UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM MICROELETRÔNICA-PGMICRO

SIDINEI GHISSONI

Decomposição de Coeficientes Trigonométricos para a Redução de Área e Potência em Arquiteturas FFT Híbridas na Base 2

Tese apresentada como requisito parcial para a obtenção do grau de Doutor em Microeletrônica

Prof. Dr. Ricardo Augusto da Luz Reis
Orientador

Porto Alegre, dezembro de 2012.

CIP - Catalogação na Publicação

Ghissoni, Sidinei
Decomposição de Coeficientes Trigonométricos para a Redução de Área e Potência em Arquiteturas FFT Híbridas na Base 2 / Sidinei Ghissoni. -- 2012.
101 f.

Orientador: Ricardo Augusto da Luz Reis.

Tese (Doutorado) -- Universidade Federal do Rio Grande do Sul, Instituto de Informática, Programa de Pós-Graduação em Microeletrônica, Porto Alegre, BR-RS, 2012.

1. Coeficientes trigonométricos. 2. FFT. 3. Métrica de porta. 4. CMM. 5. Otimização de área e potência. I. Reis, Ricardo Augusto da Luz, orient. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da UFRGS com os dados fornecidos pelo(a) autor(a).

AGRADECIMENTOS

Ao longo destes cinco anos de pesquisa para elaboração deste trabalho de doutorado tenho muito a quem agradecer e provavelmente vou me esquecer de mencionar de pessoas de quem me ajudaram.

Primeiramente queria agradecer a Deus por ter me ajudado nos momentos que mais precisava, por ter me dado força para superar todas as adversidades encontradas ao longo de todo o trabalho.

A minha família, aos meus pais Mário e Inês e, ao meu irmão Rudinei pelo apoio e compreensão, incentivando-me sempre em todos os momentos difíceis superando os desafios encontrados. Em especial à minha esposa, Rosangela, que sempre me apoiou incondicionalmente para alcançar meus objetivos e que por muitas vezes aceitou sem problemas minhas desculpas por trabalhar nas “férias” e nos fins de semanas.

Ao meu orientador Professor Dr. Ricardo Reis, pelo auxílio e dedicação durante as atividades de pesquisa. Ao professor Dr. Eduardo Costa que apesar de não ser oficialmente coorientador contribuiu com dedicação na orientação e no desenvolvimento da ideia deste trabalho. Ao professor Dr. José Monteiro que aceitou em me orientar no estágio realizado no INESC-ID Lisboa em 2010 que onde de fato iniciou o desenvolvimento desta tese. Não poderia deixar de mencionar o pesquisador Dr. Levent Aksoy do INESC-ID pelo apoio inicial no estudo da técnica CMM. Também queria agradecer ao Dr. Cristiano Lazzari pelo convívio proporcionado no tempo que morei em Lisboa e pelos macetes sobre a programação em VHDL.

Queria também deixar meu agradecimento a todos os colegas do PGMICRO que no primeiro ano de doutorado eram parceiros nas discussões sobre os conteúdos abordados nas disciplinas e pela amizade construída. Ao PGMICRO pelo apoio financeiro prestado para participação nas apresentações do trabalho de doutorado nas conferências nacionais e internacionais.

Por fim, agradeço a todos os demais que deram a sua contribuição para elaboração deste trabalho.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ASIC	*Application Specific Integrated Circuit* (Circuito integrado de Aplicação específica)
BWA	*Broadband Wireless Access*
CMM	*Constant Matrix Multiplication* (Multiplicação de Matrizes de Constantes)
CMOS	*Complementary Metal-Oxide Semiconductor* (Semicondutor Metal-Óxido Complementar)
CSA	*Carry-Save Adder* (Somador Carry-Saver)
CSD	*Canonical Signed Digit* (Representação Canônica)
CSE	*Common Subexpretions Eliminations* (Eliminação de Subexpressao Comum)
DAG	*Directed Acyclic Graphs* (Grafo Acíclico Direto)
DCT	*Discrete Cosine Transform* (Transformada Discreta Cosseno)
DFT	*Discrete Fourier Transform* (Transformada de Fourier Discreta)
DIF	Decimação no Domínio da Frequência
DIT	Decimação no Domínio do Tempo
DSP	*Digital Signal Processor* (Processador de Sinal Digital)
FA	*Full Adder* (Somador Completo)
FFT	*Fast Fourier Transform* (Transformada Rápida de Fourier)
FPGA	*Field Programmable Gate Array* (Circuito de Matrizes Programáveis)
HA	*Half Adder* (Meio Somador)
IDFT	*Inverse Discrete Fourier Transform* (Transformada de Fourier Discreta Inversa)
IFFT	*Inverse Fast Fourier Transform* (Transformada Rápida de Fourier Inversa)
MCM	*Multiple Constant Multiplication* (Multiplicação por constantes múltiplas)
MDC	*Multiple-Path Delay Commutator* (Comutador de múltiplos caminhos de atraso)

MOSFET	*Metal Oxide Semiconductor Field Effect Transistor*
MSD	*Minimum Signed Digit* (Representação Mínima de Dígitos)
MUX	Porta lógica Multiplexador
NP	*Nondeterministic Polynomial* (Polinômio não determinístico)
OFDM	*Orthogonal Frequency Division Multiplexing* (Multiplicação por divisão Ortogonal de frequência)
$R2^nSDF$	Radix-2^n *Single-path Delay Feedback* (Arquitetura FFT de Base 2^n com retorno de único caminho de atraso)
RCA	*Ripple Carry Adder* (Somador Ripple-Carry)
RemB	Reconfiguração de Múltiplos Blocos
SCM	*Single Constant Multiplication* (Multiplicação de Constantes Únicas)
SD	*Signed-Digit* (Dígito de Sinal)
SDF	*Single-Path Delay Feedback* (Retorno de único caminho de atraso)
SOP	Representação por Soma de Produtos
VLSI	*Very Large Scale Integration* (Integração de muito Alta Escala)
WI-FI	Nome comercial da rede WLAN baseada no protocolo IEEE 802.11
WIMAX	*Worldwide Interoperability for Microwave Access* (Interoperabilidade Mundial para Acesso de Microondas, nome comercial de uma rede WMAN baseada no protocolo 802.16 da IEEE)
WLAN	*Wireless Local Area Network* (Rede sem fio de área local)

LISTA DE SÍMBOLOS

j	Operador imaginário (complexo)
N	Pontos
r	Operador real
W	Coeficiente Trigonométrico (Twiddle factor)
π	Relação entre as grandezas do perímetro de uma circunferência (p) e diâmetro(d) (p/d), cujo valor aproximado é 3,1415

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A crescente utilização de equipamentos móveis que empregam a transformada rápida de Fourier (FFT) nas operações de sinal digital pode ter seu uso restrito devido ao comprometimento da durabilidade da bateria e de suas dimensões. Estas possíveis limitações de uso fazem crescer a necessidade do desenvolvimento de técnicas que visam à otimização nos três requisitos básicos de projeto digital: dissipação de potência, área e atraso. Para tanto, é abordado neste trabalho um método que realiza a implementação de arquiteturas FFT com ênfase na otimização através da decomposição dos coeficientes trigonométricos. No cálculo da FFT, as borboletas desempenham um papel central, uma vez que permitem o cálculo de termos complexos. Neste cálculo, que envolve multiplicações dos dados de entrada com coeficientes trigonométricos apropriados, a otimização das borboletas pode contribuir diretamente para a redução de potência e área. Na técnica proposta são analisados quais são os coeficientes trigonométricos existentes na arquitetura FFT utilizada como base e a escolha para decomposição será o que apresentar o menor custo de implementação em hardware. A decomposição de um coeficiente deve garantir a reconstituição de todos os demais coeficientes necessários para a implementação de toda a arquitetura FFT. Assim, a decomposição diminui o número de coeficientes necessários para reconstruir a FFT original. O conjunto dos novos coeficientes gerados são implementados com apenas somadores\subtratores e deslocamentos através de Multiplicação de Matrizes Constantes (CMM – Constant Matrix Multiplication), associados a um sistema de controle com multiplexadores que controlam o caminho para a correta operação da FFT. As implementações dos circuitos somadores/subtratores são realizadas com métrica no nível de portas lógicas, visando menor atraso e dissipação de potência para topologias com somadores dos tipos CSA (Carry Save Adder) e Ripple carry. Os resultados apresentados pelo método proposto, quando comparados com soluções da literatura, são significativamente satisfatórios, pois minimizaram a dissipação de potência e área em 30% e 24% respectivamente. Os resultados apresentam também a redução de componentes somadores necessários para a implementação de arquiteturas FFTs.

Palavras-Chave: Coeficientes trigonométricos, métrica de porta, FFT, CMM, base-2, potência, área.

Trigonometric Coefficients Decomposition for Area and Power Reduction in Hybrid Radix-2 FFT Architectures

ABSTRACT

The increasing use of mobile devices using the Fast Fourier Transform (FFT) operations in digital signal may have its use restricted due compromising the durability of the battery and its dimensions. These possible limitations on usage makes grow the need to develop techniques aimed at optimizing the three basic requirements of digital design: power dissipation, area and delay. Therefore, this thesis discusses a method that performs the FFT implementation of architectures with emphasis on optimization through decomposition of twiddle factors (trigonometric coefficients). In the FFT the butterflies play a key role, since it allows the computation of complex terms. In this calculation, which involves multiplications of input data with appropriate twiddle factors, optimization of the butterflies can contribute directly to the reduction in power and area. In the proposed technique are analyzed what are the twiddle factors existing in FFT architecture used as a basis and to choose the decomposition that provide the lowest cost hardware implementation. The decomposition of coefficient to must ensure the rebuilding of all the other twiddle factors necessary for the implementation of the architecture FFT. Thus, the decomposition decreases the number of twiddle factors needed to reconstruct the original FFT. The new sets of coefficients generated are implemented with only adders\subtracters and shifting through of Constants Matrix Multiplication (CMM). A control system of multiplexers makes the way for the correct operation of the FFT. The implementations of the circuits arithmetic adders/subtracters are performed at the gate level, seeking lower delay and power consumption for topologies with adders types of CSA (Carry Save Adder) and Ripple carry. The results presented by the proposed method, compared with literature solutions are significantly satisfactory, since minimized power dissipation and area as well as reduced component adders required for implementation architectures FFTs.

Keywords: Twiddle factors, gate-level, CMM, radix-2, low-power, area.

1 INTRODUÇÃO

A evolução tecnológica e a crescente integração de sistemas em chips, associada a uma forte demanda por meios de comunicação cada vez mais rápidos e principalmente com acesso sem fio como (BWA – *Broadband Wireless Access*), sistemas 3G, 4G e Wi-Fi, crescem vigorosamente e acabam configurando-se em complexos desafios tecnológicos. O alto desempenho computacional exigido por estes sistemas compreende atender o processamento de sinais digitais, onde o conceito da Transformada de Fourier (TF) está fortemente presente (HAYKIN; VEEN, 2001).

Computacionalmente a TF é realizada pela Transformada Discreta de Fourier (DFT- *Discrete Fourier Transform*) que, para realizar uma operação de N pontos necessitam de N^2 multiplicações, o que torna o processo computacionalmente custoso. Para reduzir este número de operações e, consequentemente, o custo de implementação em hardware, é utilizada a Transformada Rápida de Fourier (FFT- *Fast Fourier Transform*), onde o número de multiplicações é reduzido para $N \log_2^N$ (COOLEY; TUKEY, 1965). A redução do número de operações (multiplicações e adições) tornou a FFT uma das principais ferramentas no processamento de sinais digitais (HAYKIN; VEEN, 2001).

Apesar de a FFT ter reduzido o número de operações para executar a DFT, o grande problema na implementação desta arquitetura é atender aos requisitos de dissipação de potência, área e atraso, visto o grande número de operadores aritméticos envolvidos. Um dos caminhos para se tentar amenizar este problema é a reutilização de parte da própria arquitetura. Essa reutilização impacta diretamente em obter todos os coeficientes trigonométricos (*twiddle factors*) da FFT sem comprometer a área, dissipação de potência e atraso (GHISSONI et al., 2011).

A equação 1.1 apresenta o coeficiente trigonométrico (*twiddle factor*) para uma FFT genérica (VAN LOAN, 1992).

$$e^{-j\frac{2\pi k}{N}} = \cos\frac{2\pi k}{N} - j sen\frac{2\pi k}{N} \tag{1.1}$$

Pela equação 1.1 é possível observar que o coeficiente trigonométrico é composto por uma parcela real e outra imaginária, ambas variando conforme a ordem k e o número de pontos N da FFT. Assim, a reutilização desses coeficientes só se faz possível através de um sistema que permita manipular e decompor as parcelas de modo que consiga reproduzir o próprio coeficiente original com o mínimo de perdas (OH; LIM, 2005).

Outro aspecto importante no processo de decomposição dos coeficientes trigonométricos é construir arquiteturas cada vez menores e mais rápidas. Esta miniaturização ocorre pela maior proximidade de seus componentes, que acarreta uma diminuição do caminho crítico e possibilita redução do número de componentes aritméticos e de operações executadas (MACLEOD, 2005).

Para potencializar a otimização das arquiteturas FFT, decompor e reduzir o número de coeficientes não é o suficiente. Deve-se também explorar várias outras técnicas simultaneamente. Fazer a associação da técnica de Multiplicação de Matrizes de Constantes (CMM - *Constant Matrix Multiplications*) ao nível de porta lógica permite não somente aumentar a redução no número de componentes, mas também a área com menor custo em relação ao relacionamento atraso e potência (GHISSONI et al., 2010) e GHISSONI et al., 2011).

Devido às suas propriedades, o problema de decomposição dos coeficientes trigonométricos responsáveis pela multiplicação das borboletas nas FFTs de base 2 foi escolhido como tema desta pesquisa. Em especial, serão analisados os efeitos desta decomposição nos diferentes estágios das arquiteturas menores que compõem as arquiteturas híbridas, com destaque para a reutilização de parte da própria arquitetura.

As implementações dos componentes das arquiteturas estudadas dar-se-á com emprego da métrica ao nível de porta lógica e exploração de diferentes topologias de somadores utilizados com o intuito de reduzir a dissipação de potência, área e atraso.

1.1 Motivação

A demanda por dispositivos móveis com ampla variedade de funções e complexidade apresenta-se como uma grande oportunidade de estudos e está presente na maioria dos equipamentos de comunicação. Para a viabilização da transmissão de dados em alguns desses equipamentos, é necessário o uso de sistemas de modulação como multiplexação ortogonal por divisão de frequência (OFDM - *Orthogonal Frequency Division Multiplexing*), em que, para realizar estas operações, faz-se necessário o uso da FFT.

A importância que a FFT atualmente exerce sobre os meios de comunicação motiva a buscar e sanar as lacunas deixadas pela literatura no estudo de técnicas de otimização a respeito dessas arquiteturas. Assim, evidencia-se um desafio contínuo e natural para permitir o avanço de dispositivos digitais que são fundamentais para o desenvolvimento dos meios de comunicação. Nesta linha, apresenta-se como um caminho promissor o uso de técnicas de redução associadas ao reaproveitamento de componentes, visando à redução de área e dissipação de potência com o mínimo de impacto no desempenho de funcionamento.

A grande maioria das aplicações em processamento de sinais que realizam FFT requer grande desempenho. Para aumentar o desempenho dessas arquiteturas passa-se obrigatoriamente por um estudo de aplicação de técnicas que visam à redução de componentes, dissipação de potência e atraso. Na verdade, a relação dessas técnicas apresenta-se como um dilema em circuitos integrados, pois encontrar uma forma que garanta um comportamento aceitável entre elas permite obter grandes resultados, mas é um grande desafio a ser superado.

Outro aspecto a ser considerado é que as arquiteturas FFTs, por realizarem um elevado número de operações de multiplicação quando estas são implementadas em hardware, ocupam uma grande área. Por isso, é fundamental analisar a aplicação de

técnicas que permitam a redução da área. Nesta linha, vários pesquisadores se mantêm instigados a estudar e propor algoritmos no intuito de encontrar implementações de arquiteturas FFT com menor número de operações e elementos (OH; LIM, 2004), (OH; LIM, 2005), (HAN *et al*., 2008), (HAN; PARK, 2008), (MACLEOD, 2005), (GUSTAFASSON *et al*., 2006), (GUSTAFSSON, 2007a), (VORONENKO; PUSHEL, 2007), (QURESHI; GUSTAFSSON, 2009) e (WU; LIU, 2009). Entretanto, as várias pesquisas apresentadas por estes pesquisadores focam na otimização de componentes, mas não permitem a redução da área, atraso e dissipação de potência simultaneamente. Alguns algoritmos baseados em grafos para filtros FIR (AKSOY et al., 2008) podem ser ajustados para redução de componentes na forma de matrizes, mas não fornecem uma redução de área e dissipação de potência nas mesmas especificações de atraso quando submetidos à síntese lógica ou física por ferramentas comerciais.

Analisando a trigonometria matemática da FFT, a decomposição dos coeficientes trigonométricos apresenta-se como um caminho favorável, pois permite reduzir a área através da redução do número de componentes, reduzindo, portanto, a dissipação de potência (QURESHI; GUSTAFSSON, 2009). Porém, é necessário investigar meios que atendam o fato de que é necessário operar com coeficientes complexos e que estes se repetem ao longo do processo. Obter uma eficiente resposta em termos de otimização de dissipação de potência e atraso, sem comprometer significativamente a área, é essencial para todo projeto.

Outro ponto motivacional para o desenvolvimento deste trabalho é que na análise da decomposição dos coeficientes trigonométricos, não apenas a redução de componentes, mas também o atraso deve ser levado em consideração. O atraso é geralmente estimado como sendo o número máximo de operações em série que resulta em um somatório de diferentes constantes de multiplicação. Considerando que as arquiteturas FFT de N pontos englobam $\frac{N}{2}\log_2^N$ coeficientes trigonométricos, as implementações dessas arquiteturas em série ou paralelo muitas vezes não satisfazem a maioria das especificações de área e dissipação de potência exigidas (AHMADINIA; AHMAD; ARSLAN, 2007). Assim, o desenvolvimento de arquiteturas híbridas associadas a técnicas de manipulação desses coeficientes trigonométricos permite melhorar o relacionamento entre o atraso e área dessas arquiteturas.

Neste sentido, a grande importância das arquiteturas FFTs nos meios de comunicação e as lacunas ainda existentes no estudo de técnicas de otimização de área, dissipação de potência e atraso com foco em decomposição de coeficientes trigonométricos serviram como importante motivação para o desenvolvimento deste trabalho de pesquisa.

1.2 Objetivo

O principal objetivo desta tese é apresentar uma técnica que possa ser aplicada na otimização de área e dissipação de potência para arquiteturas FFTs na base 2 baseada na decomposição dos coeficientes trigonométricos das borboletas. Esta técnica deve avaliar quais são os coeficientes que permitem ser decompostos para serem implementados com menor custo de hardware. A implementação contará com um sistema de controle que, através de multiplexadores, determinará a correta sequência para as multiplicações dos novos coeficientes gerados pela decomposição.

A técnica de decomposição deve permitir concretizar o objetivo de obter através da decomposição dos coeficientes, o menor custo de a área, dissipação de potência e atraso. Assim, a análise do custo de cada operação (multiplicação adição/subtração) e das próximas deve também ser considerado. Nas técnicas apresentadas na literatura que visam o objetivo proposto, as operações são geralmente realizadas através do uso da identidade trigonométrica escolhida através de circuitos multiplexadores (HAM et al., 2008). No entanto, existem também algoritmos que utilizam Multiplicação de Constantes Múltiplas (MCM) (AKSOY et al., 2008; e QURESHI; GUSTAFSSON, 2009), que mapeiam qualquer operação de multiplicação a partir de uma entrada em adições e/ou subtrações e deslocamentos. As operações MCM são, na maioria das vezes, assumidas como uma operação de duas entradas que geralmente são executadas com Somadores *Ripple Carry* (RCA) ou somadores *Carry Look-ahead*. Estes somadores geralmente causam o aumento no tempo de operação. Desta forma, neste trabalho utiliza-se a técnica CMM (GHISSONI et al., 2010; e GHISSONI et al., 2011) onde múltiplas entradas são tratadas simultaneamente através de somadores Carry-Save (CSA) (KIM; JOA; TJIANG, 1998; e SOHN, 2004). O uso destas topologias de somadores possibilita aumentar a execução de múltiplas adições de operandos. Contudo, embora existam técnicas de mapeamento (KIM; JOA; TJIANG, 1998) que convertem adição/subtração em operações de alta velocidade usando CSAs, estas acabam não minimizando o número de somadores necessários e comprometem o desempenho da arquitetura. Assim, neste trabalho, a decomposição de coeficientes através do uso de somadores com mais de dois operandos mostrará que é possível aumentar o desempenho destas arquiteturas.

As novas arquiteturas FFTs, implementadas com menor número de coeficientes, também objetivarão analisar a distância de *Hamming* sobre a sequência das operações de multiplicação dos coeficientes realizadas nas borboletas da FFT. A redução da distância de *Hamming* entre as transições dos coeficientes implicará na redução de dissipação de potência de chaveamento do circuito e, consequentemente, da dissipação total de potência.

1.3 Contribuições do Trabalho

O trabalho de tese apresenta as seguintes contribuições:

- Apresentação de um algoritmo baseado na decomposição de coeficientes de uma FFT que permita a implementação de arquiteturas série e híbridas visando à redução de área e potência sem comprometer o atraso do circuito;
- Associação da técnica de decomposição de coeficientes trigonométricos na implementação de arquiteturas série e híbridas através do uso de CMM, permitindo, assim, uma maior redução de componentes somadores/subtratores que os apresentados por Ham *et al.* (2008), Qureshi; Gustafsson (2009) e Aksoy *et al.* (2012). Esta redução ocorre devido ao compartilhamento de coeficientes parciais, bem como a utilização de uma métrica no nível de portas lógicas na construção desses componentes para obter melhores resultados de otimização de dissipação de potência;

- Apresentação de uma redução maior em termos de área e dissipação de potência em comparação às arquiteturas híbridas apresentadas por MacLeod (2005), Ahmadinia; Ahmad; Arslan (2007) e Qureshi; Gustafsson (2009). Esta redução se dará através da adaptação das operações de multiplicação com a decomposição de algumas constantes das borboletas existentes entre os diferentes estágios da FFT e sua reutilização através de um sistema de controle que escolhe a correta multiplexação das constantes.

1.4 Organização do trabalho

O trabalho elaborado está estruturado em cinco capítulos. Após a introdução, destaca-se, já no capítulo 2, a apresentação dos conceitos e características das arquiteturas FFTs. Neste capítulo também é apresentada a revisão bibliográfica das técnicas de otimização para implementação de arquiteturas FFTs. No capítulo 3, apresenta-se o método de otimização de redução de componentes. No capítulo 4, a técnica é aplicada na reutilização da própria arquitetura baseada na decomposição das constantes trigonométricas existentes. Neste capítulo é apresentado também o ganho da técnica proposta com o uso de ordenação de coeficientes trigonométricos. Quanto às conclusões referentes ao trabalho realizado e as perspectivas para próximos trabalhos, são abordadas no capítulo 5.

2 A TRANSFORMADA RÁPIDA DE FOURIER (FFT)

A Transformada de Fourier é uma ferramenta utilizada tanto em meios de comunicação como em aplicações BWA. Em geral, nas aplicações computacionais a FFT é utilizada por trabalhar com a representação digital e por exigir uma estrutura com menos operações de multiplicação que a DFT (*Discrete Fourier Transform*). Assim, neste capítulo, serão abordados os conceitos matemáticos que servirão como auxílio para o desenvolvimento da metodologia que permite otimizar a área, atraso e dissipação de potência das arquiteturas FFTs.

2.1 A Transformada Rápida de Fourier

O desenvolvimento da computação digital fez com que operações realizadas no domínio contínuo, que necessitam de grande quantidade de memória (idealmente infinita) passassem para o domínio discreto. Desta maneira, a Transformada de Fourier e sua inversa, Transformada Inversa de Fourier, foram substituídas pela Transformada Discreta de Fourier (DFT) e Transformada Inversa Discreta de Fourier (IDFT) respectivamente (COOLEY; TUKEY, 1965).

As equações 2.1 e 2.2 representam matematicamente a DFT e IDFT:

$$X(K) = \sum_{n=0}^{N-1} x(n) \mathbf{W}_N^{Kn}, 0 \le k \le N-1 \tag{2.1}$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \mathbf{W}_N^{-Kn}, 0 \le k \le N-1 \tag{2.2}$$

$$W_N = e^{-j\frac{2\pi}{N}} \tag{2.3}$$

Nas equações 2.1 e 2.2,

K é a ordem da DFT,

N é o número de pontos da amostra do sinal,

x são os pontos no domínio do tempo e X são os pontos no domínio da frequência,

n é a posição de cada ponto e

W_N é o coeficiente trigonométrico (*twiddle factor*) representado na equação 2.3.

Pelas equações 2.1 ou 2.2 é possível observar que, para operar um conjunto de N pontos, são necessárias N^2 operações de multiplicação. Este número de operações

impacta diretamente no tempo de operação nos dispositivos computacionais que as utilizam.

Nas operações realizadas nas equações 2.1 e 2.2 os operandos X(K), x(n) podem ser complexos, deste modo, precisam realizar operações complexas. Uma multiplicação complexa gera como resultado uma parte real (r) e a outra imaginária (j).

Na equação 2.4 é apresentado o resultado da multiplicação entre dois operandos complexos quaisquer A e B. Sendo $A = (A_r + jA_j)$ e $B = (B_r + jB_j)$, onde A_r e B_r são as partes reais e A_j e B_j as partes complexas respectivamente, e o operador complexo j é igual $\sqrt{-1}$.

$$AB = (A_r + jA_j)(B_r + jB_j) \tag{2.4}$$

Fazendo a multiplicação dos operandos da equação 2.4, é obtido como resultado a expressão 2.5.

$$AB = (A_r B_r) + j(A_r B_j) + j(A_j B_r) + (A_j B_j) \tag{2.5}$$

A equação 2.5 mostra que a operação necessita 4 multiplicadores reais e 3 somadores. Na implementação em hardware as operações reais e complexas são realizadas em diferentes partes para obter resultados separados. Nas equações 2.6 e 2.7 podem ser encontradas as parcelas reais e complexas, respectivamente:

$$(AB)_r = (A_r B_r) + (A_j B_j) \tag{2.6}$$

$$(AB)_j = j(A_r B_j) + (A_j B_r) \tag{2.7}$$

Analisando as equações 2.6 e 2.7, observa-se que são necessários 4 multiplicadores e 2 somadores para implementação de um circuito em hardware, o que demonstra que a implementação de uma DFT em hardware demanda um elevado custo.

Para reduzir o número de operações e aumentar o desempenho na solução da Transformada de Fourier, ao longo dos últimos anos, foram propostos vários algoritmos tendo como destaque o algoritmo de base 2 (COOLEY; TUKEY, 1965). Este algoritmo é baseado no processamento central de borboletas (*butterflies*) que se aproveita da simetria da DFT. A operação desta borboleta permite que a computação realizada pela DFT seja dividida ao meio. Desta forma, uma sequência de N pontos é dividida em duas partes de N/2 pontos, nomeadas de par e ímpar. O nome de cada parte se deve por agruparem os endereços pares e ímpares dos pontos do conjunto de N pontos. Esta descoberta provocou um verdadeiro aumento de desempenho computacional na solução da transformada de Fourier, pois reduziu as N^2 operações de multiplicações exigidas na DFT para apenas $N \log_2^N$ multiplicações.

A figura 2.1 apresenta a operação com decimação no tempo (DIT) da FFT na base 2 para 8 pontos. Os valores da amostra são decimados a cada estágio da FFT onde as operações são executadas a cada par de entradas.

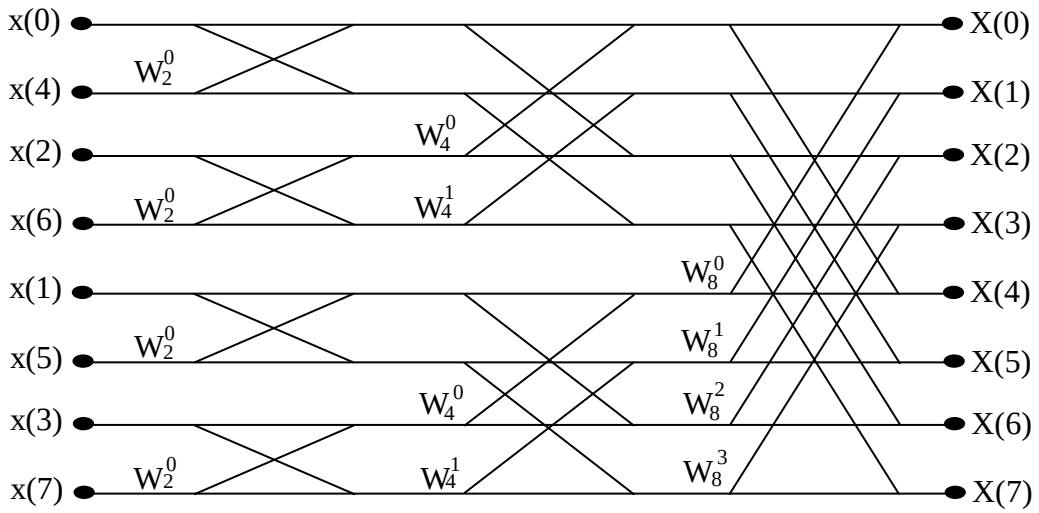


Figura 2.1: Operação da borboleta DIT na base 2 para uma FFT 8 pontos.

Na figura 2.1 pode-se observar que a coluna de borboletas de entrada está rearranjada em cima com borboletas pares e embaixo com borboletas ímpares. Para garantir que os cruzamentos de entrada mantenham a ordem de saída os pontos de entrada, são representados pela ordem inversa de seus endereços originais em binário. Assim, o endereço X(4) vem antes que X(2), pois $(2)_{10}$ e $(4)_{10}$ em representação binária são (100) e (010) respectivamente, mas após fazer a inversão do sentido dos bits (*bit invert*) passam a ser 001 e 010 respectivamente.

Vários outros algoritmos foram desenvolvidos para reduzir o tempo de processamento da DFT. Pode-se citar o algoritmo Split-radix proposto por Yavne (1968), que combina borboletas de 2 e 4 entradas para diminuir o número total de operações aritméticas. Este algoritmo originalmente possuía a desvantagem de necessitar que o tamanho da série de pontos fosse de potência 4, mas permitia ser combinado com qualquer outro algoritmo de FFT para suprir essa desvantagem. Também podem ser citados os algoritmos de Rader (1968), de Bluestein (1968), prime-factor (GOOD, 1958) e de Bruun (1978) que, apesar de não serem analisados neste trabalho, tiram proveito da arquitetura de borboleta (COOLEY; TUKEY, 1965).

2.2 Estrutura da Transformada Rápida de Fourier

A estrutura principal de uma FFT é a borboleta que é constituída por circuitos aritméticos: somadores, subtratores e multiplicadores. Estes circuitos, principalmente os multiplicadores, podem acarretar uma elevada dissipação de potência e ocupar muita área.

Baseado no teorema de Cooley e Tukey (1965), nesta seção será apresentada a estrutura de borboletas da FFT para decimação em tempo (DIT) e decimação da frequência (DIF) para base 2. A estrutura DIT será utilizada para o desenvolvimento de toda a tese.

2.2.1 Borboleta DIT na Base 2

A estrutura de borboleta radix-2, conhecida como base 2 na decimação no tempo (DIT) é apresentada na figura 2.2, com entradas X_0, X_1 e W e saídas Y_0 e Y_1. Todas as entradas e saídas são complexas, ou seja, são compostas por uma parte real (r) e outra imaginária (j):

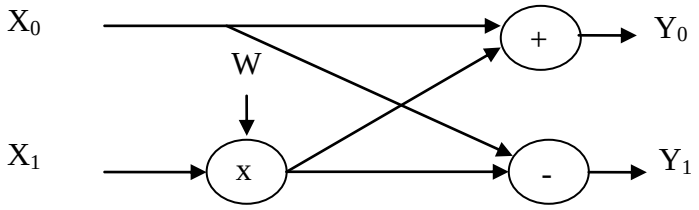


Figura 2.2: Operação borboleta DIT.

$$Y_0 = X_0 + X_1 W = [(X_0 r + jX_0 j) + [(X_1 r + jX_1 j)(Wr + jWj)] \qquad (2.8)$$

$$Y_1 = X_0 - X_1 W = [(X_0 r + jX_0 j) - [(X_1 r + jX_1 j)(Wr + jWj)] \qquad (2.9)$$

Considerando que as entradas X_0, X_1 e a constante W são valores complexos, as operações são determinadas por:

$$Y_0 = X_0 r + (X_1 rWr - X_1 jWj) + j[(X_0 j + (X_1 jWr + X_1 rWj)] \qquad (2.10)$$

$$Y_1 = X_0 r - (X_1 rWr - X_1 jWj) + j[(X_0 j - (X_1 jWr + X_1 rWj)] \qquad (2.11)$$

Pode-se observar pelas equações 2.10 e 2.11 a existência de termos comuns que podem ser reaproveitados no cálculo. Desta forma, na base 2 a borboleta DIT apresenta em sua implementação 4 multiplicadores, 3 somadores e 3 subtratores que, em termos de hardware, devido ao complemento de 2, são implementados como apenas somadores.

