hardware* como VHDL e System C, buscando-se sempre uma otimização nos resultados da síntese.

REFERÊNCIAS

[ASHENDEN 2002]ASHENDEN, P. J. *The designers guide to VHDL*. [S.l.]: Morgan Kaufmann, 2002.

[BAMPI 2005]BAMPI, S. Tendências tecnológicas e oportunidades para a indústria de componentes semicondutores no brasil - ufrgs. *UFRGS*, Porto Alegre, RS, 2005.

[BRAGA et al. 2006]BRAGA, A. et al. *ETHERNETS - Clause 4 - Media Access Control (MAC) Test Suite*. Durham, NH, 2006. Disponível em: <ftp://ftp.iol.unh.edu/pub/ethernet/test_suites/MAC/MAC_Test_Suite_v4.6.pdf>.

[BROWN e ZVONKO 2003]BROWN, S. D.; ZVONKO, V. G. *Fundamentals of digital logic with verilog design*. [S.l.]: Mc. Graw Hill, 2003.

[CADENCE 2004]CADENCE. System ic reference flow. *Cadence White Papers 2008*, San Jose, CA, 2004.

[CADENCE 2007]CADENCE. Architecting, designing, implementing, and verifying low-power digital integrated circuits. *Cadence White Papers 2007*, San Jose, CA, 2007.

[CADENCE 2008]CADENCE. *Encounter RTL Compiler*. [S.l.], 2008. Disponível em: <http://www.cadence.com/eu/Pages/rtl_compiler.aspx>.

[CILETTI 2002]CILETTI, M. D. *Advanced digital design with the verilog HDL*. [S.l.]: Prentice Hall, 2002.

[DAHAN 2007]DAHAN, N. *Logic Replication*. [S.l.], 2007. Disponível em: <http://asicdigitaldesign.wordpress.com/2007/07/25/replication/>.

[DAHAN 2007]DAHAN, N. Low power techniques - reducing switching. *Adventures in Asic: Digital Design*, 2007. Disponível em:

<http://asicdigitaldesign.wordpress.com/2007/06/15/low-power-techniques-reducing-switching/>.

[DANCE et al. 2006]DANCE, R. et al. *ETHERNETS - Clause 31 - Flow Control Test Suite*. Durham, NH, 2006. Disponível em: <ftp://ftp.iol.unh.edu/pub/ethernet/test_suites/CL31_FC/flow_control_testsuite_v1.3.pdf>.

[ECONÔMICO 2009]ECONÔMICO, V. Ceitec: Crise traz oportunidade para a área de microeletrônica. *Valor Econômico*, v. 1, 2009.

[GAJSKI 1997]GAJSKI, D. D. *Principles of digital design*. [S.l.]: Prentice Hall, 1997.

[HACHTEL e SOMENZI 2006]HACHTEL, G. D.; SOMENZI, F. *Logic Synthesis and verification algorithms*. [S.l.]: Springer, 2006.

[HENNESSY e PATTERSON 2005]HENNESSY, J. L.; PATTERSON, D. A. *Computer organization and design:the hardware-software interface*. [S.l.]: M. Kaufmann, 2005.

[HORNA et al. 2007]HORNA, C. T. et al. Implementação e validação de ip soft cores para interfaces ethernet 10/100 e 1000 mbps sobre dispositivos reconfiguráveis. *Iberchip*, Lima, Peru, 2007.

[HURST 1998]HURST, S. *Vlsi Testing: Digital and Mixed Analogue/Digital Techniques*. [S.l.]: Iet, 1998.

[IEEE 2002]IEEE. *IEEE Std 802.3*. New York, NY, 2002.

[IEEE 2004]IEEE. *Behavioural languages Part 1: VHDL language reference manual. IEEE Standard No. 61691-1-1*. [S.l.], 2004.

[IEEE 2005]IEEE. *Verilog Hardware Description Language.IEEE Standard No. 1364*. [S.l.], 2005.

[MENTOR GRAPHICS 2008]MENTOR GRAPHICS. *Leonardo Spectrum*. [S.l.], 2008. Disponível em: <http://www.mentor.com/products/fpga/synthesis/leonardo_spectrum/upload/datasheet.pdf>.

[MENTOR GRAPHICS 2008]MENTOR GRAPHICS. *ModelSim*. [S.l.], 2008. Disponível em: <http://www.mentor.com/products/fv/modelsim/>.

[MENTOR GRAPHICS 2008]MENTOR GRAPHICS. *QuestaSim Code Coverage*. [S.l.], 2008. Disponível em: <http://www.mentor.com/products/fv/questa/>.

[METCALFE e D.BOGGS 1976]METCALFE, M.; D.BOGGS. Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM, Vol. 19*, 1976.

[MICHELI 2004]MICHELI, G. D. *Synthesis and optimization of digital circuits*. [S.l.]: New York: McGraw Hill, 2004.

[MILLS 2007]MILLS, S. S. D. Gotcha again more subtleties in the verilog and systemverilog standards that every engineer should know. *SNUG - Synopsys Users Group Conference*, San Jose, CA, 2007.

[MOHOR 2002]MOHOR, I. *Eth Design Document*. [S.l.], 2002. Disponível em: <http://www.opencores.org/cvsweb.shtml/ethernet/doc/eth_design_document.pdf>.

[MOHOR 2002]MOHOR, I. *Eth Specifications*. [S.l.], 2002. Disponível em: <http://www.opencores.org/cvsweb.shtml/ethernet/doc/eth_speci.pdf>.

[MOHOR 2002]MOHOR, I. *Ethernet Datasheet OC Head*. [S.l.], 2002. Disponível em: <http://www.opencores.org/cvsweb.shtml/ethernet/doc/ethernet_datasheet_OC_head.pdf>.

