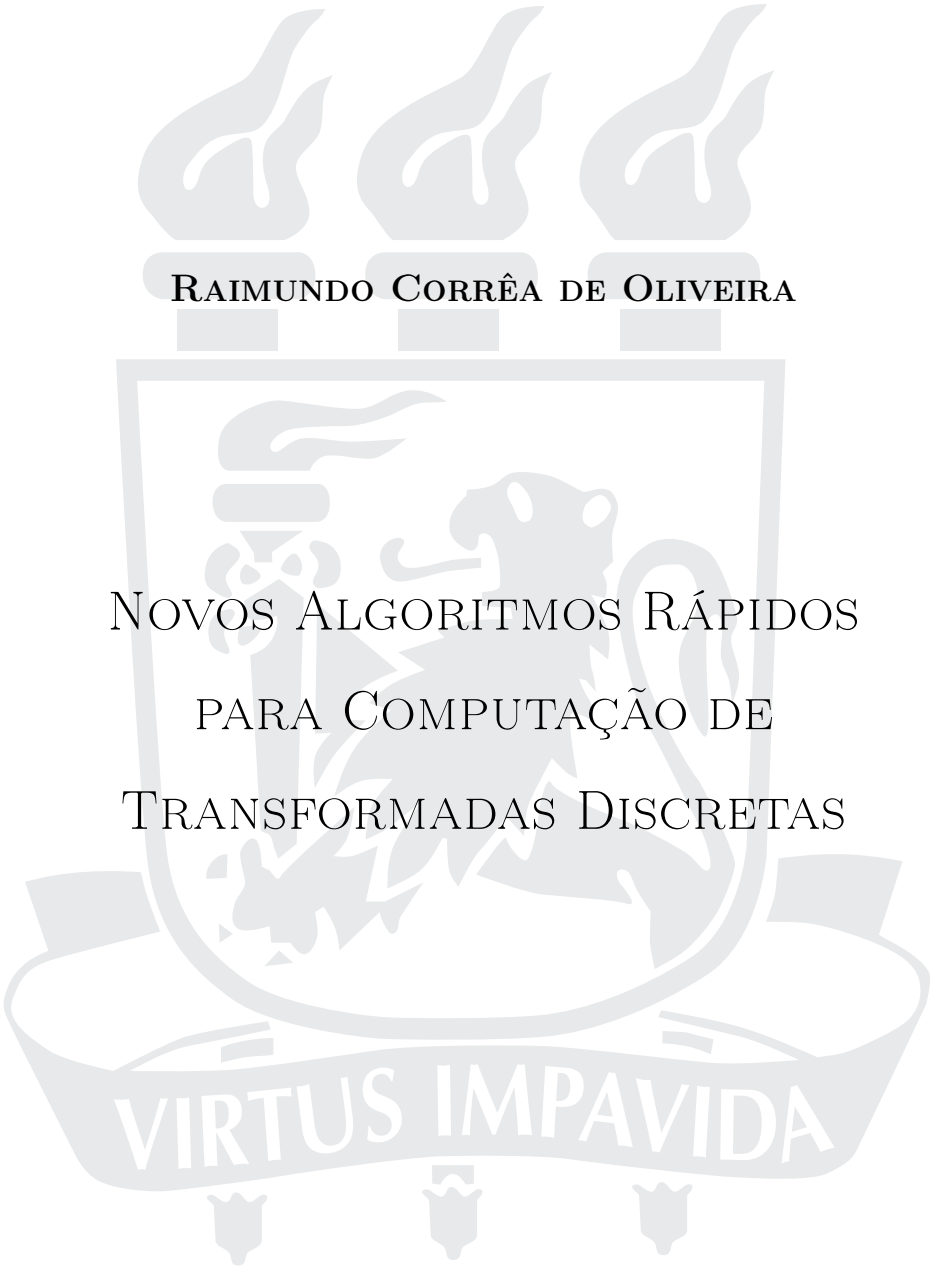
Universidade Federal de Pernambuco
Centro de Tecnologia e Geociências
Programa de Pós-graduação em Engenharia Elétrica



Raimundo Corrêa de Oliveira

Novos Algoritmos Rápidos para Computação de Transformadas Discretas

Recife, Abril de 2013.

Raimundo Corrêa de Oliveira

Novos Algoritmos Rápidos para Computação de Transformadas Discretas

Tese submetida ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Pernambuco como parte dos requisitos para obtenção do grau de **Doutor em Engenharia Elétrica**

Orientador: Prof. Ricardo Menezes Campello de Souza, Ph.D.

Recife, Abril de 2013.

Agradecimentos

Aqui expresso meus sinceros agradecimentos às pessoas que contribuíram direta e indiretamente para o desenvolvimento desta tese. Em especial agradeço:

À minha família, em especial, a minha Vó (em memória), a minha Mãe, a minha Tia, a minha esposa e filha, pelos apoios e incentivos constantes.

Ao meu orientador, professor Ricardo Campello, por me aceitar e confiar em mim como orientado no doutorado; pelas contribuições, motivações e correções; por sua disponibilidade e vibração e por ser um excelente professor. Além disso, uma ótima pessoa.

Ao professor Hélio Magalhães, pelas valorosas contribuições e incentivos.

Ao professor Rafael Lins, pelo apoio no decorrer do Dinter, sendo o coordenador do projeto DINTER que muito contribuirá para a região Norte.

Ao professor Edval dos Santos, por permitir a utilização do Laboratório de Nanodispositivos (LDN) para implementação em FPGA do algoritmo proposto.

Aos amigos do Dinter, que me ajudaram no desenvolvimento desta tese. Em especial ao Jucimar, Ricardo, Ernande e Rodrigo, pelas suas valorosas sugestões, contribuições e incentivos em nossas interessantes discussões.

Raimundo Corrêa de Oliveira

Universidade Federal de Pernambuco

17 de Abril de 2013

Resumo da Tese apresentada à UFPE como parte dos requisitos necessários para a obtenção do grau de Doutor em Engenharia Elétrica

Novos Algoritmos Rápidos para Computação de Transformadas Discretas

Raimundo Corrêa de Oliveira

Abril/2013

Orientador: Prof. Ricardo Menezes Campello de Souza, Ph.D.
Área de Concentração: Comunicações
Palavras-chaves: Transformada Discreta de Fourier, Transformada Discreta de Hartley, Transformadas Rápidas
Número de páginas: 108

Esta tese apresenta novos algoritmos rápidos para computação das transformadas discretas de Fourier (DFT) e de Hartley (DHT), denominados FFT e FHT, respectivamente. Os algoritmos FFT são baseados em uma expansão em série matricial de Laurent da matriz de transformação da DFT de comprimento $N \equiv 4 (\mathrm{mod}\ 8)$. A complexidade multiplicativa destes apresenta um ganho em relação aos algoritmos Cooley-Tukey base-2 e base-4. Os algoritmos FHT são baseados na expansão da matriz de transformação da DHT de comprimento $N \equiv 0 (\mathrm{mod}\ 4)$. Estes algoritmos rápidos apresentaram um melhor desempenho que algoritmos conhecidos para computação da DHT. Além disso, são apresentados algoritmos ótimos para esta transformada, para os comprimentos $N = 8, 12, 16$ e 24. Uma implementação em FPGA de um dispositivo que calcula as duas transformadas é apresentado; o dispositivo utilizado para implementar o projeto foi um Xilinx Spartan 3E.

Abstract of Thesis presented to UFPE as a partial fulfillment of the requirements for the degree of Doctor in Electrical Engineering

New Fast Algorithms for Computing Discrete Transforms

Raimundo Corrêa de Oliveira

April/2013

Supervisor: Prof. Ricardo Menezes Campello de Souza, Ph.D.
Area of Concentration: Communications
Keywords: Discrete Fourier Transforms, Discrete Hartley Transforms, Fast Transforms
Number of pages: 108

This thesis presents new fast algorithms for computing the discrete Fourier transform (DFT) and the discrete Hartley transform (DHT), called FFT and FHT, respectively. The FFT algorithms are based on Laurent series matrix expansion of the DFT transformation matrix of blocklength $N \equiv 4 (\text{mod } 8)$ and present a better performance than the Cooley-Tukey radix-2 and radix-4 algorithms. The FHT algorithms are based on an expansion of the DHT transform matrix of blocklength $N \equiv 0 (\text{mod } 4)$ and present a better performance than the algorithms known in the literature for computing the DHT. Specifically, FHTs with minimum multiplicative complexity for blocklengths $N = 8, 12, 16$ and 24 are presented. An FPGA implementation of a device that computes both the FFT and the FHT, based on a Xilinx Spartan 3E platform, is proposed.

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Sumário

capítulo 1

Introdução

As ideias básicas sobre a análise de Fourier remontam a 1822, quando Jean Baptiste Joseph Fourier investigou a propagação de calor em corpos sólidos [1, 2]. Nesse contexto, uma das principais ferramentas desenvolvidas foi a Transformada de Fourier, a qual tem um importante papel em várias áreas do conhecimento, especialmente em processamento de sinais [3, 4]. Algumas áreas do conhecimento que tem se beneficiado da análise de Fourier são:

- Astronomia [5];
- Imagens Médicas [6–8];
- Processamento de Voz [9–13];
- Áudio digital [14–17];
- Processamento Digital de Imagens [18–21];
- Criptografia [22];
- Codificação de Canais [23];
- Marcas d'Água [24, 25];
- Comunicações [26].

A essência da transformada de Fourier de um sinal é a sua decomposição em um somatório de senóides de diferentes frequências [27, 28]. Em casos práticos, sua avaliação não é realizada analiticamente, mas numericamente e, na maioria dos casos, não existe uma expressão analítica para o sinal a ser analisado. A Transformada Discreta de Fourier (DFT, do inglês *Discrete Fourier Transform*) é uma ferramenta que computa o espectro de frequência

de uma sequência finita de comprimento N, com uma complexidade computacional de N^2 multiplicações e $N(N-1)$ adições. O sucesso da aplicação de técnicas de transformação é devido, principalmente, à existência dos chamados "algoritmos rápidos"[29]. Com isso, técnicas para computação de transformadas discretas com uma baixa complexidade multiplicativa vem sendo objeto de interesse há um longo tempo. A seguir, apresenta-se uma breve cronologia do desenvolvimento dos principais algoritmos rápidos conhecidos na literatura:

- Em 1958, Goertzel apresentou um algoritmo para computar componentes isoladas da DFT [30];
- Os primeiros algoritmos rápidos para computar a DFT foram propostos por I. J. Good (1958, 1960) e por L. H. Thomas (1963) [31, 32]. Esses algoritmos se baseiam no teorema chinês do resto e, posteriormente, foram unificados sob o título de algoritmo de Good-Thomas ou algoritmo do fator primo (PFA, do inglês *Prime Factor Algorithm*) [33].
- Em 1965, J.W. Cooley e J.W. Tukey introduziram uma ideia revolucionária que posteriormente tornou-se conhecida como a Transformada Rápida de Fourier (FFT, do inglês *Fast Fourier Transform*) [34]. Entretanto, pode-se atribuir a Gauss algumas das idéias propostas nesse trabalho [35, 36]. A FFT é um marco na teoria de algoritmos [37] e, mais especificamente, no campo de Processamento de Sinais [38, 39];
- Em 1968, Rader [40] e, em 1970, Bluestein [41], forneceram métodos para computar a DFT por meio de uma convolução; o primeiro ficou conhecido na literatura como algoritmo Rader primo;
- Em 1969, Bergland apresentou um algoritmo com base 8 [42, 43] e Singleton apresentou uma FFT com base mista [44];
- Em 1971, Pollard desenvolveu uma transformada análoga à DFT, porém definida em um corpo finito [45];
- Em 1976 [46], Rader e Brenner propuseram uma forma alternativa da FFT. Neste mesmo ano, Winograd [47] propôs um algoritmo que combina o algoritmo de Rader primo com um outro algoritmo, criado pelo primeiro, para computar uma convolução;
- Em 1978, Winograd generalizou o método de Rader para comprimentos que sejam uma potência de um primo [48].
- Em 1984, Duhamel e Hollman apresentaram um algoritmo em que se usavam diferentes bases, o *split-radix* [49]; outras variações deste algoritmo são encontradas em [50–52];

- Em 1988 [53], Heideman estudou a complexidade multiplicativa da DFT, chegando a uma fórmula para o número de multiplicações necessárias para computá-la. Nesse mesmo ano, Tufts e Sadasiv desenvolveram um algoritmo rápido com base em funções aritméticas para o cálculo da DFT, a chamada Transformada Aritmética de Fourier (AFT, do inglês *Arithmetic Fourier Transform*), sem utilizar multiplicações em ponto flutuante [54, 55]. Detalhes sobre o desenvolvimento da AFT podem ser encontrados em [56]. Este algoritmo fornece uma aproximação do valor da DFT;
- Em 1995, Varkonyi-Koczy apresentou um algoritmo que usava a recursividade para computar a DFT [57];
- Em 2009 [58], Marti-Puig apresentou duas famílias de algoritmos FFT base-2 que tem a propriedade que ambas, entrada e saída, sejam endereçadas em ordem natural;
- Em 2011, Silva Jr. e Campello de Souza propuseram novos algoritmos, explorando os conceitos de base ciclotômica, para a computação de um número arbitrário de componentes da DFT. Esses algoritmos são ótimos, no sentido de apresentarem complexidade multiplicativa mínima [59–61];
- Outros algoritmos rápidos são apresentados e podem ser consultados em [62–64].

Um tutorial conciso e que revisa os algoritmos FFT está disponível em [65]. Um dos objetivos na pesquisa destes algoritmos é reduzir sua complexidade aritmética, isto é, o número de multiplicações e adições reais para calcular a DFT.

Em 1942 [66], R. V. L. Hartley(1890-1970) introduziu uma transformada real que é uma formulação alternativa para a transformada de Fourier. Esta é atrativa pois usa aritmética real. Pode-se citar algumas aplicações da transformada de Hartley:

- Processamento de imagem. [67–70];
- Processamento imagens sísmicas. [71];
- Cifragem de imagem ópticas. [72];
- Compressão de imagem. [73];
- Óptica e microondas. [74, 75]
- Processamento de Voz. [76, 77]
- Filtragem adaptativa. [78]
- Comunicações. [79, 80]

A seguir apresenta-se uma breve cronologia do desenvolvimento da transformada de Hartley:

- Em 1942, Hartley publicou a transformada contínua de Hartley [27, 66, 81];
- Em 1983, Bracewell desenvolveu a transformada discreta de Hartley - DHT (do inglês, *Discrete Hartley Transform*) [82];
- Em 1984, Bracewell implementou um algoritmo rápido para computação da DHT [83];
- Em 1985, Sorensen, Jones, Burrus e Heideman propuseram um conjunto de FHTs [84];
- Em 1990, Yang propos uma FHT de fator primo [85];
- Em 1993, Lun propos uma FHT de base-3/9 [86];
- Em 1994, Guoan propos um algoritmo *"split radix"* para computação da DHT [87];
- Em 1998, Campello de Souza, de Oliveira e Kauffman introduziram a transformada de Hartley sobre um corpo finito [88];
- Em 1999, Guoan Bi e Chen, propuseram uma FHT para comprimentos $N = q * 2^m$ [89];
- Em 2000, H. de Oliveira, Cintra e Campello de Souza desenvolveram uma FHT com uma decomposição multinível para a DHT [90];
- Em 2001, Cintra, H. de Oliveira e Cintra desenvolveram a transformada de Hartley truncada [91];
- Em 2005, Bouguezel e Ahmad, propuseram uma FHT de base-2/4 [92];
- Em 2009, Shah desenvolveu uma DHT de base-2 [93];
- Em 2011, Hamood e Boussakta desenvolveram uma FHT baseada em base-2^{2^m} [94].

Com o advento da VLSI (do inglês, *Very Large Scale Integration*) e o desenvolvimento dos Processadores Digitais de Sinais (DSP, do inglês *Digital Signal Processor*), que implementam técnicas de processamento de sinais, a DFT e a DHT tornaram-se ferramentas atrativas para avaliação do espectro de um sinal [95, 96]. A redução de custo dos DSPs e a capacidade crescente alcançada por processadores de dados (ou seja, dezenas de GFlps - Giga operações em ponto flutuante por segundo - e TFlops, Tera operações em ponto flutuante por segundo) [97], junto com novas e eficientes técnicas de processamento de sinais, estão tornando viáveis aplicações em tempo real para diversos tipos de sinais. Neste cenário, a DFT e DHT tornaram-se ferramentas largamente difundidas para análise espectral [98].

1.1 Contribuições

O objetivo desta tese é apresentar novos algoritmos rápidos para computação de transformadas discretas. As contribuições dadas são:

- Um novo algoritmo rápido para computação da DFT, baseado em série matricial de Laurent, para comprimentos $N \equiv 4 \mod 8$.
- Um novo algoritmo rápido para computação da DHT, baseado em expansão matricial, para comprimento $N \equiv 0 (\mod 4$.
- Algoritmos ótimos para computação da DHT para os comprimentos $8, 12, 16$ e 24.
- Projeto e implementação, no dispositivo SPARTAN 3E xc3s500e-5-fg320, de um algoritmo que é capaz de fazer a computação rápida dos coeficientes da DFT/DHT de uma sequência real de comprimento $N = 16$.
- Procedimento para redução de operações aditivas em uma matriz.

Os novos algoritmos rápidos apresentam uma complexidade melhor que a dos algoritmos mais conhecidos, sendo que suas estruturas são interessantes para implementações em VLSI e FPGA.

1.2 Organização da Tese

A tese está organizada em seis capítulos. Após esta introdução, o capítulo 2 apresenta a transformada discreta de Fourier (DFT), em que visa-se fazer a ligação da representação de sinais de tempo contínuo com a de sinais de tempo discreto; além disso, é apresentado o conceito de transformada discreta de Hartley. No capítulo 3 são apresentados os principais algoritmos rápidos para cálculo da DFT, que são os algoritmos de Cooley-Tukey, Good-Thomas e Rader-Primo. O capítulo 4 introduz a técnica proposta para computação da DFT baseada em série matricial de Laurent, apresentando exemplos da aplicação dessa nova técnica. O capítulo 5 apresenta um novo algoritmo para computação da DHT através de expansão matricial. O capítulo 6 apresenta uma implementação em FPGA das duas transformadas (Fourier e Hartley). O capítulo 7 apresenta as conclusões e sugestões para trabalhos futuros.

capítulo 2

A Transformada Discreta de Fourier - DFT

A análise spectral de sinais em tempo contínuo tem como uma de suas principais ferramentas a transformada de Fourier [99]. No entanto, muitas aplicações envolvendo esta transformada dependem de um computador digital para efetuar o processamento dos dados, sendo que seu cálculo fica inviável para sinais em tempo contínuo. Como mostrado mais adiante neste capítulo, a transformada discreta de Fourier pode ser derivada a partir da Transformada de Fourier em Tempo Discreto (TFTD). Este capítulo tem como objetivo introduzir a DFT que será utilizada para análise espectral de sinais.

2.1 Introdução

Há quatro representações de Fourier distintas, cada uma aplicável a uma classe de sinais. A Tabela 2.1 mostra a relação entre o domínio do tempo e as representações de Fourier. Nesta pode-se ver as quatro classes de sinais definidas pelos aspectos periodicidade (periódico/não periódico) e tipo (contínuo/discreto). Os sinais periódicos têm representações pelas séries de Fourier: série de Fourier (FS), discreta e não periódica; e série discreta de Fourier (DFS), discreta e periódica. A primeira se aplica a sinais periódicos de tempo contínuo, e a segunda se aplica a sinais periódicos de tempo discreto. Sinais não-periódicos têm representações pelas transformadas de Fourier: transformada de Fourier contínua (FT), contínuo e não periódico; e transformada de Fourier de tempo discreto (DTFT), contínuo e periódico. A primeira se aplica a sinais de tempo contínuo e não-periódico, e a segunda se aplica a sinais de tempo

discreto e não-periódico [100].

Tabela 2.1: Relação entre características de tempo de um sinal e sua representação de Fourier

Domínio no Tempo	**Periódico**	**Não periódico**
Contínuo	Série de Fourier (FS) (Discreta e Não periódica)	Transformada de Fourier contínua (FT) (Contínua e Não Periódica)
Discreto	Série Discreta de Fourier (DFS)/DFT (Discreta e Periódica)	Transformada de Fourier de Tempo Discreto (DTFT) (Contínua e Periódica)

Um procedimento para derivar a Transformada Discreta de Fourier é visualizado na Figura 2.1. Na figura, cada bloco representa uma operação, em que as descrições acima do mesmo indicam conceitos (regras) utilizados no respectivo bloco e as descrições abaixo indicam ferramentas de apoio (podendo ser sinais, tipo trem de impulso, impulso retangular, etc.) para a implementação da operação. Primeiro, deve-se fazer a amostragem do sinal no domínio do tempo, respeitando o critério de Nyquist [100]. Em seguida, deve-se fazer um truncamento ou janelamento desta sequência, ou seja, multiplicar a sequência, gerada pelo passo anterior, por um pulso retangular, obtendo, com isso, uma sequência finita de comprimento N. Calculamos a DTFT desta sequência, seguindo a definição. De posse da mesma, fazemos a amostragem no domínio da frequência, utilizando, para isso, um trem de impulsos cujo período é de $2\pi/N$, gerando uma sequência que é conhecida como transformada discreta de Fourier [101].

Vale ressaltar que o processo de truncamento ou janelamento por um pulso retangular introduz componentes artificiais de alta frequência, que podem ser minimazadas ao utilizarmos uma janela não retangular, sob o custo de redução da resolução em frequência. Esta perda é recuperada somente com o aumento do comprimento N do sinal em tempo discreto [102]. A seguir, uma descrição analítica do procedimento ilustrado na Figura 2.1 é apresentada.

2.2 Representação de sequências periódicas: A série discreta de Fourier (DFS)

Considere uma sequência periódica $\widetilde{v}[n]$, com período N, ou seja

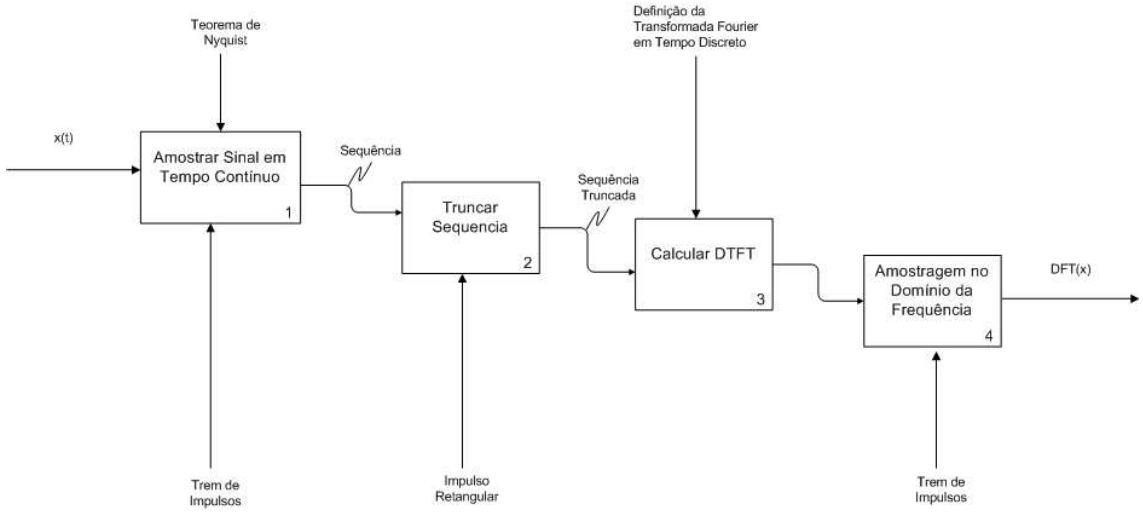


Figura 2.1: Procedimento conceitual para cálculo da DFT.

$$\widetilde{v}[n] = \widetilde{v}[n + rN], \tag{2.1}$$

para qualquer valor inteiro de r e N. Como ocorre com os sinais periódicos em tempo contínuo, tal sequência pode ser representada por uma série de Fourier, correspondendo a uma combinação linear de exponenciais complexas harmonicamente relacionadas, cujas frequências são múltiplos inteiros da frequência fundamental $(2\pi/N)$ (ou harmônicas) associada com o período da sequência $\widetilde{v}[n]$.

A série de Fourier de sinais periódicos de tempo contínuo requer infinitas exponenciais complexas harmonicamente relacionadas. Entretanto, a série de Fourier para qualquer sinal em tempo discreto com período N requer somente N exponenciais complexas harmonicamente relacionadas, como mostrado na Equação 2.2 [100],

$$\widetilde{v}[n] = \frac{1}{N} \sum_{k=0}^{N-1} \widetilde{V}[k] e^{j(2\pi/N)kn}, \tag{2.2}$$

pois $e^{j(2\pi/N)(k+lN)n} = e^{j(2\pi/N)kn} e^{j2\pi ln} = e^{j(2\pi/N)kn}$.

Os coeficientes da série de Fourier são obtidos da série periódica conforme Equação 2.3 [100],

$$\widetilde{V}[k] = \sum_{n=0}^{N-1} \widetilde{v}[n] e^{-j(2\pi/N)kn}, \tag{2.3}$$

com isso, a sequência resultante é periódica de período N.

Pode-se representar o par DFS e as equações de análise e síntese da série discreta de Fourier pelas Equações 2.4, 2.5 e 2.6, respectivamente,

- Par DFS

$$\widetilde{v}[n] \overset{DFS}{\longleftrightarrow} \widetilde{V}[k] \tag{2.4}$$

- Equação de Análise

$$\widetilde{V}[k] = \sum_{n=0}^{N-1} \widetilde{v}[n]\omega_N^{kn}. \tag{2.5}$$

- Equação de Síntese

$$\widetilde{v}[n] = \frac{1}{N} \sum_{k=0}^{N-1} \widetilde{V}[k]\omega_N^{-kn}. \tag{2.6}$$

em que $\omega_N \triangleq e^{-j(2\pi/N)}$. A dedução destas Equações pode ser encontrada em [100].

A Figura 2.2 mostra o gráfico de uma sequência periódica e sua respectiva DFS.

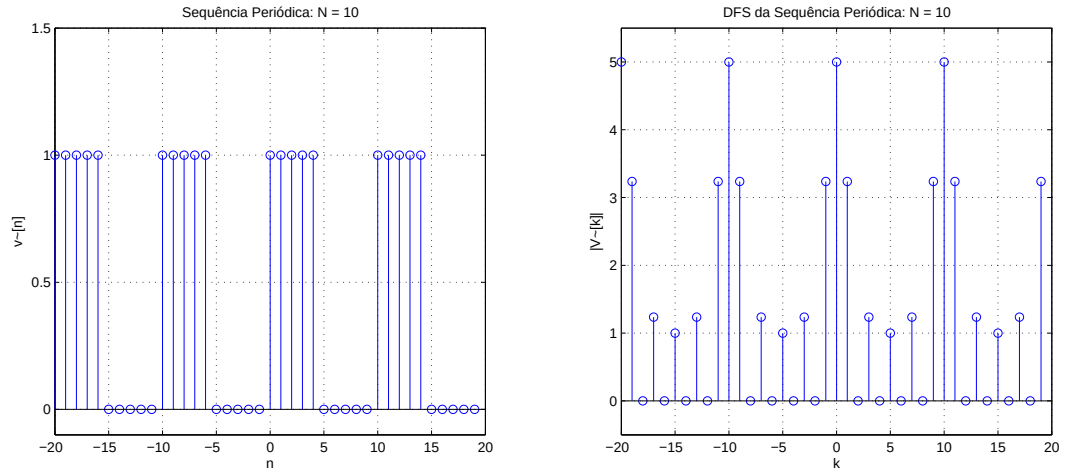


Figura 2.2: Sequência periódica e sua DFS.

2.3 A transformada de Fourier de sinais periódicos

Uma sequência que é representada como uma soma de exponenciais complexas tem uma representação por meio da transformada de Fourier como um trem de impulsos [100]. Com isso, é útil incorporar a representação da série discreta de Fourier de sinais periódicos dentro da definição da transformada de Fourier.