A figura 2.3 apresenta a estrutura para a borboleta DIT na base 2.

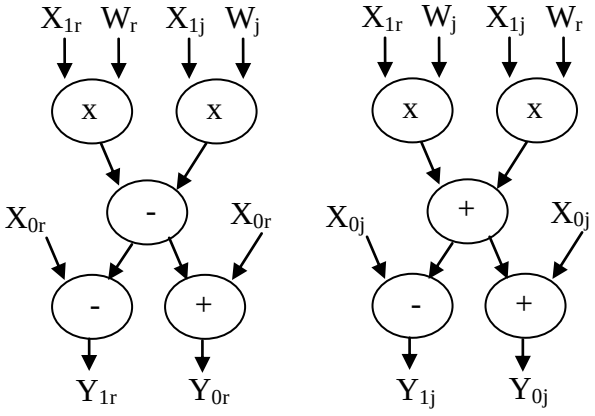


Figura 2.3: Representação dos operadores aritméticos da borboleta DIT.

Pela figura 2.3 observa-se que os coeficientes trigonométricos estão presentes em 4 multiplicações. Decompor estes coeficientes permitirá reduzir o número de operações e, consequentemente, reduzirá o hardware necessário para sua implementação.

2.2.2 Borboleta DIF na Base 2

A estrutura da borboleta na base 2 da FFT com decimação em frequência (DIF) é apresentada na figura 2.3, com entradas X_0, X_1 e W e saídas Y_0 e Y_1 .Todas as entradas e saídas são complexas, ou seja, são compostas por uma parte real (r) e outra imaginária (j):

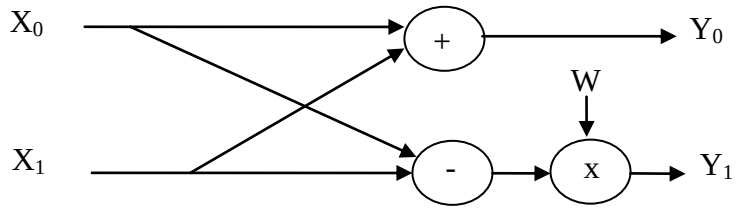


Figura 2.3: Operação borboleta DIF.

$$Y_0 = X_0 + X_1 = (X_0 r + jX_0 j) + (X_1 r - jX_1 j) \quad (2.12)$$

$$Y_1 = (X_0 - X_1)W = [(X_0 r + jX_0 j) - (X_1 r - jX_1 j)](Wr + jWj) \quad (2.13)$$

Pelas equações 2.12 e 2.13 pode-se observar a existência de termos comuns que podem ser reaproveitados no cálculo. Desta forma, na base 2 a borboleta DIF apresenta em sua implementação 4 multiplicadores, 3 somadores e 3 subtratores que, em termos de hardware, devido ao complemento de 2, são implementados como apenas somadores, como a implementação DIT.

2.3 Estrutura de Implementação da FFT

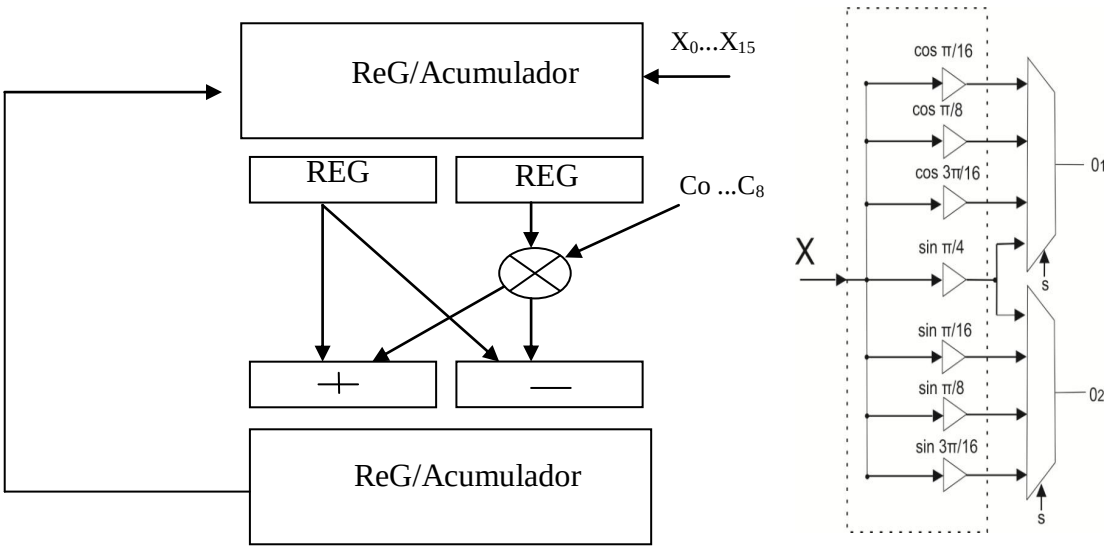
Existem várias formas de implementação de uma arquitetura FFT e nesta seção serão apresentadas três topologias básicas na base 2 que servirão de sustentação do desenvolvimento deste trabalho. A implementação de uma FFT de N pontos, DIF ou DIT, no nível de hardware pode ser realizada por três formas: série, paralela e híbrida. Dependendo da finalidade é implementada a melhor topologia de arquitetura (HAN et al., 2008). Deve-se ressaltar que cada uma dessas topologias pode agregar otimizações estruturais que, em nosso trabalho, baseiam-se na redução de componentes tratadas no capítulo 3.

2.3.1 FFT Serial

A implementação de uma FFT série é realizada através de uma única borboleta. A entrada x dos N pontos da FFT é multiplicada por cada valor trigonométrico correspondente e cada valor obtido é armazenado em um registrador para poder ser manipulado corretamente através da borboleta. A facilidade de sua implementação esbarra na necessidade de ter todas as constantes determinadas e armazenadas para sua correta multiplicação. Estes coeficientes podem ser armazenados em uma ROM ou em um sistema de registradores que permite também manter os coeficientes intermediários de cada operação de multiplicação e de soma/subtração.

A figura 2.4(a) apresenta uma arquitetura FFT DIT em série, sendo a entrada multiplicada separadamente pelo seu valor trigonométrico real e imaginário armazenado na memória. A figura 2.4(b) representa uma FFT série apresentada por (QURESHI;

GUSTAFSSON, 2009) onde é implementada com maior área, pois cada operação é determinada diretamente por cada multiplicador de coeficiente e não armazenada na memória.



a) b)

Figura 2.4: a)Esquemático de uma arquitetura FFT serial; b) FFT-DIT de 16 pontos (QURESHI; GUSTFASSON, 2009)

2.3.2 FFT Paralela

A arquitetura FFT paralela de N pontos realiza todas as operações de uma única vez. Neste tipo de arquitetura o número de borboletas aumenta de forma logarítmica dada pela expressão $\frac{N}{2}\log_2^N$. A arquitetura FFT-DIT de 16 pontos na base 2 totalmente paralela pode ser vista na figura 2.5. Para realizar a transformada de 16 pontos ao mesmo tempo, são necessárias 32 borboletas, que significam 32 multiplicadores para calcular a parte real e outros 32 multiplicadores para parte imaginária, totalizando 64 multiplicadores em cada parte divididos em 4 estágios com 8 borboletas cada (GHISSONI et al., 2010). A implementação em hardware dos componentes multiplicadores dessa arquitetura necessita de aproximadamente 53% da área total, o que acaba em muitos casos inviabilizando a implementação da FFT para vários pontos. Assim, por muitas vezes, mesmo sabendo que a implementação serial acaba provocando o aumento de atraso e dissipação de potência devido à inserção de registradores e ao aumento dos ciclos de relógio necessários para calcular todos os coeficientes, esta acaba sendo escolhida.

A tabela 2.1 apresenta os resultados de implementação de uma FFT comportamental totalmente paralela de 16 pontos que é construída com 128 multiplicadores e 256 somadores/subtratores. A síntese lógica foi realizada com a ferramenta *EncounterRTL Compiler* da CADENCE usando tecnologia UMC 130nm.

Tabela 2.1: Relação de área para somadores/subtratores versus multiplicadores

	Células	**Área(μm^2)**	**Atraso(ns)**
Somadores	3380	0,093	5,1
Multiplicadores	5409	0.104	14,8
Total	8789	0,197	19,9

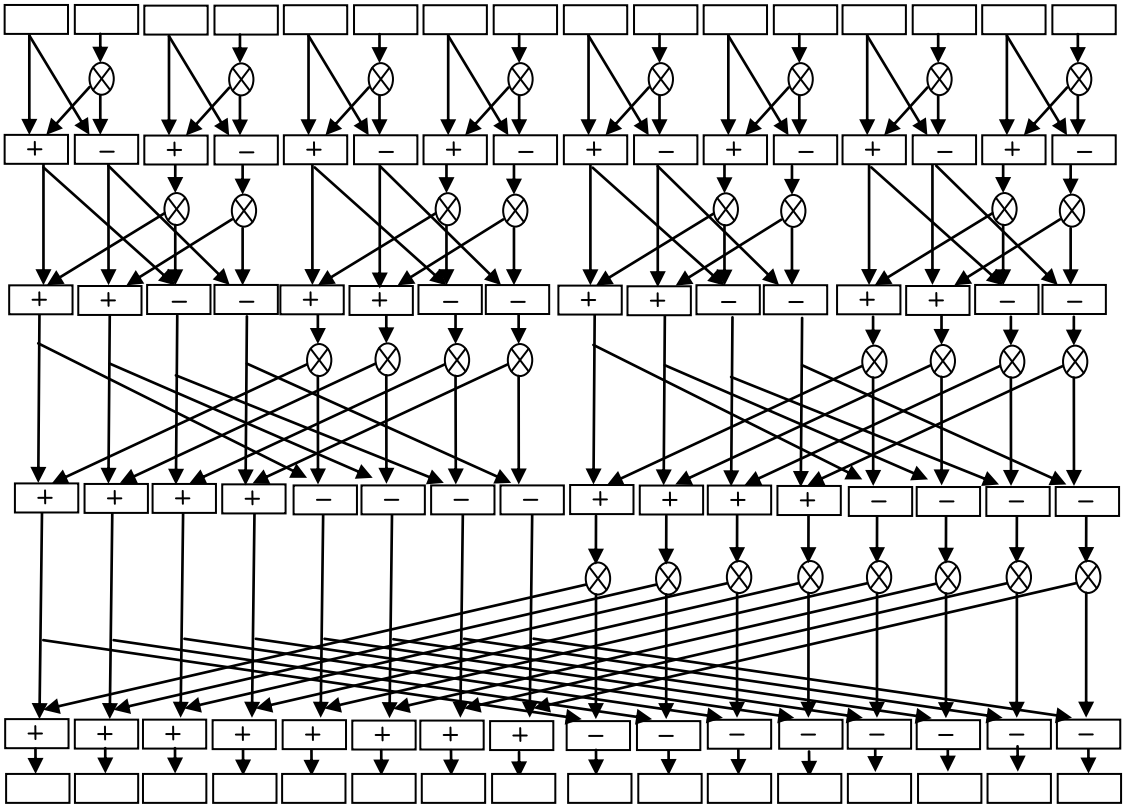


Figura 2.5- Arquitetura paralela FFT-DIT 16 pontos.

No entanto, para arquiteturas FFT de vários pontos, opta-se por implementações híbridas conhecidas como série/paralela. Essas arquiteturas necessitam menor área quando comparadas com implementações totalmente paralelas, mas não são tão rápidas como as paralelas (VACHER; BENKHEBBAB, 1994).

2.3.3 FFT Série-paralela

A arquitetura híbrida é uma associação das topologias série-paralela. Para implementar uma arquitetura FFT híbrida de 16 pontos temos uma parte paralela e outra serial. Por exemplo, uma FFT de 8 pontos totalmente paralela pode reutilizar o hardware 2 vezes para realizar todas as operações reais e complexas necessárias para uma arquitetura FFT de 16 pontos. Isso é possível através de um sistema de controle de acesso dos registradores e de cada respectiva constante (C0...C15) armazenado na memória para realizar cada multiplicação corretamente. Isto permite que as operações sejam realizadas com 12 borboletas, ou seja, 24 somadores/subtratores e 12 multiplicadores; e permite uma redução de hardware quando comparada com uma

arquitetura paralela de 16 pontos, pois precisaria do dobro de elementos. Por outro lado, proporciona um aumento no número de operações necessárias para executar todos os cálculos.

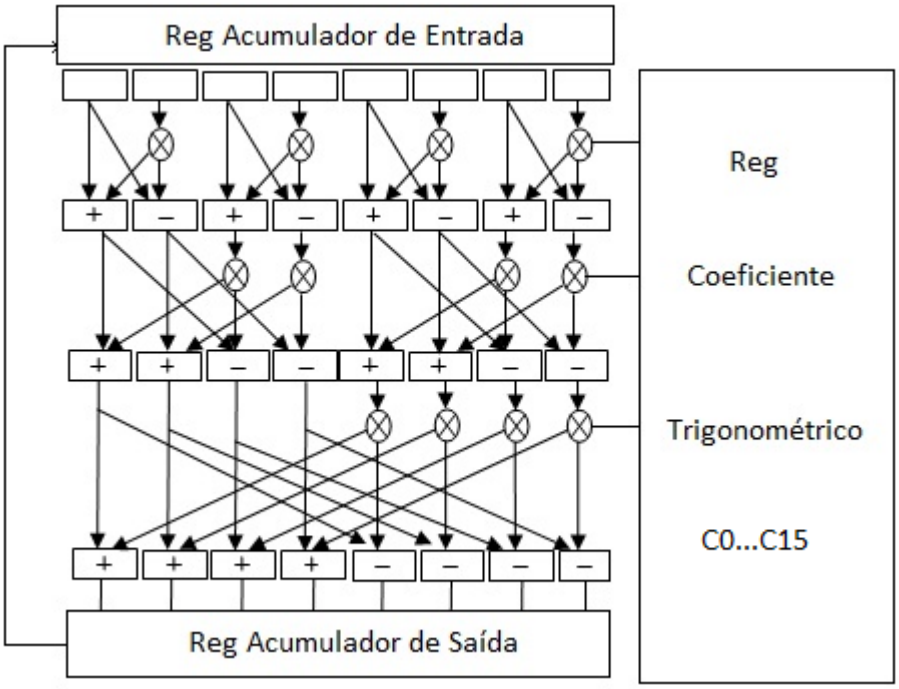


Figura 2.6: Esquemático de uma Arquitetura híbrida FFT-DIT 16 pontos na base 2.

A figura 2.6 apresenta um exemplo de arquitetura híbrida da FFT DIT 16 pontos, onde pode ser observado que há existência de registradores responsáveis para armazenar os valores de entrada e saída, bem como o registrador das operações intermediárias realizadas pela FFT paralela de 8 pontos.

A tabela 2.2 apresenta os dados e os resultados da implementação comportamental da FFT de 16 pontos utilizando as topologias série, paralela e híbrida. A síntese lógica foi realizada com a ferramenta *Encounter RTL Compiler* da CADENCE usando tecnologia UMC 130nm com 1000 vetores aleatórios de entrada.

Tabela 2.2: Relação de implementação de diferentes topologias FFTs 16 pontos

	Potência média total(µw)	**Área(µm^2)**	**Atraso(ps)**
Série	104,1	0,013	18,6
Paralela	86,1	0,197	19,9
Híbrida	83,5	0.124	20,2

Analisando os dados dos exemplos apresentados na tabela 2.2, a arquitetura serial FFT DIT de 16 pontos apresenta menor área. Isso ocorre, devido que esta topologia trata-se simplesmente da multiplicação de cada ponto de entrada pela sua constante trigonométrica correspondente com o armazenamento de cada coeficiente intermediário

em registrador para sua correta manipulação. Apesar de não demandar uma grande quantidade de hardware, é limitada pelo atraso de operação do circuito causado pela multiplicação de todos os pontos. A FFT paralela, embora tenha apresentado a menor potência normalizada por amostra, ocupa maior área entre as arquiteturas. No entanto, após analisar os dados, a arquitetura híbrida apresentou melhor custo benefício, pois apresentou menor área que as demais e a dissipação de potência total esteve próxima a FFT paralela.

2.4 Revisão Bibliográfica de Técnicas Para otimização de FFTs

A busca continuada de novas técnicas que visam melhorar a dissipação de potência e redução de área em arquiteturas FFT fizeram com que, nas últimas décadas, várias pesquisas fossem realizadas, resultando em vários trabalhos para a otimização dessas arquiteturas. Por esse propósito, nesta seção serão abordados alguns dos principais trabalhos referentes à otimização de arquiteturas FFT utilizadas para otimização de área e dissipação de potência.

2.4.1 Técnicas de Base 2 múltiplas e Pipeline

Em Jia *et al.* (1998), foi proposto um novo algoritmo de base-2/4/8, que pode efetivamente reduzir o número de multiplicações complexas de uma arquitetura FFT através da inserção de estágios de *pipeline*. Como o *pipeline* é uma técnica que insere registradores entre estágios de operação, permite que atividades sejam realizadas uma após o término de outra. No entanto, a inserção de registradores acarreta o aumento de área, o que impede a implementação de FFT maiores que 256 pontos.

Na figura 2.7, pode ser observada a implementação da FFT de base 2/4/8, onde os estágios de *pipeline* estão embutidos em cada nível de operação.

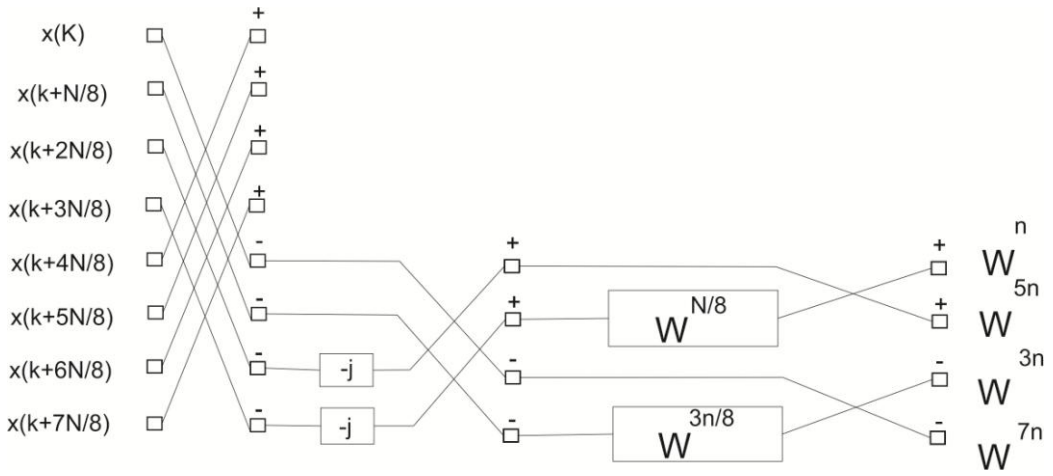


Figura 2.7: Arquitetura FFT do algoritmo de base 2/4/8 com os coeficientes trigonométricos (JIA et al., 1998).

O projeto de Jia *et al.* (1998) faz o uso do algoritmo de Cooley e Tukey (1965) em múltiplas potências na base 2 com auxílio de estágios de *pipeline* após cada multiplicação dos coeficientes W. Desta forma, permite uma redução de multiplicações necessárias para o cálculo da FFT quando comparada com FFT de menor base.

Jing e Lanjuan (2010) apresentaram uma técnica de projeto de arquitetura FFT de tamanho variável. O método permite construir uma arquitetura FFT com várias bases

múltiplas de 2. No trabalho são adotadas base 2, base 4 e base-2/4/8 para lidar com FFTs de 2^n, 4^n e 8^n pontos. As arquiteturas utilizam *pipeline* com *Multiple-Path Delay Commutator* (MDC), onde a sequência de dados é dividida em duas correntes com elementos de *buffers*. Os *buffers* salvam os dados e também fazem o controle entre um intervalo de fluxo de dados. Uma chave de controle determina a direção de dados para a unidade de borboleta, ou para o próximo *buffer*. As arquiteturas utilizam também *Single-Path Delay Feedback* (SDF). Neste tipo de arquitetura os resultados de saída de cada borboleta são armazenados em *buffers shif-register* e a cada ciclo de relógio apenas uma das saídas das borboletas é armazenada no *buffer* e as demais passam paro o próximo estágio.

A figura 2.8 apresenta a arquitetura variável com as diferentes bases propostas por Jing e Lanjuan (2010).

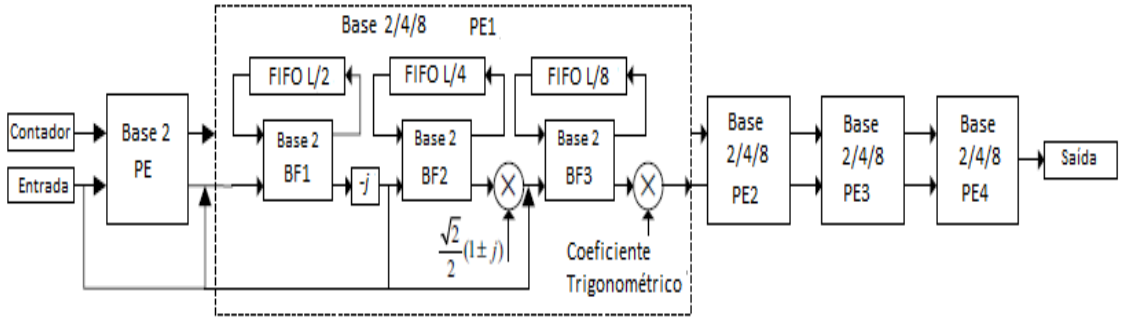


Figura 2.8: Arquitetura FFT de tamanho variável.

Pela figura 2.8 é possível observar o funcionamento da FFT de tamanho variável, onde, dependendo do número de pontos e a aplicação, é utilizado um *pipeline* com diferentes técnicas (SDF e MDC). Para FFTs menores é utilizado SDF, mas para as aplicações de FFTs maiores utilizam-se MDC. No entanto, o uso de cada técnica fica a critério do projetista.

Stevens e Suter (1998) propuseram uma otimização da arquitetura FFT baseada em processamento de diferentes taxas de operação através da tecnologia de circuito assíncrono. Neste trabalho, são aplicados conceitos de processamento de sinal à FFT para localizar e remover a necessidade de resultados globalmente partilhados no cálculo da FFT. Uma arquitetura é implementada a partir de componentes com diversas operações trigonométricas, mapeados para uma implementação assíncrona. O projeto assíncrono de comunicação pode ser elaborado usando as bibliotecas básicas. Esta técnica tem a característica de exigir várias arquiteturas menores de FFTs, tendo um tempo de operação diferente para cada uma, como visto na figura 2.9.

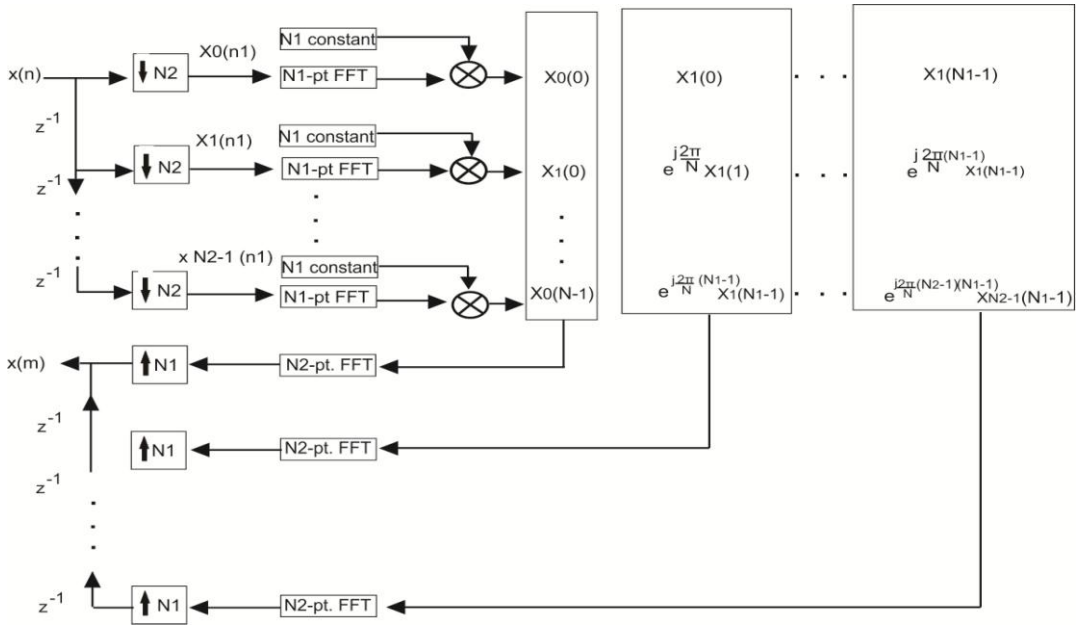


Figura 2.9: Arquitetura FFT (STEVENS, *S*UTER,1998).

A figura 2.9, apresenta o funcionamento da arquitetura FFT de Stevens e Suter (1998) onde cada ponto de entrada x é manipulado pelas várias FFTs de tamanhos menores funcionando em tempos diferentes. Devido à operação ser assíncrona, este tipo de arquitetura necessita de um sistema de controle auxiliar, causando um aumento de área.

Oh e Lim (2004) propõem um trabalho de uma eficiente arquitetura de FFT com uso de pipeline estruturado para sistemas OFDM (*Orthogonal Frequency Division Multiplexing*) com base em um algoritmode base-2^4. A arquitetura de *pipeline* com o novo algoritmo tem o mesmo número de multiplicadores que o algoritmo de base-2^2. Contudo, a complexidade do multiplicador pode ser reduzida a aproximadamente 30%, por meio da substituição da metade dos multiplicadores programáveis com a nova proposta de multiplicadores constantes. A nova proposta de multiplicador complexo constante pode melhorar a área e a dissipação de potência de um projeto FFT. No mesmo ano, Koushik *et al.* (2004) desenvolvem uma FFT OFDM rápida de até 64 pontos para aplicação em redes de acesso local sem fio usando a mesma técnica.

A figura 2.10 apresenta o trabalho desenvolvido por Oh e Lim (2004), onde podem ser observadas três estruturas responsáveis pelas operações: parte real (figura 2.10(a)), imaginária (figura 2.10(b)) e a de adição final vista na figura 2.10(c). Cada estrutura é controlada por um sinal S e seu coeficiente "m" responsável para multiplicação depende do número de estágios. Trata-se de uma estrutura serial onde cada operação de diferentes pontos é armazenada em seus respectivos estágios de *pipeline* BF1 e BF2. Estes *buffers* apresentam apenas um caminho de retorno de dados que pode ser visto na figura 2.11.

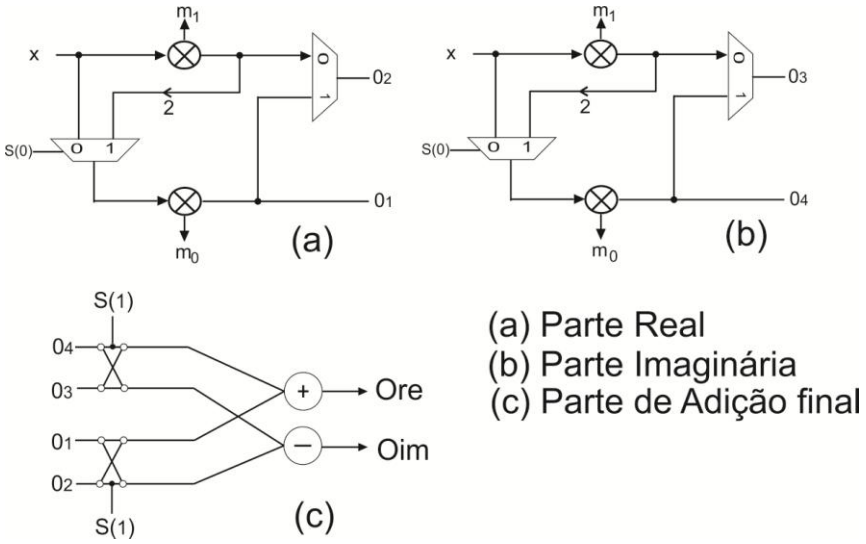


Figura 2.10: Arquitetura FFT com multiplexação dos somadores e multiplicadores (OH; LIM, 2004)

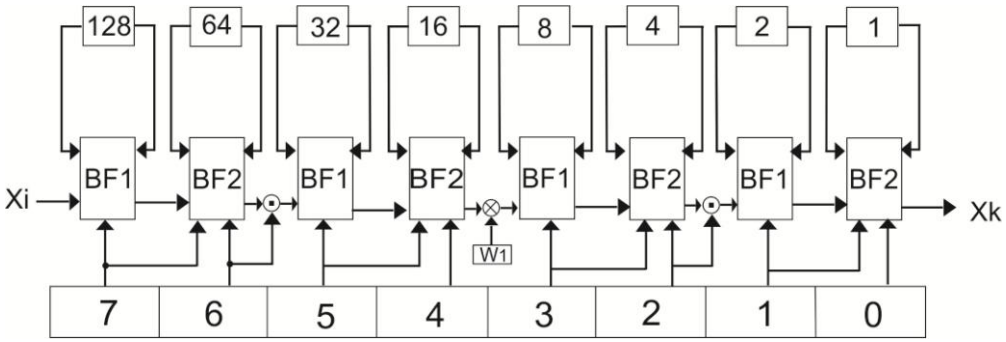


Figura 2.11: Arquitetura FFT R2^{4}SDF (OH; LIM, 2004)

2.4.2 Técnicas Baseadas na Otimização de Coeficientes na Memória

Qureshi *et al.* (2009b) apresentam uma análise de vários algoritmos de base 2^i com o objetivo de investigar qual é a melhor possibilidade de combinação de armazenamento dos coeficientes trigonométricos para cada estágio de *pipeline* da FFT. Este trabalho fornece subsídios para implementar arquiteturas FFTs em FPGAS ou mesmo em ASICS com baixa dissipação de potência. Contudo, para obter as soluções, é necessário verificar os vários algoritmos analisados, o que, dependendo do caso, pode demandar muito tempo. A figura 2.12(a) mostra o esquema da arquitetura com *pipeline* proposta, e na figura 2.12(b) é observado o diagrama dos coeficientes trigonométricos com endereçamento mapeado.