[MOHOR 2002]MOHOR, I. *Ethernet Product Brief OC Head*. [S.l.], 2002. Disponível em: <http://www.opencores.org/cvsweb.shtml/ethernet/doc/ethernet_product_brief_OC_head.pdf>.

[OPENCORES.ORG 2009]OPENCORES.ORG. *Repositório de IP Cores livres*. [S.l.], 2009. Disponível em: <http://opencores.org>.

[RABAEY 2003]RABAEY, J.; CHANDRAKASAN, A.; NIKOLIC, B. *Digital integrated circuits:A design perspctive*. [S.l.]: Pearson, 2003.

[REIS 2003]REIS, R. *Concepção de Circuitos Integrados*. [S.l.]: Editora da UFRGS, 2003.

[SEDRA e SMITH 2004]SEDRA, A. S.; SMITH, K. *Microelectronic circuits*. [S.l.]: Oxford University Press, 2004.

[TANENBAUM 2003]TANENBAUM, A. S. *Computer Network*. [S.l.]: Prentice-Hall, 2003.

[TECNOLOGIA 2003]TECNOLOGIA, M. D. C. E. *Programa Nacional de Microeletrônica*. [S.l.]: MCT, 2003.

[TECNOLOGIA 2001]TECNOLOGIA, M. da Ciência e. Programa nacional de microeletrônica : Design, atração, fixação e crescimento de empresas de projeto de componentes. *Palestra*, 2001. Disponível em: <http://ftp.mct.gov.br/Temas/info/palestras/ProgrMic.pdf>.

[TOZETTO 2007]TOZETTO, E. H. *Automação do fluxo de projeto de circuitos integrados através do desenvolvimento de uma interface gráfica paramétrica implementada utilizando-se pacotes TCL/TK*. Dissertação (Dissertação) — Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação, Campinas, SP, jul. 2007.

[WANG 2006]T.WANG, L.; WU, C.W.; WEN, X. *VLSI Test Principles and Architectures: Design for Testability (Systems on Silicon)*. [S.l.]: Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 2006.

[UYEMURA 2002]UYEMURA, J. P. *Sistemas Digitais*. [S.l.]: Thomson, 2002.

[VAHID e GIVARGIS 2002]VAHID, F.; GIVARGIS, T. *Embedded System Design: A Unified Hardware Software Introduction*. [S.l.]: John Wiley Sons, 2002.

[WEST e ESHRAGHIAN 1993]WEST, N. H. E.; ESHRAGHIAN, K. *Principles of CMOS VLSI design:a systems perspective*. [S.l.]: Addison Wesley, 1993.

[WEST e HARRIS 2005]WEST, N. H. E.; HARRIS, D. *CMOS VLSI design: A circuits and Systems Perspective*. Third. San Francisco,CA: Person, 2005.

[WESTE 2005]WESTE, N. H. E. *CMOS VLSI design: a circuits and systems perspective*. [S.l.]: Boston: Pearson, 2005.

[ZHANG 2001]ZHANG, W. Vhdl tutorial: Learn by example. *UC Riversite*, John Wiley Sons, 2001. Disponível em: <esd.cs.ucr.edu/labs/tutorial/>.

ANEXO A - Códigos Cálculo MDC - Comportamental

```
1   ----------------------------------------------------
2   -- Behvaior Design of GCD calculator
3   -- by Weijun Zhang, 05/2001
4   ----------------------------------------------------
5   library ieee;
6   use ieee.std_logic_1164.all;
7   use ieee.std_logic_arith.all;
8   use work.all;
9   ----------------------------------------------------
10  entity gcd1 is
11  port(   Data_X:    in unsigned(3 downto 0);
12          Data_Y:    in unsigned(3 downto 0);
13          Data_out:  out unsigned(3 downto 0)
14  );
15  end gcd1;
16
17  architecture behv of gcd1 is
18  begin
19      process(Data_X, Data_Y)
20          variable tmp_X, tmp_Y: unsigned(3 downto 0);
21      begin
22          tmp_X := Data_X;
23          tmp_Y := Data_Y;
24
25          for i in 0 to 15 loop
26              if (tmp_X/=tmp_Y) then
27                  if (tmp_X < tmp_Y) then
28                      tmp_Y := tmp_Y - tmp_X;
29                  else
30                      tmp_X := tmp_X - tmp_Y;
31                  end if;
32              else
33                  Data_out <= tmp_X;
34              end if;
35          end loop;
36
37      end process;
38  end behv;
39  ----------------------------------------------------
```

Código 4: Exemplo de um módulo em linguagem VHDL comportamental

ANEXO B - Códigos Cálculo MDC - RTL

```
1   ---------------------------------------------------------------------
2   -- GCD CALCULATOR (ESD book figure 2.11)
3   -- Weijun Zhang, 04/2001
4   --
5   -- we can put all the components in one document(gcd2.vhd)
6   -- or put them in separate files
7   -- this is the example of RT level modeling (FSM + DataPath)
8   -- the code is synthesized by Synopsys design compiler
9   ---------------------------------------------------------------------
10
11  -- Component: MULTIPLEXOR -------------------------------------------
12
13  library IEEE;
14  use IEEE.std_logic_1164.all;
15  use IEEE.std_logic_arith.all;
16  use IEEE.std_logic_unsigned.all;
17
18  entity mux is
19      port(       rst, sLine: in std_logic;
20                  load, result: in std_logic_vector( 3 downto 0 );
21                  output: out std_logic_vector( 3 downto 0 )
22      );
23  end mux;
24
25  architecture mux_arc of mux is
26  begin
27      process( rst, sLine, load, result )
28      begin
29          if( rst = '1' ) then
30              output <= "0000";           -- do nothing
31          elsif sLine = '0' then
32              output <= load;             -- load inputs
33          else
34              output <= result;           -- load results
35          end if;
36      end process;
37  end mux_arc;
38
39  -- Component: COMPARATOR --------------------------------------------
40
41  library IEEE;
42  use IEEE.std_logic_1164.all;
43  use IEEE.std_logic_arith.all;
44  use IEEE.std_logic_unsigned.all;
45
46  entity comparator is
47      port(       rst: in std_logic;
48                  x, y: in std_logic_vector( 3 downto 0 );
49                  output: out std_logic_vector( 1 downto 0 )
50      );
51  end comparator;
52
53  architecture comparator_arc of comparator is
54  begin
55      process( x, y, rst )
56      begin
57          if( rst = '1' ) then
58              output <= "00";             -- do nothing
59          elsif( x > y ) then
60              output <= "10";             -- if x greater
61          elsif( x < y ) then
62              output <= "01";             -- if y greater
63          else
64              output <= "11";             -- if equivalance.
65          end if;
66      end process;
67  end comparator_arc;
68
69
70
71
```