Seja $\widetilde{v}[n]$ uma sequência periódica com período N e sejam $\widetilde{V}[k]$ os correspondentes coeficientes da série discreta de Fourier. Define-se a transformada de Fourier de sinais periódicos conforme a Equação 2.7.

$$\widetilde{V}(e^{j\omega}) = \sum_{k=-\infty}^{\infty} \frac{2\pi}{N}\widetilde{V}[k]\delta\left(\omega - \frac{2\pi k}{N}\right). \tag{2.7}$$

Note que $\widetilde{V}(e^{j\omega})$ tem necessariamente periodicidade igual a 2π, haja vista que $\widetilde{V}[k]$ é periódica com período N e os impulsos são espaçados em múltiplos inteiros de $2\pi/N$, onde N é um número inteiro. Na Figura 2.2, ilustra-se um sinal periódico com período $N = 10$.

Agora, vamos considerar um sinal de comprimento finito $v[n]$, ou seja, $v[n] = 0$, exceto no intervalo $0 \leq n \leq N-1$. Através da convolução desta sequência com um trem de impulsos, podemos reescrever a sequência periódica $\widetilde{v}[n]$ conforme a Equação 2.8,

$$\widetilde{v}[n] = v[n] * \sum_{r=-\infty}^{\infty} \delta(n - rN) = \sum_{r=-\infty}^{\infty} v[n - rN]. \tag{2.8}$$

A Figura 2.3 mostra uma sequência periódica infinita, com período $N = 5$, e uma sequência finita de comprimento $N = 5$.

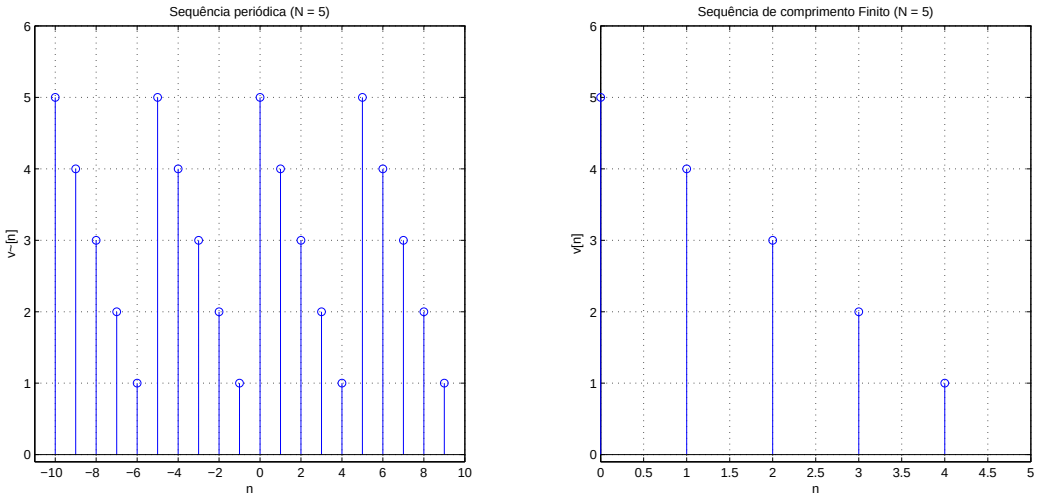


Figura 2.3: Sequência Periódica com período $N = 5$.

Diante disso, podemos definir a Transformada de Fourier de Tempo Discreto para um sequência de comprimento finito conforme Equação 2.9,

$$V(e^{j\omega}) = \sum_{n=0}^{N-1} v[n]e^{-j\omega n} = \sum_{n=0}^{N-1} \widetilde{v}[n]e^{-j\omega n}. \tag{2.9}$$

Comparando as Equações 2.9 e 2.5, podemos relacionar a transformada de Fourier com a série discreta de Fourier conforme a Equação 2.10,

$$\widetilde{V}[k] = V(e^{j\omega})\Big|_{\omega=\frac{2\pi k}{N}} . \tag{2.10}$$

Isto corresponde à amostragem da transformada de Fourier em N frequências igualmente espaçadas entre $\omega = 0$ e $\omega = 2\pi$, com um espaçamento de frequência de $2\pi/N$. Isto é chamado de resolução de frequência, porque nos diz o quão próximo estão as amostras de frequência. A Figura 2.4 ilustra a amostragem de frequência para $N = 8$.

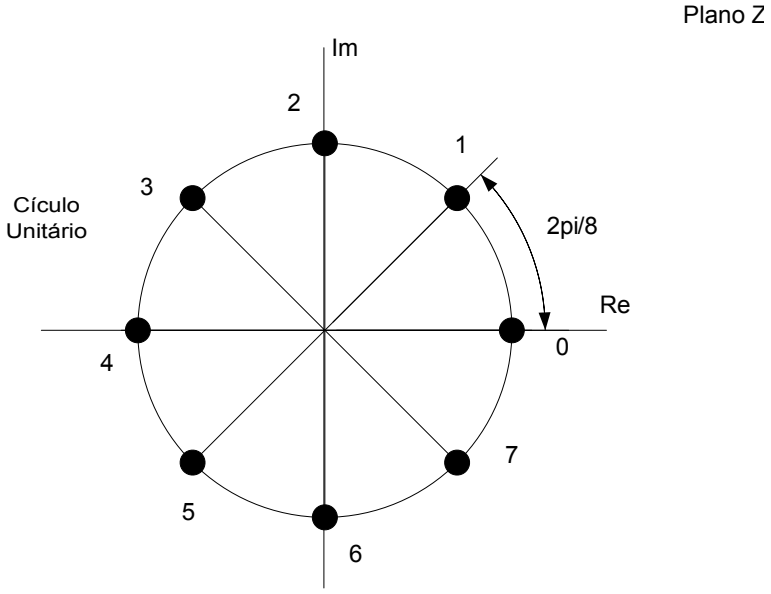


Figura 2.4: Pontos no cículo unitário ilustrando a amostragem de frequência de um sinal periódico, cujo comprimento é 8.

2.4 A transformada discreta de Fourier: DFT

Para cada sequência $v[n]$ de duração finita, podemos sempre associá-la a uma sequência periódica, de período N,

$$\widetilde{v}[n] = \sum_{r=-\infty}^{\infty} v[n - rN]. \tag{2.11}$$

A sequência de duração finita pode ser recuperada por meio de

$$v[n] = \begin{cases} \widetilde{v}[n], & 0 \leq n \leq N-1, \\ 0, & \text{caso contrário.} \end{cases} \tag{2.12}$$

Os coeficientes da DFS de $\widetilde{v}[n]$ são amostras da transformada de Fourier de $v[n]$. A sequência dos coeficientes da série de Fourier $\widetilde{V}[k]$ da sequência periódica $\widetilde{v}[n]$ é uma sequência periódica com período N. A Figura 2.5 mostra essa relação para uma sequência finita obtida de uma onda quadrada ($N = 10$) .

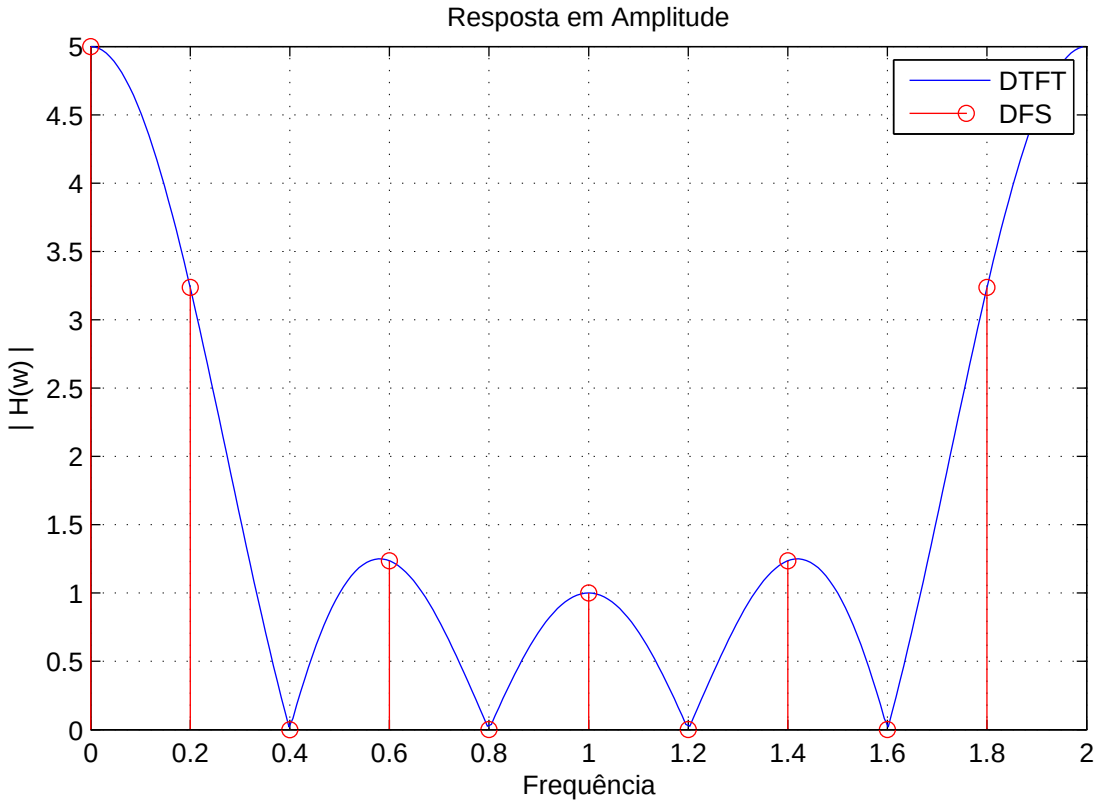


Figura 2.5: Gráfico mostrando a Relação entre a DTFT e a DFS.

Para manter a dualidade entre os domínios de tempo e frequência, escolheremos os coeficientes da série, que associamos com uma sequência de duração finita, pois esta é de duração finita correspondendo ao período de $\widetilde{V}[k]$ [100].

Esta sequência, $V[k]$, será chamada de transformada discreta de Fourier (DFT). Com isso, a DFT está relacionada com os coeficientes da DFS por

$$V[k] = \begin{cases} \widetilde{V}[k], & 0 \leq k \leq N-1, \\ 0, & \text{caso contrário.} \end{cases} \tag{2.13}$$

Como

$$\widetilde{V}[k] = \sum_{n=0}^{N-1} \widetilde{v}[n]\omega_N^{kn}$$

e

$$\widetilde{v}[n] = \frac{1}{N}\sum_{k=0}^{N-1} \widetilde{V}[k]\omega_N^{-kn},$$

portanto

$$V[k] = \begin{cases} \sum_{n=0}^{N-1} v[n]\omega_N^{kn}, & 0 \leq k \leq N-1, \\ 0, & \text{caso contrário.} \end{cases} \tag{2.14}$$

e

$$v[n] = \begin{cases} \frac{1}{N}\sum_{k=0}^{N-1} V[k]\omega_N^{-kn}, & 0 \leq k \leq N-1, \\ 0, & \text{caso contrário.} \end{cases} \tag{2.15}$$

As equações de Análise e Síntese da DFT são representadas pelas Equações 2.16 e 2.17, respectivamente.

- Equação de Análise

$$V[k] = \sum_{n=0}^{N-1} v[n]\omega_N^{kn}. \tag{2.16}$$

- Equação de Síntese

$$v[n] = \frac{1}{N}\sum_{k=0}^{N-1} V[k]\omega_N^{-kn}. \tag{2.17}$$

A relação entre as equações 2.16 e 2.17 será denotada por

$$v[n] \overset{DFT}{\longleftrightarrow} V[k] \tag{2.18}$$

Sobre estas expressões, pode-se tirar duas importantes conclusões:

1. As amostras da transformada de Fourier de tempo discreto fornecem uma efetiva representação discreta no domínio da frequência para um sinal de comprimento finito de tempo discreto.

2. Para que não haja perda de informação nessa representação, é necessário que o número N de amostras da transformada de Fourier de tempo discreto seja maior ou igual ao comprimento do sinal original.

A representação matricial da DFT é mostrada na equação 2.19,

$$\begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ \vdots \\ V_{N-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \dots & \omega^{(N-1)(N-1)} \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ \vdots \\ v_{N-1} \end{pmatrix}. \tag{2.19}$$

Em 1987, Heideman investigou a complexidade aritmética da DFT e derivou os limites inferiores da complexidade multiplicativa para computá-la [53]. Seja $\mu_{DFT}(N)$ a mínima complexidade multiplicativa da computação da DFT de um bloco de comprimento N.

Teorema 2.1 *(Heideman). Para $N = \prod_{i=1}^{m} p_i^{e_i}$ onde p_i, $i = 1, \dots, m$ são primos distintos e e_i, $i = 1, \dots, m$ são inteiros positivos, segue que*

$$\mu_{DFT}(N) = 2N - \sum_{i_1=0}^{e_1} \sum_{i_2=0}^{e_2} \dots \sum \phi \left(gcd \left(\prod_{j=1}^{m} p_j^{i_j} \right) \right) \tag{2.20}$$

em que $\phi(.)$ é a função totiente de Euler $\phi(x) = |\{n \in N | n < x \wedge gcd(n, x) = 1\}|$, $gcd(.,.)$ indica o máximo divisor comum.

Demonstração: Para uma demonstração deste teorema, o leitor pode consultar [53]. Esta prova é baseada na avaliação da complexidade multiplicativa do produto de um conjunto de polinômios. ■

2.5 A transformada discreta de Hartley: DHT

Considerada, por muitos anos, essencialmente como uma técnica para computação da transformada de Fourier, a transformada de Hartley (contínua ou discreta) tornou-se uma ferramentas muito importante, com muitas aplicações em vários campos da Engenharia [103]. Em particular, o par da transformada discreta de Hartley (DHT, do inglês *Discrete Hartley Transform* é definido, para uma sequência de comprimento N, $h[n]$, $0 \leq n \leq N-1$, pelas Equações 2.21 e 2.22 [82],

$$H[k] := \sum_{n=0}^{N-1} h[n] cas(\frac{2\pi}{N} kn) \quad 0 \leq k \leq N-1, \tag{2.21}$$

$$h[n] = \frac{1}{N} \sum_{k=0}^{N-1} H[k] cas(\frac{2\pi}{N} kn) \quad 0 \leq n \leq N-1, \tag{2.22}$$

em que $cas(.)$ denota a função *cosine and sine* definida como $cas(i) := \cos(i) + \sin(i)$. A DHT, como sua contraparte contínua, é real, e a simetria do par da transformada é uma característica valiosa para sua implementação.

2.6 Conclusões

Neste capítulo apresentamos um procedimento para se chegar à DFT, fazendo a associação com a série discreta de Fourier e a transformada de Fourier em tempo discreto. Finalizamos com a representação em forma matricial da mesma. Esta formulação servirá como base para o desenvolvimento dos próximos capítulos.

capítulo 3

A Transformada Rápida de Fourier

Com o advento da VLSI e o desenvolvimento do processador digital de sinal (DSP) para implementar técnicas de processamento de sinais, as transformadas discretas, tais como a DFT e a DHT, tornaram-se ferramentas atrativas para avaliação espectral. A redução de custo dos DSPs e a surpreendente capacidade alcançada pelos processadores têm tornado viáveis aplicações de tempo real para vários tipos de sinais. Neste cenário, as aplicações bem sucedidas de técnicas de transformação são, principalmente, devido à existência dos chamados algoritmos rápidos.

A DFT, como definida na Equação 2.16, requer N^2 multiplicações e $N(N-1)$ adições, o que limita a sua aplicação mesmo para sequências de comprimento moderado. Porém, se N é número composto, há diversas maneiras de modificar esta transformada unidimensional em uma bidimensional [29], o que proporciona uma redução na complexidade aritmética de sua computação. Em 1965, Cooley e Tukey propuseram um algoritmo rápido para computar a DFT [34]. Este algoritmo tem complexidade multiplicativa $(N/2)\log_2 N$, muito menor em comparação a N^2 da computação por definição [101]. Quando N é um número primo, existe uma FFT que computa a DFT por meio de uma convolução cíclica, usando o algoritmo rápido de Winograd para filtragem digital. Desta forma, a computação da DFT tornou-se muito mais rápida, tornando-a viável para diversas aplicações que vão desde análise de sinais até realizar uma filtragem linear rápida. Existem diversos algoritmos rápidos para cálculo da DFT [47, 65, 95, 104, 105]. Neste capítulo, abordaremos os algoritmos de Cooley-Tukey,

Good-Thomas e Rader-Primo.

3.1 O Algoritmo de Cooley-Tukey

Seja $N = N_1 N_2$. Vamos substituir cada um dos índices, na expressão para a transformada discreta de Fourier, por índices da seguinte forma:

$$\begin{aligned} n &= n_1 + N_1 n_2, \quad n_1 = 0, \ldots, N_1 - 1 \ e \ n_2 = 0, \ldots, N_2 - 1; \\ k &= N_2 k_1 + k_2, \quad k_1 = 0, \ldots, N_1 - 1 \ e \ k_2 = 0, \ldots, N_2 - 1. \end{aligned}$$

Então,

$$X_{N_2 k_1 + k_2} = \sum_{n_2=0}^{N_2-1} \sum_{n_1=0}^{N_1-1} \omega_N^{(n_1 + N_1 n_2)(N_2 k_1 + k_2)} x_{n_1 + N_1 n_2}. \tag{3.1}$$

Expandindo o produto, temos

$$\begin{aligned} X_{N_2 k_1 + k_2} &= \sum_{n_2=0}^{N_2-1} \sum_{n_1=0}^{N_1-1} \omega_N^{(n_1 N_2 k_1 + n_1 k_2 + N_1 n_2 N_2 k_1 + N_1 n_2 k_2)} x_{n_1 + N_1 n_2} \\ &= \sum_{n_2=0}^{N_2-1} \sum_{n_1=0}^{N_1-1} \omega_N^{n_1 N_2 k_1} \omega_N^{n_1 k_2} \omega_N^{N_1 n_2 N_2 k_1} \omega_N^{N_1 n_2 k_2)} x_{n_1 + N_1 n_2}. \end{aligned} \tag{3.2}$$

em que

$$\omega_N = e^{-j\frac{2\pi}{N}},$$

Definindo

$$\omega_N^{N_1} = e^{-j\frac{2\pi N_1}{N}} = e^{-j\frac{2\pi N_1}{N_1 N_2}} = e^{-j\frac{2\pi}{N_2}} = \omega_{N_2},$$

e

$$\omega_N^{N_2} = e^{-j\frac{2\pi N_2}{N}} = e^{-j\frac{2\pi N_2}{N_1 N_2}} = e^{-j\frac{2\pi}{N_1}} = \omega_{N_1},$$

tem-se

$$X_{N_2 k_1 + k_2} = \sum_{n_1=0}^{N_1-1} \omega_{N_1}^{n_1 k_1} \left[\omega_N^{n_1 k_2} \sum_{n_2=0}^{N_2-1} \omega_{N_2}^{n_2 k_2} x_{n_1 + N_1 n_2} \right]. \tag{3.3}$$

Definindo variáveis de duas dimensões,

$$x_{n_1 n_2} \triangleq x_{n_1+N_1 n_2}, \quad n_1 = 0, \ldots, N_1 - 1 \ e \ n_2 = 0, \ldots, N_2 - 1,$$
$$X_{k_1 k_2} \triangleq X_{N_2 k_1 + k_2}, \quad k_1 = 0, \ldots, N_1 - 1 \ e \ k_2 = 0, \ldots, N_2 - 1,$$

tem-se

$$X_{k_1 k_2} = \sum_{n_1=0}^{N_1-1} \omega_{N_1}^{n_1 k_1} \left[\omega_N^{n_1 k_2} \sum_{n_2=0}^{N_2-1} \omega_{N_2}^{n_2 k_2} x_{n_1 n_2} \right]. \tag{3.4}$$

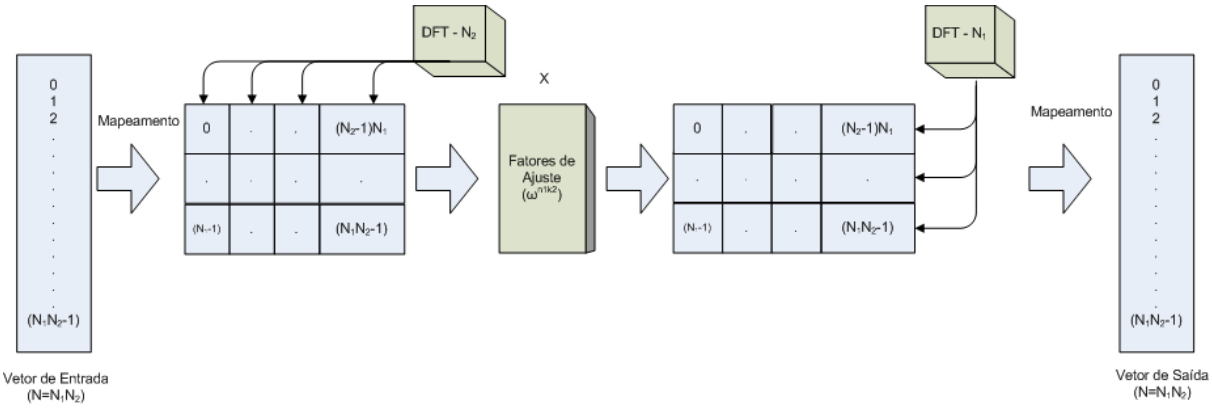


Figura 3.1: Esquema ilustrativo para a FFT Cooley-Tukey: observar a indexação de 1-D para 2-D. Apenas os índices das componentes dos vetores de entrada e de saída são indicados.

Para computação da DFT de comprimento N, pode-se seguir o seguinte procedimento:

1. Faz-se o mapeamento da sequência de entrada para uma matriz bidimensional;
2. Aplica-se em cada coluna da matriz uma DFT de comprimento N_2;
3. Multiplica-se a matriz, resultante do passo anterior, pelos fatores de ajuste $\omega_N^{n_1 k_2}$;
4. Aplica-se, em cada linha da matriz obtida no passo 3, uma DFT de comprimento N_1;
5. Faz-se o mapeamento para gerar a transformada da sequência.

A Figura 3.1 ilustra este procedimento. Embora essa forma seja mais difícil de entender que a original, o número de multiplicações e adições é menor. Na realidade, $N(N_1 + N_2 + 1)$ multiplicações complexas e $N(N_1 + N_2 - 2)$ adições complexas são necessárias, comparando com N^2 multiplicações complexas e $N(N-1)$ adições complexas, anteriormente.

3.1.1 O algoritmo de Cooley-Tukey base 2: dizimação no tempo

Seja N uma potência de 2. Pode-se separar a sequência de entrada, x_n, em componentes pares e ímpares, obtendo-se

$$X_k = \sum_{n\ par} x_n \omega_N^{kn} + \sum_{n\ impar} x_n \omega_N^{kn}. \tag{3.5}$$

Substituindo a variável n por $n = 2r$, quando n é par, e $n = 2r + 1$, quando n é ímpar, tem-se

$$\begin{aligned} X_k &= \sum_{r=0}^{(N/2)-1} x_{2r} \omega_N^{2rk} + \sum_{r=0}^{(N/2)-1} x_{2r+1} \omega_N^{(2r+1)k} \\ &= \sum_{r=0}^{(N/2)-1} x_{2r} (\omega_N^2)^{rk} + \omega_N^k \sum_{r=0}^{(N/2)-1} x_{2r+1} (\omega_N^2)^{rk}. \end{aligned} \tag{3.6}$$

Como $\omega_N^2 = e^{-2j(2\pi/N)} = e^{-j(2\pi/(N/2))} = \omega_{N/2}$, a Equação 3.6 pode ser escrita na forma

$$X_k = \sum_{r=0}^{(N/2)-1} x_{(2r)} \omega_{N/2}^{rk} + \omega_N^k \sum_{r=0}^{(N/2)-1} x_{(2r+1)} \omega_{N/2}^{rk}. \tag{3.7}$$

Observa-se que cada somatório representa uma DFT de comprimento $N/2$. Definindo

$$G_k \triangleq \sum_{r=0}^{(N/2)-1} x_{2r} \omega_{N/2}^{rk}$$

e

$$H_k \triangleq \sum_{r=0}^{(N/2)-1} x_{(2r+1)} \omega_{N/2}^{rk},$$

a Equação 3.7 pode ser reescrita da seguinte forma

$$X_k = G_k + \omega_N^k H_k, \quad k = 0, 1, \ldots, N-1. \tag{3.8}$$

Este algoritmo é conhecido como algoritmo em dizimação no tempo, pois, recursivamente, divide a sequência x_n em subsequências. Embora o índice k percorra todos valores de N, $k = 0, 1, \ldots, N-1$, cada um dos somatórios deve ser computado apenas quando k estiver na faixa de 0 a $N/2-1$, pois G_k e H_k são periódicos em k com período $N/2$. Após as duas DFT's terem sido computadas, elas serão combinadas conforme a Equação 3.8. Este procedimento é recursivamente aplicado até que cada uma das DFT's resultante seja de comprimento $N = 2$. A Figura 3.2 ilustra o procedimento para $N = 16$; em cada nível é aplicada a Equação 3.8. Como $N = 2^v$, isto é feito no máximo $v = \log_2 N$ vezes, tal que, após conseguir esta decomposição tantas vezes quanto possível, o número de multiplicações e adições é igual a $Nv = N \log_2 N$. Haja vista que temos $\log_2 N$ estágios, tem-se um total de $(N/2) \log_2 N$ de multiplicações e adições complexas.

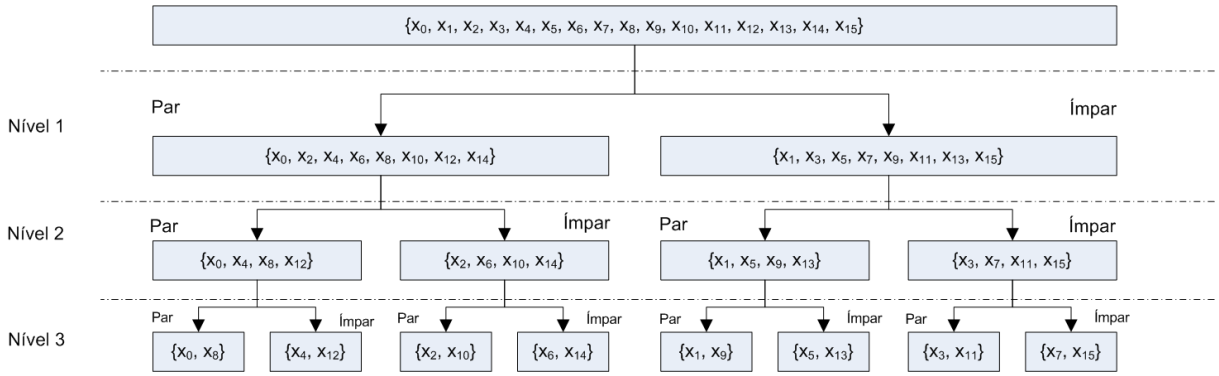


Figura 3.2: Decomposição da sequência de entrada até N=2 na FFT de Cooley-Tukey base 2.

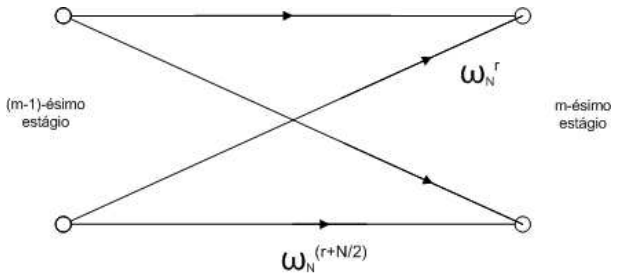


Figura 3.3: Célula básica (borboleta) para computação da FFT de Cooley-Tukey base 2.

Observando a Figura 3.2, nota-se que, do atual para o próximo estágio, a computação básica tem a forma mostrada na figura 3.3, ou seja, envolve a obtenção de um par de valores em um estágio, a partir de um par de valores do anterior, em que os coeficientes são sempre potências de ω_N e os expoentes são separados por $N/2$. Por causa da forma do grafo, esta computação elementar é chamada de borboleta (do inglês *butterfly*).

Desde que,

$$\omega_N^{N/2} = e^{-j(2\pi/N)N/2} = e^{-j\pi} = -1,$$

o fator $\omega_N^{r+N/2}$ é escrito como

$$\omega_N^{r+N/2} = \omega_N^{N/2}\omega_N^r = -\omega_N^r.$$

Com isso, a computação da borboleta da Figura 3.3 é simplificada como na Figura 3.4, que requer apenas uma multiplicação complexa em vez de duas.