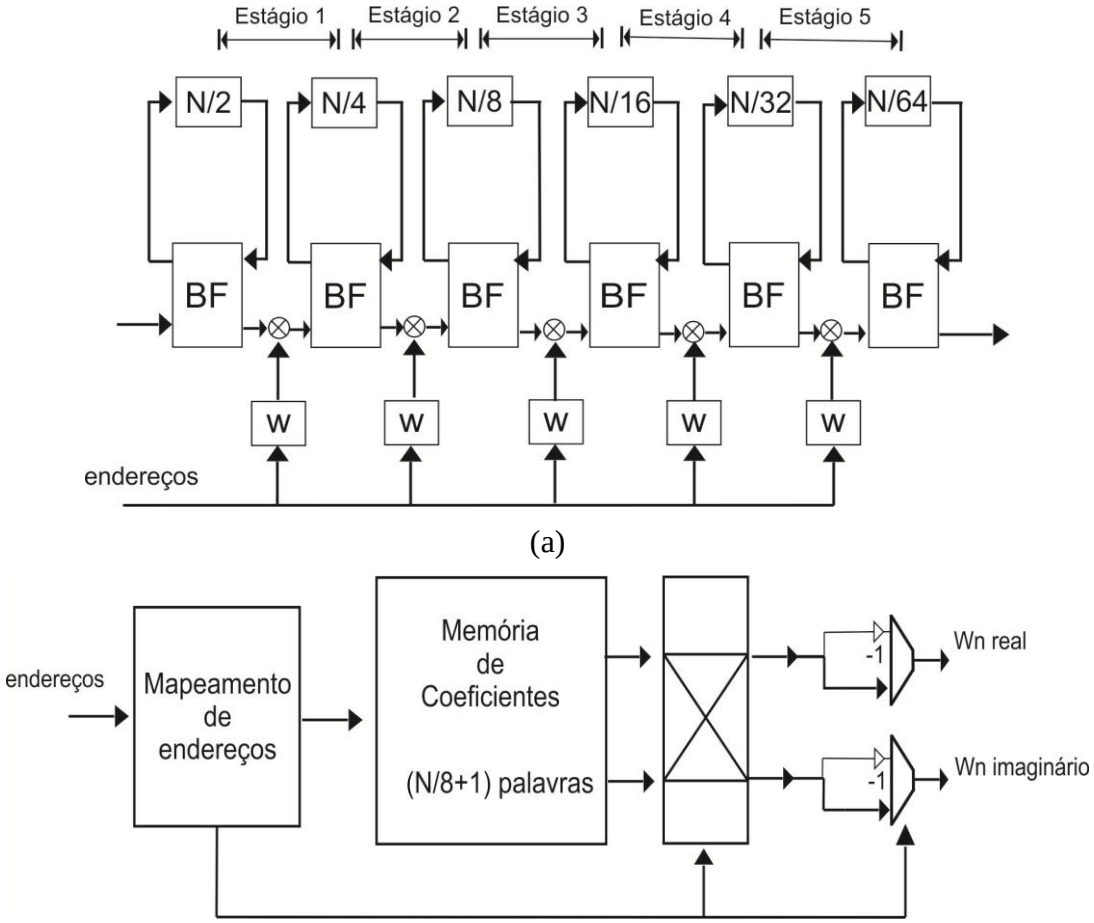


Figura 2.12: Arquitetura FFT com multiplexação dos somadores e multiplicadores (OH et al., 2004)

Xin *et al.* (2009) propõem, para o trabalho de Oh *et al*. (2004), a inserção de registradores e *buffers* extras para reordenar os dados das entradas da unidade da borboleta da figura 2.12(b), evitando, assim, a adição dos dados no módulo de geração de endereços na memória e permitindo a diminuição do caminho crítico para implementações de FFT na base 4.

A preocupação em aumentar o desempenho das operações das arquiteturas FFT que utilizam *pipeline* é também observada nos trabalhos apresentados por Lee *et al.* (2004), Cho *et al.* (2006) e Hasan; Arslan, (2007), onde o foco de redução de atraso e potência é também baseado na otimização do armazenamento dos coeficientes na memória. Nestes trabalhos, a otimização é realizada basicamente na diminuição da memória de armazenamento e na redução das atividades de chaveamento dos endereços de armazenamento dos coeficientes.

2.4.3 Técnicas Baseadas na Otimização de Dissipação de Potência Através do Reordenamento dos Coeficientes Trigonométricos

Hasan *et al.* (2003) propõem a implementação de uma arquitetura FFT de 16 pontos com baixa dissipação de potência através da minimização da atividade de chaveamento dos coeficientes trigonométricos responsáveis pela multiplicação dos operandos na

borboleta de base 4. Os coeficientes são reordenados de forma a minimizar a atividade de chaveamento entre os sucessivos coeficientes trigonométricos através da minimização da menor distância de *Hamming* para cada transição de coeficiente. A distância de *Hamming* é definida como o número de 1s de operações de XOR entre dois coeficientes binários bit a bit. A sequência original dos coeficientes trigonométricos é reordenada através da análise do coeficiente de menor distância de *Hamming* ao de maior. Por exemplo, a figura 2.13 apresenta a distância de *Hamming* dos números decimais 22 e 32 representados em binários, onde podemos constatar que a distância é igual a 4, pois na comparação bit a bit usando portas XORs dos dois números binários obtemos 4 saídas 1s.

$$(22)_{10}=(0\mathbf{1}0\mathbf{11}0)_2$$
$$\updownarrow\updownarrow\updownarrow\updownarrow$$
$$(32)_{10}=(\mathbf{1}00000)_2$$

Figura 2.13: Distância de *Hamming* entre os números decimais 22 e 32 quando representados em binário.

No trabalho de Wu e Fan (2006), é proposta a construção de arquiteturas FFTs com *pipeline* na base 4 para aplicações em sistemas de comunicação Wimax (IEEE 802.16-2004). A técnica consiste em primeiro verificar a menor distância de *Hamming* necessária para multiplicação dos coeficientes da FFT e, após, fazer a implementação da FFT com uma unidade de reordenamento (ROU) nos estágios. A figura 2.14 mostra parte da arquitetura FFT de 256 pontos desenvolvida, em que, no 3° estágio é realizada a leitura dos coeficientes e no 4° estágio na ROU é realizado o reordenamento.

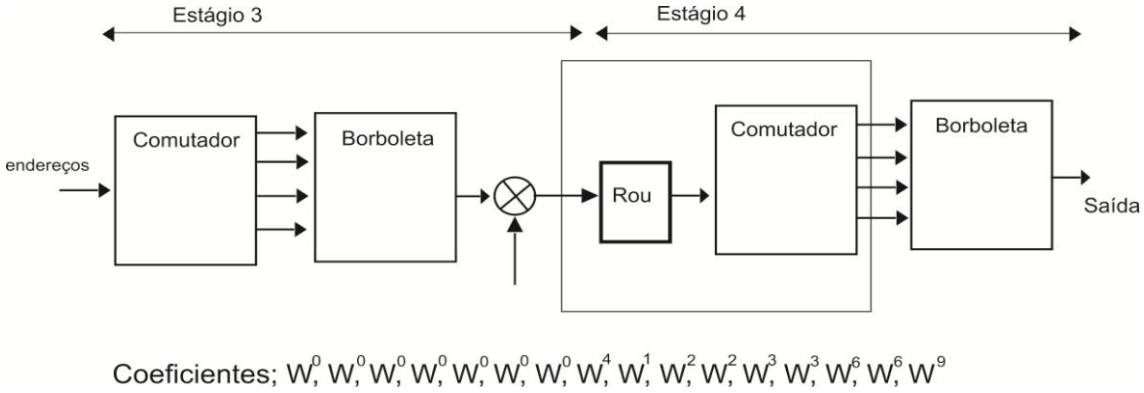


Figura 2.14: Reordenamento do 3° e 4° estágio de uma FFT de 256 pontos na base 4 usando *pipeline* (WU; FAN, 2006)

Os resultados de redução da atividade de chaveamento alcançados no método permitiram uma redução de dissipação de potência. O grande avanço da proposta apresentada por Wu; Fan (2006) em relação à proposta de Hasan *et al.* (2003) foi aplicação para FFT maiores, apesar da ideia ser basicamente a mesma.

Em Luz; Costa; Ghissoni (2012) foram explorados diferentes algoritmos baseados em heurísticas para verificar a melhor manipulação de coeficientes em arquiteturas FFT seriais. O foco principal foi realizar o reordenamento dos coeficientes trigonométricos

com a ferramenta Anedma (LUZ; COSTA; AGUIAR, 2010), que permitiu uma redução média de 10% na dissipação de potência.

A ferramenta Anedma usa uma heurística que avalia a importância de cada coeficiente no conjunto com os seus coeficientes vizinhos. Assim, se um coeficiente tem P candidatos com a mesma distância de *Hamming* para ser escolhido como um vizinho, a escolha deve ser realizada pelo coeficiente de menor importância para os demais vizinhos, tornando assim possível a obtenção de melhores resultados. Nesta heurística, P elementos são iniciados e cada um produz um ordenamento resultante da aplicação do algoritmo utilizando um ponto aleatório como elemento inicial.

Na tabela 2.3 é apresentado um exemplo hipotético com 10 elementos gerados aleatoriamente representando os coeficientes e a distância de *Hamming* entre si. A distância total de *Hamming* com ordenação lexicográfica (E1, ..., E10) é igual a 27.

Tabela 2.3: Exemplo hipotético com 10 coeficientes

Elem.	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**	**E7**	**E8**	**E9**	**E10**
E1	0	2	1	4	9	1	1	3	8	1
E2	2	0	5	9	7	2	3	2	2	5
E3	1	5	0	3	8	6	8	9	2	3
E4	4	9	3	0	2	5	1	6	7	2
E5	9	7	8	2	0	2	5	3	2	8
E6	1	2	6	5	2	0	3	2	4	3
E7	1	3	8	1	5	3	0	7	6	2
E8	3	2	9	6	3	2	7	0	1	6
E9	8	2	2	7	2	4	6	1	0	2
E10	1	5	3	1	8	3	2	6	2	0

A figura 2.15 mostra o reordenamento considerando-se como o elemento inicial E1. O elemento E1 já está inserido em uma lista de elementos bloqueados chamada Ebloq, que agora é EBloq = ([E1]). Esta lista impede que esses elementos sejam selecionados como candidatos de outros fatores. O elemento E1C representa a lista de elementos (coeficientes) candidatos, e E1, E2C representam a lista de candidatos do elemento E2, e assim sucessivamente. A lista parcial Ebloq é mostrada na Fig. 2.15 (b). A lista Ebloq final, com o total de Hamming, está reduzida a 20 e está representada na fig. 2.15 (c).

E1C = ([E3], [E6], [E7], [E10]); E2C = ([E6], [E8], [E9]); E3C = ([E9]); E4C = ([E7]); E5C = ([E4], [E6], [E9]); E6C = ([E2], [E5], [E8]); E7C = ([E4]); E8C = ([E9]); E9C = ([E8]); E10C = ([E4]). **(a)**	E1C = ([E3], [E6], [E10]); E2C = ([E6]); E3C = ([E9]); E4C = ([E7]); E5C = ([E6]); E6C = ([E2], [E5]); E7C = (); E8C = (); E9C = ([E8]); E10C = ([E4]); Finalmente, a lista EBloq é dada por: Ebloq=([E1],[E9],[E7],[E4], [E8]) **(b)**	E1C = ([E3], [E10]); E2C = ([E6]); E3C = ([E9]); E4C = ([E7]); E5C = (); E6C = ([E2], [E5]); E7C = (); E8C = (); E9C = ([E8]); E10C = (E4). Ebloq = ([E1], [E9], [E7], [E4], [E8], [E6]) **(c)**

Figura 2.15: Passos do algoritmo Anedma.

2.4.4 Técnicas Baseadas na Reconfiguração de Múltiplos Blocos (ReMB)

A técnica de reconfiguração de múltiplos blocos (ReMB) foi apresentada por Demirsoy *et al.* (2005). No entanto, uma adaptação para utilização em arquiteturas FFTs implementadas na base 4 foi proposta por Wu; Liu (2009), e permitiu obter baixa dissipação de potência na implementação. Esta técnica verifica qual é o melhor equilíbrio entre a flexibilidade e a dissipação de potência da arquitetura. A diminuição de potência é obtida basicamente utilizando um adequado tamanho do processador FFT, em vez de uma arquitetura de FFT fixa. Uma arquitetura baseada na memória é usada para projetar o processador FFT reconfigurável que pode ser configurado para diferentes tamanhos, variando de 16 a 256 pontos.

Na figura 2.16 é possível observar o fluxo apresentado por Wu; Liu (2009)

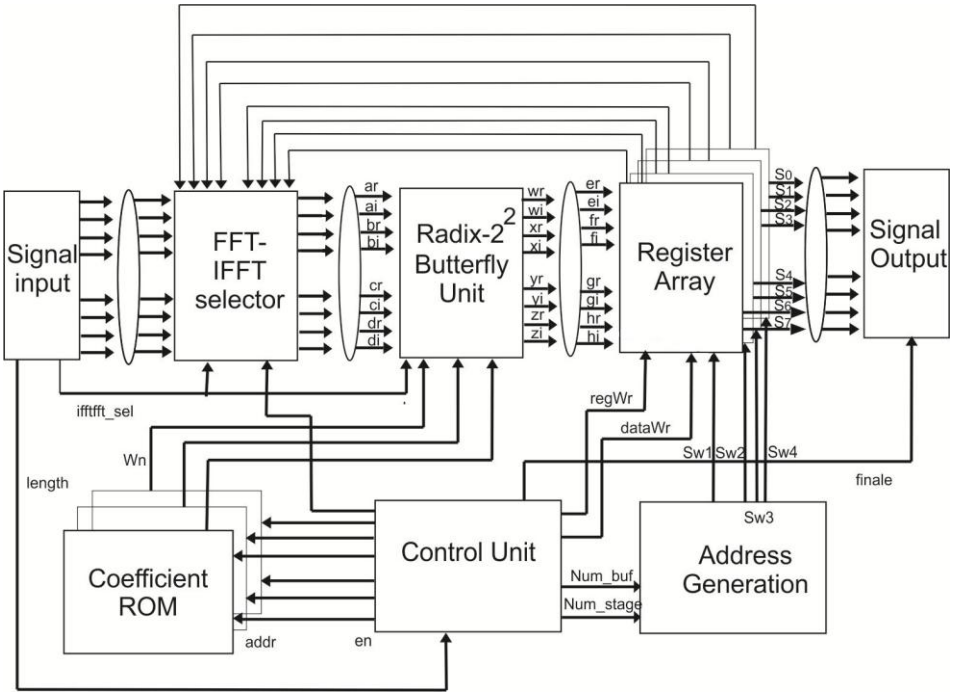


Figura 2.16: Arquitetura do processador configurável (WU; LIU, 2009).

Na figura 2.16, o gerador de endereços é responsável por definir quantos e quais são os endereços disponíveis para as operações da FFT. O número de endereços é definido pelo tamanho da FFT a ser calculada. Quanto menor o número de endereços, menor é a FFT e menor será a dissipação de potência.

2.4.5 Técnicas de Decomposição baseada no uso de Multiplicação de Constantes e de Coeficientes Trigonométricos

Uma recente linha de pesquisa para otimização de arquiteturas FFTs é baseada na decomposição de coeficientes trigonométricos responsáveis pelas operações de multiplicação nas borboletas. O grande diferencial é o uso de técnicas de multiplicação de constantes que será utilizada nesta tese. Antes de apresentar alguns trabalhos referentes à decomposição de coeficientes, será apresentada uma breve revisão sobre as técnicas de multiplicação de constantes.

2.4.5.1 Multiplicação de constante única (SCM)

A multiplicação por uma constante é usada em várias aplicações, como filtros, aritmética de precisão, criptografia, etc. A multiplicação de uma variável por uma constante pode ser decomposta através de adições, subtrações e deslocamentos. O problema de encontrar a decomposição usando o mínimo número de operações considerando o compartilhamento de adições/subtrações é chamado de problema de multiplicação de uma constante ou *Single Constant Multiplication* (SCM), e é proveniente de *Nondeterministic Polynomial Time* – NP de Cappello *et al.* (1984). A SCM é similar ao problema apresentado por Gustafsson (2007a), em que a multiplicação das constantes é realizada usando apenas adições e deslocamentos, onde a equação 2.14 permite determinar o número mínimo de operações (OPSCM) necessárias para implementar uma SCM.

$$OP_{SCM} = [\log_2 S(t_1)] \tag{2.14}$$

Onde $S(t_1)$ é o mínimo número de dígitos não zeros necessários para representar o elemento t_1 quando é definida nos termos da representação CSD (*Canonic Signed Digit*).

A representação CSD é um sistema de dígitos com sinal de base 2, tendo o conjunto de dígitos {1,0,-1} ou {1,0,2}, pois -1 é representado por 2, mas a representação mais utilizada é $\bar{1}$. Dado um valor inteiro, a representação CSD correspondente é única e possui duas propriedades (PARK et al., 2001):

- **Primeira: é que apresenta o mínimo de dígitos não zeros;**
- **Segunda: é que o produto de dois dígitos adjacentes é zero, ou seja, dois dígitos não zeros não são adjacentes.**

Baseada na representação CSD, a solução do problema SCM não pode fornecer um número inferior de operações que o próprio limite inferior existente.

Os algoritmos projetados para o problema de SCM podem classificados em três categorias:

- Métodos baseado em dígitos;
- Algoritmos de eliminação de subexpressão comum (CSE);
- Algoritmos baseados em Grafos.

O método baseado em dígitos define a constante em uma representação de um número particular e realiza a implementação do multiplicador da multiplicação da constante de sua representação. Este método é mais rápido, por apresentar uma complexidade computacional linear no número de dígitos na representação da constante. Desta forma, a multiplicação de uma constante com vários dígitos por uma variável pode ser facilmente implementada. Todavia, este método tem o pior desempenho, pois a sua solução é geralmente longe da aplicação mínima. Por exemplo, supõe-se que o número 174 seja multiplicado pela variável x e a constante seja representada na forma binária. Assim, a aplicação de 174x é:

$$174x = 0010101110x = x_{<<7} + x_{<<5} + x_{<<3} + x_{<<2} + x_{<<1} \tag{2.15}$$

Esta representação requer quatro operações de adição.

No entanto, quando a constante é definida em representação CSD, requer apenas três operações de adição/subtração:

$$174x = 010\bar{1}0\bar{1}00\bar{1}0x = x_{<<8} - x_{<<6} - x_{<<4} - x_{<<1} \tag{2.16}$$

A partilha dos produtos parciais entre a multiplicação constante tem um significativo impacto na redução do número de operações. Os algoritmos CSE basicamente encontram os padrões mais comuns sobre a representação das constantes. A heurística CSE apresentada por Lefevre (2001) é destinada para o problema de SCM no pior caso com a complexidade polinomial e pode ser usada para encontrar a solução do problema de SCM incluindo as constantes de grande porte, por exemplo, 32 bits ou 64 bits. Além disso, o algoritmo de Dimitrov (2007) inicialmente representa o número de constantes no sistema em base dupla e, em seguida, encontra uma solução através da partilha de produtos parciais.

Entretanto, as soluções desses algoritmos dependem da representação do número. Assim, o número mínimo de operações da solução do problema de SCM não pode ser garantido por estes algoritmos, embora a constante seja representada usando um número mínimo de dígitos diferentes de zero e a partilha de possíveis produtos parciais seja utilizada. Por outro lado, os algoritmos baseados em grafos não são restritos a um número de representação e consideram a constante em seu valor decimal. Os algoritmos baseados em grafossintetizam a constante através da construção de um grafo onde os vértices são rotulados com constantes e as arestas estão marcadas com o sinal e turnos.

O algoritmo baseado em grafos exato (DEMPSTER; MACLEOD, 1994) proposto para o problema de SCM, inicialmente, encontra todas as topologias possíveis, que incluem o grafo. Assim, o mínimo número de operações constantes para implementações de até 12 bits é encontrado atribuindo as constantes intermediárias aos nós das redes de forma exaustiva. O método descrito em Gustafsson *et al.* (2002) introduz simplificações nas topologias do grafo e estende o algoritmo exato de Dempster; MacLeod (1995a-b) considerando todas as possíveis implementações de no máximo cinco operações. Assim, para as constantes de até 19 bits, o mínimo número de soluções de operações é obtido. No entanto, o algoritmo exato requer um elevado esforço computacional, bem como fontes de memória devido a sua exatidão.

A representação SCM não é utilizada para a otimização de FFTs devido ao aumento de atraso causado pelo aumento de profundidade lógica do circuito, mas o seu conceito é essencial para o desenvolvimento das demais técnicas.

2.4.5.2 Multiplicação de constantes Múltiplas (MCM)

Uma evolução do problema SCM deve-se ao fato de multiplicar uma variável por um conjunto de constantes em paralelo. A execução de multiplicações de constantes múltiplas usando o número mínimo de operações de adição/subtrações é conhecida como o problema de Multiplicação de Constantes Múltiplas (MCM). O problema MCM é a generalização do problema de SCM, que também é abordado como NP (CAPELLO, 1984). O problema de MCM é encontrado nas aplicações de Filtros digitais FIR, FFT, processamento de imagem e aritmética computacional.

No problema MCM a representação utilizada é a forma mínima de dígitos (MSD). Nesta representação, o sistema de dígitos usado é o mesmo que na representação CSD. No entanto, apenas a primeira propriedade CSD é respeitada (PARK et al., 2001).

Na figura 2.17 o exemplo CSD para a constante 174, equação 2.16, é apresentado em representação MSD.

$$174x = \begin{bmatrix} 010\bar{1}0\bar{1}00\bar{1}0x \\ 00110\bar{1}00\bar{1}0x \\ 0010110\bar{0}\bar{1}0x \end{bmatrix}$$

Figura 2.17: Representação da constante 174 em MSD.

É possível observar pela figura 2.17 três possibilidades de representação da constante 174 em representação MSD, ambas com três operações de soma/subtração. A representação MSD aumenta o número de formas de representações para a mesma

constante com o mesmo número de operações. Assim, aumenta a possibilidade de compartilhamento entre as várias constantes existentes MCM (AKSOY et al., 2008).

O limite inferior do número mínimo de operações necessárias à execução MCM é também analisado em Gustafsson (2007a) e é dado pela equação 2.17.

$$OP_{MCM} = [\log_2 S(t_i)] + \sum_{i=1}^{m-1} E(S(t_i), S(t_{i+1})) \quad (2.17)$$

Onde $E(S(ti);S(ti+1))$ é calculado pelas condições a seguir:

$$E(S(t_i), S(t_{i+1})) = \begin{cases} 1, S(t_i) = S(t_{i+1}) \\ [\log_2 S(t_{i+1}) / S(t_i))], S(t_i) < S(t_{i+1}) \end{cases} \quad (2.18)$$

A equação 2.18, S(ti)<S(ti+1), no cálculo de E(S(ti), S(ti+1)) indica que não é possível calcular a constante com S(ti+1) dígitos não zero usando apenas uma operação adicional, se houver apenas constantes com no máximo S(ti) dígitos não zeros disponíveis. Assim, tendo em conta este caso, o limite inferior pode ser aumentado. Novamente, nota-se que o dado limite inferior indica que a solução do problema MCM não pode incluir o número de operações menor que o vinculado.

Para obter a solução de um problema MCM, pode ser aplicado algum dos algoritmos de SCM em cada constante alvo do problema, sem MCM, tendo em conta a troca de produtos parciais em multiplicações constantes. Como exemplo, a multiplicação das constantes múltiplas de 11 e 17 pela variável x, dada na figura 2.18(a). Observa-se na figura 2.18(b) que a operação de multiplicação realizada sem partilha parcial do produto requer quatro operações de adição. No entanto, a partilha parcial do produto de 9x em ambas as multiplicações reduz o número de operações necessárias para três, conforme ilustrado na figura 2.18(c).

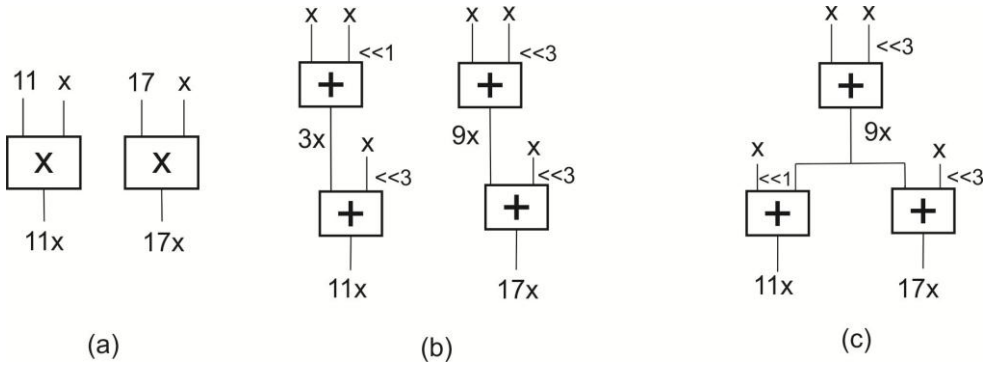


Figura 2.18: a) Multiplicação de Constantes Múltiplas. Implementação por adições e deslocamentos: b) Sem compartilhamento do produto parcial: c) Com compartilhamento do produto parcial.

A partilha do produto parcial do número de operações necessárias MCM é investigada pelo algoritmo exato destinado para o problema SCM (GUSTAFSSON, 2002) sem considerar a partilha de produtos parciais. Sendo os resultados encontrados pelo algoritmo exato (AKSOY et al., 2008), para o problema de MCM possibilita a partilha do produto parcial e reduz significativamente o número de operações necessárias para implementar a função.

2.4.5.3 Multiplicação de Matrizes de Constantes (CMM)

O problema CMM é uma versão mais genérica do problema MCM, onde a multiplicação das constantes múltiplas é realizada com apenas uma variável de entrada. O problema CMM consiste na multiplicação de uma matriz *mxn*, sendo *mx* a constante e nx_1 o vetor de entrada X. A matriz constante especifica como o vetor de saída Y = AX é obtido a partir das transformações lineares das entradas, visto na figura 2.19. Este problema é definido para encontrar o número mínimo de adições e subtrações que implementam as transformações lineares. Como o problema CMM trata de transformações lineares, é possível utilizar esta técnica para tentar reduzir o número de operações da FFT, pois o resultado de saída de uma FFT é obtido a partir das transformações lineares de suas entradas. Assim, algoritmos projetados para o problema MCM podem ser estendidos para lidar com o problema CMM, onde cada constante e variável de entrada na MCM é substituído por um vetor constante e um vetor de entrada, respectivamente.

$$\begin{pmatrix} a_{11}\ a_{12}\ \dots\ a_{1n} \\ a_{21}\ a_{22}\ \dots\ a_{2n} \\ \vdots \\ a_{m1}\ a_{m2}\ \dots\ a_{mn} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n \end{pmatrix}$$

Figura 2.19: Representação de Multiplicação de Matrizes de Constantes.

O algoritmo apresentado por Aksoy *et al.* (2012) possibilita avaliar o problema CMM e seu efeito de aplicação em termos de componentes somadores/subtratores. Para implementar todas as equações lineares, o problema CMM utiliza a representação MSD e CSD para cada constante e analisa o custo de compartilhamento entre as demais constantes geradas. A desvantagem do método é a necessidade de analisar todas as possibilidades de combinações entre os conjuntos gerados e escolher a que tiver o menor custo. Entretanto, o método permite uma redução considerável de componentes.

Um exemplo de aplicação do algoritmo CMM (AKSOY et al., 2012) pode ser visto na figura 2.20..

$$\begin{pmatrix} 17 & 24 \\ 9 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \qquad \begin{aligned} y_1 &= 17x_1 + 24x_2 \\ y_2 &= 9x_1 + 3x_2 \end{aligned}$$

Figura 2.20: Exemplo de aplicação de CMM.

Primeiramente cada constante é gerada a partir da representação MSD. A seguir, cada combinação linear da representação MSD de uma constante é associada à outra, e assim sucessivamente.

O algoritmo CMM tenta implementar os coeficientes de saída realizando compartilhamento entre as entradas envolvidas, cujo resultado do custo dos blocos pode ser visto na Figura 2.21.

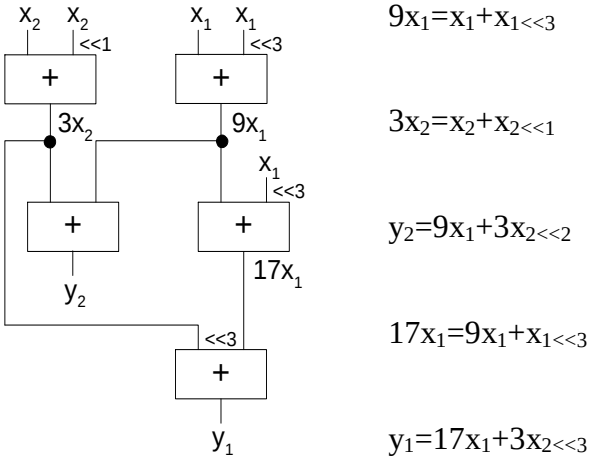


Figura 2.21: Representação da combinação linear com suas subexpressoes.

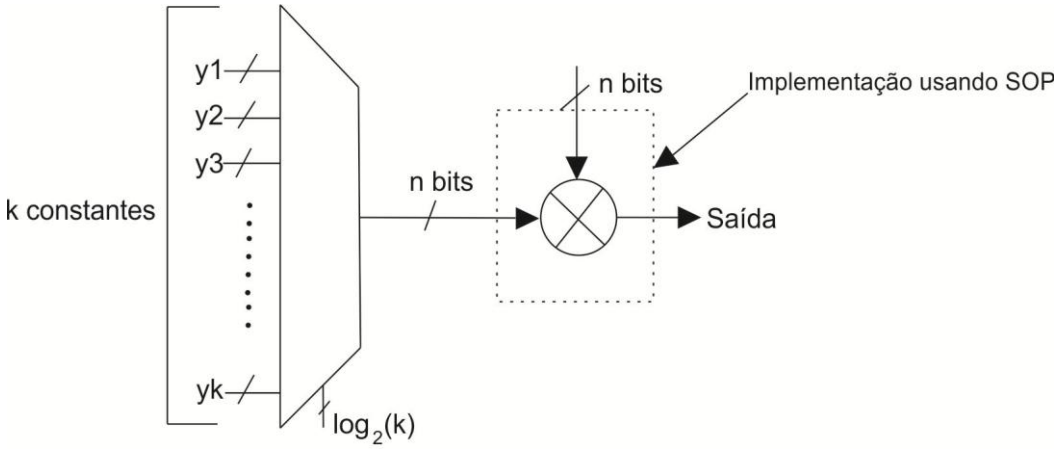
Na implementação das equações lineares da figura 2.21 usando CMM são necessários apenas 5 somadores. Embora exista uma série de outros algoritmos CMM, o algoritmo de Aksoy *et al.* (2012) apresentou melhores resultados de redução de componentes somadores/subtratores e será usado como base para o desenvolvimento e comparação no trabalho.

2.4.5.4 Aplicações de Decomposição de constantes em implementações FFT

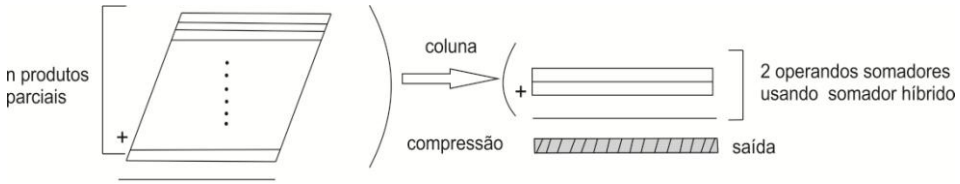
No trabalho de Macleod (2005) são apresentados métodos para implementar a multiplicação de FFTs através da rotação em ponto fixo. As multiplicações são substituídas puramente pelas adições, subtrações e deslocamento. Estes métodos permitem minimizar o custo de implementação, reduzindo o número de adições e subtrações, com um determinado nível de precisão. Projetos baseados nessas técnicas são comparados com os projetos com base na aplicação da DFT de multiplicação de matrizes. Tais técnicas permitem possíveis implementações VLSI dos rotadores e as arquiteturas FFTs podem atingir uma velocidade muito alta de processamento. Os métodos podem ser usados para fornecer qualquer precisão escolhida entre uma faixa de 12 a 26 bits de precisão. Em média, os roteadores são mostrados para serem implementáveis com 10, 12 ou 15 somadores para atingir precisões de 12, 16 ou 20 bits, respectivamente.

O emprego de MCM em otimização de FFTs é também visto em Karkala *et al.* (2010), onde o método aritmético baseado em representação de soma de produtos (SOP), visto na figura 2.22, otimiza os coeficientes parciais. O algoritmo baseado em SOP, que utiliza termos parciais Max-SAT (PMSAT) (LIN et al., 2008) para otimizar ainda mais o SOP é chamado de algoritmo melhorado. As tentativas do algoritmo melhorado para reduzir o número de produtos parciais de SOP são realizadas por deslocamentos de coeficientes para encontrar outros coeficientes quando possível. O algoritmo explora múltiplas implementações de cada coeficiente usando um formato

MSD (PARK et al., 2001) e a valorização da exclusividade mútua dentro de certos grupos de produtos parciais. O algoritmo para gerar as implementações de Hardware da Transformada Rápida de Fourier (FFT) requer os dados de entrada e deve ser multiplicado por um dos vários coeficientes constantes. O método mostra que suas abordagens apresentam uma melhoria significativa na área e atraso, quando comparadas com Reconfiguração de Hardware e as simulações dos principais métodos apresentados por Das (2007). Nesses métodos foram aplicadas diferentes técnicas, para a síntese da aritmética do fluxo de dados, sendo que a chave para o desempenho eficiente de qualquer bloco aritmético sintetizado foi projetar uma arquitetura apropriada no contexto do circuito e das restrições de tempo exigidas.