Código 5: Exemplo de um módulo em linguagem VHDL - RTL

ANEXO B - Códigos Cálculo MDC - RTL (Cont.)

```
1   -- Component: SUBTRACTOR ------------------------------------------
2   library IEEE;
3   use IEEE.std_logic_1164.all;
4   use IEEE.std_logic_arith.all;
5   use IEEE.std_logic_unsigned.all;
6
7   entity subtractor is
8       port(       rst: in std_logic;
9                   cmd: in std_logic_vector( 1 downto 0 );
10                  x, y: in std_logic_vector( 3 downto 0 );
11                  xout, yout: out std_logic_vector( 3 downto 0 )
12      );
13  end subtractor;
14
15  architecture subtractor_arc of subtractor is
16  begin
17      process( rst, cmd, x, y )
18      begin
19          if( rst = '1' or cmd = "00" ) then      -- not active.
20              xout <= "0000";
21              yout <= "0000";
22          elsif( cmd = "10" ) then                -- x is greater
23              xout <= ( x - y );
24              yout <= y;
25          elsif( cmd = "01" ) then                -- y is greater
26              xout <= x;
27              yout <= ( y - x );
28          else
29              xout <= x;                          -- x and y are equal
30              yout <= y;
31          end if;
32      end process;
33  end subtractor_arc;
34  -- Component: REGISTER ----------------------------------------------------
35
36  library IEEE;
37  use IEEE.std_logic_1164.all;
38  use IEEE.std_logic_arith.all;
39  use IEEE.std_logic_unsigned.all;
40
41  entity regis is
42      port(       rst, clk, load: in std_logic;
43                  input: in std_logic_vector( 3 downto 0 );
44                  output: out std_logic_vector( 3 downto 0 )
45      );
46  end regis;
47
48  architecture regis_arc of regis is
49  begin
50      process( rst, clk, load, input )
51      begin
52          if( rst = '1' ) then
53              output <= "0000";
54          elsif( clk'event and clk = '1') then
55              if( load = '1' ) then
56                  output <= input;
57              end if;
58          end if;
59      end process;
60  end regis_arc;
61
62  -- component: FSM controller ----------------------------------------
63
64  library IEEE;
65  use IEEE.std_logic_1164.all;
66  use IEEE.std_logic_arith.all;
67  use IEEE.std_logic_unsigned.all;
68
69  entity fsm is
70      port(       rst, clk, proceed: in std_logic;
71                  comparison: in std_logic_vector( 1 downto 0 );
72                  enable, xsel, ysel, xld, yld: out std_logic
73      );
74  end fsm;
```

Código 6: Exemplo de um módulo em linguagem VHDL - RTL

ANEXO B - Códigos Cálculo MDC - RTL (Cont.)

```
1   architecture fsm_arc of fsm is
2
3       type states is ( init, s0, s1, s2, s3, s4, s5 );
4       signal nState, cState: states;
5
6   begin
7       process( rst, clk )
8       begin
9           if( rst = '1' ) then
10              cState <= init;
11          elsif( clk'event and clk = '1' ) then
12              cState <= nState;
13          end if;
14      end process;
15
16      process( proceed, comparison, cState )
17      begin
18          case cState is
19
20          when init =>    if( proceed = '0' ) then
21                              nState <= init;
22                          else
23                              nState <= s0;
24                          end if;
25
26          when s0 =>      enable <= '0';
27                          xsel <= '0';
28                          ysel <= '0';
29                          xld <= '0';
30                          yld <= '0';
31                          nState <= s1;
32
33          when s1 =>      enable <= '0';
34                          xsel <= '0';
35                          ysel <= '0';
36                          xld <= '1';
37                          yld <= '1';
38                          nState <= s2;
39
40          when s2 =>      xld <= '0';
41                          yld <= '0';
42                          if( comparison = "10" ) then
43                              nState <= s3;
44                          elsif( comparison = "01" ) then
45                              nState <= s4;
46                          elsif( comparison = "11" ) then
47                              nState <= s5;
48                          end if;
49
50          when s3 =>      enable <= '0';
51                          xsel <= '1';
52                          ysel <= '0';
53                          xld <= '1';
54                          yld <= '0';
55                          nState <= s2;
56
57          when s4 =>      enable <= '0';
58                          xsel <= '0';
59                          ysel <= '1';
60                          xld <= '0';
61                          yld <= '1';
62                          nState <= s2;
63
64
```