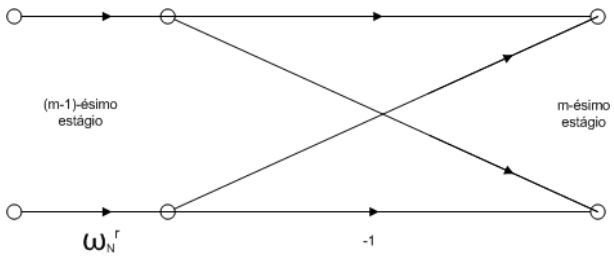


Figura 3.4: Célula básica simplificada para a computação da FFT de Cooley-Tukey base 2.

3.2 O algoritmo de Good-Thomas

Este algoritmo é também conhecido como Algoritmo do Fator Primo (PFA, do inglês *Prime Factor Algorithm*) e é baseado na fatoração do comprimento $N = N_1 N_2$ em potências de primos distintos. O algoritmo é ilustrado na Figura 3.5, para caso $N = 15$. N_1 e N_2 devem ser relativamente primos e o mapeamento que arranja os N componentes da sequência a ser transformada em uma matriz $N_2 \times N_1$, é baseado no Teorema Chinês do Resto. A Figura 3.6 mostra um exemplo de como os dados de entrada são re-arranjados para uma sequência de comprimento $N = 15$. Os elementos do vetor de entrada são armazenados em uma matriz bidimensional, iniciando pelo canto superior esquerdo e listando os componentes como uma "diagonal estendida". Como o número de colunas e linhas são coprimos, a diagonal estendida passa por todos os elementos do vetor [106]. Após esta etapa, é computada a DFT da matriz. A ordem dos componentes na matriz de saída é diferente da ordem dos componentes na matriz de entrada.

A derivação do algoritmo de Good-Thomas é baseada no teorema chinês do resto [29]. O indíce de entrada é descrito pelos seus resíduos como segue:

$$\begin{aligned} n_1 &= n \pmod{N_1}, \\ n_2 &= n \pmod{N_2}. \end{aligned}$$

Este é o mapa dos indíces de entrada, n, "descendo"a diagonal estendida de uma matriz bidimensional (n_1, n_2). Pelo teorema chinês do resto, existem dois números inteiros N' e N'' tais que os índices de entrada são recuperados da seguinte forma:

$$n = n_1 N'' N_2 + n_2 N' N_1 \pmod{N},$$

em que N' e N'' são números inteiros que satisfazem

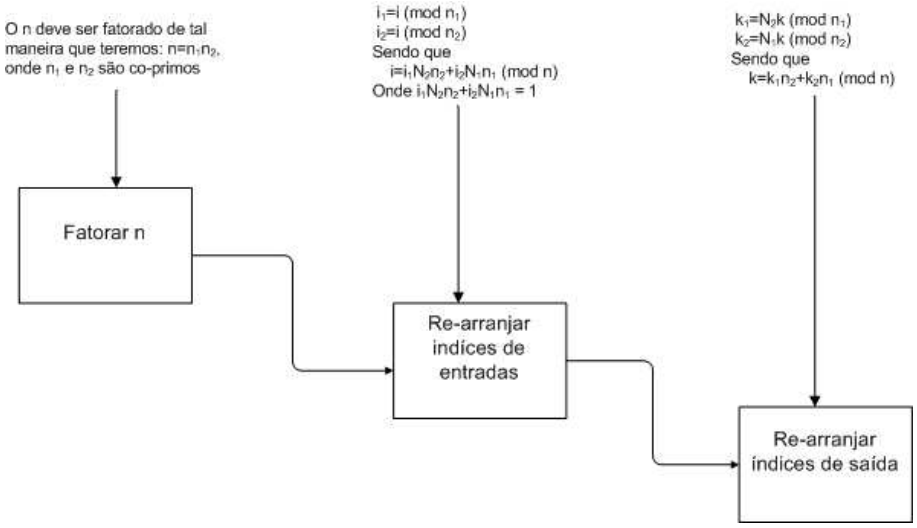


Figura 3.5: Diagrama ilustrando o procedimento da FFT de Good-Thomas.

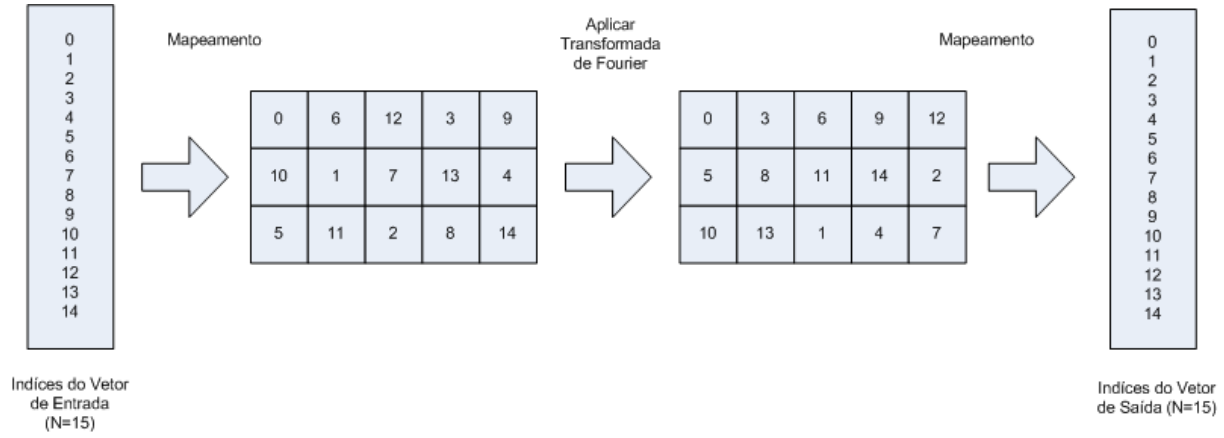


Figura 3.6: Exemplo da FFT de Good-Thomas para $N = 15$.

$$N'N_1 + N''N_2 = 1.$$

Os indíces de saída são descritos por

$$\begin{aligned} k_1 &= N''k \pmod{N_1}, \\ k_2 &= N'k \pmod{N_2}. \end{aligned}$$

Para este propósito, N' e N'' são reduzidos módulo N_1 e módulo N_2 respectivamente. O índice de saída k é recuperado como segue

$$k = n_2k_1 + n_1k_2 \pmod{N}.$$

Agora, com estes novos índices, convertemos a transformada discreta de Fourier

$$V_k = \sum_{n=0}^{N-1} \omega^{nk} x_n$$

na expressão

$$V_{n_2k_1+n_1k_2} = \sum_{n_2=0}^{N_2-1} \sum_{n_1=0}^{N_1-1} \omega^{(n_1N''N_2+n_2N'N_1)(N_2k_1+N_1k_2)} v_{n_1N''N_2+n_2N'N_1} \tag{3.9}$$

Mudando $N_2k_1 + N_1k_1$ por (k_1, k_2) e $n_1N''N_2 + n_2N'N_1$ por (n_1, n_2), vem

$$\begin{aligned} V_{k_1,k_2} &= \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \omega^{N''(N_2)^2 n_1k_1} \omega^{N'(N_1)^2 n_2k_2} v_{n_1n_2} \\ &= \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \beta^{n_1k_1} \gamma^{n_2k_2} v_{n_1,n_2}, \end{aligned} \tag{3.10}$$

em que $\beta = \omega^{N''(N_2)^2}$ e $\gamma = \omega^{N'(N_1)^2}$. Os termos β e γ são as N_1-ésima e N_2-ésima raízes unitária, respectivamente, da DFT de N_1-pontos e N_2-pontos. Note que $\beta = (\omega_2^N)^{N''N_2}$. Haja vista que $\omega^{N_2} = e^{-j2\pi/N_1}$ e $N''N_2 = 1 \pmod{N_1}$, logo $\beta = e^{-j2\pi/N_1}$. De maneira similar, temos que $\gamma = e^{-j2\pi/N_2}$.

A Equação 3.10 está agora no formato de uma DFT bidimensional $N_1 \times N_2$. O número de multiplicações é $N(N_1 + N_2)$. Se o comprimento das linhas/colunas for composto, as DFT's podem ser computadas por outra transformada rápida de Fourier. Desta maneira, uma transformada cujo comprimento N tem N_l fatores coprimos é computada numa forma que requer aproximadamente $N \sum_l N_l$ multiplicações e adições [29].

3.3 O algoritmo Rader-Primo

Este algoritmo usa a convolução para cálculo da DFT definida pela Equação 2.16, requer apenas algumas operações de indexação e uma convolução cíclica de comprimento $(N-1)$. É usado para computar a DFT em qualquer corpo F sempre que o comprimento da sequência é um primo p. Neste caso, podemos fazer uso da estrutura de $GF(p)$, campo de Galois de p elementos [107], para reindexar os componentes do vetor de entrada.

Se π denota um elemento primitivo de $GF(p)$, então cada elemento desse corpo pode ser expresso com uma potência de π [40].

No que se segue, a expressão da DFT (2.16) será reescrita com n e k sendo potências do elemento primitivo π. Como n e k podem assumir o valor zero e zero não é uma potência de π, então as componentes de frequência ($k = 0$) e tempo ($n = 0$) devem ser tratadas separadamente. Com isso, o cálculo das componentes da transformada será dada pelas Equações (3.11) e (3.12), para as componentes com índice zero e as demais, respectivamente.

$$V_0 = \sum_{n=0}^{p-1} \omega^{nk} v_n, \tag{3.11}$$

$$V_k = v_0 + \sum_{n=1}^{p-1} \omega^{nk} v_n \quad k = 1, \ldots, p-1. \tag{3.12}$$

Com n variando de 1 até $(p-1)$, seja $r(n)$ o inteiro tal que, em $GF(p)$, $\pi^{r(n)} = n$ e $\pi^{r(k)} = k$. A Equação (3.13) mostra uma nova forma de escrever a transformada.

$$V_{\pi^{r(k)}} = V_0 + \sum_{p-1}^{n=1} (\omega^{\pi^{r(n)+r(k)}} - 1) v_{\pi^{r(n)}}, \tag{3.13}$$

em que a função $r(n)$ é um mapeamento de $\{1, 2, \ldots, p-1\}$ em $\{1, 2, \ldots, p-1\}$, ou seja, uma permutação.

Definindo $r(k) = l$ e $r(n) = p - 1 - j$, e usando j como índice do somatório, temos a Equação 3.14,

$$\begin{aligned} V_{\pi^l} &= V_0 + \sum_{j=0}^{p-2} (\omega^{\pi^{l+p-1-j}}) v_{\pi^{p-1-j}} \\ &= V_0 + \sum_{j=0}^{p-2} (\omega^{\pi^{l-j}\pi^{p-1}}) v_{\pi^{p-1-j}} \\ &= V_0 + \sum_{j=0}^{p-2} (\omega^{\pi^{l-j}}) v_{\pi^{p-1-j}} \quad l = 0, \ldots, p-2 \end{aligned} \tag{3.14}$$

pois $\pi^{p-1} = 1$. Finalmente, a Equação (3.15) mostra uma modificação feita na Equação 3.14,

$$V_l^{'} - V_0 = \sum_{j=0}^{p-2} (\omega^{\pi^{l-j}}) v_j^{'} \quad l = 0, \ldots, p-2, \tag{3.15}$$

em que $V_l^{'} = V_{\pi^l}$ e $v_j^{'} = v_{\pi^{p-1-j}}$ são sequências permutadas da saída e entrada, respectivamente. Esta equação é uma convolução cíclica entre o vetor de entrada permutado $v = [v_j^{'}]$ e o vetor $g = [\omega^{\pi^j} - 1]$, ou seja, $\omega^{\pi^l} * v_l^{'}$. Defini-se o polinômio de Rader como sendo

$$g(x) = \sum_{j=0}^{p-2} (\omega^{\pi^j} - 1)v^j. \tag{3.16}$$

Pela permutação dos índices de entrada e saída, mudamos a transformada discreta de Fourier de comprimento (p) em uma convolução cíclica de comprimento ($p-1$), a qual pode ser computada por meio do algoritmo de Winograd [29].

3.4 Conclusões

Neste capítulo foi apresentada a transformada rápida de Fourier, que são algoritmos rápidos para computar a DFT. O ganho computacional foi demonstrado com relação a definição da DFT. Finda a apresentação do estado da arte, em que os algortimos de Cooley-Tukey, Good-Thomas e Radder-Primo foram detalhados. O próximo capítulo inicia as contribuições pessoais.

capítulo 4

Transformada Rápida de Fourier baseada em Série Matricial de Laurent

Neste capítulo uma nova transformada rápida de Fourier para sequências de comprimento $N = 8m + 4$, baseada em séries matriciais de Laurent, é apresentada. Resultados de complexidade aritmética expressos em multiplicações reais não-triviais são apresentados para comprimentos de bloco inferiores a 65, mostrando que o algoritmo tem um desempenho próximo às FFTs prévias. Uma descrição detalhada do algoritmo é feita para os casos em que $m = 1$ e $m = 2$. Este novo algoritmo rápido será implementado para comprimentos particulares de N, ou seja, do tipo $N \equiv 4(\text{mod } 8)$, porém o mesmo é estendido para $N \equiv 0(\text{mod } 4)$.

4.1 DFT como uma série matricial Laurent

O primeiro passo em direção a FFT proposta nesta tese é reescrever a Equação (2.16) na forma matricial:

$$\begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ \vdots \\ V_{N-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W & W^2 & \dots & W^{N-1} \\ 1 & W^2 & W^4 & \dots & W^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W^{N-1} & W^{2(N-1)} & \dots & W^{(N-1)(N-1)} \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ \vdots \\ v_{N-1} \end{pmatrix}. \tag{4.1}$$

ou $(V_k) = [DFT].(v_n)$. Haja vista que $W := e^{-j\frac{2\pi}{N}}$ tem ordem N, existem somente N potências distintas de W no conjunto $\{W^0, W^1, W^2, W^3, \dots, W^{(N-1)}\}$.

A FFT descrita neste capítulo lida com blocos de comprimento $N \equiv 4(\text{mod } 8)$ para garantir que exista sempre uma potência de W que produzirá os autovalores da DFT, ou seja, $\pm 1, \pm j$ [24]. Estes termos não contribuem para a complexidade multiplicativa, pois

$$W^0 = 1,\ W^{N/4} = -j,\ W^{N/2} = -1 \ e\ W^{3N/4} = j. \tag{4.2}$$

Os expoentes de W na Equação 4.2 geram um conjunto de quatro pontos que estão no eixo real ou imaginário. Este fato está associado ao conjunto

$$C_0 = \{0,\ N/4,\ N/2,\ 3N/4\}$$

e estamos procurando simetrias particulares nos quatro quadrantes do plano de Argand-Gauss [108].

O conjunto dos expoentes das potências distintas de W, $\{W^0, W^1, W^2, W^3, \ldots, W^{(N-1)}\}$, é então particionado em N classes (vale a pena observar que $4|N$):

$$C_m := \{x \in \mathbb{N} \cap [0, N) | 4x \equiv 4m(\text{mod } N)\},$$

em que $\mathbb{N}$ é o conjunto dos números naturais e $m = 0, 1, 2, \ldots, (N-1)$.

Proposição 4.1 *As classes $\{C_m\}$ produzem uma partição do conjunto dos inteiros $\{0, 1, 2, \ldots, (N-1)\}$, ou seja, $\forall m \neq m'$, $C_m \cap C_{m'} = \emptyset$ e $\bigcup_{m=0}^{N-1} C_m = \{0, 1, 2, \ldots, (N-1)\}$.*

Demonstração: Suponha (por redução ao absurdo) que exista um inteiro $m \neq m'$ tal que $C_m \cap C_{m'} \neq \emptyset$. Então, existe um elemento comum $x \in C_m$ e $x \in C_{m'}$ tal que $4x \equiv 4m(mod\, N)$ e $4x \equiv 4m'(mod\, N)$. Com isso $4m \equiv 4m'(mod\, N)$, que é o mesmo que $m \equiv m'(mod\, N/4)$, uma contradição. A cardinalidade do conjunto C_m para cada m é $||Cm|| = 4$. Existe $N/4$ classes disjuntas, tais que $\left\| \bigcup_{m=-(N/4-1)}^{N/4-1} C_m \right\| = 4(N/4) = 4$ e as classes C_m formam uma partição de $\{0, 1, 2, \ldots, N-1\}$. ■

Por simplicidade, lidamos apenas com matrizes de expoentes de W na matriz da DFT. Vamos definir uma matriz (N x N), $M := (kn(\text{mod } N))$, cujos elementos pertencem ao conjunto $\{0, 1, 2, \ldots, N-1\}$. Também definimos um operador χ_l sobre uma matriz $(N \times N)$, para cada $l = 0, 1, 2, \ldots, N-1$, que produzirá um nova matriz binária $(N \times N)$ cujos elementos são $\left(\delta_{l, m_{k,n}}\right)$, em que δ é o símbolo de Kronecker.

Finalmente, definimos uma matriz M_m associada com cada classe C_m, para $m = \pm 0, \pm 1, \pm 2, \ldots, \pm\left(\frac{N}{4}\right)/2$, dadas por

$$M_m := \sum_{l \in C_m} (-j)^{\frac{4(l-m)}{N}} \chi_l(M). \tag{4.3}$$

Por exemplo, $m = 0$ corresponde à parte aditiva da matriz de transformação da DFT:

$$M_0 = 1.\chi_0(M) - j.\chi_{N/4}(M) - 1.\chi_{N/2}(M) + j.\chi_{3N/4}(M).$$

Considere uma (possivelmente infinita) matriz A expressa em termos de blocos de matrizes na forma

$$A = (\ldots, A_{-1}, A_0, A_1, \ldots),$$

em que A_l são as submatrizes $(N \times N)$ de A. A partir dessa matriz, a seguinte expressão formaliza a série de potências chamada de série de Laurent da matriz A [109, 110]:

$$A(z) := \sum_{l=-\infty}^{+\infty} A_l z^l. \tag{4.4}$$

Então, $A(z)$ é uma série de Laurent com coeficientes matriciais. Em casos particulares em que $A(z) := \sum_{l=N_1}^{N_2} A_l.z^l$, $N_1,\ N_2\ \in\ Z$, então

$$g\ :=\ N_2 - N_1 + 1$$

é o genus de $A(z)$ e A [110].

Agora, vamos nomear M como a matriz associada com as submatrizes M_m:

$$M := \left(M_{-\frac{N/4-1}{2}}, \ldots, M_{-1}, M_0, M_1, \ldots, M_{\frac{N/4-1}{2}} \right). \tag{4.5}$$

A série de Laurent da matriz M é

$$M(z) := M_{-\frac{N/4-1}{2}} \frac{1}{z^{\frac{N/4-1}{2}}} + \ldots + M_{-2}\frac{1}{z^2} + M_{-1}\frac{1}{z} + M_0 + M_1 z + M_2 z^2 + \ldots + M_{\frac{N/4-1}{2}} z^{\frac{N/4-1}{2}}. \tag{4.6}$$

cujo genus é $g = N/4$ (nas ferramentas de banco de filtros, a notação da série de Laurent é conhecida como representação polifásica [111]).

A avaliação do espectro discreto de Fourier corresponde ao produto da sequência de dados em tempo discreto pela matriz de transformação da DFT, $[DFT] = M(z)\,|_{z=W}$, isto é, a matriz de transformação da DFT é

$$M(z)\,|_{z=W} = \sum_{m=-\frac{(N/4)-1}{2}}^{\frac{(N/4)-1}{2}} M_m W^m. \tag{4.7}$$

Haja vista que as multiplicações por W^m e $W^{-m} = (W^m)^*$ para um determinado valor de m são essencialmente equivalentes, estas matrizes são combinadas ao considerar M_{-m}, M_m e escrever este par de matrizes na forma escalonada padrão (SEF - *standard echelon form*, refenciado aqui como rref, (do inglês *row-reduced echelon form*), sendo este o nome da função nos aplicativos Matlab e Mathcad).

Por simplificação, consideremos somente as potências:
$\{W^0, W^1, W^3, \ldots, W^{(N-1)(N-1)}\}$;
$\{W^0, W^1, W^3, \ldots, W^{(N-1)}\}$;
$\{W^0, W^1, W^{-1}, W^2, W^{-2}, \ldots, W^{\frac{(N/4)-1}{2}}, W^{-\frac{(N/4)-1}{2}}\}$;
$\Re W = \cos\left(\frac{2\pi}{N}\right)$ e $\Im W = -\text{sen}\left(\frac{2\pi}{N}\right)$.

Para $N \equiv 0(\text{mod } 8)$ existe uma falta de simetria, com mais termos positivos que negativos na expressão 4.7. Por exemplo, para $N = 8$, a decomposição assume a forma: $M(z)\,|_{z=W} = M_0 + M_1 W$. Para $N = 16$ o algoritmo produz $M(z)\,|_{z=W} = M_{-1}W^* + M_0 + M_1 W + M_2 W^2$. Em geral, a única classe assimétrica nas séries, que é adicionada às classes simétricas é $C_{N/8},\ N \geq 8$.

4.2 O novo algoritmo FFT

O algoritmo rápido é escrito em termos das matrizes $\{M_m\}$ de acordo com as seguintes decomposições:

$$\Re eDFT = \left\{ \Re e(M_0) + \sum_{m=1}^{\frac{(N/4)-1}{2}} \Re e\,(M_m + M_{-m}) \cos(\frac{2\pi m}{N}) + \sum_{m=1}^{\frac{(N/4)-1}{2}} \Im m\,(M_m - M_{-m}) \sin(\frac{2\pi m}{N}) \right\}, \tag{4.8}$$

$$\Im mDFT = \left\{ \Im m(M_0) + \sum_{m=1}^{\frac{(N/4)-1}{2}} \Im m\,(M_m + M_{-m}) \cos(\frac{2\pi m}{N}) - \sum_{m=1}^{\frac{(N/4)-1}{2}} \Re e\,(M_m - M_{-m}) \sin(\frac{2\pi m}{N}) \right\}. \tag{4.9}$$

As matrizes $\Re e(M_0)$ e $\Re e(M_m \pm M_{-m})$ são escritas em SEF, e também as correspondentes matrizes $\Im m(M_0)$ e $\Im m(M_m \pm M_{-m})$.

A complexidade multiplicativa do algoritmo rápido é computada por

$$\sum_{m=1}^{(\frac{N}{4}-1)/2} posto \begin{pmatrix} \Re e(M_m + M_{-m}) \\ \Im m(M_m + M_{-m}) \end{pmatrix} + posto \begin{pmatrix} \Re e(M_m - M_{-m}) \\ \Im m(M_m - M_{-m}) \end{pmatrix}. \tag{4.10}$$

Em todos os casos examinados até aqui, nenhuma redução do *posto* foi alcançada quando empilhamos as matrizes $\Re e(M_m \pm M_{-m})$ e $\Im m(M_m \pm M_{-m})$, e a complexidade multiplicativa da FFT foi sempre dada por

$$\sum_{m=1}^{(\frac{N}{4}-1)/2} posto(\Re e(M_m + M_{-m})) + posto(\Im m(M_m + M_{-m})). \tag{4.11}$$

No exemplo $N = 8$, existem somente duas matrizes associadas com os termos multiplicativos, ou seja:

$$\Re e(M_1) = \Im m(M_1) = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{pmatrix},$$

então somente duas multiplicações por $\cos(\pi/4) = \sin(\pi/4)$ são requeridas. Vale a pena observar que o número de multiplicações reais é duas unidades menor que a computada pela Equação 4.10 quando $N \equiv 0 (\text{mod } 4)$, pois uma multiplicação por $e^{j\pi/4}$ é incluída.

4.3 Uma FFT para blocos de comprimento $N = 12$

Para $N = 12$, iniciamos reunindo os elementos dos expoentes na classe $\{0, 3, 6, 9\}$, que não está associada com multiplicações (veja 2.16): isto corresponde ao conjunto C_0. A matriz M com os expoentes dos termos da matriz DFT é

$$M = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 0 & 2 & 4 & 6 & 8 & 10 & 0 & 2 & 4 & 6 & 8 & 10 \\ 0 & 3 & 6 & 9 & 0 & 3 & 6 & 9 & 0 & 3 & 6 & 9 \\ 0 & 4 & 8 & 0 & 4 & 8 & 0 & 4 & 8 & 0 & 4 & 8 \\ 0 & 5 & 10 & 3 & 8 & 1 & 6 & 11 & 4 & 9 & 2 & 7 \\ 0 & 6 & 0 & 6 & 0 & 6 & 0 & 6 & 0 & 6 & 0 & 6 \\ 0 & 7 & 2 & 9 & 4 & 11 & 6 & 1 & 8 & 3 & 10 & 5 \\ 0 & 8 & 4 & 0 & 8 & 4 & 0 & 8 & 4 & 0 & 8 & 4 \\ 0 & 9 & 6 & 3 & 0 & 9 & 6 & 3 & 0 & 9 & 6 & 3 \\ 0 & 10 & 8 & 6 & 4 & 2 & 0 & 10 & 8 & 6 & 4 & 2 \\ 0 & 11 & 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \end{pmatrix}.$$

Somente existem $N/4 = 3$ classes, ou seja:

$C_0 = (0, 3, 6, 9), \quad 0 \equiv 24 \equiv 12 \equiv 36 (\text{mod } 12)$

$C_1 = (1, 4, 7, 10), \quad 4 \equiv 28 \equiv 16 \equiv 40 (\text{mod } 12)$

$C_{-1} = (11, 2, 5, 8), \quad 44 \equiv 20 \equiv 8 \equiv 32 (\text{mod } 12)$

Neste caso particular, o maior indice é $(N/4-1)/2 = 1$. Ou seja, C_{-1}, C_0, C_1 são partições de $\{0, 1, 2, \ldots, 11\}$, como esperado.

É direto observar que dado C_0, os elementos de C_1 são derivados ao se adicionar $1 (\text{mod } N)$ a cada elemento de C_0; os de C_{-1}, ao subtrair $1 (\text{mod } N)$ a cada elemento de C_0, e assim por diante.

Em seguida, para elucidar a abordagem, assumimos o conjunto das potências de W:

$\{1, W, W^2, W^3, W^4, W^5, W^6, W^7, W^8, W^9, W^{10}, W^{11}\}$

Haja vista que $W^0 = 1$, $W^3 = -j$, $W^6 = -1$, $W^9 = j$, as seguintes classes são consideradas:

$C_0 = \{0, 3, 6, 9\} \ \Rightarrow \ 1, -j, -1, j.$

$C_1 = \{1, 4, 7, 10\} \Rightarrow \ W^1 = 1.W, W^4 = -j.W, W^7 = -W, W^{10} = j.W.$

$C_{-1} = \{11, 2, 5, 8\} \ \Rightarrow \ W^{11} = W^*, W^2 = -jW^*, W^5 = -W^*, W^8 = j.W^*.$

As operações envolvendo produtos pelos autovalores (elementos de C_0) e/ou o conjugado de um complexo não devem ser consideradas como multiplicações em ponto flutuante.

As matrizes de interesse no algoritmo são:

1. $M_0 = 1\chi_0(M) - j\chi_3(M) - 1\chi_6(M) + j\chi_9(M)$.

Esta matriz aditiva M_0 é separada em suas partes real e imaginária.
A parte real da matriz M_0 é

$$\Re e(M_0) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

que fornece o $posto(\Re e(M_0)) = 6$.

Na SEF, a parte real da matriz é $rref(\Re e(M_0))$ é

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

A matriz $\Im m(M_0)$ é

$$\Im m(M_0) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \end{pmatrix},$$

que, por sua vez, produz $posto(\Im m(M_0)) = 2$.