(a) Arquitetura MCM



(b) SOP baseado em MCM

Figura 2.22: Soma de produtos baseado em MCM para FFT (KARKALA et al., 2010).

Na figura 2.22, os produtos parciais dos produtos de soma que representam cada saída da FFT são otimizados através do compartilhamento por MCM. O menor SOP após o uso de MCM é escolhido para implementação.

Soluções baseadas em arquiteturas paralelas através da utilização de vários núcleos para aumento de velocidade das arquiteturas FFTs foram exploradas juntamente com diferentes combinações de técnicas híbridas de baixa potência (HAN et al., 2008). As técnicas utilizadas visam o uso de unidades, que substituem os multiplicadores complexos nas borboletas das arquiteturas FFTs, através da utilização de comutadores de baixa potência, que se baseiam no uso de *pipeline*, bem como da utilização de arquitetura paralela. O método faz uso de MCM para compartilhar os coeficientes trigonométricos dos diversos multiplicadores que se encontram no mesmo estágio das arquiteturas híbridas, assim reduz a dissipação de potência. Para o correto funcionamento, os comutadores são controlados por um sistema de controle de sinais.

No entanto, este método otimiza somente borboletas da FFT que estão no mesmo estágio e, desta forma, acaba acarretando um considerável atraso devido à comutação entre os vários núcleos utilizados. A figura 2.23 apresenta um esquemático do projeto da FFT R4SDC 16 pontos proposto no estudo.

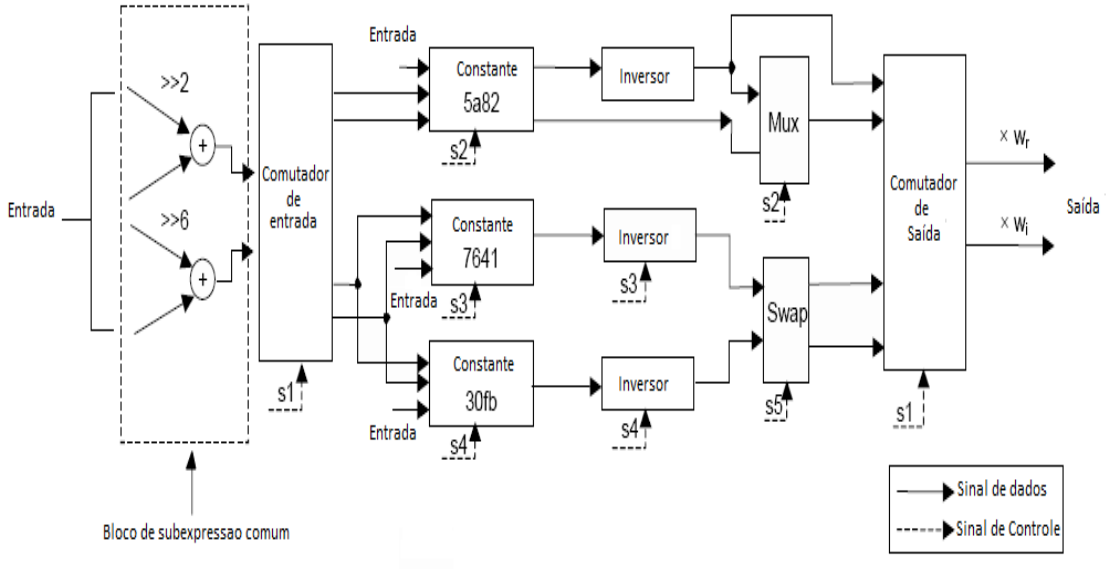
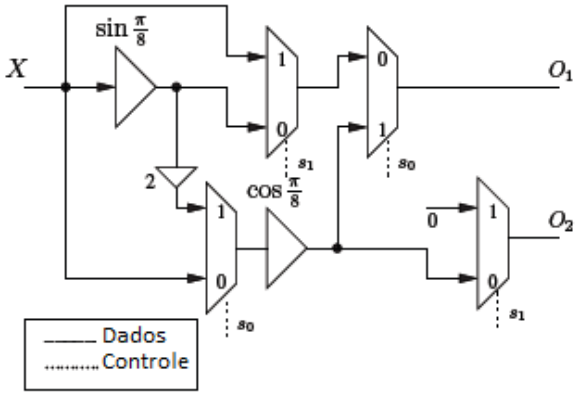


Figura 2.23: Operação da FFT R4SDC 16 pontos (HAN et al., 2008).

Na figura 2.23 a partilha dos coeficientes ocorre em cada estágio da FFT onde estão representadas em formato hexadecimal. Este trabalho servirá de comparação do método de decomposição de coeficientes que será proposto no capítulo 4.

2.4.5.5 Técnicas de Decomposição de Coeficientes Trigonométricos

Além do uso de MCM para otimização de coeficientes trigonométricos pode ser aproveitado à própria identidade trigonométrica da matemática. Pensando nisto, Oh e Lim (2005) utilizaram a matemática da identidade trigonométrica, $sen2\theta = 2sen\theta\cos\theta$, na implementação de arquiteturas FFT de 8 pontos. A metodologia apresentada sugere a substituição dos somadores do circuito pelo uso de multiplexadores controlados que seguem o caminho da matemática trigonométrica. A multiplicação das constantes propostas pode ser observada na figura 2.24.



s_0	s_1	O_1	O_2
0	1	1	0
0	0	$\sin\frac{\pi}{8}$	$\cos\frac{\pi}{8}$
1	0	$\sin\frac{\pi}{4}$	$\cos\frac{\pi}{4}$

a) b)

Figura 2.24: a) Multiplicação das constantes com multiplexadores: b)Sinais de controle dos multiplexadores (OH et al., 2005).

Na figura 2.24(a) cada operação de multiplicação da FFT real e imaginária é determinada pelos caminhos de dados traçados pelos multiplexadores controlados através dos sinais s_0 e s_1, vistos na figura 2.24(b).

De certa forma, o trabalho de Oh *et al.* (2005) é um avanço da técnica apresentada na seção 2.4.1 para arquiteturas OFDM, pois apresenta uma solução para FFTs de 32 pontos através de controle de multiplexadores em projetos com *pipeline* (OH et al., 2004).

Neste contexto de otimização de área através da identidade trigonométrica, Qureshi e Gustafsson (2009) propõem uma redução na complexidade para multiplicação complexa através do uso de MCM entre as multiplicações trigonométricas para FFTs de até 32 pontos. A figura 2.25 representa o conjunto de coeficientes trigonométricos implementados por MCM sendo controlados por multiplexadores. Esses coeficientes são implementados através de compartilhamentos parciais, vistos na seção 2.4.5.2, o que permite uma redução do número de componentes e de área. Entretanto, esta metodologia acaba acarretando aumento de atraso para arquiteturas maiores de 32 pontos, em virtude da profundidade lógica requerida.

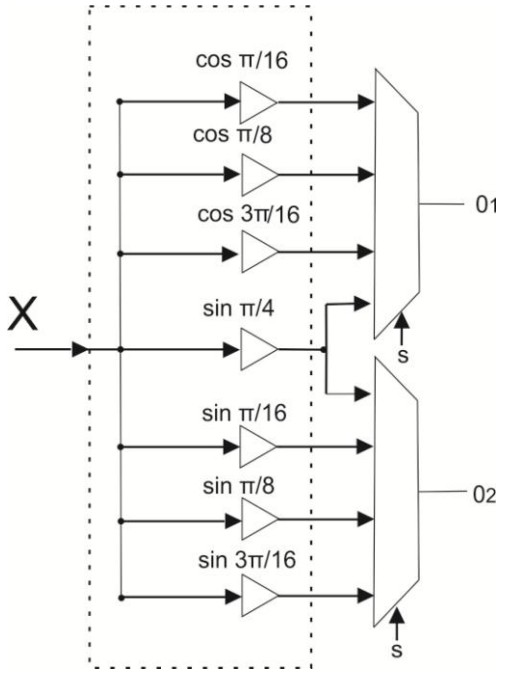


Figura 2.25: Multiplicação de constantes Múltiplas baseada na multiplexação das constantes complexas da arquitetura FFT (QURESHI, GUSTFSSON, 2009).

Por reduzirem a complexidade das arquiteturas FFTs os trabalhos de Qureshi (2009) e Oh (2005) proporcionaram um avanço à técnica de reconfiguração de Multiplexadores utilizada na implementação de operações com coeficientes fixos em operações DSPs proposta por TURNER *et al.* (2001). A técnica de reconfiguração até então operava os valores através da associação de valores na base 2 somados e multiplicados pelos caminhos determinados pelos multiplexadores, como pode ser visto na figura 2.26.

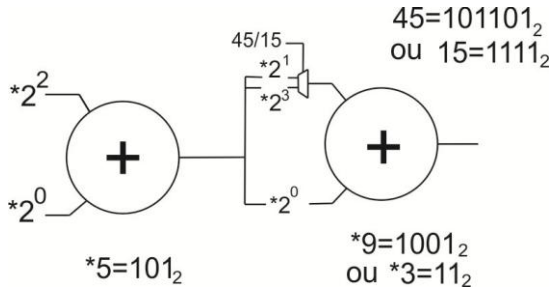


Figura 2.26: Multiplicação de um valor por 45 ou 15.

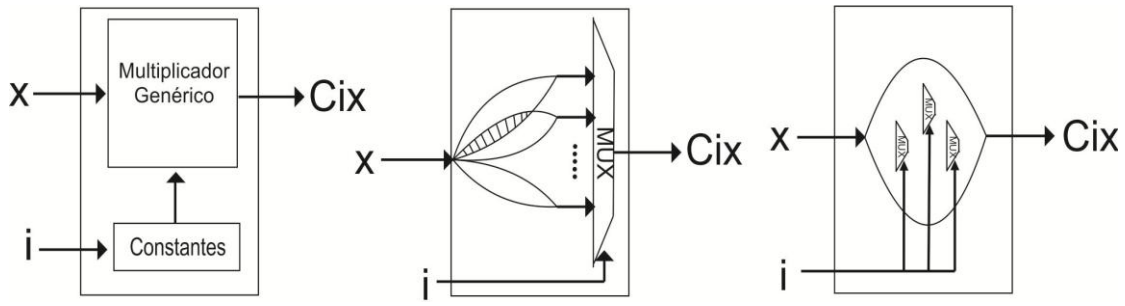
A figura 2.26 apresenta um exemplo de multiplicação de um valor por 45 ou 15. O valor de entrada é multiplicado por 2^2 e somado pela multiplicação de 2^0; a seguir, o resultado é multiplicado por 2^0 e somado com 2^3 para obter 45 ou somado por 2^1 para obter 15. A grande desvantagem do método apresentado por Turner *et al.* (2001) é o aumento da profundidade lógica do circuito.

Usando a ideia de grafos acíclicos direcionados (DAG), Tummeltshammer *et al.* (2007) apresentam um trabalho de Multiplexação no tempo para o problema MCM. Neste trabalho, é analisado o efeito na área causado na implementação de circuitos aritméticos quando são multiplicados por um valor de entrada de ponto fixo seletivamente escolhido por uma das várias constantes de ponto fixo existentes. O

algoritmo apresentado gera uma classe de soluções baseadas na associação de multiplexadores que permitem multiplicações pela multiplexação das múltiplas constantes de multiplicação presentes em tabelas de circuitos, como visto na figura 2.27(a), e uso de associações de multiplexadores paralelos e sequenciais, como visto nas figuras 2.27(b) e (c) respectivamente.

A avaliação compara a solução com um estilo de execução de base, que emprega um multiplicador completo e uma tabela para as constantes. A análise mostra que o método consegue uma vantagem significativa em área, ao preço do aumento da latência, para tamanhos de problema (em termos do número de constantes) que depende da largura de bits da entrada e das constantes. O trabalho mostra ainda que a solução é mais adequada para as células básicas específicas de circuitos integrados construídos sobre blocos de multiplicadores reconfiguráveis (ReMB).

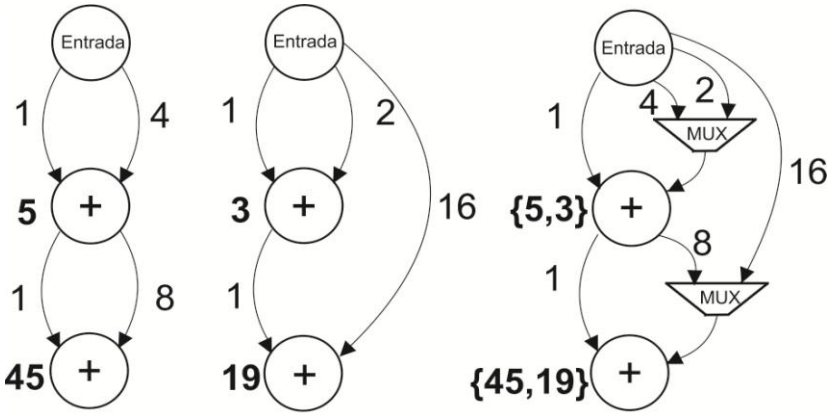
Uso de DAG, pode gerar três diferentes implementações:



(a) (b) (c)

Figura 2.27: Bloco do multiplicador controlado por multiplexador de constantes. a) Multiplicador genérico b) multiplicador paralelo; c) multiplicador através de compartilhamento de somadores e multiplexadores (TUMMELTSHAMMER et al., 2007).

A figura 2.28 apresenta um exemplo para obter os valores 9 e 45 usando o DAG (TUMMELTSHAMMER et al., 2007). Na figura 2.27(a) e (b) os valores são obtidos separadamente através da relação das arestas, e em (c) os mesmos valores são obtidos através de compartilhamento dos componentes somadores e multiplexadores. Esta técnica apresentada para multiplicação através de compartilhamento de somadores e multiplexadores também será utilizada no desenvolvimento da proposta de tese.



(a) (b) (c)

Figura 2.28: a) DAG ótimo para constante 45; b) Para a constante 19; c) multiplicador paralelo; c) multiplicador através de compartilhamento de somadores e multiplexadores (TUMMELTSHAMMER et al.,2007).

Qureshi e Gustfasson (2010) apresentaram um algoritmo para decomposição de coeficientes trigonométricos de FFTs de até 4096 pontos para implementações em FPGA usando *pipeline*. O algoritmo realiza a otimização dos coeficientes através da árvore binária relativa à atividade de chaveamento das operações de multiplicação correspondentes a FFT associada. Cada nó da árvore representa um coeficiente trigonométrico e a solução de otimização é representada pela equação 2.19.

$$X\left[QK_1 + K_2\right] = \sum_{n_1=0}^{P-1}\left[\left(\sum_{n_2=0}^{Q-1} x\left[n_1 + Pn_2\right] W_Q^{n_2 k_2}\right) W_M^{n_1 k_2}\right] W_P^{n_1 k_1} \qquad (2.19)$$

$$0 \le n_1, k_1 \le P-1; 0 \le n_2, k_2 \le Q-1$$

M, P e Q são considerados potências de 2, isto é N = $^{2p+q}$ e Q = 2^q onde *p* e *q* são números positivos inteiros. Assim, a DFT de N pontos é decomposta em DFTs Q e P de N pontos cada. Tipicamente as novas DFTs são divididas novamente em DFTs menores conforme a atividade de chaveamento.

A figura 2.25 apresenta os resultados para uma FFT de 4096 segundo uma decomposição binária balanceada (LEE et al., 2009), visto na figura 2.25 (a-c) e o resultado utilizando a decomposição baseada no algoritmo apresentado na equação 2.19, pode ser visto na figura 2.25(b-d).

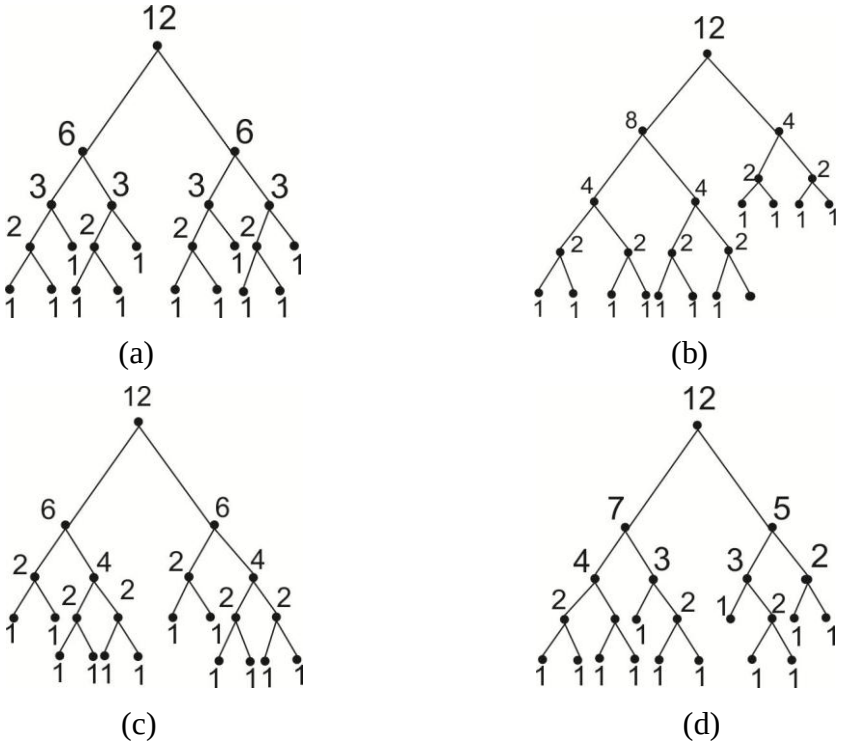


Figura 2.29: Arvore binária: a-c)Decomposição balanceada(LEE et al., 2009); b-d) Decomposição baseado na atividade de chaveamento das multiplicações dos coeficientes trigonométricos (QURESHI; GUSTAFSSON, 2010).

Embora a técnica de decomposição balanceada possa ser implementada para FFTs de vários pontos, demanda construção de hardware paralelo, o que compromete o custo da arquitetura e dificilmente será implementada em hardware.

2.4.5.6 Conclusão sobre os o métodos de decomposição encontrados na literatura

Após a apresentação de vários métodos de decomposição de coeficientes trigonométricos para as arquiteturas FFTs, apresentam-se alguns comentários a respeito. O método apresentado por Oh *et al.* (2005) é apenas associativo, ou seja, substitui a expressão trigonométrica por seu equivalente da tabela trigonométrica. As multiplicações dos coeficientes desse método são controladas pelo uso de multiplexadores. A contribuição de Qureshi e Gustafsson (2009) para o método de Oh *et al.* foi simplesmente a aplicação da técnica MCM nos coeficientes. Os métodos MCM (AKSOY et al., 2008) e CMM (AKSOY et al., 2012) baseiam-se no compartilhamento de subexpressões através do uso de combinações da representação CSD e MSD. Embora sejam métodos genéricos que permitem decompor qualquer conjunto de constantes, os resultados mostram que as representações MSD não representam sempre os melhores resultados. Analisando exclusivamente o problema CMM, é possível afirmar que, se em um conjunto o número de coeficientes for substituído por um conjunto com menos coeficientes obtidos por múltiplos de si mesmos, é provavel que o número de operações seja reduzido. No entanto, isto implica que os novos coeficientes gerados consigam reconstituir os coeficientes abstraídos com o menor aumento de componentes. Encontrar justamente uma forma que garanta que a diminuição do número desses coeficientes não comprometa a estrutura como um todo será o desafio a ser perseguido neste trabalho.

2.5 Resumo

Neste capítulo, apresentou-se uma breve revisão a respeito dos algoritmos da transformada de Fourier na base 2. Neste contexto, é possível projetar arquiteturas FFTs com topologia série, paralela ou mesmo uma mescla das duas (híbridas), sendo que as arquiteturas seriais necessitam menor área que arquiteturas totalmente paralelas, mas demandam maior tempo de processamento. Assim, em muitas aplicações, as arquiteturas híbridas apresentam-se como a melhor opção por permitirem construir arquiteturas rápidas sem comprometer significativamente a área e a dissipação de potência. Também neste capítulo foram apresentados os principais e mais recentes métodos utilizados para otimização de área e dissipação de potência na implementação de arquiteturas FFTs. Neste contexto, foi apresentada a otimização através de diferentes bases e uso de *pipeline*. Também foi exposto o embasamento sobre diferentes formas de multiplicação de constantes que permitem reduzir o número de componentes devido ao compartilhamento de hardware. Neste sentido, foram apresentadas técnicas que buscam reduzir a complexidade da borboleta na base 2 da FFT através da decomposição das constantes trigonométricas que na literatura ainda não foram ainda esgotadas. Nesta descrição de técnicas também se buscou obter subsídios para permitir comparar a proposta de tese a ser elaborada no capítulo 3.

3 TÉCNICAS PROPOSTAS PARA A REDUÇÃO DE COMPONENTES SOMADORES, ÁREA E DISSIPAÇÃO DE POTÊNCIA NAS ARQUITETURAS FFT

De uma forma geral, a literatura já provê diversos métodos para a otimização de área e dissipação de potência para arquiteturas FFTs. A FFT, por ser uma transformação linear, possibilita tratar o problema da decomposição como uma forma de otimização de coeficientes. A decomposição muitas vezes não se trata apenas de fazer compartilhamento entre os diversos componentes existentes, mas sim verificar também a relação de custo em termos de processamento e implementação no nível de hardware. Desta forma, nessa seção apresenta-se um método que permite fazer a decomposição dos coeficientes trigonométricos visando redução do número desses coeficientes e, consequentemente, a sua redução de implementação em hardware.

3.1 Decomposição de Constantes para Arquiteturas FFTs

Para as operações de uma arquitetura FFT de N pontos, são necessários N/2 coeficientes trigonométricos determinados pela equação 1.1. O número desses coeficientes interfere diretamente na dissipação de potência e área, pois estão diretamente relacionados com a estrutura aritmética requerida pela arquitetura (OH et al., 2005).

O método de implementação de arquiteturas FFTs a ser proposto tem como ideia central a decomposição dos coeficientes trigonométricos não inteiros responsáveis pela multiplicação das FFTs. Os coeficientes inteiros (0 e ±1) podem ser facilmente implementados em hardware, pois utilizasse apenas um multiplexador que deixa ou não passar o valor de entrada.

A decomposição de coeficientes deve gerar um novo conjunto com menor número de coeficientes e com custo de componentes somadores/subtratores menor quando comparado ao conjunto original obtido por puramente CMM.

3.1.1 Método Para Decomposição de Coeficientes Trigonométricos não inteiros para uma FFT Serial

Para o desenvolvimento do método algumas regras e passos devem ser considerados:

Primeiro: Um único coeficiente será responsável de gerar todos os outros coeficientes existentes. Assim, será possível recompor a própria arquitetura através da associação dos componentes decompostos.

Segundo: Os coeficientes limites do conjunto à esquerda e a direita, juntamente com o próprio coeficiente a ser decomposto, serão utilizados para gerar três coeficientes

auxiliares. Primeiramente o coeficiente base usado para decomposição é chamado de *intermediate_constantb0*, o coeficiente *intermediate_constantb1* passa ser o coeficiente de menor valor do conjunto e o coeficiente *intermediate_constantb2* é valor da diferença entre o coeficiente de maior valor e o próprio valor base a ser decomposto. Com isso, o conjunto de coeficientes para decomposição não se resume apenas ao número original e às buscas das associações de representações CSD e MSD ou de equivalentes trigonométricos como vistos na literatura.

Terceiro: Cada novo coeficiente parcial (*intermediate_constantbi*) tenta encontrar os demais coeficientes por apenas múltiplos de si mesmo. Isto permite que a implementação em hardware seja possível com apenas somas/subtrações e deslocamentos. No entanto, se não for possível, é realizada a subtração do coeficiente a ser encontrado por um dos coeficientes do novo subconjunto, e verificada novamente a possibilidade de encontrar pelo processo de múltiplos.

Quarto: O somatório de todos os coeficientes gerados não deve exceder o valor do próprio coeficiente decomposto. Deste modo, faz-se que o coeficiente a ser escolhido encontra-se na zona intermediaria do conjunto e coeficientes maiores que o coeficiente base pode simplesmente ser obtido através de deslocamento de um dos coeficientes já determinados anteriormente (*intermediate_constant*). Mantendo este limite a diferença entre os coeficientes extremos é menor, o que permite reduzir o custo de componentes determinado pela função custo.

Quinto: O custo de implementação dos novos coeficientes deve ser o menor possível em termos de componentes. Para isso será analisado o custo de implementação baseado na função custo. Esta função analisa o número de novos coeficientes gerados através de seus múltiplos.

Sexto: A implementação do novo conjunto de coeficientes será realizada por CMM. Desta forma, a nova decomposição CMM deve levar em conta não só a decomposição das constantes em si, mas as associações dos coeficientes a serem combinados para gerar os demais coeficientes que foram supridos pela decomposição.

O fluxograma da figura 3.1 apresenta de forma geral as regras e passos citados para decomposição de cada coeficiente da FFT.

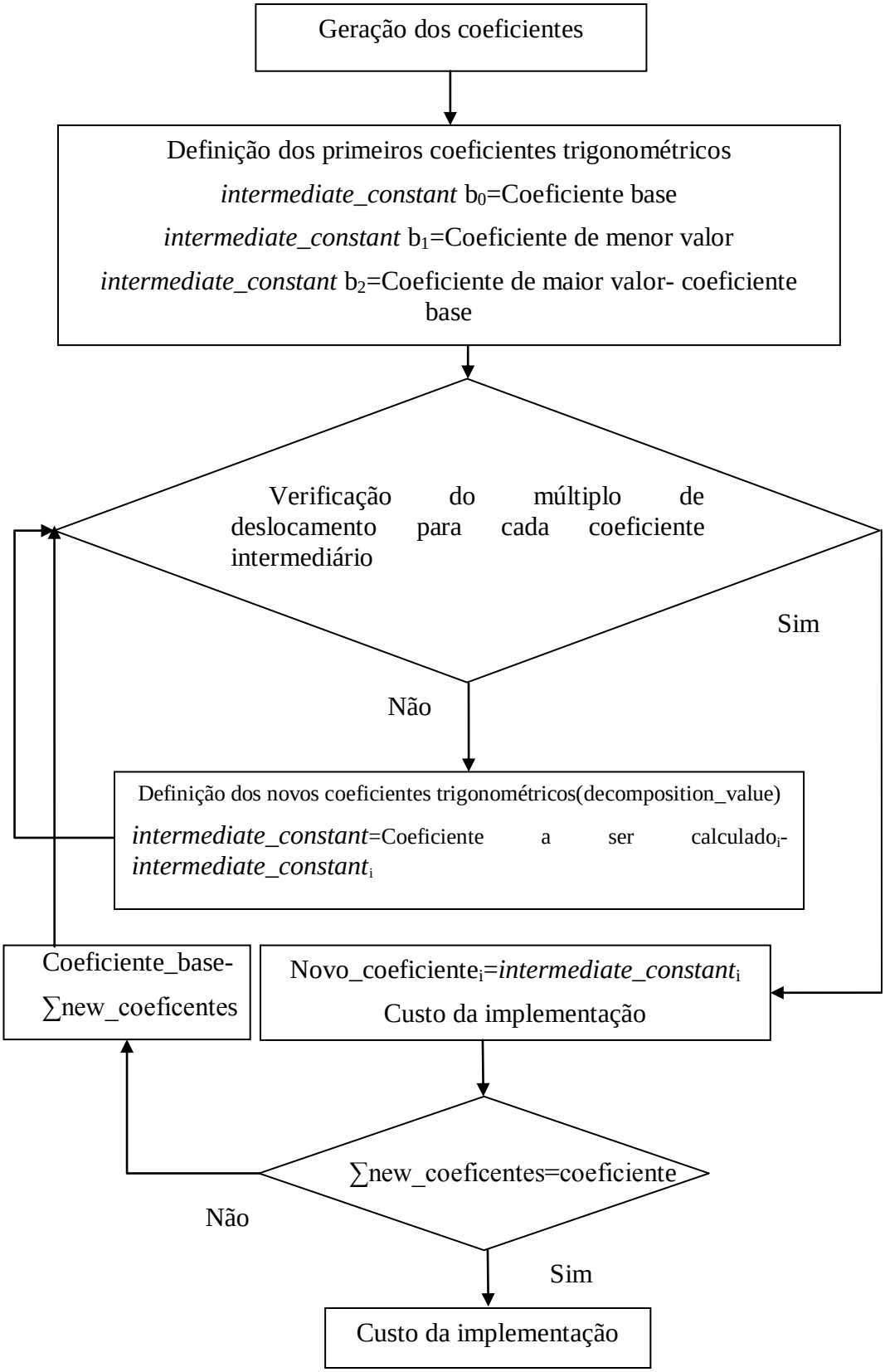


Figura 3.1: Fluxograma de decomposição proposto.

Para apresentar o funcionamento do método do fluxograma da figura 3.1, foi desenvolvido o pseudocódigo mostrado na figura 3.2. Com base no pseudocódigo, serão explicadas detalhadamente as operações do método.

```
1:Main(){
2: For all twiddle factors(tf)
3:(int_c0, int_c1, int_c2) =
      (tf, minimum_value, maximum_value -tf)
4: repeat
5:    N=N+1
6: (new_constanti, shift)= decomposed(for all int_c)
7:    if tf < sum(new_constant) then
8:         int_ci = decomposed
      (tf - sum(new_constant))
9:  until  tf > sum(new_constanti)
10: cost_i = cost_function(for  all(new_constant,shift,N))
11: return cost_i}
```

a)

```
1:Function decomposed (int_c)
2: Repeat
3:   (temp_const, shift) =
 only_shifting(int_ci)
4:  if shift = 0 then
5:    (temp_const, shift) = only_shifting
(for   all const_not_calculated - temp_const)
6:  else
7:         new_constant= temp_const
8: Until all constant set found
9:return new_constanti, shift
```

b)

Figura 3.2: Pseudocódigo para decomposição dos coeficientes. a) função principal; b) Função decompose.

No processamento apresentado na figura 3.2, cada um dos coeficientes originais (tf) não inteiros da FFT é convertido em positivo, e para cada coeficiente original, três valores intermédios que auxiliarão no processo de decomposição (int_c0, int_c1 e int_c2) são definidos (linha 3a). O valor int_c0 é o próprio valor (tf) que será decomposto, int_c1 é o mínimo valor do conjunto de constantes da FFT a ser decomposta e int_c2 é o valor determinado pela subtração do coeficiente de maior valor do conjunto pelo coeficiente base (tf). De posse destes valores intermediários (int_c) no iterativo loop (linhas de 4a a 10a), é iniciado o processo de decomposição pela função decompose (figura. 3.2b).

A função decompose, em primeiro lugar, chama a função only_shifting (linha 3b) para verificar se a partir do conjunto de coeficientes intermediários já obtidos (int_c), as demais constantes podem ser determinadas usando apenas deslocamento.

A figura 3.3 apresenta o pseudocódigo para função only_shfiting.

1:**Function only_shifting (a)**

2: Repeat

3: st= log a for all constants of FFT(tf)

4: if a = 0 then

5: a=sft

(for all const_not_calculated - temp_const)

6: else

7: new_constant= temp_const

8: Until all constant set found

9:**return (b, c)**

Figura 3.3: Função only_shifting.

A função only_shfiting retorna como resultado o valor da constante obtida por deslocamento, temp_const e o número de deslocamentos (shift) necessários para obter o valor de temp_const. Se nenhuma das outras constantes puder ser determinada por deslocamento (shift igual a 0), uma nova verificação é realizada fazendo subtração de cada coeficiente do conjunto que ainda não foi calculado pelos coeficientes já calculados (temp_const) (linha 5b). Se o deslocamento permanecer igual a 0 após todas as tentativas, o último valor calculado é escolhido para temp_const. Este processo é repetido até que todos os coeficientes sejam determinados.

Após varias iterações, observou-se pelo processo experimental que, a fim de garantir que o coeficiente tf decomposto consiga o máximo de reutilização na implementação CMM, o somatório do novo conjunto de coeficientes resultante da decomposição não pode ser maior que o próprio valor de tf. Assim, se esta soma for maior que o tf original, todos os coeficientes são recalculados, fazendo primeiro a diferença de tf com a soma de todos eles (figura 3.2- linha 8a). Por exemplo, sem essa condição para a FFT de 32 pontos, seriam necessários 6 novos coeficientes para reconstituir a multiplicação. Por outro lado, se esta condição for considerada, o novo conjunto é gerado com apenas 5 novos coeficientes.