Código 7: Exemplo de um módulo em linguagem VHDL - RTL

ANEXO B - Códigos Cálculo MDC - RTL (Cont.)

```
          when s5 =>      enable <= '1';
                          xsel <= '1';
                          ysel <= '1';
                          xld <= '1';
                          yld <= '1';
                          nState <= s0;

          when others =>  nState <= s0;

          end case;
      end process;
  end fsm_arc;

  ------------------------------------------------------------------------
  -- GCD Calculator: top level design using structural modeling
  -- FSM + Datapath (mux, registers, subtracter and comparator)
  ------------------------------------------------------------------------
  library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_arith.all;
  use IEEE.std_logic_unsigned.all;
  use work.all;

  entity gcd is
      port(       rst, clk, go_i: in std_logic;
                  x_i, y_i: in std_logic_vector( 3 downto 0 );
                  d_o: out std_logic_vector( 3 downto 0 )
      );
  end gcd;

  architecture gcd_arc of gcd is

  component fsm is
      port(       rst, clk, proceed: in std_logic;
                  comparison: in std_logic_vector( 1 downto 0 );
                  enable, xsel, ysel, xld, yld: out std_logic
      );
  end component;

  component mux is
      port(       rst, sLine: in std_logic;
                  load, result: in std_logic_vector( 3 downto 0 );
                  output: out std_logic_vector( 3 downto 0 )
      );
  end component;

  component comparator is
      port(       rst: in std_logic;
                  x, y: in std_logic_vector( 3 downto 0 );
                  output: out std_logic_vector( 1 downto 0 )
      );
  end component;

  component subtractor is
      port(       rst: in std_logic;
                  cmd: in std_logic_vector( 1 downto 0 );
                  x, y: in std_logic_vector( 3 downto 0 );
                  xout, yout: out std_logic_vector( 3 downto 0 )
      );
  end component;
```

Código 8: Exemplo de um módulo em linguagem VHDL - RTL

ANEXO B - Códigos Cálculo MDC - RTL (Cont.)

```
component regis is
    port(       rst, clk, load: in std_logic;
                input: in std_logic_vector( 3 downto 0 );
                output: out std_logic_vector( 3 downto 0 )
    );
end component;

signal xld, yld, xsel, ysel, enable: std_logic;
signal comparison: std_logic_vector( 1 downto 0 );
signal result: std_logic_vector( 3 downto 0 );

signal xsub, ysub, xmux, ymux, xreg, yreg: std_logic_vector( 3 downto 0 );

begin
    -- doing structure modeling here
    -- FSM controller
    TOFSM: fsm port map(    rst, clk, go_i, comparison,
                            enable, xsel, ysel, xld, yld );
    -- Datapath
    X_MUX: mux port map( rst, xsel, x_i, xsub, xmux );
    Y_MUX: mux port map( rst, ysel, y_i, ysub, ymux );
    X_REG: regis port map( rst, clk, xld, xmux, xreg );
    Y_REG: regis port map( rst, clk, yld, ymux, yreg );
    U_COMP: comparator port map( rst, xreg, yreg, comparison );
    X_SUB: subtractor port map( rst, comparison, xreg, yreg, xsub, ysub );
    OUT_REG: regis port map( rst, clk, enable, xsub, result );

    d_o <= result;

end gcd_arc;

--------------------------------------------------------------------------------
```

Código 9: Exemplo de um módulo em linguagem VHDL - RTL

ANEXO C - Recomendações para síntese usando Verilog

Elaborou-se uma listagem com os principais pontos observados, para boas práticas para a codificação visando otimizar a síntese usando Verilog.

- Código identado e legível facilita o entendimento;
- Utilizar codificações genéricas, não dependentes de tecnologia;.
- Utilizar nomes dedutíveis nos sinais;
- Utilizar comentários para auxiliar no entendimento;
- Apenas um *"statement"* por linha;
- Dividir a lógica em módulos diferentes para tornar os caminhos críticos menores;
- Maximizar a utilização de *clock gate* para diminuir consumo, deve-se tomar cuidado com a metaestabilidade;
- Observar a não implementação de *loops* com lógica combinacional;
- Nunca associar o valor X a nenhum sinal;
- Dar preferência ao uso de *case statments*, ao invés de *if and else* statements;
- Simplificações lógicas podem reduzir caminhos críticos do design;
- O compartilhamento de recursos instanciados no código é uma boa técnica para reduzir área;
- A duplicação de registradores é uma boa técnica para se diminuir o *"fan-out"* do circuito e atender os requisitos de timing;
- A utilização da técnica de *bus inversion* pode gerar uma diminuição significativa na dissipação de potência dinâmica, mas o custo com implementação do mecanismo de controle, geralmente supera os ganhos de potência dinâmica com uma maior utilização de potência estática e área;
- De maneira geral todas as simplificações lógicas,principalmente dentro de lógica seqüencial, onde as ferramentas de síntese, não estão tão otimizadas ainda, resultam em ganhos significativos;
- A técnica de separação de *nets* com alta taxa de comutação em lógicas especiais, resulta em grandes reduções de potência dinâmica dissipada como visto na pratica durante o desenvolvimento do trabalho
- A extração de *mux* desnecessários, principalmente em statments *if, elsif* e *else* encadeados sempre que possível deve ser empregada para diminuir área;
- A maneira da representação de números binários, pode ser empregada em *designs ulta-lowpower* em projetos normais o resultado praticamente não é alterado;
- A maneira de realizar o reset, síncrono ou assíncrono pode influenciar muito no *timing* do *design*.

ANEXO D - Publicação ERAD 2007

Implementação de um *IP Core* MAC para Uso em Sistemas Embarcados

Lúcio Renê Prade, Vitor Ângelo P. Righi, Rafael Ramos dos Santos

Universidade de Santa Cruz do Sul - UNISC
Avenida Independência, 2293 - Santa Cruz do Sul / RS - CEP: 96815-900.
lucio@mx2.unisc.br, vitorrighi@mx2.unisc.br, rsantos@unisc.br

Introdução

Sistemas embarcados são tipicamente projetados para a execução de tarefas específicas tendo restrições de tamanho, consumo e desempenho como principais requisitos de projeto [BER 02]. Para interagir com outros sistemas, fazem uso de protocolos de comunicação.