Na SEF, a matriz imaginária é

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \end{pmatrix}.$$

2. $M_1 = 1\chi_1(M) - 1\chi_7(M) - j\chi_4(M) + j\chi_1 0(M)$.
A parte real da matriz M_1 é

$$\Re e(M_1) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

então, $posto(\Re e(M_1)) = 2$; a SEF da $\Re e(M_1)$ é

$$LI_1 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}.$$

A parte imaginária da matriz M_1 é

$$\Im m(M_0) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \end{pmatrix},$$

e $posto(\Im m(M_1)) = 6$; a SEF da $\Im m(M_1)$ é

$$LI_2 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

3. $M_{-1} = 1\chi_1 1(M) - 1\chi_5(M) - j\chi_2(M) + j\chi_8(M)$.
A parte real da matriz M_{-1} é

$$\Re e(M_{-1}) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Então, o $posto(\Re e(M_{-1})) = 2$; além disso, a SEF da matriz é exatamente a mesma da matriz $\Re e(M_1)$, ou seja, $LI_3 = LI_1$.

A parte imaginária da matriz M_{-1} é

$$\Im m(M_{-1}) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{pmatrix}.$$

Desse modo, o $posto(\Im m(M_{-1})) = 6$, sendo que a SEF desta matriz é a mesma de $\Im m(M_1)$, ou seja, $LI_4 = LI_2$. Na realidade, $\Im m(M_{-1})$ é esencialmente uma permutação de linha (or coluna) de $\Im m(M_1)$. Em seguida, para avaliar a complexidade multiplicativa da FFT de comprimento 12, determina-se o *posto* das matrizes:

$\Re e(M_m + M_{-m})$ e $\Im m(M_m + M_{-m})$,

$\Re e(M_m - M_{-m})$ e $\Im m(M_m - M_{-m})$.

As quatro matrizes de pré-adições associadas com os ramos multiplicativos do algoritmo são

$rref\Re e(M_1 + M_{-1}) = (01000 - 10 - 10001)$ e

$$rref\Im m(M_1 - M_{-1}) = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{4.12}$$

$rref\Re e(M_1 + M_{-1}) = (0100010 - 1000 - 1)$

$$rref\Im m(M_1 - M_{-1}) = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \end{pmatrix} \tag{4.13}$$

Por simplicidade, tais matrizes são colocadas na forma compacta. Nesta notação os índices indicam o número de repetições que o mesmo tem na matriz original:

$$(010_3 \pm 10 - 10_3 \pm 1) \begin{pmatrix} 0 & 1 & 0_3 & \pm 1 & 0 & -1 & 0_3 & \pm 1 \\ 0_2 & 1 & 0_7 & \pm 1 & 0 & & & \\ 0_4 & 1 & 0_3 & \pm 1 & 0_3 & & & \end{pmatrix}$$

As Figuras 4.1 e 4.2 apresentam o diagram de blocos do algoritmo FFT, separando as partes real e imaginária, respectivamente. A complexidade multiplicativa total é de quatro multiplicações reais, sendo que não são consideradas as quatros multiplicações por $\text{sen}(\pi/6) = 0,5$. Este atende ao limite inferior de Heideman [53] e é muito distante das 144 multiplicações necessárias para computar a DFT por sua definição.

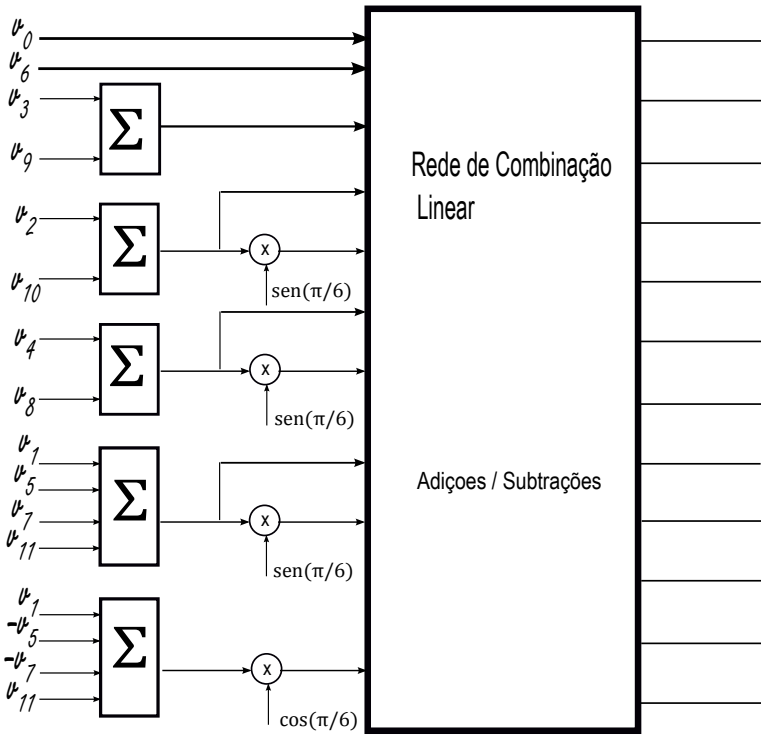


Figura 4.1: Diagrama para computação da parte real de uma DFT de comprimento $N = 12$.

De acordo com esta abordagem, as amostras são combinadas por:

$v_1 \mp v_5;\ v_7 \pm v_{11};\ v_2 \pm v_{10};\ v_4 \pm v_8$.

A apresentação aqui foi dividida em duas figuras de modo a esclarecer a natureza intrinseca do algoritmo FFT proposto: a modificação necessária no circuito na parte real ($\Re e$) para computar a correspondente parte imaginária ($\Im m$) é inverter o sinal das amostras de entrada.

4.4 Comentários sobre a FFT para outros comprimentos

Para $N = 20$, existem exatamente $N/4 = 5$ classes, correspondendo a $m = 0, \pm 1, \pm 2$:

$C_0 = \{0, 5, 10, 15\},\ C_1 = \{1, 6, 11, 16\},\ C_{-1} = \{19, 4, 9, 14\},\ C_2 = \{2, 7, 12, 17\}$ e $C_{-2} = \{18, 3, 8, 13\}$.

As matrizes correspondentes são

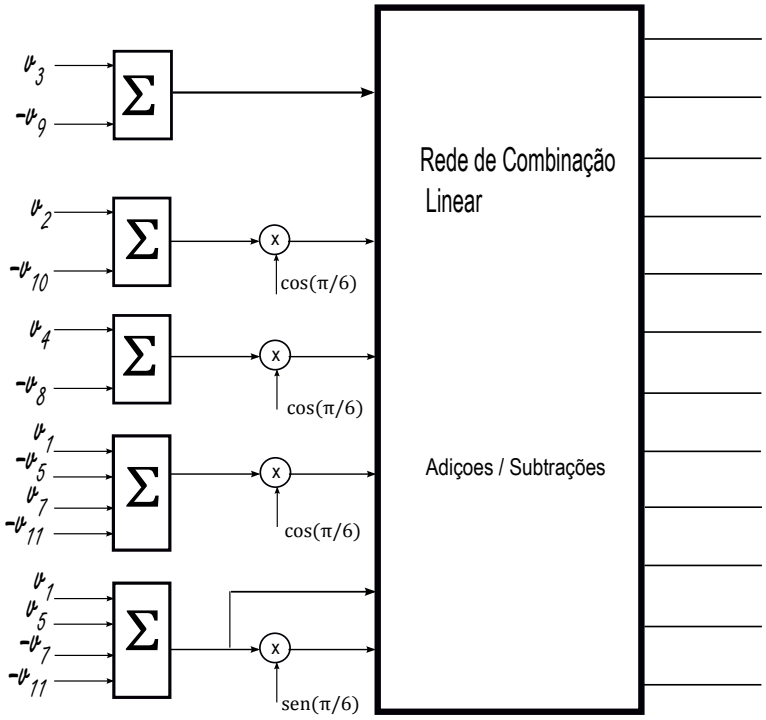


Figura 4.2: Diagrama para computação da parte imaginária de uma DFT de comprimento $N = 12$.

$$rref\Re e(M_1 + M_{-1}),\ rref\Re e(M_1 - M_{-1}),$$
$$rref\Im m(M_1 + M_{-1}),\ rref\Im m(M_1 - M_{-1}),$$
$$rref\Re e(M_2 + M_{-2}),\ rref\Re e(M_2 - M_{-2}),$$
$$rref\Im m(M_2 + M_{-2}),\ rref\Im m(M_2 - M_{-2}).$$

ou seja,

$$\begin{pmatrix} 0 & 1 & 0_7 & \pm 1 & 0 & -1 & 0_7 & \pm 1 & \\ 0_3 & 1 & 0_3 & \pm 1 & 0_5 & -1 & 0_3 & \pm 1 & 0_2 \end{pmatrix}$$

e

$$\begin{pmatrix} 0 & 1 & 0_7 & \pm 1 & 0 & -1 & 0_7 & \pm 1 & \\ 0_2 & 1 & 0_{15} & \pm 1 & 0 & & & & \\ 0_3 & 1 & 0_3 & \pm 1 & 0_5 & 1 & 0_3 & \pm 1 & 0_2 \\ 0_4 & 1 & 0_{11} & \pm 1 & 0_3 & & & & \\ 0_6 & 1 & 0_7 & \pm 1 & 0_5 & & & & \\ 0_8 & 1 & 0_3 & \pm 1 & 0_7 & & & & \end{pmatrix}$$

A Tabela 4.1 e 4.2 apresentam o número de multiplicações reais em ponto flutuante e de adições, respectivamente, necessárias para computar a FFT para blocos de comprimento

$N \leq 60$. Vale resaltar que a segunda coluna indica um *benchmark* (pseudo-algoritmo) para servir de comparação com comprimentos diferentes das potências de 2.

Tabela 4.1: Complexidade multiplicativa e aditiva da FFT baseada na Série Matricial de Laurent

N	$N \log_2 N$ (arredondado)	Laurent-FFT
12	43	8
20	86	32
28	135	72
36	186	88
44	240	200
52	296	288
60	354	208

Tabela 4.2: Complexidade aditiva da FFT baseada na Série Matricial de Laurent

N	Laurent-FFT
12	46
20	191
28	244
36	475
44	625
52	858
60	1087

Uma comparação com o limite inferior de Heideman (Equação 2.20) não foi feita na Tabela 4.1, pois μ_{DFT} fornece o número mínimo de multiplicações complexas. A Tabela 4.4 apresenta a complexidade aditiva para a FFT baseada na série matricial de Laurent.

O algoritmo rápido introduzido aqui pode ser usado também para qualquer bloco de comprimento $N \equiv 0 (\text{mod } 4)$, pois assegura a presença dos quatro autovalores da DFT, mas não há simetria ideal na série formal. Com isso, mesmo embora esta FFT não tenha sido concebida primariamente para blocos de comprimento que seja uma potencia de quatro [111, 112], o algoritmo também pode ser usado neste caso e resultados de complexidade são mostrados na Tabela 4.3, em comparação com o algoritmo padrão Cooley-Tukey base-2. O limitante de Heideman-Burrus [113] no número mínimo de multiplicações reais necessário para computar uma DFT de comprimento $N = 2^n$ é $\mu_r = 4N - 2\{(\log_2 N)^2 + (\log_2 N) + 2\}$. Então, mesmo se tais comprimentos não são a principal preocupação do algoritmo, o número de

Tabela 4.3: Complexidade multiplicativa da FFT baseada na Série Matricial de Laurent para comprimento de potências de 2

N	$N \log_2 N$ (arredondado)	Radix-2 (real nontrivial)	Rader-Brenner	Heideman-Burrus $\mu_r(N)$	Laurent-FFT
8	24	4	4	2	2
16	64	12	10	10	12
32	160	88	34	16	54
64	384	264	196	84	224

Tabela 4.4: Complexidade aditiva da FFT baseada na Série Matricial de Laurent para comprimentos de potências de 2

N	Laurent-FFT
8	22
16	82
32	354
64	1254

multiplicações necessário para esta nova FFT é próximo ao valor de μ_r. Com isso, nenhum fato conflitante existe aqui, pois simetrias particulares (tais como $e^{-j\pi/4}$) provavelmente não foram contabilizadas em [113].

4.5 Conclusões

Um novo algoritmo rápido, proposto no escopo desta tese, para computar a DFT de comprimento $N \equiv 4 (\text{mod } 8)$ foi apresentado, que é baseado nas simetrias das matrizes associadas com o desenvolvimento em séries de Laurent; com isso, apresentou-se uma FFT para comprimentos que não sejam as costumeiras potências de dois. Um simples e ilustrativo exemplo é apresentado em detalhes para $N = 12$, mas o procedimento inteiro é sistemático. A complexidade multiplicativa da FFT é avaliada, sendo alcançados valores menores que $N \log_2 N$, para $N = 12, 20, 28, 36, 44, 52, 60$. Ainda que exista um grande número de técnicas diferentes e elegantes para análise de espectro, incluindo a abordagem aritmética [56, 114] ou transformada de wavelets [115], que estão entre as melhores escolhas, as FFTs ainda são uma técnica extremamente difundida. A FFT apresentada aqui é também de fácil implementação usando DSP ou Circuitos Integrados de alta velocidade e baixo custo, como mostrado no Capítulo 6.

capítulo 5

Expansão Matricial para Cálculo da Transformada Discreta de Hartley

Ao longo dos anos, algoritmos rápidos, em termos de complexidade multiplicativa, foram introduzidos para o cálculo da transformada discreta de Hartley (DHT) [83, 84, 90, 105, 116]. Neste capítulo, um novo algoritmo rápido para computação da DHT é apresentado para sequências de comprimento $N \equiv 0 (\text{mod } 4)$, o qual é baseado na expansão da matriz da transformada. Em termos de complexidade multiplicativa, o algoritmo apresenta um melhor desempenho que algoritmos anteriormente conhecidos para a transformada de Hartley. Uma descrição detalhada do cálculo da DHT com comprimento de bloco 8 e 16 é mostrada. Alguns algoritmos ótimos são apresentados para os comprimentos $12, 16$ e 24, além do comprimento $N = 8$ que não necessitou de otimizações.

5.1 Expandindo a matriz da DHT

O primeiro passo em direção à FHT (do inglês *fast Hartley transform*) proposta nesta tese é reescrever a Equação 2.21 na forma matricial

$$\begin{pmatrix} H_0 \\ H_1 \\ H_2 \\ \vdots \\ H_{N-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & cas(1) & cas(2) & \dots & cas(N-1) \\ 1 & cas(2) & cas(4) & \dots & cas(2(N-1)) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & cas(N-1) & cas(2(N-1)) & \dots & cas((N-1)(N-1)) \end{pmatrix} \begin{pmatrix} h_0 \\ h_1 \\ h_2 \\ \vdots \\ h_{N-1} \end{pmatrix}, \tag{5.1}$$

ou $H(k) = [DHT]h(n)$, em que, por simplicidade, denota-se $cas(\frac{2\pi}{N}kn)$ por $cas(kn)$. Considerando que, para l e r inteiros, $cas(\frac{2\pi}{N}l) = cas(\frac{2\pi}{N}(l + rN))$, somente existem N argumentos distintos de $cas(.)$ na matriz da transformada. Para todo comprimento de bloco par do tipo $N = 4m$, existe um argumento ki que produz um dos autovalores da DHT, a saber $1, -1$. Estes termos não contribuem para a complexidade multiplicativa, pois

$$cas(0) = cas(N/4) = 1 \tag{5.2}$$

e

$$cas(N/2) = cas(3N/4) = -1. \tag{5.3}$$

Os argumentos de $cas(.)$ nas Expressões 5.2 e 5.3 geram um conjunto de dois pontos que pertencem ao eixo real. Este fato está associado com a classe

$$C_0 = \{0, N/4, N/2, 3N/4\}.$$

O conjunto de argumentos distintos de $cas(.)$, $\mathbb{Z}_N = \{0, 1, 2, \dots, N-1\}$, é então particionado em classes C_m, $m = 0, \pm1, \pm2, \dots, \pm\frac{n-1}{2}$, denominadas classes do cas, de modo que

- Para $N = (2k+1)4$, existem $n = \frac{N}{4} = 2k+1$ classes;
- Para $N = (2k)4$, existem $n = \frac{N}{4} + 1 = 2k+1$ classes.

A Tabela 5.1 descreve as classes C_m; observe que $|C_m| = 4,\ \forall m$.

Exemplo 5.1

1. Para $N = 20$, tem-se $n = N/4 = 5$ classes, descritos na tabela

C_m	Elementos	de	C_m	
C_2	2	12	3	13
C_1	1	11	4	14
C_0	0	10	5	15
C_{-1}	19	9	6	16
C_{-2}	18	8	7	17

2. Para $N = 16$, tem-se $n = \frac{N}{4} + 1 = 5$ classes, descritos na tabela

C_m	Elementos	de	C_m	
C_2	2	10	2	10
C_1	1	9	3	11
C_0	0	8	4	12
C_{-1}	15	7	5	13
C_{-2}	14	6	6	14

•

Tabela 5.1: Descrição das classes C_m do cas.

C_m	Elementos	de	C_m	
$C_{(\frac{n-1}{2})}$	$\frac{n-1}{2}$	$\frac{N}{2} + \frac{n-1}{2}$	$\frac{N}{4} - \frac{n-1}{2}$	$\frac{3N}{4} - \frac{n-1}{2}$
$\vdots$	$\vdots$	$\vdots$		
C_3	3	$\frac{N}{2} + 3$	$\frac{N}{4} - 3$	$\frac{3N}{4} - 3$
C_2	2	$\frac{N}{2} + 2$	$\frac{N}{4} - 2$	$\frac{3N}{4} - 2$
C_1	1	$\frac{N}{2} + 1$	$\frac{N}{4} - 1$	$\frac{3N}{4} - 1$
C_0	0	$\frac{N}{2}$	$\frac{N}{4}$	$\frac{3N}{4}$
C_{-1}	$N - 1$	$\frac{N}{2} - 1$	$\frac{N}{4} + 1$	$\frac{3N}{4} + 1$
C_{-2}	$N - 2$	$\frac{N}{2} - 2$	$\frac{N}{4} + 2$	$\frac{3N}{4} + 2$
C_{-3}	$N - 3$	$\frac{N}{2} - 3$	$\frac{N}{4} + 3$	$\frac{3N}{4} + 3$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$C_{-(\frac{n-1}{2})}$	$N - \frac{n-1}{2}$	$\frac{N}{2} - \frac{n-1}{2}$	$\frac{N}{4} + \frac{n-1}{2}$	$\frac{3N}{4} + \frac{n-1}{2}$

No exemplo 5.1(2) observa-se que a classe C_2 e a classe C_{-2} apresentam elementos repetidos. Esse não é um fato isolado e ocorre sempre que N é um múltiplo par de 4, como mostrado nas Proposições 5.1 e 5.2

Proposição 5.1 *Se N é um múltiplo par de 4, então, $C_{(\frac{n-1}{2})}$:*

1. *Os elementos $\frac{n-1}{2}$ e $\frac{N}{4} - \frac{n-1}{2}$ são iguais.*
2. *Os elementos $\frac{N}{2} + \frac{n-1}{2}$ e $\frac{3N}{4} - \frac{n-1}{2}$ são iguais.*

Demonstração: Como para $N = (2k)4$, tem-se $n = \frac{N}{4} + 1$, então

1. $\frac{N}{4} - \frac{n-1}{2} = n - 1 - \frac{n-1}{2} = \frac{n-1}{2}$.
2. $\frac{N}{2} + \frac{n-1}{2} = \frac{N}{2} + \frac{N}{8} = \frac{5N}{8} = \frac{6N}{8} - \frac{N}{8} = \frac{3N}{4} - \frac{n-1}{2}$.

■

Proposição 5.2 *Se N é um múltiplo par de 4, então, na classe $C_{-(\frac{n-1}{2})}$:*

1. *Os elementos $N - \frac{n-1}{2}$ e $\frac{3N}{4} + \frac{n-1}{2}$ são iguais.*
2. *Os elementos $\frac{N}{2} - \frac{n-1}{2}$ e $\frac{N}{4} + \frac{n-1}{2}$ são iguais.*
3. $cas(N - \frac{n-1}{2}) = cas(\frac{N}{2} - \frac{n-1}{2}) = 0$.

Demonstração: De forma análoga à demonstração da Proposição 5.1 tem-se

1. $N - \frac{n-1}{2} = N - \frac{N}{8} = \frac{7N}{8} = \frac{7N}{8} + \frac{N}{8} = \frac{3N}{4} + \frac{n-1}{2}$.
2. $\frac{N}{2} - \frac{n-1}{2} = \frac{N}{2} - \frac{N}{8} = \frac{3N}{8} = \frac{N}{4} + \frac{N}{8} = \frac{N}{4} + \frac{n-1}{2}$.
3. $cas(N - \frac{n-1}{2}) = cas(N - \frac{N}{8}) = cas(\frac{7N}{8}) = cas(\frac{2\pi}{N}\frac{7N}{8}) = cas(\frac{7\pi}{4}) = 0$.
   $cas(\frac{N}{2} - \frac{n-1}{2}) = cas(\frac{N}{2} - \frac{N}{8}) = cas(\frac{3N}{8}) = cas(\frac{2\pi}{N}\frac{3N}{8}) = cas(\frac{3\pi}{4}) = 0$.

■

Considerando as Proposições 5.1 e 5.2, as classes do *cas* para $N = (2k)4$ são mostradas na Tabela 5.2. Como na classe $C_{-(\frac{n-1}{2})}$ todos os valores do $cas(.)$ são nulos, a mesma não é apresentada na tabela. Além disso, os elementos repetidos na classe $C_{(\frac{n-1}{2})}$ não são mostrados. Diante desses fatos, no que se segue consideramos que existem $\frac{N}{4}$ classes, $\forall\ N \equiv 0(\text{mod } 4)$.

Exemplo 5.2

Para $N = 24$, tem-se $n = N/4 + 1 = 7$ classes do *cas* descritos na tabela

•

C_m	Elementos	de	C_m	
C_3	3	15		
C_2	2	14	4	16
C_1	1	13	5	17
C_0	0	12	6	18
C_{-1}	23	11	7	19
C_{-2}	22	10	8	20

Tabela 5.2: Descrição das classes do cas para $N = (2k)4$.

C_m	Elementos	de	C_m	
$C_{(\frac{n-1}{2})}$	$\frac{n-1}{2}$	$\frac{N}{2}+\frac{n-1}{2}$		
$\vdots$	$\vdots$	$\vdots$		
C_3	3	$\frac{N}{2}+3$	$\frac{N}{4}-3$	$\frac{3N}{4}-3$
C_2	2	$\frac{N}{2}+2$	$\frac{N}{4}-2$	$\frac{3N}{4}-2$
C_1	1	$\frac{N}{2}+1$	$\frac{N}{4}-1$	$\frac{3N}{4}-1$
C_0	0	$\frac{N}{2}$	$\frac{N}{4}$	$\frac{3N}{4}$
C_{-1}	$N-1$	$\frac{N}{2}-1$	$\frac{N}{4}+1$	$\frac{3N}{4}+1$
C_{-2}	$N-2$	$\frac{N}{2}-2$	$\frac{N}{4}+2$	$\frac{3N}{4}+2$
C_{-3}	$N-3$	$\frac{N}{2}-3$	$\frac{N}{4}+3$	$\frac{3N}{4}+3$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$C_{-(\frac{n-3}{2})}$	$N-\frac{n-3}{2}$	$\frac{N}{2}-\frac{n-3}{2}$	$\frac{N}{4}+\frac{n-3}{2}$	$\frac{3N}{4}+\frac{n-3}{2}$

Vamos introduzir a matriz de argumentos do $cas(.)$ na Equação 2.21, uma matriz $N \times N$ $A := (a_{kn})$ em que $a_{kn} = (kn(\text{mod } N))$. O operador B_l aplicado à matriz $N \times N$, para cada $l = 0, 1, 2, \ldots, N-1$, produz uma matriz binária $N \times N$ cujos elementos são $(\delta_{l,m_{k,n}})$, em que δ é o símbolo de Kronecker.

Associada à cada classe C_m definimos a matriz M_m como

$$M_m := \sum_{l \in C_m} sgn(cas(l)) B_l(A), \tag{5.4}$$

em que $sgn(x)$ retorna o sinal de x. Assim, por exemplo, $m = 0$ corresponde à parte aditiva da matriz da DHT,

$$M_0 = B_0(A) - B_{N/2}(A) + B_{N/4}(A) - B_{3N/4}(A).$$

As seguintes proposições mostram as simetrias da função $cas(.)$, que são importantes na construção do algoritmo rápido descrito neste capítulo.

Proposição 5.3 *1.* $cas(m + \frac{N}{2}) = -cas(m)$.

2. $cas(m + \frac{N}{4}) = cas(N - m)$.

3. $cas(m + \frac{3N}{4}) = -cas(N - m)$.

Demonstração:

1. $cas(m + \frac{N}{2}) = cas(\frac{2\pi}{N}(m + \frac{N}{2})) = cas(\frac{2\pi m}{N} + \pi) = \cos(\frac{2\pi m}{N} + \pi) + sen(\frac{2\pi m}{N} + \pi) = -cas(m)$.
2. $cas(m + \frac{N}{4}) = cas(\frac{2\pi}{N}(m + \frac{N}{4})) = cas(\frac{2\pi m}{N} + \frac{\pi}{2}) = \cos(\frac{2\pi m}{N} + \frac{\pi}{2}) + sen(\frac{2\pi m}{N} + \frac{\pi}{2}) = cas(N - m)$.
3. $cas(m + \frac{3N}{4}) = cas(\frac{2\pi}{N}(m + \frac{3N}{4})) = cas(\frac{2\pi m}{N} + \frac{3\pi}{2}) = \cos(\frac{2\pi m}{N} + \frac{3\pi}{2}) + sen(\frac{2\pi m}{N} + \frac{3\pi}{2}) = -cas(N - m)$.

■

A partir da matriz M_m, $m = 0, 1, 2, \ldots, \frac{N}{4} - 1$, a matriz da DHT é expressa pela seguinte expansão:

1. Para N um múltiplo par de 4:

$$[DHT] = \sum_{m=-((((N/4)-1)/2)+1)}^{((N/4)-1)/2} M_m cas(m). \tag{5.5}$$

2. Para N um múltiplo ímpar de 4:

$$[DHT] = \sum_{m=-(((N/4)-1)/2)}^{((N/4)-1)/2} [C_c(M_m) + C_s(M_m)]cas(m), \tag{5.6}$$

em que as matrizes $C_c(M_m)$ e $C_s(M_m)$ são definidas como

$$C_c(M_m) = B_m(A) - B_{m+\frac{N}{2}}(A),$$
$$C_c(M_m) = B_{m+\frac{N}{4}}(A) - B_{m+\frac{3N}{4}}(A).$$

A complexidade multiplicativa dos algoritmos (Equações 5.5 e 5.6) é computada, respectivamente, por

$$\sum_{m=-(((N/4)-1)/2+1)}^{((N/4)-1)/2} posto(M_m) \tag{5.7}$$

e

$$\sum_{m=-(((N/4)-1)/2)}^{((N/4)-1)/2} posto(C_c(M_m)). \tag{5.8}$$

A complexidade aditiva pode ser computada da seguinte maneira:

1. Empilhamos as matrizes que representam as pré-adições, por

$$MAdd = \sum_{m=0}^{(N/4)-1} empilhar(M_m). \tag{5.9}$$

2. Somente com as linhas distintas de $MAdd$ fazemos a contagem das operações por

$$\sum_{i=0}^{\#linhas(MAdd)} MAdd[i,] - 1. \tag{5.10}$$

3. Para cada matriz de pré-adição, M, existe uma matriz representando o somatório final da FHT, cuja contagem é feita por

$$\sum_{i=0}^{\#linhas} \sum_{m=((N/4-1)/2)+1}^{(N/4)-1} MF_m[i,] - 1. \tag{5.11}$$

Ressalta-se que o procedimento apresentado no Apêndice A deve ser aplicado tanto na matriz $MAdd$ quanto nas matrizes MF.

O procedimento para computar a DHT é sumarizado da seguinte maneira:

1. Calcular a matriz de argumentos (A);
2. Calcular a matriz de classes;
3. Repetir para todas as classes:
   - Computar a matriz binária dada pelas Equações 5.5 e 5.6;
   - Computar a matriz binária na forma echelon padrão (SEF), referida aqui como $rref$;
   - Computar as multiplicações em ponto flutuante na matriz binária SEF;
   - Computar as adições para calcular as components da DHT.