Como o foco principal do método é a redução de área, foi desenvolvida a função cost_function (figura 3.2-linha 10a), apresentada na Equação 3.1. Esta função determina o custo de implementação para cada coeficiente trigonométrico original submetido para decomposição. O novo conjunto de coeficiente obtidos a partir da decomposição do coeficiente trigonométrico de base que apresentar o menor custo utilizado para decomposição é escolhido para a implementação.

$$\cos t\_function_i = \sum_{n=0}^{n-1} \left(new\_cons\tan t_i * \left(1 - shift_i\right)\right) \tag{3.1}$$

New_constant significa os novos coeficientes temporários obtidos, n é o número de novos coeficientes gerados e shift é o valor de deslocamento para cada coeficiente.

Este método permite que o novo conjunto de coeficientes possa ser implementado com uma maior quantidade de coeficientes obtidos através de deslocamentos. Desta forma, é possível implementar uma arquitetura FFT com um número menor de componentes, o que permite uma redução da área, como também de dissipação de potência.

3.1.2 Decomposição dos Coeficientes Trigonométricos não inteiros para a FFT de 32 pontos

A FFT de 32 pontos tem os seguintes coeficientes trigonométricos reais e complexos: 0, ±1, ± 0,195, 0,381, ±,0,555, ±0,707, , ±0,831, ± 0,9213 e ±0,9803. Considerando que os coeficientes inteiros 0 e ±1 em implementações não demandam hardware, apenas um complemento de dois para obter a parte negativa, nesta decomposição apenas cada coeficiente não inteiro será analisado pelo algoritmo apresentado na figura 3.3. Em primeiro lugar, a função lê o valor dos coeficientes trigonométricos da FFT e começa o processo de decomposição. No caso da FFT 32 pontos, usando como base para decomposição a constante (0,707) o primeiro coeficiente selecionado é o de menor valor, no caso (0,195).

O segundo coeficiente intermediário é escolhido pela diferença entre o maior valor dos coeficientes existentes e o coeficiente que será usado para a decomposição. No caso da FFT 32 pontos, usando como base para decomposição a constante (0,707), o segundo coeficiente intermediário resultante será dado por: (0,9803-0,707=0,273). A partir desta nova constante, é verificada a possibilidade de encontrar outras constantes através de deslocamentos pela função only_shifthing. Caso contrário, o processo determina as constantes (0,195; 0,273; 0,707) para serem usadas na tentativa de determinar todas as demais constantes restantes através das diferenças e deslocamentos.

Seguindo o mesmo procedimento, a constante 0,1855 é determinada pela diferença entre 0,381 e 0,195. Depois que a função é chamada novamente para encontrar uma nova constante por apenas deslocamento, a constante 0,1855 permite encontrar a constante 0,555 através da multiplicação por 3, e assim, é escolhida como uma nova constante temporária. Em termos de custo de hardware é representado por: (0,555 = 0,1855 * 2 (= 0,1855 <<1) + 0,1855). O novo subconjunto é adicionado (0,381, 0,195, 0,1855, 0,707) e o processo se repete até que todas as constantes necessárias sejam encontradas. Os principais passos do algoritmo proposto podem ser visto na terceira coluna da tabela 3.1.

Na tabela 3.1, podem ser observados os valores de custo de decomposição da FFT de 32 pontos para os coeficientes (0,707; 0,555; 0,831), onde é constatado que o coeficiente (0,707) apresenta o menor custo de acordo com a equação (3.1). Assim, usando o coeficiente 0,707 como base para a decomposição, o novo conjunto de coeficientes decomposto é gerado e selecionado com os valores (0,053; 0,124; 0,149; 0,1855; 0,195).

Tabela 3.1: Custo para escolha do conjunto de constantes

Constantes	**Custo Eq. (5.1)**	**Passos para escolha da Nova Constante usando 0,707 como base para decomposição.**	**Seleção de novas constantes**
0.555	0.597	1° 0.1950	
0.707	**0.415** (0.053*(1-2)+ 0.124*(1-0)+ 0.149*(1-0)+ 0.1855*(1-1)+ 0.195*(1-0))	2° 0.273=0.983-0.707 3° 0.1855=0.381-0.1950 4° 0.555=3*0.1855 5° 0.124=0.831-0.707 6° 0.149= 0.273-0.124	0.053; 0.124; 0.149; 0.1855; 0.195
0.831	0.526	7° 0.214=0.921-0.707 8° 4*0.0535=0.214	

Os resultados de decomposição da tabela 3.1 mostram que com apenas 5 novos coeficientes é possível realizar as operações da FFT de 32 pontos. Na tabela só foi apresentado o custo de decomposição de 3 coeficientes trigonométricos da FFT de base devido ao fato de que o método apresenta o menor custo no centro dos coeficientes trigonométricos disponíveis. Isto torna desnecessário o cálculo do custo de decomposição para os demais coeficientes.

Após apresentar o funcionamento do algoritmo para a FFT de 32 pontos, a tabela 3.2 apresenta os resultados de redução do conjunto de coeficientes obtidos com o uso do método de decomposição para a FFTs de até 256 pontos.

Tabela 3.2: Redução do número de coeficientes para FFTs.

FFTs	**Conjunto Original**	**Novo conjunto**	**Redução**	**Tempo de processamento (ms)**
16	3	2	1	41
32	7	5	2	81
64	15	12	3	132
128	31	26	5	378
256	63	52	11	681

A tabela 3.2 mostra que houve uma redução do número de coeficientes para todas as FFTs decompostas. Para a FFT de 16 pontos, a redução foi de apenas um coeficiente trigonométrico, mas para a FFT de 256 pontos a redução foi de 11 coeficientes. Assim, constata-se que, com o aumento do número de coeficientes da FFT envolvida, é possível obter uma maior redução de coeficientes. Isso ocorre porque o aumento do número de coeficientes diminui a diferença entre a constante de base (tf) usada para decomposição.

Outro aspecto observado é que a nossa solução pode ser determinada rapidamente. Usando a ferramenta Matlab em um PC Pentium de 2.4GHz com 2GB de RAM memória, foi possível obter os resultados em menos de 1s.

3.1.3 Inserção de Multiplexadores de Controle

Embora tenha ocorrido a redução do número de coeficientes, é necessário criar um mecanismo de controle que permita sua reutilização. Assim, nesta etapa, são inseridos multiplexadores e sinais de controle que permitem escolher de forma correta as constantes e a reutilização nos estágios correspondentes.

A figura 3.2 apresenta a implementação da parte responsável pela multiplicação da FFT de 32 pontos, onde o novo conjunto de coeficientes foi quantizado usando o formato Q15 para 16 bits de largura e os novos coeficientes assumem 3473; 8126; 9764; 12.156; 12.779, respectivamente. A fim de assegurar a correta multiplicação dos coeficientes trigonométricos, é gerado um sistema de controle através da multiplexação dos coeficientes apresentado por Tummeltshammer *et al.* (2007), onde a geração dos sinais de controle varia com o número de coeficientes e de reutilizações necessárias para fazer a implementação de tal modo, para a FFT de 32 pontos, que os sinais de controle c0, c1 c2 determinam o correto coeficiente trigonométrico a ser operado.

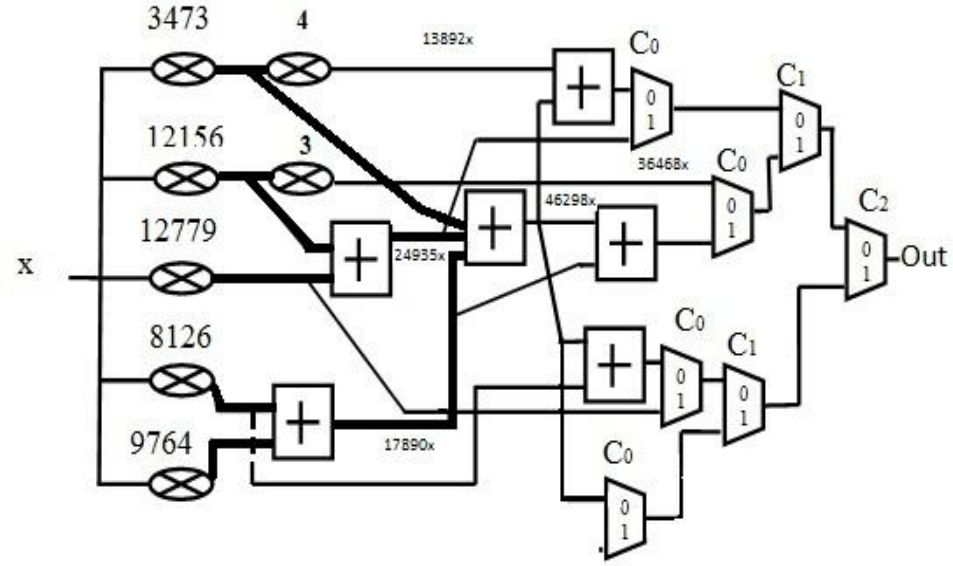


Figura 3.2: Representação da FFT de 32 pontos com as constantes decompostas.

A figura 3.2 mostra o hardware necessário para o cálculo de todos os coeficientes trigonométricos, onde é possível observar que o coeficiente 0,707 (46298) é calculado a partir das constantes decompostas quando os sinais de controle acionados são: c2 = 1 c1 = 1 c0 = 0, ou seja, [3473+ (12156+12779) + (8126 +9764)].

Através da tabela 3.3 é possível observar a combinação de todos os sinais de controle responsáveis pelo cálculo dos coeficientes trigonométricos da FFT 32 pontos.

Tabela 3.3: Sinais de controle das constantes.

Sinais de Controle			Coeficientes Trigonométricos no formato Q15
C2	**C1**	**C0**	
0	0	0	0,923(60190)
0	0	1	0,383(24945)
0	1	0	0,5555(36468)
1	1	1	0,9803(64188)
1	0	0	0,831(54424)
1	0	1	0,195(12779)
1	1	0	0,707(46298)

Como não foram encontrados na literatura métodos que reduzem o número de coeficientes trigonométricos através de sua decomposição para depois reutilizá-los, não foi possível apresentar dados comparativos.

Após decompor os coeficientes trigonométricos e inserir os Muxs de controle, será analisada na próxima seção a implementação destas arquiteturas utilizando a técnica CMM com somadores RC e CSA.

3.1.4 Uso de somadores CSA com CMM

Objetivando tirar maior proveito no uso de CMM (AKSOY et al. 2012), propomos neste trabalho a otimização em CMM através da utilização de somadores CSA. A partir do algoritmo de Aksoy *et al.* (2012), são verificadas as subexpressões que podem ser compartilhadas para associações de 3 operandos. Embora o algoritmo possa ser modificado para otimizar com qualquer associação de somadores, substituir somadores RC por CSA ou por qualquer outro não garante redução de área devido a condições estruturais.

Primeiramente foi realizado um estudo de impacto sobre a implementação de somadores RC e CSA em função da conversão dos circuitos. Desta forma, pode-se verificar a relação na implementação de área, fazendo a síntese de um somador RC e CSA para 16 bits, usando blocos de 4 bits, como visto na figura 3.3. Pela análise, observou-se um aumento de 178% de área em relação ao somador RC quando sintetizado com tecnologia UMC 130nm. Desta forma, concluiu-se que, se todos os circuitos fossem implementados com somadores CSA, o número de blocos diminuiria, mas a área aumentaria, visto que as implementações são realizadas de forma estrutural sem auxílio de otimização por parte da ferramenta comercial.

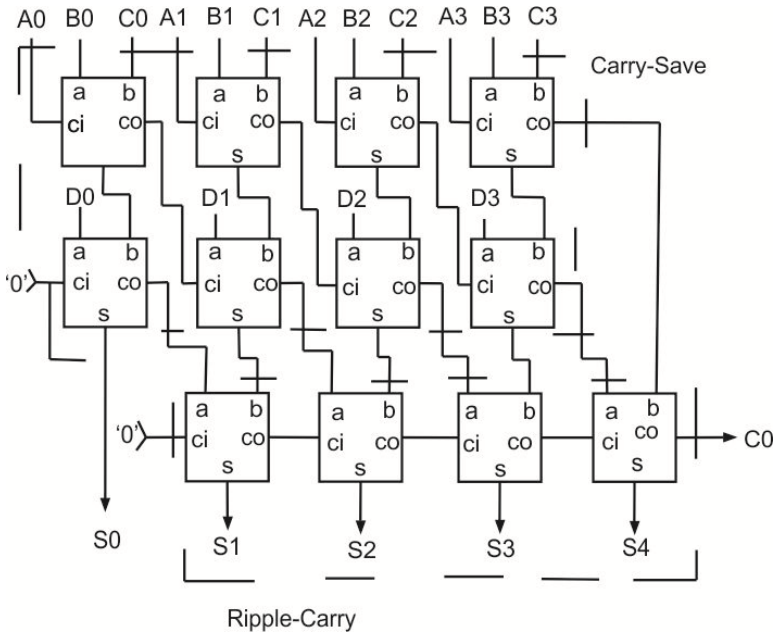
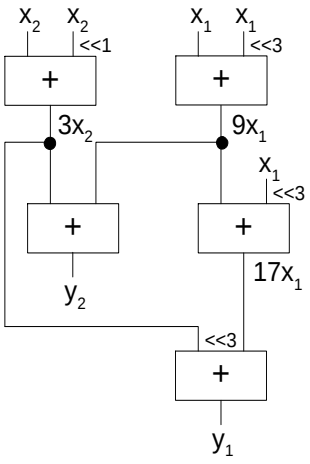


Figura 3.3: Somador CSA de 4 bits.

Baseado nestes dados, introduziu-se uma alteração no algoritmo de Aksoy *et al.* (2012) que garantisse o menor número de componentes, mas sem comprometer a área quando implementado. Depois de uma série de comparações, estabeleceu-se a relação de manter a implementação do circuito com RC entre 70 a 80% assim, garantiria a melhor relação de redução de componentes e área após a síntese.

A alteração do algoritmo de Aksoy *et al.* (2012) foi analisar as saídas e tentar associá-las em operações de CSA mantendo entre 70 a 80% o número de operações RC.

A figura 3.4 representa o uso de CMM com otimização de somadores CSA e RC.



$9x_1=x_1+x_{1<<3}$

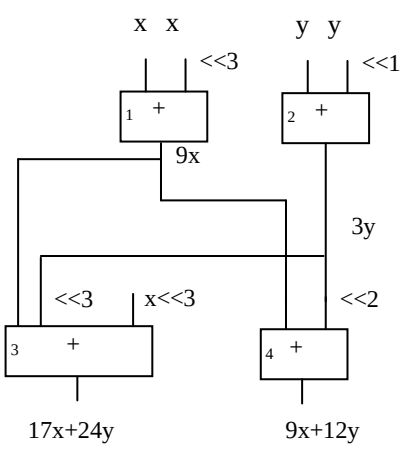
$3x_2=x_2+x_{2<<1}$

$y_2=9x_1+3x_{2<<2}$

$17x_1=9x_1+x_{1<<3}$

$y_1=17x_1+3x_{2<<3}$

(a)



$z\_10=x+x<<3$

$z\_11=y+y<<1$

$z_1=z\_10+z\_11<<4+x<<3$

$z_2=z12+z\_11$

(b)

Figura 3.4: Transformação linear com partilha de subexpressão comum para somadores a) RC; b) CSA.

Analisando-se os resultados da figura 3.4 (b), que usa somadores CSA, observa-se uma redução de 1 componente aritmético em comparação ao resultado apresentado com somadores RC, da figura 3.3 (a). Esta redução permite uma redução na área.

Na tabela 3.4 pode-se observar primeiramente o resultado de saída gerado pelo algoritmo aplicando CMM na seção 2.4.5. A partir dessas expressões, o algorimo apresenta subexpressões através da otimização com RCA, CSA e deslocamentos. E finalmente, cada expressão é associada a uma saída ligada ao sistema de controle através dos multiplexadores.

Tabela 3.4: Expressões para implementação fornecida pelo algoritmo desenvolvido.

Expressões geradas
O1: (+3473*X1) = +(+3473*X1)<<0
O2: (+8126*X1) = +(+4063*X1)<<1
O3: (+9764*X1) = +(+2441*X1)<<2
O4: (+12156*X1) = +(+3039*X1)<<2
O5: (+12779*X1) = +(+12779*X1)<<0
Subexpressões geradas
S1: (+127*X) = -X<<0 +X<<7
S2: (+257*X) = +X<<0 +X<<8
S3: (+901*X) =+S2<<2 -S1<<0
S4:(+2440*X)=+S2<<3+X<<8+X<<7
S5:(+12779X)=+X<<14-S3<<2-X<<0
S6:(+24935X)=+S5<<0+O3<<2
O1: (+3473*X) = +S2<<4 –X<<9-S1<<0
O2: (+4063*X) = -X<<0 +S1<<5
O3:(+3039*X)=X<<10+O2<<0
O4:(+17890X)=+O2<<1+S4<<2+X<<2

O5:(+46298X)=+S6<<0+ O1<<0 + O4<<0
Saídas de dados controladas pelo Multiplexador
SC0: (60190) =+O1<<2+O5<<0
SC1: (24935) =+ S6<<0
SC2: (36468) =+O3<<2 +O3<<1
SC3: (64168)=+O4<<0+O5<<0
SC4: (54424) = +O5<<0 +O2<<1
SC5: (12779) =S5<<0
SC6: (46298) =O5<<0

Após a otimização CMM com somadores de 3 operandos como CSA, pode-se conseguir a redução dos componentes necessários para implementação da arquitetura decomposta da figura 3.2. Assim, a figura 3.5 mostra a implementação utilizando a abordagem CMM. Neste caso, o cálculo do coeficiente (0,707) é dado de acordo com os passos da tabela 3.5.

Tabela 3.5: Expressões para o cálculo do coeficiente trigonométrico usando CMM.

S1: (+127*X) = -X<<0 +X<<7	O1: (+3473*X) = +S2<<4 –X<<9-S1<<0
S2: (+257*X) = +X<<0 +X<<8	O2: (+4063*X) = -X<<0 +S1<<5
S3: (+901*X) =+S2<<2 -S1<<0	O3:(+3039*X)=X<<10+O2<<0
S4:(+2440*X)=+S2<<3+X<<8+X<<7	O4:(+17890X)=+O2<<1+S4<<2+X<<2
S5:(+12779X)=+X<<14-S3<<2-X<<0	O5:(+46298X)=+S6<<0+ O1<<0 + O4<<0
S6:(+24935X)=+S5<<0+O3<<2	---------

A figura 3.5 permite analisar o multiplicador trigonométrico da FFT serial de 32 pontos implementada com apenas somadores/subtratores e multiplexadores inseridos na seção 3.1, onde o sinal de saída depende dos sinais de controle da tabela 5.2 e o cálculo do coeficiente (0,707) pode ser obtido seguindo o caminho em negrito observado na própria figura.

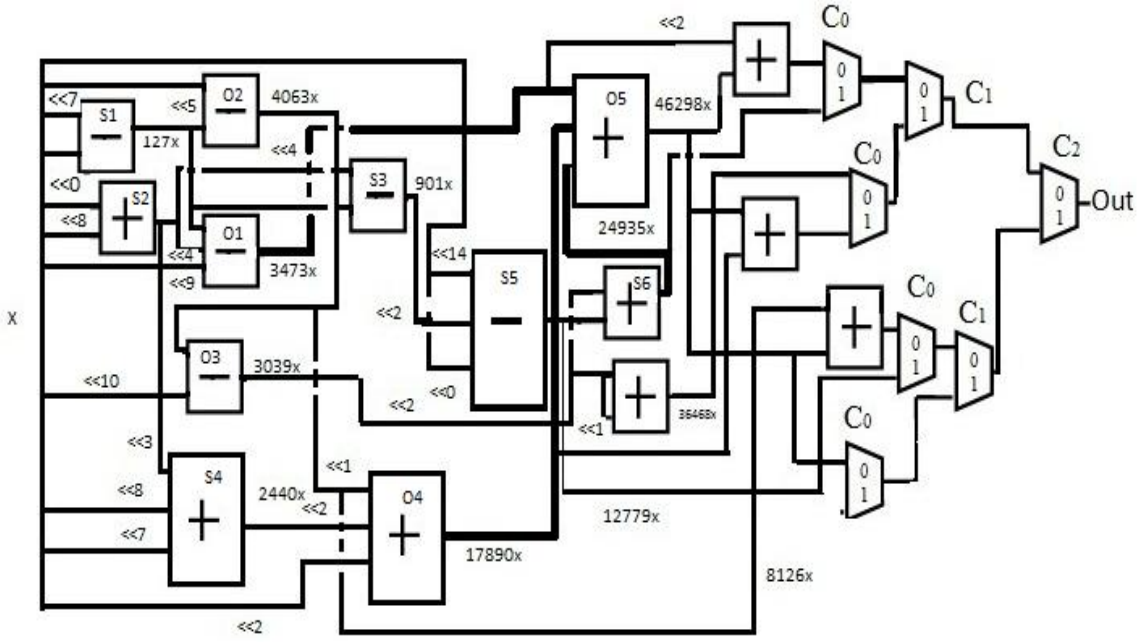


Figura 3.5: Esquemático da FFT de 32 pontos com as constantes decompostas e implementadas com CMM.

3.1.5 Resultado de redução de componentes com largura de bits variável

Para mostrar a eficiência na redução de componentes obtida com o método de decomposição de coeficientes trigonométricos na multiplicação da FFT, é apresentado na tabela 3.6 o custo de implementação da FFT de 32 pontos com largura de bits variável em relação ao estado da arte.

Tabela 3.6: Complexidade de multiplicação dos coeficientes trigonométricos para a FFT de 32 pontos

Bits	**(AKSOY et al. 2012**	**(QURESHI; GUSTAFASSON, 2009)**		**(HAM *et al.*, 2008)**		**Proposto**	
	#Somador/ subtrator	***#Somador/ subtrator***	***#Mux***	***#Somador/ subtrator***	***#Mux***	***#Somador/ subtrator***	***#Mux***
9	10	9	9	8	8	8	7
10	11	9	9	9	8	9	7
11	11	10	9	10	8	10	7
12	12	11	9	11	8	11	7
13	12	12	9	13	8	11	7
14	13	12	9	15	8	12	7
15	14	12	9	15	8	13	7
16	21	14	9	15	8	15	7
17	21	14	9	18	8	15	7
18	22	14	9	19	8	16	7
19	23	15	9	18	8	15	7
20	24	16	9	22	8	16	7

Na tabela 3.6 constata-se que a redução do número de componentes somadores e multiplexadores obtidos pelo método desenvolvido é sempre próxima ou maior quando comparada com os resultados apresentados por Ham *et al.* (2008) e Qureshi; Gustafsson (2009) bem como puramente CMM apresentado por Aksoy *et al.* (2012). No entanto, a aplicação de puramente CMM não necessita de multiplexadores, o que prova a eficácia do método na redução de componentes.

3.1.6 Resultado de redução de componentes para diferentes FFTs

Na tabela 3.7 é mostrado o número de somadores e multiplexadores (MUXs) para multiplicação de FFT de diferentes números de pontos após a decomposição das constantes e implementação com CMM. Para tal, foram utilizados MUX 2:1 como um sistema de controle que habilita a reconfiguração das constantes selecionadas para a parte da multiplicação da borboleta.

O resultado do número de componentes multiplexadores e somadores para FFTs de até 256 pontos pode ser visto na tabela 3.7.

Tabela 3.7: Número de componentes Somadores e Multiplexadores para multiplicação da arquitetura FFT de 16 bits.

FFT-pontos	**(AKSOY, et al., 2012)**	**(QURESHI; GUSTAFSSON, 2009)**		**(HAM et al., 2008)**		**Proposto**	
	#Somador/ subtrator	***#Somador/ subtrator***	***#Mux***	***#Somador/ subtrator***	***#Mux***	***#Somador/ subtrator***	***#Mux***
16	13	15	3	9	4	10	3
32	21	14	9	15	8	15	7
64	32	-	-	-	-	20	9
128	45	-	-	-	-	29	12
256	61	-	-	-	-	41	14

Como pode ser observado na tabela 3.7, os resultados obtidos pelo método de decomposição apresentaram menos componentes somadores\subtratores do que os apresentados pela literatura. Na comparação com puramente CMM de Aksoy *et al.* (2012) o método decomposição apresentou melhores resultados em termos de redução de componentes somadores/subtratores, embora na implementação puramente CMM não sejam necessários multiplexadores. Isso ocorre porque no método proposto os coeficientes decompostos são determinados através da aproximação por diferenças, e a partir deles tenta-se encontrar os demais coeficientes por múltiplos deles mesmos; e a partir dos novos coeficientes é aplicado CMM associando os respectivos coeficientes originais a serem calculados. A grande vantagem do método é que a decomposição deixa os coeficientes adequados para serem implementados com a representação CSD e MSD.

Os ganhos sobre a estratégia adotada em Qureshi *et al.* (2009) usa multiplicadores e somadores controlados por multiplexadores seguindo o caminho da identidade trigonométrica, onde se usam sete constantes para ser decompostas, sendo que no método proposto são utilizadas apenas cinco constantes após a decomposição. Outra vantagem do nosso método é o uso de CMM, que permite reduzir os componentes através da partilha através dos cinco novos coeficientes. Comparando com o trabalho de Ham *et al.* (2008), onde os coeficientes da FFT são implementados usando MCM com estágios de *pipeline* fixo para FFTs de 16 e 64 pontos e otimizações adicionais dos componentes não são permitidos, o método de decomposição proposto utilizou menos multiplexadores devido à redução do número de coeficientes trigonométricos. Embora, com exceção do trabalho puramente CMM de AKSOY *et al.* (2012), só tenha sido possível comparar a nossa solução com as soluções da literatura para FFTs menores (16 e 32 pontos), os resultados mostram a importância dos resultados encontrados.

3.2 Otimização dos componentes ao nível de portas lógicas

Após a decomposição do número de coeficientes obtido com o método proposto, é observada uma redução no número de componentes necessários para implementação, mas isso não garante encontrar uma redução na dissipação de potência. Nos trabalhos sobre otimização de componentes de Han *et al.* (2008) e Qureshi; Gustafsson (2009), é comentado que seus métodos dificilmente obterão resultados satisfatórios em termos de potência após a realização da síntese para implementação, mas não apresentam

resultados comprobatórios. Assim, para garantir a redução de potência juntamente com a redução de componentes, será analisado o efeito em nível de implementação em hardware.

3.2.1 Implementação de somadores/subtratores com 3 operandos em nível de porta

Na tentativa de garantir que a redução de componentes também reduza a dissipação de potência, neste trabalho foi explorada a utilização da métrica em nível de porta para os somadores CSA. Esta técnica é uma avanço em relação à técnica apresentada primeiramente por Aksoy *et al.* (2007) em implementações de somadores RCA para filtros FIR.

Neste trabalho, propõe-se avaliar três operandos de entrada na otimização. Uma vez que os deslocamentos são realizados simplesmente por fios em termos de hardware, os coeficientes parciais da FFT são considerados números ímpares. Assim, existem três diferentes tipos de operações que podem ser consideradas:

$A_{<<SA} + SA_{<<SB} + C_{<<SC}$ (somador, onde SA = 0, SB = S1, S2 = SC)

$A_{<<SA} - B_{<<SB} - C_{<<SC}$ (subtrator, onde SA = S1, SB = 0, SC = S2)

$A_{<<SA} - B_{<<SB} - C_{<<SC}$ (subtrator, onde SA = 0, SB = S1, S2 = SC)

Para uma dada operação que representa os números nas entradas da operação A, B e C, SA, SB e SC denotam o número de deslocamentos sobre as entradas A, B e C, respectivamente. O custo das operações é dado em termos de:

i) número de somadores completos - # FA,
ii) o número de somadores meia - # HA,
iii)) número do Carry Save Somadores - # CSA,

Onde Sn denota o número de deslocamentos; nA, nB e nC apresentam o número de bits de A, B e C nas entradas respectivamente; *nm* é o número mínimo de bits (min (Na + SA, NB + SB, nC + SC)) e *nM* é o número máximo de bits (max (Na + SA, NB + SB, nC + SC)).

O custo para analisar entradas com sinal ou sem sinal pode será mostrado na tabela 3.8:

Tabela 3.8: Custo da função para entradas com sinal ou sem sinal

Entrada Sem sinal	**Entrada com Sinal**
#FA=nM-(nB-nC) -1	#FA=nM –nm -1
#HA= nM- nm -1	#HA= 1
#CSA=max(nA,nB,nC)	#CSA=nM– max(nB,nC) – 1

Observa-se, na tabela 3.8, que o número de bits nas entradas de uma operação depende também da largura de bits da entrada em que as constantes são multiplicadas, denotado por Nb. O custo de cada operação é determinado considerando-se os números com ou sem sinal e, uma vez que estes levam a diferentes implementações, devido à extensão de sinal é analisado por considerar a sobreposição entre as entradas tendo em conta a especificação dos casos.

Neste trabalho são abordadas apenas implementações de números com sinal, visto do uso em arquiteturas de FFT. A figura 3.6 apresenta um exemplo do modelo proposto.

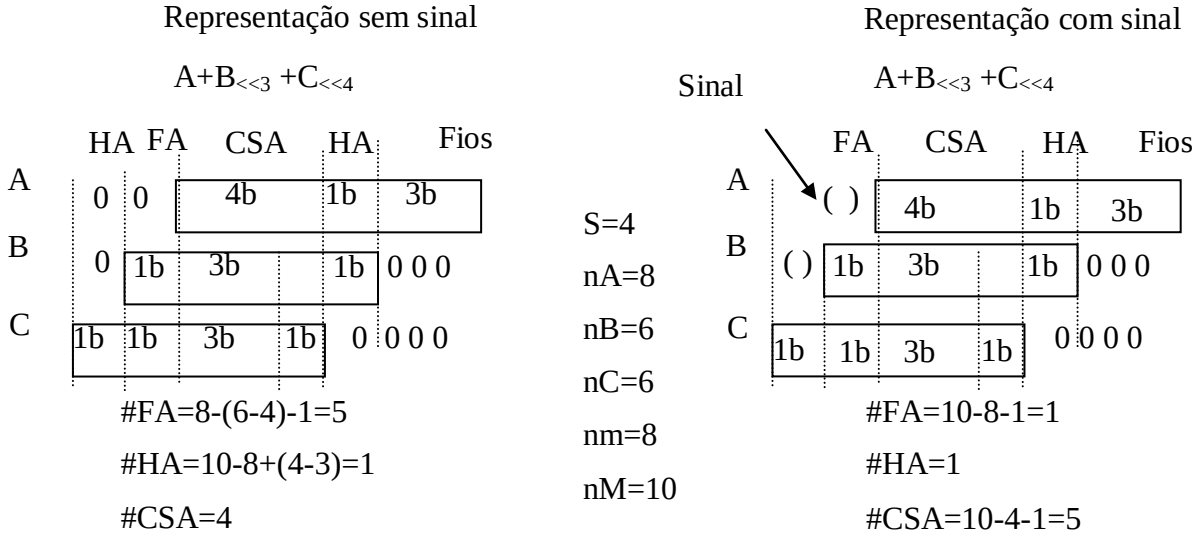


Figura 3.6: Exemplo de custo computacional da operação $A+B_{<<S1}+C_{<<S2}$ realizadas em representação de dígitos sem sinal e com sinal.

Observa-se, pela figura 3.6 com somadores de mais de 2 operandos, tanto para representações de dígitos sem sinal e com sinal, que as operações são realizadas através da utilização de fios que representam os deslocamentos, além dos somadores completos (FA), CSA e meio somadores (HA).

O método analisa basicamente coluna por coluna de bits que devem ser adicionados, e assim usa o melhor somador correspondente para obter a menor área de implementação. Se a coluna tiver 3 bits '1', usa para fazer a implementação um somador CSA, se tiver apenas 2 bits '1', utiliza um somador RC. E finalmente, se tiver somente um bit '1', utiliza um meio somador. Desta maneira, é possível reduzir gradativamente a área do circuito.

Em nosso trabalho Ghissoni *et al.* (2010), foi apresentada a aplicação do uso de CMM em FFTs associada à métrica de porta apresentada por Aksoy *et al.* (2008) em filtros FIR. Na tabela 3.10 são apresentados os resultados de área, dissipação de potência, atraso e o número de células obtidas com a aplicação do algoritmo de MCM (AKSOY et al., 2008) e os circuitos somador/subtrator implementadas ao nível da porta. A síntese lógica foi realizada com a ferramenta *Encounter RTL Compiler* da CADENCE para a tecnologia de UMC 130nm. Três arquiteturas FFT foram implementadas (8 pontos e 16 bits 16 pontos para 16 bits e 32-pontos e 32 bits). As implementações são baseadas tanto na implementação comportamental e estrutural com o uso dos métodos CMM ao nível de porta. As implementações das arquiteturas FFTs são descritas em VHDL e a técnica desenvolvida otimiza os operadores aritméticos. Todas as simulações são limitadas pelo atraso do caminho dos circuitos. Assim, as soluções otimizadas apresentam quase o mesmo desempenho em relação às descrições comportamentais, mas com menor área e dissipação de potência.

