UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

EXPLORAÇÃO DE OPERADORES ARITMÉTICOS NA TRANSFORMADA RÁPIDA DE FOURIER

DISSERTAÇÃO DE MESTRADO

Mateus Beck Fonseca

Santa Maria, RS, Brasil
2010

EXPLORAÇÃO DE OPERADORES ARITMÉTICOS NA TRANSFORMADA RÁPIDA DE FOURIER

por

Mateus Beck Fonseca

Dissertação apresentada ao Programa de Pós-Graduação em Informática, Área de concentração em Microeletrônica, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Mestre em Computação**.

Orientador: Prof. Dr. João Baptista dos Santos Martins
Co-orientador: Prof. Dr. Eduardo Antonio César da Costa

Santa Maria, RS, Brasil

2010

Universidade Federal de Santa Maria
Centro de Tecnologia
Programa de Pós-Graduação em Informática

A Comissão Examinadora, abaixo assinada,
aprova a Dissertação de Mestrado

EXPLORAÇÃO DE OPERADORES ARITMÉTICOS NA TRANSFORMADA RÁPIDA DE FOURIER

elaborada por
Mateus Beck Fonseca

como requisito parcial para obtenção do grau de
Mestre em Computação

COMISSÃO EXAMINADORA:

João Baptista dos Santos Martins, Dr.
(Presidente/Orientador)

Eduardo Antonio César da Costa, Dr.
(Co-orientador)

Renato Perez Ribas, Dr. (UFRGS)

Sérgio José Melo de Almeida, Dr. (UCPel)

Leonardo Londero de Oliveira, Dr. (UFSM)

Santa Maria, 22 de Outubro de 2010

Dedico este trabalho à minha namorada e a meus pais.

AGRADECIMENTOS

Agradeço aos meus pais, Paulo e Sílvia por sempre encorajarem meus estudos, ao Sr. Ivo Beck e família pelos debates filosóficos e principalmente à minha namorada Lislaine Cansi pelo incentivo durante a elaboração deste trabalho.

Um agradecimento especial ao meu orientador, João Baptista dos Santos Martins, pela orientação deste trabalho e pelas oportunidades oferecidas. Aos professores Giovani Baratto e André Luiz Aita pelas conversas informais elucidatórias. Agradeço também aos demais professores do Gmicro que indiretamente colaboraram para o desenvolvimento deste trabalho.

Um agradecimento especial também ao meu co-orientador, Eduardo Antonio César da Costa, pelo suporte teórico/ técnico e trabalho conjunto nas elaborações de artigos. Em contíguo ao pessoal do Laboratório de Microeletrônica e processamento de sinais da UCPel salientando-se: João Alterman, Leandro Zafalon Pieper e Alexsandro Schiavon, pelas aulas de operadores aritméticos.

Cabe ressaltar minha gratidão a Leonardo Londero de Oliveira e Fernando Luis Herrmann por dividirem sempre seus conhecimentos com paciência e dedicação e Taimur Gibran Rabuske Kuntz pelo trabalho conjunto na criação de uma nova topologia de porta lógica XOR.

A todos os colegas do Gmicro agradeço pela amizade e apoio. Com ênfase a todos que conviveram comigo durante esta jornada dividindo a sala 278, e que não foram ainda citados, em especial: Tiago Oliveira Weber, Lucas Teixeira, Douglas Camargo Foster, Crístian Müller, Tiago Guedes da Luz Martins e Paulo César Comassetto de Aguirre.

Agradeço também os professores Giovani Baratto e Alice de Jesus Kozakevicius por seus comentários extremamente valiosos durante a banca de avaliação prévia desta dissertação que deram o rumo final para este trabalho.

Também merecem ser citados e sou muito grato: Marinelma Aimi de Carvalho, secretária do PPGI da UFSM, por sua colaboração durante este período, CNPQ pelo suporte oferecido, Cadence e o programa CI-BRASIL do Ministério da Ciência e Tecnologia - MCT.

"*Omnia mea mecum porto*"

Bias

RESUMO

Dissertação de Mestrado
Programa de Pós-Graduação em Informática
Universidade Federal de Santa Maria, RS, Brasil

EXPLORAÇÃO DE OPERADORES ARITMÉTICOS NA TRANSFORMADA RÁPIDA DE FOURIER

AUTOR: MATEUS BECK FONSECA
ORIENTADOR: JOÃO BAPTISTA DOS SANTOS MARTINS
CO-ORIENTADOR: EDUARDO ANTONIO CÉSAR DA COSTA

Local da Defesa e Data: Santa Maria, 22 de Outubro de 2010.

A redução no consumo de potência na transformada rápida de Fourier (FFT) é importante pois sua aplicação cresce em sistemas embarcados movidos à bateria. Sendo assim este trabalho tem como foco a aplicação de técnicas de redução de potência para projetos específicos de algoritmos da FFT. O objetivo é realizar uma exploração arquitetural no elemento central de cálculo da FFT, borboleta na base 2 com decimação no tempo, bem como a aplicação de operadores aritméticos eficientes na estrutura interna desta borboleta. As técnicas aplicadas à borboleta têm por objetivo a redução do consumo de potência através de exploração arquitetural e codificação de dados. São apresentadas cinco diferentes topologias de borboleta, sendo uma destas, proposta no âmbito deste trabalho utilizando três multiplicadores reais é baseada no armazenamento prévio do produto dos valores real e imaginário dos coeficientes. A vantagem desta topologia é a possibilidade do uso de somadores compressores 4:2, que realiza a soma simultânea de quatro operandos, com reduzido caminho crítico. Como estes somadores compressores apresentam portas XOR no caminho crítico, é proposta neste trabalho uma nova porta XOR, que é baseada no uso de transistores de passagem. Esta nova porta lógica XOR foi aplicada em somadores compressores 3:2 e 4:2, que são aplicados nos blocos somadores das borboletas. Os circuitos digitais foram desenvolvidos em linguagem de descrição de hardware e alguns em esquemáticos de nível elétrico. Resultados de área, potência e contagem de células na síntese lógica em 180nm a 100MHz e 20MHz com análise de atividade de chaveamento para 10.000 vetores aleatórios de entrada foram obtidos e simulações no nível elétrico em um ambiente de sinais digitais e analógicos misto também foram realizadas para a avaliação dos compressores com a nova topologia de porta XOR. As análises mostram que os somadores compressores 3:2 apresentam menor consumo de potência com o uso da nova porta XOR. Entretanto, o mesmo não se observa em relação ao compressor 4:2 que apresenta um menor consumo de potência utilizando a porta XOR CMOS. Como as estruturas de borboleta avaliadas utilizam uma quantidade significativa de operadores aritméticos nas suas estruturas internas, foram utilizadas diferentes estratégias de projeto para as suas implementações. Inicialmente foram utilizados os operadores aritméticos da ferramenta de síntese automática (Cadence). Após, foram utilizados operadores aritméticos dedicados (somadores compressores com a nova porta XOR, somadores RNS e multiplicadores *array*). Os resultados mostram que as borboletas apresentam menores consumos de potência com o uso dos somadores compressores em suas estruturas.

Palavras-chave: FFT; Borboleta base 2 DET; Operadores aritméticos digitais; Mapeamento Lógico; baixa potência.

ABSTRACT

Master's Dissertation
Programa de Pós-Graduação em Informática
Federal University of Santa Maria, RS, Brazil

ARITHMETICS OPERATORS EXPLORATION IN FAST FOURIER TRANSFORM

AUTHOR: MATEUS BECK FONSECA
ADVISOR: JOÃO BAPTISTA DOS SANTOS MARTINS
CO-ADVISOR: EDUARDO ANTONIO CÉSAR DA COSTA

Place and Date: Santa Maria, October 22nd, 2010.

The power consumption reduction in the fast Fourier transform (FFT) is important because applications in battery-powered embedded systems grows daily. Thus this work focuses on the application of techniques to reduce power in specific projects of FFT algorithms. The goal is to achieve an architectural exploration in the FFT core, the decimation in time butterfly radix-2 and the efficient implementation of arithmetic operators in the internal structure of this butterfly. The techniques applied to the butterfly are aimed at reducing power consumption through architectural exploration and data encryption. Five different butterfly topologies are shown, one of those, proposed in this work uses three real multipliers, and is based on the previous storage of the product of real and imaginary values of the twiddle factors. The advantage of this topology is the possibility of using 4:2 adder compressors, which performs the sum of four operands simultaneously with reduced critical path. These adder compressors have XOR gates in the critical path, is proposed in this paper a new XOR gate circuit, which is based on the use of pass transistors logic. This new XOR gate circuit has been applied to adder compressors 3:2 and 4:2, which are applied to adders blocks of the butterflies. Digital circuits have been developed in hardware description language and some in the electrical schematic level. Results of area, power consumption and cell count in the logic synthesis in 180nm at 100MHz and 20MHz with switching activity analysis for 10,000 random input vectors were obtained for this work. The electrical level simulations in an environment of mixed digital and analog signals were also performed to the evaluation of the compressors with new topology of XOR gate. Analyses show that 3:2 adder compressor has lower power consumption using the new XOR gate circuit. However, the same conclusion was not achieve in relation to the 4:2 adder compressor which has a lower power consumption using the CMOS XOR gate. Butterfly structures evaluated uses a significant amount of arithmetic operators in their internal structures, so was used different design strategies for implementation. Initially was used the arithmetic operators of automatic synthesis tool (Cadence). After, used dedicated arithmetic operators (adder compressors with the new XOR gate circuit, RNS adders and *array* multipliers). The results show that butterflies have lower power consumption with the use of adder compressors in their internal structures.

Keywords: FFT; Butterfly Radix-2; Digital arithmetic operators; Logic Sinthesys; Low Power.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

BWA	*Broadband Wireless Access* / Acesso sem fio em banda larga
CSA	*Carry Save Adder* / Somador *Carry Save*, sem tradução
CMOS	*Complementary Metal-Oxide Semiconductor* / Semicondutor Metal-Óxido Complementar
DET	Decimação Em Tempo
DEF	Decimação Em Frequência
DFT	*Discrete Fourier Transform*/ Transformada de Fourier Discreta
FA	*Full-Adder* / Somador Completo
FFT	*Fast Fourier Transform* / Transformada Rápida de Fourier
HA	*Half-Adder* / Meio-somador
HDL	*Hardware Description Language* / Linguagem de descrição de circuitos
IEEE	*Institute of Electrical and Electronics Engineers* / Instituto de Engenheiros Eletricistas e Eletrônicos.
IDFT	*Inverse Discrete Fourier Transform* / Transformada de Fourier Discreta Inversa
MOSFET	*Metal Oxide Semiconductor Field Effect Transistor*
MUX	Porta lógica Multiplexador
OFDM	*Orthogonal Frequency Division Multiplexing* / Multiplexação por divisão ortogonal de frequência (canais separados em frequência)
OSI	*Open Systems Interconnection* / Interconexão de Sistemas Abertos
P&D	Pesquisa e Desenvolvimento
PHY	*Physical layer* / Camada física da rede OSI
Pipelining	Uso de registradores para dividir circuitos lógicos por ciclos de relógio
RCA	*Ripple Carry Adder* / Somador com propagação de 'vai-um'(*carry*)
RF	Rádio Frequência
RNS	*Residue Number Systems* /Sistema de número residual

ROM	*Read Only Memory* / Memória de somente leitura
RTL	*Register Tranfer Level*/ Nível de transferência de Registradores
TF	Transformada de Fourier
ULA	Unidade Lógica e Aritmética
UMTS	*Universal Mobile Telecommunications System* / Sistema de comunicação móvel universal
Wi-Fi	Nome comercial da rede WLAN baseada no protocolo IEEE 802.11
WiMAX	*Worldwide Interoperability for Microwave Access* / Interoperabilidade Mundial para Acesso de Micro ondas, nome comercial de uma rede WMAN baseada no protocolo 802.16 da IEEE.
WLAN	*Wireless Local Area Network* / Rede sem fio de área local
WMAN	*Wireless Metropolitan Area Network* / Rede sem fio de área metropolitana
XOR	*eXclusive-OR* / Porta lógica ou exclusivo

LISTA DE SÍMBOLOS

$\alpha, \beta, \gamma, \delta$	Sinais aleatórios em circuitos
A	Sinal de entrada complexo da borboleta
a_t	Probabilidade A_t normalizada
A_t	Atividade do nó de saída
A_i	Parte imaginária do sinal de entrada complexo da Borboleta
A_r	Parte real do sinal de entrada complexo da Borboleta
b	Elemento da Base B_{RNS}
B	Sinal de entrada complexo da borboleta
B_{RNS}	Base de representação do sistema RNS
C	Sinal de saída complexo da Borboleta
C_{ap}	Capacitância de um circuito CMOS qualquer
C_i	Parte imaginária do sinal de saída complexo da Borboleta
C_r	Parte real do sinal de saída complexo da Borboleta
D	Sinal de saída complexo da Borboleta
D_i	Parte imaginária do sinal de saída complexo da Borboleta
D_r	Parte real do sinal de saída complexo da Borboleta
e	Número de Euler, constante matemática irracional de valor aproximado $2,7182\ldots$
E	Razão entre diferentes tecnologias
f	Frequência de operação de um circuito digital qualquer
$F(S)$	TF de $f(x)$
$f(x)$	Sinal qualquer em função de x, dado pela Figura 1
i	Operador complexo $\sqrt{-1}$
I_{CC}	Corrente de curto-circuito
j	Índice utilizado nas equações RNS

k	Índice de um ponto amostrado em $F(S), \forall\ 0 \leqslant k \leqslant P-1$
m	Número que representa a divisão nos operandos dos multiplicadores Array em grupos de $m\,bits$.
n	Índice de um ponto amostrado em $f(x), \forall\ 0 \leqslant n \leqslant P-1$
N	Alusão a um número inteiro qualquer
N_b	Número de *bits* utilizados nos operandos multiplicados
P	Número de pontos da FFT
P_{CC}	Potência de curto-circuito
P_{CH}	Potência de chaveamento
π	Razão do perímetro de uma circunferência pelo seu diâmetro, constante matemática irracional de valor aproximado $3,1415\ldots$
S	Sinal de saída em alguns circuitos digitais
t	Representação em RNS de T
T	Número qualquer a ser representado em RNS
t_c	Intervalo de tempo de comutação das chaves CMOS
u	Representação em RNS de U
U	Número qualquer a ser representado em RNS
V	Tensão de alimentação de um circuito digital
$W, W(n)$	Conjunto de valores de W_P a ser armazenado em ROM
$W_i, W_i(n)$	Parte imaginária de W
W_P	Twiddle Factor / Fator de giro
$W_r, W_r(n)$	Parte real de W
$X(k)$	Sinal de amostra de $F(S)$
$x(n)$	Sinal de amostra de $f(x)$
$\|\|$	Número de camadas de *Pipeline* aplicadas nas estruturas

SUMÁRIO

1 INTRODUÇÃO

A Transformada Discreta de Fourier (DFT - *Discrete Fourier Transform*) é uma das aplicações matemátivas mais utilizadas no processamento digital de sinais (DSP - *Digital Signal Processing*). Contudo, a DFT não é computada diretamente, sendo preferido o cálculo da Transformada Rápida de Fourier (FFT - *Fast Fourier Transform*), pois esta é composta de um conjunto de algoritmos com menor custo computacional (menos cálculos) capazes de avaliar a DFT, obtendo valores muito aproximados.

O número de aplicações da FFT inclui praticamente todos os campos da engenharia e da ciência, por prover, por exemplo, análise espectral, solução de equações diferenciais parciais e multiplicação polinomial (HAYKIN; VEEN, 2001; BRACEWELL, 1999). E é nos sistemas de comunicação embarcados em banda larga que sua empregabilidade é mais discutida atualmente devido ao seu consumo de energia elevado (NG et al., 2007).

Dispositivos de comunicação sem fio comumente são alimentados por um elemento armazenador de energia (bateria) ou um pequeno gerador de energia, e possuem capacidade energética limitada. A DFT é uma operação comum na arquitetura de vários sistemas de comunicação sem fio voltados para alta taxa de dados e é, também, um dos blocos destes sistemas com maior consumo de energia (NG et al., 2007). Logo, a redução do consumo de potência na FFT reflete no aumento da duração do tempo de vida útil das baterias.

Em aplicações DSP, dada a natureza dos algoritmos, que incluem um uso intenso de operações aritméticas de soma e multiplicação, as suas implementações geralmente apresentam um elevado consumo de potência, devido principalmente aos circuitos multiplicadores.[1] Desta forma, torna-se importante a aplicação de técnicas para a redução do consumo de potência em circuitos da área de DSP, em diferentes níveis de abstração. O algoritmo da FFT, reduz o número de operações aritméticas no processo da DFT, contudo o volume de operações necessário para computar a FFT ainda é alto. Desta forma, torna-se importante a aplicação de técnicas de baixa potência em circuitos da área de DSP, explorando a otimização em níveis de abstração mais

[1] Uma relação entre o consumo de potência de um somador e um multiplicador é dada no Capítulo 5.

baixos e estes métodos devem incluir a redução da atividade de chaveamento dos operadores aritméticos, por ser um fator diretamente proporcional ao consumo de potência .

Neste trabalho diferentes estruturas e técnicas são avaliadas para reduzir o consumo do operador básico da FFT - a borboleta, através da comparação de diferentes estruturas de borboletas e com o uso de diferentes circuitos de operadores aritméticos. Em particular, são utilizados multiplicadores eficientes do tipo *array* e somadores do tipo compressores nas diferentes estruturas da borboleta da FFT. Para tal, diferentes estruturas da borboleta com decimação no tempo (DET) serão apresentadas. Os operadores aritméticos eficientes serão utilizados nas estruturas internas das borboletas. Nos circuitos somadores compressores, a estrutura interna é basicamente composta por portas lógicas XOR e multiplexadores com reduzido caminho crítico (CHANG; GU; ZHANG, 2004). Devido ao uso intenso, uma nova estrutura de porta lógica XOR é proposta para a redução do consumo de potência nos somadores compressores.

1.1 Motivação

Com a crescente demanda por comunicação móvel, dispositivos eletrônicos embarcados com ampla variedade de funções e alta complexidade tornaram-se um grande mercado mundial. Estes equipamentos caracterizam-se pela capacidade de processamento atrelada ao tempo de duração da carga das baterias, integração de funções variadas e pelas características físicas de fácil maneabilidade e baixo peso.

Equipamentos para comunicação móvel cada vez agregam mais periféricos e funcionalidades necessitando de mais processamento, como por exemplo rádio, jogos, TV digital, internet e outros. Portanto, meios para a otimização do consumo de energia se tornam importantes para a mobilidade não tornar-se dependente da manutenção da carga das baterias (JARDIM, 2007).

Para a viabilização da transmissão de dados sem fio em banda larga (BWA - *Broadband Wireless Access*), esquemas de modulação como a Multiplexação Ortogonal por Divisão de Frequência (OFDM - *Orthogonal Frequency Division Multiplexing*) tem sua empregabilidade necessária, e sistemas de transmissão como o Wi-Fi (IEEE..., 2007), o WiMAX (IEEE..., 2003) e a TV digital Brasileira (ABNT, 2008) usam-na, realizando esta operação através da Transformada Rápida de Fourier (FFT - *Fast Fourier Transform*).

De acordo com (LOCKE, 2007), o uso de sistemas de comunicação em BWA como o WiMAX é a tendência de uso futuro por ser mais barato que outras tecnologias como o 3G e o sistema de comunicação móvel universal (UMTS - *Universal Mobile Telecommunications System*). Isto devido principalmente ao custo do espectro (por apresentar maior eficiência espectral

pela maior densidade de uso), além de outras vantagens, tais como melhor imunidade a ruídos gerados por multipercurso do sinal e modulação adaptativa (diferentes técnicas de modulação para usuários com diferente relação sinal/ruído). Contudo, o uso do WiMAX como substituto do 3G e do UMTS requer o uso de FFTs com mais de 1000 pontos de computação [2], o que requer uma operação eficiente em consumo de energia (LOCKE, 2007).

Partindo do pressuposto de que sistemas de comunicação em banda larga são amplamente utilizados, e o seu uso cresce cada vez mais, assim há demanda por sistemas com o uso da FFT mais complexos. Logo, reduzir o consumo de energia de um sistema de comunicação de dados melhora a mobilidade do usuário por aumentar o tempo de carga das baterias dos equipamentos portáteis. Dessa forma, o estudo de uma FFT com a aplicação de técnicas que possibilitem uma redução no consumo de energia se torna a principal motivação deste trabalho.

1.2 Objetivos

O objetivo deste trabalho é a aplicação de diferentes técnicas de redução de potência em circuitos voltados à área de DSP. O estudo busca explorar a viabilidade de diferentes alternativas de arquiteturas específicas para algoritmos da área DSP, com aplicação de técnicas de redução de potência. Em particular, uma classe específica de algoritmo FFT é investigado. O principal objetivo é reduzir o consumo de potência dos circuitos a partir da aplicação de técnicas voltadas à redução da atividade de chaveamento. Neste contexto, investiga-se a utilização de técnicas de codificação de dados em operadores aritméticos que são utilizados como módulos básicos nos circuitos da borboleta da FFT. Em particular, é utilizada a codificação/ decodificação RNS - *Residue Number System*.

A codificação RNS consiste no uso dos resíduos dos números para os cálculos das operações aritméticas. Como a codificação RNS divide o número em partes e processa cada parte separadamente, ela fornece uma grande paralelização do sistema. No caso de operações de soma, este aspecto contribui de forma significativa para a redução da propagação do *carry*, o que ocasiona em um aumento na velocidade desta operação. Portanto a codificação RNS pode ser uma boa alternativa para circuitos da área DSP, tais como FFT, que utilizam um número significativo de somadores e multiplicadores.

Nas estruturas internas das borboletas da FFT com decimação no tempo implementadas neste trabalho, os caminhos críticos são compostos por estruturas de circuitos somadores em cascata. Desta forma, utilizam-se somadores compressores eficientes que têm por objetivo a

[2] Pontos de amostra ou entrada.

soma simultânea de vários operandos, o que possibilita reduções significativas no caminho crítico das estruturas implementadas. Como estes somadores compressores são compostos, em suas estruturas internas, por portas lógicas XOR e circuitos multiplexadores, apresenta-se neste trabalho uma nova estrutura de porta XOR, cujo principal objetivo é a redução do consumo de potência dos somadores compressores.

1.3 Metodologia

Diferentes circuitos aritméticos digitais foram otimizados e/ou adaptados para as estruturas em teste para a comparação no consumo de potência. Os circuitos digitais foram escritos em linguagem de descrição de hardware (HDL - *Hardware Description Language*) Verilog, que também foi usada na criação dos programas de verificação de funcionalidade. Exemplos de circuitos codificados em Verilog podem ser encontrados no Apêndice A.

O mapeamento (ou síntese lógica) foi realizado utilizando a tecnologia de 180nm da biblioteca XFAB voltada pra baixa potência, com 1,8V de alimentação (X-FAB, 2008), seguindo-se o fluxo de projeto de circuitos digitais para baixa potência sugerido pelas ferramentas de síntese (CADENCE, 2008), que incluem análise de chaveamento pré e pós-síntese lógica. Apenas os métodos de ativação do circuito pelo relógio do sistema (*clock gating*) e isolação de operandos não foram utilizados, pois respectivamente o sistema em estudo aqui apresentado não possui grandes lógicas sequenciais nem operandos em desuso.