5.2 Uma FHT de comprimento $N = 8$

Para $N = 8$, iniciamos por reunir os argumentos nas classes $\{0, 4, 2, 6\}$, que não estão associados com multiplicações. Estes correspondem ao conjunto C_0. A matriz A com os argumentos dos termos na matriz da DHT é

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 2 & 4 & 6 & 0 & 2 & 4 & 6 \\ 0 & 3 & 6 & 1 & 4 & 7 & 2 & 5 \\ 0 & 4 & 0 & 4 & 0 & 4 & 0 & 4 \\ 0 & 5 & 2 & 7 & 4 & 1 & 6 & 3 \\ 0 & 6 & 4 & 2 & 0 & 6 & 4 & 2 \\ 0 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \end{pmatrix}.$$

Existem somente $N/4 = 2$ classes, que são $C_0 = (0, 4, 2, 6)$ e $C_1 = (1, 5)$. Neste caso particular, o maior indice é $(N/4) - 1 = 1$.

As operações envolvendo produtos pelos autovalores, associados aos elementos de C_0, não devem ser consideradas como multiplicações em ponto flutuante.

As matrizes de interesse no algoritmo são:

$$M_0 = B_0(A) - B_4(A) + B_2(A) - B_6(A),$$

e

$$M_1 = B_1(A) - B_5(A).$$

A matriz aditiva M_0 é

$$M_0 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \end{pmatrix},$$

que fornece $posto = 6$. Em SEF

$$rref(M_0) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

A matriz de pré-adição M_1, associada às multiplicações, é

$$M_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

cujo posto é 2 e sua SEF é

$$rref(M_1) = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{pmatrix}.$$

Em seguida, para avaliar a complexidade multiplicativa da FHT de comprimento 8, determinamos o $posto(M_m)$.

A matriz de pré-adição associada com os ramos multiplicativos do algoritmo é

$$rref(M_1) = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{pmatrix}.$$

A Figura 5.1 mostra um diagrama para implementação do algoritmo FHT. A complexidade multiplicativa é de duas multiplicações em ponto flutuante, que atinge o limite inferior de Heideman [53]. Da Equação 5.5, a expansão matricial da DHT é $[DHT] = M_0 + M_1 cas(1)$, ou seja,

$$[DHT] = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \end{pmatrix} cas(1).$$

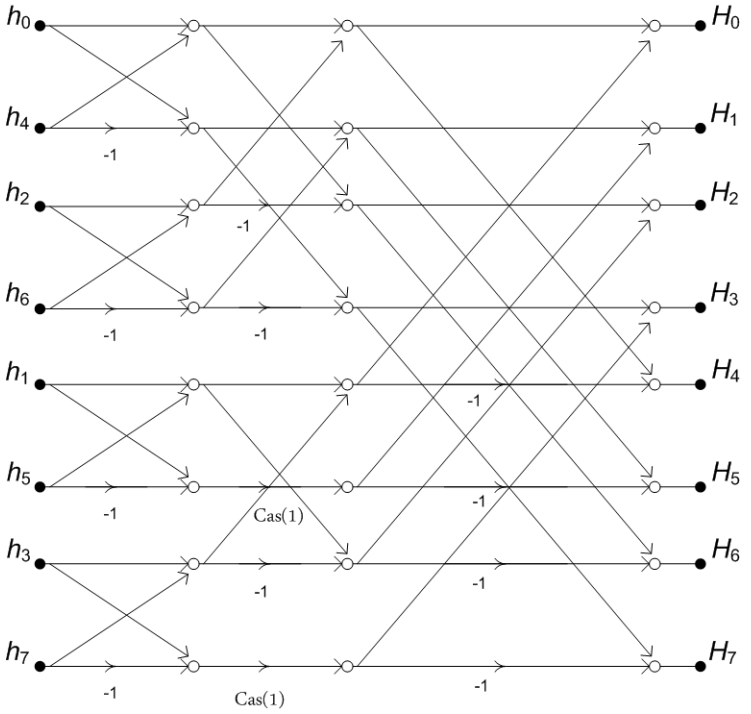


Figura 5.1: Esquema para a computação da FHT de comprimento $N = 8$.

O pseudo-código apresentado a seguir decreve os cálculos, mostrando as multiplicações em ponto flutuante. Além disso, podemos ver que a combinação linear de somadores envolve multiplicações apenas por 1 e -1, que são triviais.

Algoritmo 5.1: Computando as pré-adições e multiplicações em ponto flutuante.

```
1  para i = 0 até (N − 1) faça
2  |  VM(i) = 0 ;
3  fim para
4  para j = 0 até (N − 1) faça
5  |  para j = 0 até (N − 1) faça
6  |  |  se MRed(i,j) = 1 então
7  |  |  |  para j = j até (N − 1) faça
8  |  |  |  |  Tmp = Tmp + MRed(i, j) ;
9  |  |  |  fim para
10 |  |  |  VM(i) = Tmp ∗ Cas(m) ;
11 |  |  fim se
12 |  fim para
13 fim para
```

Algoritmo 5.2: Computando o vetor das pós-adições.

```
1  para i = 0 até (N − 1) faça
2  |  para j = 0 até (N − 1) faça
3  |  |  MC(j, i) = MC(j, i) + M(j, i) ∗ VM(i) ;
4  |  fim para
5  fim para
6  para i = 0 até (N − 1) faça
7  |  para j = 0 até (N − 1) faça
8  |  |  H(i) = H(i) + MC(i, j) ;
9  |  fim para
10 fim para
```

5.3 Uma FHT de comprimento $N = 16$

Para $N = 16$ existem 4 classes, a saber

$$
\begin{aligned}
C_2 &= \{2, 10\}, \\
C_0 &= \{0, 8, 4, 12\}, \\
C_1 &= \{1, 9, 3, 11\}, \\
C_{-1} &= \{15, 7, 5, 13\}.
\end{aligned}
$$

A matriz A com os argumentos dos termos na matriz DHT é

$$
A = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\
0 & 2 & 4 & 6 & 8 & 10 & 12 & 14 & 0 & 2 & 4 & 6 & 8 & 10 & 12 & 14 \\
0 & 3 & 6 & 9 & 12 & 15 & 2 & 5 & 8 & 11 & 14 & 1 & 4 & 7 & 10 & 13 \\
0 & 4 & 8 & 12 & 0 & 4 & 8 & 12 & 0 & 4 & 8 & 12 & 0 & 4 & 8 & 12 \\
0 & 5 & 10 & 15 & 4 & 9 & 14 & 3 & 8 & 13 & 2 & 7 & 12 & 1 & 6 & 11 \\
0 & 6 & 12 & 2 & 8 & 14 & 4 & 10 & 0 & 6 & 12 & 2 & 8 & 14 & 4 & 10 \\
0 & 7 & 14 & 5 & 12 & 3 & 10 & 1 & 8 & 15 & 6 & 13 & 4 & 11 & 2 & 9 \\
0 & 8 & 0 & 8 & 0 & 8 & 0 & 8 & 0 & 8 & 0 & 8 & 0 & 8 & 0 & 8 \\
0 & 9 & 2 & 11 & 4 & 13 & 6 & 15 & 8 & 1 & 10 & 3 & 12 & 5 & 14 & 7 \\
0 & 10 & 4 & 14 & 8 & 2 & 12 & 6 & 0 & 10 & 4 & 14 & 8 & 2 & 12 & 6 \\
0 & 11 & 6 & 1 & 12 & 7 & 2 & 13 & 8 & 3 & 14 & 9 & 4 & 15 & 10 & 5 \\
0 & 12 & 8 & 4 & 0 & 12 & 8 & 4 & 0 & 12 & 8 & 4 & 0 & 12 & 8 & 4 \\
0 & 13 & 10 & 7 & 4 & 1 & 14 & 11 & 8 & 5 & 2 & 15 & 12 & 9 & 6 & 3 \\
0 & 14 & 12 & 10 & 8 & 6 & 4 & 2 & 0 & 14 & 12 & 10 & 8 & 6 & 4 & 2 \\
0 & 15 & 14 & 13 & 12 & 11 & 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1
\end{pmatrix}.
$$

As matrizes relevantes são

1. $M_0 = B_0(A) - B_8(A) + B_4(A) - B_{12}(A)$,

$$M_0 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix},$$

que fornece o $posto(M_0) = 8$.

Na SEF, a matriz M_0 é

$$M_0 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

2. $M_1 = B_1(A) - B_9(A) + B_3(A) - B_{11}(A)$,

$$M_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix},$$

que fornece o $posto(M_1) = 4$.

Na SEF, a matriz M_1 é

$$M_1 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -0 & 0 & -0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}.$$

3. $M_{-1} = B_{15}(A) - B_7(A) + B_5(A) - B_{13}(A)$,

$$M_{-1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix},$$

que fornece o $posto(M_{-1}) = 4$.

Na SEF, a matriz M_{-1} é

$$M_{-1} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}.$$

4. $M_2 = B_2(A) - B_{10}(A)$,

$$M_2 = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{pmatrix},$$

que fornece o $posto(M_2) = 4$.

Na SEF, a matriz M_2 é

$$M_2 = \begin{pmatrix}
0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0
\end{pmatrix}.$$

A complexidade aditiva para a DHT de comprimento $N = 16$ é de 64 operações. Da Equação 5.6, a expansão matricial da DHT é

$$[DHT] = M_0 + M_1 cas(1) + M_{-1} cas(-1) + M_2 cas(2).$$

O esquema para a computação da DHT de comprimento 16 é mostrado na Figura 5.2

Podemos observar nesta Figura (destacado no retângulo) que as componentes de entrada $(v_0, v_2, v_4, v_6, v_8, v_{10}, v_{12}, v_{14})$ servem como entrada para uma DHT de comprimento 8, inerente à DHT de comprimento 16.

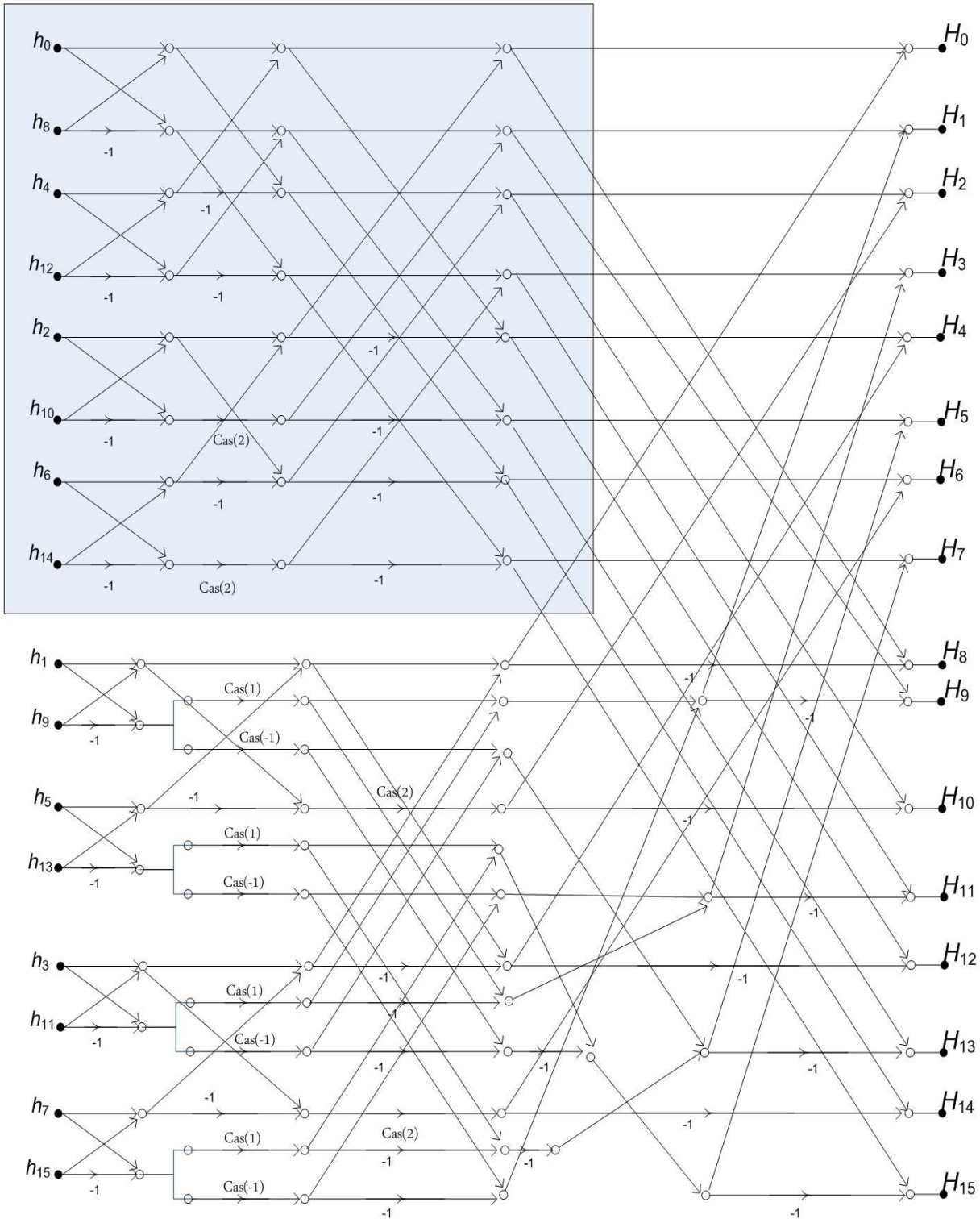


Figura 5.2: Esquema para a computação da FHT de comprimento 16*, destacando-se a existência da FHT de comprimento* 8 *no esquema.*

Tabela 5.3: Complexidade do algoritmo FHT baseado em expansão matricial, para uma sequência real, em termos do número de multiplicações em ponto flutuante, e dos algoritmos FHT base-2, base-4 e Split-Radix.

N	base-2	base-4	*Split-Radix*	Limite inferior de Heideman	Expansão Matricial FHT
8	4	-	2	2	2
16	20	14	12	10	12
32	68	-	42	32	40
64	196	142	124	84	96
128	516	-	330	198	256
256	1284	942	828	438	640
512	3076	-	1994	932	1408
1024	7172	5294	4668	1936	3328
2048	16388	-	10698	3962	7680
4096	36868	27310	24124	8034	16384

Os resultados de complexidade multiplicativa para a FHT baseada na expansão matricial são mostrados na Tabela 5.3, em comparação com o limite inferior de Heideman e os algoritmos padrões FHT base-2, base-4 e *Split-Radix* [84]. A Figura 5.3 mostra a complexidade multiplicativa dos algoritmos na Tabela 5.3 como uma função de N. Deve-se observar que as complexidades para os comprimentos $N = (8, 32, 128, 512, 2048)$, para o algoritmo base-4, não são reportadas em [84].

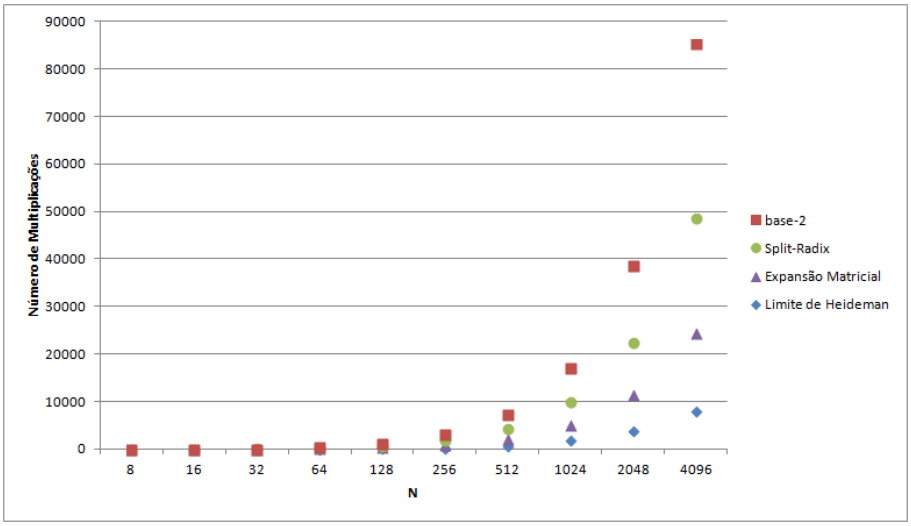


Figura 5.3: Complexidade multiplicativa para os algoritmos FHT (base-2, Split-Radix e expansão matricial) e limite inferior de Heideman, em função do comprimento N da transformada

Os resultados da complexidade aditiva para a FHT baseada na expansão matricial são mostrados na tabela 5.4, em comparação com os algoritmos padrão FHT base-2, base-4 e *Split-Radix* [84].

Tabela 5.4: Complexidade aditiva do algoritmo FHT baseado em expansão matricial e dos algoritmos FHT base-2, base-4 e Split-Radix.

N	base-2	base-4	*Split-Radix*	Expansão Matricial FHT
8	26	-	22	22
16	74	70	64	64
32	194	-	166	166
64	482	450	416	416
128	1154	-	998	998
256	2690	2498	2336	2336
512	6146	-	5350	5350

5.4 Otimizando a FHT de comprimento $N = 12$

O algoritmo proposto efetua a computação da DHT de comprimento 12 com 16 multiplicações. Nesta seção é feita uma otimização do mesmo para que esta DHT seja computada com o número mínimo de multiplicações, ou seja, 4.

Inicia-se por reunir os argumentos nas 3 classes, a saber

$$\begin{aligned} C_1 &= \{1, 7, 2, 8\}, \\ C_0 &= \{0, 6, 3, 9\}, \\ C_{-1} &= \{11, 5, 4, 10\}. \end{aligned}$$

em que C_0 não está associada com multiplicações, C_{-1} está associada ao $cas(-1) = cas(-\pi/6)$ e C_1 está associada ao $cas(1) = cas(\pi/6)$.

Proposição 5.4 *Se N é da forma $12i$ ou $\frac{12}{5}i$, então $D := cas(i) - cas(N - i) = 1$.*

Demonstração: $D := cas(i) - cas(-i) = \cos(i) + sen(i) - \cos(i) + sen(i)$,

$D = 2sen(i) = 2sen(\frac{2\pi}{N}i)$.

Para $N = 12i$, $D = 2sen(\frac{2\pi}{12i}i) = 2sen(\frac{\pi}{6}) = 1$;

Para $N = \frac{12}{5}i$, $D = 2sen(\frac{2\pi}{12i/5}i) = 2sen(\frac{5\pi}{6}) = 1$; ■

Definindo-se

$$\begin{aligned}
h_a &= h_1 + h_7, \\
h_b &= h_1 - h_7, \\
h_c &= h_2 + h_8, \\
h_d &= h_2 - h_8, \\
h_e &= h_4 + h_{10}, \\
h_f &= h_4 - h_{10}, \\
h_g &= h_5 + h_{11}, \\
h_h &= h_5 - h_{11},
\end{aligned}$$

as componentes da transformada em que ocorrem as multiplicações são

$$\begin{aligned}
H_1 &= (h_b + h_d)cas(1) + (h_f - h_h)cas(-1) \text{ e} \\
H_5 &= -(h_f - h_h)cas(1) - (h_b - h_d)cas(-1); \\
H_2 &= (h_a - h_e)cas(1) + (h_c - h_g)cas(-1) \text{ e} \\
H_{10} &= -(h_c - h_g)cas(1) - (h_a - h_e)cas(-1); \\
H_4 &= -(h_c + h_g)cas(1) + (h_a + h_e)cas(-1) \text{ e} \\
H_8 &= -(h_a + h_e)cas(1) + (h_c + h_g)cas(-1); \\
H_7 &= (h_d - h_b)cas(1) + (h_f + h_h)cas(-1) \text{ e} \\
H_{11} &= -(h_f + h_h)cas(1) - (h_d - h_b)cas(-1).
\end{aligned}$$

Aplicando a relação $cas(1) - cas(-1) = 1$, estas expressões, obtida da Proposição 5.4 para $N = 12$, podem ser reescrita como

$$
\begin{aligned}
H_1 &= [(h_b + h_d) + (h_f - h_h)]cas(1) + (h_f - h_h) \text{ e} \\
H_5 &= -[(h_b + h_d) + (h_f - h_h)]cas(1) - (h_b + h_d); \\
H_2 &= [(h_a - h_e) + (h_c - h_g)]cas(1) + (h_c - h_g) \text{ e} \\
H_{10} &= -[(h_a - h_e) + (h_c - h_g)]cas(1) - (h_a - h_e); \\
H_4 &= [(h_a + h_e) - (h_c + h_g)]cas(1) + (h_a + h_e) \text{ e} \\
H_8 &= -[(h_a + h_e) - (h_c + h_g)]cas(1) + (h_c + h_g); \\
H_7 &= [(h_d - h_b) + (h_f + h_h)]cas(1) + (h_f + h_h) \text{ e} \\
H_{11} &= -[(h_d - h_b) + (h_f + h_h)]cas(1) - (h_d - h_b).
\end{aligned}
$$

Observa-se que existem apenas 4 multiplicações distintas em ponto flutuante, todas por $cas(1) = cas(\pi/6)$, valor que atinge o limite inferior de complexidade multiplicativa de Heideman. A Figura 5.4 apresenta um esquema para implementar este o algoritmo ótimo. O diagrama foi construído em termos da transformada de Walsh-Hadamard de comprimento $N = 2$ [90], Figura 5.5.

5.5 Otimizando a FHT de comprimento $N = 16$

O produto dos números complexos $(a + jb)$ e $(c + jd)$ requer quatro multiplicações reais, uma vez que se

$$e + jf := (a + jb)(c + jd)$$

então

$$e = ac - bd$$

e

$$f = ad + bc.$$

Em [29] é apresentado um algoritmo rápido para computar este produto, que requer 3 multiplicações. Especificamente,

$$e = (a - b)d + a(c - d) \tag{5.12}$$

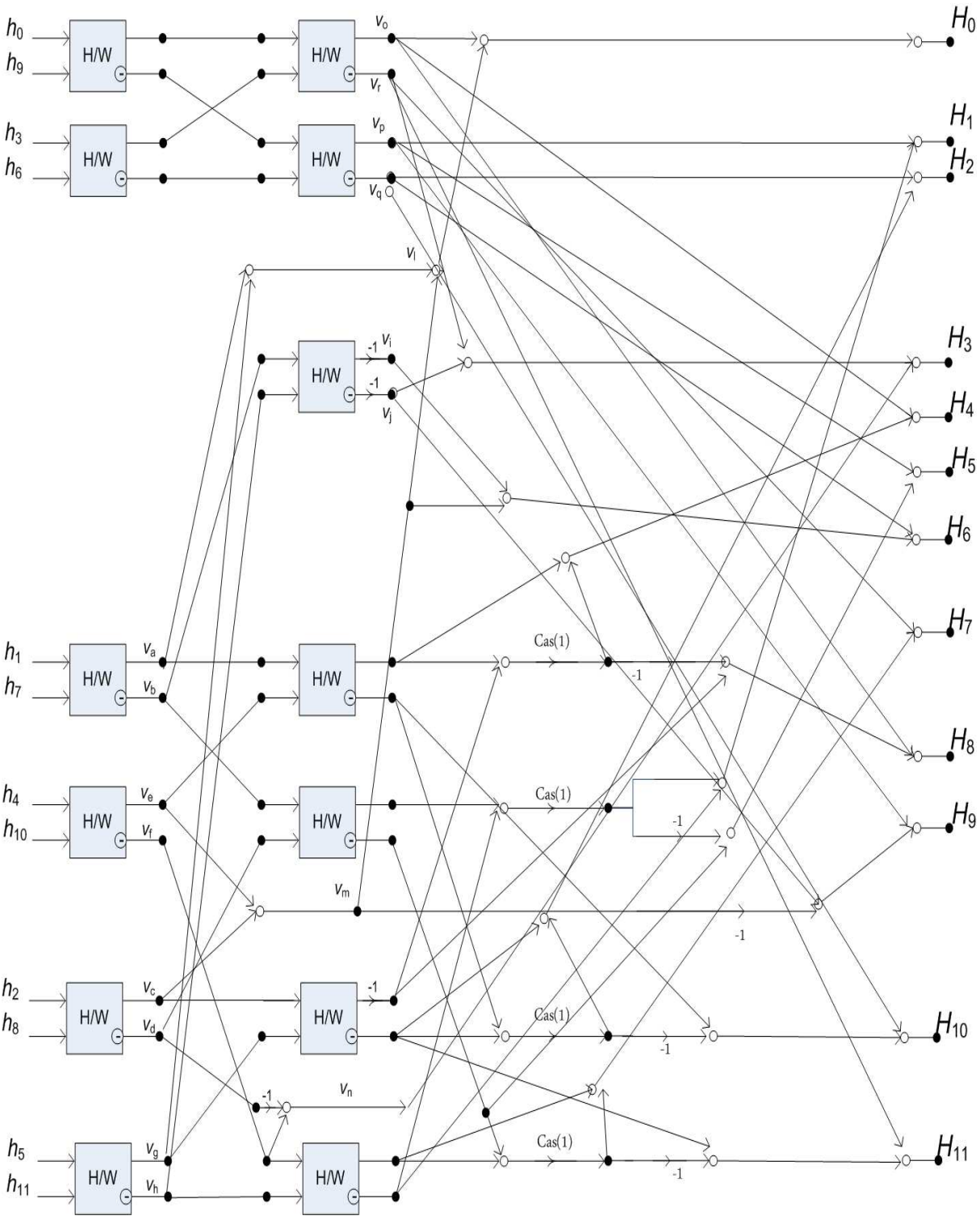


Figura 5.4: Algoritmo ótimo para a computação de uma DHT de comprimento $N = 12$.