Este trabalho tem por objetivo a implementação de um *IP Core (Intellectual Property Core)* MAC baseado em um *core* de domínio público. A implementação compreende a verificação, validação, otimização e síntese do módulo MAC para plataformas baseadas em FPGAs, inicialmente.

Implementação Proposta

A função do MAC (*Media Access Control*) é permitir que dispositivos compartilhem a capacidade de transmissão de uma rede. Ele controla o acesso ao meio de transmissão de modo a se ter um uso ordenado e eficiente deste meio [TAN 03].

O trabalho está sendo desenvolvido tendo com base um *IP Core* de domínio público disponível no repositório Opencores [OPE 06]. Este *Core* deve desempenhar todas as funções do protocolo MAC previstas no padrão IEEE 802.3. Assim, pode ser otimizado e utilizado no desenvolvimento de sistemas embarcados.

Resultados

Para os estudos apresentados neste trabalho, foram utilizados dois dispositivos FPGAs da Xilinx [XIL 06]. O Spartan 3 permite a síntese de circuitos com até 1920 *Slices* alcançando até 50 MHz de freqüência. O Virtex-II PRO por sua vez, além de

permitir a síntese de circuitos com até 13696 *Slices,* pode alcançar até 100MHz de freqüência.

As comparações mostraram que é possível sintetizar o MAC para ambos os FPGAs. A síntese demonstrou entretanto que para o Spartan 3, foram consumidos 1230 *Slices* (64%), o que deixaria pouco espaço disponível para a síntese de outros circuitos.

A síntese para o Virtex-II PRO consumiu apenas 8% dos *Slices* disponíveis demonstrando que com este FPGA seria possível integrar circuitos mais complexos.

	Spartan 3	**Virtex-II PRO**
Slice (Área Ocupada)	1230 (64%)	1228 (8%)
Freqüência Máxima (MHz)	99.632	162.640
Consumo (mW)	37	495,63
IOBs (Interfaces de Entradas e Saídas)	151 (87%)	151 (27%)

Tabela 1 – Comparação dos dados de síntese para FPGAs Spartan 3 e Virtex-II PRO

Conclusões

Os trabalhos futuros estão focados na validação e otimização do MAC para que seja possível utilizar uma implementação eficiente de acordo com a aplicação, possibilitando a escolha do dispositivo que melhor se adequa a situação.

É importante enfatizar que o grande benefício do emprego de FPGAs para o projeto de sistemas embarcados é a flexibilidade permitida através da reconfiguração. A reconfiguração permite entre outras coisas que sejam efetuadas otimizações no sistema tanto ao nível de hardware quanto no nível de software baseado essencialmente na aplicação que se está projetando. Além disso, é possível também que a configuração do FPGA, dos IPs empregados e as otimizações do software sejam reconfiguradas estaticamente ou dinamicamente como mostrado no trabalho Reconfigurabilidade Dinâmica e Remota de FPGAs [RIB 02].

Referências

[BER 02] BERGER, A. **Embedded systems design**: an introduction to processes, tools, and techniques. Lawrence: CMP Books, 2002. 237p.

[TAN 03] TANENBAUM, A.S. **Redes de computadores**. Rio de Janeiro: Elsevier, 2003. 945p.

[OPE 06] OPENCORES. **Repositório de *IP Cores*.** Disponível em: www.opencores.org. 2006.

[XIL 06] XILINX INC. **Site do Fabricante Xilinx.** Disponível em: www.xilinx.com.2006.

[RIB 02] RIBEIRO, A. A. , MARQUES, E. **Reconfigurabilidade Dinâmica e Remota de FPGAs**. USP 2002.

ANEXO E - Publicação ERAD 2008

Aplicação de Técnicas de Aumento de Desempenho no Projeto de um Circuito Integrado Digital para Comunicação de Dados

Lúcio Renê Prade[1], Vitor Righi[1], Robert Torrel[1], Rafael Ramos dos Santos[1]

[1]Programa de Pós-Graduação em Sistemas e Processos Industriais
Universidade de Santa Cruz do Sul
Grupo de Projeto de Sistemas Embarcados e Microeletrônica (GPSEM) – Sala 5341
Caixa Postal 188 – 96815-900 – Santa Cruz do Sul – RS – Brazil

{lucio,vitorrighi,roberttorrel}@mx2.unisc.br, rsantos@unisc.br

1. Introdução

Muitas são as pesquisas desenvolvidas para a otimização de sistemas, diminuição do consumo de potência e aumento do desempenho. No desenvolvimento de sistemas embarcados tais restrições são muito importantes e em alguns casos a substituição de processadores de propósito geral e do softwares por circuitos de uso específicos -ASIC (*Application Specific Integrated Circuit*) ajuda no desempenho destes sistemas. Adicionando-se a estes técnicas de *low-power* e *timing*, pode-se aumentar ainda mais o desempenho do sistema. Neste trabalho pretende-se mostrar o caso da implementação de um circuito integrado para comunicação de dados em redes *Ethernet* [Tanenbaum 2003] que poderá substituir o MAC (*Media Access Control*) desenvolvido em software. Será realizada a comparação de duas implementações do MAC onde primeiramente somente serão aplicadas as técnicas padrão e posteriormente serão aplicadas as técnicas de otimização.