Para o teste da nova topologia de porta XOR, apresentada na Seção 4.3.5, aplicada em operadores aritméticos, utilizou-se ferramentas de simulação de circuitos analógicos, pois a sua caracterização[3] através das ferramentas padrão CMOS (Semicondutor de metal-óxido complementar, *Complementary Metal-Oxide-Semiconductor*), as quais as ferramentas atuais de caracterização estão voltadas, se mostrou inviável devido à incompatibilidade de sua topologia, por ser construída utilizando um padrão misto CMOS e lógica de passagem.

1.4 Contribuições do trabalho

A partir do estudo realizado para a exploração das diferentes alternativas arquiteturais na borboleta da FFT com decimação no tempo, com a aplicação de técnicas de redução de potência, apresentam-se as seguintes contribuições:

- Nova estrutura de multiplicador complexo com três multiplicadores reais. A maioria das

[3] Determinação dos parâmetros para modelagem de um circuito.

aplicações utiliza multiplicações complexas com quatro multiplicadores reais. Neste trabalho foi realizada uma exploração arquitetural, onde foi possível verificar o impacto da redução do número de multiplicadores reais (três multiplicadores) no aumento de desempenho;

Aplicação desta nova estrutura em borboletas DET na base 2.[4]

- Teste e comparação de diferentes estruturas de borboletas DET na base 2 com três e quatro multiplicadores reais. Neste ponto do trabalho foi possível estabelecer o relacionamento entre a redução do número de multiplicadores reais com o aumento do caminho crítico. Pôde-se perceber que o aumento do caminho crítico pode ser compensado com o uso de estágios de *pipeline*.

- Teste e comparação de diferentes operadores aritméticos, multiplicadores *Array* e CSA, somadores CSA, compressores e codificação em RNS aplicados em borboletas DET na base 2.

- Nova topologia de porta XOR. Esta nova topologia é baseada em lógica de passagem mista com lógica CMOS (Seção 4.3.5);

  Aplicação desta nova estrutura de porta XOR em circuitos somadores compressores. Os circuitos somadores compressores com a nova topologia de porta XOR foram aplicados nas borboletas da FFT.

- Desenvolvimento de rotinas para a estimativa de consumo de potência com análise da atividade de chaveamento. A atividade de chaveamento de um circuito digital influencia diretamente no seu consumo de potência (Seção 2.3.2.1), logo esta análise é importante para uma medida de consumo de potência mais acurada. A estimativa foi realizada em simulação antes e após a síntese física das estruturas de borboletas.

- Desenvolvimento de programas para verificação funcional de operadores aritméticos. São importantes para a validação dos operadores a serem aplicados nas borboletas. Esses programas foram escritos em Verilog. No Apêndice A pode ser encontrado um exemplo de programa para verificação de um circuito multiplicador.

1.5 Organização

No Capítulo 2 são apresentadas fundamentações teóricas para o entendimento da FFT e de seu núcleo, bem como do consumo de potência em circuitos CMOS. Técnicas para a Redução

[4] A definição de borboletas DET na base 2 encontra-se na Seção 2.2.1.

de Potência em circuitos digitais são apresentadas no Capítulo 3. O Capítulo 4 demonstra topologias originais e adaptadas de sistemas digitais aritméticos aplicados nas borboletas da FFT. Os resultados das simulações aplicadas aos dispositivos do quarto capítulo são apresentados no Capítulo 5 e no Capítulo 6 são comentadas as conclusões e idéias para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os principais aspectos que contribuem para as parcelas de consumo de potência em circuitos CMOS. Também são apresentados os principais conceitos sobre a Transformada de Fourier, sua versão computacional, a FFT, as equações básicas para o cálculo da mesma e o elemento constituinte do cálculo da sua implementação, a borboleta.

2.1 Transformada Rápida de Fourier

A Transformada Rápida de Fourier (FFT) é utilizada por ser uma alternativa de construção computacional mais eficiente em relação à Transformada Discreta de Fourier (DFT - *Discrete Fourier Transform*) por apresentar menos operações de multiplicação e soma em seu algoritmo (COOLEY; TUKEY, 1965; HAYKIN; VEEN, 2001).

A DFT é a versão computacional da Transformada de Fourier (TF), pois o cálculo computacional da TF é inviável devido à necessidade de uma memória infinita, pois tanto o sinal original $f(x)$ da Figura 1 e sua transformada $F(S)$ na Figura 1 são contínuos. Para que seja possível a realização da DFT, no lugar da TF, torna-se necessário amostrar o sinal de entrada $f(x)$ e aplicar a DFT em trechos de P pontos.

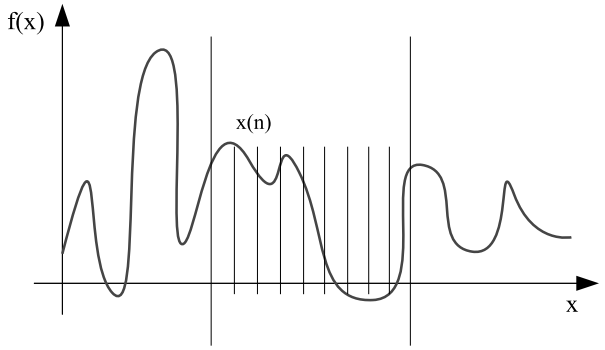


Figura 1: Sinal contínuo $f(x)$ e sua amostra $x(n)$ com P pontos

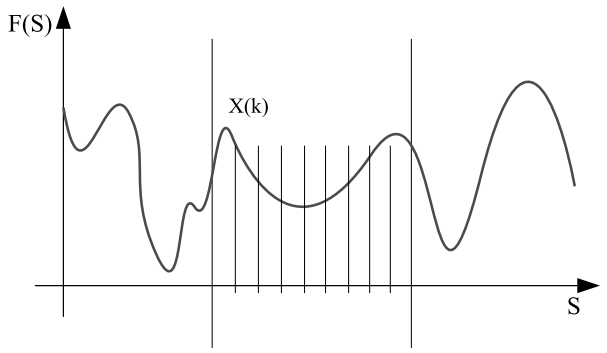


Figura 2: Sinal contínuo $F(S)$ e sua amostra $X(k)$ com P pontos

Amostrando-se o sinal $f(x)$ da Figura 1 obtém-se o sinal $x(n)$, $\forall\ 0 \leqslant n \leqslant P-1$ [1] e aplicando-se a DFT neste sinal obtém-se a função $X(k)$ da Figura 2, nas Equações 2.1 e 2.2.

$$X(k) \leftrightarrows DFT \leftrightarrows x(n),\ \forall\ 0 \leqslant k \leqslant N-1\ \&\ \forall\ 0 \leqslant n \leqslant N-1 \tag{2.1}$$

A Equação 2.3 apresenta a Transformada de Fourier Discreta Inversa (IDFT - *Inverse Discrete Fourier Transform*). Os índices n e k são reais e inteiros e definem um ponto de amostra nas funções em que são empregados.

$$X(k) = \sum_{n=0}^{P-1} x(n) \cdot W_P^{k.n} \tag{2.2}$$

$$x(n) = \frac{1}{P} \sum_{k=0}^{P-1} X(k) \cdot W_P^{-k.n} \tag{2.3}$$

$$W_P = e^{\frac{-i \cdot 2 \cdot \pi}{P}} \tag{2.4}$$

O termo W_P presente nas Equações 2.2 e 2.3 é conhecido como *twiddle factor* e pode ser calculado a partir da Equação 2.4.

A DFT opera numa sequência de P pontos, referida a $x(n)$. Este intervalo pode (usualmente) ser tomado como uma amostragem de tamanho uniforme e período finito da função $f(x)$. A DFT de $x(n)$ é também uma sequência de P pontos, dada por $X(k)$ na Equação 2.2. As funções $x(n)$ e $X(k)$ são, em geral, complexas. Para aplicações em sistemas de comunicação, a FFT realiza a multiplexação de um sinal proveniente de um bloco de modulação, em que a modulação cria e separa as partes real e imaginária do sinal de entrada (RAO; RADHAMANI, 2008). Mais detalhes sobre as relações entre transformadas e séries de Fourier podem ser encontradas em

[1] Lê-se para todo n maior ou igual a zero e menor ou igual a $P-1$.

(HAYKIN; VEEN, 2001).

A Figura 3 mostra o fluxo a estrutura básica de um algoritmo de FFT na base 2 com decimação no tempo (DET) de 8 pontos ($P = 8$). Neste algoritmo, os valores das amostras no tempo são decimados durante cada estágio da FFT. As operações são efetuadas em um par de sinais de entrada. Na estrutura mostrada na Figura 3, uma borboleta[2] representa a parte central de cálculo da FFT aproveitando-se da simetria da DFT. A complexidade da borboleta está associada ao tipo de algoritmo utilizado. Maiores detalhes sobre a estrutura da borboleta utilizada neste trabalho são apresentados na Seção 2.2.

O termo DET deve-se ao fato de ser necessária uma reorganização da sequência do vetor de entrada, na Figura 3 os elementos da entrada encontram-se com sua ordem alterada e os elementos da saída estão em ordem. Isto deve ocorrer para o aproveitamento da simetria da DFT para o cálculo da FFT, separando as entradas pares das ímpares a cada nível de borboletas. Uma FFT também pode ser Decimada Em Frequência (DEF), com reordenação dos elementos de saída, entrada ordenada e uma alteração na borboleta. Mais detalhes sobre a obtenção do algoritmo da FFT e a obtenção da base da borboleta podem ser encontrados em (BAAS, 1999).

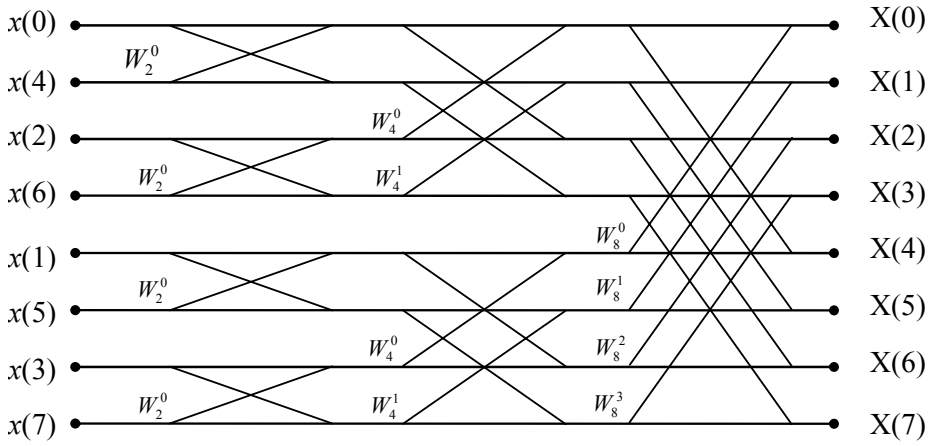


Figura 3: FFT de 8 pontos construída com borboletas com decimação no tempo - DET na base 2

A coluna da direita de borboletas na Figura 3, conectada às saídas, agrupa as entradas pares acima e ímpares abaixo, no nível central de borboletas altera-se novamente, (pois a entrada $x(2)$ agora é ímpar e $x(5)$ é par), e o nível da esquerda conectado à entrada não altera a ordem pois cada DFT possui apenas duas entradas não tendo como reagrupar em pares e ímpares. Mais detalhes sobre a obtenção do algoritmo da FFT e a obtenção da raiz borboleta podem ser encontrados em (BAAS, 1999). No cálculo da FFT, a borboleta é o operador chave da trans-

[2]Cada cruzamento na figura em forma de xis, denomina-se borboleta pela sua semelhança em desenho com as asas do inseto, também chamada de ampulheta.

formada, bloco capaz de utilização recursiva, tal como a unidade lógica e aritmética (ULA)[3] de um microprocessador é usada.

2.1.1 Implementação da FFT

As partes Reais e Imaginárias do termo $W_P^{k.n}$ da Equação 2.2 podem ser armazenados em memória somente de leitura (ROM - *Read Only Memory*), pois os fatores envolvidos em seu cálculo são constantes matemáticas e o número P, que é o número de pontos da FFT, pode ser previamente estipulado, juntamente com os números n e k. Os valores dos *twiddle factors* a serem armazenados são dados pela Equação 2.5 .

$$\begin{aligned} W(n) = W_P^n = e^{\frac{-i\cdot 2\cdot \pi \cdot n}{P}} \\ W_r(n) = \Re(W(n)) \\ W_i(n) = \Im(W(n)) \end{aligned} \tag{2.5}$$

A Equação 2.5 determina os valores reais $W_r(n)$ e valores imaginários $W_i(n)$ de $W(n)$ a serem armazenados para uma FFT com P pontos, posteriormente citados apenas como W_r, W_i e W. Nota-se que os valores de k não precisam ser inseridos na equação, pois para uma FFT $X(k)$, k valores de W_P serão necessários, sendo portanto k a variável de seleção dos valores de W na ROM. Uma rotina no programa Matlab para a obtenção e visualização dos *twiddle factors* é apresentada no Anexo B.

Para os valores de $W_P^{-k.n}$ da Equação 2.3 do cálculo da IFFT, não é necessário nenhum armazenamento de dados extra caso também seja necessário o cálculo da IFFT[4] pois os valores reais e imaginários são iguais em módulo aos valores reais e imaginários de $W_P^{k.n}$, Equação 2.2, sendo necessário um pequeno circuito extra para o cálculo. De forma análoga, para os valores de $W_r(n)$ e $W_i(n)$ não é necessário que ambos sejam armazenados, bastando um deles ser armazenado para a obtenção do outro, pois a diferença de valores está agora relacionada à posição de memória, o que igualmente exige uma lógica extra para a leitura do dado.

Caso a aplicação da FFT seja com pontos P variáveis, basta armazenar o maior número de W's para satisfazer a FFT com o maior número de pontos. Outras técnicas para a redução do número de W's armazenados podem ser vistas em (KIM; PARK, 2007).

Tendo-se os valores de W armazenados em ROM, a FFT pode ser realizada utilizando a

[3] Mais detalhes sobre a unidade lógica e aritmética podem ser encontrados em (HENNESSY; PATTERSON, 2002)

[4] A camada PHY dos sistemas de comunicação geralmente implementa recepção e transmissão, usando a FFT e a IFFT, respectivamente

estrutura borboleta tendo como única entrada o sinal a ser transformado, pois os valores de W são lidos da memória.

O número de borboletas na FFT aumenta de forma logarítmica, conforme o aumento do número de pontos P, de acordo com a expressão: $\frac{P}{2}\log_2 P$. Uma implementação com $P = 8$, como a da Figura 3 não é utilizada na prática, pois a maioria das aplicações requer valores de P entre 64 e 2048.[5]. Logo, para $P = 64$, cento e noventa e duas borboletas são necessárias para uma implementação totalmente paralela (6 estágios com 32 borboletas cada estágio), o que se torna inviável, devido à grande quantidade de recursos de hardware necessária. Estes detalhes de implementação são mais explorados no Capítulo 5, onde resultados de área são apresentados.

Devido à inviabilidade física de construção de uma FFT totalmente paralela, diversas topologias de construção da FFT são discutidas atualmente na literatura (COSTA, 2002; BAAS, 1999; GONZALEZ-CONCEJERO et al., 2008; CHHATBAR, 2009; ZHAO; ERDOGAN; ARSLAN, 2005), sendo o modelo *radix-2 single-path delay feedback* (R2SDF) um dos mais utilizados.

Embora existam diversas formas de implementar a FFT, todas as topologias utilizam a borboleta como operador básico, o que justifica o estudo no sentido de otimização da mesma.

2.1.2 Multiplicação complexa

Como os operandos de $x(n)$, $X(k)$ e W_P nas Equações 2.2 e 2.3 são complexos, para a aplicações BWA, seu produto é calculado através de uma multiplicação complexa. Uma multiplicação complexa representa uma multiplicação polinomial, em que cada operador apresenta dois elementos, a parte Real e a parte Imaginária.

Na Equação 2.6 é apresentada a multiplicação entre os operandos B e W. Sendo B e W duas variáveis complexas quaisquer $B = B_r + i \cdot B_i$ e $W = W_r + i \cdot W_i$ com suas partes reais B_r e W_r e suas partes imaginárias B_i e W_i, respectivamente e i representando o operador complexo $\sqrt{-1}$.

$$B \cdot W = (B_r + i \cdot B_i) \cdot (W_r + i \cdot W_i) \tag{2.6}$$

Multiplicando os termos da Equação 2.6, encontra-se a Equação 2.7. Nesta equação, a solução da multiplicação complexa apresenta quatro multiplicadores reais e três somadores.

[5] Aplicações com diferentes valores de P nas telecomunicações são encontradas em (ANDREWS; GHOSH; MUHAMED, 2007), (NG et al., 2007), (IEEE..., 2003) e (ABNT, 2008), por exemplo.

$$B \cdot W = (B_r \cdot W_r) + i \cdot (B_r \cdot W_i) + i \cdot (B_i \cdot W_r) + (B_i \cdot W_i) \tag{2.7}$$

Na Equação 2.7 a solução da multiplicação complexa encontrou quatro multiplicadores reais e três somadores, entretanto, para a criação de circuitos em hardware, as variáveis reais e imaginárias são tratadas e armazenadas de forma separada, gerando duas respostas, Equações 2.8 e 2.9.

$$(B \cdot W)_r = (B_r \cdot W_r) + (B_i \cdot W_i) \tag{2.8}$$

$$(B \cdot W)_i = i \cdot [(B_r \cdot W_i) + (B_i \cdot W_r)] \tag{2.9}$$

A partir das Equações 2.8 e 2.9, observa-se a necessidade do uso de quatro multiplicadores e dois somadores para a descrição de um circuito em hardware de um multiplicador complexo. Na Seção 2.2.1 e na Figura 5 é apresentada a estrutura de uma borboleta para essa solução. Formas não triviais de realização de uma multiplicação complexa incluem a reordenação algébrica dos termos das Equações 2.8 e 2.9, chegando-se à solução de uma multiplicação complexa com três multiplicadores reais e alguns somadores (ou subtrações) adicionais.

Dezesseis algoritmos distintos para a realização de uma multiplicação complexa, utilizando três multiplicadores reais, são apresentados em (WENZLER; LUDER, 1995). Contudo, esses algoritmos podem ser agrupados em apenas três grupos de estruturas diferentes em construção de hardware. Isso se for considerada a igualdade entre circuitos somadores e subtratores e desconsiderada a alteração de ordem entre os sinais de entrada. Na Seção 4.2.2 são apresentadas as borboletas usando estes esquemas de multiplicação.

2.2 Estrutura da borboleta

A borboleta é constituída de circuitos aritméticos e representa o operador central de uma FFT. A sua estrutura interna é composta por um elevado número de circuitos aritméticos, o que pode acarretar em um grande consumo de potência, principalmente devido aos circuitos multiplicadores. Este elevado número de operadores aritméticos inviabiliza a implementação de uma FFT na forma totalmente paralela. para a avaliação da FFT.

Nesta seção são apresentados os algoritmos para a implementação da borboleta de uma FFT na base 2, com decimações no tempo e na frequência.

2.2.1 Borboleta DET na base 2

Na Figura 4 está representada a estrutura da borboleta na base 2 com decimação no tempo (DET), onde A e B são entradas complexas, W é a constante complexa da Equação 2.4 e C e D são as saídas complexas. As saídas complexas são apresentadas nas Equações 2.10 e 2.11.

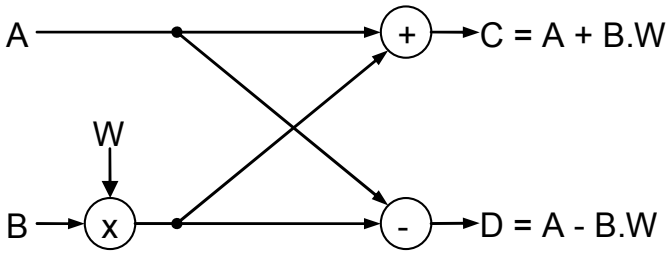


Figura 4: Diagrama do operador borboleta DET na base 2

Como as entradas *A, B e W* são valores complexos, as operações são complexas e as saídas também serão, tendo as suas representações dadas pelas Equações 2.10 e 2.11.

$$C = A + B \cdot W = (A_r + i \cdot A_i) + [(B_r + i \cdot B_i) \cdot (W_r + i \cdot W_i)] \tag{2.10}$$

$$D = A - B \cdot W = (A_r + i \cdot A_i) - [(B_r + i \cdot B_i) \cdot (W_r + i \cdot W_i)] \tag{2.11}$$

Para as Equações 2.10 e 2.11 i é o operador complexo $\sqrt{-1}$ e cada variável possui uma parte real, representada pelo sub-índice r e uma imaginária, representada pelo sub-índice i.

Efetuando as multiplicações e separando as partes reais e imaginárias obtém-se as Equações 2.12 e 2.13.

$$C = A_r + (B_r \cdot W_r - B_i \cdot W_i) + i \cdot [A_i + (B_i \cdot W_r + B_r \cdot W_i)] \tag{2.12}$$

$$D = A_r - (B_r \cdot W_r - B_i \cdot W_i) + i \cdot [A_i - (B_i \cdot W_r + B_r \cdot W_i)] \tag{2.13}$$

Observa-se pelas equações 2.12 e 2.13 que há termos comuns que podem ser reaproveitado o cálculo (necessitam ser calculados apenas uma vez). Portanto, a borboleta DET na base 2 apresenta em sua implementação 4 multiplicadores, 3 somadores e 3 subtratores reais, conforme pode ser observado na Figura 5. Devido a construção física dos somadores e subtratores ser quase semelhante, frequentemente são encontrados na literatura apenas como somadores.

Tendo-se então 4 multiplicadores e 6 somadores, todos reais.

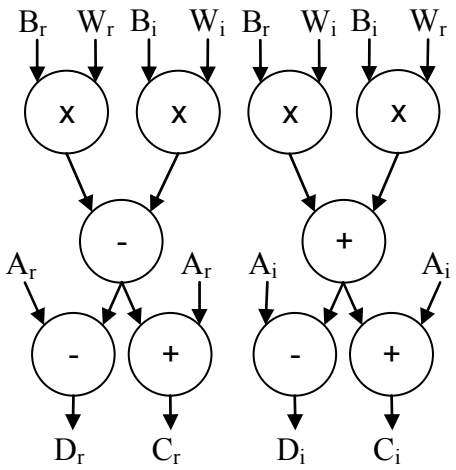


Figura 5: Estrutura de operadores para borboleta DET na base 2 com 4 multiplicadores, 3 somadores e 3 subtratores - Estrutura A

2.2.2 Borboleta DEF na base 2

A borboleta com decimação em frequência (DEF) apresentada na Figura 6, com entradas *A*, *B* e *W* e saídas *C* e *D* (todas complexas) possui sua construção irregular, ou seja, o caminho do sinal através da estrutura não é uniforme, pois a multiplicação é realizada apenas para uma saída levando a tempos distintos do cálculo da resposta.

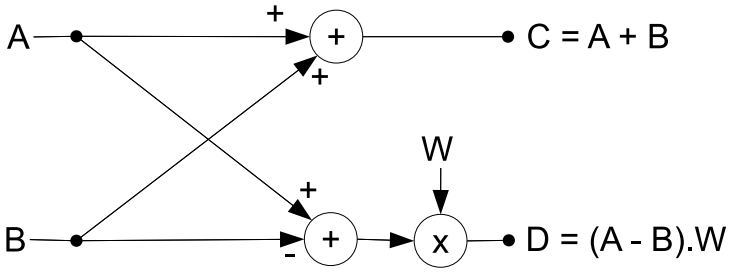


Figura 6: Diagrama da borboleta DEF na base 2

Para esta borboleta, as técnicas de diminuição do número de multiplicadores reais aplicadas na borboleta DET apresentadas neste trabalho na Seção 4.2.2 e no Capítulo 4 também podem ser aplicadas. Contudo, a aplicação de circuitos somadores eficientes[6] não pode ser adotada, devido a soma presente antes da multiplicação no fluxo de sinal da Figura 6. Essa soma não pode ser agrupada na soma do multiplicador complexo, sob pena do aumento do número de multiplicadores reais.

[6]Descritos no Capítulo 3.

Devido a estes motivos, caminho irregular do sinal e incapacidade de otimização dos somadores, a borboleta DEF não é explorada neste trabalho.

2.3 Consumo de potência em circuitos digitais CMOS

O consumo de potência é um dos principais parâmetros no projeto de circuitos digitais. Neste contexto, a potência de pico é utilizada como um dos pontos de análise da confiabilidade dos circuitos. Entretanto, o fator mais crítico é o consumo médio de potência dos circuitos que operam durante um determinado tempo [CHA 1995]. Nesta parte do trabalho, investiga-se as principais fontes de consumo de potência em circuitos CMOS. A tecnologia CMOS é particularmente abordada, por ser utilizada na maior parte dos projetos de circuitos digitais e por combinar aspectos de alto desempenho com baixo consumo de potência.

Para entender o consumo de energia em circuitos digitais é necessário conhecer as características construtivas e elétricas dos mesmos, que podem ser estudadas em (RABAEY, 2009). Nesta Seção são apresentados os pontos básicos das principais fontes de consumo de potência em circuitos CMOS.

2.3.1 Potência estática

O consumo de potência estática ou *Leakage* ocorre devido às correntes de fuga presentes nos transistores MOSFET (*Metal Oxide Semiconductor Field Effect Transistor*) e nos elementos parasitas que são criados pela conexão com o substrato. Esta parcela do consumo de potência está presente no circuito mesmo quando não há alteração dos valores de entrada (condição de operação em *standby*). Mais detalhes podem ser encontrados em (COSTA, 2002). Devido à tecnologia usada nas simulações presentes neste trabalho (180nm), este tipo de consumo de potência se torna desprezível por ser mais de mil vezes menor que a potência dinâmica.

2.3.2 Potência dinâmica

A potência dinâmica é proveniente dos consumos de potência provocados pelas alterações nos estados do circuito, quando há atividade de chaveamento nas suas entradas. É produzida pelas cargas e descargas das capacitâncias do circuito, a partir da atividade de chaveamento, pelo curto-circuito que ocorre durante a comutação do sinal de entrada, quando flui uma corrente diretamente da fonte de alimentação para o terra e pela dissipação de potência pelas transições espúrias, chamadas de *glitches* ou *hazards*.

2.3.2.1 Potência pela carga e descarga das capacitâncias

Esta parcela de consumo de potência é apresentada na Equação 2.14 (RABAEY, 2009), onde P_{CH} é a potência de chaveamento dissipada por um circuito digital usando lógica CMOS, C_{ap} é a capacitância de carga no nó de saída, V é a tensão de alimentação do circuito, f é a frequência de operação e A_t é a atividade do nó de saída, medida em eventos por segundo para uma carga-descarga completa.

$$P_{CH} = \frac{1}{2} \cdot C_{ap} \cdot A_t \cdot V^2 = P_{CH} = \frac{1}{2} \cdot a_t \cdot f \cdot C_{ap} \cdot V^2 \tag{2.14}$$

No caso de projetos síncronos, a atividade A_t não representa simplesmente f (frequência), mas em geral representa uma probabilidade de atividade normalizada a_t (menor do que 1 para modelo de atraso zero) e é computada em função da estatística de entrada e modelos lógicos, pois nem todos os nós mudam em um determinado ciclo de relógio.

2.3.2.2 Potência de curto-circuito

A potência de curto-circuito é causada pelo acionamento simultâneo por um curto intervalo de tempo dos transistores tipos PMOS *P Channel Metal Oxide Semiconductor* e NMOS *N Channel Metal Oxide Semiconductor*. Isto ocorre quando um circuito CMOS estático é chaveado por um sinal de entrada com tempos de subida e descida não zero, o que acarreta numa corrente de curto circuito que flui diretamente da fonte de alimentação para o terra. Este efeito pode ser visualizado na Figura 7, com as curvas aproximadas das correntes de curto-circuito para um circuito inversor.

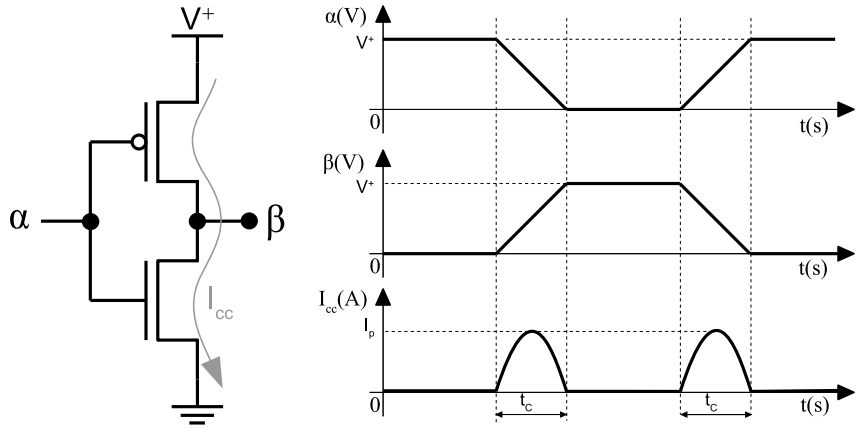


Figura 7: Exemplo de corrente de curto-circuito em um inversor CMOS

Na Figura 7 é ilustrado o processo de curto-circuito intrínseco da topologia CMOS para um circuito inversor, sendo α a entrada, β a saída do circuito e I_{CC} a corrente de curto-circuito que

ocorre no curto intervalo de tempo t_C. Esta corrente é limitada pelas resistências dos dispositivos e das interconexões. A potência de curto-circuito é dada pela Equação 2.15 (RABAEY, 2009).

$$P_{CC} = I_{CC} \cdot V \cdot t_C \cdot f \tag{2.15}$$

Nota-se, pela equação 2.15 que a potência de curto-circuito também é diretamente proporcional à frequência de operação f. A potência de curto-circuito também depende do valor de pico da corrente de curto-circuito I_{CC}, da tensão de operação do circuito e da duração do intervalo de comutação t_C.

2.3.2.3 Atividade de *glitching*

O consumo de potência pela atividade de *Glitching* ou *Hazards* está relacionado a transições múltiplas que ocorrem em nós dos circuitos em um ciclo de relógio, antes de alcançar o nível lógico correto. Estas múltiplas transições (chamadas de transições espúrias) estão relacionadas aos atrasos característicos de propagação entre portas lógicas ou blocos de circuitos CMOS. A propagação de transições espúrias acarreta em consumo extra de potência dinâmica devido ao aumento da atividade de chaveamento no circuito (mais correntes de curto-circuito e mais cargas e descargas das capacitâncias de saída).

A atividade de *glitching* pode ser melhor entendida pelo exemplo da Figura 8. Nesta figura, com entradas α_1, α_2 e α_3 quaisquer, é gerada uma saída δ com sinais intermediários β e γ. Ocorre que, quando são alterados os valores de entrada, devido aos diferentes tempos de propagação dos sinais pelas diferentes portas lógicas e conexões, os sinais intermediários γ_1 e γ_2 apresentam resultados que são calculados em tempos diferentes, levando a porta lógica da saída a computar resultados intermediários. O chaveamento extra dos transistores que compõe estas portas lógicas é o que causa o consumo de potência adicional.

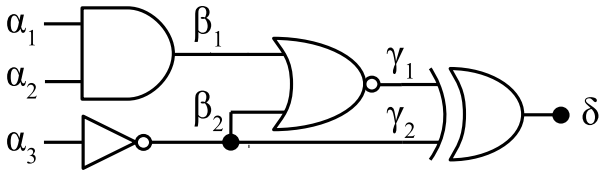


Figura 8: Exemplo de circuito digital

Uma forma de minimizar os sinais espúrios nos circuitos é reduzindo o caminho crítico, acrescentando barreiras de registradores em locais apropriados. Esta técnica é chamada *pipeline* e é explicada com maiores detalhes na Seção 3.1. Outra possibilidade é alterar a lógica digital no nível de transistores, usando lógica de passagem, como será visto na Seção 4.3.5. Outras

técnicas podem ser vistas em (RABAEY, 2009), contudo não são exploradas neste trabalho.

2.3.3 Relação de potência entre diferentes tecnologias

De acordo com (RABAEY, 2009), relações entre diferentes tecnologias podem ser estabelecidas e os parâmetros de uma tecnologia podem ser projetados para outra. Em um cenário de escalonamento integral, onde todos parâmetros alteram-se de acordo com a tecnologia, a Equação 2.16 pode ser calculada.

$$\begin{aligned} E &= \frac{Tecnologia1}{Tecnologia2}, \forall E > 1 \\ P_1 &= \frac{1}{E^2} \cdot P_2 \end{aligned} \tag{2.16}$$

A Equação 2.16 é empregada aplicando a razão entre as tecnologias E na determinação da Potência. A potência altera-se quadraticamente, pois a tensão e a corrente alteram-se na proporção de $1/S$. Em cenários onde o escalonamento não é integral, o cálculo de E pode ser dado pela relação entre as tensões de alimentação das tecnologias.

2.4 Resumo do capítulo

Neste capítulo de fundamentação teórica foram explorados conceitos sobre a Transformada de Fourier, sua versão computacional, a FFT, as fórmulas básicas de cálculo para a mesma e o elemento constituinte do cálculo de sua implementação, a borboleta, juntamente com suas representações gráficas e equações. Também foram apresentadosos os principais aspectos do consumo de potência dos circuitos digitais CMOS. Foram abordados aspectos das principais fontes de consumo de potência nos circuitos, ou seja, potência estática e potência dinâmica, com ênfase para os efeitos de consumo de potência pelo curto-circuito e pela atividade de chaveamento (consumo de potência na capacitância de carga e consumo de potência pela atividade de *glitching*). No capítulo seguinte são descritas as técnicas e circuitos para a otimização do consumo de potência aplicadas nas borboletas deste trabalho.

3 TÉCNICAS UTILIZADAS PARA A OTIMIZAÇÃO DA BORBOLETA DET NA BASE 2

O consumo de potência de um sistema digital depende das decisões de projeto tomadas em diferentes níveis do fluxo de desenvolvimento de um sistema digital, pois diferentes técnicas podem ser aplicadas nos variados níveis. Os níveis de abstração (Arquitetura, Algoritmo ou Micro Arquitetura, Função, Lógica e Circuito) de um sistema digital são ilustrados na Figura 9 baseada no modelo proposto por (GAJSKI; KUHN, 1983) e o fluxo de projeto é dado pelo caminho traçado entre os nós da Figura.

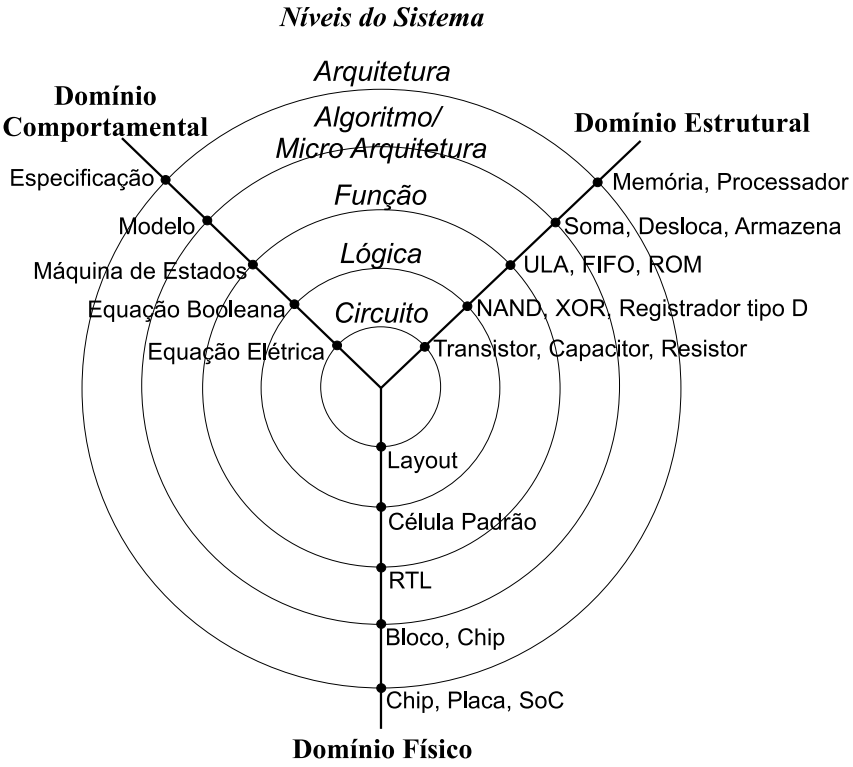


Figura 9: Níveis de abstração em um sistema digital

Na Figura 9 tem-se a visualização dos níveis de abstração em um sistema digital com os

domínios Físico, Comportamental e Estrutural discriminados, e suas divisões de acordo com o nível do sistema. Na mesma Figura, algumas siglas representam o domínio de acordo com o nível de abstração do sistema. A sigla RTL, por exemplo, refere-se ao Nível de Transferência de Registradores (*Register Tranfer Level*). Por outro lado, FIFO é um acrônimo para o inglês *First In, First Out* (primeiro a entrar, primeiro a sair) que é um tipo de memória para os dados temporários. Finalmente, SoC é a sigla para o inglês *System on Chip* (sistema em chip).

De acordo com (CHANDRAKASAN; SHENG; BRODERSEN, 1992), o principal aspecto relacionado com a redução do consumo de potência é o estabelecimento da tensão da fonte de alimentação em limites mínimos, sendo estes limites associados com a tecnologia CMOS utilizada. Estes limites devem levar em consideração aspectos como as mudanças nas condições de operação do circuito como a variação da temperatura ou ruído, além do aspecto do atraso característico, que tende a aumentar com a redução da tensão de alimentação.

Outro aspecto importante a ser considerado na redução do consumo de potência de circuitos CMOS é a atividade de chaveamento. Enquanto que a atividade de chaveamento é funcional, isto é, requerida para propagar e manipular informações, há uma quantidade substancial de atividades não úteis em circuitos digitais e que devem ser minimizadas em projetos de baixa potência. Um dos principais aspectos que contribui para estas atividades não úteis está relacionado com transições espúrias devido a atrasos de propagação de *glitches*. Uma das formas de minimizar a atividade de *glitching* em circuitos digitais é a utilização da técnica de *pipeline*, que é composta da utilização de barreiras de registradores em locais estratégicos do circuito. Esta técnica será explicada com maiores detalhes na Seção 3.1.

O projeto de circuitos integrados de baixa potência exige a combinação de técnicas nos diferentes níveis de abstração. Estas técnicas se baseiam principalmente em modelos de atraso simplificados e na modelagem de potência a partir do comportamento dos sinais lógicos com probabilidades de transições. A precisão nos resultados destas técnicas está associada ao nível de abstração. Nos níveis mais baixos de abstração (função, circuitos, lógico), pode-se ter uma avaliação precisa do consumo de potência, pois são disponíveis mais detalhes de implementação dos circuitos. Por outro lado, nos níveis mais altos de abstração (sistema, arquitetura e algoritmo), uma precisão relativa do consumo de potência é mais importante do que uma precisão absoluta, visto que nestes níveis de abstração o que se deseja realmente saber é se uma alternativa de projeto é melhor do que outra.

Neste trabalho, são exploradas técnicas de redução de potência em diferentes níveis de abstração para o algoritmo de Transformada Rápida de Fourier. Neste algoritmo, a borboleta representa a parte central de cálculo das amostras. Como a borboleta é composta por um número

elevado de operadores aritméticos, utiliza-se neste trabalho operadores aritméticos eficientes para o aumento de desempenho e redução do consumo de potência da borboleta. Um destes operadores aritméticos é o somador compressor. Este somador pode realizar a soma simultânea de mais de dois operandos. Em particular, neste trabalho são utilizados circuitos somadores compressores 3:2 e 4:2, que realizam a soma simultânea de três e quatro operandos respectivamente. Estes somadores compressores são utilizados nas estruturas internas das borboletas. Estes somadores compressores são compostos, em suas estruturas internas, por portas XOR e circuitos multiplexadores. Neste trabalho, são propostas novas estruturas de portas XOR para serem utilizadas nos somadores compressores e reduzir os seus consumos de potência. A idéia é a redução do consumo de potência das borboletas na base 2 com decimação no tempo, a partir da utilização dos somadores compressores com as novas portas XOR propostas. A seguir são apresentadas as principais técnicas de redução de potência utilizadas neste trabalho.

3.1 *Pipelining*

Pipelining é uma técnica para o aumento de desempenho de circuitos digitais, e que pode também ser utilizada para a redução de potência, atuando na redução da atividade de sinais espúrios de circuitos que apresentam uma grande profundidade lógica. A Figura 10 mostra um exemplo de estrutura de *pipeline*, que opera pela divisão do sistema em estágios ou blocos, armazenando o resultado de cada estágio em registradores intermediários. Estes registradores podem ser controlados pelo sinal de relógio do sistema (sistema síncronos) ou controlados por variáveis do próprio sistema (assíncronos). Neste trabalho os estágios de *pipeline* são síncronos devido ao fato da leitura dos resultados de saída ocorrerem em intervalos de tempo também controlados pelo relógio do sistema, possibilitando a escrita em um ciclo de relógio e a leitura alguns ciclos após (de acordo com o número de estágios de *pipeline* existentes no sistema).

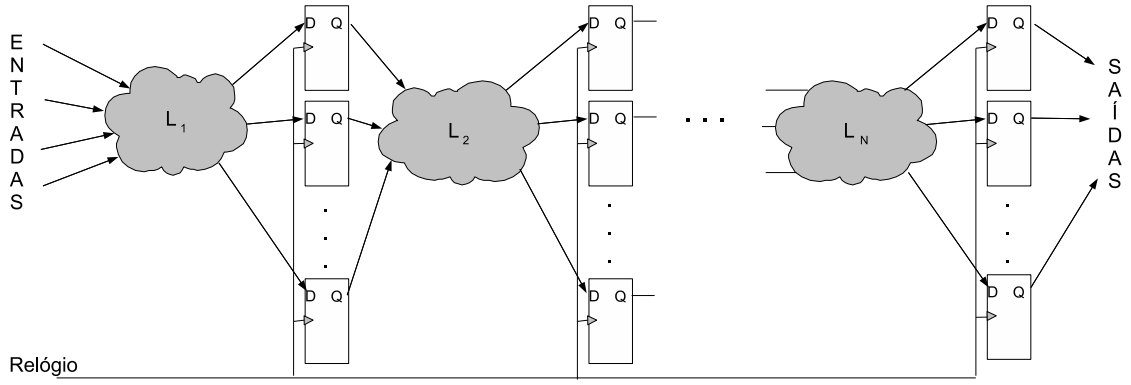


Figura 10: Exemplo de *Pipelining*

A redução do consumo de potência pelo uso da técnica *pipeline* é obtida pela redução de

sinais espúrios (atividade de *glitching*), a partir da redução da profundidade lógica (caminho crítico) dos circuitos. Em circuitos que apresentam uma grande profundidade lógica (como multiplicadores do tipo *array*, por exemplo), a atividade de *glitching* é muito intensa. Desta forma, o uso de registradores em locais estratégicos do circuito pode minimizar a atividade de sinais espúrios, visto que o sinal somente se propaga ao longo do circuito após o comando do sinal de relógio dos registradores.

Outra forma de redução de consumo de potência pela técnica de *pipeline* é pela utilização de células lógicas mais lentas na síntese lógica, caso a frequência seja mantida a mesma de antes da divisão em blocos de *pipeline*.

Neste trabalho o uso de *pipeline* se dá através do uso de barreiras de registradores entre os operadores aritméticos das borboletas na base 2 com decimação no tempo. Maiores detalhes do uso desta técnica nas borboletas serão apresentados no Capítulo 4.

Outra vantagem do uso desta técnica é o compartilhamento de recursos. A cada ciclo de relógio novas entradas podem ser alimentadas no circuito e novas saídas são geradas, ao invés de se esperar por todo o tempo de processamento do circuito (determinado pela soma dos períodos $L_1 + L_2 + \ldots + L_N$, na Figura 10). Contudo, o uso de barreiras de registradores impõe um consumo extra gerado por seus circuitos digitais, fato este, que impede seu uso indiscriminado.

Neste trabalho as barreiras de registradores que formam o *Pipeline* são demonstradas nas figuras pelo uso de linhas horizontais tracejadas sem identificação.

A redução do consumo de potência pelo uso da técnica de *pipeline* também pode ser obtida pela redução da tensão de alimentação. De fato, as transformações empregadas para a redução de tensão de acordo com a frequência de operação dos circuitos são baseadas no aumento do nível de concorrência no sistema, ou seja, mais hardware pode ser utilizado e diversas tarefas podem ser realizadas. Neste caso, as transformações mais típicas são *pipeline* e paralela.

Operações em *pipeline* resultam em penalidade em área, pois requerem hardware adicional. Um aumento em área pode resultar em aumento de capacitância chaveada, com aumento em dissipação de potência. Entretanto, a potência diminui quadraticamente com a tensão de alimentação (Equação 2.14) e aumenta somente linearmente com a capacitância chaveada. Deste modo, esta técnica de redução da tensão de alimentação pode ser empregada com bons resultados, contudo não é explorada neste trabalho.

3.2 Somador *carry save*

O somador *carry save* (CSA - *Carry Save Adder*) (KIM; JAO; TJIANG, 1998) tem a característica de efetuar a soma de três números simultaneamente sem a propagação interna do *carry*. A idéia básica é que os números a serem somados possam ser reduzidos para dois, e então conectados a um circuito somador com propagação de *carry* (RCA - *Ripple Carry Adder*), como mostra o exemplo da Figura 11.