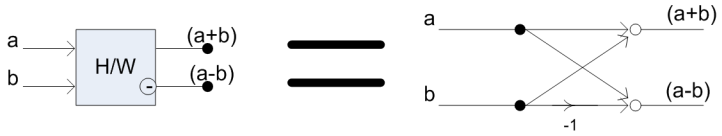


Figura 5.5: Diagrama para representação da transformada de Walsh-Hadamard para comprimento $N = 2$.

e

$$f = (a-b)d + b(c+d). \tag{5.13}$$

No que se segue, este algoritmo é usado para reduzir a complexidade multiplicativa da FHT obtida pela técnica de expansão matricial. O procedimento envolve apenas as componentes ímpares da sequência h de comprimento 16, cuja DHT se deseja computar, conforme indica a Figura 5.6. Definindo-se

$$\begin{aligned} h_a &= h_1 - h_9, \\ h_b &= h_5 - h_{13}, \\ h_c &= h_3 - h_{11}, \\ h_d &= h_7 - h_{15}, \end{aligned}$$

e considerando a Figura 5.6, pode-se obter as seguintes expressões:

$$\begin{aligned} A &= h_a(cas(1)) + h_c(cas(1)) = (h_a + h_c)cas(1) \\ B &= h_a(cas(-1)) + h_c(cas(-1)) = (h_a + h_c)cas(-1) \\ E &= h_a(cas(1)) - h_c(cas(1)) = (h_a - h_c)cas(1) \\ F &= h_a(cas(-1)) - h_c(cas(-1)) = (h_a - h_c)cas(-1) \end{aligned}$$

e

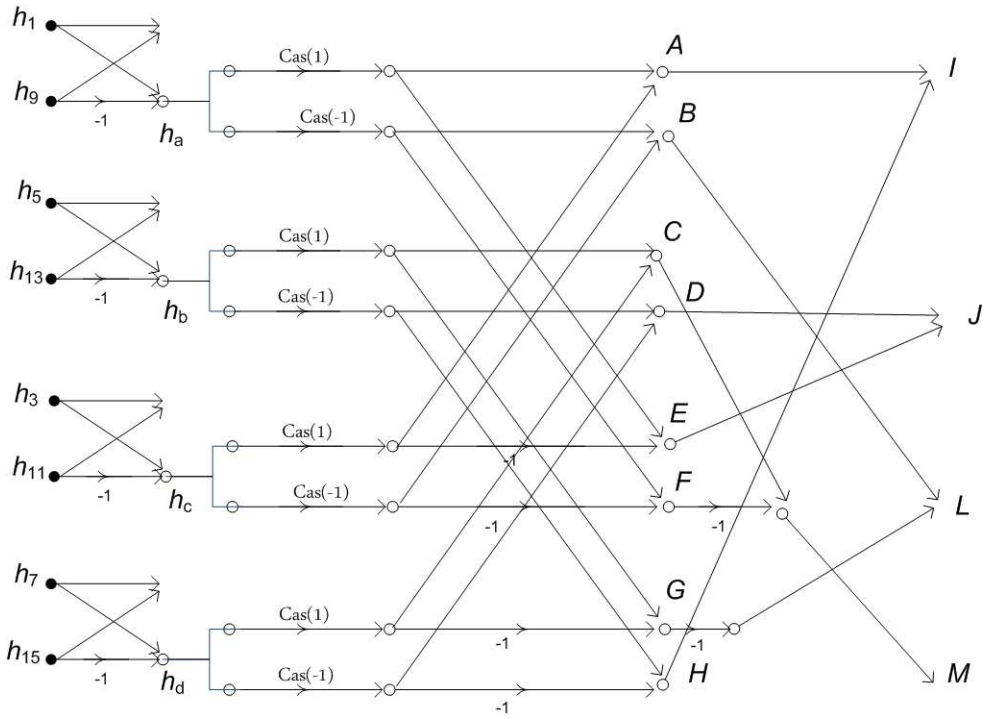


Figura 5.6: Multiplicações para computar a DHT de comprimento 16, envolvendo as componentes ímpares da sequência h a ser transformada.

$$
\begin{aligned}
C &= h_b(cas(1)) + h_d(cas(1)) = (h_b + h_d)cas(1);\\
D &= h_b(cas(-1)) + h_d(cas(-1)) = (h_b + h_d)cas(-1);\\
G &= h_b(cas(1)) - h_d(cas(1)) = (h_b - h_d)cas(1);\\
H &= h_b(cas(-1)) - h_d(cas(-1)) = (h_b - h_d)cas(-1).
\end{aligned}
$$

Fazendo a combinação de (A, H), (D, E), (B, G) e (C, F) temos

$$
\begin{aligned}
I &= A + H = (h_a + h_c)cas(1) + (h_b - h_d)cas(-1);\\
J &= D + E = (h_a - h_c)cas(1) + (h_b + h_d)cas(-1);\\
L &= B - G = (h_a + h_c)cas(-1) - (h_b - h_d)cas(1);\\
M &= C - F = -(h_a - h_c)cas(-1) + (h_b + h_d)cas(1).
\end{aligned}
$$

Pode-se notar que os valores de I e L tem a mesma forma que a multiplicação de números complexos; com isso pode-se aplicar as Equações 5.12 e 5.13 para reduzir o número de

multiplicações. As novas expressões de I e L são

$$\begin{aligned} I &= (h_a + h_c)cas(1) + (h_b - h_d)cas(-1) \\ &= [cas(1) + cas(-1)](h_b - h_d) + [(h_a + h_b) + (h_b - h_d)]cas(1), \\ L &= (h_a + h_c)cas(-1) - (h_b - h_d)cas(1) \\ &= [cas(1) + cas(-1)](h_b - h_d) + [(h_a + h_b) - (h_b - h_d)]cas(-1). \end{aligned}$$

Com isso, uma multiplicação deixou de ser efetuada; além disso, nenhuma adição será acrescentada, pois $cas(1)+cas(-1) = 2cos(1)$. Aplicando a mesma operação para J e M, obtém-se

$$\begin{aligned} J &= (h_b + h_d)cas(-1) + (h_a - h_c)cas(1) \\ &= [cas(1) + cas(-1)](h_a - h_c) + [(h_b + h_d) + (h_a - h_c)]cas(1), \\ M &= (h_b + h_d)cas(1) - (h_a - h_c)cas(-1) \\ &= [cas(1) + cas(-1)](h_a - h_c) + [(h_b + h_d) - (h_a - h_c)]cas(-1). \end{aligned}$$

O algoritmo proposto anteriormente efetuou a computação da DHT com 12 multiplicações. Com a redução de duas multiplicações, como mostrado nas expressões de I, J, L e M, obtém-se um algoritmo ótimo, pois o mesmo efetua a computação da DHT de comprimento 16 com um número mínimo de multiplicações [53].

A Figura 5.7 mostra um esquema para a implementação deste algoritmo.

5.6 Otimizando a FHT de comprimento $N = 24$

Inicia-se por reunir os argumentos na classe $C_0 = \{0, 12, 6, 18\}$, que não estão associados com multiplicações. Além desta, tem-se as classes $C_{-1} = \{23, 11, 7, 19\}$, associada ao $cas(-1) = cas(-\pi/12)$, $C_1 = \{1, 13, 5, 17\}$ associada ao $cas(1) = cas(\pi/12)$, $C_{-2} = \{22, 10, 8, 20\}$, associada ao $cas(-2) = cas(-\pi/6)$, $C_2 = \{2, 14, 4, 16\}$ associada ao $cas(2) = cas(\pi/6)$ e $C_3 = \{3, 15\}$ associada ao $cas(3) = cas(\pi/4)$.

Assim como na DHT de comprimento $N = 12$, tem-se os ângulos $\pi/6$ e $-\pi/6$ formando a relação $cas(\pi/6) - cas(-\pi/6) = 1$. Em relação às classes sugeridas pode-se exprimir como $cas(2) - cas(-2) = 1$. Definindo-se

$$
\begin{aligned}
h_a = h_1 + h_{13} \quad &, \quad h_b = h_1 - h_{13}, \\
h_c = h_2 + h_{14} \quad &, \quad h_d = h_2 - h_{14}, \\
h_e = h_3 + h_{15} \quad &, \quad h_f = h_3 - h_{15}, \\
h_g = h_4 + h_{16} \quad &, \quad h_h = h_4 - h_{16}, \\
h_i = h_5 + h_{17} \quad &, \quad h_j = h_5 - h_{17}, \\
h_l = h_6 + h_{18} \quad &, \quad h_m = h_6 - h_{18}, \\
h_n = h_7 + h_{19} \quad &, \quad h_o = h_7 - h_{19}, \\
h_p = h_8 + h_{20} \quad &, \quad h_q = h_8 - h_{20}, \\
h_r = h_9 + h_{21} \quad &, \quad h_s = h_9 - h_{21}, \\
h_t = h_{10} + h_{22} \quad &, \quad h_u = h_{10} - h_{22}, \\
h_v = h_{11} + h_{23} \quad &, \quad h_x = h_{11} - h_{23}.
\end{aligned}
$$

As componentes da transformada em que ocorrem as multiplicações por $cas(1)$, $cas(-1)$ e $cas(3)$ são

$$
\begin{aligned}
H_1 &= (h_b + h_j)cas(1) + (h_o - h_x)cas(-1) + (h_f)cas(3) \\
H_3 &= ((h_b - h_j) + h_r)cas(3) \\
H_7 &= (h_o + h_x)cas(1) + (h_b - h_j)cas(-1) - (h_r)cas(3) \\
H_9 &= (h_f - (h_o - h_x))cas(3)
\end{aligned}
$$

Sabendo que $cas(-1) = \frac{cas(3)}{2}$, tem-se

$$
\begin{aligned}
H_1 &= (h_b + h_j)cas(1) + ((\frac{h_o - h_x}{2}) + h_f)cas(3) \\
H_3 &= ((h_b - h_j) + h_r)cas(3) \\
H_7 &= (h_o + h_x)cas(1) + ((\frac{h_b - h_j}{2}) - h_r)cas(3) \\
H_9 &= (h_f - (h_o - h_x))cas(3).
\end{aligned}
$$

As componentes da transformada em que ocorrem multiplicações por $cas(2)$ e $cas(-2)$ são

$$
\begin{aligned}
H_1 &= (h_d + h_h)cas(2) - (-h_q + h_u)cas(-2) \text{ e} \\
H_5 &= (-h_q + h_u)cas(2) - (h_d + h_h)cas(-2), \\
H_2 &= (h_a + h_c - h_n - h_p)cas(2) - (-h_g + h_i + h_t - h_v)cas(-2) \text{ e} \\
H_{10} &= (-h_g + h_i + h_t - h_v)cas(2) - (h_a + h_c - h_n - h_p)cas(-2), \\
H_4 &= (h_a - h_g + h_n - h_t)cas(2) - (-h_c + h_i - h_p + h_v)cas(-2) \text{ e} \\
H_{20} &= (-h_c + h_i - h_p + h_v)cas(2) - (h_a - h_g + h_n - h_t)cas(-2), \\
H_7 &= (-h_d + h_h)cas(2) - (-h_q - h_u)cas(-2) \text{ e} \\
H_{11} &= (-h_q - h_u)cas(2) - (-h_d + h_h)cas(-2), \\
H_8 &= (-h_c - h_i - h_p - h_v)cas(2) - (-h_a - h_g - h_n - h_t)cas(-2) \text{ e} \\
H_{16} &= (-h_a - h_g - h_n - h_t)cas(2) - (-h_c - h_i - h_p - h_v)cas(-2), \\
H_{14} &= (-h_a + h_c + h_n - h_p)cas(2) - (-h_g - h_i + h_t + h_v)cas(-2) \text{ e} \\
H_{22} &= (-h_g - h_i + h_t + h_v)cas(2) - (-h_a + h_c + h_n - h_p)cas(-2).
\end{aligned}
$$

Aplicando a relação $cas(2) - cas(-2) = 1$, chega-se a

$$
\begin{aligned}
H_1 &= ((h_d + h_h) - (-h_q + h_u))cas(2) - (-h_q + h_u) \text{ e} \\
H_5 &= ((-h_q + h_u) - (h_d + h_h))cas(2) - (h_d + h_h), \\
H_2 &= ((h_a + h_c - h_n - h_p) - (-h_g + h_i + h_t - h_v))cas(2) - (-h_g + h_i + h_t - h_v) \text{ e} \\
H_{10} &= ((-h_g + h_i + h_t - h_v) - (h_a + h_c - h_n - h_p))cas(2) - (h_a + h_c - h_n - h_p), \\
H_4 &= ((h_a - h_g + h_n - h_t) - (-h_c + h_i - h_p + h_v))cas(2) - (-h_c + h_i - h_p + h_v) \text{ e} \\
H_{20} &= ((-h_c + h_i - h_p + h_v) - (h_a - h_g + h_n - h_t))cas(2) - (h_a - h_g + h_n - h_t), \\
H_7 &= ((-h_d + h_h) - (-h_q - h_u))cas(2) - (-h_q - h_u) \text{ e} \\
H_{11} &= ((-h_q - h_u) - (-h_d + h_h))cas(2) - (-h_d + h_h), \\
H_8 &= ((-h_c - h_i - h_p - h_v) - (-h_a - h_g - h_n - h_t))cas(2) - (-h_a - h_g - h_n - h_t) \text{ e} \\
H_{16} &= ((-h_a - h_g - h_n - h_t) - (-h_c - h_i - h_p - h_v))cas(2) - (-h_c - h_i - h_p - h_v), \\
H_{14} &= ((-h_a + h_c + h_n - h_p) - (-h_g - h_i + h_t + h_v))cas(2) - (-h_g - h_i + h_t + h_v) \text{ e} \\
H_{22} &= ((-h_g - h_i + h_t + h_v) - (-h_a + h_c + h_n - h_p))cas(2) - (-h_a + h_c + h_n - h_p).
\end{aligned}
$$

Observa-se que existem somente 6 multiplicações em ponto flutuante, todas por $cas(2) = cas(\pi/6)$. No total foram 12 multiplicações em ponto flutuante, atingindo o limite de Heide-

man. A Figura 5.8, apresenta um esquema para o algoritmo ótimo, enquanto que na expansão matricial existem 32 multiplicações em ponto flutuante.

5.7 Conclusões

Um novo algoritmo rápido para a transformada discreta de Hartley de comprimento $N \equiv 0(\text{mod } 4)$ foi proposto, o qual é baseado em uma nova técnica para construção de uma expansão matricial da matriz de transformação. O procedimento faz uso das simetrias da expansão matricial e da função $cas(.)$ para reduzir a complexidade aritmética para computar o espectro discreto de Hartley. Exemplos detalhados para ilustrar a técnica foram apresentados para $N = 8$ e 16, mas o procedimento é sistemático. A transformada rápida de Hartley apresentada aqui é também simples de implementar usando um DSP ou circuitos integrados de alta velocidade e baixo custo. O algoritmo apresenta um melhor desempenho, em termos de complexidade multiplicativa, que os algoritmos FHT padrão Cooley-Tukey base-2, base-4 e *Split-Radix*; além disso, tem o mesmo desempenho, em termos de complexidade aditiva, que o algoritmo FHT *Split-Radix*.

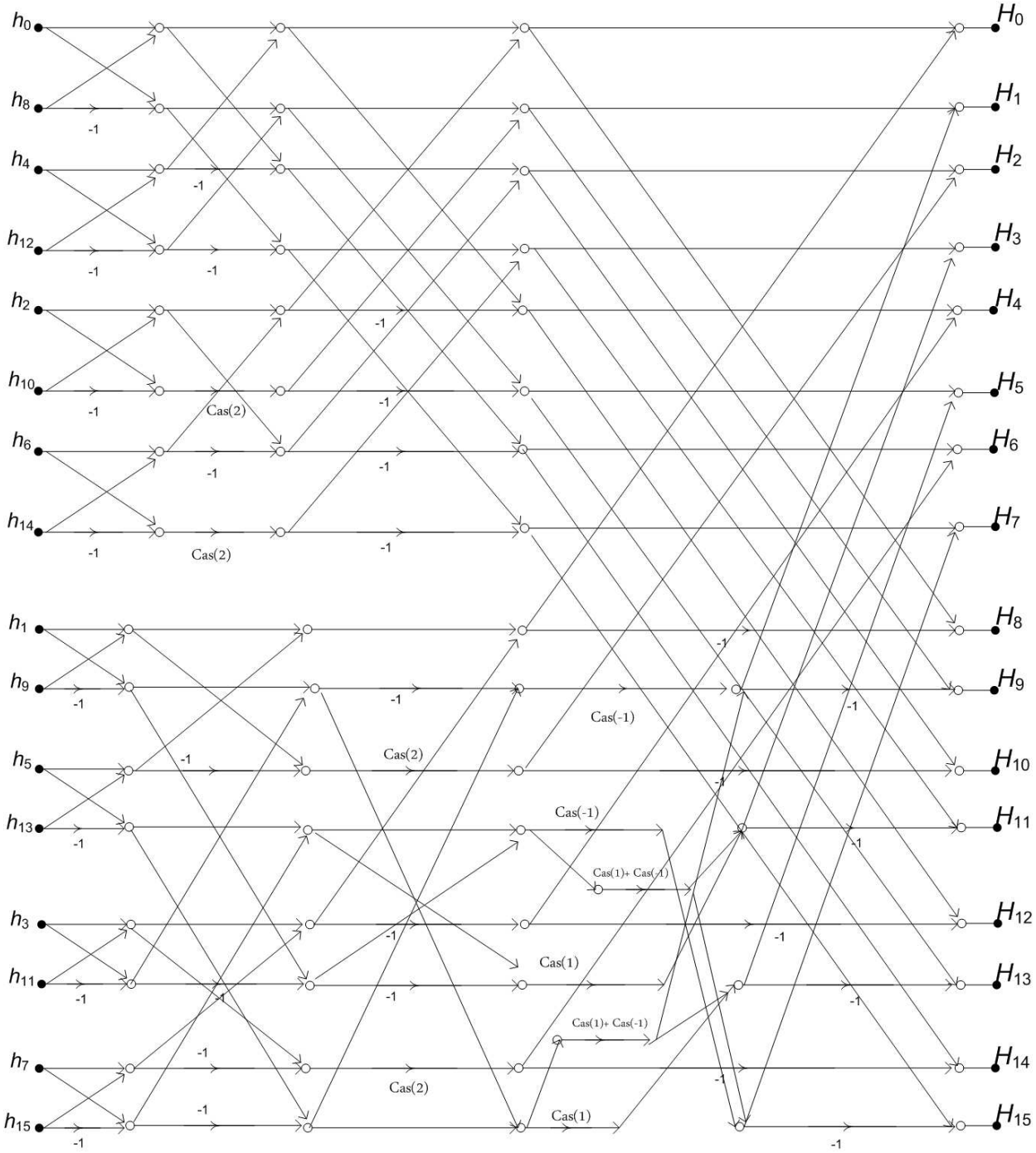


Figura 5.7: Algoritmo ótimo para a computação de uma DHT de comprimento $N = 16$.

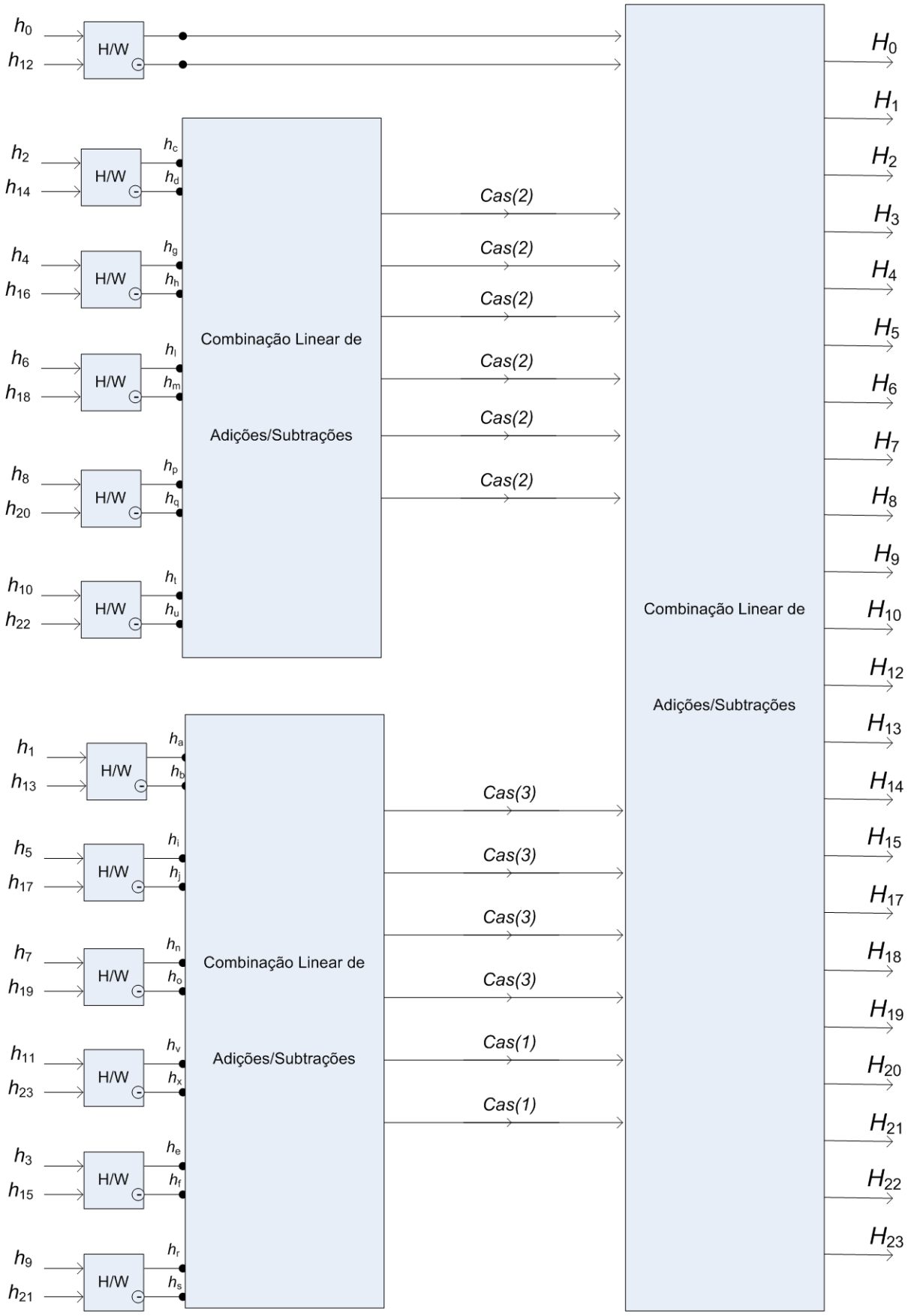


Figura 5.8: Algoritmo ótimo para a computação de uma DHT de comprimento $N = 24$.

capítulo 6

Uma Implementação Flexível das Transformadas de Fourier e Hartley de Comprimento 16 Baseada em Séries Matriciais de Laurent

6.1 Introdução

Este capítulo descreve uma arquitetura flexível para implementação de uma nova transformada rápida para computação das transformadas discretas de Fourier e de Hartley, que é baseada em uma série matricial de Laurent. O dispositivo calcula as transformadas baseado em uma único bit para seleção da operação. A estrutura e síntese do *hardware* são apresentados, o qual processa uma transformada rápida de comprimento 16 em 65 ns, com um dispositivo Xilinx SPARTAN 3E.

6.2 *Field Programmable Gate Arrays* (FPGA)

Em 1984, a Xilinx Inc. desenvolveu dispositivos lógicos programáveis capazes de serem configurados para reproduzir o comportamento de um hardware. Estes foram chamados de *Field Programmable Gate Arrays* (FPGA) [117, 118]. Estes dispositivos são formados por blocos lógicos programáveis que são conectados por interligações programáveis, o que permite a criação de circuitos lógicos, sendo limitados pela área e memória disponíveis. Sua flexibilidade é dada pela facilidade de configuração através de uma descrição de hardware escrita em VHDL ou Verilog. O FPGA é composto basicamente por três tipos de componentes:

- Bloco Lógico Cofigurável - CLB(do inglês *Configuration Logical Block*): são blocos construídos com flip-flops e lógica combinacional que permitem a construção de elementos lógicos funcionais.
- Bloco de Entrada/Saída - IOB(do inglês *Input/Output Block*): fazem a interface entre CLBs, funcionando como buffers de entrada e saída.
- Chaves Programáveis (do inglês *Switch Matrix*): representam a conexão entre os blocos lógicos. Permitem a conexão de CLBs e IOBs usando trilhas com conexão programáveis.

A Figura 6.1 mostra os blocos lógicos que estão disponíveis nos dispositivos FPGAs.

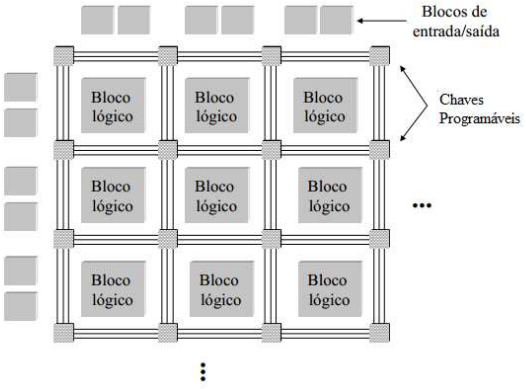


Figura 6.1: Arquitetura básica de um dispositivo FPGA.

6.3 O Algoritmo FFT/FHT

Das Equações 2.16 e 2.21, é aparente que a DHT é computada pela DFT conforme a Equação 6.1.

$$H_k = \Re e(V_k) - \Im m(V_k). \tag{6.1}$$

O algoritmo rápido é escrito de acordo com a seguinte decomposição matricial da DFT:

$$\Re eDFT = \left\{ \Re e(M_0) + \sum_{m=1}^{\frac{(N/4)-1}{2}} \Re e\,(M_m + M_{-m}) \cos(\frac{2\pi m}{N}) + \sum_{m=1}^{\frac{(N/4)-1}{2}} \Im m\,(M_m - M_{-m}) \sin(\frac{2\pi m}{N}) \right\}, \tag{6.2}$$

$$\Im mDFT = \left\{ \Im m(M_0) + \sum_{m=1}^{\frac{(N/4)-1}{2}} \Im m\,(M_m + M_{-m}) \cos(\frac{2\pi m}{N}) - \sum_{m=1}^{\frac{(N/4)-1}{2}} \Re e\,(M_m - M_{-m}) \sin(\frac{2\pi m}{N}) \right\}, \tag{6.3}$$

em que as matrizes M_m são dadas pela Equação 6.4.

$$M_m := \sum_{l \in C_m} (-j)^{\frac{4(l-m)}{N}} \chi_l(M). \tag{6.4}$$

O operador χ_l ($l = 0, 1, 2, \ldots, N-1$) atua em uma matriz $(N \times N)$, produzindo uma nova matriz binária $(N \times N)$ cujos elementos são $(\delta_{l,m})$ com $m := m_{k,n}$, em que δ é o símbolo de Kronecker. As matrizes $\Re e(M_0)$, $\Re e(M_m \pm M_{-m})$, $\Im m(M_0)$ e $\Im m(M_m \pm M_{-m})$ são então escritas na forma padrão.

A complexidade multiplicativa da transformada rápida é computada pela Equação 6.5.

$$\sum_{m=1}^{(\frac{N}{4}-1)/2} posto \begin{pmatrix} \Re e(M_m + M_{-m}) \\ \Im m(M_m + M_{-m}) \end{pmatrix} + posto \begin{pmatrix} \Re e(M_m - M_{-m}) \\ \Im m(M_m - M_{-m}) \end{pmatrix}. \tag{6.5}$$

Em todos os casos examinados até aqui, nenhuma redução do *posto* foi alcançada quando empilhamos as matrizes $\Re e(M_m \pm M_{-m})$ e $\Im m(M_m \pm M_{-m})$, e a complexidade multiplicativa da FFT foi sempre dada por 6.6.

$$\sum_{m=1}^{(\frac{N}{4}-1)/2} posto(\Re e(M_m + M_{-m})) + posto(\Im m(M_m + M_{-m})). \tag{6.6}$$

De 6.2 e 6.3, os components da DHT são computados pela Equação 6.1.

6.4 Metodologia de projeto e arquitetura

O projeto foi realizado por meio das seguintes fases:

- Especificação;

- Descrição VHDL;
- Simulação comportamental;
- Síntese.

6.4.1 Especificação

O projeto visa a construção de um dispositivo para computação rápida da DFT/DHT, para uma sequência de comprimento 16; todos os valores da sequência de entrada são representados por uma palavra de 16-bits. A computação é feita utilizando aritmética de ponto fixo (7 bits). Todo componente da sequência de saída é representado por uma palavra de 32-bits, com 16 bits para sua parte real e 16 bits para sua parte imaginária. Com um bit de seleção, o usuário escolhe qual das transformadas será computada. A Figura 6.2 ilustra como ficaram as entradas e saídas no dispositivo.

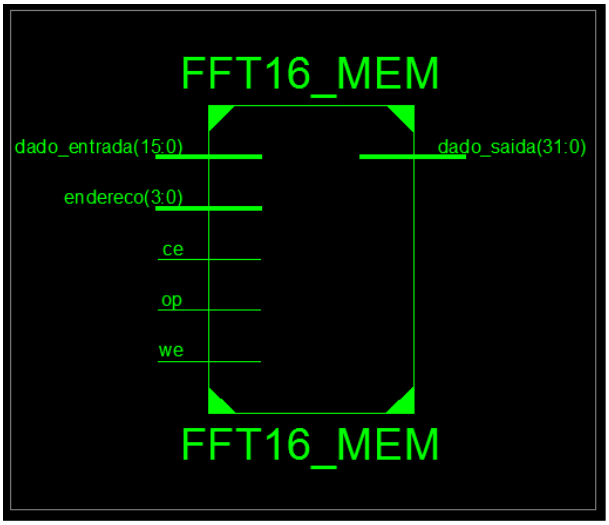


Figura 6.2: Visão geral da estrutura do dispositivo para computar a FFT/FHT.

6.4.2 Descrição VHDL

A descrição VHDL do dispositivo foi gerada com a ajuda do Simulink do Matlab^{TM}. A Figura 6.8 ilustra a implementação feita no Simulink^{TM}.

6.4.3 Simulação Comportamental

Com a ferramenta Xilinx ISE, a descrição VHDL foi compilada e simulada para verificar os dados de saída. Erros de sintaxe foram retirados neste momento. A Tabela 6.1 mostra a

quantidade de recursos utilizada pela implementação.

Tabela 6.1: Recursos utilizados no dispositivo FPGA para implementação da FFT/FHT de comprimento 16.

Descrição do Recurso	Quantidade
Number of Slices:	1611 *out of* 4656 (34%)
Number of Slice Flip Flops:	656 *out of* 9312 (7%)
Number of 4 input LUTs:	2894 *out of* 9312 (31%)
Number of bonded IOBs:	56 *out of* 232 (24%)
Number of MULT18X18SIOs:	12 *out of* 20 (60%)
Number of GCLKs:	2 *out of* 24 (8%)

A simulação comportamental foi realizada pela ferramenta Xilinx ISE e os arquivos de *testbench* foram gerados. Os estímulos (valores de entrada) foram inseridos e os resultados coletados de acordo com os diagramas da Fig. 6.3 (DHT) e Fig. 6.4 (DFT).

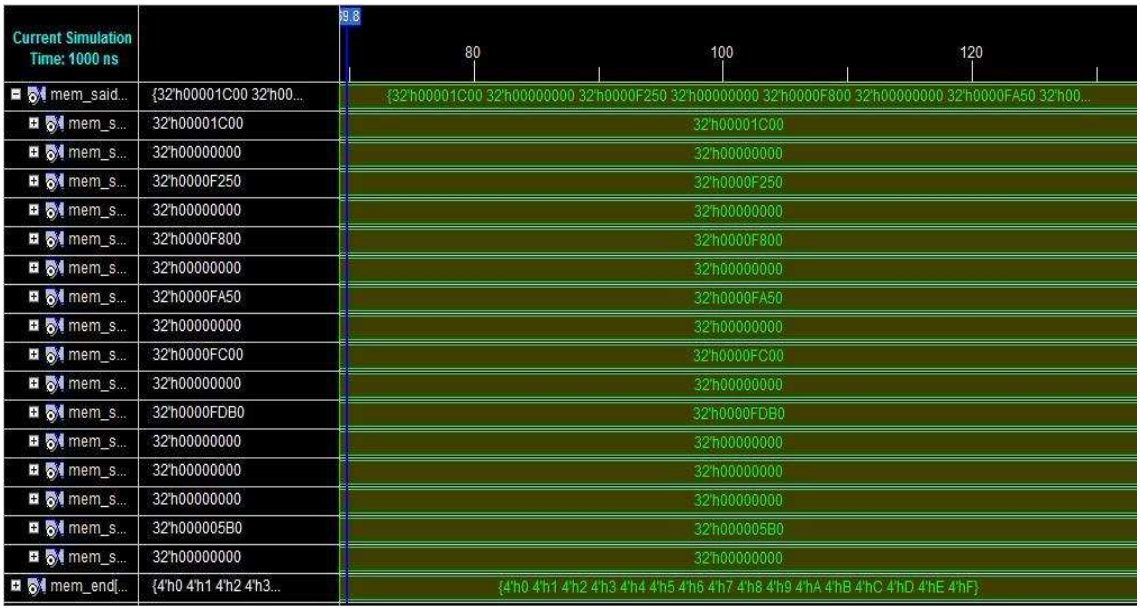


Figura 6.3: Resultado de simulação quando a operação está selecionada para computar a DHT.

6.4.4 Síntese

Nesta etapa, o código VHDL foi analisado e otimizado pela ferramenta de síntese, para em seguida criar-se uma implementação eficiente do dispositivo. Um esquema a Nível de Registrador de Transferencia (RTL - *Register Transfer Level*) foi gerado.

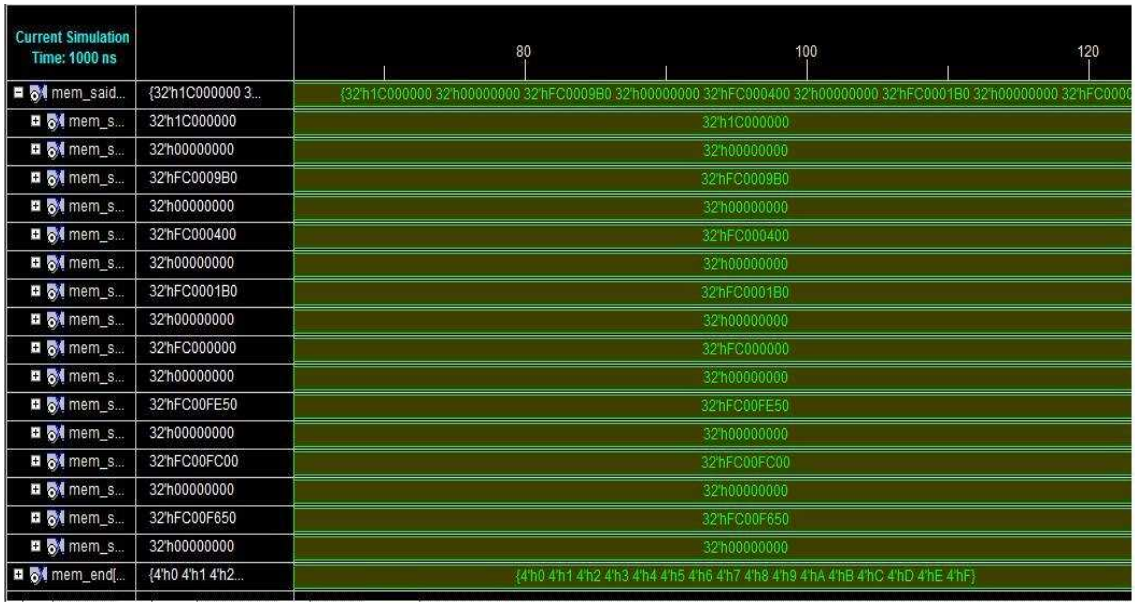


Figura 6.4: Resultado de simulação quando a operação está selecionada para computar a DFT.

A ferramenta de construção gerou os arquivos para *burn-in* no dispositivo FPGA escolhido, o dispositivo Spartan 3E, xc3s500e-5-fg320. Na simulação de pós-síntese o processo no circuito finalizou em 65 ns, e todos os resultados obtidos confirmaram a simulação comportamental.

6.4.5 Arquitetura

A arquitetura do dispositivo é baseada em dois blocos principais, o gerenciador de memória e o bloco núcleo, como mostrado na Figura 6.5. O bloco gerenciador de memória armazena as componentes do sinal de entrada e as componentes da transformada de saída.

O bloco núcleo é responsável pelo cáculo dos coeficientes da DFT, conforme Figura 6.6. Destes, o bloco gerenciador de memória seleciona e computa a transformada desejada.

Após o armazenamento na memória, o dispositivo calcula a transformada baseada no operador de seleção (DFT/DHT). Assim, a transformada é armazenada na memória, da seguinte maneira:

- Para a DFT: os primeiros dois bytes armazenam as componentes reais e os últimos dois armazenam as componentes imaginárias;
- Para a DHT: somente os últimos 16 bits amazenam as componentes da transformada.

O bloco núcleo foi gerado apartir da implementação em Simulink^{TM}. A complexidade aritmética para este algoritmo rápido de comprimento 16 é de 12 multiplicações em ponto

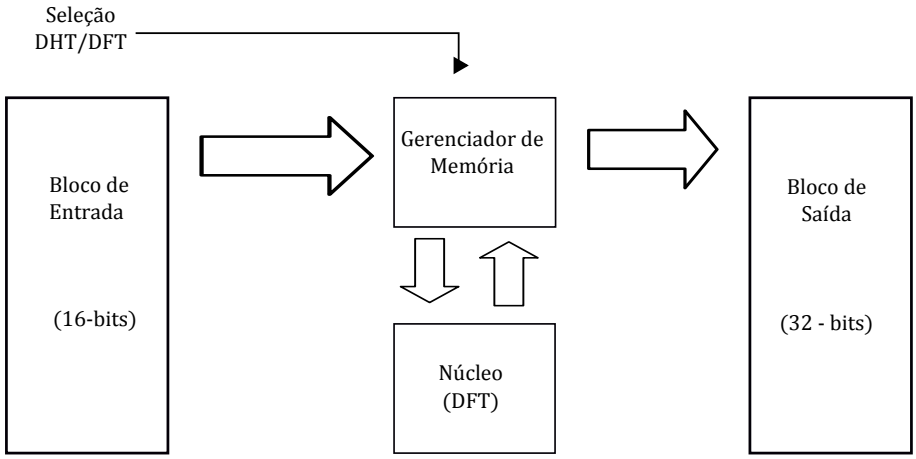


Figura 6.5: Arquitetura do algoritmo FFT/FHT. O bloco de computação aritmética corresponde ao bloco núcleo.

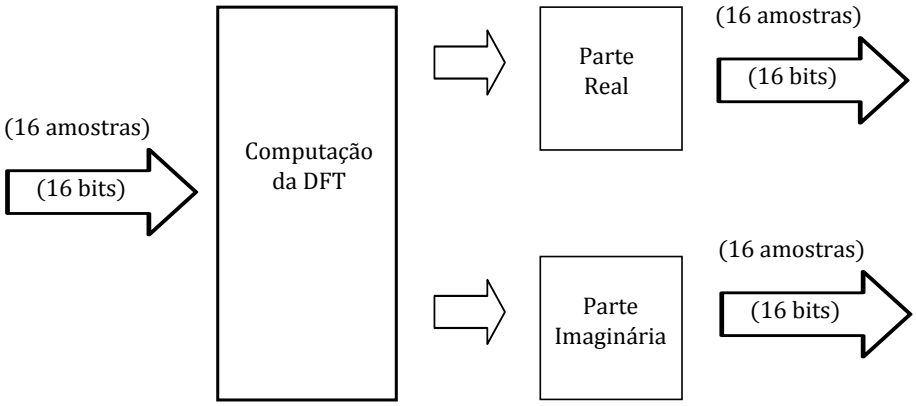


Figura 6.6: Arquitetura do bloco núcleo. Este bloco é usado por ambas as transformadas (Veja Figura 6.5).

fixo e 101 adições.

As sequências de entrada e saída, na Figura 6.7, são mostradas em formato hexadecimal ao se usar o bloco de conversão de tipos de dados. O bloco núcleo é mostrado no meio, que corresponde à Figura 6.6. Suas duas sequências de saída são mostradas também, ou seja, DFT (Real) e DFT (Imaginária). O bloco de adição (Add, Figura 6.7) fornece diretamente a DHT, que é mostrada também em formato hexadecimal.

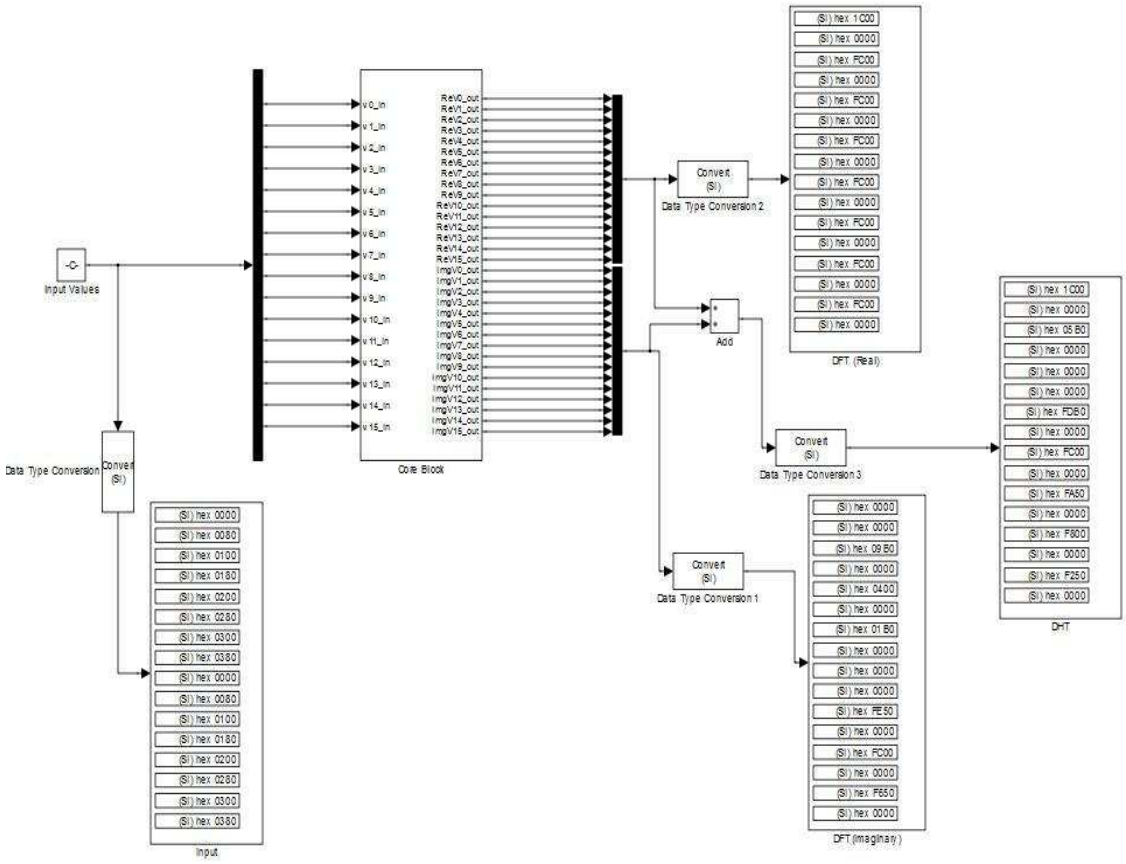


Figura 6.7: Simulação implementada em Simulink^{TM} do algoritmo FFT/FHT de comprimento 16.

6.5 Resultados de Simulação

A implementação foi simulada utilizando a ferramenta Xilinx ISE onde podemos verificar a precisão do dispositivo e os resultados foram comparados com os obtidos a partir do Simulink$^{\mathrm{TM}}$. Os resultados são mostrados nas Figuras 6.3, 6.4 e 6.7. A simulação foi feita com diversas sequências de entrada. Para a sequência particular $v = \{0\,1\,2\,3\,4\,5\,6\,7\,0\,1\,2\,3\,4\,5\,6\,7\}$, os resultados são mostrados na Tabela 6.2 e 6.3.

As sequências de saída da DFT realizada pela simulação, foram exatamente as mesmas obtidas pelo Simulink$^{\mathrm{TM}}$(Coluna 3, Tabela 6.3). Ambas tem uma pequena diferença em relação a sequência computada pela rotina interna do MatLab$^{\mathrm{TM}}$(Coluna 3, Tabela 6.2). O erro de quantização ($\leq 0,3\%$) é devido ao uso da aritmética de ponto fixo em vez da de ponto flutuante. No entanto, esse erro de magnitude é aceitável para muitas aplicações, incluindo áudio, sinais biomédicos e processamento da fala [29].

Tabela 6.2: Valores da DFT/DHT gerados via MatLab^{TM}.

Índice (k)	Dados de Entrada	Dados DFT	de Saída DHT
0	0	56.000	56.0000
1	1	0	0
2	2	-8.0000 +19.3137j	-27.3137
3	3	0	0
4	4	-8.0000 + 8.0000 j	-16.0000
5	5	0	0
6	6	-8.0000 + 3.3137j	-11.3137
7	7	0	0
8	0	-8.0000	-8.0000
9	1	0	0
10	2	-8.0000 - 3.3137j	-4.6863
11	3	0	0
12	4	-8.0000 - 8.0000j	0
13	5	0	0
14	6	-8.0000 -19.3137j	11.3137
15	7	0	0

6.6 Considerações Finais

Este capítulo apresentou o projeto e implementação, no dispositivo SPARTAN 3E xc3s500e-5-fg320, de um algoritmo que é capaz de fazer a computação rápida dos coeficientes da DFT/DHT de uma sequência real de comprimento $N = 16$. O dispositivo computa a DFT/DHT em 65 ns, que é aceitável para aplicações tais como áudio, processamento de sinais biomédicos e xDSL. Uma implementação do dispositivo em ponto flutuante está atualmente em investigação.

Tabela 6.3: Valores da simulação comportamental (Xilinx) da DFT/DHT de comprimento 16.

Índice (k)	Dados de Entrada	Dados DFT	de Saída DHT
0	0	56	56
1	1	0	0
2	2	-8.0000 +19.375j	-27.375
3	3	0	0
4	4	-8.0000 + 8.0000 j	-16.0000
5	5	0	0
6	6	-8.0000 + 3.375j	-11.375
7	7	0	0
8	0	-8.0000	-8.0000
9	1	0	0
10	2	-8.0000 - 3.375j	-4.625
11	3	0	0
12	4	-8.0000 - 8.0000j	0
13	5	0	0
14	6	-8.0000 -19.375j	11.375
15	7	0	0

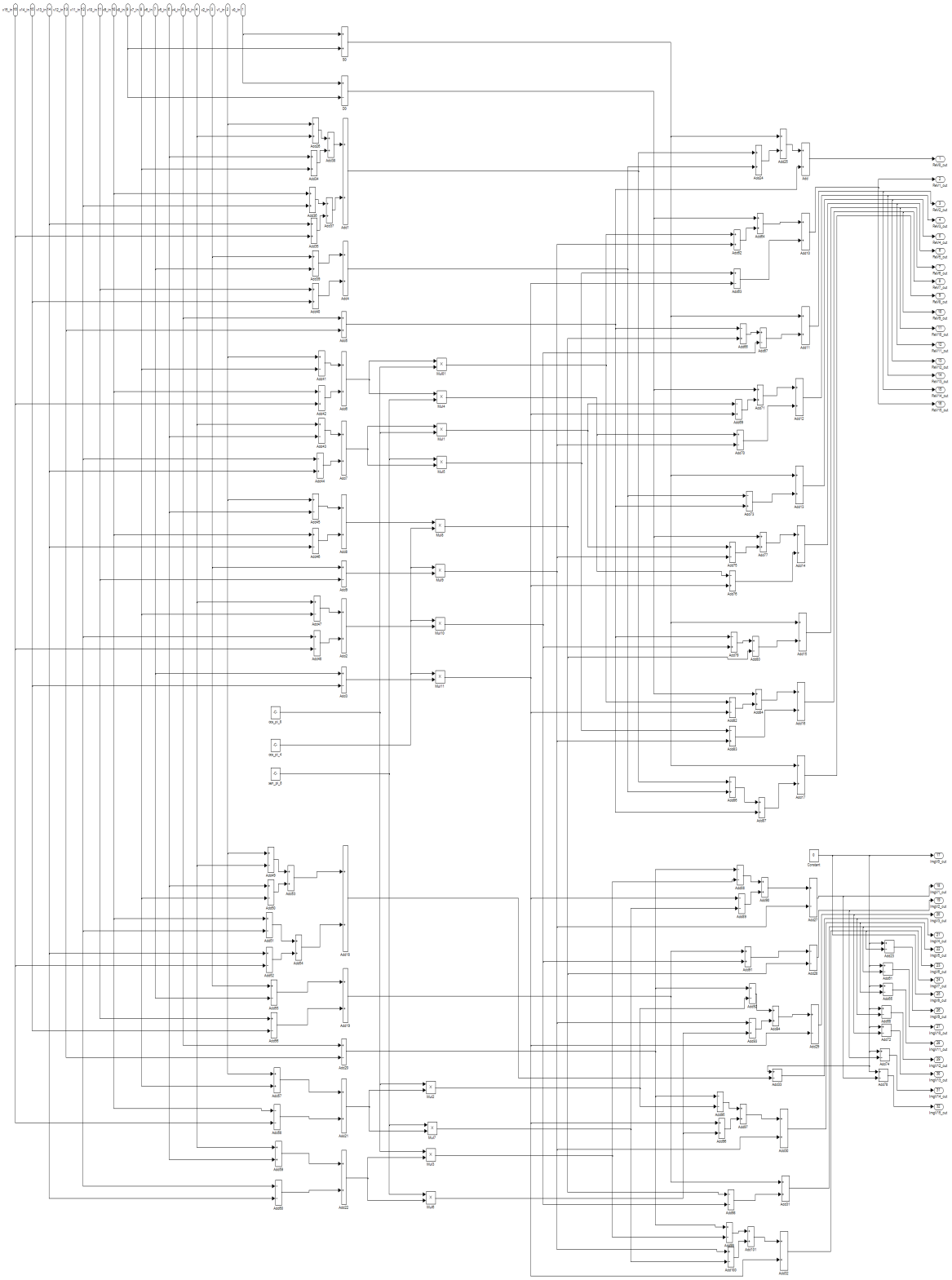


Figura 6.8: Implementação em Simulink^{TM} da FFT/FHT de comprimento 16.

capítulo 7

Conclusões

Esta Tese apresentou novos algoritmos rápidos para computação de transformadas discretas. O Capítulo 2 derivou a Transformada Discreta de Fourier, a partir da Série Discreta de Fourier e da Transformada de Fourier em Tempo Discreto e finalizou com a representação em forma matricial da mesma. Além disso, este capítulo apresentou a definição da transformada discreta de Hartley.

O Capítulo 3 apresentou a transformada rápida de Fourier, que são algoritmos rápidos para computar a DFT. Os algortimos de Cooley-Tukey, Good-Thomas e Radder-Primo foram detalhados.

O Capítulo 4 apresentou um novo algoritmo rápido para computar a DFT de comprimento $N \equiv 4 (\text{mod } 8)$, que é baseado nas simetrias das matrizes associadas com o desenvolvimento em série de Laurent da matriz de transformação; com isso, apresentou-se uma FFT para comprimentos que não sejam as costumeiras potências de dois. Um exemplo elucidativo foi apresentado para $N = 12$. A complexidade multiplicativa da FFT foi avaliada, sendo alcançados valores menores que $N \log_2 N$, para $N = 12, 20, 28, 36, 44, 52, 60$.

O Capítulo 5 apresentou um novo algoritmo rápido para a transformada discreta de Hartley de comprimento $N \equiv 0 (\text{mod } 4)$, que é baseado em uma nova técnica para construção de uma expansão matricial da matriz de transformação. O procedimento faz uso das simetrias da expansão matricial para reduzir a carga computacional para computar o espectro discreto de Hartley. O algoritmo apresentou um melhor desempenho, em termos de complexidade multiplicativa, que os algoritmos FHT padrões Cooley-Tukey base-2, base-4 e *Split-Radix*. Além disso, este capítulo apresentou algoritmos ótimos, em termos de complexidade multiplicativa,

para os comprimentos $8, 12, 16$ e 24.

O capítulo 6 apresentou o projeto e implementação, no dispositivo SPARTAN 3E xc3s500e-5-fg320, de um algoritmo que é capaz de fazer a computação rápida dos coeficientes da DFT/DHT de uma sequência real de comprimento $N = 16$. O dispositivo computa a DFT/DHT em 65 nsec, que é aceitável para aplicações tais como áudio, processamento de sinais biomédicos e xDSL.

7.1 A Transformada Rápida de Fourier

Os algoritmos rápidos (FFT) propostos na literatura para computar a transformada discreta de Fourier, de um modo geral, não atingem o limitante inferior de Heideman para a complexidade multiplicativa. A FFT proposta nesta tese, baseada em uma expansão em série matricial de Laurent, também apresenta esta característica. Entretanto, seu desempenho para os comprimentos N investigados, do tipo $N \equiv 4(\bmod 8)$, é superior ao das melhores FFT descritas na literatura, conforme indicado nas Tabelas de complexidade aritmética 7.1 e 7.1 (Tabelas 4.1 e 4.3, reapresentadas aqui por conveniência).

Tabela 7.1: Complexidade multiplicativa da FFT baseada na Série Matricial de Laurent

N	$N \log_2 N$ (arredondado)	Laurent-FFT	Ganho (%)
12	43	8	81 %
20	86	32	63 %
28	135	72	47 %
36	186	88	53 %
44	240	200	17 %
52	296	288	3 %
60	354	208	41 %

Tabela 7.2: Complexidade multiplicativa de algoritmos FFT em termos do número de multiplicações reais não triviais, para computar a DFT de uma sequência real de comprimento N ($N = 2^r$): Radix-2, Rader-Brenner, Laurent-FFT, limitante de Heideman.

N	$N \log_2 N$	Radix-2	Rader-Brenner	Heideman-Burrus	Laurent-FFT
8	24	4	4	2	2
16	64	12	10	10	12
32	160	88	34	16	54
64	384	264	196	84	224

Embora tenham sido apresentados exemplos específicos no capítulo 4, para $N = 12$ e 20, o procedimento é sistemático podendo ser implementado para qualquer valor de $N \equiv 4 (\text{mod } 8)$.

7.2 A transformada rápida de Hartley

Desde que foi proposta em 1984, a transformada discreta de Hartley vem sendo estudada para que sua computação seja a mais eficiente possível. As propostas de algoritmos rápidos, para computar a DHT, na maioria dos casos, são semelhantes aos algoritmos rápidos para computação da DFT, com desempenho similar. A FHT proposta nesta tese é sistemática e serve para comprimentos $N \equiv 0 \mod 4$, e não apenas para comprimentos que são potências de 2. Seu desempenho, em termos de complexidade multiplicativa, mostrou-se melhor que o dos tradicionais algoritmos padrão base-2, base-4 e split-radix, como se pode observar na Figura 7.1 (extraída dos dados apresentados nas Tabelas 5.3 e 5.4).

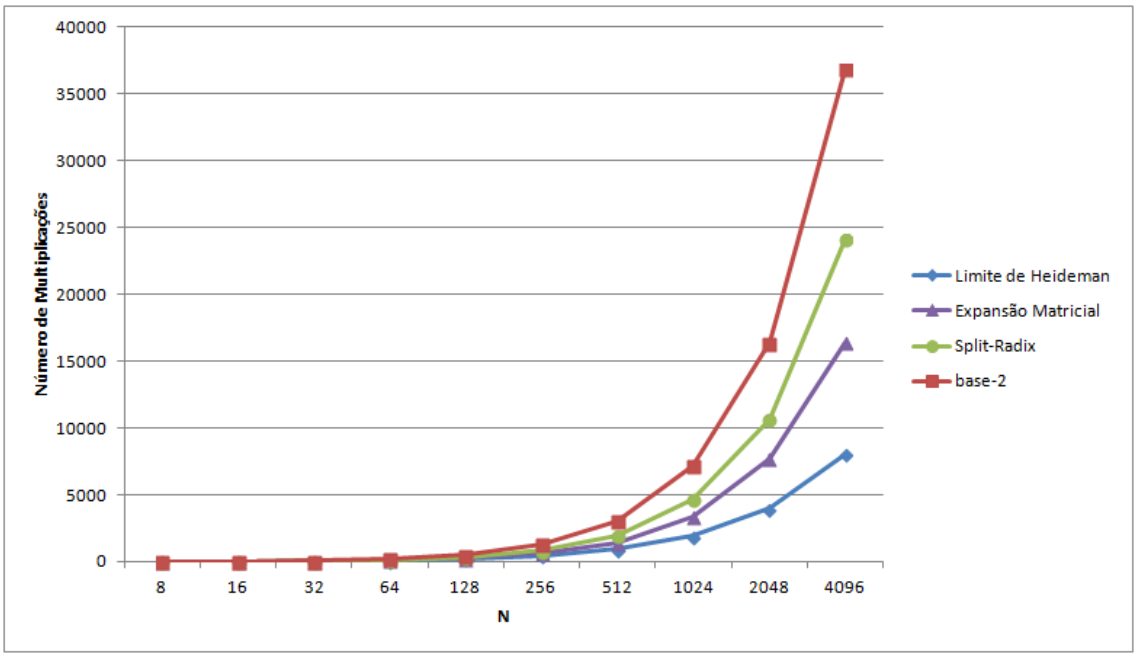


Figura 7.1: Complexidade multiplicativa para os algoritmos FHT (base-2, Split-Radix e expansão matricial) e limitante inferior de Heideman, em função do comprimento N da transformada.

Esta nova transformada rápida de Hartley pode ser melhorada para se chegar ao limitante mínimo de multiplicações necessárias para sua computação. Diante desse cenário, algoritmos ótimos para computação da DHT para os comprimentos $N = 8, 12, 16$ e 24 foram apresentados. Estes podem ser facilmente implementados em circuitos integrados, FPGA ou DSP. A

exploração de características da função $cas(.)$ e a técnica de expansão matricial desenvolvida nesta tese, permitiram o desenvolvimento dos mesmos.

7.3 Sugestões para Trabalhos Futuros

Como sugestões para trabalhos futuros relativos aos temas apresentados nesta Tese, podemos indicar:

- Busca de FHTs ótimas (aquelas que apresentam complexidade multiplicativa igual ao limitante de Heideman) para $N = 12k$, considerando as propriedades da funçao $cas(\frac{2\pi}{N}i)$ para esse tipo de comprimento;
- Implementação desses novos algoritmos rápidos, em um ambiente Linux, e comparação com outras rotinas para verificação do custo de memória e processador;
- Implementação desses novos algoritmos rápidos em uma plataforma paralela com as linguagem CUDA ou OpenCl, podendo ser utilizadas placas de Unidades de Processadores Gráficos (GPU, do inglês *Graphics Processor Unit*);
- Implementação, com as novas técnicas propostas, da transformada discreta de Fourier dos tipos II, III e IV [119];
- Implementação, com as novas técnicas propostas, da transformada discreta de Hartley dos tipos II, III e IV [120].
- Aplicação e investigação dessas novas técnicas em transformadas bidimensionais de Fourier e de Hartley;
- Aplicação e investigação dessas novas técnicas em transformadas bidimensionais de Fourier e de Hartley sobre corpos finitos;
- Projeto e Implementação em FPGA para os algoritmos padrões disponíveis e comparação, com os novos algoritmos propostos, em relação à área de uso do dispositivo.
- Investigação de um novo algoritmo rápido para computar a DCT e a DST, baseado em expansões matriciais semelhantes às apresentadas nesta Tese.

Referências

[1] J. Fourier, *Théorie analytique de la chaleur*. France: Académie des Sciences, 1822.

[2] J. P. Kahane, "Le retour de Fourier," *compte rendus Académie des Sciences*, pp. 1 – 9, 2005.

[3] P. Diniz, E. Silva, and S. Netto, *Digital Signal Processing: System Analysis and Design*. New York, USA: Cambridge University Press, 2010.

[4] J. W. Cooley, P. Lewis, and P. Welch, "Fast Fourier transform and its applications," *IEEE Trans on Education*, vol. 12, no. 1, pp. 28–34, 1969.

[5] J. G. Ables, "Fourier transform photography: a new method for x-ray astronomy," *Proceedings of the Astronomical Society of Australia, Vol. 1, p.172*, vol. 1, p. 172, 1968.

[6] B. Birkfellner, *Aplied Medical Image*. USA: CRC Press, 2011.

[7] F. Grandmont, L. Drissen, and G. Joncas, "Development of an imaging fourier transform spectrometer for astronomy," pp. 392–401, 2003. [Online]. Available: +http://dx.doi.org/10.1117/12.457339

[8] A.-P. Bernier, F. Grandmont, J.-F. Rochon, M. Charlebois, and L. Drissen, "First results and current development of spiomm: an imaging fourier transform spectrometer for astronomy," pp. 626 949–626 949–9, 2006. [Online]. Available: +http://dx.doi.org/10.1117/12.671410

[9] S. A. Fulop, *Speech Spectrum Analysis*. Berlin Heidelberg, Germany: Springer, 2011.

[10] M. Portnoff, "Time-scale modification of speech based on short-time fourier analysis," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 29, no. 3, pp. 374 – 390, jun 1981.

[11] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 32, no. 2, pp. 236 – 243, apr 1984.

[12] K. Paliwal, J. Lyons, and K. Wojcicki, "Preference for 20-40 ms window duration in speech analysis," in *Signal Processing and Communication Systems (ICSPCS), 2010 4th International Conference on*, dec. 2010, pp. 1 –4.

[13] A. V. Oppenheim, "Speech spectrograms using the fast Fourier transform," *Spectrum, IEEE*, vol. 7, no. 8, pp. 57 –62, aug. 1970.

[14] A. M. Kondoz, *Digital Speech: Coding for Low Bit Rate Communication Systems.* New Jersey, USA: John Wiley e Sons, Ltd., 2004.

[15] M. Arnold, "Audio watermarking: features, applications and algorithms," in *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, vol. 2, 2000, pp. 1013 –1016 vol.2.

[16] S. Pfeiffer, S. Fischer, and W. Effelsberg, "Automatic audio content analysis," in *Proceedings of the fourth ACM international conference on Multimedia*, ser. MULTIMEDIA '96. New York, NY, USA: ACM, 1996, pp. 21–30. [Online]. Available: http://doi.acm.org/10.1145/244130.244139

[17] A. Ramalingam and S. Krishnan, "Gaussian mixture modeling of short-time fourier transform features for audio fingerprinting," *Information Forensics and Security, IEEE Transactions on*, vol. 1, no. 4, pp. 457 –463, dec. 2006.

[18] R. C. Gonzalez and R. E. Woods, *Processamento Digital de Imagens.* BR: Pearson, 2010.

[19] I. Uzun, A. Amira, and A. Bouridane, "Fpga implementations of fast fourier transforms for real-time signal and image processing," *Vision, Image and Signal Processing, IEE Proceedings* -, vol. 152, no. 3, pp. 283 – 296, june 2005.

[20] B. Reddy and B. Chatterji, "An fft-based technique for translation, rotation, and scale-invariant image registration," *Image Processing, IEEE Transactions on*, vol. 5, no. 8, pp. 1266 –1271, aug 1996.

[21] I. Reed, T. Truong, Y. Kwoh, and E. Hall, "Image processing by transforms over a finite field," *Computers, IEEE Transactions on*, vol. C-26, no. 9, pp. 874 –881, sept. 1977.

[22] J. L. Massey, "The discrete Fourier transform in coding and cryptography," *IEEE Information Theory Workshop*, February 1998.

[23] R. M. Campello de Souza, E. S. V. Freire, and H. M. Oliveira, "Fourier codes," *Proceedings of the ISCTA´09*, vol. 1, pp. 370 – 375, August 2009.

[24] A. Poljicak, L. Mandic, and D. Agic, "discrete Fourier transform-based watermarking method with optimal implementation radius," *J. Eletron. Imaging*, vol. 20, 033008, August 2011.

[25] I. Kitamura, S. Kanai, and T. Kishinami, "Copyright protection of vector map using digital watermarking method based on discrete fourier transform," in *Geoscience and Remote Sensing Symposium, 2001. IGARSS '01. IEEE 2001 International*, vol. 3, 2001, pp. 1191 –1193 vol.3.

[26] T. A. Wysocki, B. Honary, and B. J. Wysocki, *Signal Processing for Telecommunications and Multimedia*. Boston, USA: Springer, 2005.

[27] R. N. Bracewell, "The Fourier transform," *Scientific American*, pp. 62–69, 1989.

[28] E. O. Brigham, *The fast Fourier transform and its applications*. New Jersey, USA: Prentice-Hall, 1988.

[29] R. E. Blahut, *Fast Algorithms for Signal Processing*, ser. Fast Algorithms for Signal Processing. Cambridge University Press, 2010.

[30] G. Goertzel, "An algorithm for the evaluation of finite trigonometric series," *American Mathematics Monthly*, vol. 65, pp. 34 – 35, 1958.

[31] I. J. Good, "The interaction algorithm and practical Fourier analysis," *Journal of the Royal Statistical Society*, vol. B20, pp. 361 – 372, 1958.

[32] L. H. Thomas, "Using a computer to solve problems in physics," *Applications of Digital Computers*, 1963.

[33] I. J. Good, "The relationship between two fast Fourier transforms," *IEEE Transactions on Computers*, vol. C-20, pp. 310 – 317, 1971.

[34] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.

[35] M. Heideman, D. Johnson, and C. Burrus, "Gauss and the history of the fast Fourier transform," *ASSP Magazine, IEEE*, vol. 1, pp. 14–21, 1984.

[36] J. Cooley, P. Lewis, and P. Welch, "Historical notes on the fast Fourier transform," *Proceedings of the IEEE*, vol. 55, pp. 1675 – 1677, 1967.

[37] B. Cipra, "The Best of the 20th Century: Editors Name Top 10 Algorithms," *SIAM News*, vol. 33, no. 4, May 2000.

[38] W. T. Cochran, J. W. Cooley, D. L. Favin, H. D. Helms, R. A. Kaenel, W. W. Lang, G. C. Maling, D. E. Nelson, C. M. Rader, and P. D. Welch, "What is the fast Fourier transform?" *Proceedings of The IEEE*, vol. 55, pp. 1664–1674, 1967.

[39] R. C. Singleton, "On computing the fast Fourier transform," *Communications of The ACM*, vol. 10, no. 10, pp. 647–654, 1967.

[40] C. Rader, "Discrete Fourier transforms when the number of data samples is prime," *Proceedings of the IEEE*, vol. 56, pp. 1107 – 1108, 1968.

[41] L. Bluestein, "A linear filtering approach to the computation of discrete Fourier transform," *IEEE Transactions on Audio and Electroacoustics*, vol. 18, pp. 451 – 455, 1970.

[42] G. Bergland, "A radix-eight fast Fourier transform subroutine for real-valued series," *Transactions on Audio and Electroacoustics*, vol. 17, pp. 138–144, 1969.

[43] D. Takahashi, "A radix-16 FFT algorithm suitable for multiply-add instruction based on goedecker method," *Proceedings. 2003 International Conference on Multimedia and Expo, 2003. ICME '03.*, vol. 2, pp. 845–848, 2003.

[44] R. C. Singleton, "An algorithm for computing the mixed radix fast Fourier transform," *Transactions on Audio and Electroacoustics*, vol. 17, pp. 93–103, 1969.

[45] J. M. Pollard, "The fast Fourier transform in a finite field," *Mathematics of Computation*, vol. 25, pp. 365 – 374, 1971.

[46] C. Rader and N. Brenner, "A new principle for fast Fourier transformation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 24, pp. 264–266, 1976.

[47] S. Winograd, "On computing the discrete Fourier transform," *Proc. Nat. Acad. Sci. Mathematics*, vol. 73, pp. 1005–1006, 1976.

[48] ——, "On computing the discrete Fourier transform," *Mathematics of Computation*, vol. 32, pp. 175 – 199, 1978.

[49] P. Duhamel and H. Hollman, "Split-radix FFT algorithm," *Electronics Letters*, vol. 20, pp. 14–16, 1984.

[50] H. Sorensen, C. Burrus, and D. Jones, "A new efficient algorithm for computing a few DFT points," *IEEE International Symposium on Circuits and Systems, 1988.*, vol. 2, pp. 1915–1918, 1988.

[51] D. Takahashi, "An extended split-radix FFT algorithm," *IEEE Signal Processing Letters*, vol. 8, pp. 145–147, 2001.

[52] S. G. Johnson and M. Frigo, "A modified split-radix FFT with fewer arithmetic operations," *IEEE Transactions on Signal Processing*, vol. 55, pp. 111–119, 2007.

[53] M. T. Heideman, *Multiplicative complexity, convolution, and the DFT*. New York, NY, USA: Springer-Verlag New York, Inc., 1988.

[54] D. W. Tufts and G. Sadasiv, "The arithmetic Fourier transform," *Acoust., Speech Signal Process. Mag.*, vol. 5, pp. 13 – 17, 1988.

[55] A. J. A. Paschoal, "A transformada aritmética de Fourier," Master's thesis, Programa de Pós-Graduação em Engenharia Elétrica, UFPE, 1993.

[56] R. J. S. Cintra and H. M. de Oliveira, "How to interpolate in arithmetic transform algorithms," *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, vol. IV, pp. 4169 – 4169, May 2002.

[57] A. Varkonyi-Koczy, "A recursive fast Fourier transformation algorithm," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 42, pp. 614–616, 1995.

[58] P. Marti-Puig, "Two families of radix-2 FFT algorithms with ordered input and output data," *Signal Processing Letters*, vol. 16, pp. 65–68, 2007.

[59] G. J. Silva Jr. and R. M. Campello de Souza, "Teoria da complexidade aditiva para transformadas," *Anais do XXIX Simpósio Brasileiro de Telecomunicações*, Outubro 2011.

[60] ——, "Transformada rápida de Fourier otimizada," *Anais do XXIX Simpósio Brasileiro de Telecomunicações*, Outubro 2011.

[61] G. J. da Silva Jr. and R. M. C. de Souza, "Minimum multiplicative complexity algorithm for computing a single component of the discrete fourier transform," *Digital Signal Processing*, vol. 23, no. 3, pp. 1040 – 1043, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1051200413000110

[62] C. Burrus, "An in-place, in-order prime factor FFT algorithm," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-29, pp. 806–817, 1981.

[63] V. H. Sorensen, D. L. Jones, M. T. Heideman, and C. S. Burrus, "Real-valued fast Fourier transform algorithms," *IEEE Transactions on acoustics, speech, and signal processing*, vol. ASSP-35, pp. 849–863, 1987.

[64] S. Bouguezel, M. O. Ahmad, and M. N. S. Swamy, "A new radix-2/8 FFT algorithm for length-$q \times 2^m$ DFTs," *IEEE Transactions on circuits and systems-I: Regular Papers*, vol. 51, pp. 1723–1732, 2004.

[65] P. Duhamel and M. Vetterli, "Fast Fourier transforms: a tutorial review and a state of the art," *Signal Process.*, vol. 19, pp. 259–299, April 1990.

[66] R. V. L. Hartley, "A more symmetrical fourier analysis applied to transmission problems," *Proceedings of the IRE*, vol. 30, no. 3, pp. 144–150, March.

[67] C. Paik, G. Cote, J. DePonte, and M. Fox, "Fast hartley transforms for spectral analysis of ultrasound doppler signals," *Biomedical Engineering, IEEE Transactions on*, vol. 35, no. 10, pp. 885 –888, oct. 1988.

[68] C. Paik and M. Fox, "Fast hartley transforms for image processing," *Medical Imaging, IEEE Transactions on*, vol. 7, no. 2, pp. 149 –153, jun 1988.

[69] ——, "Comparison of hartley and cas-cas transforms in medical image compression," in *Bioengineering Conference, 1989., Proceedings of the 1989 Fifteenth Annual Northeast*, mar 1989, pp. 127 –128.

[70] M. Fox, "Real time frequency domain processing of medical images," in *Bioengineering Conference, 1988., Proceedings of the 1988 Fourteenth Annual Northeast*, mar 1988, pp. 285 –286.

[71] H. Kuhl, M. D. Sacchi, and J. Fertig, "The Hartley transform in seismic imaging," *Society of Exploration Geophysicists - Geophysics*, vol. 66, pp. 1251 – 1257, 2001.

[72] L. Chen and D. Zhao, "Optical image encryption with Hartley transforms," *Optical society of America - Optical letters*, vol. 31, pp. 3438 – 3440, 2006.

[73] P. J. Paul and P. N. Girija, "A novel VLSI architecture for image compression," *Proceedings of the eighth IEEE International Symposium on Multimedia (ISM'06)*, 2006.

[74] J. Fabrega, M. Moreolo, and G. Junyent, "Constant envelope coherent optical ofdm based on fast hartley transform," in *Transparent Optical Networks (ICTON), 2011 13th International Conference on*, june 2011, pp. 1 –4.

[75] J. Villasenor, "Optical hartley transforms," *Proceedings of the IEEE*, vol. 82, no. 3, pp. 391 –399, mar 1994.

[76] R. Muralishankar, L. Kaushik, and A. Ramakrishnan, "Time-scaling of speech and music using independent subspace analysis," in *Signal Processing and Communications, 2004. SPCOM '04. 2004 International Conference on*, dec. 2004, pp. 310 – 314.

[77] E. Chilton, "The hartley transform applied to speech coding," in *Speech Coding, IEE Colloquium on*, oct 1989, pp. 9/1 –9/6.

[78] R. Chen, Z. E, M. Asharif, and K. Yamashita, "Adaptive filter by using hartley transform extended correlation lms algorithm in the double-talk condition," in *Innovative Computing Information and Control, 2008. ICICIC '08. 3rd International Conference on*, june 2008, p. 394.

[79] Z. Sembiring, M. Malek, and H. Rahim, "Low complexity ofdm modulator and demodulator based on discrete hartley transform," in *Modelling Symposium (AMS), 2011 Fifth Asia*, may 2011, pp. 252 –256.

[80] Z. Sembiring and M. Syahruddin, "Performance analysis of discrete hartley transform based ofdm modulator and demodulator," in *Intelligent Systems, Modelling and Simulation (ISMS), 2012 Third International Conference on*, feb. 2012, pp. 674 –679.

[81] R. Bracewell, "Assessing the hartley transform," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 38, no. 12, pp. 2174 –2176, dec 1990.

[82] R. N. Bracewell, "Discrete Hartley transform," *J. Opt. Soc. Am.*, vol. 73, pp. 1832–1835, 1983.

[83] ——, "The fast Hartley transform," *Proceedings of The IEEE*, vol. 72, pp. 1010–1018, 1984.

[84] H. Sorensen, D. Jones, C. Burrus, and M. Heideman, "On computing the discrete Hartley transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, pp. 1231–1238, 1985.

[85] D. Yang, "Prime factor fast hartley transform," *Electronics Letters*, vol. 26, no. 2, pp. 119 –121, jan. 1990.

[86] D.-K. Lun and W.-C. Siu, "Fast radix-3/9 discrete hartley transform," *Signal Processing, IEEE Transactions on*, vol. 41, no. 7, pp. 2494 –2499, jul 1993.

[87] G. Bi, "A split radix algorithm of discrete hartley transform," in *Intelligent Information Systems,1994. Proceedings of the 1994 Second Australian and New Zealand Conference on*, nov-2 dec 1994, pp. 165 –168.

[88] R. Campello de Souza, H. de Oliveira, and A. Kauffman, "The hartley transform in a finite field," in *Telecommunications Symposium, 1998. ITS '98 Proceedings. SBT/IEEE International*, aug 1998, pp. 245 –250 vol.1.

[89] G. Bi and Y. Q. Chen, "Fast dht algorithms for length n=q*2m," *Signal Processing, IEEE Transactions on*, vol. 47, no. 3, pp. 900 –903, mar 1999.

[90] H. M. de Oliveira, R. J. S. Cintra, and R. M. Campello de Souza, "Multilevel Hadamard decomposition of discrete Hartley transforms," *Anais do XVIII Simpósio Brasileiro de Telecomunicações*, Setembro 2000.

[91] R. J. S. Cintra, H. M. de Oliveira, and C. O. Cintra, "The rounded Hartley transform," *Proc. IEEE/SBrT Int. Telecomm. Symp.*, pp. 455–460, 2002.

[92] S. Bouguezel, M. Ahmad, and M. Swamy, "An approach for computing the radix-2/4 dit fht and fft algorithms using a unified structure," in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, may 2005, pp. 836 – 839 Vol. 1.

[93] G. Shah and T. Rathore, "A new fast radix-2 decimation-in-frequency algorithm for computing the discrete hartley transform," in *Computational Intelligence, Communication Systems and Networks, 2009. CICSYN '09. First International Conference on*, july 2009, pp. 363 –368.

[94] M. Hamood and S. Boussakta, "New radix-based fht algorithm for computing the discrete hartley transform," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, may 2011, pp. 1581 –1584.

[95] G. Bergland, "Fast Fourier transform hardware implementations – an overview," *IEEE Trans. on Audio and Electroacoustics*, vol. 17, no. 2, pp. 104–108, 1969.

[96] H. Miyanaga and H. Yamauchi, "A 400 mflops FFT processor vlsi architecture," *IEICE TRANSACTIONS on Electronics*, vol. E74-C, no. 11, pp. 3845–3851, 1991.

[97] W. F. Yen, S. D. You, and Y. C. Chang, "Real-time FFT with pre-calculation," *Comput. Electr. Eng.*, vol. 35, pp. 435–440, May 2009.

[98] A. Edelman, P. McCorquodale, and S. Toledo, "The future fast Fourier transform?" *SIAM J. Sci. Comput.*, vol. 20, pp. 1094–1114, January 1999.

[99] A. Papoulis, *The Fourier Integral and Its Applications.* New York, USA: McGraw-Hill Book Company, 1962.

[100] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing.* New Jersey, USA: Prentice Hall, 2009.

[101] E. O. Brigham and R. E. Morrow, "The fast Fourier transform," *IEEE Spectrum*, vol. 4, pp. 63–70, 1967.

[102] S. J. Orfanidis, *Introduction to Signal Processing.* Rutgers University: Prentice Hall, Inc., 2010.

[103] K. J. Olejniczak and G. T. Heydt, "Special section on the Hartley transform," *Proceedings of the IEEE*, vol. 82, pp. 372–447, March 1994.

[104] A. M. Despain, "Very fast Fourier transform algorithms hardware for implementation," *IEEE Transactions on Computers*, vol. C–28, no. 5, pp. 333–341, 1979.

[105] H. M. D. OLIVEIRA and R. M. C. D. SOUZA, "A fast algorithm for computing the Hartley/Fourier spectrum," *Anais da Academia Brasileira de Ciências*, vol. 73, pp. 468 – 468, 09 2001. [Online]. Available: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0001-37652001000300025&nrm=iso

[106] D. M. Burton, *Elementary number theory*, 3rd ed. McGraw-Hill, 1997.

[107] R. J. McEliece, *Finite Fields for Computer Scientists and Engineers.* Massachusetts, USA: Kluwer Academic Publishers, 1987.

[108] G. Ávila, *Variáveis Complexas e Aplicações.* Rio de Janeiro, RJ: LTC, 2000.

[109] K. Adel-Ghaffar, "Long division from laurent series matrices and the optimal assignment problem," *Linear Algebra and its Application*, vol. 280, pp. 189–197, September 1998.

[110] H. L. Resnikoff and R. Wells, *Wavelet Analysis: the Scalable structure of Information.* New York, USA: Springer–Verlag, 1998.

[111] P. Diniz, E. Silva, and S. Netto, *Digital signal processing: system analysis and design.* New York, USA: Cambridge University Press, 2002.

[112] H. M. de Oliveira, V. L. Sousa, H. A. N. Silva, and R. M. Campello de Souza, "Radix-2 fast Hartley transform revisited," *I Congresso de Informática da Amazônia*, vol. 1, pp. 285–292, April 2001.

[113] M. T. Heideman and C. S. Burrus, "On the number of multiplications necessary to compute a length-2n DFT," *IEEE transactions on acoustics, speech, and signal processing*, vol. 34, pp. 91–95, April 1986.

[114] R. J. S. Cintra and H. M. de Oliveira, "A short survey on arithmetic transforms and the arithmetic Hartley transform," *Journal of the Brazilian Telecommunications Society*, vol. 19, pp. 68–79, August 2004.

[115] H. M. de Oliveira, *Análise de Sinais para Engenheiros: Uma abordagem via WAVELETS.* Rio de Janeiro, RJ, Brasil: Série da Soc. Bras. de Telecomunicações, Brasport, 2007.

[116] J. G. Liu, F. H. Y. Chan, F. K. Lam, H. F. Li, and G. S. K. Fung, "Moment-based fast discrete Hartley transform," *Signal Process.*, vol. 83, pp. 1749–1757, August 2003.

[117] R. Woods, J. McAllister, G. Lightbody, and Y. Yi, *FPGA-based implementation of complex signal processing system.* Great Britain: A John Wiley and Sons, Ltd. now Publishers Inc., 2008.

[118] D. Chen, J. Cong, and P. Pan, *FPGA design automation: a survey.* USA: now Publishers Inc., 2006.

[119] V. Britanak and K. Rao, "The fast generalized discrete Fourier transforms: A unified approach to the discrete sinusoidal transforms computation," *Signal Processing*, vol. 79, pp. 135 – 150, 1999.

[120] N. Hu, H. Chang, and O. K. Ersoy, "Generalized discrete Hartley transfonns," *IEEE Transaction on Signal Processing*, vol. 40, pp. 2931 – 2940, 1992.

APÊNDICE A

Fatorando uma Matriz $A(n \times n)$ em Matrizes Bielementares

A.1 Introdução

Definição A.1 *Uma matriz bielementar é uma matriz, com elementos racionais, em que cada uma das suas linhas tem no máximo, dois elementos não nulos; além disso, as linhas com dois elementos não nulos, consideradas como vetores no* $\mathbb{R}^N$*, devem apresentar direções distintas.*

Definição A.2 *A distância de Hamming entre dois vetores* x *e* y *é o número de posições em que as componentes de cada vetor são diferentes e é denotada por* $d(x, y)$*.*

Definição A.3 *O peso de Hamming de um vetor* $x = (x_0, ..., x_{n-1})$ *é o número de componentes não nulos de* x *e é denotado por* $w(x)$*.*

Exemplo A.1

Considerando $x = (1101)$ e $y = (1100)$, tem-se $w(x) = 3$, $w(y) = 2$ e $d(x, y) = 2$.

•

A.2 Procedimento para Fatoração da Matriz

O procedimento para fatoração de uma Matriz $A(n \times n)$ em matrizes bielementares consiste em:

1. Escolher o par de colunas que apresenta a menor distância de Hamming dentre todos os pares de colunas de A. Caso mais de um par atenda à condição escolher aleatoriamente um tal par;
2. Iniciar a construção de uma matriz $B(n \times n)$ a partir do par de colunas escolhido em (1), escrevendo apenas as linhas distintas do par e preenchendo com 0 (zeros) as demais posições destas linhas;
3. Preencher as linhas restantes com 0 (zeros) e um único 1 (um), colocado na primeira linha disponível na posição j (coluna) correspondente à coluna com maior peso de Hamming em A. Caso haja mais de uma coluna com tal peso, escolher aleatoriamente uma tal coluna. Repetir para as demais linhas; neste passo as colunas escolhidas no passo 1 não são consideradas.
4. Ecotrar a matriz C tal que $A = CB$;
5. Verificar se a matriz C é bielementar. Caso positivo, a fatoração está terminada. Caso contrário, repetir o procedimento a partir de (1).

Exemplo A.2

Deseja-se computar $V = Av$ em que

$$\begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & -1 \\ 1 & 0 & -1 & 1 \\ 1 & -1 & 1 & 1 \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{pmatrix}.$$

A matriz A apresenta 9 operações de adições, porém fazendo uma inspeção visual nota-se que existem adições repetidas. Vamos aplicar o procedimento descrito anteriormente para eliminarmos estas repetições.

No primeiro passo, deve-se escolher o par de colunas que tem menor distância de Hamming. O par que satisfaz esta condição é formado pelas colunas 0 e 3. A partir destas colunas, constrói-se a matriz B, conforme o passo 2, ou seja, escolhe-se as linhas distintas do par e preenche-se todas as outras posições com zero; para as linhas restantes, inicia-se preenchendo com um único 1, na coluna de maior peso de Hamming; então segue-se a ordem decrescente de peso de Hamming até a última coluna. Neste exemplo, primeiro a coluna 2 e em seguida a coluna 1. Observa-se que a matriz B obtida é bielementar

$$B = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

No passo 4, encontra-se a matriz C, tal que, $A = CB$. O resultado foi

$$C = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & 0 & 1 & -1 \end{pmatrix}.$$

Analisando a matriz C desta primeira fatoração (passo (5)) percebe-se que a mesma não é bielementar; salienta-se que o procedimento só se encerrará quando todas as matrizes forem bielementares e, como C não satisfaz esta condição, volta-se ao passo (1). Para iniciar novamente o procedimento, faz-se $A = C$,

$$A = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & 0 & 1 & -1 \end{pmatrix}.$$

Aplicando o passo (1), escolhemos as colunas 0 e 2 da nova matriz A para formação da matriz B; em seguida efetuamos os passos (2) e (3) obtendo a matriz B,

$$B = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

No próximo passo (4), deve-se encontrar a matriz C que satisfaça à condição de fatoração $A = CB$, com isso, esta ficou da seguinte forma

$$C = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & -1 \end{pmatrix}.$$

Analisando esta nova matriz C, ela satisfaz a condição de matriz bielementar, portanto a fatoração está terminada. O processo gerou a seguinte fatoração

$$A = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

Fazendo a contagem das operações necessárias após a fatoração, chegamos a 6 adições, ou seja, houve uma redução de $33,3\%$ em relação à matriz original. A figura A.1 mostra uma implementação da fatoração.

•

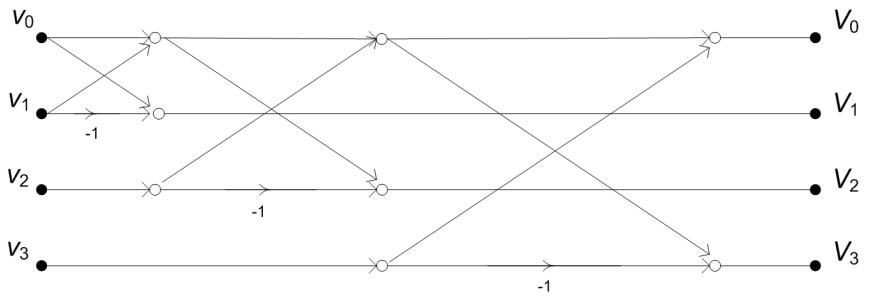


Figura A.1: Esquema para a computação aditiva referente à matriz do exemplo A.2. Um possível vetor de entrada (v) e um vetor de saída (V) ilustram o comportamento da implementação.