2. O MAC *Ethernet*

O MAC (*Media Access Control*) estudado utiliza o padrão 802.3 [IEEE 2005], conhecido como *Ethernet*, que usa o conceito de colisão chamado CSMA/CD (*Carrier-Sense Multiple Access with Collision Detection*) para comunicação. O *IP Core* utilizado é do repositório Opencores.Org [OpenCores.Org 2007]. A escolha do mesmo foi devido a sua boa documentação e o código descrito utilizando-se linguagem Verilog (linguagem de descrição de hardware, usada para facilitar o *design* de circuitos digitais em FPGAs e ASICs). Na Figura 1 pode-se observar a estrutura do MAC *Ethernet*. Ele é composto por diversos submódulos em verilog que possuem funcionalidades específicas como as filas de recebimento e transmissão de pacotes eth_rxethmac e eth_txethmac, os registradores de configuração e dados eth_registers, a interface de configuração do meio físico eth_miim, a máquina de estados eth_maccontrol , os registradores de *status* eth_macstatus e a interface com o barramento wishbone que é largamente utilizado em sistemas embarcados.

3. Técnicas de Otimização do Projeto

As técnicas de otimização para projetos de hardware são classificadas em três grupos: técnicas de redução de *timing*, redução de área e redução de potência dissipada. Neste trabalho, utilizou-se apenas dois grupos julgados como os mais importantes para o ganho de desempenho (relação consumo/processamento) que são as técnicas de redução de potência dissipada dentre as quais podemos citar: *clock gating*, inversão de barramento,

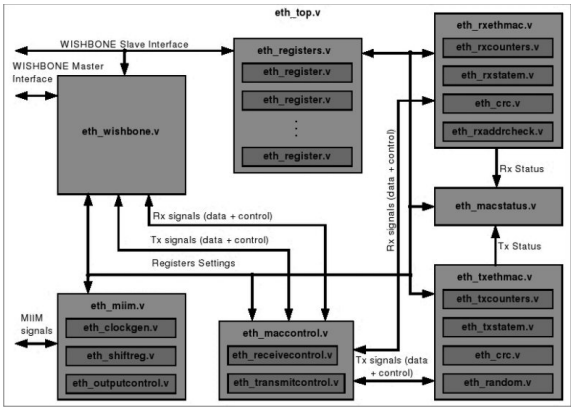

Figura 1. Diagrama de blocos da estrutura do MAC *Ethernet* utilizado [OpenCores.Org 2007].

representação de números binários e as técnicas de redução de *timing*: simplificação e replicação lógica [Cummings 2002].

As técnicas de diminuição da potência dissipada empregadas têm como objetivo reduzir a dissipação de potência dinâmica através da redução do número de chaveamento dos transistores. Já as técnicas de redução de *timing* buscam diminuir o nível de encadeamento de lógica fazendo com que a propagação do sinal estável seja mais rápida [Cummings 2001].

4. Resultados

Aplicando-se as duas técnicas de otimização de *timing* aos submódulos do MAC *Ethernet*, utilizando-se a tecnologia de fabricação tsmc 0.35 e dando especial atenção aos caminhos críticos do circuito, foi possível um incremento da freqüência inicial de trabalho de 115 MHz para 132 MHz nas simulações realizadas. A aplicação das técnicas de redução de potência dissipada estão em fase de implementação e análise dos resultados. Após testadas separadamente, pretende-se implementar uma versão do circuito empregando as duas técnicas de otimização e verificar o impacto dessa implementação em relação ao circuito original sem nenhuma otimização.

Referências

Cummings, C. E. (2001). Verilog coding styles for improved simulation efficiency. Sunburst Design, Inc.

Cummings, C. E. (2002). Verilog-2001 behavioral and synthesis enhancements. Sunburst Design, Inc.

IEEE (2005). Institute of electrical and electronics engineers, inc.,ieee 802.3.

OpenCores.Org (2007). Repositório de ip cores. http://www.opencores.org.

Tanenbaum, A. S. (2003). Redes de computadores. Elsevier Rio de Janeiro.

ANEXO F - Publicação SForum 2007

VERIFICATION COVERAGE AND SYNTHESIS OF AN ETHERNET ASIC

Lúcio R. Prade, Marco A. B. Hennes, Vitor A. P. Righi, Robert Torrel, Rafael R. dos Santos

{lucio, hennes, vitorrighi, roberttorrel}@mx2.unisc.br
rsantos@unisc.br

Universidade de Santa Cruz do Sul – UNISC
Av. Independência 2293, Bloco 53
Santa Cruz do Sul, RS, 96815-900, Brazil

ABSTRACT

This paper presents the verification coverage and logic synthesis results of Ethernet MAC Controller logic. A testplan for the functional verification was developed and the resulting coverage in terms of statements, branches and conditions was evaluated. Furthermore, the design was synthesized to a Virtex II Pro FPGA (for validation purposes) and later to an ASIC using the TSMC 0.35 technology.

1. INTRODUCTION

One of the most popular data communication techniques is the Ethernet. It was originally based on the idea of computers communicating over a broadcast transmission media. The coaxial cable was later replaced by a twisted-pair of wires connecting computers through hubs or switches.

Nowadays, several embedded systems are using such technology to perform data exchange among different systems. The specification of the MAC Ethernet (Media Access Control Ethernet) is described by the IEEE 802.3 standard and it defines the scheme known as the carrier sense multiple access with collision detection (CSMA/CD) which determines the way computers access a shared channel [4].

In this paper we revisit the normal ASIC flow and focus on the functional verification coverage and logic synthesis of a subset of the MAC Ethernet logic.

2. DESIGN FLOW

There are four basic stages in the design flow of an ASIC (Application-specific Integrated Circuit): design, test, fabrication and packaging [2] (Figure 1). The design stage is usually divided into three major tasks: conceptualization and modeling, synthesis and optimization and finally validation. On the conceptualization and modeling an idea is transformed into a model, which captures the functionality that the resulting circuit would perform. The synthesis consists of refining the model, from an abstract level into a more detailed one that defines the necessary features and characteristics for fabrication. The validation is the task where the model and its lower level representations are verified against the original specification.