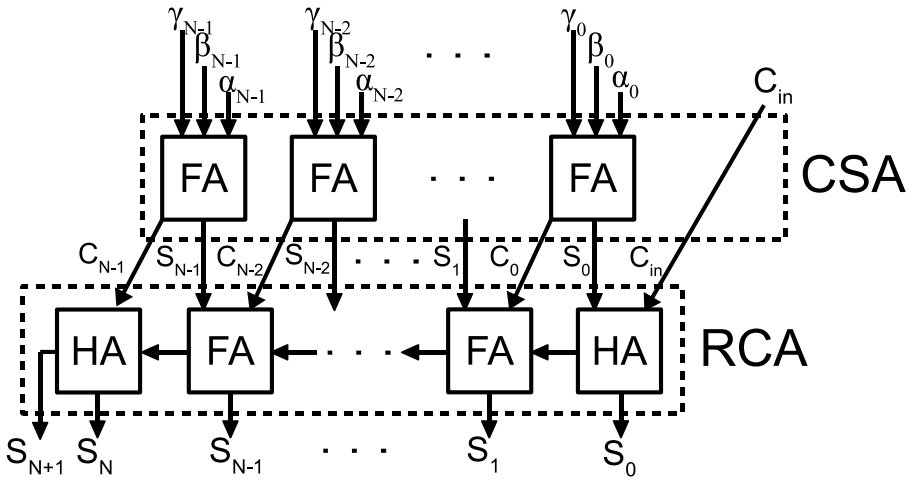


Figura 11: Somador *Carry Save* com *N*-bits.

Como pode ser observado na Figura 11, o *carry* interno é propagado somente na linha final de soma identificada como RCA para a adição dos sinais *C* e *S*.

Para casos em que a soma de mais de três operandos se faz necessária, a replicação da camada CSA pode ser efetuada, preservando uma única estrutura de RCA para o cálculo da soma entre os dois elementos finais, formando uma árvore de conexões. Árvores de conexões podem ser implementadas em diferentes topologias, como por exemplo *Array, Wallace, Overturned* e atraso balanceado (PARHAMI, 2000; MOU; JUTAND, 1992; WALLACE, 1964; ZURAS; MCALLISTER, 1986).

O somador CSA se torna bastante rápido devido ao fato de não propagar o *carry* para um circuito do mesmo nível no caminho crítico, tendo-se a propagação para o nível seguinte, onde o mesmo é computado, como pode ser visto na Figura 11. Devido à sua característica de velocidade de operação, este somador é bastante utilizado pela ferramenta de síntese utilizada neste trabalho [1]. Particularmente, este somador é gerado pela ferramenta de síntese nos circuitos de soma quando são utilizados os operadores '+' e '-' no código de descrição de hardware HDL e também internamente na estrutura do multiplicador *array* utilizado neste trabalho e descrito

[1] Cadence Encounter(R) RTL Compiler v08.10-s108_1, por inspeção do circuito gerado.

na Seção 3.5.1.

3.3 Somadores compressores

As estruturas de somadores compressores são amplamente utilizadas em arquiteturas de multiplicadores para baixo consumo de potência e alto desempenho. A soma e conexão dos operandos são basicamente compostas por portas lógicas ou exclusivo (XOR, *eXclusive-OR*) e multiplexadores (MUX). As estruturas mais frequentes na literatura são os compressores 3:2 [2] (CHANG; GU; ZHANG, 2004), 4:2 (OKLOBDZIJA V. G., 1996), 5:2 (CHANG; GU; ZHANG, 2004), e 7:2 (ROUHOLAMINI M. KAVEHIE; NAVI, 2007). Entretanto, para as estruturas de borboletas exploradas neste trabalho, somente os dois primeiros compressores mencionados são utilizados, pois não são computadas somas simultâneas com mais de quatro operandos. Estas estruturas básicas podem ser usadas como blocos de construção para o desenvolvimento de somadores compressores de ordem maior, como pode ser visto em 3.3.2.

3.3.1 Estrutura básica

Compressores 3:2 apresentam três entradas (α, β e γ) e geram duas saídas (*carry* e *sum*), como mostrado na Figura 12. A estrutura básica deste compressor pode ser empregada como somador completo (FA - *Full Adder*), em que é realizada a soma de dois números de um bit (α e β) com um bit de *carry* de entrada (γ) e produzindo uma saída com o resultado da soma (*sum*) e outra com o *carry* de saída, em que o *carry* de entrada é proveniente do bloco compressor prévio (*carry* de saída).

A estrutura do compressor 3:2 é otimizada, visto que utiliza apenas um MUX para a geração do *carry* e duas portas lógicas XOR para a geração do resultado de soma, como presente em (ROUHOLAMINI M. KAVEHIE; NAVI, 2007). Visando a redução do consumo de potência, os circuitos do compressor 3:2 (portas XOR e multiplexador) podem ser implementados com lógica de passagem, para a redução da atividade de chaveamento, bem como a redução da potência de curto-circuito. Como pode ser visto na Figura 12, o caminho crítico do compressor 3:2 é dado pelas duas portas lógicas XOR, que são utilizadas na obtenção do resultado de soma (*sum*).

A estrutura do somador compressor 4:2 é apresentada no Figura 13 com cinco entradas α, β, γ, δ e C_{in}, e três saídas C_{out}, *carry* e *sum*. O nome 4:2 se refere às entradas e saídas principais

[2] Lê-se três para dois.

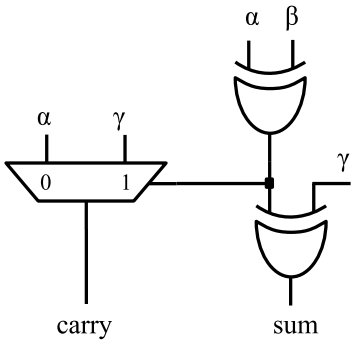


Figura 12: Somador Compressor básico 3:2

do circuito, sendo C_{in} e C_{out} usados na propagação de resultados intermediários na construção do somador de N-bits.

O circuito 4:2 tem como maior caminho crítico três portas lógicas XOR, das entradas α, β, γ e δ à saída *sum*. Um ponto importante a ser considerado deste circuito é a independência da saída de *carry* (C_{out}) em relação à entrada de *carry* (C_{in}). Este aspecto habilita a utilização deste circuito com alto desempenho.

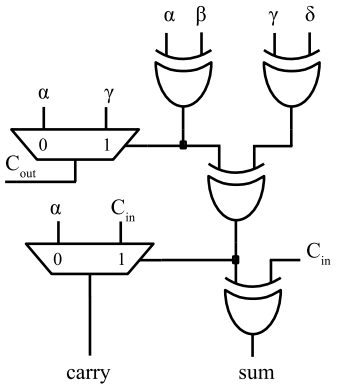


Figura 13: Somador Compressor básico 4:2

3.3.2 Somador compressor de N-bits

Para construir um somador compressor 3:2 ou 4:2 de N-bits é necessária uma recombinação dos resultados parciais de *carry*. Para fazer este agrupamento, uma linha de circuitos meio-somadores e somadores completos são conectados na topologia RCA (Somador com propagação do *carry*, - *Ripple Carry Adder*) como apresentado na Figura 14, em que α, β, γ e δ são os diferentes números com N-bits de entrada a serem somados, para um compressor 4:2, e S (de $S_{0\cdots N+2}$) o resultado da soma.

Como pode ser analisado na Figura 14, a saída *carry* (C_{out}) de um bloco compressor 4:2

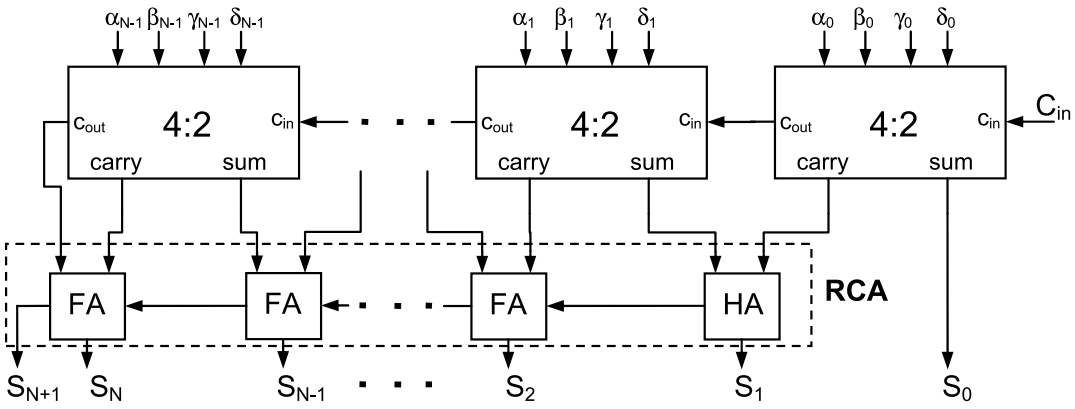


Figura 14: Somador Compressor 4:2 com N-bits

é conectada ao *carry* de entrada (C_{in}) do bloco adjacente. A linha de meio somador (HA) e somadores completos (FA) (representado pelo circuito interno as linhas pontilhadas da Figura 14) representa o gargalo no desempenho de um somador compressor, pois os *carries* são propagados por todos os blocos de soma. Desta forma, deve-se utilizar uma estrutura de soma mais eficiente para esta linha de somadores.

3.4 Sistema Numérico por Resíduos

Com a crescente utilização de operadores aritméticos nas mais diversas áreas, como por exemplo, nas borboletas da FFT na base 2 com decimação no tempo explorada neste trabalho, torna-se cada vez mais necessário o estudo de métodos para tornar os circuitos aritméticos mais rápidos. O presente trabalho tem como objetivo o estudo de operadores aritméticos usando a codificação RNS (*Residue Number System*), para comparação com os operadores CSA e somadores compressores.

A codificação RNS consiste no uso dos resíduos dos números para os cálculos das operações aritméticas. Como a codificação RNS divide o número em partes e processa cada parte separadamente, ela fornece uma grande paralelização do sistema. No caso de operações de soma, este aspecto contribui de forma significativa para a redução da propagação do *carry*, o que ocasiona um aumento no desempenho desta operação.

No sistema numérico por resíduos um número T qualquer é representado por seus resíduos, através de uma base RNS B_{RNS} ($B_{RNS} = (b_1, b_2, \cdots, b_N)$), sendo N um número qualquer de elementos da base, desde que esse número T seja menor que o mínimo múltiplo comum de B_{RNS}. A representação em RNS pode ser calculada pela Equação 3.1, em que a operação *mod* representa o resto da divisão dos números.

$$t_j = T \, mod \, b_j, \; \forall \; 0 \leqslant j \leqslant N \tag{3.1}$$

Na Equação 3.1 se observa a conversão de um número T para a codificação RNS, gerando um conjunto de números t com N unidades (SZABÓ; TANAKA, 1967).

A vantagem deste método é a possível paralelização de operações aritméticas, pois o sistema RNS não é um sistema posicional. Assim, tendo-se dois números quaisquer T e U e suas respectivas representações RNS em uma base B_{RNS}, $(t_1, t_2, \cdots, t_N)_{B_{RNS}}$ e $(u_1, u_2, \cdots, u_N)_{B_{RNS}}$, as operações simples de adição, multiplicação e subtração podem ser realizadas em paralelo, conforme a Equação 3.2, sendo a, m e s os respectivos resultados.

$$\begin{aligned} a &= t_j + u_j \, mod \, b_j \\ m &= t_j * u_j \, mod \, b_j \\ s &= t_j - u_j \, mod \, b_j \end{aligned} \tag{3.2}$$

Como no exemplo da Equação 3.2, cada resíduo pode ser processado separadamente e, dessa forma, a execução de uma operação em um número de N-bits pode se transformar em três operações de um número formado por menos bits, como mostra a Tabela 1, o que pode possibilitar melhor desempenho em alguns casos[3], como apresentado em (OMONDI; PREMKUMAR, 2007).

Tabela 1: Alteração do número de *bits* de acordo com a codificação (SHIAVON, 2010)

Base	**8 bits**	**16 bits**	**32 bits**
$(2^n-1, 2^n, 2^n+1)$	$(2^3-1, 2^3, 2^3+1)$	$(2^6-1, 2^6, 2^6+1)$	$(2^{11}-1, 2^{11}, 2^{11}+1)$
$(2^{n-1}-1, 2^n-1, 2^n)$	$(2^{4-1}-1, 2^4-1, 2^4)$	$(2^{6-1}-1, 2^6-1, 2^6)$	$(2^{12-1}-1, 2^{12}-1, 2^{12})$
$(2^n-1, 2^n+1, 2^{2n}+1)$	$(2^3-1, 2^3+1, 2^6+1)$	$(2^5-1, 2^5+1, 2^{10}+1)$	$(2^9-1, 2^9+1, 2^{18}+1)$

De acordo com a Tabela 1 em um somador de 8 *bits* utilizando a base $(2^n-1, 2^n, 2^n+1)$ o número de bits das somas modulares cai para 3. Também é observado que em uma soma de 32 *bits* a redução no número de bits é ainda maior, de 32 *bits* para 11 *bits*.

Nos últimos anos a utilização da codificação RNS vem apresentando um crescimento significativo, sendo utilizada em várias áreas (PARHAMI, 2000), tais como filtros digitais, que utilizam um número significativo de circuitos somadores e multiplicadores. A utilização deste sistema nas diversas áreas da computação é limitada pelas etapas de codificação (conversão binário para RNS) e decodificação (conversão RNS para binário), que embora venham avançando, ainda representam o gargalo desta operação. A maior complexidade deste sistema é a etapa de

[3] Analisando-se a tríade velocidade de operação, consumo de potência e área.

decodificação (BARANIECKA; JULLIEN, 1978), devido às multiplicações necessárias, tendo um consumo de potência elevado. Uma ilustração dos elementos envolvidos em uma operação em RNS é mostrada na Figura 15.

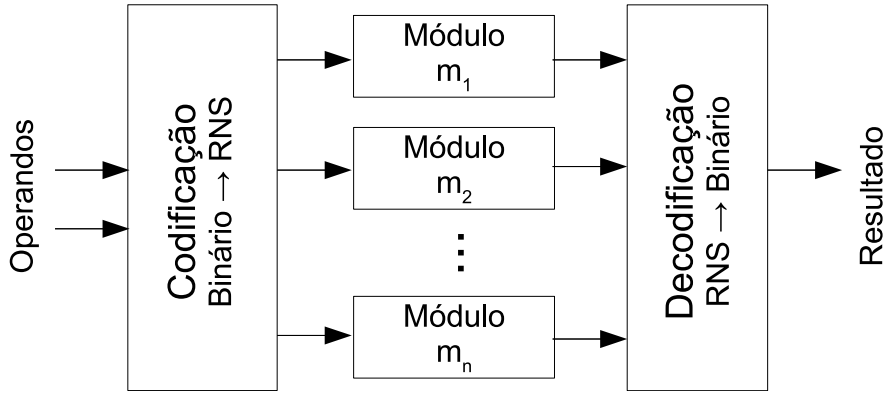


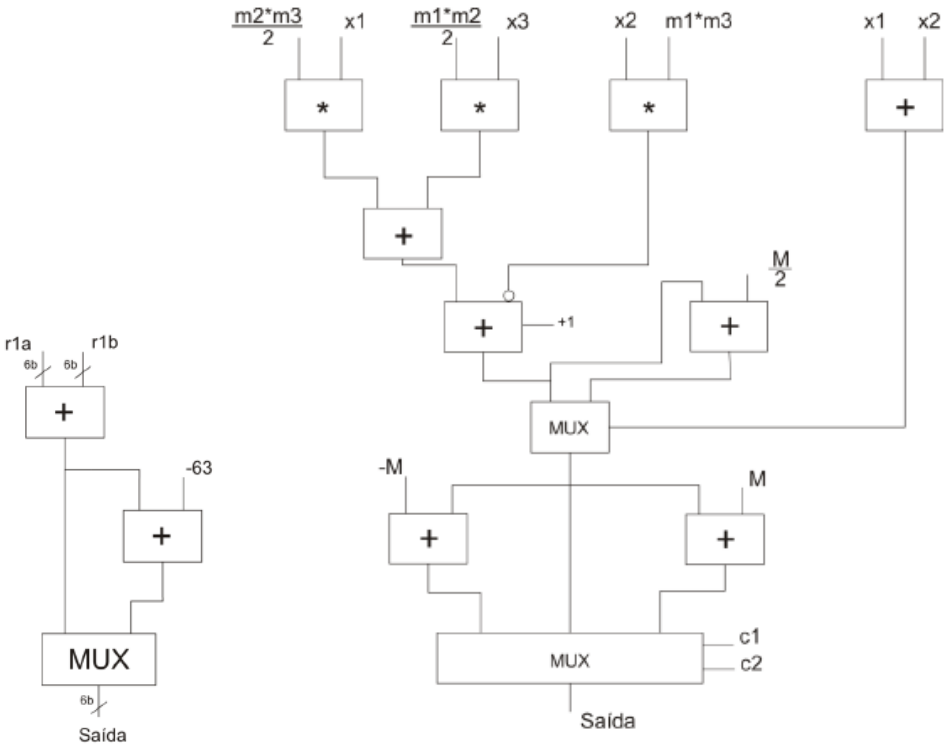
Figura 15: Elementos utilizados em uma operação RNS

Na Figura 15 os operandos de entrada (em binário) passam por uma codificação, que é um circuito composto por circuitos somadores com a finalidade de extrair o resto do número quando dividido por determinada base em todas as posições j da mesma: $T_j = T \, mod \, b_j$. Após essa etapa, o número já é representado por seus resíduos (em RNS) e pode ser processado paralelamente por operações ariméticas simples, conforme a equação 3.2. Cada módulo executa a operação desejada entre os operandos (soma, multiplicação, etc.), fornecendo na saída de cada uma das operações modulares em paralelo um resíduo que compõe a saída do circuito. Nesse ponto, o valor da operação já está realizada, porém representada por seus resíduos. Dessa forma, necessita-se executar a etapa de decodificação para obter o valor da saída em binário. Os circuitos que realizam a etapa de decodificação são construídos usando circuitos somadores e multiplicadores, e muito comumente são baseados no Teorema do Resto Chinês (THUN, 1986). Circuitos com exemplos das diferentes complexidades das etapas de codificção e decodifição são apresentados na Figura 16.

Na Figura 16 os circuitos de codificação (Figura 16a) e decodificação (Figura 16b) são ilustrados e pode-se comparar a diferença de complexidade entre eles, sendo o circuito de decodificação RNS mais complexo por utilizar circuitos somadores, multiplexadores e multiplicadores não presentes na codificação.

3.4.1 Base RNS utilizada

A base é formada por números inteiros primos entre si (o máximo divisor comum é o número 1). Dessa forma, garante-se a unicidade da representação RNS em uma certa base de um número inteiro qualquer, ou seja, é impossível haver duas representações RNS diferentes



(a) Codificação (b) Decodificação

Figura 16: Exemplos de complexidade das etapas de Codificação e Decodificação do RNS (SHIAVON, 2010)

para um mesmo número. A abrangência da representação é dada multiplicando-se todos os módulos que compõem a base. A Equação 3.3 representa o cálculo da abrangência, em que N é um número inteiro qualquer que nessa equação define o número de elementos da base.

$$B_{RNS} = b_{N-1} \cdot b_{N-2} \cdot \ldots \cdot b_0 \tag{3.3}$$

A base, também chamada de *moduli*, utilizada nesse trabalho é $(2^N - 1, 2^N, 2^N + 1)$, sendo N um número qualquer, uma das principais bases em estudo atualmente (BI; JONES, 1988). A partir dessa base foram criadas as etapas de codificação e decodificação, além das etapas modulares.

Como o produto $(2^N - 1) \cdot (2^N) \cdot (2^N + 1)$ precisa ser maior que o maior valor a ser representado nas operações da borboleta para não haver repetições de valores, foi escolhido um valor de $N = 6$.

3.5 Multiplicadores digitais

Entre os operadores aritméticos, os módulos multiplicadores são os mais comuns em operações DSP. Em particular, os multiplicadores paralelos são os que realizam as operações de multiplicação com a maior velocidade.

A operação de multiplicação paralela é realizada a partir de somas sequenciais de produtos parciais. Os produtos parciais são gerados por células de multiplicação intermediárias e o *carry* é propagado para cada grupo.

Circuitos Multiplicadores Digitais são circuitos basicamente compostos por deslocamentos e somas. A Figura 17 ilustra uma multiplicação entre dois operandos binários.

A Figura 17 ilustra uma multiplicação entre dois operandos binários.

$$
\begin{array}{cccccccc}
 & & & 1 & 0 & 1 & 1 & (11) \\
 & & \times & 1 & 0 & 0 & 1 & (\,9) \\
\hline
 & & & 1 & 0 & 1 & 1 & \\
 & & 0 & 0 & 0 & 0 & & \\
 & 0 & 0 & 0 & 0 & & & \\
+ & 1 & 0 & 1 & 1 & & & \\
\hline
0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \quad (99)
\end{array}
$$

Figura 17: Exemplo de multiplicação entre dois números binários

Na Figura 17 dois números binários de 4 *bits* (11 e 9 em decimal) são multiplicados. Para cada valor '1' do multiplicador o primeiro operando é replicado ($1 \times 1 = 1$), e para cada valor '0' uma linha de zeros é acrescentada ($0 \times 1 = 0$) através da função lógica E, implementada com uma porta lógica *AND*. Para cada nova ordem há um deslocamento à esquerda da replicação. Feito isto a soma dos resultados intermediários é necessária, com propagação do *carry*. O resultado é um número com o dobro de *bits*, para uma representação sem perdas ou arredondamentos.

Com o exemplo da Figura 17 percebe-se que para operadores com maior número de *bits* maior é a complexidade do somador final, devido ao número de operandos e propagação do *carry*. Por exemplo para números com 16 *bits* uma soma de 16 operandos precisa ser efetuada, se utilizada esta estrutura básica, o que se torna dispendioso devido ao aumento do atraso, da área necessária e da potência consumida.

Para reduzir a soma entre todos os resultados intermediários, diferentes esquemas de configuração do circuito multiplicador são apresentados na literatura e podem ser encontrados em (COSTA, 2002; OLIVEIRA, 2005; PIEPER, 2008).

3.5.1 Multiplicadores *array*

Um multiplicador tipo *array* binário traduz o exemplo da Figura 17 diretamente para hardware, para $W = 4$ bits. Tem-se as W linhas de produtos parciais, cada uma composta de produtos de níveis de W bits, que pode ser arranjada em uma simples e regular estrutura do tipo *array*. Cada produto de bit é simplesmente uma porta lógica *AND*.

No multiplicador *array* a operação é realizada pela soma dos produtos parciais em sequência. Os produtos parciais são obtidos com portas lógicas *AND* a partir das operações 1 x W. Os W bits menos significativos dos produtos são produzidos no lado direito da estrutura *array*. A Figura 18 exemplifica este processo, onde o circuito de uma multiplicação de 4 bits é ilustrado.

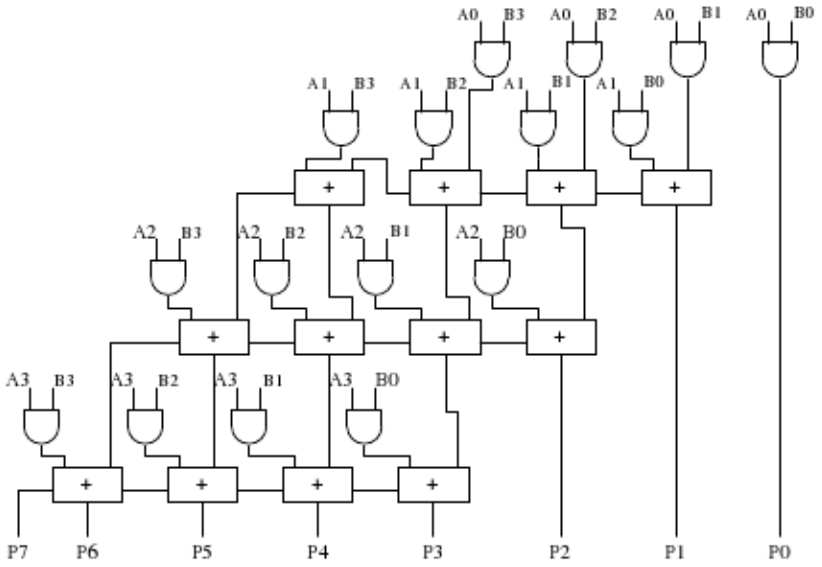


Figura 18: Exemplo de multiplicador Array binário de 4 bits (COSTA, 2002)

Uma grande quantidade de trabalhos de pesquisa visando o projeto de eficientes e regulares arquiteturas de circuitos multiplicadores tem sido desenvolvida, devido a sua grande complexidade e grande utilização. Esquemas de multiplicação, tais como *bit-section, Baugh-Wooley, Hwang* (HWANG, 1979) propõem a implementação de arquiteturas de multiplicadores em complemento de 2. Entretanto, estes tipos de estrutura não apresentam implementações eficientes, devido à forma irregular do tipo árvore-*array* utilizada. Além disso, estas estruturas realizam operações de multiplicação bit a bit, que apresentam uma elevada profundidade lógica, visto que neste tipo de multiplicador há várias linhas de produtos parciais ($W - 1$).

O aspecto da profundidade lógica é um dos principais aspectos de consumo de potência em circuitos multiplicadores, visto que quanto maior a profundidade lógica, maior a propagação de sinais espúrios na estrutura interna do circuito multiplicador.

Módulos multiplicadores com operando com sinal são comuns em muitas aplicações DSP. Neste contexto, os multiplicadores Wallace (WALLACE, 1964) estão entre os mais rápidos. Entretanto, estes multiplicadores não apresentam boa regularidade e não são explorados neste trabalho. Por outro lado, o multiplicador Booth (CHERKAUER; FRIEDMAN, 1996; SEIDEL; MCFEARIN; MATULA, 2001) apresenta um bom aspecto de regularidade, além da redução das linhas de produtos parciais (utiliza-se aproximadamente metade dos produtos parciais).

Apesar do algoritmo de Booth proporcionar uma maior simplicidade para a a implementação da sua arquitetura, torna-se difícil projetar arquiteturas para operar em bases maiores do que 4 (maior impacto na redução de linhas de produtos parciais), devido à complexidade em pré-computar, no termo multiplicador, um crescente número de múltiplos do termo multiplicando.

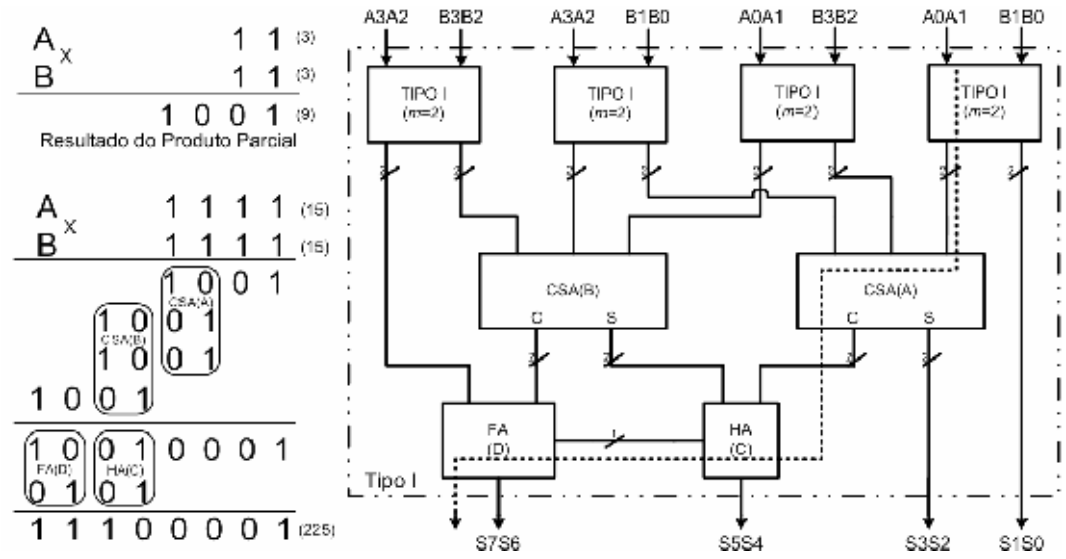
Em (COSTA, 2002) foi proposta uma arquitetura de multiplicador *array* operando na base 2^m, cujos objetivos foram alcançar melhores desempenhos e menores consumos de potência a partir da redução dos termos dos produtos parciais, mantendo-se a mesma regularidade de um multiplicador *array*. Foi mostrado que as arquiteturas propostas podem ser mais naturalmente estendidas à operações de multiplicação em bases maiores utilizando menos níveis lógicos e, desta forma, apresentando menos transições espúrias.

Em (PIEPER, 2008) foi realizada uma otimização dos multiplicadores *array* propostos por (COSTA, 2002)) utilizando circuitos somadores eficientes como o somador CSA nos módulos de multiplicação dedicados. Desta forma, esses módulos de multiplicação foram otimizados e os multiplicadores *array* podem operar em bases maiores (16, 64, 256, etc...) com alto desempenho e reduzido consumo de potência. Este multiplicador é utilizado neste trabalho como parte integrante da estrutura interna das borboletas. A seguir são mostrados alguns aspectos deste multiplicador.

3.5.1.1 Multiplicadores *array* na base 2^m

Em um multiplicador *array* a quantidade de linhas de soma é determinada pela razão $(\frac{N_b}{m}) - 1$, sendo N_b o número de *bits* utilizados nos operandos multiplicados e m o número que divide os operandos em grupos de $m\,bits$. O que torna 2^m uma base binária e cada um destes grupos pode ser visto como uma representação de um dígito nesta base, e tem a vantagem de multiplicar mais de 1 *bit* no mesmo instante, de acordo com o valor de m do multiplicador, com isto conseguimos a redução da profundidade lógica (ou linhas de soma). Na Figura 19 um exemplo de um cálculo de multiplicação (a) e o circuito que realiza a operação (b) são ilustrados.

Na Figura 19 os blocos tipo 1 (m = 2) representam uma multiplicação de 4 bits sem sinal, representado na Figura 18.



(a) Cálculo (b) Circuito

Figura 19: Multiplicação composta de blocos $m = 2$ e CSA (PIEPER, 2008)

Na operação de um multiplicador tipo *array* binário, os produtos parciais são realizados em uma forma paralela. A multiplicação de números binários positivos é realizada da mesma forma em que a operação é realizada com números decimais.

O multiplicador *array* é uma técnica de multiplicação digital otimizada para redução de consumo de potência com baixo atraso de cálculo. Esta técnica consegue diminuir a quantidade de elementos a serem somados no resultado intermediário. Multiplicadores *array* diminuem o atraso devido a redução da profundidade lógica e o consumo de potência devido a redução da atividade de *glitching*.

Mais detalhes sobre o multiplicador *array* podem ser encontrados em (PIEPER, 2008).

3.6 Resumo do capítulo

Neste capítulo foram apresentadas as principais técnicas para redução do consumo de potência a serem utilizadas nas borboletas na base 2 com decimação no tempo. Foram apresentados conceitos e características das técnicas *pipeline* e de codificação RNS. Também foram apresentados os principais conceitos dos operadores aritméticos a serem utilizados nas estruturas internas das borboletas. A seguir seram apresentadas as arquiteturas de borboletas que utilizaram os operadores aritméticos descritos neste capítulo.

4 ARQUITETURAS DE BORBOLETAS DET NA BASE 2

Neste capítulo são apresentadas topologias de estruturas com diferentes operadores aritméticos para aplicação em borboletas DET na base 2.

4.1 Estrutura de borboleta com quatro multiplicadores reais

Na Seção 2.2.1 foi apresentada uma topologia de borboleta com quatro multiplicadores reais, três somadores e três subtratatores. Entretanto, a aplicação das camadas de *pipeline* não foi destacada, Figura 5. As camadas de *pipeline* são mostradas na Figura 20, sendo estas destacadas nas linhas pontilhadas.

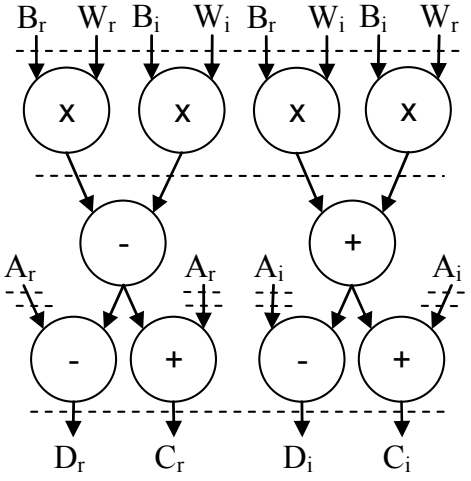


Figura 20: Estrutura A com camadas de *pipeline* destacadas pelas linhas pontilhadas

A Figura 20 ilustra uma borboleta DET na base 2 com quatro multiplicadores, sendo esta a construção mais direta de implementação de um multiplicador complexo (2.1.2). As linhas pontilhadas representam os estágios de *pipeline*. Na verdade, esta figura agrega apenas um estágio de *pipeline*, que é representado pelas linhas pontilhadas nas saídas dos multiplicadores. As barreiras de registradores aplicadas às saídas da borboleta são requisitos da ferramenta de

síntese para a aplicação das restrições de tempo. As barreiras de registradores nas entradas da borboleta têm por objetivo a comparação justa com as outras estruturas, visto que todas estas apresentam duas barreiras de registradores entre as entradas e as saídas.

4.2 Estruturas de borboletas com três multiplicadores reais

Diferentemente do apresentado na estrutura original da borboleta, mostrada na Figura 20, as borboletas podem ser também construídas com três multiplicadores reais, como é mostrado nesta seção.

4.2.1 Borboleta DET na base 2 proposta

Sendo A, B e W as entradas de uma borboleta DET na base 2 da Figura 4 do Capítulo 2 e C e D as saídas da mesma. Pode-se partir das Equações 2.12 e 2.13, alterando-as algebricamente e agrupando os operadores B e W, chegar-se às Equações 4.1, 4.2, 4.3 e 4.4, de modo que o termo entre colchetes da Equação 4.1 é idêntico matematicamente ao primeiro termo em parênteses da Equação 2.12, ou seja, a parte real de $B \cdot W$.

$$C_r = A_r + [(B_r + W_i) \cdot (W_r - B_i) - (W_r \cdot W_i - B_r \cdot B_i)] \tag{4.1}$$

$$C_i = A_i + [(B_i + W_i) \cdot (W_r - B_r) - (W_r \cdot W_i + B_r \cdot B_i)] \tag{4.2}$$

$$D_r = A_r - [(B_r + W_i) \cdot (W_r - B_i) - (W_r \cdot W_i - B_r \cdot B_i)] \tag{4.3}$$

$$D_i = A_i - [(B_i + W_i) \cdot (W_r - B_r) - (W_r \cdot W_i + B_r \cdot B_i)] \tag{4.4}$$

Para as Equações 4.1, 4.2, 4.3 e 4.4, como os termos W_r e W_i são pré-calculados e armazenados em memória ROM, para todas as aplicações quando N é constante (ou variável, se de acordo com (KIM; PARK, 2007) armazenados os valores para N máximo), o termo $W_r \cdot W_i$ também pode ser pré-calculado e armazenado em memória ROM. Portanto, com estas equações economiza-se um multiplicador real para a implementação da borboleta DET na base 2, visto que os outros termos com multiplicação repetem-se, e portanto, podem ser calculados apenas uma vez. A Figura 21 ilustra as operações aritméticas desta borboleta.

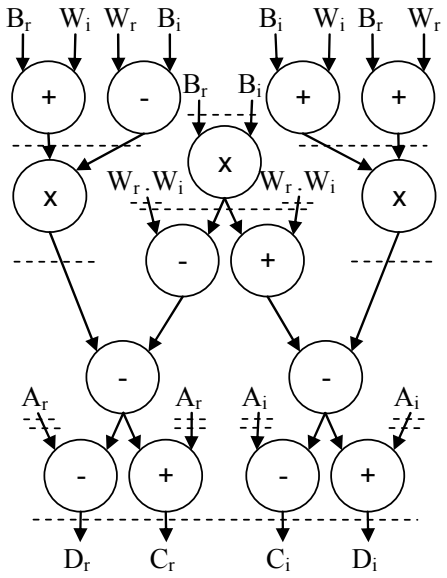


Figura 21: Estrutura proposta de operadores para borboleta DET na base 2 com 3 multiplicadores e 12 somadores

A estrutura da Figura 21 apresenta três multiplicadores reais em seu algoritmo, porém possui doze operações de soma (seis de adição e seis de subtração), seis a mais que a borboleta da Figura 20. Esse compromisso entre a escolha de um multiplicador a menos, em detrimento a alguns somadores a mais. É explorado neste trabalho também com outras estruturas de borboleta na Seção 4.2.2. Esta estrutura de borboleta proposta é referida em algumas partes do texto como estrutura E.

O multiplicador complexo da arquitetura proposta não é encontrado em nenhuma das dezesseis alternativas de multiplicadores complexos de (WENZLER; LUDER, 1995). Isto, devido ao fato do artifício da redução em um multiplicador real na multiplicação complexa ser efetuado através do armazenamento em memória de uma operação das entradas. Fato este não computado na exploração matemática de (WENZLER; LUDER, 1995).

Observa-se o uso de registradores *pipeline* nas saídas dos multiplicadores. Esta barreira de registradores, além de reduzir o caminho crítico (que é dado apenas pelo circuito multiplicador), também tem a função de reduzir a propagação de sinais espúrios (*glitching*) ao longo dos outros operadores aritméticos da borboleta. Os registradores nas saídas da borboleta são utilizados para habilitar o uso das restrições de tempo da ferramenta de síntese.

4.2.2 Outras estruturas de borboleta DET

A multiplicação complexa presente na borboleta DET na base 2 quando calculada pode levar a diferentes soluções contendo diferentes quantidades de multiplicadores reais (Seção

2.1.2). Analisando as técnicas de multiplicação complexa propostas por (WENZLER; LUDER, 1995), percebe-se que a construção em hardware das dezesseis soluções leva a apenas três circuitos distintos. A Figura 22 ilustra essas técnicas de solução da multiplicação complexa aplicadas na borboleta DET na base 2.

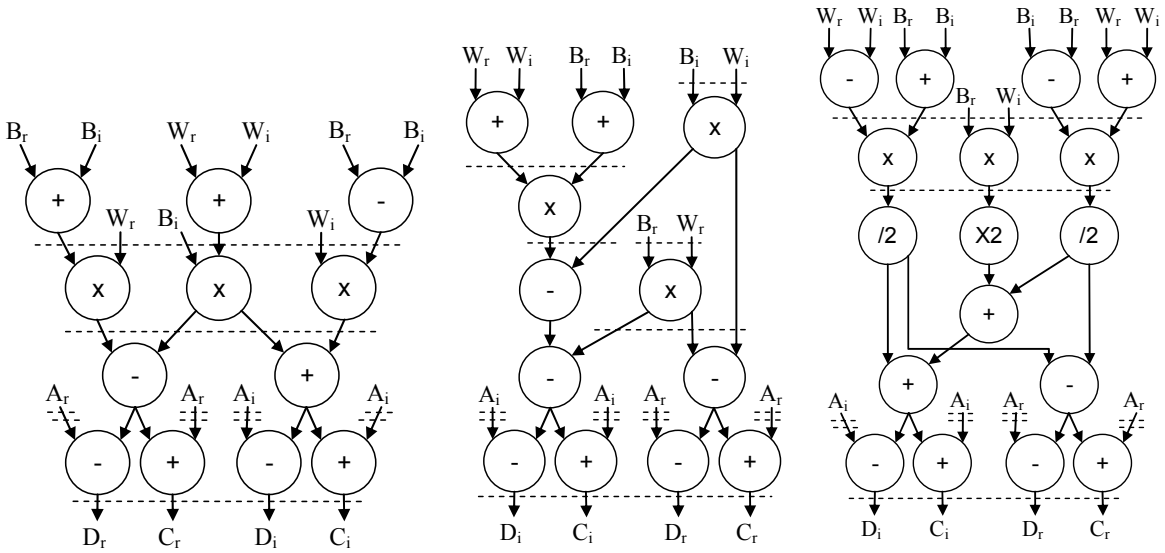


Figura 22: Estruturas de borboleta usando os algoritmos de multiplicação complexa de (WENZLER; LUDER, 1995)

Conforme a Figura 22, a transformação algébrica do algoritmo de multiplicação complexa com quatro multiplicadores reais para três multiplicadores reais, leva a três esquemas diferentes de estrutura de borboleta DET na base 2. A redução do número de multiplicadores reais só é possível pelo aumento do caminho crítico do sinal com uma operação de soma/subtração antes das multiplicações no fluxo do sinal.

Observa-se na Figura 22, as diferentes possibilidades de uso de *pipeline* nas três estruturas apresentadas (destacado pelas linhas pontilhadas) para a redução do caminho crítico. Assim como na estrutura original A e na estrutura proposta E, a barreira de registradores nas saídas das borboletas é utilizada devido ao uso das restrições de tempo da ferramenta de síntese.

4.3 Operadores aritméticos dedicados aplicados nas borboletas

Como pôde ser observado nas cinco topologias de borboleta apresentadas, há um grande número de operadores aritméticos de multiplicação e soma/subtração envolvidos nos cálculos dos operandos. Desta forma, são utilizadas duas metodologias de implementação das borbole-

tas, sendo uma delas com o uso dos operadores aritméticos da ferramenta de síntese e a outra com o uso de multiplicadores aritméticos, sistema RNS e somadores compressores implementados no nível lógico e um somador compressor dedicado implementado no nível de circuito elétrico.

4.3.1 Borboletas com operadores da síntese lógica

As estruturas de borboleta utilizando os operadores da síntese automática do programa Cadence Encounter(R) RTL Compiler v08.10-s108_1 foram codificadas empregando-se o uso direto dos operadores de soma (+), subtração (-) e multiplicação (*) nas borboletas quando descritas em código HDL, sendo as Figuras 20, 22 e 21 as ilustrações das mesmas. Portanto o programa de síntese pode escolher o melhor circuito de operação de seu algoritmo e mapear para a biblioteca satisfazendo os parâmetros de projeto (tempo, potência, área)[1].

Dando liberdade para a ferramenta, ela aplica somadores do tipo CSA (Seção 3.2) para a soma de três ou mais operandos e multiplicadores do tipo *Carry-Save Multiplication* (ZIMMERMANN, 2009) para a computação de $(A+B)*C$, por exemplo (conforme observado por inspeção no circuito gerado).

4.3.2 Borboletas com multiplicadores *array*

As estruturas de multiplicação ilustradas nas Figuras 20, 22 e 21 foram também implementadas utilizando o Multiplicador *array* descrito na Seção 3.5.1. Neste caso, ao invés de deixar por conta da ferramenta de síntese, o operador de multiplicação é inserido no código HDL, cuja descrição é realizada no nível de portas lógicas. Neste caso, o multiplicador *array* utilizado é o proposto em (PIEPER, 2008) por apresentar bons resultados de desempenho e consumo de potência, quando comparado com outros multiplicadores presentes na literatura (multiplicador Booth, por exemplo). Os resultados de síntese lógica podem ser analisados na Seção 5.2.

4.3.3 Borboletas com somadores RNS

Os somadores presentes após os multiplicadores no fluxo do sinal das borboletas foram também implementados utilizando o sistema RNS de acordo com a Seção 3.4. Sua localização pode ser melhor compreendida observando as Figuras 23, 24 e 25.

A Figura 26 ilustra as operações internas de um dos blocos RNS da estrutura B, por exem-

[1] Restrições de tempo e potência foram impostas a ferramenta de síntese, dando liberdade de área.

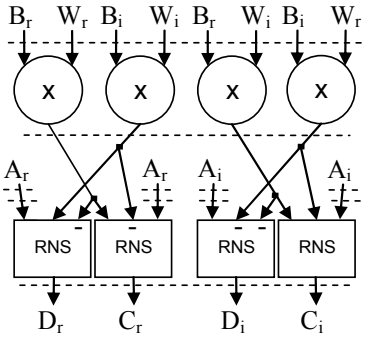
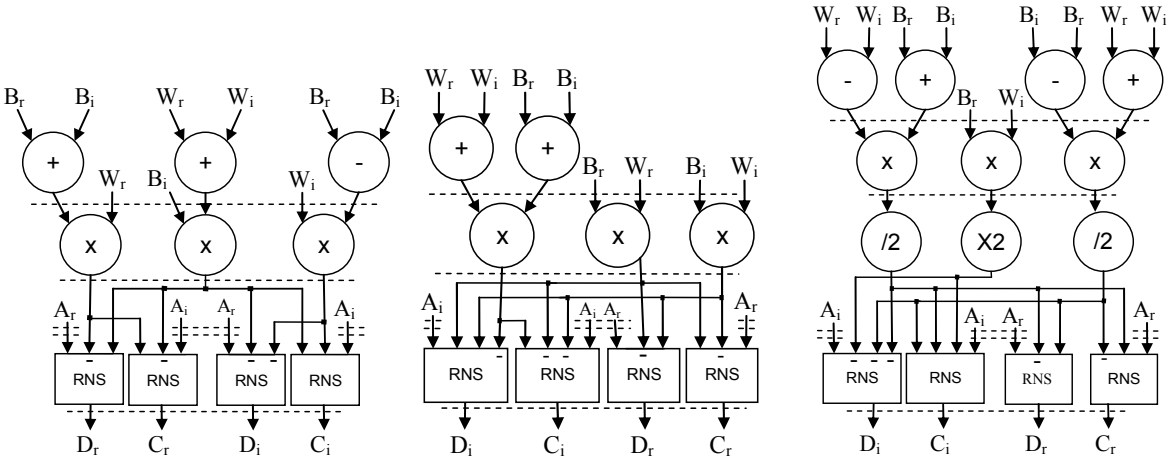


Figura 23: Estrutura A com somadores RNS



(a) - Estrutura B - (b) - Estrutura C - (c) - Estrutura D -

Figura 24: Estruturas de borboleta usando os algorítmos de multiplicação complexa de (WENZLER; LUDER, 1995) com somadores RNS

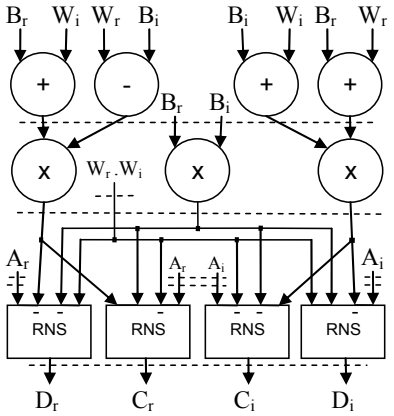


Figura 25: Estrutura proposta com somadores RNS

plo, em que o primeiro bloco RNS à esquerda da Figura 24a é caracterizado, tendo como entradas os sinais provenientes dos blocos multiplicadores e a entrada A_r e como saída o sinal

D_r.

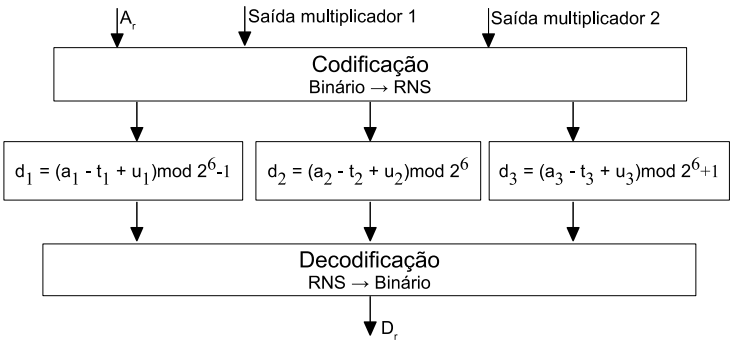


Figura 26: Detalhe da operação interna do RNS

O sistema RNS foi somente utilizado nos somadores situados após os multiplicadores para possibilitar a comparação com as estruturas de borboletas usando os somadores compressores.

4.3.4 Borboletas com somadores compressores 3:2 e 4:2

Circuitos somadores compressores foram inseridos nas estruturas DET na base 2 para avaliação e comparação. As estruturas de borboletas com os somadores compressores podem ser visualizadas nas Figuras 27, 28 e 29.

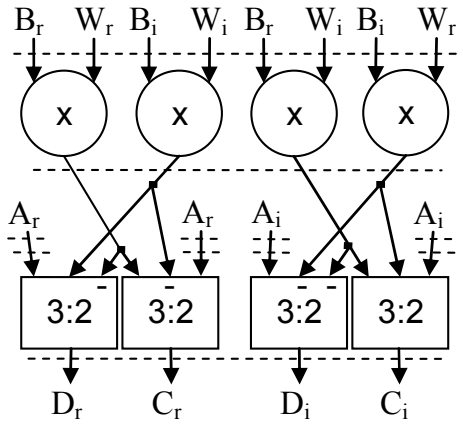
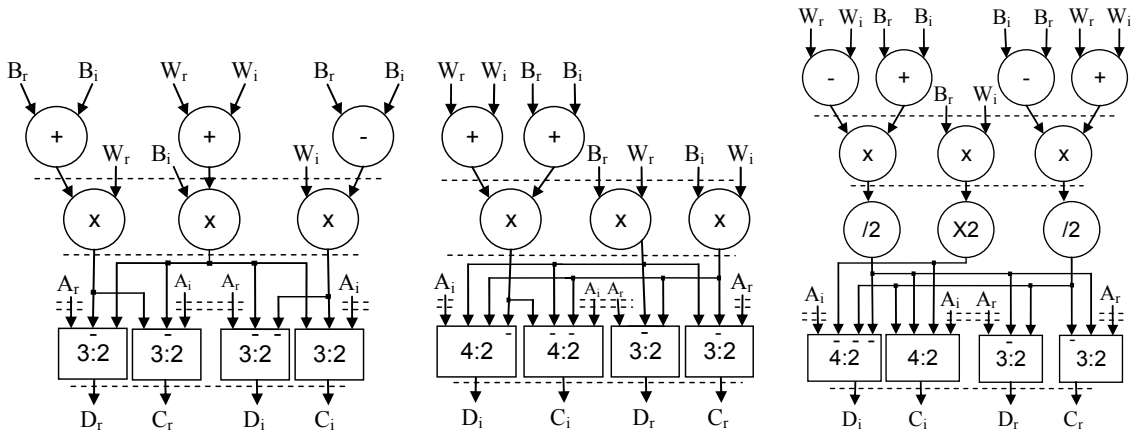


Figura 27: Estrutura A com somadores compressores

Como visto na Seção 3.3, o somador compressor apenas realiza operações de soma, e nas Figuras 28, 27 e 29 os compressores necessitam realizar as subtrações entre sinais para o perfeito funcionamento da estrutura, como pode ser comparado com a Figura 20. Para tanto a operação de subtração é criada com o complemento de 2 do sinal, invertendo o sinal e somando 1. A inversão é feita bit a bit por circuitos inversores CMOS, e a soma com 1 é feita aplicando



(a) - Estrutura B - (b) - Estrutura C - (c) - Estrutura D -

Figura 28: Estruturas de borboleta usando os algoritmos de multiplicação complexa de (WENZLER; LUDER, 1995) com somadores compressores

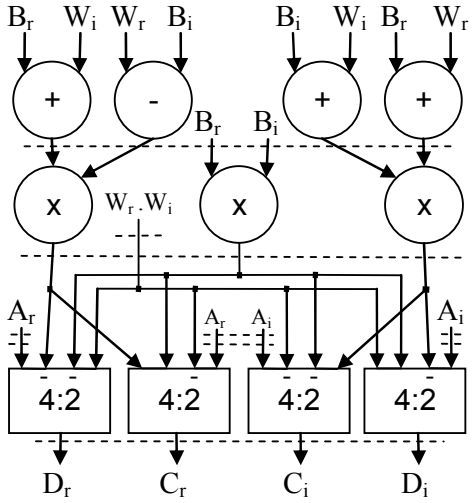


Figura 29: Estrutura proposta com somadores compressores

um *carry* de entrada em somadores compressores. Para a estrutura C com somadores compressores, ilustrada na Figura 28b que possui dois sinais a serem subtraídos, o *carry* de entrada foi realizado com dois bits. Para o caso da estrutura D apresentada na Figura 28c, onde três sinais necessitam de subtração, dois entram no *carry* do compressor 4:2 e é criado um *carry* de entrada no RCA compõe o compressor (Figura 14)transformando o HA do circuito em um FA.

4.3.5 Porta lógica XOR otimizada

Como mostrado no Capítulo 3, os somadores compressores apresentam na sua estrutura interna circuitos multiplexadores e portas XOR, estando estas portas presentes no caminho crí-

tico. Desta forma, neste trabalho é proposta uma topologia de porta XOR otimizada[2], baseada em transistores de passagem, para aplicação nos somadores compressores.

Existem diversos tipos ou famílias de lógicas digitais usando transistores PMOS (*P-Channel Metal Oxide Semiconductor*) e NMOS (*N-Channel Metal Oxide Semiconductor*), sendo a lógica CMOS a mais utilizada na indústria contemporânea por combinar velocidade de operação, pequena área e baixo consumo de energia (RABAEY, 2009). A família CMOS apresenta como característica na construção das portas básicas um reduzido número de transistores. Entretanto, isto não ocorre com a porta lógica XOR, o que é passível de estudo por se tratar de uma porta lógica com uso intenso em circuitos aritméticos.

Famílias de circuitos digitais são descritas e comparadas em (ZIMMERMANN; FICHTNER, 1997), inclusive a Lógica PL (*Pass Gates*, transistores de passagem). A família PL tem como características capacitância e resistência de entrada maiores que a lógica CMOS, o que é capaz de reduzir a potência de *glitching*, pelo efeito de filtro passa-baixas criado por essas características elétricas.

Circuitos CMOS convencionais, em combinação com lógica de passagem, permitem implementações eficientes de portas lógicas básicas e portas complexas, sendo constituídas de poucos transistores e poucas conexões. As desvantagens residem no grande tamanho do transistor PMOS, ocasionando alta capacitância de entrada e área, e na pequena capacidade de drenagem da saída causada por transistores em série.

Uma porta lógica XOR dedicada e otimizada foi criada para aplicação em somadores compressores. A XOR desenvolvida é apresentada na Figura 30, onde as entradas são representadas por α e β, a porta NOR desta Figura é uma porta padrão CMOS com quatro transistores e a saída é dada no ponto com o nome da porta.

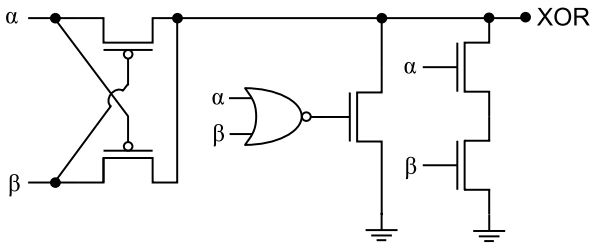


Figura 30: Estrutura proposta de porta lógica XOR

As principais características desta porta são:

- Apresenta saída com valores lógicos fortes para todas as combinações de valores de en-

[2]Não encontrada na literatura (WANG; FANG; FENG, 1994; ZIMMERMANN; FICHTNER, 1997; BUI; ALSHERAIDAH; WANG, 2000; GOEL; ELGAMEL; BAYOUMI, 2003; GOEL et al., 2006)

trada.[3]

- Ausência do curto-circuito provocado pela lógica CMOS no caminho principal.
- A porta NOR CMOS utilizada alimenta apenas um transistor. Logo pode ser projetada com dimensões mínimas.
- Possui alta resistência de entrada, criando o efeito de um filtro passa-baixas, comportamento desejado em somadores compressores por atenuar os resultados transientes (*Glitching*) e, consequentemente, a atividade de chaveamento.
- Não necessita de inversores para drenagem das entradas.

O circuito desta XOR possui elementos em comum com a XOR proposta por (WANG; FANG; FENG, 1994). Contudo, a saída não apresenta valores fortes para todas as combinações de entrada. Outras portas lógicas XOR baseadas em portas de passagem[4] necessitam de valores complementares para as entradas, o que acarreta em aumento de área e consumo de potência devido aos circuitos inversores no caminho principal. A dificuldade em encontrar uma XOR voltada para baixo consumo com valores de saída efetivos foram os fatores mandatórios para a proposição desta topologia.

4.3.6 Somadores compressores com porta XOR otimizada

A porta XOR otimizada apresentada na Figura 30 foi inserida aos somadores compressores. Entretanto, esta nova porta XOR não foi inserida na linha do somador RCA.

A porta lógica XOR apresentada na Seção 4.3.5 foi comparada com uma porta lógica CMOS padrão aplicada nos somadores compressores das Figuras 12 e 13. A implementação do somador RCA do compressor foi realizada com portas CMOS. Isto devido ao fato de que no somador RCA, o 'gargalo' de tempo imposto pela propagação dos *carries* exige uma lógica voltada para desempenho em frequência, sendo usado portanto lógica CMOS na construção das mesmas.

A simulação foi realizada em um ambiente de sinais digitais e analógicos mistos (AMS - *Analog/Mixed Signal*), sendo as entradas dos somadores geradas aleatoriamente em um intervavo de tempo de 10ns, caracterizando um ciclo de relógio de 100MHz, e o restante do circuito é analógico utilizando modelos esquemáticos de transistores da Cadence Virtuoso(R) 6.1.4 para 180nm, a qual foi a ferramenta de simulação.

[3] Transistores MOS do tipo P conduzindo valor lógico '1' e do tipo N conduzindo valor lógico '0'.

[4] Podem ser encontradas em (BUI; AL-SHERAIDAH; WANG, 2000; ZIMMERMANN; FICHTNER, 1997).

Os circuitos somadores compressores foram simulados por 500ns [5] e a Tabela 2 apresenta os resultados de consumo de potência para os somadores 3:2 e 4:2 com circuitos implementados a partir das portas lógicas XOR CMOS (ZIMMERMANN; FICHTNER, 1997) e a otimizada da Seção 4.3.5.

Tabela 2: Resultados de Potência (μW) para os somadores compressores 3:2 e 4:2 simulados no nível elétrico, com porta XOR CMOS e a porta XOR otimizada

Compressor	**3:2**	**4:2**
XOR CMOS	668,1	771,0
XOR Otimizada	641,3	805,4

Como pode ser observado na Tabela 2, o consumo de potência dos somadores compressores 3:2 com a porta XOR otimizada é menor do que o consumo de potência do compressor com o uso da porta XOR CMOS. Entretanto, este mesmo cenário não se mantém em circuitos compressores 4:2 Isto ocorre pois a relação da duração da corrente pelo fator da atividade de *glitching* é maior no compressor 4:2, como pode ser observado na Figura 31. Nos gráficos desta figura, observa-se que a corrente do circuito CMOS (em azul e linhas contínuas) possui picos mais altos que a corrente do circuito com a XOR otimizada (em vermelho e linhas pontilhadas). Contudo, o valor médio da corrente durante a transição é maior para a XOR otimizada. Também se observa nesse mesmo gráfico, que o sinal de saída apresentado do circuito com a XOR otimizada (linha pontilhada) apresenta menos espúrios (*glitches*) e é mais lento que o sinal de saída do somador compressor implementado com a XOR CMOS.

A Figura 32, obtida de ilustra os comportamentos das tensões em uma porta de saída do circuito somador compressor 4:2 com a XOR otimizada (em vermelho e linha pontilhada) e com a XOR CMOS. Também ilustra as correntes nos dois circuitos, sendo a linha pontilhada a corrente no circuito somador compressor com a XOR otimizada.

Na Figura 32 o gráfico da tensão do somador compressor com a XOR otimizada apresenta menos sinais espúrios (*glitches*) e é mais lento que o sinal de saída do somador compressor implementado com a XOR CMOS, e a corrente do circuito com a XOR CMOS tem um pico maior, mas a sua duração é menor que a corrente do circuito utilizando a XOR otimizada.

4.4 Resumo do capítulo

Este capítulo apresentou cinco diferentes estruturas de borboletas DET com várias topologias de operadores aritméticos (Operadores de síntese, CSA, Somadores Compressores, so-

[5] Aplicação de 50 vetores de entrada.

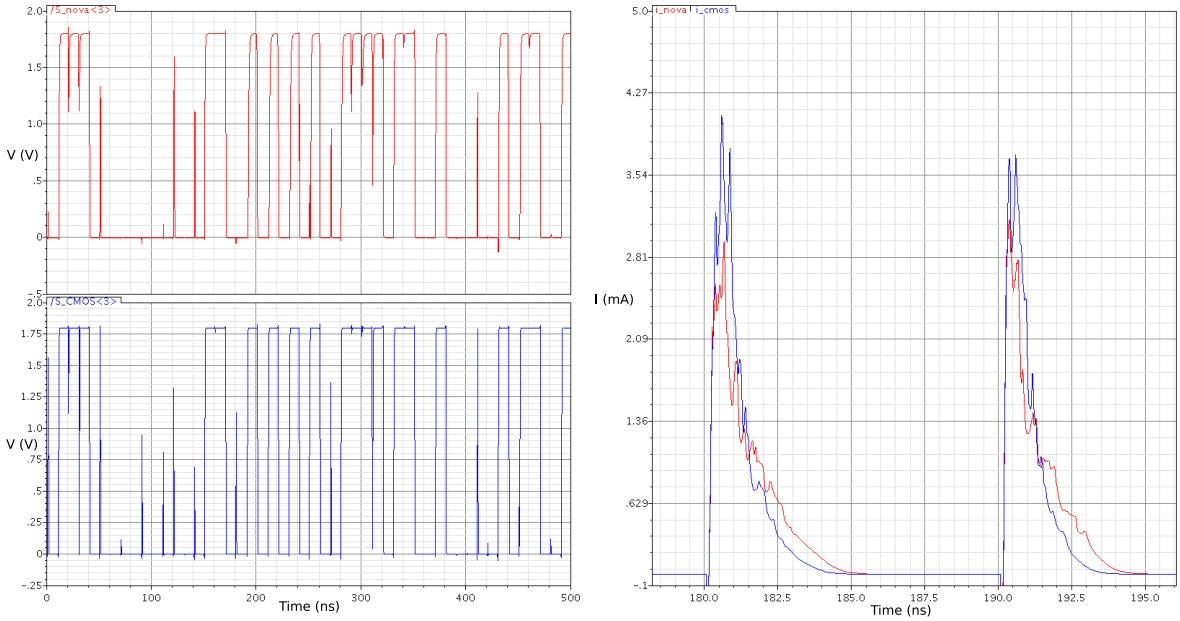


Figura 31: Formas de onda para o compressor 3:2, sinais em linha contínua são provenientes do circuito composto por XOR CMOS e em linha pontilhada pela XOR otimizada

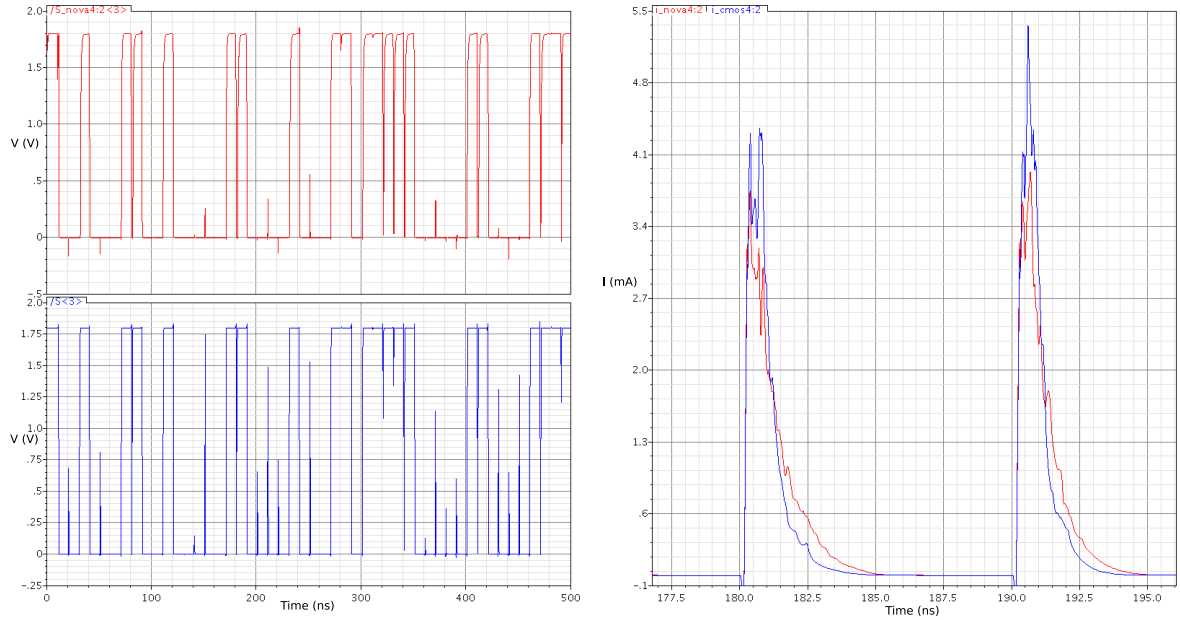


Figura 32: Formas de onda para o compressor 4:2, sinais em linha contínua são provenientes do circuito composto por XOR CMOS e em linha pontilhada pela XOR otimizada

madores com a codificação RNS e Multiplicador *Array*), as quais apresentam resultados no Capítulo 5 (seguinte). Este capítulo também apresentou a topologia de uma nova porta XOR. Foram apresentados resultados de consumo de potência com a utilização da porta XOR proposta nos somadores compressores 3:2 e 4:2. No próximo capítulo, são abordados os resultados

obtidos com as borboletas utilizando os diferentes operadores.

5 RESULTADOS OBTIDOS

Neste capítulo são apresentados os resultados obtidos em simulações elétricas e relatórios de sínteses lógicas para os circuitos descritos nos capítulos anteriores (Capítulos 3 e 4). Para a obtenção dos resultados foram realizadas sínteses lógicas com a tecnologia de 180nm e extraídos os resultados de área, contagem de células e consumo de potência. A potência foi analisada com atividade de chaveamento para 10.000 vetores aleatórios de entrada, quando utilizado o programa de síntese lógica da Cadence Encounter(R) RTL Compiler v08.10-s108_1. Foram utilizadas frequências de operação de 100MHz para a maioria dos circuitos. Em alguns casos, quando o circuito em análise não conseguiu convergência de operação para 100MHz, a frequência de operação utilizada foi de 20MHz.

O valor de 10.000 vetores foi determinado verificando-se em simulação que com esta quantidade se sinais de entrada era possível cobrir a maioria dos estados digitais de funcionamento dos circuitos, sendo os valores obtidos para menos menos pontos com um erro considerável, e com mais pontos sem alteração significativa, em relação ao esforço de simulação dispendido.

5.1 Resultados das Borboletas Usando os Operadores de síntese

Nesta seção são apresentados os resultados das arquiteturas de borboleta na base 2 DET, onde estas foram implementadas dando liberdade aos algoritmos da ferramenta de síntese, ou seja, os operadores aritméticos utilizados são os da própria ferramenta de síntese, conforme Seção 4.3.1. Em (ZIMMERMANN, 2009), são abordados alguns detalhes de como uma síntese lógica opera e como seus operadores aritméticos são selecionados.

Todas as estruturas de borboletas foram escritas em linguagem de descrição de hardware Verilog. A síntese lógica dos circuitos implementados foi realizada em tecnologia 180nm com a biblioteca da X-FAB (X-FAB, 2008) com frequências de operação de 100MHz e 20MHz. Os resultados de síntese lógica com 100MHz podem ser vistos na Tabela 3. Nesta Tabela são apresentados resultados das borboletas utilizando uma e duas camadas de *pipeline*.

Tabela 3: Valores de Potência (mW), Área (μm^2) e Contagem de células para as borboletas na base 2 DET em 180nm a 100MHz com operadores de síntese da ferramenta

	Potência		**Área**		**Células**	
Estrutura	1 ‖	2 ‖	1 ‖	2 ‖	1 ‖	2 ‖
A	6,22	8,12	44.314	49.529	1.944	1.988
B	11,50	8,83	42.036	46.519	2.031	1.827
C	7,81	8,85	43.103	46.823	1.934	1.777
D	10,31	8,68	45.157	48.050	2.211	1.856
E	10,57	10,11	48.966	51.819	2.227	1.889

‖ Número de camadas de *Pipeline*

Na Tabela 3 a estrutura E se refere à estrutura proposta neste trabalho (Seção 4.2.1). Observa-se que o menor consumo de potência é apresentado pela estrutura A, que utiliza apenas uma camada de *pipeline*, pois a segunda camada de registradores só efetua um atraso no circuito, visto que é aplicada às entradas, não trazendo nenhum benefício ao circuito. Entretanto, esta segunda camada de registradores tem o seu uso justificado para a comparação com as demais estruturas que utilizam duas barreiras de *pipeline* no circuito. Logo, conclui-se que o aumento de aproximadamente 2mW da estrutura A, comparando-se uma camada com duas camadas de *pipeline*, deve-se ao consumo extra gerado pelos registradores.

Nas estruturas B, D e E da Tabela 3, o consumo de potência é reduzido, quando são utilizadas duas barreiras de *pipeline*, o que leva à conclusão de que mesmo com os 2mW adicionais de consumo, devido à segunda camada de *pipeline*, as menores complexidades[1] das estruturas conseguiram reduzir o consumo de potência. O mesmo não se aplica à estrutura C, que pelo fato de não apresentar o mesmo grau de redução de complexidade em relação às estruturas B, D e E, teve um aumento no seu consumo de potência, como pode ser observado nessa mesma Tabela.

Observou-se que a redução de consumo de potência nas estruturas se deve ao fato de que diferentes células lógicas são utilizadas no mapeamento do circuito, e que células mais lentas não consomem potência tanto quanto células lógicas rápidas. Algumas estruturas estão aptas ao maior uso destas células mais lentas. Outro fator da redução do consumo de potência é devido à redução da atividade de chaveamento nos nós do circuito, causados pela divisão do circuito imposta pelas barreiras de registradores do *pipeline*. Também constata-se que o número de células não é fator determinante para o consumo de potência, visto que não é a quantidade, mas sim quais células estão sendo utilizadas.

Para fins de comparação com as estruturas de borboletas que não atingiram as restrições de tempo para 100MHz quando mapeados usando os operadores aritméticos *multiplicador Array*

[1] Referente a 1ª linha de circuitos somadores e aos multiplicadores.

e *somador RNS*, as estruturas utilizando os operadores da síntese da Cadence foram mapeadas em 20MHz, os resultados estão presentes na Tabela 4.

Tabela 4: Valores de Potência (mW), Área (μm^2) e Contagem de células para as borboletas DET na base 2 em 180nm a 20MHz com operadores de síntese

	Potência		**Área**		**Células**	
Estrutura	1 ‖	2 ‖	1 ‖	2 ‖	1 ‖	2 ‖
A	6,06	7,20	42.673	47.543	1.546	1.656
B	9,96	7,79	36.638	42.962	1.260	1.392
C	7,74	7,99	38.322	44.677	1.295	1.425
D	9,88	8,34	39.226	45.477	1.374	1.505
E	10,01	9,24	42.457	49.609	1.383	1.528

‖ Número de camadas de *Pipeline*

Na Tabela 4, observa-se uma pequena alteração de valores quando comparada com a Tabela 3. Isto ocorre devido às diferentes células lógicas utilizadas no mapeamento das estruturas de borboletas, pois com restrições de tempo menores, o algorítmo de síntese lógica pode escolher um menor número de células lógicas, e células mais lentas para realizar a mesma função lógica.

A estrutura C, com uma camada de *pipeline*, apresenta o menor consumo de potência entre as estruturas com três multiplicadores, devido ao fato desta estrutura apresentar operação de soma antes da operação de multiplicação em apenas um multiplicador[2]. Como menos operações aritméticas necessitam ser realizadas em um mesmo intervalo de tempo, as células utilizadas nos multiplicadores que não possuem circuito somador nas suas entradas primárias podem ser mais lentas, e portanto consomem menos potência.

5.2 Resultados das borboletas com Multiplicador *Array*

A Tabela 5 apresenta os resultados de síntese lógica em 180nm a 20MHz para as estruturas de borboletas DET na base 2 usando o multiplicador *array*. Por não satisfazer as restrições de tempo para 100MHz foi escolhido 20MHz para a comparação com os resultados das borboletas RNS Seção 5.3. Os demais operadores foram implementados com os operadores de síntese da Cadence. A estrutura E, presente na tabela, refere-se à estrutura proposta neste trabalho (Seção 4.2.1).

Na Tabela 5, observa-se que o maior consumo de potência e maior área são apresentados na estrutura A. Isto ocorre devido à maior complexidade do multiplicador *array*, em relação ao multiplicador utilizado na síntese automática. Desta forma, a estrutura que utiliza um mul-

[2]Nas estruturas B, D e E a maioria dos multiplicadores apresentam operação de soma antes do circuito multiplicador.

Tabela 5: Resultados de Potência (mW) Área (μm^2) e Contagem de Células para as borboletas na base 2 DET com multiplicador Array em 180nm a 20MHz

Estrutura	Potência 2‖	Área 2‖	Células 2‖
A	10,96	56.080	2.421
B	9,71	44.437	1.452
C	8,82	44.216	1.407
D	9,43	45.372	1.504
E	9,95	49.764	1.496

‖ Número de camadas de *Pipeline*

tiplicador a mais em seu algoritmo (estrutura A), é penalizada. As demais estruturas também apresentam consumo de potência e área maiores que os da Tabela 4. Contudo, os valores de área são bem próximos.

Como os resultados de consumo de potência utilizando o multiplicador *array* apresentaram valores maiores que os dos operadores da síntes, os multiplicadores da síntese (Cadence) foram implementados nos demais testes.

5.3 Resultados das borboletas com somadores RNS

As borboletas implementadas com o sistema RNS (apresentadas na Seção 4.3.3) não atingiram as restrições de tempo impostas na síntese lógica para 100MHz, sendo somente sintetizadas com folga de tempo positiva para a frequência de operação de 20MHz e com duas camadas de *Pipeline*.

A Tabela 6 apresenta os resultados de síntese lógica em tecnologia de 180nm para 20MHz de frequência de operação para as estruturas de borboletas sintetizadas com a técnica RNS (Seção 4.3.3). As borboletas com a codificação RNS foram testadas apenas com duas barreiras de *pipeline* por não terem atingido as restrições de tempo com apenas uma barreira.

Tabela 6: Resultados em Potência (mW) Área (μm^2) e Contagem de Células para as borboletas DET na base 2 RNS em 180nm a 20MHz

Estrutura	Potência 2‖	Área 2‖	Células 2‖
A	29,09	70.920	3.194
B	33,41	66.063	2.976
C	34,48	64.627	2.899
D	33,20	58.161	2.659
E	39,65	69.712	3.113

‖ Número de camadas de *Pipeline*

Os resultados da Tabela 6 quando comparados com as sínteses de 20MHz das Tabelas 4 e 1 anteriores apresentam resultados muito maiores. Isto se deve à complexidade das etapas de codificação e decodificação necessárias para o sistema, pois apresentam um hardware adicional para a realização destas etapas [3]. As borboletas utilizando a codificação RNS foram implementadas somente nos somadores após os multiplicadores no caminho do sinal com o intuito de comparar com os resultados com os somadores compressores (Seção 5.4). Dentre os resultados das borboletas com os somadores RNS, o menor consumo de potência é apresentado na estrutura A, mas com a maior área de ocupação.

5.4 Resultados das borboletas com somadores compressores

As estruturas de borboletas utilizando somadores compressores foram sintetizadas utilizando as células padrão da XFAB (X-FAB, 2008) em 180nm a 100MHz e 20MHz, com análise de chaveamento para 10.000 vetores de entrada aleatórios. Os resultados de síntese a 100MHz são apresentados na Tabela 7.

Tabela 7: Resultados de Potência(mW), Área (μm^2) e Contagem de Células para borboletas DET na base 2 com somadores compressores em 180nm a 100MHz

	Potência		**Área**		**Células**	
Estrutura	1 ‖	2 ‖	1 ‖	2 ‖	1 ‖	2 ‖
A	5,80	6,51	48.197	53.144	1.898	2.019
B	10,53	6,96	46.623	49.320	2.283	1.826
C	8,53	7,58	46.322	50.872	2.127	1.979
D	8,82	8,21	48.956	51.622	1.806	2.007
E	9,25	9,09	52.581	57.107	2.573	2.212

‖ Número de camadas de *Pipeline*

Na Tabela 7, pode-se comparar os resultados de consumo de potência para uma e para duas camadas de *pipeline*. Com exceção da estrutura A, que aumentou seu consumo de potência (conforme esperado), as demais estruturas reduziram o consumo de potência, mesmo com o consumo extra da barreira de registradores adicional. Isto ocorreu devido à redução da atividade de chaveamento imposta pela divisão do circuito em blocos menores e pelo uso de células lógicas mais lentas, pois as restrições de tempo do circuito alteraram, aumentando-se o tempo permitido ao circuito para realizar a devida operação.

O aumento do consumo de potência da estrutura A era esperado, pois de acordo com a Figura 20, a segunda barreira de registradores do *pipeline* é posicionada na entrada do circuito e não no interior do mesmo por não ser necessária. Logo, o aumento de potência quando

[3] As etapas de codificação e decodificação são ilustradas na Seção 4.3.3.

comparada a estrutura A com uma e duas camadas de *pipeline* se deve justamente ao consumo extra causado pelo circuito do segundo *pipeline*. O que ressalta o mérito da técnica de *pipeline* para a redução de potência, pois mesmo com o consumo adicional imposto pelo circuito houve uma redução no consumo de potência para as demais estruturas, B, C, D e E (estrutura proposta).

Os resultados de síntese lógica das borboletas DET na base 2 utilizando somadores compressores a 20MHz são apresentados na Tabela 8.

Tabela 8: Resultados de Potência(mW), Área (μm^2) e Contagem de Células para borboletas DET na base 2 com somadores compressores em 180nm a 20MHz

	Potência		**Área**		**Células**	
Estrutura	1 ‖	2 ‖	1 ‖	2 ‖	1 ‖	2 ‖
A	5,44	6,17	44.708	49.430	1.648	1.744
B	10,11	6,53	39.752	46.067	1.414	1.546
C	8,06	7,31	39.737	46.076	1.407	1.536
D	8,44	8,06	33.984	46.992	1.200	1.599
E	9,03	8,56	44.216	51.358	1.553	1.698

‖ Número de camadas de *pipeline*

Os resultados de consumo de potência para a estrutura B operando a 20MHz 8, quando comparados com os resultados de síntese a 100MHz 7, provam que a redução do consumo de potência se deve à redução da atividade de chaveamento quando aplicada a segunda camada de *pipeline* e não ao uso de células mais lentas no circuito (circuito operando de forma mais lenta implica em menos atividade de chaveamento).

A estrutura B apresenta os mais altos consumos de potência para as Tabelas 7 e 8 com um estágio de *pipeline* e um dos mais baixos para dois estágios de *pipeline*. Se a redução de potência fosse apenas devido ao uso de células mais lentas, e, portanto com menor consumo, o simples fato de reduzir as restrições de tempo na síntese, pelo uso de uma frequência de operação mais baixa, deveria levar os resultados de consumo de potência a serem mais próximos dos da Tabela 7 para dois estágios de *pipeline*. O fato de não haver reduzido de forma significativa o consumo de potência, quando sintetizado em 20MHz, prova a eficácia da técnica de *pipeline* para a redução da atividade de *glitching*. O mesmo raciocínio vale para as demais estruturas, mas é mais evidente para a estrutura B com os somadores compressores.

5.4.1 Resultados das borboletas com somadores compressores utilizando a XOR otimizada

A partir dos resultados de simulação da Tabela 2 foi realizada a projeção dos consumos de potência para as estruturas de borboletas utilizando os somadores compressores com a XOR

otimizada operando a 100MHz. Estas estimativas são apresentadas na Tabela 9.

Tabela 9: Projeção dos resultados de Potência(mW) para borboletas DET na base 2 com somadores compressores utilizando a XOR otimizada em 180nm a 100MHz

	Potência	
Estrutura	1 ‖	2 ‖
A	5,69	6,40
B	10,42	6,85
C	8,55	7,60
D	8,84	8,23
E	9,39	9,14

‖ Número de camadas de *Pipeline*

Vale ressaltar que os resultados apresentados na Tabela 9 são projeções de consumo de potência. Estas projeções são realizadas pelo fato de que uma simulação completa da borboleta no ambiente de simulação elétrica (Virtuoso) seria dispendiosa em termos de tempo de análise e configuração do ambiente. Desta forma, a estimativa é projetada na tabela em questão considerando a simulação elétrica apenas dos somadores compressores, substituindo estes valores dos resultados obtidos na elaboração da Tabela 7. A simulação no ambiente elétrico foi validada simulando os circuitos CSA e comparando com os obtidos na síntese lógiica.

Comparando os resultados das Tabelas 7 e 9, observa-se que as estruturas A e B reduziram seus consumos de potência, enquanto as outras estruturas aumentaram. Isto ocorre devido ao fato de que as estruturas A e B somente utilizam somadores compressores 3:2, os quais individualmente tem menores consumos de potência em relação ao somador compressor 4:2 com a porta XOR otimizada.

5.5 Comparações com a literatura

Em (TAKALA; PUNKKA, 2006), pode-se encontrar uma comparação de borboletas com diferentes arquiteturas. Porém, o autor analisa o uso de recursos de hardware compartilhados em série, o que foge do escopo deste trabalho, pois acarreta na necessidade de uma frequência maior de trabalho, na ordem do número de elementos compartilhados. Também compara arquiteturas de borboletas com três multiplicadores e com quatro multiplicadores, como pode ser visto em (TAKALA; PUNKKA, 2006). O autor do artigo realizou a síntese lógica das borboletas em 0,11μm, mas sem deixar claro quais operadores aritméticos foram utilizados.

Na Tabela 10, observa-se resultados de comparação de consumo de potência deste trabalho em relação ao trabalho apresentado em (TAKALA; PUNKKA, 2006). Como pode ser visto na Tabela 10, os resultados de consumo de potência obtidos neste trabalho são maiores que

os presentes na literatura, porém, considerando que foram simulados em uma tecnologia de canal maior, são muito promissores se observada a projeção dos resultados para a tecnologia de 0,11μm (tecnologia utilizada em (TAKALA; PUNKKA, 2006)).

Tabela 10: Resultados de Potências (mW) entre borboletas na base 2 em 100MHz

	Tecnologia	**4 Multiplicadores**	**3 Multiplicadores**
Este trabalho	0,18μm	5,69	6,85
Projeção deste trabalho†	0,11μm	2,53	3,04
(TAKALA; PUNKKA, 2006)	0,11μm	4,34	4,22

†Projeção dos resultados deste trabalho para a tecnologia especificada de acordo com a Equação 2.16

Na Tabela 10 são apresentados os melhores resultados deste trabalho para borboletas com quatro e três multiplicadores em 100MHz, uma projeção destes resultados calculada conforme a Seção 2.3.3 para a tecnologia de 0,11μm com 1,2V de tensão de alimentação e os resultados de (TAKALA; PUNKKA, 2006) para 100MHz. Cabe ressaltar que a latência das borboletas de (TAKALA; PUNKKA, 2006) para quatro multiplicadores é de três ciclos de relógio, não sendo especificada para as borboletas com três multiplicadores, enquanto que a latência para a estrutura A, que foi utilizada nessa tabela, para o resultado de quatro multiplicadores é de dois ciclos de relógio (1 estágio de *pipeline*) e a latência da estrutura B utilizada para o resultado de três multiplicadores é de três ciclos de relógio (2 estágios de *pipeline*).

5.6 Análise crítica

A escolha da borboleta deve ser efetuada levando em conta os critérios resultantes das simulações deste capítulo, ou seja, desempenho (velocidade), área e potência. A borboleta A é a estrutura que apresenta o melhor desempenho, por ter o menor caminho crítico, e o menor consumo de potência.

Para uma implementação de uma FFT, que leve em conta velocidade de operação, qualquer uma das estruturas implementadas poderia ser utilizada, pois todas atenderam as restrições de tempo que permitem a operação em 100MHz. Entretanto, se a métrica for o número de operações aritméticas, a estrutura de borboleta A deve ser a escolhida, pois esta apresenta o menor número destas operações.

As estruturas de borboleta B, C, D e E apresentam três multiplicadores. Destas as estruturas B e C apresentam o mesmo número de operadores aritméticos, ou seja, três multiplicadores e nove somadores/subtratores. Entretanto, a estrutura B é a que apresenta o menor consumo de potência entre as estruturas com três multiplicadores, pelo fato de utilizar apenas somadores compressores 3:2, que são mais otimizados em termos de consumo de potência. A estrutura E

proposta neste trabalho, embora não tenha obtido ganhos em termos de consumo de potência, é a única estrutura que utiliza apenas somadores compressores 4:2. Logo, esta estrutura oferece um potencial de otimização, desde que os somadores compressores sejam otimizados. Além disso, a estrutura E proposta é uma estrutura original, não encontrada na literatura, visto que um resultado intermediário pode ser pré-avaliado e armazenado em memória. É uma estrutura que possui um fluxo de sinal regular (todas as saídas possuem o mesmo grau de caminho crítico).

A estrutura D apresenta diferentes caminhos críticos para as saídas (juntamente com a estrutura C) e também apresenta um conjunto de operações de multiplicação e divisão por 2 (facilmente implementados em hardware), mas que provocam um aumento do erro no sinal de saída. Logo, a sua utilização deve levar em conta estes fatores.

O uso de codificação em HDL com operadores não instanciados, ou seja, o uso dos operadores '+', '-' e '*' no código é mais fácil, gera um código mais limpo e inteligível. Entretanto, a codificação fica por conta da ferramenta de síntese para a escolha de seus circuitos, o que dependendo da aplicação pode não ser uma boa escolha, pois os resultados presentes neste capítulo provam que os circuitos gerados pela síntese lógica nem sempre possuem os melhores resultados, principalmente se o objetivo for o menor consumo de potência.

De acordo com os resultados desse Capítulo, o uso de somadores compressores é a melhor alternativa para a soma de mais de dois sinais simultâneos, e deve ser utilizado em um projeto de circuito digital de borboleta para a aplicação específica de FFT.

O uso de multiplicadores *array* provou não ser a melhor opção por apresentar consumo de potência maior que o da síntese com '*' no código HDL sem instanciação a um multiplicador dedicado. Entretanto, as diferenças apresentadas não foram muito grandes, e caso não se disponha de uma ferramenta de síntese eficiente para o mapeamento do circuito, o multiplicador *array* é uma boa opção de escolha par a implementação dos circuitos de multiplicação da borboleta. Inclusive, caso o uso de operadores mais eficientes, como a própria porta XOR proposta neste trabalho e os somadores compressores forem aplicados no multiplicador *array*, talvez o consumo de potência se equipare ou seja menor que o consumo do multiplicador gerado automaticamente pela ferramenta. Esta é uma análise que deve ser utilizada posteriormente, e que está presente na Seção 6.1 (trabalhos futuros).

A codificação RNS apresentou os piores resultados de consumo de potência e não atingiu as restrições de tempo, mas é uma técnica que merece uma maior atenção, pois neste trabalho foi empregada com fins de comparação com o somador compressor. O sistema numérico por resíduos apresenta uma paralelização do algoritmo, isto torna possível um maior desempenho do sistema. Contudo, a codificação e a decodificação para o RNS são etapas de razoável com-

plexidade, não devendo este sistema ser utilizado para a implementação de algoritmos mais simples que estas etapas e com baixos números de bits de operandos. O sistema RNS parece mais promissor para efetuar algoritmos que envolvam operadores com mais de 32 bits, pois mesmo com os circuitos adicionais da paralelização, a economia de energia seria possível pela redução da propagação de *carry*, por exemplo.

Análises de diferentes operadores aritméticos aplicados nas borboletas da FFT não são muito comuns na literatura, nem tão pouco a comparação de diferentes borboletas. Este trabalho realiza a comparação extensiva de diferentes técnicas em diferentes níveis de abstração, voltados para a economia de energia, e consegue resultados melhores que a literatura encontrada, e melhores que os empregados nas ferramentas de uso tecnológico (no caso do uso de somadores compressores). Embora a estrutura proposta não tenha apresentado os melhores resultados em termos de desempenho, área e consumo de potência, serviu de ponto de partida para a otimização das demais estruturas, pois a investigação de diferentes técnicas tentando otimizá-la foi naturalmente aplicada às demais estruturas, efetivando assim os bons resultados comparativos.

Uma análise da equivalência de consumo de um circuito multiplicador com um circuito somador de mesmo número de bits com dois operandos de entrada pode ser estimada analisando os resultados das tabelas para as diferentes estruturas de borboleta. Avalia-se que o consumo de potência de um multiplicador equivala ao consumo de dois ou três circuitos somadores, devido a proximidade dos resultados de consumo de potência das estruturas A e B, principalmente.

5.7 Resumo do capítulo

Nesse Capítulo são apresentados os resultados obtidos em simulações e relatórios de sínteses lógicas, para as estruturas de borboletas empregando os circuitos de operadores aritméticos descritos nos Capítulos anteriores. No final do Capítulo, é efetuada uma análise crítica dos resultados obtidos neste trabalho. No capítulo seguinte é apresentada a conclusão desta dissertação e as sugestões de trabalhos futuros.

6 CONCLUSÃO

Neste trabalho foram apresentadas cinco diferentes topologias de borboletas na base 2 com decimação no tempo (DET). Uma dessas topologias é a estrutura original que foi traduzida diretamente das equações da borboleta DET na base 2 (utilizando quatro multiplicadores reais). Três outras topologias apresentadas foram baseadas em um trabalho da literatura que propõe dezesseis formas diferentes para a realização de uma multiplicação complexa com apenas três multiplicadores reais. Entretanto, foi verificado que essas dezesseis formas de multiplicação podem ser agrupadas em três formas distintas, devido à equivalência de operadores e ao reposicionamento de sinais de entrada. Finalmente, uma quinta topologia foi proposta no âmbito deste trabalho (também com três multiplicadores reais) e é baseada no armazenamento prévio do produto dos valores real e imaginário dos coeficientes. Os resultados mostraram que apesar da topologia proposta não apresentar ganhos em termos de consumo de potência, ela pode ser melhor explorada a partir do uso de somadores compressores 4:2. Como esta é a única estrutura que possibilita o uso deste tipo de compressor sem o uso do compressor 3:2, logo se este somador 4:2 for implementado de forma eficiente (com reduzido consumo de potência), isto provocará um impacto na redução do consumo de potência da borboleta.

Como as estruturas de borboleta apresentadas utilizam uma grande quantidade de operadores aritméticos (de sete a quinze) nas suas estruturas internas, foram utilizadas diferentes estratégias de projeto para as suas implementações. Inicialmente foram utilizados os operadores aritméticos da ferramenta de síntese lógica automática (Cadence). Após, foram utilizados operadores aritméticos dedicados (somadores compressores, somadores RNS e multiplicadores *array*). Os resultados mostraram que as borboletas apresentam menores consumos de potência com o uso dos somadores compressores em suas estruturas.

Como os somadores compressores apresentam em suas estruturas internas circuitos multiplexadores e portas lógicas XOR (estando estas portas lógicas presentes no caminho crítico), foi proposta neste trabalho uma nova topologia de porta lógica XOR baseada em transistores de passagem. Esta porta lógica XOR é mais eficiente do que uma porta lógica XOR baseada em CMOS nesta aplicação, pelo fato de que ela possibilita a redução da atividade de chaveamento

pelo efeito do filtro RC em suas entradas e não apresenta efeito de consumo de potência de curto-circuito no caminho principal. Esta nova porta lógica XOR foi aplicada aos somadores compressores 3:2 e 4:2. Os resultados mostraram que o somador compressor 3:2 apresenta menor consumo de potência quando do uso da nova porta XOR (quando comparado com o uso da porta XOR da biblioteca da XFAB). Este mesmo cenário não se observou para o somador compressor 4:2, onde o uso da nova porta XOR aumentou o seu consumo de potência, conforme presente no Capítulo 5.

6.1 Trabalhos Futuros

De acordo com o estudo realizado e os resultados obtidos neste trabalho, abre-se a perspectiva para a realização de novas etapas que darão prosseguimento à linha de pesquisa seguida nesta dissertação de mestrado:

- Realizar síntese lógica das estruturas de borboleta descritas neste trabalho com tecnologias de canal menor (65nm, 45nm) para analisar o impacto do percentual de consumo de potência estática em relação à potência dinâmica.
- Aplicar os mesmos testes realizados nas borboletas com os variados operadores aritméticos para diferentes larguras de bits (8, 32, 64 e 128). Espera-se que para valores mais altos de largura de bits, o sistema RNS obtenha resultados melhores do que os apresentados neste trabalho. Além disso, outro aspecto a ser explorado com o uso da codificação RNS é o uso de estágios de *pipeline*, que pode ser utilizado nos circuitos das etapas de codificação/ decodificação.
- Alicação de engenharia reversa nos circuitos gerados pela síntese lógica automática para o entendimento do sistema e comparação com a literatura.
- Realizar a síntese física e a prototipação dos circuitos descritos neste trabalho, no sentido de comprovar experimentalmente os valores simulados e aqui apresentados.
- Reordenar os valores de W_P (Equação 2.4) a serem armazenados em memória ROM, de forma com a distância de Hamming (diferença entre bits em palavas consecutivas) possa ser o menor possível. O objetivo é reduzir a atividade de chaveamento nas entradas dos multiplicadores a partir da ordenação dos coeficientes armazenados em memória ROM.
- Caracterizar a porta lógica XOR proposta e seu dual com inversor na saída, realizando um dimensionamento otimizado, o seu *layout*, levantamento de curvas de resposta e comparações com outras portas XOR presentes na literatura.

- Aplicar a técnica de redução da tensão de alimentação do circuito (*Voltage Scaling*) para a XOR proposta e observar a relação de redução da potência em relação à perda de velocidade de operação.

- Aplicar a técnica de *pipelining* com redução da tensão de alimentação, conforme descrito na Seção 3.1. A tensão de alimentação pode ser reduzida a partir do aumento do atraso com a manutenção do *throughput* original.

- Realizar um projeto *full-custom* para os somadores compressores 3:2 e 4:2 com a nova porta XOR otimizada e comparar com a literatura.

- Aplicar os compressores com a XOR proposta descrita no Capítulo 4 em outros operadores aritméticos como o somador RNS (descrito na Seção 3.4) e/ou circuitos multiplicadores, visando a redução do consumo de potência dos mesmos.

- Estender o estudo de substituição do multiplicador complexo de quatro por três multiplicadores reais para borboletas na base 4.

- Escolher uma metodologia (apresentadas em 2.1.1) e implementar a estrutura completa da FFT usando as borboletas descritas neste trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDREWS; GHOSH; MUHAMED. *Fundamentals of WiMAX: Understanding Broadband Wireless Networking*. Prentice Hall PTR, 2007. Hardcover. ISBN 0132225522. Disponível em: <http://www.worldcat.org/isbn/0132225522>.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. *NBR 15607-1*: Televisão digital terrestre - canal de interatividade parte 1: Protocolos, interfaces físicas e interfaces de software. Rio de Janeiro, 2008.

BAAS, B. M. *An Approach to Low-Power, High Performance, Fast Fourier Transform Processor Design*. 186 p. Tese (Doutorado em Filosofia) — Universidade de Stanford, Stanford, CA, EUA, 1999.

BARANIECKA, A.; JULLIEN, G. On decoding techniques for residue number system realizations of digital signal processing hardware. *IEEE Transactions on Circuits and Systems*, v. 25, n. 11, p. 935–936, nov 1978. ISSN 0098-4094.

BI, G.; JONES, E. Fast conversion between binary and residue numbers. *Electronics Letters*, IEE, v. 24, n. 19, p. 1195–1197, 9 1988.

BRACEWELL, R. *The Fourier Transform and Its Applications*. 3. ed. [S.l.]: McGraw-Hill Science/Engineering/Math, 1999. Hardcover. ISBN 0073039381.

BUI, H. T.; AL-SHERAIDAH, A.; WANG, Y. New 4-transistor xor and xnor designs. In: *ASICs, 2000. AP-ASIC 2000. Proceedings of the Second IEEE Asia Pacific Conference on*. [S.l.: s.n.], 2000. p. 25 –28.

CADENCE DESIGN SYSTEMS, INC. *Low Power in Encounter RTL Compiler*. 2655 Seely Ave., San Jose, CA 95134, USA., 2008. Product Version 8.1.101.

CHANDRAKASAN, A.; SHENG, S.; BRODERSEN, R. Low-power CMOS digital design. *IEEE Journal of Solid-State Circuits*, New York, NY, USA, v. 27, n. 4, p. 473–484, 1992.

CHANG, C.-H.; GU, J.; ZHANG, M. Ultra low-voltage low-power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, v. 51, n. 10, p. 1985 – 1997, oct. 2004. ISSN 1549-8328.

CHERKAUER, B.; FRIEDMAN, E. A hybrid radix-4/radix-8 low power, high speed multiplier architecture for wide bit widths. *IEEE International Symposium on Circuits and Systems - ISCAS*, v. 4, p. 53–56, 1996.

CHHATBAR, T. D. High Speed High Throughput FFT/IFFT Processor ASIC for Mobile WiMax. *ICETET*, IEEE Computer Society, Los Alamitos, CA, USA, p. 402–405, 2009.

COOLEY, J.; TUKEY, J. An algorithm for the machine calculation of the complex fourier series. *Math. Comput*, v. 19, p. 297 – 301, 1965.

COSTA, E. A. C. da. *Operadores Aritméticos de Baixo Consumo para Arquiteturas de Circuitos DSP*. 193 p. Doutorado em Ciência da Computação — UFRGS, Porto Alegre, RS, 2002.

GAJSKI, D.; KUHN, R. New VLSI Tools. *Computer*, v. 16, n. 12, p. 11 –14, dec. 1983. ISSN 0018-9162.

GOEL, S.; ELGAMEL, M.; BAYOUMI, M.; HANAFY, Y. Design methodologies for high-performance noise-tolerant xor-xnor circuits. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, v. 53, n. 4, p. 867 – 878, 2006. ISSN 1549-8328.

GOEL, S.; ELGAMEL, M. A.; BAYOUMI, M. A. Novel design methodology for high-performance xor-xnor circuit design. In: *Proceedings of the 16th symposium on Integrated circuits and systems design*. [S.l.: s.n.], 2003. (SBCCI '03), p. 71–. ISBN 0-7695-2009-X.

GONZALEZ-CONCEJERO, C.; RODELLAR, V.; ALVAREZ-MARQUINA, A.; ICAYA, E.; GOMEZ-VILDA, P. An FFT/IFFT Design versus Altera and Xilinx Cores. In: *Reconfigurable Computing and FPGAs, 2008. ReConFig '08. International Conference on*. [S.l.: s.n.], 2008. p. 337 –342.

HAYKIN, S.; VEEN, B. V. *Sinais e Sistemas*. Porto Alegre, RS: Bookman, 2001. ISBN 0471164747.

HENNESSY, J. L.; PATTERSON, D. A. *Computer architecture: a quantitative approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002. ISBN 1-55860-596-7.

HWANG, K. *Computer Arithmetic: Principles, Architecture and Design*. New York, NY, USA: John Wiley and Sons, 1979. 53-56 p.

IEEE 802.11: Wireless lan medium access control MAC and physical layer PHY specifications. [S.l.], 2007. Disponível em: <http://standards.ieee.org>.

IEEE 802.16: Standard for local and metropolitan area networks. [S.l.], 2003. Disponível em: <http://standards.ieee.org>.

JARDIM, F. d. M. *Treinamento avançado em Redes Wireless*. São Paulo, SP: Digerati Books, 2007.

KIM, J.-H.; PARK, I.-C. Long-point fft processing based on twiddle factor table reduction. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, Oxford University Press, Oxford, UK, E90-A, n. 11, p. 2526–2532, 2007. ISSN 0916-8508.

KIM, T.; JAO, W.; TJIANG, S. Arithmetic optimization using carry-save-adders. In: *Design Automation Conference, 1998. Proceedings*. [S.l.: s.n.], 1998. p. 433 – 438.

LOCKE, D. It's No Contest: 3G Spectrum is More Expensive than WiMAX. *Pyramid Reserch*, Pyramid Research, Cambridge, MA, EUA, 2007.

MOU, Z.-J.; JUTAND, F. 'overturned-stairs' adder trees and multiplier design. *Computers, IEEE Transactions on*, IEEE Computer Society, Los Alamitos, CA, USA, v. 41, n. 8, p. 940 –948, aug 1992. ISSN 0018-9340.

NG, M. C.; VIJAYARAGHAVAN, M.; DAVE, N.; ARVIND; RAGHAVAN, G.; HICKS, J. From wifi to wimax: Techniques for high-level ip reuse across different ofdm protocols. *5th IEEE & ACM Int. Conf. on Formal Methods and Models for Codesign, MEMOCODE*, p. 71 – 80, junho 2007.

OKLOBDZIJA V. G., V. D. L. S. S. A method for speed optimized partial product reduction and generation of fast parallel multipliers using and alghoritmic approach. *IEEE Transaction on Computers*, IEEE Computer Society, 1996.

OLIVEIRA, L. L. *Prototipação e Análise de Circuitos Multiplicadores Array de Baixo Consumo*. 93 p. Dissertação (Mestrado em Engenharia Elétrica) — UFSM, Santa Maria, RS, 2005.

OMONDI, A.; PREMKUMAR, B. *Residue Number Systems: Theory and Implementation*. London, UK: Imperial College Press, 2007. ISBN 1860948669, 9781860948664.

PARHAMI, B. *Computer arithmetic: algorithms and hardware designs*. Oxford, UK: Oxford University Press, 2000. ISBN 0-19-512583-5.

PIEPER, L. Z. *Circuitos Multiplicadores Array de Baixo Consumo de Potência Aplicados a Filtros Adaptativos*. 94 p. Dissertação (Mestrado em Ciência da Computação) — UCPEL, Pelotas - RS, 2008.

RABAEY, J. *Low Power Design Essentials*. New York, NY, EUA: Springer Publishing Company, Incorporated, 2009. ISBN 0387717129, 9780387717128.

RAO, G. R. K.; RADHAMANI, G. *WiMAX: a wireless technology revolution*. Broken Sound Parkway NW and Boca Raton, FL, EUA: Taylor & Francis Group, Auerbach Publications, 2008. Hardcover. ISBN 0849370590.

ROUHOLAMINI M. KAVEHIE, O. M. A. J. S.; NAVI, K. A new design for 7:2 compressors. *Int. Conf. on Computer Systems and Applications*, IEEE/ACS, 2007.

SEIDEL, P.; MCFEARIN, L.; MATULA, D. Binary multiplication radix-32 and radix-256. *Symposium on Computer Arithmetic*, IEEE Computer Society, p. 23–32, 2001.

SHIAVON, A. O. *Estudo e Implementação de Operadores Aritméticos Utilizando a Codificação RNS*. 22 f. Monografia (Trabalho de Conclusão de Curso) — Curso de Engenharia Elétrica, Universidade Católica de Pelotas, Pelotas, RS, 2010.

SZABÓ, N. S.; TANAKA, R. I. *Residue arithmetic and its applications to computer technology*. New York, NY, USA: McGraw-Hill, 1967. xvi + 236 p. (McGraw-Hill series in information processing and computers).

TAKALA, J.; PUNKKA, K. Scalable fft processors and pipelined butterfly units. *J. VLSI Signal Process. Syst.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 43, n. 2-3, p. 113–123, 2006. ISSN 0922-5773.

THUN, R. On residue number system decoding. *Acoustics, Speech, and Signal Processing, IEEE Transactions on*, v. 34, n. 5, p. 1346–1347, oct 1986.

WALLACE, C. S. A suggestion for a fast multiplier. *Electronic Computers, IEEE Transactions on*, EC-13, n. 1, p. 14 –17, feb. 1964. ISSN 0367-7508.

WANG, J.-M.; FANG, S.-C.; FENG, W.-S. New efficient designs for xor and xnor functions on the transistor level. *Solid-State Circuits, IEEE Journal of*, v. 29, n. 7, p. 780 –786, jul 1994. ISSN 0018-9200.

WENZLER, A.; LUDER, E. New structures for complex multipliers and their noise analysis. *Circuits and Systems, 1995. ISCAS '95., 1995 IEEE International Symposium on*, v. 2, p. 1432 –1435 vol.2, apr-3 may 1995.

X-FAB SEMICONDUCTOR FOUNDRIES AG. *0.18 μm CMOS Process*: 0.18 Micron Modular RF enabled CMOS Logic and Analog Technology. Haarbergstrasse 67, D-99097 Erfurt, Alemanha, Agosto 2008. Disponível em: <http://www.xfab.com>.

ZHAO, Y.; ERDOGAN, A. T.; ARSLAN, T. A low-power and domain-specific reconfigurable fft fabric for system-on-chip applications. *Parallel and Distributed Processing Symposium, International*, IEEE Computer Society, Los Alamitos, CA, USA, v. 4, p. 169a, 2005. ISSN 1530-2075.

ZIMMERMANN, R. Datapath synthesis for standard-cell design. *Computer Arithmetic, IEEE Symposium on*, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 207–211, 2009. ISSN 1063-6889.

ZIMMERMANN, R.; FICHTNER, W. Low-power logic styles: Cmos versus pass-transistor logic. *Solid-State Circuits, IEEE Journal of*, v. 32, n. 7, p. 1079–1090, Jul 1997. ISSN 0018-9200.

ZURAS, D.; MCALLISTER, W. Balanced delay trees and combinatorial division in vlsi. *Solid-State Circuits, IEEE Journal of*, v. 21, n. 5, p. 814 – 819, oct 1986. ISSN 0018-9200.

APÊNDICE A – EXEMPLOS DE COFICAÇÃO HDL EM VERILOG

A.1 Codificação de uma borboleta em Verilog

Codificação da estrutura A (Figura 20) em verilog com circuitos de operadores aritméticos (adição, subtraçãoo e multiplicação) a serem gerados automáticamente na síntese lógica.

```
module radix_2_A_tool
  (clk, W_r, W_i, A_r, A_i, B_r, B_i, C_r, C_i, D_r, D_i );

// Definindo parametros
  parameter
    TAM = 16 ;      // quantidade de bits

// Definindo portas de entrada
  input  wire
    clk ;
  input  wire [TAM-1:0]
    W_r, W_i, A_r, A_i, B_r, B_i ;

// Definindo portas de saída
  output reg [TAM-1:0]
    C_r, C_i, D_r, D_i ;

// Definindo sinais internos
  reg [TAM-1:0]    // (Mudar para wire se instanciar)
    Resul_1 ,      // resultado da multiplicação 1
    Resul_2 ,      // resultado da multiplicação 2
    Resul_3 ,      // resultado da multiplicação 3
```

```
    Resul_4 ,      // resultado da multiplicação 4
    Temp_A_r,      // temporário para criar pipeline
    Temp_A_i,      // ...
    Temp_W_r,      // ...
    Temp_W_i,      // ...
    Temp_B_r,      // ...
    Temp_B_i;      // ...
    Pipe_A_r,      // conectado ao segundo pipeline
    Pipe_A_i,      // ...

// Lógica Sequencial,
  always @(posedge clk)  // 2 pipelines
  begin
    Temp_W_r <= W_r ;
    Temp_W_i <= W_i ;
    Temp_B_r <= B_r ;
    Temp_B_i <= B_i ;
    Resul_1 <= Temp_B_r * Temp_W_r ;
    Resul_2 <= Temp_B_i * Temp_W_i ;
    Resul_3 <= Temp_B_i * Temp_W_r ;
    Resul_4 <= Temp_B_r * Temp_W_i ;
    Temp_A_r <= A_r ;
    Temp_A_i <= A_i ;
    Pipe_A_r <= Temp_A_r;
    Pipe_A_i <= Temp_A_i;
  end

// implementação da saída com registradors
  always @(posedge clk)
  begin
    C_r <= Pipe_A_r +    Resul_1  + (- Resul_2) ;
    C_i <= Pipe_A_i +    Resul_3  +    Resul_4  ;
    D_r <= Pipe_A_r + (- Resul_1) +    Resul_2  ;
    D_i <= Pipe_A_i + (- Resul_3) + (- Resul_4) ;
  end
```

```
endmodule // ——— radix_2_A_tool ———
```

A.2 Programa Verficador em Verilog

Codificação de um programa verificador funcional para circuitos multiplicadores, com comandos automáticos para a criação de arquivo para uso na análise de chaveamentoa.

```
‘timescale 1 ns / 1 ns          // Passo de cálculo p/ simulação
module tb_multipliers;

// Definindo parametros
 localparam integer
        PERIOD    = 10;         // periodo do relógio
  parameter integer
        TAM       = 16 ,        // Tamanho em bits dos operadores
        NULO      =  0 ,        // zero
        UM_P      =  1 ,        // um
        UM_N      =  2 ,        // menos um
        RAND      =  3 ;        // números aleatórios
  integer   i ;                 // contador para loop

// Entradas para o DUT
  reg signed [TAM-1:0]
        A ,                     // Operando 1
        B ;                     // Operando 2

// Saídas do DUT
  wire [TAM*2-1:0] S;           // Resultado do DUT

// Declaração de sinais internos
  reg    clk;                   // Relógio do sistema
  reg    alow_random            // habilita valores aleatórios
  reg [45*8:1]  message ;       // vetor 8 bits p/ caracter ASCII
  reg signed [TAM-1:0]
        value_A, value_B ;      // alocação valores randômicos
```

```
  wire [TAM-1:0]
       ZERO     = 16'b0 ,     //  zero -> 0
       POS_ONE  = 16'b01 ,    //  um -> 1
       NEG_ONE  = 16'hFFFF;   //  menos um -> -1
  reg signed [TAM*2-1:0]
       S_test ;               // resultado de verificação

// <<< Instanciação do Multiplicador <<<<<<<<<<<<<<<<<<<<<<<<<<
// trocar "mult16b" pelo nome da "entity" em VHDL ou
// módulo em verilog a ser testado
  mult16b DUT
  (
  .A  ( A ) ,                 // .nome_aqui (nome_lá)
  .B  ( B ) ,
  .S  ( S )
  );
// <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

// --- geração do relógio ---
  initial clk = 1'b0;
  always #(PERIOD/2) clk = ~ clk;

// --- geração de valor aleatório ---
  always @(negedge clk)
    begin
      if ( alow_random )
        begin
          value_A <= 16'h1 + {$random}%(16'hFFFF) ;
          value_B <=  0    + {$random}%(65536)    ;
        end
    end

// ---------------- Mostra mensagens ----------------------------
  always @(message)           // todo check_out tem mensagem
    begin
    $display ("␣%s␣", message);
```

```
    $display ("␣%h␣Esperado,␣%h␣Encontrado",S ,S_test );
  end

// ———————— Aplica estímulos nas entradas ————————————————
  initial
  begin
    // Comandos para analisar a atividade de chaveamento
    $dumpfile ("pre.vcd");    // cria arquivo pre.vcd
    $dumpvars;                // passa valores ao arquivo
    $dumplimit(1000000000);   // limita o arquivo .vcd em 1GB

    // Inicia aplicação de estímulos
    @(posedge clk) alow_random = 0;
    calculate (POS_ONE, POS_ONE);   // 1 x 1 = 1!
    check_out (UM_P);               // compara e mostra
    @(posedge clk) ;
    calculate (NEG_ONE, POS_ONE);   // -1 x 1 = -1!
    check_out (UM_N);
    @(posedge clk) ;
    calculate (POS_ONE, NEG_ONE);   // 1 x -1 = -1!
    check_out (UM_N);
    @(posedge clk) ;
    calculate (NEG_ONE, NEG_ONE);   // -1 x -1 = 1!
    check_out (UM_P);
    @(posedge clk) ;
    calculate (ZERO, POS_ONE);      // 0 x 1 = 0!
    check_out (NULO);
    @(posedge clk) ;
    calculate (NEG_ONE, ZERO);      // -1 x 0 = 0!
    check_out (NULO);
    @(posedge clk) ;
    calculate (16'h5A8F, 16'h5663);// 5A8F x 5663 = 1E8F0F4D!
    S_test = 32'h1E8F0F4D;
    check_out (RAND);

    // valores aleatórios
```

```
    @(posedge clk) alow_random = 1;// entradas randomicas
    for (i=0; i <= 10000; i=i+1)   // 10.000 vetores de teste
    begin
      repeat(2) @(posedge clk);
      calculate (value_A, value_B);
      check_out (RAND);            // compara e mostra
    end
    $stop;
  end

// ------------ tasks ------------------------------------------
task calculate (      // entra valores para mandar ao DUT
  input reg signed [TAM-1:0] a,
  input reg signed [TAM-1:0] b
                );
  begin
    A = a;            // envia sinal para o DUT
    B = b;            // ...
    S_test =  a * b; // Calcula para compação com DUT
  end
endtask

task check_out ( // compara c/ entrada e mostra
  input integer EXPECTED
                );
  begin
  @(negedge clk) case (EXPECTED)
  NULO : if (S != 0 || S != S_test )
           message = "␣Experado␣ZERO␣\n␣TESTE␣FALHOU␣\n";
         else
           message = "␣Teste␣zero␣passou␣\n";

  UM_P: if ( S != 1 || S != S_test )
          message = "␣Experado␣UM␣\n␣TESTE␣FALHOU␣\n";
        else
          message = "␣Teste␣um␣passou␣\n";
```

```
  UM_N : if ( S != 32'hFFFFFFFF || S != S_test )
           message = "␣Experado␣MENOS␣UM␣\n␣TESTE␣FALHOU␣\n";
         else
           message = "␣Teste␣menos␣um␣passou␣\n";

  RAND : if ( S != S_test )
           message = "␣VALORES␣ALEATORIOS\n␣TESTE␣FALHOU␣\n";
         else
           message = "␣Teste␣aleatorio␣passou␣\n";

  default: message = "␣Falha␣no␣Teste␣e␣no␣TB␣";
  endcase
  end
endtask

endmodule        \\ ------ tb_multipliers --------
```

APÊNDICE B – CÁLCULO DO TWIDDLE FACTOR W

A seguinte rotina pode ser escrita no Matlab para o cálculo e visualização do fator de giro *W*.

```
P = 64 ; % Número de pontos da FFT
k = P;
for k = 0 : +1 : P/2-1
  W  (1, k+1) = single( exp(-i*2*k*pi/P) );
  W_r(1, k+1) = real( W(1, k+1) ) ;
  W_i(1, k+1) = imag( W(1, k+1) ) ;
end
% converte cartesiano para polar
[Phase, Module]= cart2pol(W_r, W_i);
polar(Phase, Module,'or');
```

APÊNDICE C – ARTIGOS PUBLICADOS

FONSECA, M. B.; COSTA, E. A. C. da; MARTINS, J. B. S **Architectural Exploration in the Butterflies of the Radix-2 Decimation in Time FFT Algorithm**. In: XVI Iberchip Workshop, 2010, Foz do Iguaçu - Brasil.

FONSECA, M. B.; COSTA, E. A. C. da; MARTINS, J. B. S **Synthesis Based Dedicated Radix-2 DIT Butterflies Structures for a Low Power FFT Implementation**. In: SIM 2010 - XXV South Symposium on Microeletronics, 2010, Porto Alegre - Brasil.