Na tabela 3.9 é possível observar a grande redução de área e dissipação de potência usando a abordagem CMM, quando comparada com a implementação comportamental (ainda que todas as implementações apresentem quase o mesmo atraso). Na verdade, o

grande ganho na área e potência é conseguido pela associação das abordagens de redução implementadas em nível de porta e pela redução no número de componentes conseguida através de CMM. Por exemplo, para a arquitetura de 16 pontos, 32 borboletas são usadas na arquitetura FFT e cada borboleta requer quatro multiplicadores para o cálculo da parte real e complexa. Assim, pelo uso da abordagem CMM, uma grande redução no número de multiplicadores é viável, uma vez que estes multiplicadores são substituídos por somadores/subtratores. A substituição dos multiplicadores em geral por somadores/subtratores com uma profundidade lógica reduzida também tem permitido uma redução significativa em termos do número de células, área e dissipação de potência.

Tabela 3.9: Resultados da Aplicação de CMM em Arquiteturas DIT FFT na base 2.

Arquiteturas	**Células**	**Área (μm^2)**	**Potência (mw)**	**Atraso (ns)**
FFT 8P,16b	3178	19240	8.9	12478
Comportamental	12741	36127	35.2	12439
Ganho(%)	**75**	**46.6**	**74.7**	**-0.3**
FFT 16P,16b	5138	36802	15.7	16852
Comportamental	19761	68509	41.8	16806
Ganho(%)	**74**	**46.2**	**62.4**	**-0.2**
FFT 32P,32b	9328	59864	29.5	26809
Comportamental	34411	145248	75,3	26850
Ganho(%)	**72.9**	**58.7**	**60**	**0.15**

Fonte: (GHISSONI et al., 2010)

3.2.2 Resultado de Síntese Lógica

Para mostrar o impacto sobre a redução da complexidade da borboleta usando o método de decomposição, foram implementadas as arquiteturas FFTs para 16 e 32 pontos FFT. As arquiteturas foram implementadas em linguagem de descrição hardware VHDL, e os componentes somadores/subtratores foram implementados com a métrica de porta da seção 3.2.1. A síntese lógica foi realizada com a ferramenta *Design Compiler* da Synopsys, com tecnologia CMOS UMC 130nm e a potência média foi avaliada pela ferramenta *Design Power* também da Synopsys com 5000 vetores de aleatórios de entrada.

Todas as simulações são limitadas pelo atraso do caminho dos circuitos, e as mesmas topologias de somadores e multiplexadores foram utilizadas para garantir uma comparação justa entre as soluções analisadas partir da literatura.

Tabela 3.10: Resultado de síntese lógica do método proposto

Pontos FFTs (16 bits)	**Solução Proposta**			**(QURESHI; GUSTAFSSON, 2009)**			**(HAM *et al.*, 2008)**		
	Área (um^2)	***Potência (uW)***	***Atraso (ps)***	***Área (um^2)***	***Potência (uW)***	***Atraso (ps)***	***Área (um^2)***	***Potência (uW)***	***Atraso (ps)***
16	7319	7.1	9127	9506	8.3	9315	7415	7,2	8615
32	9891	8.4	9479	10016	8.7	9512	10357	8.9	8889

A tabela 3.10 apresenta reduções de até 24% e 15% na área e dissipação de potência, respectivamente, quando comparadas com a solução da FFT de 16 pontos (QURESHI et al., 2009). No entanto, para FFT de 32 pontos, a redução foi de 2% para área e 4% para a potência. Estes melhores resultados apresentados pelo método proposto devem-se à diminuição do número de componentes (somadores e multiplexadores) obtida com a decomposição das constantes. A tabela 3.11 também mostrou uma redução de apenas 1% da área e no aumento de 4% na potência, quando comparado com a solução de Han *et al.*(2008) para a FFT 16 pontos, e uma redução de 6% na área e 6% na potência pra a FFT de 32 pontos. Embora o método proposto tenha apresentado resultados próximos ao trabalho de Han *et al.* (2008), deve-se considerar que no trabalho proposto não foi utilizado *pipeline* entre os estágios da FFT, como tem sido feito no trabalho mencionado. Outro aspecto a ser observado é que o método proposto sempre produziu soluções com menor complexidade em termos de menor número de somadores e multiplexadores. Portanto, as soluções podem ser sintetizadas rapidamente para uma variedade de diferentes números de pontos para as arquiteturas FFT quantizadas em 16 bits. Usando a ferramenta MatLab em um Pentium PC, operando a frequência de 2.4GHz com 2 GB de memória RAM, foi possível obter os somadores e multiplexadores necessários para a FFT de 256 pontos a uma largura de 16 bits em 1.8s, enquanto para a FFT de 16 pontos na mesma largura foram necessários apenas 0,7s. Isso prova a simplicidade do método, que gera soluções com pouco esforço computacional.

Na tabela 3.11 foi aplicado o uso de *pipeline* como tentativo de redução do atraso do circuito. Foram instalado o mesmo número de estágios que Han *et al.* (2008) e verificar o comportamento do circuito.

Tabela 3.11: Resultado de síntese lógica do método proposto com pipeline

Pontos FFTs (16 bits)	**Solução Proposta**			**(QURESHI; GUSTAFSSON, 2009)**			**(HAM *et al.*, 2008)**		
	Área (um^2)	***Potência (uW)***	***Atraso (ps)***	***Área (um^2)***	***Potência (uW)***	***Atraso (ps)***	***Área (um^2)***	***Potência (uW)***	***Atraso (ps)***
16	7477	6.6	8715	9578	8.3	9378	7415	7,2	8615
32	9903	8.2	8907	10183	8.9	9581	10357	8.9	8889

A tabela 3.11 apresenta a redução de 22% e 16% em área e potência, respectivamente, quando comparado com a solução de QURESHI *et al.* (2009) para FFT de 16 pontos. No entanto, para 32 pontos FFT, a redução foi de até 11% e 8% na área e de potência, respectivamente. Estes melhores resultados apresentados pela nossa solução são, devido à diminuição do número de componentes (somadores e multiplexadores) obtidos com a decomposição das constantes. A tabela 3.12 também mostrou a redução de 2% da área e da diminuição de 2% na dissipação de potência quando comparado com a solução de Han *et al.* (2008) para a FFT de 16 pontos e uma redução de 5% na área e 5% em potência para a FFT de 32 pontos. No entanto, deve-se destacar que os resultados de atraso com *pipeline* foram praticamente os mesmos que Han *et al.* (2008), mas melhores que QURESHI *et al.* (2009) o que comprova a eficiência da nossa proposta de solução.

3.2.3 Resultado de redução de transistores

Para comprovar a redução de componentes devido o uso do método de decomposição foi realizada a análise do número de transistores dos circuitos implementados. Para a síntese física foi usada a ferramenta IC Compiler da Synopsys na tecnologia UMC 130µm e os circuitos foram implementados de forma estrutural.

Tabela 3.12: Número de transistores para FFTs variáveis de 16 bits.

FFT-pontos	**(AKSOY, et al., 2012)**	**(QURESHI; GUSTAFSSON, 2009)**	**(HAM et al., 2008)**	**Proposto**
	Transistores	***Transistores***	***#Transistores***	***#Transistores***
16	8072	11628	8198	8124
32	14112	14424	14596	14256
64	21504	-	-	20206
128	30240	-	-	26788
256	40992	-	-	35768

A tabela 3.12 apresenta o resultado de redução no número de transistores quando implementados com o método de decomposição e comparados com os demais trabalhos da literatura, como já visto na tabela 3.7. Os resultados mostram uma redução de transistores em todas as comparações com QURESHI; GUSTAFSSON (2009) e

próximos com HAM et al. (2008). Esta redução esta diretamente relacionada com o número de componentes, como estes são implementados de forma estrutural o menor número de componentes gera o menor número de transistores. Na comparação com CMM de AKSOY et al. (2012) que não usa multiplexadores a redução de transistores nos circuitos decompostos foi observada apenas para arquiteturas acima de 32 pontos. Isto ocorre porque embora na decomposição proposta necessite de multiplexadores para controlar as operações, a redução no número de componentes somadores/subtratores é maior do que puramente CMM nessas arquiteturas, como visto na tabela 3.7. Na implementação de FFTs menores o número de multiplexadores acaba influenciando no número de transistores totais. Pois, na implementação de um multiplexador embora necessite menos transistores que um componente somador/subtrator vários deles acabam ser significativos e fazem com que a implementação puramente CMM para FFT de até 32 pontos seja 1% melhor.

3.3 Resumo

Neste capítulo foi apresentado o método de decomposição dos coeficientes trigonométricos responsáveis para a multiplicação da borboleta em uma FFT. O método consiste em determinar o conjunto de novos coeficientes através de aproximações entre um coeficiente base e os demais coeficientes a serem determinados. A vantagem do método é que estes novos coeficientes podem ser determinados por múltiplos que em hardware são facilmente implementados por deslocamento. Outra vantagem é que o número de coeficientes após a decomposição necessários para reconstituir a FFT original diminui com o aumento do número de coeficientes envolvidos. Outra diferença observada com o método foi o desenvolvimento da função custo baseada em dar maior peso para coeficientes que possam ser implementados com apenas deslocamentos dos demais coeficientes calculados. Os resultados obtidos após a síntese lógica usando somadores RC juntamente com CSA demonstram que a redução de coeficientes e sua implementação ao nível de porta uma redução de potência, quando comparada com os demais trabalhos do estado da arte. Mas a grande vantagem observada pelo método foi a melhoria da decomposição em termos de componentes somadores/subtratores quando comparada com a técnica puramente CMM, onde as decomposições são baseadas nas associações de representações CSD e MSD.

4 OTIMIZAÇÃO DE ARQUITETURAS HÍBRIDAS ATRAVÉS DA DECOMPOSIÇÃO DOS COEFICIENTES TRIGONOMÉTRICOS

A decomposição de coeficientes trigonométricos originais da parte de multiplicação da FFT na base 2 mostrou atender o quesito de otimização como visto no capítulo anterior. Analisar o efeito desta decomposição em FFTs híbridas pode garantir ganhos para arquiteturas de muitos pontos. Assim, neste capítulo será abordado o uso do método de decomposição em arquiteturas híbridas e verificado a redução de área e dissipação de potência.

4.1 Método Para Decomposição de Coeficientes Trigonométricos em Arquiteturas Híbridas

O segundo método proposto foi à utilização da decomposição de coeficientes trigonométricos na implementação de FFTs híbridas que permitem redução de dissipação de potência quando comparada com uma arquitetura totalmente paralela ou série. O método e sua contribuição para o estado da arte foram descritos no artigo (GHISSONI et al., 2011; e GHISSONI et al., 2012).

Para implementar uma arquitetura híbrida deverá ser utilizado uma arquitetura FFT paralela como base. Esta arquitetura base, por ser dividida em estágios, o processo de decomposição dos coeficientes pode ser realizada em diferentes estágios o que permite que todos os coeficientes necessários para multiplicação sejam obtidos de forma eficiente com redução de área e dissipação de potência.

O fluxograma da figura 4.1 representa os procedimentos a serem seguidos que são explicados nas subseções seguintes.

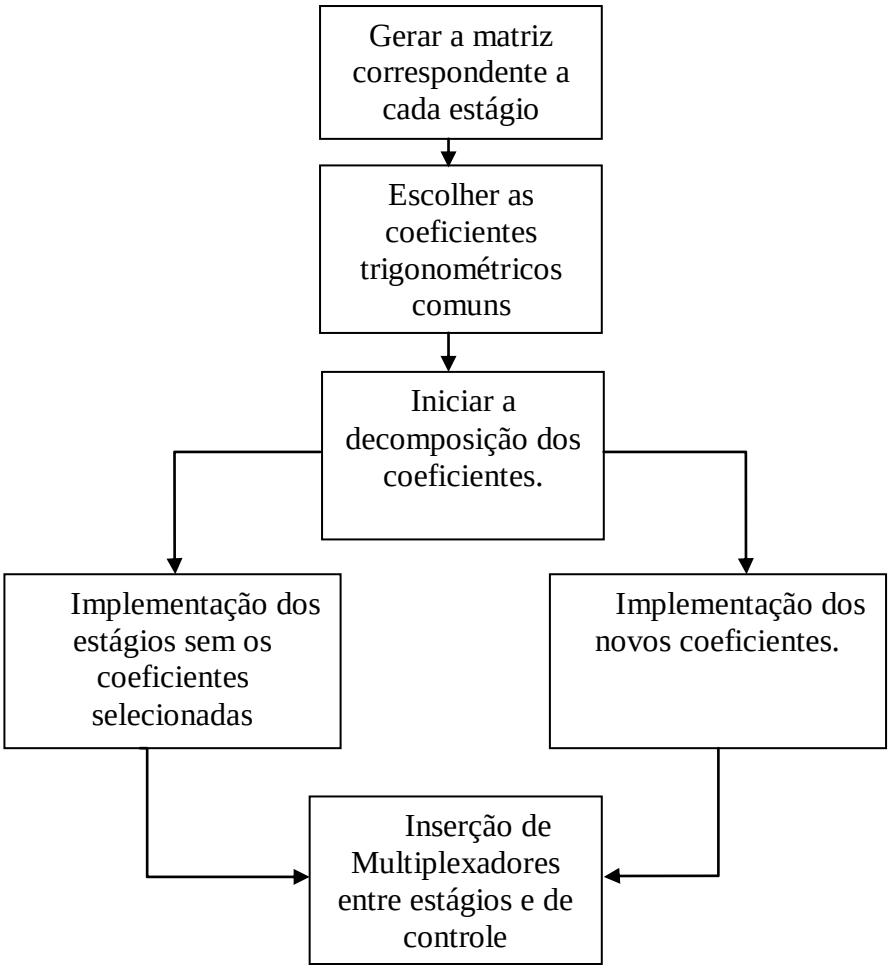


Figura 4.1: Fluxograma geral do mecanismo de implementação da FFT híbrida.

4.1.1 Entrada do número de pontos e geração das matrizes dos estágios

Através do número de pontos e a largura de bits são gerados os coeficientes da FFT através das equações, 2.1, 2.2 e 2.3 apresentadas por Cooley; Tukey (1965). Como a FFT base (paralela) é dividida por estágios são gerados matrizes de coeficientes para cada estágio. Neste trabalho os coeficientes gerados são para arquitetura FFT DIT.

4.1.2 Escolha e decomposição dos coeficientes nos diferentes estágios da FFT paralela

Para implementação de uma arquitetura FFT híbrida onde a base é uma arquitetura FFT paralela, à escolha à decomposição dos coeficientes trigonométricos deve avaliar os estágios dessa arquitetura base. Pois, uma a arquitetura híbrida por condições estruturais, deve facilitar a reutilização parcial ou total de seus componentes a fim de satisfazer as operações dos estágios que deverão ser operados após a decomposição. Desta forma, o uso de uma FFT paralela de menos pontos após a decomposição deve permitir a completa operação da FFT a ser implementada.

Para implementação o primeiro passo é verificar se em cada estágio da FFT paralela utilizada como base, quais são os coeficientes que serão decompostos. A partir do(s)

coeficiente(s) escolhido(s) para decomposição é realizado o cálculo dos coeficientes intermediários baseada no método proposto na seção 3.1.

4.1.3 Implementação dos estágios que possuem coeficientes selecionados.

Pelo método visto na seção 3.1 todos os coeficientes de cada estágio da FFT base são decompostos e o que tiver o menor custo de implementação dado pela função custo (visto na seção 3.1) será escolhido.

Os estágios que possuem coeficientes selecionados são implementados em duas etapas: primeiro são implementadas os coeficientes extraindo aqueles escolhidos para decomposição e em segundo são implementados os novos coeficientes decorrentes da decomposição através da utilização de CMM associado à métrica de porta desenvolvida na seção 3.2.

A figura 4.2 representa o fluxo de implementação dos estágios usando CMM:

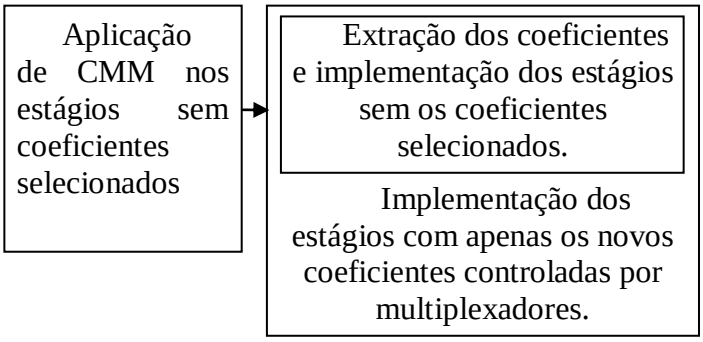


Figura 4.2: Implementação dos estágios realizada pelo método desenvolvido.

Deve se destacar que apesar das implementações serem realizadas separadamente um sistema de controle com multiplexadores permite controlar a correta escolha do coeficiente para multiplicação e os acessos dos registradores intermediários.

4.2 Implementação de uma arquitetura FFT híbrida de 32 pontos

Embora brevemente já apresentado o fluxo de projeto para a FFT híbrida, nesta seção será apresentado a implementação de uma arquitetura FFT híbrida de 32 pontos DIT construída sobre uma arquitetura FFT DIT paralela de 8 pontos. Desta forma, será necessário reutilizar parte da arquitetura FFT de 8 pontos para obter todos coeficientes necessários para o cálculo das saídas.

Uma arquitetura FFT de 32 pontos totalmente paralela possui 16 borboletas responsáveis por operações dos operandos reais e outras 16 para os operandos imaginários. Esta arquitetura possuirá 5 estágios determinados pela relação logarítmica da equação 4.1.

$$Est = \log_2^N \tag{4.1}$$

Onde N é o número de pontos da FFT totalmente paralela e Est o número de estágios.

Para as operações dos 32 pontos é necessário analisar todos os coeficientes de seus estágios. Como a base é uma FFT paralela de 8 pontos, inicialmente, são analisados os três diferentes estágios desta arquitetura. Assim, a manipulação dos coeficientes deve atender as operações parcias e armazená-las em um registrador para serem reutilizadas posteriormente.

Os três estágios da FFT de 8 pontos totalmente paralela apresentam os seguintes coeficientes trigonométricos: 0, ±1, ±0,707. Mas, para implementar a FFT de 32 pontos são necessários, os seguintes coeficientes trigonométricos: 0, ± 0,195, 0,381, ±,0,555, ±0,707, , ±0,831, ± 0,9213 e ±0,9803. Ao analisar o conjunto de coeficientes observa-se que apenas os coeficientes ±0,381 e ±0,923 são diferentes para o quarto estágio da arquitetura de 32 pontos. Já que para o quinto estágio os coeficientes ±0,1950, ±0,555, ±0,831 e ±0,9803 também são diferentes. Como os coeficientes de maior valor na maioria dos casos, segundo Aksoy *et al.* (2008) precisa de maior área a ser implementado, devemos encontrar o conjunto de coeficientes que pode ser manipulado de forma eficiente para não comprometer as operações executadas pela estrutura. Desta forma, é necessário analisar todos os coeficientes da arquitetura da FFT de 8 pontos antes de decompôr-la.

Ao observar os três estágios da FFT de 8 pontos, os coeficiente que podem atender todos os demais encontram-se entre 0,195 e 0,980. Assim, a forma que foi encontrada para atender a falta desses coeficientes foi a decomposição do coeficiente 0,707 usando a técnica de decomposição apresentada na seção 3.1.

A grande vantagem deste método é que apesar de aparentemente calcular um número maior de coeficientes só define os que serão implementados se atenderem às restrições de área, que neste caso são determinadas pela função custo da equação 3.1.

Pelo fluxo visto na figura 4.2 o primeiro e segundo estágios da FFT paralela de 8 pontos são implementados utilizando apenas CMM. O terceiro estágio será implementado em duas etapas: A primeira utilizando CMM sem os coeficientes escolhidos para decomposição, e a segunda utilizando apenas os coeficientes decompostos.

Para a implementação da FFT híbrida de 32 pontos deve-se notar que é necessário o desenvolvimento de uma arquitetura real e outra imaginária. Como o terceiro estágio da FFT paralela de 8 pontos apresenta 2 coeficientes (0,707), estes podem ser utilizados a fim de aumentar a velocidade de processamento em ambas as partes da arquitetura.

A figura 4.3 apresenta a estrutura da arquitetura híbrida. Nesta estrutura um registrador de entrada é responsável para armazenar os 32 pontos do sinal de entrada. A estrutura FFT de 8 pontos responsável para fazer as operações de 8 pontos é dividida em 3 estágios onde o terceiro estágio agrega a decomposição de coeficientes. Este estágio é controlado por um sistema de multiplexadores que permite o armazenamento das operações intermediarias dos relativos 8, 16 e 32 pontos. E finalmente, um registrador de saída com os resultados finais.

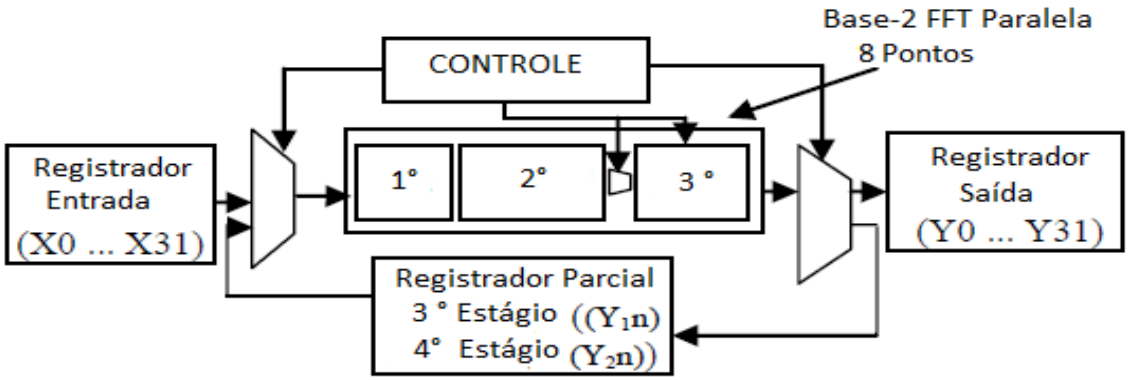


Figura 4.3: Esquemático da arquitetura FFT híbrida de 32 pontos

Na figura 4.4 é possível observar a implementação do terceiro estágio da FFT. Na entrada existe um registrador que armazena 8 a 8 pontos, 2 coeficientes decompostos e um registrador intermediário para armazenar os dados para sua reutilização posterior.

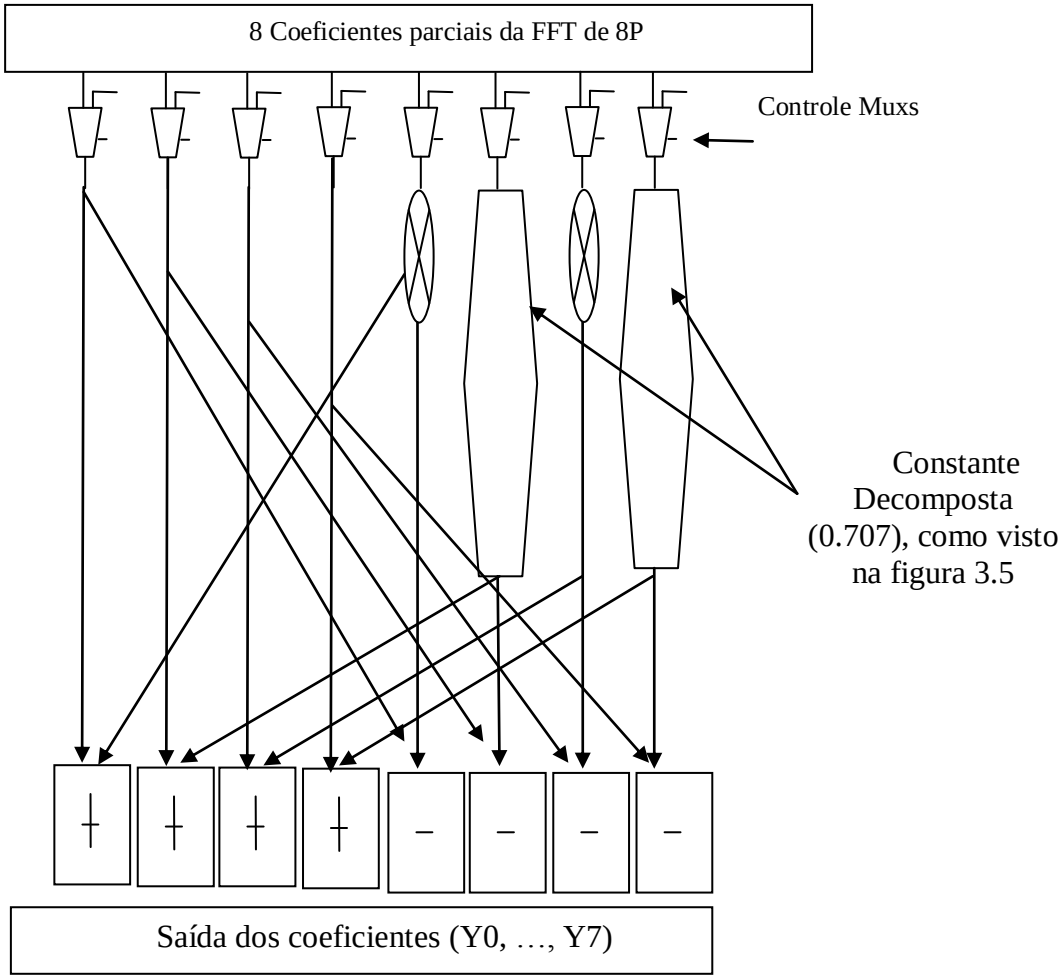


Figura 4.4: Esquemático da Implementação do terceiro estágio da FFT 8 pontos com o coeficiente trigonométrico (0,707) após a decomposição.

As operações da arquitetura híbrida para a parte real e imaginária podem ser realizadas de acordo com os seguintes passos:

1) **Cálculo dos 32 coeficientes relacionados com a FFT de 8 pontos (Y1n).**

Os 32 coeficientes parciais (Y1n) são calculados diretamente da arquitetura FFT de 8 pontos, em 4 etapas (8 a 8 respeitando a ordem correta) e em seguida, os coeficientes são armazenados no registrador intermediário.

2) **Cálculo dos 32 coeficientes relacionados com a quarto estágio**.

Os coeficientes (Y2n) são relacionados com os coeficientes do quarto estágio da FFT de 32 pontos. Estes coeficientes parciais são determinados pela reutilização do terceiro estágio da FFT de 8 pontos, por meio do processamento dos coeficientes (Y1n) que foram previamente armazenados. A ordem correta dos coeficientes (0,381, 0,707 e 0,923), que são necessários para as multiplicações nesta etapa, são determinados pelo sistema de controle no coeficiente decomposto, como visto na figura 4.4. Como são dois coeficientes modificados no terceiro estágio, em cada ciclo são calculados 4 coeficientes (Y2n), desta forma são necessários 8 ciclos para calcular os 32 coeficientes deste estágio. Após, os resultados de cada operação são então armazenados novamente no registrador intermediário.

3) **Cálculo dos 32 coeficientes relacionados com a quinto estágio**.

Nesta etapa é repetido o mesmo procedimento anterior, onde os coeficientes (Y2n), são multiplicados por cada uma das constantes correspondente ao quinto estágio da arquitetura de 32 pontos, ou seja, 0,1950, 0,555, 0,707, 0,831, 0,923 e 0,9803, através da correta ordem do sistema de controle e, portanto, todos os valores calculados são enviados para a saída

Na seção a seguir são apresentados os resultados de dissipação de potência, área e atraso obtidos com a implementação e sintetizados logicamente.

4.2.1 Resultados da FFT para arquiteturas híbridas

As arquiteturas híbridas foram descritas em linguagem de descrição hardware VHDL, e os componentes somadores/subtratores foram implementados com a métrica de porta de Ghissoni *et al.* (2010) adaptada para CSA apresentada na seção 3.2. A síntese lógica foi realizada com a ferramenta *Encounter RTL Compiler* da CADENCE com tecnologia CMOS UMC 130nm sendo que para as arquiteturas estruturais foi restringido à possibilidade de otimização fornecida pela ferramenta.

A síntese das arquiteturas estruturais exigem restrições da ferramenta que garantem que os componentes descritos sejam sintetizados sem alterações provocadas pelas bibliotecas da ferramenta.

Primeiramente deve ser destacado que a decomposição dos coeficientes trigonométricos realizada pelo método proposto permite obter resultados melhores que os obtidos sem o método. Essa afirmação pode ser validada através dos resultados da tabela 4.1 onde foi implementada a FFT de 8 pontos de forma estrutural com decomposição e sem decomposição associada a implementação com CMM e métrica de porta com CSA.

Tabela 4.1: Resultados da FFT DIT 8 pontos utilizada como base

	Sem decomposição	**Com decomposição**
Área (μm^2)	258,70	228,68
Atraso (ps)	11891	12304
Potência média (µw)	29.1	27,1

Na tabela 4.1 é observado uma redução de 12% em área e 7% em dissipação de potência para a implementação da FFT paralela com decomposição de coeficientes em relação à implementação sem decomposição. Esta redução de área e de dissipação deve-se principalmente ao fato de que a decomposição dos coeficientes diminui o número de coeficientes a serem implementados que age diretamente na diminuição de dissipação de potência. No entanto, tanto na implementação estrutural quanto comportamental foi observado que o resultado de atraso na arquitetura FFT com decomposição foi maior que o obtido na arquitetura sem decomposição. Isto ocorre, pois a decomposição dos coeficientes trigonométricos causa um aumento no caminho crítico na implementação, justamente porque os coeficientes trigonométricos responsáveis pela multiplicação da FFT são determinados pela associação destes novos coeficientes.

Para avaliar os ganhos das arquiteturas projetadas será analisado a relação do produto atraso-potência (PDP-*Power-delay-product*) dada pela equação:

$$PDP = P_D t_D \tag{4.2}$$

Sendo P_D a potência dinâmica e t_D o atraso do circuito.

Avaliando pela métrica PDP é possível avaliar que a melhor arquitetura projetada é a por decomposição, pois seu PDP obtido é 279 nJ, contra 290 nJ para a arquitetura projetada sem decomposição. Pela análise, quanto menor o valor de PDP, melhor a tecnologia do ponto de vista de maior velocidade e menor dissipação de potência.

Após analisar a implementação da FFT base de 8 pontos será observado o comportamento de dissipação de potência, área e atraso nas operações híbridas de 32 pontos através da tabela 4.2.

Tabela 4.2: Resultados de comparação do uso do método de decomposição.

	Sem decomposição		**Método Proposto**	
	Estrutural	**Comportamental**	**Estrutural**	**Comportamental**
Área (μm^2)	4865,8	6852,2	4420,1	6390,1
Atraso (ps)	13819	13715	14406	13956
Potência média (µw)	55,1	61,7	52,6	60,9

Na tabela 4.2, pode-se observar que a redução de potência média total obtida na arquitetura FFT híbrida implementada através da decomposição dos coeficientes trigonométricos que permite a reutilização da própria arquitetura foi de 30% e 14% para área e dissipação de potência respectivamente, quando comparada com a arquitetura implementada de maneira comportamental. Na verdade, a substituição de

multiplicadores nas borboletas nos estágios das FFTs de somadores/subtratores leva à redução do dissipação de potência do circuito, quando comparada com a descrição comportamental, onde os operadores aritméticos a partir da ferramenta comercial são utilizados automaticamente através da biblioteca de células. Outro aspecto interessante é observado os resultados entre as implementações estruturais da FFT com decomposição e a sem decomposição, neste caso, o ganho com a aplicação do método foi de 11% em área e 6% em dissipação de potência. Esta redução deve-se principalmente que a decomposição dos coeficientes diminui o número de coeficientes a serem implementados, além do que na implementação híbrida sem decomposição os coeficientes dos estágios precisam de componentes multiplicadores para realizar as multiplicações dos diferentes coeficientes da FFT base. Já analisando as arquiteturas comportamentais a redução fornecida pelo método foi melhor apenas 1% em dissipação de potência e 5% em área. Isto é devido basicamente à redução dos componentes obtida pelo método, visto que a ferramenta utiliza os componentes disponíveis em sua biblioteca.

No entanto, mais uma vez foi observado que o resultado de atraso na arquitetura FFT com decomposição foi maior do que a arquitetura sem decomposição tanto na implementação estrutural quanto na comportamental. Isto ocorre, como já observado anteriormente, porque a decomposição dos coeficientes trigonométricos provoca um aumento no caminho crítico para sua implementação.

Fazendo a análise das arquiteturas da tabela 4.2 pela métrica PDP observa-se que na implementação comportamental sem decomposição o valor obtido foi 846nJ e com decomposição 840 nJ. Já para as implementações estruturais sem decomposição o valor foi de 761 nJ e com decomposição 757 nJ. Pelos resultados obtidos ficou comprovado o ganho das implementações estruturais sobre as comportamentais. Sendo também observada uma pequena vantagem nos resultados das implementações quando aplicado o método de decomposição de coeficientes.

Após a análise do método proposto para as FFTs Híbridas de 32 pontos foi realizada a comparação com diferentes trabalhos presentes na literatura de FFTs híbridas com até 256 pontos.

A tabela 4.3 apresenta os resultados para as arquiteturas híbridas das FFT de 32 pontos com todas as técnicas incorporadas nas seções 3.1 e 3.2, bem como uma comparação com o estado da arte onde as síntese foram realizadas com os mesmos requisitos da tabela 4.2.

Tabela 4.3: Resultados decomposição de coeficientes na base 2 para implementação da arquitetura híbrida FFT DIT de 32 pontos com uma FFT paralela de 8 pontos de base.

Circuitos	**Área (μm^2)**	**Potência média (µW)**	**Atraso (ps)**
FFT 32P,16b- Híbrida	4420,1	52,6	14406
Comportamental	6390,1	60,9	13956
(AHMADINIA; AHMAD; ARSLAN, 2007)	4943,9	54,7	13745
(MACLEOD, 2005)	5025,6	54,5	14275

A tabela 4.3 apresenta os resultados para a arquitetura FFT híbrida de 32 pontos, utilizando a metodologia apresentada anteriormente na seção 4.1. Como pode ser observado, ocorrem reduções de até 15% no dissipação de potência e 31% na área quando comparado com arquiteturas implementadas de forma comportamental usando síntese lógica da ferramenta comercial CADENCE. Esses resultados comprovam a eficiência das arquiteturas híbridas que foram implementadas usando pequenos módulos de uma arquitetura FFT de 8 pontos. Estes módulos utilizam CMM e métrica de nível de porta com CSA, que permite a substituição dos multiplicadores das borboletas por somadores/subtratores e *shifters* de forma eficiente, ao invés de usar o operador aritmético da biblioteca, como é usado na ferramenta comercial. Portanto, a decomposição dos coeficientes em outros coeficientes mais adequados, permite as FFTs híbridas serem implementadas com menos área e menor dissipação de potência quando comparadas a arquiteturas que utilizam apenas o CMM e métrica de nível de porta, como pode ser observado na tabela 4.1 e 4.2 para arquiteturas FFT de 8 e 32 pontos respectivamente.

Outro ponto observado na tabela 4.3 é que o método permitiu reduções de 11% e 5% na área e potência respectivamente, quando comparados com a solução de Ahmadinia; Ahmad; Arslan (2007) e 13% e 4% na área e potência, respectivamente, quando comparado com a solução de Macleod (2005). Os resultados comprovam a eficiência da FFT híbrida de 32 pontos que reorganiza os estágios das borboletas da FFT de 8 pontos, através da reutilização de parte da própria arquitetura. Por outro lado, como já observado também na tabela 4.2, o método mostrou um aumento no atraso em relação ao Macleod (2005) e Ahmadinia; Ahmad; Arslan (2007). Este aumento deve-se ao uso dos estágios em série nas borboletas da arquitetura FFT totalmente paralela e, principalmente, devido à decomposição de parte do terceiro estágio em cinco novas constantes que foram implementadas de forma que possam ser reutilizadas.

Avaliando os resultados da tabela 4.3 pela métrica PDP a melhor arquitetura é a híbrida proposta neste trabalho. Pois, seu PDP foi 642 nJ contra 653 nJ apresentado por Macleod (2005) e 647nJ para Ahmadinia; Ahmad; Arslan (2007). Analisando a FFT comportamental o seu PDP ficou em 715 nJ apresentando um resultado acima das demais arquiteturas. Embora, nossos resultados de atraso tenham ficado maiores que a

literatura analisando pela métrica PDP nossos ganhos são melhores que os apresentados pelas demais arquiteturas.

4.3 Decomposição de múltiplos coeficientes em diferentes estágios de FFTs híbridas

A utilização de uma FFT paralela de menos pontos como base para uma FFT híbrida acarreta no problema na demora no processamento das operações. Este problema ocorre devido à reutilização de parte da arquitetura necessitando vários ciclos para determinar todos os coeficientes intermediários.

Uma maneira de contornar este problema utilizando o método de decomposição desenvolvido é decompor mais de um coeficiente em diferentes estágios. Isto é possível aplicando o mesmo método de decomposição apresentado na seção 3.1 associado com a implementação de arquiteturas híbridas visto na seção 4.2. Com isso, foram analisadas arquiteturas FFTs híbridas de N pontos para diferentes arquiteturas FFTs paralelas utilizadas como base.

Nesta implementação foi analisado a possibilidade decompor vários coeficientes em diferentes estágios a fim de reduzir o número de ciclos para calcular os coeficientes intermediários da FFT base. Desta forma, basta verificar se o número de coeficientes decompostos for a metade do número de pontos da FFT base, então é possível determinar todos os coeficientes intermediários de cada estágio com apenas um ciclo de relógio na reutilização da arquitetura. Embora, o número de ciclos dependa do número de coeficientes decompostos esta prerrogativa não é sempre válida. Desta forma, para tratar da decomposição de múltiplos coeficientes foram definidos alguns critérios:

- Uso do método de decomposição de coeficientes apresentado na seção 3.1, onde o coeficiente de menor custo é escolhido;
- Decompor o segundo coeficiente de menor custo a partir o último estágio que já tenha coeficientes decompostos até encontrar o número de coeficiente que permita o cálculo em apenas uma reutilização da FFT base.

Pelos critérios estabelecidos na implementação de FFTs híbridas com as arquiteturas FFTs paralelas de 16, 32 e 64 pontos utilizadas como base foram realizadas a decomposição do coeficiente de menor custo a partir do terceiro estágio. Para a FFT paralela de 16 pontos o coeficiente decomposto do terceiro estágio foi (0.707) e no quarto estágio foi o coeficiente (0,831).

Para a implementação da FFT de 32 pontos tendo como base a arquitetura FFT paralela de 16 pontos no processo de reutilização dos coeficientes intermediários, o cálculo necessita de menos reutilizações da FFT de 8 pontos. Após calcular os 8 coeficientes decompostos relativos aos terceiro estágio, o cálculo dos coeficientes do quarto e quinto estágios são realizados com apenas uma reutilização da arquitetura de base. Isto acontece pelo aumento de coeficientes decompostos que neste caso, são 4 no terceiro e 4 no quarto. Caso fosse apenas decomposto um único estágio, seria necessário reutilizar 2 vezes a FFT base para determinar todos os coeficientes intermediários referentes ao presente estágio. Embora, a decomposição de mais coeficientes acarrete em um pequeno aumento de área na implementação da FFT base, possibilita na redução do número de ciclos de relógio necessários para executar a operação da FFT híbrida.

No processo de decomposição a FFT paralela de 32 pontos teve decomposto o mesmo coeficiente no terceiro, quarto e quinto estágio. Já para a FFT de 64 pontos além do coeficiente (0.707) decomposto no terceiro, quarto e quinto estágios foram

decomposto o coeficiente (0.831) no quinto e no sexto estágio. Alem, do coeficiente (0,555) também decomposto no sexto estágio.

Com base na decomposição em diferentes estágios é reduzido o número de ciclos de reutilização da arquitetura FFT de base.

4.3.1 Resultados para FFTs híbridas maiores

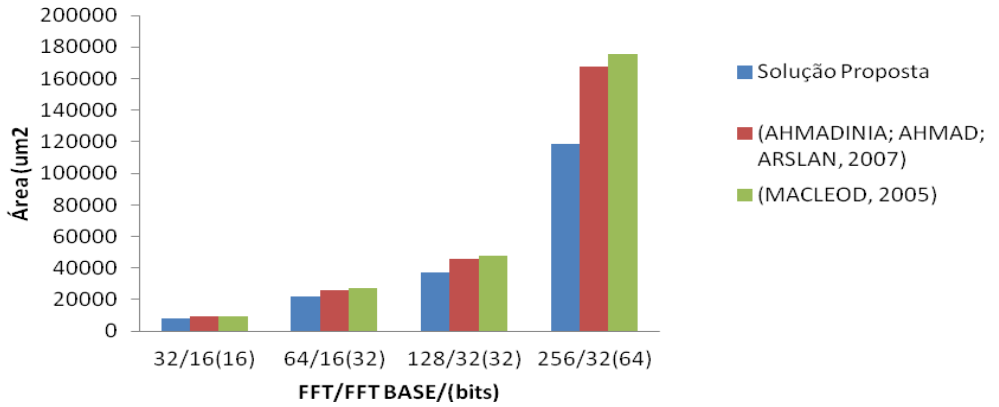
A tabela 4.4 apresenta o número de vezes que é necessário reutilizar a FFT de base para multiplicação dos coeficientes intermediários em cada arquitetura da FFT real e imaginária usadas para base de FFTs híbridas.

Tabela 4.4: Resultados do número de ciclos na multiplicação da FFT base parte real ou imaginário.

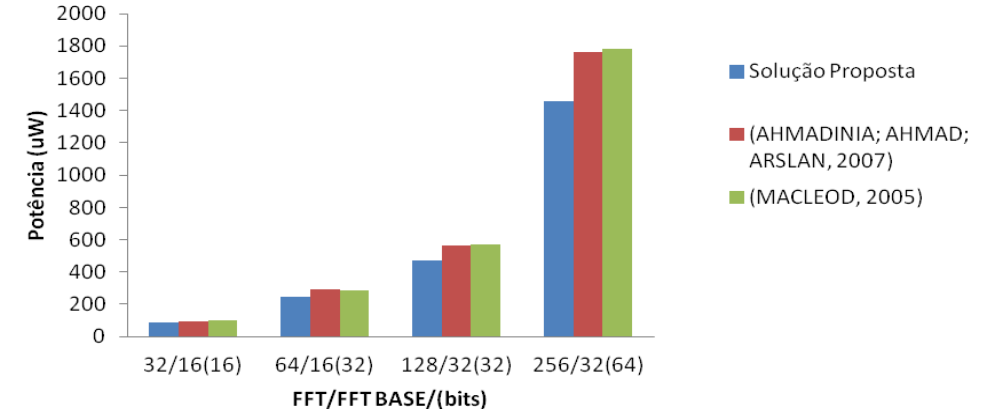
Arquitetura Híbridas ***Pontos/ FFT_BASE(bits)***	**Estágios decompostos**	**Coeficientes decompostos**	**Ciclos para reutilização dos estágios da FFT base para multiplicação de cada coeficiente intermediário**
32/8(16)	1	4	2
32/16(16)	2	8	1
64/16(32)	2	8	1
128/32(32)	3	16	1
256/32(64)	3	16	1

Observando os resultados da tabela 4.4, com exceção da FFT de 32 pontos tendo como base uma FFT de 8 pontos, o processo de reutilização da arquitetura é realizado em apenas um ciclo.

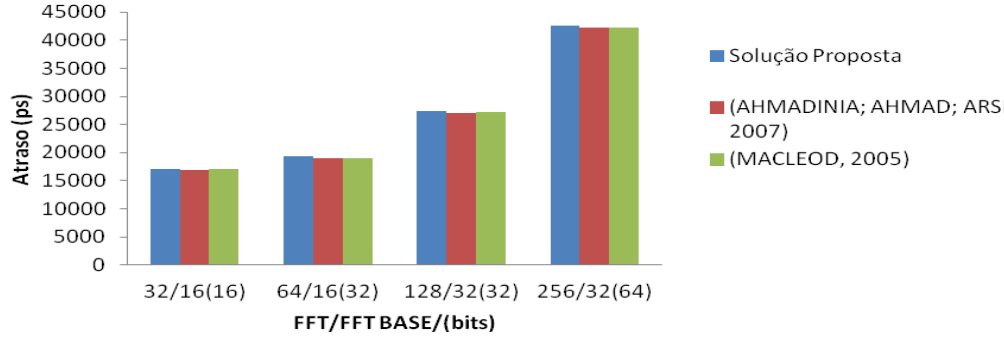
Na figura 4.5 são apresentados os resultados de área, dissipação de potência e atraso obtidos com a síntese lógica realizada com a ferramenta *Design Compiler da Synopsys* na tecnologia CMOS UMC 130nm. A potência foi avaliada pela ferramenta *Design Power* também da Synopsys e com a frequência máxima foi de 100MHZ, em virtude da limitação das FFTs Híbridas dadas pela literatura e a fim de garantir as mesmas condições para comparação dos dados.



(a)



(b)



(c)

Figura 4.5: Resultados da híbrida FFT DIT com FFT paralelas de base usando decomposição de coeficientes na base 2.

Os resultados apresentados na figura 4.5 comprovam a eficiência das arquiteturas híbridas que foram implementadas usando pequenos módulos FFT menores. Na figura 4.5(a)(b) os ganhos variaram de 29% área e 17% em dissipação de potência respectivamente em relação a Ahmadinia; Ahmad; Arslan (2007) e de 32% de área e 18% de dissipação de potência respectivamente em relação a Macleod (2005). Outro aspecto observado nos resultados é que com o aumento do número de pontos das FFTs paralelas utilizadas como base nas operações da FFTs híbridas, aumentaram também o ganho em área e dissipação de potência quando comparada com a literatura. Esta afirmação pode ser comprovada pelos dados da FFT híbrida de 256 pontos que utiliza

como base a FFT de 32 pontos. Portanto, a decomposição dos coeficientes para outros mais adequados, selecionando os coeficientes intermediários mais indicados, permitiu que a FFTs híbridas possam ser implementadas com menor área e menor dissipação de potência quando comparada com as arquiteturas dos demais trabalhos da literatura.

Como visto na figura 4.5(c), o resultado de atraso das arquiteturas híbridas projetadas com o método proposto quando comparadas com a literatura foram um pouco maiores. O que reforça a necessidade de aplicar técnicas para redução de atraso nos próximos trabalhos a serem realizados. No entanto, comprova a redução de área e dissipação de potência.

Outro ponto observado é que embora a frequência utilizada para simulação foi de 100MHz em virtude de condições da literatura para a FFT híbrida de 32 pontos a frequência máxima alcançada foi de 134MHz.

4.3.2 Análise da redução dissipação de potência baseados no uso de pipeline

Para potencializar o aumento na redução de potência foi inserido estágios de *pipeline* nas saídas de cada estágio das FFTs paralelas usadas como base na implementação das FFTs híbridas. A inserção de *pipeline* permite o aumento no processamento com um pequeno aumento de área.

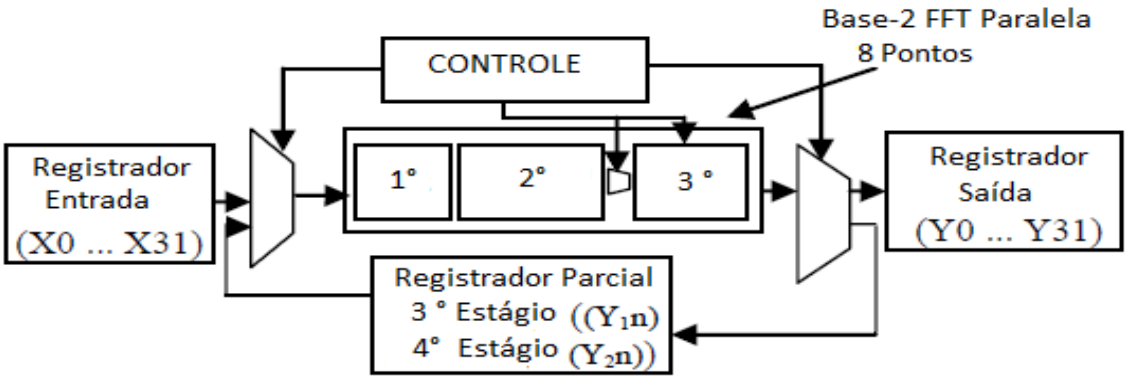
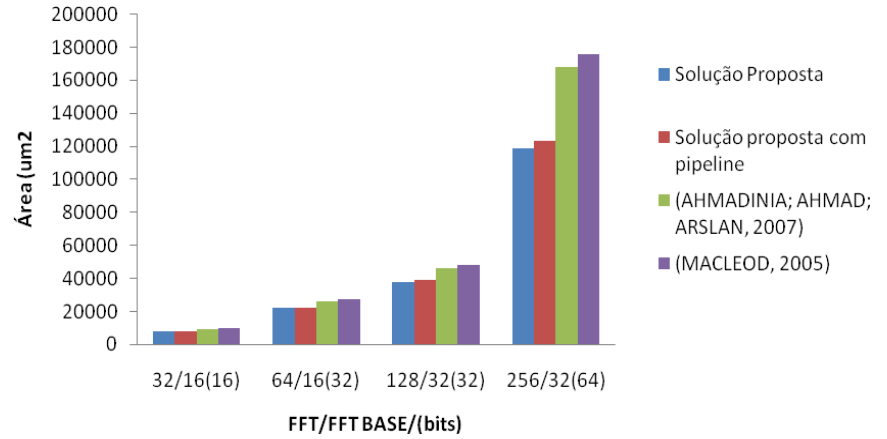
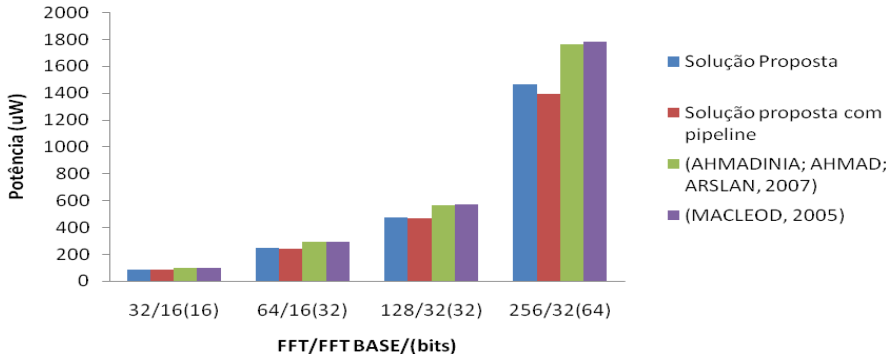


Figura 4.6: Estágios de *pipeline* na estrutura da FFT híbrida.

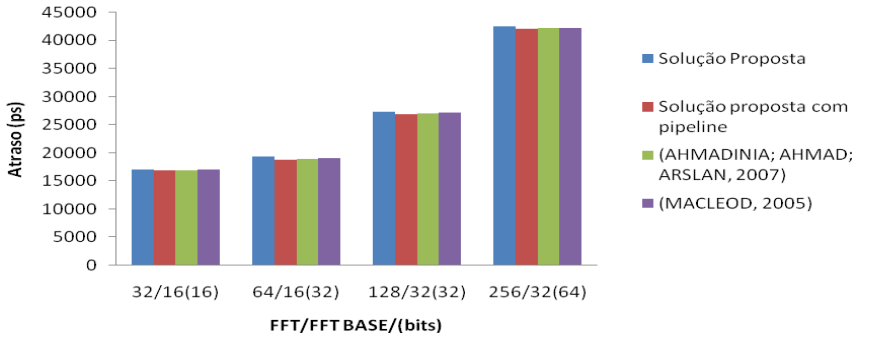
Na figura 4.7 são apresentados os resultados de área, dissipação de potência e área obtidos com a síntese lógica realizada com a ferramenta *Design Compiler da Synopsys* na tecnologia CMOS UMC 130nm. A potência foi avaliada pela ferramenta *Design Power* também da Synopsys. A frequência máxima utilizada foi de 100 MHz, em virtude da limitação das FFTs Híbridas dadas pela literatura e a fim de garantir as mesmas condições para comparação dos dados.



(a)



(b)



(c)

Figura 4.7: Resultados da híbrida FFT DIT com FFT paralelas de base usando decomposição de coeficientes na base 2 e *pipeline*.

Os resultados apresentados na figura 4.7(c) comprovam a melhora do atraso com a inserção de estágios de *pipeline* em relação de Ahmadinia, Ahmad, Arslan (2007) e Macleod (2005). Fazendo uma comparação com Ahmadinia; Ahmad; Arslan (2007) que também usa *pipeline* na sua implementação observa-se que nossa redução foi ainda maior. Isso ocorre porque na implementação proposta o número de coeficientes diminui.

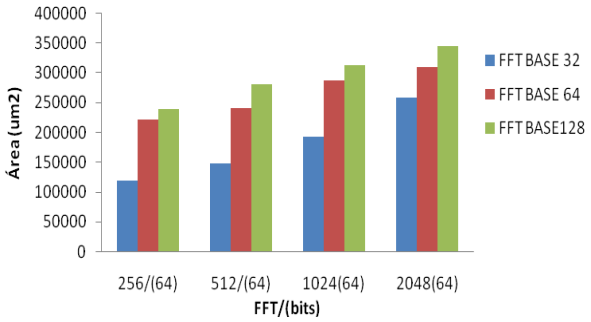
Na figura 4.7(a) embora a área tenha aumentado em média 4% com a inserção dos estágios de *pipeline* em relação aos dados da figura 4.5(a) os resultados ficaram abaixo de Ahmadinia, Ahmad, Arslan (2007) e Macleod (2005). Outro ponto também observado na figura 4.7(b) foi que ocorreu uma redução de 5% na potência dissipada em relação ao não uso de *pipeline*. Isto ocorre devido a redução de ciclos de reuso da FFT

base, pois a inserção de *pipeline* apesar de aumentar a área reduz a necessidade de reuso de hardware.

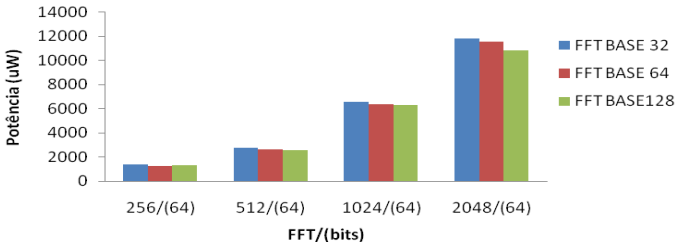
4.3.3 Análise da variação da FFT base sobre a geração da FFT híbrida

Como visto nas seções 4.3.1 e 4.3.2 o método decomposição apresentou resultados melhores que os comparados com a literatura. Na comparação foi utilizado sempre uma relação de 4 vezes entre a FFT híbrida e a FFT base como encontrado na literatura. Para verificar área e potência em função da variação da FFT base utilizada, será realizada a implementação de FFTs híbridas variando de 256 a 2048 pontos implementadas com FFTs bases de 32, 64 e 128 pontos cada.

Novamente para a síntese lógica foi realizada com a ferramenta *Design Compiler da Synopsys* na tecnologia CMOS UMC 130nm. A potência foi avaliada pela ferramenta *Design Power* também da Synopsys e a frequência máxima utilizada foi de 100 MHz.



(a)



(b)

Figura 4.8: Variação das FFTs bases na geração das FFTs híbridas.

A figura 4.8(a) destaca a variação da área na implementação da FFT híbrida com diferentes FFTs base. É observado um crescimento da área para implementação da FFT híbrida com o aumento da FFT base. Este aumento deve-se ao fato de que uma FFT de base com menos pontos quando decomposta precisa gerar todos os coeficientes. E justamente este processo acaba ocasionando um aumento de área. A necessidade da FFT de menos pontos em precisar reutilizar mais vezes a própria arquitetura para calcular todas as operações acarreta no aumento de dissipação de potência, pois aumenta a potência dinâmica do circuito que acaba contribuindo no aumento da potência média total, como visto na figura 4.8(b).

4.4 Análise da redução dissipação de potência baseados no uso de coeficientes reordenados

Para reduzir ainda mais a dissipação de potência das FFTs híbridas foi utilizado a técnica de reordenamento de coeficientes usando a ferramenta ANEDMA (LUZ; COSTA; AGUIAR, 2010).

Em nossa implementação FFT híbrida, os coeficientes ou conjunto deles são sequencialmente executados em cada conjunto de borboletas paralelas de seus estágios. No entanto, em cada conjunto de operações da FFT paralela, os resultados do cálculos são independentes da sequência de cálculo de coeficientes de multiplicação. Assim, os coeficientes podem ser rearranjados de modo a que a distância de *Hamming* entre palavras consecutivas pode ser reduzido. Para uma FFT híbrida que utiliza apenas o número de coeficientes relativos a FFT paralela para executar as operações o reordenamento deve ser realizado por parcelas e deve ser feito a partir dos valores de entrada.

Na tabela 4.5 são apresentados os ciclos de utilização de coeficientes responsáveis pela execução da FFT de 32 pontos DIT através da FFT DIT de 8 pontos utilizada como base. A partir das operações do 2° ciclo é realizado o reordenamento dos coeficientes. Pois, como a FFT de base é paralela para manter a coerência das multiplicações e evitar erros, somente é possível fazer o reordenamento quando é realizado operações sequenciais.

Tabela 4.5: Coeficientes reordenados na FFT DIT base de 8 pontos para operações da FFT híbrida de 32 pontos.

Clicos de utilização coeficientes	**Coeficientes de sem reordenamento**	**Coeficientes de com reordenamento**
1°	W^0_1, W^0_2,W^1_2, W^0_4, W^1_4,W^2_4 e W^3_4	W^0_1, W^0_2,W^1_2, W^0_4, W^1_4,W^2_4 e W^3_4
2°,3°, 4°,5°, 6°, 7°, 8°, 9°	W^0_8, W^1_8,W^2_8, W^3_8, W^4_8,W^5_8 e W^6_8, W^7_8	W^0_8, W^5_8, W^2_8, W^6_8, W^3_8, , W^1_8, W^7_8, W^4_8
10°, 11°,12°, 13°,14°, 15°, 16°, 17°, 18°, 19°, 20°, 21°, 22°, 23°, 24°, 25°	W^0_{16},W^1_{16}, W^2_{16}, W^3_{16}, W^4_{16}, W^5_{16}, W^6_{16}, W^7_{16}, W^8_{16},W^9_{16}, W^{10}_{16}, W^{11}_{16}, W^4_{16}, W^{12}_{16}, W^{13}_{16}, W^{14}_{16}, W^{15}_{16}	W^0_{16}, W^3_{16}, W^1_{16}, W^{15}_{16}, W^2_{16}, W^{14}_{16}, W^4_{16}, W^5_{16}, W^6_{16}, W^{10}_{16}, W^7_{16}, W^9_{16}, W^{11}_{16}, W^4_{16}, W^{12}_{16}, W^{13}_{16}, W^8_{16}

Através da tabela 4.5 é possível verificar a variação do conjunto de coeficientes na FFT base pelo ciclo de utilização. Por exemplo, após o reordenamento dos coeficientes, o 3° ciclo que é executado na estrutura decomposta deixou de executar na ordem o coeficiente W^1_8 para executar o coeficiente W^5_8. Isso deve-se a redução da distância de *hamming* após o reordenamento com a ferramenta Anedma.

Após, o reordenamento, foi realizada novamente a síntese. A tabela 4.6 apresenta os resultados de reordenamento dos coeficientes das arquiteturas implementadas com

variação das FFTs bases na geração das FFTs híbridas nas mesmas condições que obtidas na figura 4.8.

Tabela 4.6: Resultados de potência média após reordenamento de coeficientes.

Pontos/ (bits)	**FFT BASE 32**	**FFT BASE 32 reordenados (%)**	**FFT BASE 64**	**FFT BASE 64 reordenados (%)**	**FFT BASE 128**	**FFT BASE 128 reordenados (%)**
256/(64)	1392	-1,87	1271	-1,02	1315	-0,77
512/(64)	2745	-2,48	2619	-1,53	2546	-1,15
1024/(64)	6557	-6,79	6346	-2,62	6312	-2,19
2048/(64)	11783	-11,00	11541	-6,64	10786	-4,27

A tabela 4.6 apresenta claramente a redução de potência média após o uso de reordenamento de coeficientes com a ferramenta ANEDMA. Observou-se que na implementação de uma FFT híbrida com a mesma FFT base, à medida que cresce o número dessas FFT híbridas ocorre uma maior redução de dissipação de potência. Isto ocorre devido o aumento de coeficientes da FFT híbrida que são utilizados no reuso no conjunto dos estágios da FFT base. Quanto maior o reuso da FFT base maior a possibilidade de reduzir a distância de *Hamming* e consequentemente maior à redução de potência de chaveamento que contribui na redução da potência média total. Por outro lado, observou-se que com o aumento da FFT base ocorreu uma redução do ganho de redução de potência em função do reordenamento. Isto ocorre, porque com aumento da FFT base diminui o conjunto de coeficientes que podem ser reordenados sequencialmente pela ferramenta, ou seja, diminui a possibilidade de reduzir a distância de *Hamming* e consequentemente à possibilidade de redução de potência.

4.5 Resultados de erro das FFTs Híbridas

Para validar o funcionamento das arquiteturas FFTs híbridas foi desenvolvido um sistema de análise de erro decorrente das respostas das operações das FFTs implementadas. Através da ferramenta Matlab foi criado uma rotina que calcula os resultados da FFT em ponto fixo tendo como entrada os mesmos vetores utilizados para simulação em cada FFT e o resultado de saída das FFTs simuladas logicamente com a ferramenta ModelSIM.

A figura 4.9 apresenta o diagrama de verificação de erro desenvolvido para as FFTs implementadas. A verificação é realizada pela análise dos valores de saída fornecidos das ferramentas Modelsim e pelo algoritmo de cálculo da FFT do Matlab após serem excitados por vetores de entrada.

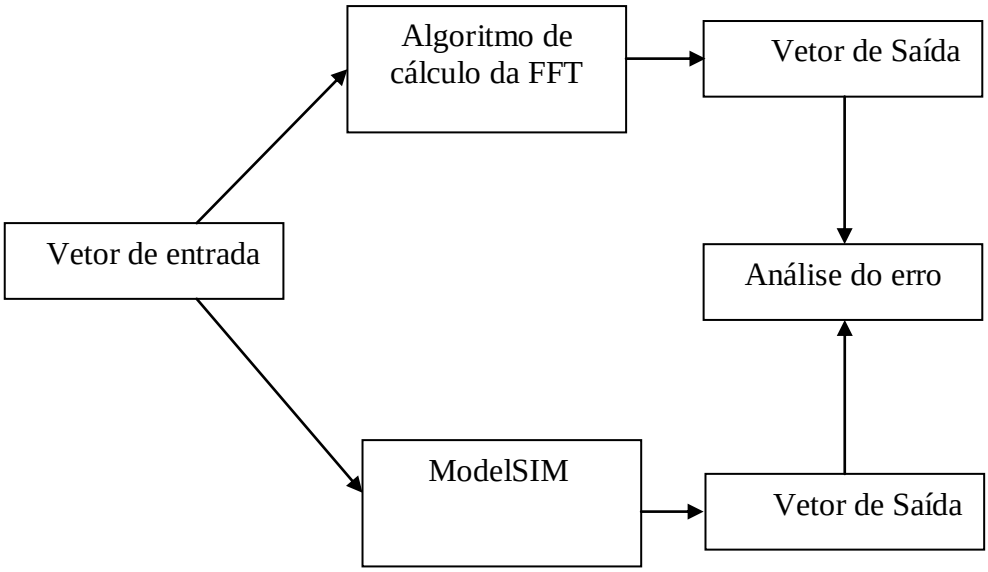


Figura 4.9: Esquema para análise de erro das FFTs híbridas implementadas.

O erro médio calculado é determinado pela equação 4.3.

$$Erro_{med} = \frac{1}{p}\sum_{i=1}^{p}\left(\frac{R_{Mi} - R_{Si}}{R_{Mi}}\right) \tag{4.3}$$

Sendo:

R_M a resposta de saída do Matlab, R_S a resposta de saída do Modelsim com o modelo proposto e *p* o número de amostras.

A tabela 4.7 apresenta os resultados percentuais do erro gerado nas FFTs híbridas implementadas com o método proposto e simuladas com até 5000 vetores aleatórios de entrada gerados pela ferramenta Matlab. O número de vetores aleatórios varia devido a não convergência na ferramenta de síntese lógica *Design Compiler* da Synopsys.

Tabela 4.7: Resultados do erro médio gerado pelas FFTs híbridas implementadas.

Arquitetura Híbridas Pontos/FFT_BASE(bits)	**Vetores aleatórios**	**Erro Médio %**
32/8(16)	5000	0,81
32/16(16)	5000	0,32
64/16(32)	1000	1,6
128/32(32)	1000	1,47
256/32(64)	500	2,33
256/64(64)	500	2,15
256/128(64)	500	2,02
512/32(64)	500	3,27
512/64(64)	500	3,11
512/128(64)	500	2,66
1024/32(64)	500	3,92
1024/64(64)	500	3,66
1024/128(64)	500	3,36
2048/32(64)	500	4,63
2048/64(64)	500	4,54
2048/128(64)	500	4,31

Os resultados apresentados na tabela 4.7 mostram um erro mínimo de 0,32 % para as operações realizadas pela FFT híbrida de 32 pontos tendo como base uma FFT paralela de 8 pontos com 16 bits de largura quando submetido a 5000 vetores aleatórios de entrada. No entanto, observou-se um erro de 4,63% para a FFT de 2048 pontos tendo como base uma FFT de 32 pontos com largura de 64 bits. O erro em todas as FFTs decorre principalmente pelo fato que a FFT híbrida ao passo de realizar as operações nos módulos de base paralela necessita fazer truncamento dos bits mais significantes (*Most Significant Bit*) para sua reutilização. E é justamente nos truncamentos é que o erro se propaga à medida que se repete a reutilização da arquitetura base. Desta forma, a medida que é utilizada uma FFT de base com menor pontos, maior é a incidência deste erro. Deve-se ressaltar que o número de o número de vetores utilizados diminui com o aumento da complexidade da FFT. Isto ocorre a fim de garantir a convergência nas ferramentas de síntese.

4.6 Resumo

Neste capítulo foi apresentado a implementação de FFTs híbridas usando o modelo de decomposição apresentado no capítulo 3. Neste conceito, foi mostrado que as arquiteturas FFTs híbridas implementadas com o método apresentaram resultados significantemente melhores quando comparados com a literatura, pois permitiu implementar arquiteturas FFTs maiores com menor dissipação de potência e área. Devido à regularidade do método foi aplicado o reordenamento dos coeficientes da FFT base na execução da FFT híbrida que permitiu obter resultados de área e dissipação de potência melhores quando comparados com obtidos sem reordenamento. O erro médio devido o truncamento dos dados encontrados nas FFTs híbridas não comprometeu o seu funcionamento. Embora, dependendo para aplicação deva ser considerado e tratado.

5 CONCLUSÕES

O aumento no número de dispositivos móveis que utilizam operações complexas, como a FFT, bem como a necessidade de que estes equipamentos apresentem a menor dissipação de potência a fim de garantir o seu maior uso sem precisar recarregar a bateria, tem motivado o estudo de técnicas que visam à redução de componentes e de dissipação para que não comprometam o atraso do circuito.

Neste trabalho, foi apresentada uma técnica que permite realizar a otimização de área e dissipação de potência através da decomposição dos coeficientes trigonométricos responsáveis pelas multiplicações dos coeficientes na borboleta da FFT. A técnica possibilita reutilização de componentes, o que permite que uma arquitetura FFT de menos pontos consiga realizar as operações de uma arquitetura de mais pontos. A técnica de decomposição associada à implementação de arquiteturas com o uso de CMM e métrica de porta para CSA apresentou melhores resultados de área e dissipação de potência que os apresentados por MacLeod (2005), Ahmadinia; Ahmad; Arslan (2007), Qureshi; Gustafsson, (2009), e Aksoy *et al*. (2012).

O trabalho de redução de componentes desenvolvido pela decomposição de coeficientes trigonométricos foi analisado concomitantemente com a redução de dissipação de potência. A implementação dos blocos desses componentes foi baseada em estruturação em nível de porta para somadores/subtratores RCA e CSA. A implementação ao nível de porta lógica permitiu substituir somadores/subtratores mais complexos por menos complexos, quando possível. Assim, possibilitou a implementação das arquiteturas com menor área e dissipação de potência, mantendo a performance de operação da arquitetura.

Os ganhos na redução de componentes somadores/subtratores conseguido em relação ao uso de CMM simples apresentado por Aksoy *et al.* (2012) ocorrem em função de que no método proposto a decomposição utiliza, além dos coeficientes a serem decompostos, coeficientes auxiliares. Estes coeficientes auxiliares são constantes intermediárias cujos valores são limitados pelo menor e maior valor do conjunto dos coeficientes da FFT a serem decompostas.

Observou-se também que, no método apresentado para decomposição de coeficientes trigonométricos, é possível obter uma redução de área e dissipação de potência simplesmente pelo reaproveitamento de parte dos componentes decompostos. Para isso, foi necessário desenvolver um sistema de multiplexadores que controlasse corretamente as operações a serem realizadas no novo conjunto de coeficientes gerados.

O erro nas operações das arquiteturas híbridas construídas após a decomposição não comprometeu sua funcionalidade esperada. Entretanto, observa-se nessas arquiteturas que, quando uma FFT de menor pontos é usada para implementar uma FFT de maior pontos, esta pode comprometer os ganhos de redução de potência conseguidos com a decomposição. Isso ocorre porque, quando é realizado o processo de reutilização, são

necessários mais ciclos para realizar as operações da FFT. Como a potência de chaveamento está diretamente vinculada ao número de ciclos, o aumento de ciclos causa um aumento da dissipação de potência do circuito. Assim, em termos de redução de componentes e dissipação de potência, é coerente utilizar uma relação de 4 vezes entre a FFT maior e menor.

Para potencializar ainda mais a otimização de potência, foi analisado o reordenamento dos coeficientes trigonométricos das operações das borboletas da FFT. Como a potência de chaveamento está relacionada com a atividade de chaveamento do circuito, e esta depende diretamente da distância de *Hamming* entre os coeficientes responsáveis pelas operações de multiplicação das borboletas da FFT, o seu reordenamento permitiu a sua redução. Nesta etapa, foi utilizada a ferramenta Anedma, cuja função é encontrar qual é a menor distância de *Hamming* para a sequência de coeficientes trigonométricos responsáveis pela multiplicação na borboleta da FFT. Neste aspecto, foi observado que, com o aumento do número de coeficientes da FFT, foi possível obter uma maior redução de dissipação de potência devido à maior possibilidade de reordenamento dos coeficientes pela ferramenta.

Uma significativa contribuição apresentada neste trabalho foi à inserção da técnica CMM associada com otimização ao nível de porta para somadores CSA (GHISSONI et al., 2010-2011) que até então eram tratados com apenas com dois operandos (AKSOY et al., 2008). Este avanço permitiu que as arquiteturas FFTs desenvolvidas não apresentassem problemas pertinentes ao atraso e mantivessem o mesmo nível de performance quando otimizada área e dissipação de potência.

No trabalho de Qureshi; Gustafsson (2009), onde foi avaliado o desempenho de método de reconfiguração de multiplexadores para redução de componentes, os autores não conseguiram concluir que seu método pudesse reduzir a dissipação de potência em sínteses lógicas ou físicas realizadas por ferramentas comerciais, pois apenas apresentaram a análise de redução do número de componentes. No entanto, como visto neste trabalho, isso não invalida a ideia de auxílio de outras técnicas para que seja melhorada a otimização de operadores aritméticos em determinadas estruturas. Nesta tese de doutorado, pode-se comprovar que a redução de componentes permite reduzir também a dissipação de potência, mas sempre se estiver com o propósito para isso.

Por outro lado, foi evidenciada a existência de uma falha nos métodos apresentados na literatura responsáveis por otimizar uma arquitetura FFT nos três eixos: área, atraso e dissipação de potência. Isso se deve praticamente ao fato desses métodos utilizarem apenas uma técnica específica. De certa forma, os métodos existentes, com pequenas exceções, desprezam indiretamente que o uso de várias técnicas simultaneamente possibilita aumentar a diminuição de área e dissipação de potência sem comprometer significativamente o aumento do atraso. A fim de suprir essa falha, nesta tese de doutorado foram utilizadas, além da técnica de decomposição de coeficientes, a técnica CMM, métrica de porta e a reutilização de parte da própria arquitetura. Desta maneira, é possível trabalhar simultaneamente com os diferentes problemas causados pelo relacionamento dos três requisitos de otimização analisados (atraso, área e dissipação de potência).

Portanto, esta tese de doutorado serviu de instrumento de otimização de arquiteturas FFTs na base 2 no âmbito da análise da área e dissipação de potência mantendo o compromisso com a performance e funcionalidade.

5.1 Trabalhos Futuros

Embora esta tese de doutorado tenha apresentado uma técnica que permita otimização de área e dissipação de potência baseada na decomposição de coeficientes trigonométricos, deve-se aprimorar a otimização do atraso. Um dos passos a seguir será a redução do caminho de dados através do uso de somadores rápidos. Outro passo será a inserção de vários estágios de *pipeline* nas arquiteturas híbridas. Nesta linha, é interessante testar algoritmos existentes na literatura que já fazem isso.

Outra atividade interessante a ser desenvolvida será a aplicação da decomposição de coeficientes trigonométricos em diferentes topologias de arquiteturas FFTs, bem como de DCTs. Assim, será possível analisar o efeito da metodologia apresentada sobre as demais arquiteturas, como já estudado por Salbego; Ghissoni; Rahmeier (2012).

A aplicação da técnica de decomposição nas arquiteturas DCT 2D, por exemplo, permitirá analisar o comportamento de múltiplas constantes com a verificação da possibilidade do compartilhamento de componentes entre os diferentes níveis de multiplicação que podem ser reutilizados.

Um aspecto também a ser abordado nos próximos trabalhos é a verificação da redução de componentes para DCTs híbridas. Embora a arquitetura seja diferente da FFT, a reutilização de parte da própria arquitetura pode ser realizada da mesma forma. No entanto, para esta arquitetura os limites laterais dos valores desses coeficientes devem ser mais bem estabelecidos, visto que o equacionamento desta transformada é resultante de uma multiplicação de mais de um coeficiente trigonométrico.

Espera-se reduzir consideravelmente a dissipação de potência total das arquiteturas através da redução da potência de chaveamento de *glitchings* obtida pelo reordenamento dos novos coeficientes decompostos nessas novas arquiteturas. Entretanto, esta redução só se fará condizente se as arquiteturas apresentarem um número grande de coeficientes.

Também será analisado o uso deste sistema de decomposição para fins genéricos. Assim, será possível verificar o comportamento dos novos conjuntos de coeficientes gerados a partir do conjunto original na implementação de qualquer sistema linear de coeficientes.

De certa maneira, as atividades a serem realizadas nos próximos trabalhos devem sempre analisar o impacto da variação de área e dissipação de potência para que não comprometa o atraso do circuito.

REFERÊNCIAS

AKSOY, L.; COSTA, E.; FLORES, P.; MONTEIRO, J. Optimization of Area in Digital FIR Filters using Gate-Level Metrics. In: DESIGN AUTOMATION CONFERENCE, 2007. **Proceedings...** DAC '07. 44th ACM/IEEE, [S.l.:s.n], 2007. p. 420-423.

AKSOY, L.; GUNES, E.; FLORES, P. An Exact Breadth-First Search Algorithm for the Multiple Constant Multiplications Problem. In: IEEE NORCHIP CONFERENCE, 2007. **Proceedings...,** [S.l.:s.n], 2008. p. 41-46.

AKSOY, L.; COSTA, E.; FLORES, P.; MONTEIRO, J. Optimization Algorithms for the Multiplierless Realization of Linear Transforms, **ACM Transactions on Design Automation of Electronic Systems**, [S.l.:s.n], v.17 , n.0, jan 2012.

AHMADINIA, A.; AHMAD, B.; ARSLAN, T. System Level Modelling of Reconfigurable FFT Architecture for System-on-Chip Design, In: SECOND NASAIESA CONFERENCE ON ADAPTIVE HARDWARE AND SYSTEMS, 2007. **Proceedings...,** [S.l.:s.n], 2007. p. 169-175.

BENINI, L. et al. Analysis of hazard contribution to power dissipation for CMOS IC´s. In: ACM/IEEE INTERNATIONAL WORKSHOP ON LOW POWER DESIGN**,** 24, 1994, Napa. **Proceedings** , Napa: ACM Press, 1994. p. 27-32.

BLUESTEIN, L. I. A Linear Filtering Approach To The Computation Of The Discrete Fourier Transform. **NORTHEAST ELECTRONICS RESEARCH AND ENGINEERING MEETING RECORD 10**, [S.l.:s.n], 1968. p. 218–219.

BRUUN, G. Z-Transform DFT filters and FFTs**. IEEE Trans. on Acoustics,** Speech and Signal Processing (ASSP) 26, [S.l.:s.n], p.56–63, 1978.

CAPPELLO, P.; STEIGLITZ, K. Some Complexity Issues in Digital Signal Processing, **IEEE Transactions on Acoustics, Speech, and Signal Processing**, [S.l.], v.32, n. 5, p.1037–1041, 1984.

CHO, H.; KIM, M.; KIM, D.; KIM, J. $R2^2$SDF FFT implementation with coefficient memory reduction scheme, In: VEHICULAR TECHNOLOGY CONF., 2006. **Proceedings…,** [S.l.:s.n], 2006. p. 1-4.

CIRIT, M., Estimating dynamic power consumption of CMOS circuits. In: IEEE INTERNATIONAL CONFERENCE COMPUTER AIDED DESIGN, 1987, Santa Clara. **Proceedings…** Santa Clara (USA): IEEE Computer Society Press, 1987. p. 534-537.

CHOO, H.; MUHAMMAD, K.; ROY, K. Complexity Reduction of Digital Filters Using Shift Inclusive Diferential Coeficients. **IEEE Transactions on Signal Processing**, 52(6), [S.l.:s.n], p. 1760-1772, Jun 2004.

COOLEY, J. W.; TUKEY, J. W. An Algorithm for the Machine Calculation of Complex Fourier Series**. Mathematics of Computation**, [S.l.], v.19, n.90, 1965. p. 297–301.

COSTA, E. **Operadores Aritméticos de Baixo Dissipação para Arquiteturas de Circuitos DSP.** 2000. f. 193. Tese (Doutorado em Ciência da Computação)– Instituto de Informática, UFRGS, Porto Alegre.

COSTA, E.; FLORES, P.; MONTEIRO, J. Exploiting General Coefficient Representation for the Optimal Sharing of Partial Products in MCMs, In**:** SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, 2006. **Proceedings…** [S.l.:s.n], 2006. p. 161–166.

COSTA, E.; FLORES, P.; MONTEIRO, J. Maximal Sharing of Partial Terms in MCM under Minimal Signed Digit Representation, In: IEEE EUROPEAN CONFERENCE ON CIRCUIT THEORY AND DESIGN. **Proceedings…,** [S.l.:s.n], 2005. p. 221–224.

DAS, S., Design automation techniques for datapath circuits, Ph.D. Dissertação, University of Colorado, 2007.

DEMPSTER, A. G.; MACLEOD, M. D. Constant integer multiplication using minimum adders, Proc. Inst. Electr. Eng.—Circuits, Devices Systems, [S.l.:s.n], vol. 141, n. 5, p. 407‑413, Oct. 1994.

DEMPSTER A. G.; MACLEOD, M. D. General algorithms for reduced-adder integer multiplier design**, Electronics Letters**, [S.l.:s.n], v.31, n.21, p. 1800–1802, 1995.

DEMPSTER A. G.; MACLEOD, M. D. Use of Minimum-adder Multiplier blocks in FIR digital filters, **IEEE Trans. Circuits Syst**. II, [S.l.:s.n], v.42, n.9, p. 569–577, 1995.

DEMPSTER, A.; DEMIRSOY, S.; KALE, I. Designing Multiplier Blocks with Low Logic Depth, **PROCEEDINGS OF IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS**, [S.l.:s.n], p. 773–776. 2002.

DEMIRSOY, S.; KALE, I.; DEMPSTER, A. Synthesis of reconfigurable multiplier blocks−Part II: Algorithm, In: **PROC. IEEE INT. SYMP. CIRCUITS SYST**., [S.l.:s.n], v.1, 2005. p. 540‑543.

DIMITROV, V.; IMBERT, L.; ZAKALUZNY, A. Multiplication by a Constant is Sublinear, **PROCEEDINGS OF THE IEEE SYMPOSIUM ON COMPUTER ARITHMETIC**, [S.l.:s.n], 2007. p. 261–268.

GHISSONI, S.; COSTA, E.; MONTEIRO, J.; REIS, R. Efficient Multiplication for FFT Based on Twiddle Factor Decomposition. In: **19th IEEE ICECS**, Sevilha (2012), 2012. p. [s.n].

GHISSONI, S.; COSTA, E.; MONTEIRO, J.; REIS, R. Combination of Constant Matrix Multiplication and Gate-Level Approaches for Area and Power Efficient Hybrid Radix-2 DIT FFT Realization. In: **27TH SOUTH SYMPOSIUM ON MICROELECTRONICS**, Santo Ângelo (2012), 2012. p. [s.n].

GHISSONI, S.; COSTA, E.; MONTEIRO, J.; REIS, R. Reducing Area and Power in FFT Architectures Using Twiddle Factor Decomposition Approach. In: **18TH IEEE ICECS**, Beirute (2011), 2011. p. 567-570.

GHISSONI, S.; COSTA, E.; MONTEIRO, J.; REIS, R. Optimization area and power of radix-2 decimation in time (DIT) FFT implementation based on multiple constant multiplication in the stages. In: **26TH SOUTH SYMPOSIUM ON MICROELECTRONICS**, Gramado (2011), 2011. p. s/n.

GHISSONI, S.; COSTA, E.; LAZZARI, C.; MONTEIRO, J.; AKSOY, L.; REIS, R. Radix-2 Decimation in Time (DIT) Implementation Based on a Matrix-Multiple Constant Multiplication Approach. In: **17TH IEEE ICECS**, Athens (2010), 2010. p. 859–862.

GHISSONI, S.; COSTA, E.; MONTEIRO, J.; REIS, R. Decimation in time (DIT) FFT implementation based on multiple constant multiplication approach. In**: 25TH SOUTH SYMPOSIUM ON MICROELECTRONICS**, Porto Alegre 2010. p. s/n.

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.

GOOD, I. J. The interaction algorithm and practical Fourier analysis. J. R. Statist. Soc. B 20, [S.l.], p.361–372, 1958.

GUSTAFSSON, O.; DEMPSTER, A. G.; L. WANHAMMAR, Extended results for minimum-adder constant integer multipliers, In: **IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS** (ISCAS '02), Scottsdale, Ariz, USA, v.1, p. 73–76, 2002.

GUSTAFSSON, O.; WANHAMMAR, L. ILP Modeling of the Common Subexpression Sharing Problem, **PROCEEDINGS OF INTERNATIONAL CONFERENCE ON ELECTRONICS, CIRCUITS AND SYSTEMS**. 2002. p. 1171–1174.

GUSTAFSSON O.; DEMPSTER, A. G.; JOHANSSON, K.; MACLEOD, M. D. ; WANHAMMAR, L. Simplified design of constant coefficient multipliers, **CIRCUITS, SYSTEMS AND SIGNAL PROCESSING**, [S.l.:s.n], v.25, n.2, 2006. p. 225–251.

GUSTAFSSON, O. A difference based adder graph heuristic for multiple constant multiplication problems. In: **PROC. IEEE INT. SYMP. CIRCUITS AND SYSTEMS**, New Orleans, LA, p. 1097‑1100, 2007.

GUSTAFSSON, O. Lower Bounds for Constant Multiplication Problems, **IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing**, [S.l.:s.n], 54(11), p. 974–978, 2007.

HAN, W.; ARSLAN, T.; ERDOGAN, A. T.; HASAN, M. High-performance low-power FFT cores, **ETRI Journal**, [S.l.:s.n], v.30, n.3, p. 451‑460, June 2008.

HAN, J. H.; PARK, I.C. FIR Filter Synthesis Considering Multiple Adder Graphs for a Coefficient, **IEEE Transactions on Computer-Aided Design of Integrated Circuits**, [S.l.:s.n], 27(5), p. 958–962. 2008.

HASAN M.; ARSLAN, T.; THOMPSON, J. S. A Novel Coefficient Ordering based Low Power Pipelined Radix-4 FFT Processor for Wireless LAN Applications, **Transactions on Consumer Electronics**, [S.l.:s.n], v.49, n.1, Fev. 2003.

HASAN M.; ARSLAN, T. Scheme for reducing size of coefficient memory in FFT processor, **Electronics Letters**, [S.l.:s.n], v.38, n.4, p. 163‑164, Fev 2007.

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

HORN, R. A.; JOHNSON, C. R. Topics in Matriz Analysis, Cambridge University Press, Cambridge, MA, 1991.

IEEE std 802.16^{TM}-2004 (Revision of IEEE Std 802.16-2001), IEEE Standard for Local and metropolitan area networks, 3 Park Avenue, New York, NY 10016-5997, USA.

JIA, L.; GAO,Y.; ISOAHO, J. ; TENHUNEN, H. A New VLSI-Oriented FFT Algorithm and Implementation, **PROC. OF ELEVENTH ANNUAL IEEE INT'L ASIC CONFERENCE**, [S.l.:s.n], p. 337-341, 1998.

JING, HE; LANJUAN, MA; XINYU, XU. Design of A Length-Variable FFT Processor, Multimedia Technology (ICMT), 2010 International Conference on, p 1-4, 2010.

KANG, H.J.; PARK, I.C. FIR Filter Synthesis Algorithms for Minimizing the Delay and the Number of Adders, **IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing**, [S.l.:s.n], 48(8), p.770–777. 2001.

KARKALA, V.; WANSTRATH, J.; LACOUR, T.; KHATRI, P. S. Efficient Arithmetic Sum-of-Product (SOP) based Multiple Constant Multiplication (MCM) for FFT, **PROCEEDINGS OF IEEE ICCAD**, [S.l.:s.n], p. 735–738, 2010.

KIM, T.; JAO, W.; TJIANG, S. Circuit Optimization using Carry-Save-Adder Cells, **IEEE Transactions on Computer-Aided Design of Integrated Circuits**, [S.l.:s.n], 17(10), p. 974–984. 1998.

KIM, N. et al. Leakage Current: Moore's Law Meet Static Power, **IEEE Computer Special Issue on power and Temperature-Aware Computer**, [S.1], n.36, v.12, p.68-75, DEC 2003.

KOUSHIK, M.; ECKHARD, G.; ULRICH, J., A 64-Point Fourier Transform Chip for high-speed wireless LAN application using OFDM, **IEEE Journal of Solid-State Circuits**, [S.l.:s.n], p. 484‑493, 2004.

LEE, S.; KIM, DUK-BAI; PARK, SIN-CHONG. Power-efficient design of memory based FFT processor with new addressing scheme, In: **PROC. INT. SYMP. COMMUNICATIONS AND INFORMATION TECHNOLOGY**, [S.l.:s.n], 2004. p. 678‑681.

LEE, HYUN-YONG; PARK, IN-CHEOL. Balanced binary-tree decomposition for area-efficient pipelined FFT processing, **IEEE Transactions on Circuits and Systems-I**, [S.l.:s.n], v.54, n.4, p. 889‑900, April 2009.

LEFEVRE, V. **Multiplication by an Integer Constant**, Technical report, Institut National de Recherche en Informatique et en Automatique. 2001.

LIN, H.; SU, K.; LI, C.; ARGELICH, J., Inc WMaxSatz, Max-SAT Evaluation, 2008.

LUZ, A.; COSTA, E.; AGUIAR, M. Ordering and Partitioning of Coefficients Based on Heuristic Algorithms for Low Power FIR Filter Realization. 23RD SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN – SBCCI, [S.l.:s.n], p. 180-185, 2010.

LUZ, A.; COSTA, E.; AGUIAR, M. Exploring the Use of Heuristic-Based Algorithms for the Ordering and Partitioning of Coefficients for Low Power Efficient FIR Filters

Realization. 24RD SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN – SBCCI, [S.l.:s.n], p. 91-96, 2011.

LUZ, A. G. ; Costa, E. ; GHISSONI, S. . Reducing Power Consumption in FFT Architectures by Using Heuristic-Based Algorithms for the Ordering of the Twiddle Factors. In: 3RD IEEE LATIN AMERICAN SYMPOSIUM ON CIRCUITS AND SYSTEMS, 2012, Solidaridad. Latin American Symposium on Circuits and Systems, [S.l.:s.n], 2012.

MACLEOD, M. D. Multiplier less Implementation of Rotators and FFTs , **EURASIP Journal on Applied Signal Processing**, [S.l.:s.n], p 2903-2910, Sept 2005.

MARTIN, Kenneth W. Digital integrated circuit design, New York: Oxford University Press, 2000.

MARTINS, J.B dos S. **Estimativa de Capacitâncias e Dissipação de Potência em Circuitos Combinacionais CMOS no Nível Lógico**. 2000. f. 112. Tese (Doutorado em Ciência da Computação)– Instituto de Informática, UFRGS, Porto Alegre.

MELEK P. L.A. Operação De Circuitos Lógicos Cmos De (Ultra)-Baixo Dissipação, 2004. Dissertação de Mestrado em engenharia elétrica da Universidade Federal de Santa Catarina, 2004.

OH, JUNG-YEOL; MYOUNG-SEOH LIM, Fast Fourier Transform Processor Based on Low-power and Area-efficient Algorithm, **IEEE ASIA-PACIFIC CONFERENCE ON ADVANCED SYSTEM INTEGRATED CIRCUITS** (AP-ASIC2004), [S.l.:s.n], 2004. p. 198-201.

OH, J. E.; LIM, M. S. New radix-2 to the 4th power pipeline FFT processor, **IEICE Trans. Electron**, [S.l.:s.n], vol. E88-C, n.8, p. 1740‑1764, Aug. 2005.

PARK, I.C.; KANG, H.J. Digital Filter Synthesis Based on Minimal Signed Digit Representation, **PROCEEDINGS OF DESIGN AUTOMATION CONFERENCE**, [S.l.:s.n], 2001. p. 468–473.

PROAKIS, J. G.; MANOLAKIS, D. G., Digital Signal Processing. Prentice Hall, Third Edition, 1995.

QURESHI, F.; GUSTAFSSON, F. O. Low-complexity reconfigurable complex constant multiplication for FFTs: **IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS. ISCAS 2009.** [S.l.:s.n], 2009. p. 1137-1140.

QURESHI, F.; GUSTAFSSON, F. Analysis of twiddle factor memory complexity of radix-2^i pipelined FFTs, In: **PROC. ASILOMAR CONF. SIGNALS SYST. COMP**., Pacific Grove, CA, 2009. p. 1-4.

QURESHI, F.; GUSTAFSSON, F. O. 4k-point FFT algorithms based on optimized twiddle factor multiplication for FPGAs, In: **MICROELECTRONICS AND ELECTRONICS (PRIMEASIA), 2010 ASIA PACIFIC CONFERENCE ON POSTGRADUATE RESEARCH**, [S.l.:s.n], 2010. p. 225-228.

RABAEY, J. M., Digital Integrated Circuits: A Design Perspective. New Jersey: Prentice-Hall, 1996.

RADER, C. M. Discrete Fourier transforms when the number of data samples is prime. Proc. IEEE 56, [S.l.], p.1107–1108, 1968.

REIS, R. A. da L. **Concepção de Circuitos Integrados**. Porto Alegre, Saga Luzzatto, 2000.

ROY, S. Low-power-driven synthesis algorithms for sequential and combinational circuits, 1998, Tese de doutorado: University of Illinois at Urbana-Champaign, 1998.

SALBEGO, M.; GHISSONI, S.; RAHMEIER, J. G. N. An 8 point DCT-II multiplierless architecture based on CMM. In: 27th South Symposium on Microelectronics, 2012, São Miguel das missões. **Proceedings** do XXVII SIM, 2012. [S.l.:s.n], 2012. p. 1-4.

SHEN, A.; et al. On Average Power Dissipation and Random Pattern Testability of CMOS Combinational Logic Networks, IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN – ICCAD-92, **Proceedings …** Santa Clara: Academic, 1992, p. 402 – 407.

SOHN, A., Computer Architecture – Introduction and Integer Addition. Computer Science Department – New Jersey Institute of Technology, 2004.

STEVENS, K.; SUTER, B., A Mathematical Approach to a Low Power FFT Architecture, **IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS**, [S.l.:s.n], v.2, 1998. p. 21-24.

SWARTZLANDER, E.E.; YOUNG, W.K.W.; JOSEPH, S.J. A radix 4 delay commutator for fast Fourier transform processor implementation, **Solid-State Circuits, IEEE Journal of**, [S.l.:s.n], p. 702 - 709, Out 1984.

TUMMELTSHAMMER, P.; HOE, J.C.; PUSCHEL, M. Time-Multiplexed Multiple-Constant Multiplication, **IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems**, [S.l.:s.n], v.26, n.9, p. 1551-1563. Set 2007.

TURNER, R., COURTNEY, T.; WOODS, R. Implementation of fixed DSP functions using the reduced coefficient multiplier, In: **Proc. IEEE INT. CONF. ACOUST., SPEECH, SIGNAL PROCESS.**, [S.l.:s.n], 2001, v.2, p. 881‑884.

VACHER, A.; BENKHEBBAB, M., A VLSI Implementation of Parallel Fast Fourier Transform, **EUROPEAN DESIGN AND TEST CONFERENCE**, [S.l.:s.n], 1994. p. 250‑255.

VAN LOAN, C. Computational Framework for FAST Fourier Transform, SIAM Philadelphia, PA, 1992.

VORONENKO, Y.; M. P¨USCHEL, Multiplierless Multiple Constant Multiplication, **ACM Trans. Algorithms**, v.3, n.2, Mai 2007.

WEST, N.; ESHRAGHIAN, K., **Principles of CMOS VLSI design**. 2nd.ed. Boston: Addison-Wesley, 1994.

WU, G-D.; LIU, Y-M. Radix-2^2 Based Low Power Reconfigurable FFT Processor. **IEEE INTERNATIONAL SYMPOSIUM ON INDUSTRIAL ELECTRONICS (ISlE 2009**) Seoul Olympic Parktel, Seoul, Korea, 2009. July 5-8.

WU. M. J. ; FAN, Y. C. Coefficient ordering based pipelined FFT/IFFT with minimum switching activity for low power WiMAX communication system, In: **Proc. IEEE TENTH INT. SYMP. CONSUMER ELECTRONICS**, [S.l.:s.n], 2006, p. 1-4.

XIN, X.; ORUKLU, E.; SANIIE, J. Fast memory addressing scheme for radix-4 FFT implementation, **ELECTRO/INFORMATION TECHNOLOGY, 2009. IEEE INTERNATIONAL CONFERENCE ON ,WINDSOR.** [S.l.:s.n], 2009. p. 437–440.

YAVNE, R. An Economical Method For Calculating The Discrete Fourier Transform. **Proc.** AFIPS Fall Joint Computer Conf. 33, [S.l.], p.115–125, 1968.

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.

APÊNDICE <PUBLICAÇÕES>

Para demonstrar a eficácia do método desenvolvido nesta tese de doutorado, na tabela a seguir é apresentado os artigos publicados em conferências e revistas da área de microeletrônica.

Título	Veículo de Publicação
Efficient Multiplication for FFT Based on Twiddle Factor Decomposition	ICECS 2012
Reducing Area and Power in FFT Architectures Using Twiddle Factor Decomposition Approach	SIM 2012
Combination of Constant Matrix Multiplication and Gate-Level Approaches for Area and Power Efficient Hybrid Radix-2 DIT FFT Realization	ICECS 2011
Optimization area and power of radix-2 decimation in time (DIT) FFT implementation based on multiple constant multiplication in the stages	SIM 2011
Radix-2 decimation in time (DIT) FFT implementation based on a matrix-multiple constant multiplication approach	ICECS 2010
Decimation in time (DIT) FFT implementation based on multiple constant multiplication approach	SIM 2010
Otimização de dissipação de potência e atraso para circuitos CMOS utilizando SCCGS com topologia Anti-radiação	IBERCHIP 2009
Analysis of power consumption using a new methodology for the capacitance Modeling of complex logic gates with dogbone transistor	SFORUM 2009
Analysis of Power Consumption Using a New Methodology for the Capacitance Modeling of Complex Logic Gates	PATMOS 2009 LNCS-SPRINGER 2010