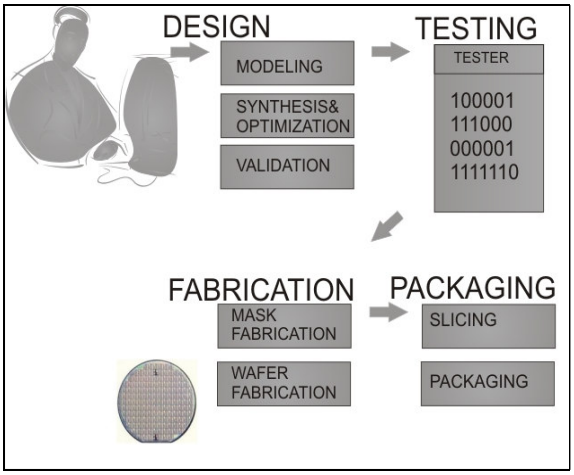

Figure 1.The four stages of the design flow of an integrated circuit

An actual ASIC design is usually taken from the highest level of abstraction through the lowest level. The first representation of an idea is normally coded using an HLL (High Level Programming Language) or an HDL (Hardware description Language) [1]. This first description details the main functionality without the associated details. In other words, the high level model describes the specific functionality which is intended to be built into the circuit.

The circuit synthesis adds to the initial model by appending more detailed information about the actual circuit needed to realize the functionality being modeled. The model is a representation of the functionality without details. As the synthesis proceeds generating lower levels of abstractions, more details are added to the model until the actual geometric representation of the circuit is obtained.

3. THE ETHERNET MAC

This study was developed based on a public IP-Core available at the Opencores repository [6]. The MAC core was originally written in Verilog and it represents the full functionality of a MAC Ethernet according to the IEEE 802.3 standard [4].

Figure 2 shows the block diagram of the MAC core where the top-level (eth_top.v) and its submodules are presented. The complete MAC model is represented by seven submodules named: eth_mii, eth_registers, eth_maccontrol, eth_txethmac, eth_rxethmac, eth_wishbone, eth_macstatus and some logic used for synchronization, multiplexing and registering outputs.

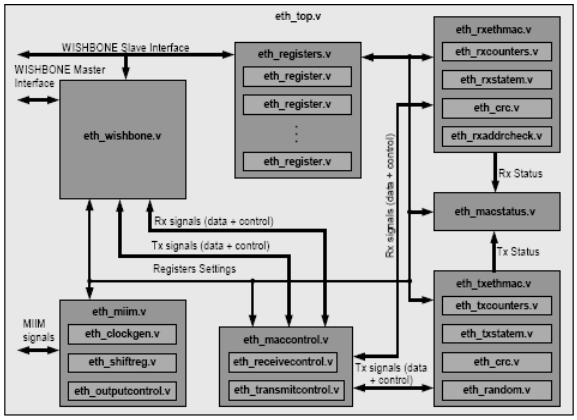

Figure 2. Block diagram of the MAC IP-core [6].

The goal of this work is to carry a design throughout all the stages. In order to achieve that without spending too much time designing such a complex device, the MII (Media Independent Interface) submodule was chose to get started with. The main reason behind this choice is that by designing the MII submodule it will be possible to fabricate a device and test it without having to design the complete MAC module.

The MII submodule is an interface to the external Ethernet PHY chip. It is used for setting the PHY´s configuration registers and to access its status. On top of that, a whole set of other functions necessary to work with the external PHY are also implemented in the MII such as a clock divider and logic to synchronize signals between the PHY and the MAC.

The MII interface consists mainly of two signals: clock (MDC) and the bi-directional data signal (MDIO). The MDIO signal needs to combine the input signal Mdi, the output signal Mdo and the enable signal MdoEn.

4. FUNCTIONAL VERIFICATION

A testplan was developed in order to verify the functionality [3] of the MII submodule where a set of 17 testcases cover the main aspects of this submodule. The software Questa [7] by Mentor Graphics was used to measure the verification coverage of such testcases. Table 1 presents the coverage results of this analysis.

Verification Coverage - MII									
	Stmts			Branches			Conditions		
	Active	Hits	%Cov	Active	Hits	%Cov	Active	Hits	%Cov
eth_miim	83	83	100	46	45	97.8	18	18	100
eth_clockgen	10	10	100	10	10	100	0	0	100
eth_outputcontrol	15	15	100	8	8	100	0	0	100
eth_shiftreg	12	12	100	16	16	100	0	0	100

Table 1. MII verification coverage results

The results show that 100% of the statements, 97.8% of the Branches were hit and 100% of the conditions were also hit. This represents a very high level of coverage for the MII submodule where only one Branch has not been hit by the stimulus generated through the testcases used for verification.

The testplan will be augmented to add the testcases suggested by the IOL InterOperabilty Laboratory [5].

5. LOGICAL SYNTHESIS

The Leonardo Spectrum [7] was used to synthesize the submodule for both ASIC and FPGA. Tables 2 and 3 show the cells count and synthesis results, respectively for the ASIC synthesis using the TSMC 0.35 micron technology [8] and for the Virtex II Pro FPGA.

Cell	References	Total Area
and02	1 x 1	1 gates
and03	2 x 2	3 gates
aoi21	11 x 1	14 gates
aoi22	**3 x 1**	4 gates
aoi221	2 x 2	4 gates
aoi222	6 x 2	13 gates
buf02	4 x 1	4 gates
dffr	75 x 6	419 gates
dffs_ni	1 x 6	6 gates
inv01	16 x 1	12 gates
inv02	19 x 1	14 gates
mux21	6 x 2	11 gates
mux21_ni	43 x 2	78 gates
nand02	15 x 1	15 gates
nand03	5 x 1	6 gates
nand04	8 x 1	12 gates
nor02_2x	2 x 1	2 gates
nor02ii	10 x 1	12 gates
nor03_2x	5 x 1	6 gates
nor04	10 x 1	15 gates
oai21	16 x 1	20 gates
oai22	1 x 1	1 gates

Cell	References	Total Area
oai222	2 x 2	4 gates
oai32	1 x 2	2 gates
oai422	1 x 3	3 gates
or02	5 x 1	6 gates
or03	5 x 2	8 gates
or04	1 x 2	2 gates
xnor2	5 x 2	10 gates
xor2	5 x 2	11 gates

Table 2. Cells count - TSMC 0.35.

Number of ports :	66
Number of nets :	350
Number of instances :	286
Number of gates :	718

Table 3. Synthesis results - TSMC 0.35.

The Leonardo Spectrum [7] was also used to synthesize the same circuit for the FPGA Virtex II Pro [9].

This FPGA was used to prototype the module and will also be used for testing the device when it gets back from the fab. Tables 4 and 5 present the synthesis results for the targeted FPGA.

Cell	*Library*	*References*	*Total Area*
BUFGP	xcv2p	1 x 1	1 BUFGP
FDC	xcv2p	75 x 1	75 Dffs or Latches
FDP	xcv2p	1 x 1	1 Dffs or Latches
IBUF	xcv2p	39 x 1	39 IBUF
LUT1	xcv2p	3 x 1	3 Function Generators
LUT2	xcv2p	18 x 1	18 Function Generators
LUT3	xcv2p	34 x 1	34 Function Generators
LUT4	xcv2p	96 x 1	96 Function Generators
MUXF5	xcv2p	5 x 1	5 MUXF5
OBUF	xcv2p	25 x 1	25 OBUF

Table 4. Cells count - Virtex II Pro.

Number of ports :	66
Number of nets :	337
Number of instances :	297
Number of references to this view :	0
Number of BUFGP :	1
Number of Dffs or Latches :	76
Number of ports :	66
Number of Function Generators :	151
Number of IBUF :	39
Number of MUXF5 :	5
Number of OBUF :	25
Number of gates :	151
Number of accumulated instances :	297

Table 5. Synthesis results - Virtex II Pro.

6. FUTURE WORK

In order to test the device designed, a test platform will be built using the Virtex II Pro board. The MII ASIC will be placed in between the existing PHY and the FPGA. The MII submodule will be subtracted from the original RTL MAC module.

An external board will be designed to place the MII ASIC and make the necessary connections between the FPGA, the MII ASIC and the PHY. The FPGA will hold the MAC (less the MII) and it will drive signals to the MII ASIC which will then drive signals to the PHY, and vice-versa. When transmitting data, the FPGA will send signals to the MII ASIC which will pass down to the PHY. When receiving data, the MII ASIC will get signals from the PHY, process them and send up to the FPGA.

With this strategy, it will be possible to put the MII ASIC into an actual environment where real Ethernet frames will be sent and received through a real data network. Figure 3 shows the block diagram of the test environment.

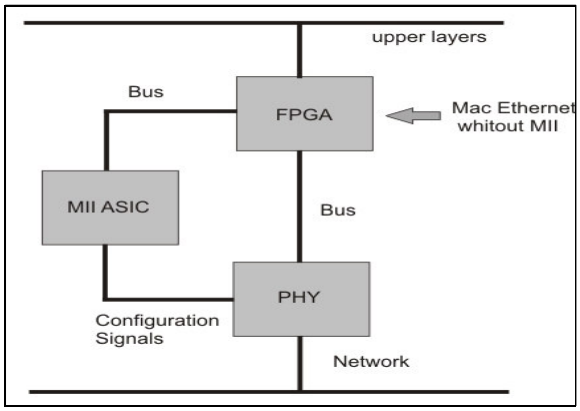

Figure 3. - MII ASIC test environment.

7. CONCLUSIONS

In this paper it was presented a brief description of the design flow with emphasis on the functional verification coverage and the synthesis of a MII module for both FPGA and ASIC. The results showed that a high degree of coverage was achieved and the circuit was successfully synthesized for both FPGA and ASIC.

The next step is to augment the testplan in order to add testcases specified by the IOL and carry the design

through the electrical characterization and layout to then send it to the foundry. At last, the test environment under development will be used to carry out the testing of the resulting circuit.

8. ACKNOWLEDGMENTS

The authors gratefully thank the UNISC support in the form of scholarships and the CEITEC technical support.

9. REFERENCES

[1] Ciletti, M. D., Advanced Digital Design with the Verilog HDL, Pearson Education, New Jersey, 2003.

[2] Micheli, G. D., Synthesis and Optimization of Digital Circuits, McGraw-Hill, USA, 1994.

[3] Wile, B., Goss, J. C. , Roesner, W., Comprehensive Functional Verification - The Complete Industry Cycle, Elsiever, San Francisco, 2005.

[4] IEEE 802.3 (Institute of Electrical and Electronics Engineers, Inc.), http://www.ieee.org.

[5] IOL InterOperability Laboratory, http://www.iol.unh.edu/services/testing/fe/testsuites/X.

[6] Opencores, http://www.opencores.org.

[7] Mentror Graphics, www.mentor.com.

[8] Taiwan Semiconductors Manufacturing CompanyLimited ,http://www.tsmc.com/english/b_technology/b_technology_index.htm.

[9] Xilinx Virtex II Pro , www.xilinx.com.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo