Universidade Federal de Juiz de Fora
Curso de Graduação em Engenharia Elétrica
Habilitação em Sistemas Eletrônicos

Mateus Mostaro de Oliveira

IMPLEMENTAÇÃO EM FPGA DE UM SISTEMA DE COMPRESSÃO DE SINAIS ELÉTRICOS UTILIZANDO REPRESENTAÇÃO ESPARSA

Monografia de Conclusão de Curso

Juiz de Fora
2017

Mateus Mostaro de Oliveira

Implementação em FPGA de um Sistema de Compressão de Sinais Elétricos Utilizando Representação Esparsa

Monografia apresentada a Coordenação do Curso de Engenharia Elétrica da Universidade Federal de Juiz de Fora, Habilitação em Sistemas Eletrônicos, como requisito para aprovação na disciplina CEL046 - Trabalho Final de Curso.

Orientador: Prof. Carlos Augusto Duque, D.Sc.
Coorientador: Prof. Leandro Rodrigues Manso Silva, Dr. Eng.

Juiz de Fora
2017

Ficha catalográfica elaborada através do programa de geração
automática da Biblioteca Universitária da UFJF,
com os dados fornecidos pelo(a) autor(a)

Mostaro de Oliveira, Mateus.
Implementação em FPGA de um Sistema de Compressão de Sinais Elétricos / Mateus Mostaro de Oliveira. -- 2017.
64 p.

Orientador: Carlos Augusto Duque
Coorientador: Leandro Rodrigues Manso Silva
Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Juiz de Fora, Faculdade de Engenharia, 2017.

1. Introdução. 2. Revisão Bibliografica. 3. Representação Esparsa para compressão de sinais elétricos. 4. Implementação em FPGA do algoritmo de compressão de sinais. 5. Resultados. I. Augusto Duque, Carlos, orient. II. Rodrigues Manso Silva, Leandro, coorient. III. Título.

Mateus Mostaro de Oliveira

Implementação em FPGA de um Sistema de Compressão de Sinais Elétricos Utilizando Representação Esparsa

Monografia apresentada a Coordenação do Curso de Engenharia Elétrica da Universidade Federal de Juiz de Fora, Habilitação em Sistemas Eletrônicos, como requisito para aprovação na disciplina CEL046 - Trabalho Final de Curso.

Aprovada em 12 de Julho de 2017.

BANCA EXAMINADORA:

Prof. Carlos Augusto Duque, D.Sc.
Universidade Federal de Juiz de Fora, UFJF

Prof. Leandro Rodrigues Manso Silva, Dr. Eng.
Universidade Federal de Juiz de Fora, UFJF
Coorientador

Prof. Luciano Manhães de Andrade Filho, D.Sc.
Universidade Federal de Juiz de Fora, UFJF

Dedico este trabalho à minha família, amigos e colegas de curso.

AGRADECIMENTOS

Aos Professores Carlos Augusto Duque, D.Sc.e Leandro Rodrigues Manso Silva, Dr. Eng. pela orientação, ensinamentos, amizade, atenção e paciência durante todo o desenvolvimento deste trabalho.

Agradeço aos meus pais, Maria Cristina e Sebastião Eduardo que sempre me apoiaram nas minhas decisões. Agradeço pelo esforço que fizeram, sempre priorizando a minha educação. Se hoje cheguei onde estou é graças a ajuda de vocês.

Aos companheiros do laboratório LAPTEL, em especial ao Henrique Moreira Monteiro, Carlos Henrique Martins, Eder Kapisch, Renato Ribeiro e Victor Morais, que me ajudaram nas pesquisas. Agradeço a todos ainda pelos ensinamentos e pela amizade.

A todos os colegas de curso pelas horas de estudos e trabalhos feitos, aprendi muito com eles durante a trajetória acadêmica e espero levar essas amizades para o resto de minha vida.

Agradeço aos meus tios e tias que sempre estiveram presentes nas horas certas, me apoiando, aconselhando e corrigindo quando necessário, em especial a Tia Maranice Mostaro.

Ao CNPq, CAPES, FAPEMIG, à Universidade Federal de Juiz de Fora e à Faculdade de Engenharia pelo suporte financeiro e por prover as ferramentas necessárias para o desenvolvimento deste trabalho.

A vida é igual andar de bicicleta. Para manter o equilíbrio é preciso se manter em movimento.

Albert Einstein, Físico teórico

RESUMO

O armazenamento de sinais de tensão e corrente durante um longo período de tempo, leva a uma grande quantidade de gastos com memória. Diante deste fato, técnicas de compressão de sinais tornam-se importantes neste contexto. O presente trabalho apresenta a implementação em *Field Programmable Gate Array* (FPGA) de um algoritmo de representação esparsa utilizando dicionários redundantes, aplicada a compressão de sinais elétricos, advindos de sistemas elétricos de potência. O algoritmo será descrito usando um dicionário formado por elementos das bases de Fourier e Wavelet, que são selecionados com base na constituição de sinais elétricos de potência. Para realizar a compressão utiliza-se técnicas de processamento digital de sinais em tempo real, aplicadas em processadores dedicados, embarcados em FPGA. São realizados testes de simulação funcional, utilizando o Modelsim da Altera e os resultados são comparados com implementações feitas em MATLAB.

Palavras-chave: Compressão de Sinais, Representação Esparsa, FPGA.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

FPGA *Field Programmable Gate Array*

PQDA Analisador de Dados de Qualidade de Energia (do inglês, *Power Quality Data Analyzer*)

FDR Registrador de Dados de Falha (do inglês, *Fault Data Recirder*)

SEP sistemas elétricos de potência

PL Programação Linear

MOD *Method of Optimal Directions*

MP *Matching Pursuit*

NMSE Erro Médio Quadrático Normalizado, (do inglês, *Normalized Mean Squared Error*)

COR Correlação Cruzada

RTE Porcentagem de energia mantida

FFT Transformada rápida de Fourier, (do inglês, *Fast Fourier Transform*)

DFT Transformada Discreta de Fourier, (do inglês, *Discret Fourier Transform*)

IFFT Transformada Rápida de Fourier Inversa , (do inglês, *Inverse Fast Fourier Transform*)

IDFT Transformada discreta de Fourier Inversa, (do inglês, *Inverse Discret Fourier Transform*)

MSB *Most Significant Bit*

LSB *Least Significant Bit*

FIFO *First In First Out*

Mux Multiplexador

Demux Demultiplexador

RISC *Reduced Instruction Set Computer*

DSP Processamento Digital de Sinais

ALU Unidade Lógica Aritmética

DSP processador digital de sinais (do inglês, *Digital Signal Processor*)

SUMÁRIO

1 INTRODUÇÃO

1.1 IDENTIFICAÇÃO DO PROBLEMA

A aquisição continua de sinais elétricos é necessária em muitas aplicações de sistemas de potência, tais como proteção, controle e análise de qualidade de energia. A razão para adquirir e armazenar esta grande quantidade de dados é apoiada pelo fato de que o pós-processamento dos dados pode revelar informações não observadas anteriormente e que podem auxiliar na melhoria do sistema, na otimização de algoritmos, entre outros aspectos. Além da aplicação natural, que é avaliar a qualidade do sistema.

O aumento da demanda por energia elétrica e a crescente utilização de geração dispersa e distribuída, interligadas aos sistemas elétricos de potência, tornam a operação, o controle e a proteção destes sistemas missões cada vez mais complexas. Além disso, visando aprimorar o fornecimento da energia elétrica, grande parte dos investimentos das empresas de transmissão e distribuição é destinada a fortalecer tecnologicamente sua infraestrutura operacional. Portanto, existe uma crescente necessidade de se caracterizar corretamente o comportamento desses sistemas.

Cerca de duas décadas atrás, devido à desregulamentação do setor elétrico em todo o mundo, usuários e empresas publicas passaram a se preocupar com os impactos causados por problemas de qualidade de energia. Esses problemas surgem como resposta ao uso massivo de cargas não-lineares e equipamentos eletrônicos em residências, centros comerciais e instalações industriais. Além disso, foi percebido que essas deficiências poderiam se intensificar caso não fossem corretamente corrigidas. Isto fez com que a monitoração da qualidade de energia elétrica recebesse muita atenção durante as duas últimas décadas, o que impulsionou e continua impulsionando o desenvolvimento de equipamentos capazes de detectar, armazenar, classificar e analisar dados de qualidade da energia do sistema. Várias tecnologias têm surgido com o intuito de monitorar o comportamento de sistemas de energia elétrica em diferentes níveis de tensão.

O armazenamento de dados brutos do sinal elétrico (tensão e corrente) de forma

continua não é tarefa simples, devido à grande quantidade de dados a serem armazenados na memória, posteriormente transferidos para uma central de processamento onde os dados poderão ser processados. Adicionalmente, poucos equipamentos comerciais estão disponíveis no momento para gravação de dados de oscilografia(forma de onda dos sinais) durante um longo período de tempo e em uma taxa de aquisição relativamente alta, visando a recuperação dos sinais elétricos. Em geral os equipamentos disponíveis para essa finalidade são orientados a aplicações específicas e são utilizados apenas para a aquisição de um curto período de tempo do sinal de falha ou sinal de perturbação. Dois exemplos de equipamentos comerciais que adquirem sinais brutos são o Analisador de Dados de Qualidade de Energia (do inglês, *Power Quality Data Analyzer*) (PQDA) e o Registrador de Dados de Falha (do inglês, *Fault Data Recirder*) (FDR). O PQDA é especializado em detectar e registrar os distúrbios.

No novo cenário de *Smart-Grids* descrito acima, onde o comportamento da rede elétrica ainda não é claramente compreendido pelos cientistas e engenheiros, a aquisição de dados brutos e de forma contínua para serem processados pode ser extremamente importante para identificar erros e desvios na operação, proteção e controle do sistema e também podem ser utilizados para encontrar informações aparentemente escondidas no sinal.

1.2 MOTIVAÇÃO

No cenário descrito acima, a gravação dos dados de oscilografia será de grande importância em sistemas elétricos de potência (SEP), especialmente no contexto de rede elétricas inteligentes. Portanto é esperado que os equipamentos que irão armazenar estes dados estejam munidos de técnicas capazes de reduzir a quantidade de dados que serão armazenados e/ou transmitidos, sem que haja perdas de informação relevantes. Isto abre espaço para a pesquisa, desenvolvimento e implementação de novas técnicas de compressão de dados, em especial as que possibilitam uma maior taxa de compressão mesmo que ao custo de uma complexidade computacional maior.

Esses equipamentos devem fazer uso de técnicas de compressão, operando em tempo real para que o armazenamento seja eficiente. A maioria dos métodos de compressão sinais elétricos é baseada na Transformada de *Wavelet*, nas suas mais diversas formas de aplicação. Porém, outras técnicas estão começando a ser utilizados nessa área. Como por exemplo, técnicas de representação esparsa de sinais em dicionários redundantes, largamente utilizadas para compressão de imagens. Uma característica dessa técnica é a

elevada complexidade computacional, que muitas vezes impossibilita a implementação em tempo real. Portanto, há necessidade de se realizar um estudo das diversas técnicas de representação esparsa para avaliar sua viabilidade de implementação neste tipo de equipamento.

Como consequência natural a utilização de hardwares reconfigurável, como FPGA se torna atrativa. Como o uso dessa tecnologia, é possível sintetizar, em hardware, algoritmos de processamento de sinais eficazes para compressão, mas que demandam elevado custo computacional. Adicionalmente é possível desenvolver e embarcar processadores para realizar tarefas especificas. Portanto os FPGAs se mostram como uma tecnologia muito promissora para constituir a base que contém a inteligência necessária para um equipamento que armazene as formas de onda de tensão e corrente advindas de um SEP.

1.3 OBJETIVOS

O presente trabalho tem como objetivo implementar, em processadores embarcados em FPGA, um algoritmo de representação esparsa de sinais visando sua utilização em um sistema de compressão de sinais elétricos de sistemas de potência.

Dentro desse contexto, é proposto avaliar sua viabilidade da sua utilização em tempo real, bem como os recursos necessários do FPGA e a qualidade dos resultados comparados com os resultados em MATLAB.

1.4 ESTRUTURA DA MONOGRAFIA

O atual trabalho foi subdividido em seis capítulos descritos a seguir:

No Capítulo 2 será feita uma revisão bibliográfica dos conceitos de decomposição atômica de sinais e representação esparsa de sinais via dicionários redundantes.

No Capítulo 3 será realizado o estudo da técnica de representação esparsa de sinais em dicionários redundantes, e será apresentado o algoritmo utilizado neste trabalho.

No Capítulo 4 é apresentado a implementado em FPGA do algoritmo de compressão de sinais elétricos.

No Capítulo 5 serão apresentados resultados de compressão obtidos para o sistema de compressão proposto.

Por fim, o Capítulo 6 apresentará conclusões gerais deste trabalho e algumas propostas para a continuidade desta pesquisa serão feitas.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo será feita uma revisão bibliográfica a respeito os principais temas que constituem o foco deste trabalho compactação de sinais elétricos e representação esparsa de sinais.

2.1 REPRESENTAÇÃO ESPARSA DE SINAIS ATRAVÉS DE DICIONÁRIOS REDUNDANTES

Antes de introduzir a representação esparsa, deve-se compreender o conceito de decomposição atômica de sinais, que consiste em utilizar formas de onda pré-definidas para expressar sinais. Essas formas de onda são chamadas de átomos e são elementos de um conjunto denominado dicionário. Essa decomposição é considerada adaptativa, pois os elementos do dicionário são escolhidos de acordo com o sinal que se deseja representar. Matematicamente falando, tem-se uma matriz $\mathbf{A} \in \mathbb{R}^{N \times M}$, chamada de dicionário, um vetor de coeficientes $\mathbf{x} \in \mathbb{R}^{M}$ e o sinal $\mathbf{b} \in \mathbb{R}^{N}$ é representado como uma combinação linear das M colunas (átomos) de $\mathbf{A}$, como descrito pela equação (2.1).

$$\mathbf{Ax} = \mathbf{b} \tag{2.1}$$

Técnicas de decomposição atômica de sinal são utilizadas em trabalhos nas mais diversas áreas, tais como: para remoção de ruídos em (KRIM et al., 1999) análise harmônica (GRIBONVAL & BACRY, 2003) (DONOHO et al., 1998) extração de parâmetros (JAGGI et al., 1998) decomposição tempo frequência (GOODWIN & VETTERLI, 1999) (VERA-CANDEAS et al., 2004) entre outras.

Em problemas de representação atômica, a matriz $\mathbf{A}$ é considerada um dicionário redundante, pois possui mais elementos, ou funções, do que as necessárias para se estabelecer uma base. Dessa forma, a matriz $\mathbf{A}$ possui mais colunas do que linhas (M>N), e portanto o sistema mostrado na equação (2.1) possui varias soluções. A solução que utiliza o menor número de elementos do dicionário é considerada a representação mais

esparsa do sinal. Portanto, técnicas de representação atômica podem também ser utilizadas em compressão de sinais, em que o problema é justamente encontrar uma solução para (2.1) que seja esparsa.

Dessa maneira, as técnicas que se utilizam de representação esparsa encontram dois principais problemas: (i) dada uma matriz dicionário, como encontrar a solução da equação (2.1) com o menor número de elementos; e (ii) como construir a matriz dicionário.

Tratando-se do primeiro problema, tem-se um sistema de equações indeterminado e procura-se uma solução especifica deste sistema, portanto deve-se estabelecer um critério que consiga expressar as características desejadas dessa solução. Uma forma de se introduzir esse critério é através de uma função custo *J(.)*, e então um problema de otimização generalizado pode ser escrito da seguinte forma:

$$(P_{\mathrm{J}}) : \min_{\mathbf{x}} J(\mathbf{x}) \text{ sujeito a } \mathbf{b} = \mathbf{A}\mathbf{x} \tag{2.2}$$

A Equação (2.2) significa que deseja-se minimizar a função, *J(x)*, em relação à $\mathbf{x}$, e esse $\mathbf{x}$ deve estar sujeito à equação $\mathbf{b} = \mathbf{A}\mathbf{x}$. Vale a pena ressaltar então que essa função custo *J(x)* determinará que tipo de solução será obtida para o problema. Uma função de custo muito utilizada é a norma Euclidiana, ou norma l_2, elevada ao quadrado $||x||_2^2$ O problema que resulta dessa escolha é chamado de P_2, definido pela Equação (2.3), e possui solução única (ELAD & AHARON, 2010) como mostrada em (2.3):

$$(P_2) : \min_{\mathbf{x}} ||\mathbf{x}||_{\mathbf{2}}^{\mathbf{2}} \text{ sujeito a } \mathbf{b} = \mathbf{A}\mathbf{x} \tag{2.3}$$

$$\hat{\mathbf{x}}_{opt} = \mathbf{A}^{+}\mathbf{b} \tag{2.4}$$

em que, $\mathbf{A}^{+}$, é a pseudo inversa de $\mathbf{A}$, dada por $\mathbf{A^T(AA^T)^{-1}}$. A escolha pela norma l_2 é fundamentada pelo fato de possuir uma solução fechada, e por se assemelhar a técnicas de minimização de erro médio quadrático, que são muito difundidas em algoritmos populares de processamento de sinais, como Filtro de Wiener, Filtro de Kalman e *Least Squares*. Porém, em situações onde se requer uma solução esparsa, deve-se escolher uma função custo que promova esparsidade em $\mathbf{x}$. Em (ELAD & AHARON, 2010) é matematicamente demonstrado que, dado um problema de minimização da norma l_p, em que:

$$||\mathbf{x}||_p = \sum_i |x_i|^p \tag{2.5}$$

quanto menor o valor de p, mais esparsa é a solução. Desta forma, uma possível escolha para a função de custo que tende a gerar soluções esparsas é: $J(\mathbf{x}) = ||\mathbf{x}||_1$. A utilização da norma l_1 tende a promover a esparsidade já que esta é dada pela soma dos valores absolutos dos elementos de $\mathbf{x}$. O problema de otimização se torna então:

$$(P_1) : \min_{\mathbf{x}} ||\mathbf{x}||_1 \text{ sujeito a } \mathbf{b} = \mathbf{A}\mathbf{x} \tag{2.6}$$

Diferentemente de (P_2), não é possível minimizar l_1 analiticamente, por possuir uma descontinuidade, e portanto necessita ser resolvido por algum método numérico de otimização. Uma técnica bastante utilizada é a de se transformar (P_1) em um problema de Programação Linear (PL). Para isto, escreve-se $\mathbf{x = u - v}$, em que $\mathbf{u}$ é um vetor que contenha somente os elementos positivos, e $\mathbf{v}$ somente o módulo dos elementos negativos do vetor $\mathbf{x}$. Pode-se então criar o vetor $\mathbf{z = [u^T, v^T]}$ que é a concatenação dos vetores $\mathbf{u}$ e $\mathbf{v}$. Pode-se verificar que a norma l_1 de $\mathbf{x}$ pode ser escrita da seguinte forma: $||\mathbf{x}||_1 = 1^{\mathrm{T}}\mathbf{(u + v)}$ e que $\mathbf{Ax = A(u - v) = [A, -A]z}$. Este artifício matemático elimina a operação de modulo do processo de minimização, eliminando assim, a descontinuidade. Portanto o problema (P_1) pode ser reescrito como mostrado na Equação (2.7) e ser solucionado por técnicas de PL conhecidas na literatura.

$$\min_{\mathbf{z}} \mathbf{1^T z}_1 \text{ sujeito a } \mathbf{b} = [\mathbf{A}, -\mathbf{A}]\mathbf{z},\ \mathbf{z} \geq \mathbf{0} \tag{2.7}$$

É sabido que normas l_{p}, para $p < 1$, geram soluções ainda mais esparsas. Porém estas não são ditas normas formais, visto que algumas propriedades podem não ser satisfeitas. Entre as normas l_{p}, para $p < 1$, a norma l_0 é a que melhor descreve esparsidade, já que seu valor é o número de elementos não nulos contidos no vetor. O problema (P_{J}) se transforma agora em (P_0) e é mostrado na seguinte equação:

$$(P_0) : \min_{\mathbf{x}} ||\mathbf{x}||_0 \text{ sujeito a } \mathbf{b} = \mathbf{A}\mathbf{x} \tag{2.8}$$

A solução para o problema mostrado em (2.8) apresenta alguns desafios devido à natureza discreta e descontínua da norma l_0. A solução de (P_0) é um problema clássico de busca combinatória, em que deve-se gerar todos os possíveis subsistemas esparsos

$\mathbf{b} = \mathbf{A_S x_S}$, em que $\mathbf{A_S}$ é uma matriz que contenha apenas as $|S|$ colunas da matriz $\mathbf{A}$ com índices contidos em S e testar se cada um desses subsistemas podem ser resolvidos. A complexidade dessa solução é exponencial em m (número de colunas da matriz $\mathbf{A}$), portanto, (P_0) é classificado como um problema NP-*Hard* (LEEUWEN, 1990).

Já que a solução direta do problema (P_0) é inviável em termos computacionais, deve-se buscar outras possíveis soluções confiáveis. É observado que a tarefa de encontrar a variável $\mathbf{x}$ pode ser dividida em duas etapas: encontrar o seu suporte, ou seja, os índices dos elementos não nulos e determinar o valor desses elementos. Então, uma maneira de se atacar esse problema é focar na obtenção do suporte, pois uma vez obtido, os valores de seus coeficientes podem ser encontrados com a aplicação da Equação (2.4), para $\mathbf{A}^+ = \mathbf{AS}$ (ELAD & AHARON, 2010). Este raciocínio leva à família dos algoritmos *greedy* ou gulosos.

Esses algoritmos são inicializados com o vetor de solução $\mathbf{x_0} = \mathbf{0}$ e com um resíduo inicial dado por $\mathbf{r_0} = \mathbf{b - Ax_0} = \mathbf{b}$. A cada iteração k é executado um estágio de varredura, onde o objetivo é encontrar a coluna da matriz $\mathbf{A}$ que mais se assemelha ao resíduo obtido na iteração anterior $\mathbf{r}^{k-1}$. Essa coluna é então adicionada ao suporte de x e uma solução provisória para o problema é encontrada. O novo resíduo $\mathbf{r^k} = \mathbf{b - Ax^k}$ é então calculado e algum critério de parada é avaliado. Caso a aproximação ainda não seja suficientemente boa, volta-se ao estágio de varredura e adiciona-se mais um elemento ao suporte de $\mathbf{x}$.

Em processamento de sinais, pode-se dar destaque ao algoritmo denominado *Matching Pursuit* (MP), proposto em (MALLAT & ZHANG, 1993), em que após o estágio de varredura apenas o coeficiente do elemento adicionado é calculado. Este algoritmo será tratado com mais destaque no Capítulo 3 deste trabalho.

A escolha do dicionário é um ponto fundamental, pois impactará diretamente na esparsidade da aproximação. A utilização de dicionários pré-definidos geralmente levam a transformações mais rápidas, porém são limitados em representar, satisfatoriamente, de forma esparsa apenas a classe de sinais para a qual foi projetado. Uma maneira de superar essa limitação é a utilização de dicionários adaptativos. Para isso é necessário um conjunto de dados de treinamento que contenha sinais parecidos com os sinais que se deseja representar, e então, os elementos do dicionário serão construídos de acordo com esse conjunto. A escolha por um dicionário que aprende acarreta um custo computacional elevado. Existem na literatura vários métodos que realizam esse treinamento do dicionário. Estes métodos utilizam um conjunto de dados de treinamento $\mathbf{Y} = (y_i)_{i=1}^N$

constituídos de exemplos dos sinais que se deseja representar, e partem do pressuposto que existe uma matriz dicionário **A** que é capaz de fornecer uma representação esparsa para cada elemento y_{i} desse conjunto. A tarefa é então encontrar qual é essa matriz **A**.

Pode-se citar alguns métodos como os utilizados em (LEWICKI & OLSHAUSEN, 1999), em que dado o conjunto de treinamento **Y**, procuram o dicionário que maximize a razão de verossimilhança $P(\mathbf{Y}|\mathbf{A})$. O método denominado *Method of Optimal Directions* (MOD) apresentado em (ENGAN; AASE & HUSOY, 1999)é mais próximo do K-*means*, utilizado em problemas de clusterização de vetores (FORGY, 1965), e se utiliza de um estágio de aproximação esparsa, seguido de um estágio de atualização do dicionário. Uma maneira de aliar a eficiência do MOD com a maneira natural de se considerar preferências no dicionário é a utilização da maximização da probabilidade a posteriori P(A|Y) (KREUTZ-DELGADO & RAO, 2000). Uma técnica para a atualização de dicionários constituídos por uma união de bases ortonormais é proposta em (LESAGE et al., 2005). Dessa forma, uma de suas maiores vantagens é a relativa simplicidade do algoritmo de busca para realizar a representação esparsa. Este algoritmo é mais simples dos que os outros mostrados devido a duas características: utiliza um dicionário estruturado e realiza sua atualização de forma estruturada. Uma generalização direta do algoritmo K-*means* é o K-SVD proposto por (AHARON; ELAD & BRUCKSTEIN, 2006). Esse algoritmo apresenta elevada eficiência devido à sua etapa de representação esparsa eficiente e seu método de atualização do dicionário que é do tipo Gauss-Seidel

3 REPRESENTAÇÃO ESPARSA PARA COMPRESSÃO DE SINAIS ELÉTRICOS

Neste capítulo será mostrado o uso de uma técnica de representação esparsa de sinais através de dicionários redundantes aplicado à compressão de sinais de sistemas elétricos de potência. Será discutido a maneira como o dicionário utilizado foi construído.

3.1 CONSTRUÇÃO DO DICIONÁRIO

Em representação esparsa de sinais, um dos dicionários mais conhecidos e utilizados é o dicionário de Gabor (GABOR, 1946), devido a sua capacidade de representar bem qualquer tipo de sinal, que contenha tanto componentes estacionários quanto transitórios. Entretanto, é necessário um dicionário com muitos elementos para que se tenha uma boa representação, o que afeta diretamente o desempenho dos algoritmos utilizados para encontrá-la. Dessa forma, quando busca-se um dicionário que seja capaz de ser aplicado em tempo real, o dicionário de Gabor mostra-se não ser o mais eficiente.

Em sistemas elétricos de potência, as ferramentas mais utilizadas em representação de sinais são a transformada de Fourier, que utiliza como base funções senoidais, e a transformada Wavelet, que utiliza funções com suporte temporal finito. A decomposição linear de um sinal em somente uma dessas bases não é muito flexível (MALLAT & ZHANG, 1993). A base de Fourier representa bem sinais estacionários no tempo e a base de Wavelet é mais indicada para sinais com suporte temporal finito.

Sabe-se que os sinais de tensão e corrente em um sistema elétrico de potência possuem tanto componentes estacionários, ou quase estacionários (componente fundamental, harmônicos e interharmônicos) quanto componentes não estacionários amortecidos, como é o caso dos transitórios oscilatórios. Os primeiros são bem representados pela base de Fourier e os últimos pela base Wavelet. Sendo assim, um dicionário formado pela união dessas duas bases seria capaz de fornecer uma boa representação para esse tipo de sinal. O fato de que o dicionário é constituído pela união de duas bases or-

togonais implicará diretamente no desempenho dos algoritmos que serão descrito e foram testados neste trabalho. Como será visto mais à frente, alguns algoritmos de representação esparsa possuem uma etapa de ortogonalização, que é custosa computacionalmente. Devido à característica mencionada do dicionário utilizado, o desempenho desses algoritmos com e sem essa etapa não são muito diferentes, possibilitando assim a sua utilização na forma mais simples e menos custosa.

Um dos fatores que pesaram na escolha desse dicionário foi o seu tamanho reduzido, já que o número de elementos no dicionário influi diretamente no tempo de busca e, para uma operação online, é necessária uma busca rápida. Para os testes que serão mostrados nas seções subsequentes, o dicionário possui a seguinte composição: 100 elementos da base de Fourier, ou seja, 50 senóides e 50 cossenóides nas frequências harmônicas múltiplas de 60 Hz; e 128 funções de base wavelet, construídas a partir da wavelet mãe Daubechies, formando assim um dicionário com 228 elementos com 128 amostras cada.

3.2 ALGORITMOS PARA REPRESENTAÇÃO ESPARSA

Uma maneira de se encontrar a solução para os problema descrito em (2.8) é através de busca exaustiva. Porém, essa solução se mostra inviável em quase todas as situações. Por exemplo, supondo que a matriz **A** possua **n** linhas e **m** colunas, e que o vetor **b** seja uma combinação linear de no máximo elementos dessa matriz, é necessário enumerar todas as possíveis combinações de k_0 elementos de **A** e testá-las.

Para solucionar esse problema, diversos algoritmos têm sido desenvolvidos. Esses algoritmos podem ser classificados em três principais classes: i) os algoritmos gulosos, que têm como o objetivo minimizar as funções (P_0) ou (P_0,ε); ii) os algoritmos que relaxam a norma l_0 e têm como objetivo minimizar as funções (P_1) ou (P_1,ε); e iii) os algoritmos iterativos de redução (*Iterative Shrinkage Algorithm*) que minimizam uma forma mais geral.

Neste trabalho não foram utilizados algoritmos pertencentes à classe ii, devido ao seu elevado custo computacional, o que inviabiliza sua aplicação online. Os algoritmos pertencentes às outras classes, que serão utilizados, estão descritos nas próximas seções.

3.2.1 ALGORITMOS GULOSOS

Um algoritmo guloso abandona a busca exaustiva e trabalha com uma série de atualizações termo a termo, iniciando com a aproximação $\mathbf{x_0} = \mathbf{0}$ e adicionando elementos iterativamente para construir uma aproximação $\mathbf{x^k}$ contendo k elementos não nulos. Um algoritmo dessa classe, largamente utilizado em processamento de sinais, é o *Matching Pursuit* (MP) descrito na próxima seção.

3.2.2 MATCHING PURSUIT - MP

O MP é um algoritmo guloso em que a cada iteração seleciona o elemento do dicionário que possui maior projeção ortogonal no sinal residual (assumindo que os elementos do dicionário estejam normalizados). O elemento selecionado é adicionado ao suporte da solução e o coeficiente de todos os elementos do vetor é atualizado via Regressão Linear. Uma descrição deste algoritmo está mostrada na Figura 1.

O algoritmo funciona da seguinte maneira: no estágio de varredura, os produtos internos entre o resíduo $\mathbf{r^{k-1}}$ e cada elemento (coluna) $\mathbf{a_j}$ da matriz $\mathbf{A}$ são calculados; no estágio de atualização do suporte, o índice j_0 do elemento $\mathbf{a_j}$ que apresentou maior valor de produto interno é adicionado ao suporte; a atualização da solução provisória, os coeficientes que já faziam parte do suporte S^{k-1} são mantidos inalterados e o novo coeficiente, referente à j_0 é escolhido como sendo $\mathbf{z}$, que é o valor do produto interno entre o elemento j_0 e a matriz $\mathbf{A}$ e o resíduo $\mathbf{r^{k-1}}$; por fim o novo resíduo $\mathbf{r^k}$ é calculado e o critério de parada é avaliado.

$$\mathbf{A}'_{SK}(\mathbf{A}_{SK}\mathbf{x}_{SK} - \mathbf{b}) = -\mathbf{A}'_{SK}\mathbf{r}^k = 0 \tag{3.1}$$

em que $\mathbf{r^k = b - Ax^k}$. A equação (3.1) mostra que as colunas da matriz A que fazem parte do suporte S^k são necessariamente ortogonais ao resíduo $\mathbf{r^k}$, e portanto, não serão escolhidas novamente nas futuras iterações.

Objetivo: Aproximar a solução de $(P_0) : \min_{\mathbf{x}} \text{sujeito a } \mathbf{Ax} = \mathbf{b}$

Inicialização:

- Número de elementos do vetor $\mathbf{x}$: $k = 0$
- Solução inicial: $\mathbf{x}^0 = \mathbf{0}$
- Resíduo inicial: $\mathbf{r}^0 = \mathbf{b} - \mathbf{Ax}^0 = \mathbf{b}$
- Suporte da solução inicial: $S^0 = Suporte\{\mathbf{x}^0\} = \emptyset$

Iteração principal: $k = k + 1$ e em seguida realizar os seguintes estágios:

- **Varredura:** Calcular os erros $\epsilon(j) = \min_{z_j} \left\|\mathbf{a}_j z_j - \mathbf{r}^{k-1}\right\|_2^2$ utilizando o valor ótimo $z_j^* = \mathbf{a_j' r}^{k-1} / \left\|\mathbf{a}_j\right\|_2^2$
- **Atualização do suporte:** Encontrar o índice j_0 relativo tal que $\epsilon(j_0) \leq \epsilon(j)$ e atualizar o suporte $S^k = S^{k-1} \cup j_0$
- **Atualização da solução provisória:** Fazer $\mathbf{x}^k = \mathbf{x}^{k-1}$ e atualizar o elemento $x^k(j_0) = x^k(j_0) + z_j^*$
- **Atualização do resíduo:** Calcular $\mathbf{r}^k = \mathbf{b} - \mathbf{Ax}^k$
- **Critério de Parada:** Se $\left\|\mathbf{r}^k\right\|_2 \geq \epsilon_0$, realizar outra iteração.

Saída: Apos k iterações tem-se o vetor de coeficientes $\mathbf{x^k}$

Figura 1: MP - *Matching Pursuit*.

4 IMPLEMENTAÇÃO EM FPGA DO ALGORITMO DE COMPRESSÃO DE SINAIS

No presente capítulo será apresentado de forma detalhada como foi implementado no FPGA o algoritmo de compressão de sinais utilizando representação esparsa em dicionários redundantes, abordando as funções de cada bloco lógico utilizado, detalhando sua lógica interna bem como suas entradas e saídas.

Ainda será discutido os códigos implementados em linguagem de programação C no processador embarcado na FPGA e os *hardware* implementados em Verilog.

4.1 FERRAMENTAS

Antes de iniciar de fato a apresentação da implementação do projeto é importante mencionar os programas que auxiliaram na construção do sistema de compressão.

A primeira ferramenta a ser destacada é o *software* Quartus II. Esse *software* desenvolvido pela Altera permite que através de uma linguagem descritiva de Hardware (HDL), sejam sintetizados circuitos digitais para as mais diversas aplicações. No caso deste trabalho, a linguagem escolhida foi o Verilog, pois se trata de uma linguagem simples e de fácil implementação.

Em conjunto com o Quartus, o *software* ModelSim foi utilizado para a simulação do circuito programado. Através de um *testbench*, as entradas do circuito foram estimuladas e as saídas observadas, possibilitando fazer uma análise do comportamento do circuito programado.

Ainda, foram utilizadas os *software* Flex e Bison em conjunto com o CodeBlocks, são responsáveis por fazer a interpretação da linguagem C desenvolvida. Essa ferramentas auxiliam no desenvolvimento dos compiladores C e Assembler que foram desenvolvidos para gerar os códigos de máquina a ser executado pelo processador embarco na FPGA.

Neste trabalho os processadores implementados no FPGA, são baseados na arqui-

tetura *Reduced Instruction Set Computer* (RISC) e possui memórias separadas para dados e instruções (Arquitetura Harvard). Apenas recursos internos do FPGA são utilizados, como memórias e blocos de processador digital de sinais (do inglês, *Digital Signal Processor*) (DSP). Além disso, a Unidade Lógica Aritmética (ALU) de cada processador contém somente os recursos necessários para o algoritmo nele implementado. A ALU utiliza de aritmética de ponto flutuante, permitindo, simultaneamente, rápido desenvolvimento de *software* (não é necessário se preocupar com quantização) e resultados precisos.

4.2 DIVISÃO DO ALGORITMO DE COMPRESSÃO DE SINAIS

Nesta seção será apresentada como foi feita a divisão do algoritmo compressão de sinais, detalhando a função de cada processador presente no sistema.

Tendo em vista que o objetivo é executar o algoritmo MP em tempo real, o sistema proposto será divido em três processadores principais, cada um contendo uma função especifica no algoritmo de representação esparsa. Cada bloco possui um processador embarcado, sendo assim foram desenvolvidos três códigos diferentes. O primeiro bloco é responsável por executar o procedimento de FFT no sinal de entrada e ordenar o vetor gerado na saída. O segundo bloco lógico desenvolvido é responsável por realizar o produto interno entre o sinal de entrada e as Wavelets presentes na matriz dicionário. Esses dois processadores implementam o estágio de varredura do algoritmo, foram utilizados dois processadores em paralelo visando a diminuição do tempo necessário para esse estágio.

O terceiro e último processador recebe as informações advindas dos blocos anteriores, e a partir dos dados de entrada os organiza em um vetor. Depois de receber todos os componentes da entrada, ele percorre o vetor em busca de seu maior valor em módulo. Após identificar o máximo elemento do vetor, o algoritmo armazena este valor, juntamente com o valor de sua posição (índice do vetor). Com estas informações armazenadas o código identifica qual sinal do dicionário deve gerar para reconstruir o sinal de entrada. O terceiro processador também analisa o resíduo do sinal de entrada, verificando a finalização do processo de compressão. A Figura 2, ilustra o diagrama do sistema proposto.

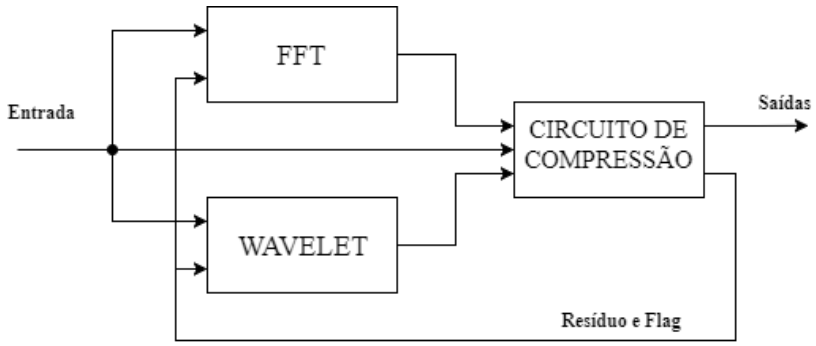

Figura 2: Diagrama de blocos do sistema de compressão proposto.

4.3 TRANSFORMADA RÁPIDA DE FOURIER (FFT)

A Transformada rápida de Fourier FFT é um algoritmo computacional capaz de realizar a Transformada Discreta de Fourier, (do inglês, *Discret Fourier Transform*) (DFT) de forma mais rápida, ou seja, realizando um menor número de operações.

A DFT é uma ferramenta matemática capaz de decompor um função no domínio do tempo em seus componentes de frequência, ou seja, é uma representação no domínio da frequência do sinal original. O caminho inverso pode ser feito através da Transformada discreta de Transformada discreta de Fourier Inversa, (do inglês, *Inverse Discret Fourier Transform*) (IDFT), ou ainda na sua forma rápida, a Transformada Rápida de Fourier Inversa , (do inglês, *Inverse Fast Fourier Transform*) (IFFT). Um exemplo de uso da DFT é na analise de harmônicos da rede elétrica, sendo este um dos parâmetros para a medição da qualidade de energia.

A DFT e a FFT produzem o mesmo resultado, no entanto a segunda é realizada de uma forma mais rápida. Por definição, a DFT é obtida através da Equação (4.1).

$$X_k = \sum_{n=0}^{N-1} x[n] e^{-j2\pi k \frac{n}{N}} \tag{4.1}$$

Como pode-se observar, a DFT é obtida decompondo uma sequência de valores em componentes de diferentes frequências.

Para compactar o sinal é necessário encontrar a projeção do sinal de entrada com as colunas do dicionário. Nas 100 primeiras colunas da matriz dicionário estão presentes os componentes de Fourier, ou seja, 50 componentes harmônicos de cosseno (do harmônico fundamental até o quinquagésimo harmônico) e 50 componentes harmônicos de seno, respectivamente.

Aplicando a FFT, pode-se encontrar as componentes de frequência de um sinal onde a parte real representa as componentes cossenoidais e a parte imaginaria representa as componentes senoidais. Portanto neste projeto pare encontrar a relação entre um sinal de entrada e as 100 primeiras posições da matriz dicionário, será utilizado o algoritmo da FFT.

4.3.1 IMPLEMENTAÇÃO EM FPGA DA FFT

A implementação do algoritmo da transformada rápida de Fourier em FPGA, foi feita utilizando a linguagem de descrição de hardware Verilog. O bloco principal do sistema é um processador embarcado, esse processador também é construído utilizando a linguagem Verilog e sua proposição encontra-se em (KAPISCH, 2015). O sistema completo na forma de diagrama de blocos é apresentado na Figura 3.

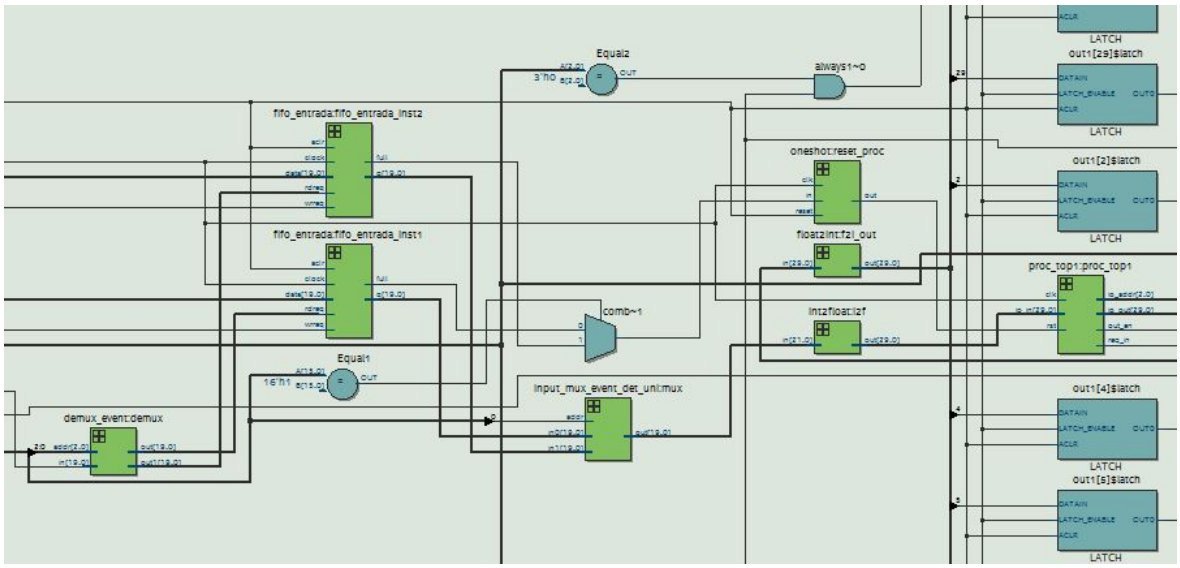

Figura 3: Diagrama de blocos do sistema responsável no calculo da FFT.

4.3.1.1 PROCESSADOR EMBARCADO

No processador embarcado é realizado o algoritmo da FFT. Para implementar este algoritmo utilizou-se um código em C da estrutura da *Butterfly*. O nome desta estrutura se origina do fato de como a estrutura é desenhada, ilustrado na Figura 4. A *butterfly* é comumente dividida em dois métodos diferentes: *radix*-2 e *radix*-4. No presente trabalho, apenas o primeiro método é abordado.

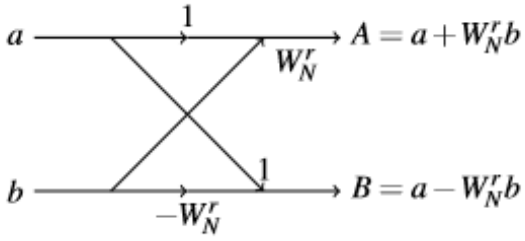

Figura 4: *Butterfly.*

O método da butterfly consiste basicamente em dividir um vetor de tamanho N até que ele possua apenas dois elementos. Obviamente, esse vetor deve ser uma potência de dois. Caso não seja, a operação de *zero-padding* deve ser feita, a qual consiste em adicionar zeros no final do vetor ate que seu tamanho seja uma potência de dois.

Para realizar a implementação do algoritmo, a primeira etapa a ser executada é uma ordenação dos dados de entrada. No entanto, antes de realizar a ordenação, é importante separar a parte real dos dados de entrada da parte imaginária. A parte real deve ser inserida nas posições pares do vetor de entrada e a parte imaginária nas posições ímpares do vetor, sendo assim, gerando um vetor de tamanho *2N* para uma FFT de tamanho *N*.

A ordenação dos dados deve ser feita da seguinte forma, o primeiro dado do vetor de tamanho N deve receber o valor zero em binário, o segundo deve receber um e assim por diante até o último, que receberá *N-1* em binário. Feito isto, cada posição do vetor de entrada tem um valor binário associado. Os valores binários devem ser lidos de trás para frente e reordenados de forma crescente, ou seja, o *Most Significant Bit* (MSB) deve ser lido como o *Least Significant Bit* (LSB) e vice-versa. Deve-se atentar que neste ponto tem-se um vetor de tamanho $2N$ e não N, já que a parte real e imaginária estão separadas. Sendo assim, a ordenação deve ser feita cuidadosamente para que tanto a parte real, quanto a imaginária sejam reordenadas devidamente.

Apenas após a ordenação, os dados estão prontos para serem inseridos nas *Butterflies*. Esta etapa será composta por DFTs de dois elementos. Calculado, a etapa seguinte será constituída de DFTs composta pela combinação dos resultados das DFTs de dois elementos. E assim por diante, até todas as etapas serem concluídas, obtendo assim um vetor de tamanho N, no qual em cada posição haverá um elemento da DFT obtida. O código que implementa o algoritmo da FFT em C é apresentado pela Figura 5.

```
n = nn*2.0;
j = 1;
i = 1;

mmax = 2;
ind = 0;
while (mmax < n)
{
    istep = mmax*2;
    wtemp = sin1[ind];
    wpr   = -2.0*wtemp*wtemp;
    wpi   = sin2[ind];
    wr    = 1.0;
    wi    = 0.0;
    ind   = ind+1;

    m = 1;
    while (m < mmax)
    {
        i = m;
        while(n >= i)
        {
            j = i + mmax;

            tempr     = wr*data[j-1]   - wi*data[j];
            tempi     = wr*data[j] + wi*data[j-1];

            data[j-1]   = data[i-1]   - tempr;
            data[j+1-1] = data[i] - tempi;
            data[i-1]   = data[i-1]   + tempr;
            data[i+1-1] = data[i] + tempi;

            i = i + istep;
        }

        wtemp = wr;
        wr = wtemp*wpr - wi*wpi + wr;
        wi = wi*wpr + wtemp*wpi + wi;

        m = m+2;
    }

    mmax = istep;
}
```

Figura 5: Código em C da *Butterfly*.

Após a execução de todas as iterações necessárias, a FFT é concluída e os dados de saída podem ser interpretados. As duas primeiras posições do vetor de saída de tamanho $2N$ serão os dados reais e imaginários da componente de frequência zero, respectivamente. As duas posições seguintes serão compostas pelos dados da menor frequência positiva e a frequência continuará aumentando conforme as posições do vetor sejam percorridas.

Para exportar os resultados de projeção do sinal de entrada com a matriz dicionário, o vetor de saída do algoritmo da FFT é ordenado da seguinte forma: as 50 primeiras posições do vetor de saída são compostas pelas 50 primeiras posições pares diferentes de zero, do vetor ordenado pela FFT, ou seja as componentes reais, que representam as componentes harmônicos cossenoidais. Da quinquagésima posição até a centésima posição do vetor de saída, é composto pelas 50 primeiras posições impares do vetor ordenado pela FFT, ou seja as componentes imaginários, que representam as componentes harmônicos senoidais.

4.3.1.2 MEMÓRIAS

Conforme apresentado na Figura 3, o sistema utiliza duas memórias do tipo *First In First Out* (FIFO), como o próprio nome diz, estas memórias funciona de modo que o primeiro dado escrito, será o primeiro dado a ser lido, funcionando como um *buffer* para o sinal de entrada. Nesse projeto utilizou-se uma FIFO de 128 posições, já que esse é o tamanho da janela a ser utilizada na FFT, com 16 bits em cada posição, visto que é o valor de quantização do sinal. A primeira FIFO recebe o sinal de entrada do sistema, ou seja, o a ser compactado. A segunda FIFO recebe o sinal de resíduo do algoritmo de compressão de sinais.

Para realiza uma escrita de dados nessa FIFO, é necessário que o sinal *wrreq* seja nível lógico alto durante a borda de subida do *clock*, esse sinal foi gerado a cada período de amostragem, e sendo assim, as amostras do sinal são escritas na FIFO durante o processo da amostragem. Para se iniciar o cálculo da FFT, é necessário que a janela de 128 pontos esteja preenchida, ou seja, que a FIFO esteja completa. A FIFO possui um sinal de saída que é o *full*, esse sinal vai para nível lógico alto quando a FIFO está completa.

4.3.1.3 DEMAIS BLOCOS

Para realizar o controle das FIFOS do circuito de compressão propostos utiliza-se de um Multiplexador (Mux) e um Demultiplexador (Demux). O Mux tem função de controlar em qual FIFO o sinal de entrada será escrito. Para controlar a porta de saída, o Mux recebe em sua entrada de endereço($ADDR$) um sinal de controle advindo do terceiro processador. O Demux tem função de controlar em qual FIFO será lida pelo processador. Assim como o Mux, o acDemux também recebe em sua entrada de endereço($ADDR$) um sinal de controle advindo do terceiro processador.

O bloco de *Oneshot* é uma maquina de estados, controlada pelas saídas *full* das FIFOs, a função deste bloco é controlar o *reset* do processador. O bloco *floattoint* tem a função de converter os dados do tipo *float*(número com ponto flutuante) para int(número inteiro). O bloco *inttofloat* tem a função contraria ao bloco *floattoint*.

4.4 PRODUTO INTERNO COM AS WAVELETS

Como dito anteriormente o dicionário utilizado neste trabalho é composto por componentes de Fourier e Wavelet. Nesta seção será apresentado como foi calculado a

projeção ortogonal dos 128 coeficientes de Wavelet. Para implementar essa operação utilizou-se o mesmo circuito do algoritmo da FFT, ou seja, utilizou o circuito apresentado na Figura 3. Porém implementando um código em C diferente no processador embarcado.

O dicionário é formado por seis wavelets fundamentais, deslocadas. Essas wavelet são mostradas na Figura 6. As características de cada tipo de wavelet e suas posições no dicionário são apresentadas na Tabela 1.

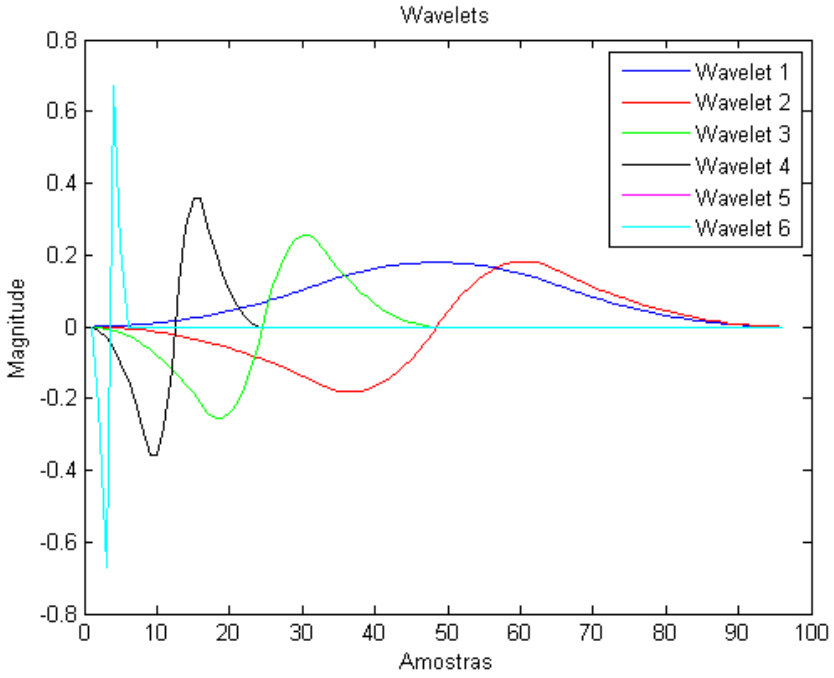

Figura 6: Wavelets fundamentais do dicionário.

Tabela 1: Wavelet presentes no dicionário

Wavelet	Número de pontos	Número de deslocamentos	Posição no dicionário
Wavelet 1	96	4	100 até 103
Wavelet 2	96	4	104 até 107
Wavelet 3	48	8	108 até 115
Wavelet 4	24	16	116 até 131
Wavelet 5	12	32	132 até 163
Wavelet 6	6	64	164 até 227

Na Figura 7 é apresentado uma parte do código C responsável por realizar o produto interno da Wavelet 1 com o sinal de entrada. Onde i é o número de deslocamentos , j

e k são auxiliares para percorrer os vetores, o vetor *data* é o sinal de entrada, *data1* é o vetor contendo os valores da primeira wavelet e *prod2* recebe o resultado do produto. Na saída do algoritmo os vetor contendo os resultados de produto interno de cada wavelet são organizados conforme a posição do dicionário.

```
//produto interno da primeira wavelet
k = 0;
j = 0;
i = 0;

while(i < 4)
{
    while(j < 96)
    {
        if(k < 128)
        {
            prod1[i] = prod1[i] + data1[j]*data[k];
            j = j + 1;
            k = k + 1;
        }
        else
        {
            k = 0;
        }
    }
    i = i + 1;
    j = 0;
}
```

Figura 7: Código em C responsável por realizar o produto interno da Wavelet 1.

4.5 CIRCUITO DE COMPRESSÃO

Como descrito na seção 4.1, o projeto foi dividido em três processadores. Nesta seção abordaremos o circuito implementado em FPGA responsável por compactar o sinal de entrada, a partir dos dados de entrada. O circuito desenvolvido em linguagem de descrição de *Hardware* é apresentado na Figura 8.

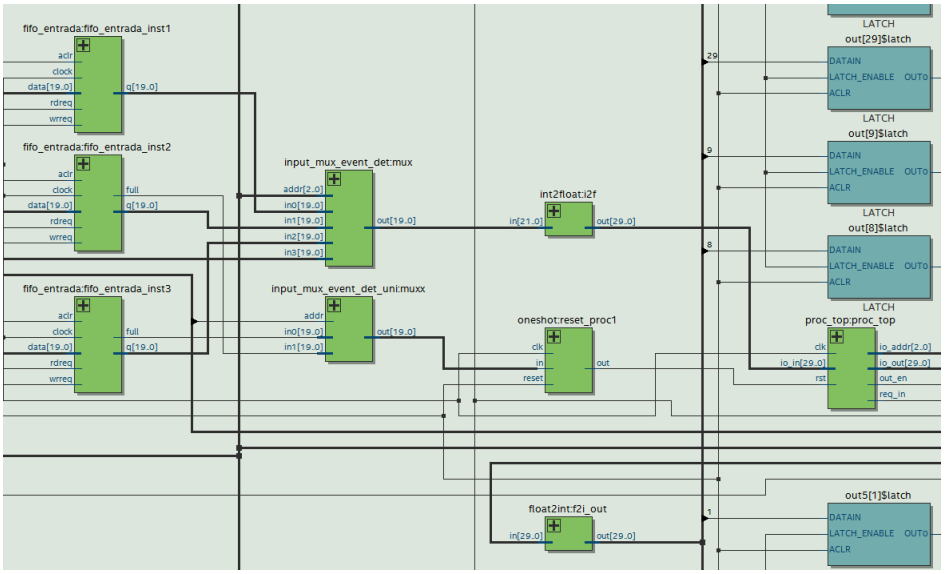

Figura 8: Diagrama de blocos do sistema de compressão de sinais.

Ao observa a Figura 8, percebe-se a presença dos mesmos blocos da Figura 3, todos os possuem as mesmas funções descritas na seção 4.3, porém para este circuito é necessário a presença de três FIFOS. A primeira FIFO recebe a saída do algoritmo da FFT, a segunda FIFO recebe o resultado do produto interno das wavelets e a terceira FIFO recebe o sinal de entrada.

4.5.1 ALGORITMO IMPLEMENTADO NO PROCESSADOR DE CIRCUITO DE COMPRESSÃO

Nesta seção será descrito todas as etapas desenvolvidas no sistema de compressão.

A primeira etapa do algoritmo é iniciar a saída de controle como zero ($Flag = 0$). Após esta etapa o algoritmo armazena o sinal de entrada em um vetor para realizar operações futuras. Nesta etapa o vetor responsável por armazenar resíduo é zerado, para que não ocorra erros nas iterações seguintes. Depois de realizar a operações descritas acima o valor da saída de controle é modificado para 1 ($Flag = 1$), após isto o algoritmo começa a armazenar os dados recebidos das duas primeiras FIFOs em um vetor, conforme apresentado na Figura 9.

```
//Flag ----------------------------------------------------------------
    flag = in(3);
    //out(7,33);
    //out(7,flag);
// Sinal de Entrada ----------------------------------------------------
 if(flag == 0)
 {
    ord = 0;
    pos = 0;
    maior = 0;
    i = 0;
    // Sinal de entrada
    while (i < 128)
    {
        data3[i] = in(2);
        i = i + 1;
    }

    i = 0;
    while (i < 128)
    {
        data_res[i] = data3[i];
        data_out1[i] = 0;
        i = i + 1;
    }
```

Figura 9: Algoritmo responsável por organizar as entradas.

Com as 228 posições do vetor dos coeficientes preenchidas, o algoritmo calcula o módulo deste vetor. Com o vetor de módulo calculado o algoritmo varre o vetor em busca de seu maior valor em modulo. Após localizar valor máximo, o algoritmo armazena o seu valor original juntamente com o índice de sua respectiva posição no suporte. A Figura 10 mostra como é programada em C esta etapa do projeto.

```
//Modulo ------------------------------------------------------
        i = 0;
    while (i < 228)
    {
        modulo[i] = data[i]*data[i];
        i = i + 1;
    }
//algoritmo de ordem-------------------------------------------
k = 0;
maior = 0;
    while(k < 228)
    {
        if(maior < modulo[k])//
        {
           ord = data[k];
           maior = modulo[k];
           pos = k;
        }
        k = k + 1;
    }
    if(pos < 100)
    {
           pos = pos - 1;
    }
    else
    {
        pos = pos;
    }
```

Figura 10: Algoritmo por calcular o módulo do vetor e encontrar o máximo.

Com o suporte atualizado o algoritmo identifica qual sinal deve gerar para atualizar o vetor de resíduo e o sinal reconstruído. Para realizar esta operação programou-se diversas funções que representam os componentes de seno e cosseno e cada uma das wavelets descritas na seção 4.3.

Com o resíduo calculado, o algoritmo calcula a energia do resíduo, e verifica se a energia é menor que o limiar estabelecido. Caso a energia for menor que o limiar a variável Flag volta para zero e o algoritmo espera a FIFO com o sinal de entrada ser preenchido para reiniciar o sistema de compressão. Caso a energia for maior que o limiar, o resíduo é enviado para a saída e o algoritmo irá para a próxima iteração. A energia é calculado conforme a Equação (4.2).

$$Energia = \sum_{n=0}^{N-1} residuo^{2}[n] \tag{4.2}$$

4.6 IMPLEMENTAÇÃO EM TEMPO REAL

Nesta seção será realizada uma analise da implementação em tempo real do algoritmo de compressão de sinais e as características desejáveis neste sistema.

A implementação em tempo real de um sistema de compressão de sinais levanta algumas questões importantes, como custo computacional, Hardware adequada e adaptar o sistema para o processamento em tempo real. A questão do custo computacional é avaliada como a capacidade que o sistema tem para realizar os processamentos necessários dentro de um intervalo de tempo pré-determinado. Esse intervalo de tempo está associado ao período de atualização de cada amostra do sinal de entrada.

O sistema proposto possui uma frequência de amostragem de 7680Hz (128 pontos por ciclo de 60Hz), enquanto que a frequência interna do *clock* da FPGA para os circuitos de processamento, pode ser configurada para até 200MHz. Isso permite que a FPGA tenha 26041 ciclos de *clock* para realizar todos os processos de cada amostra.

Para avaliar a implementação em tempo real aplicou-se no sistema um sinal de *clock*, com período fixo de *1ps*. Além disso o sistema armazena um dado de entrada nas FIFOs a cada 6000 ciclos de *clock*. Com esses dados, montou-se a Tabela 2, que apresenta o número de *clocks* necessários para operação da cada processador.

Tabela 2: Relação entre o número de *clocks* para execução de cada processador

Processador	Número de *Clocks*
FFT	188902
Wavelet	219514
Processador de compressão	1045795

A partir dos dados da tabela verificou-se que para uma implementação em tempo real é necessário no minimo 1045795 ciclos de *clock*, ou seja 8170 *clocks* para cada amostra. Para a compressão de um sinal elétrico com frequência fundamental de 60Hz o sistema precisa operar em uma frequência próxima de 16 MHz. Como é possível operar o sistema com uma frequência maior que 16MHz, pode-se afirmar que o sistema desenvolvido é capaz de ser implementado em tempo real em FPGA. A Figura 11, apresenta o sistema operando em tempo real no *software* Modelsim.

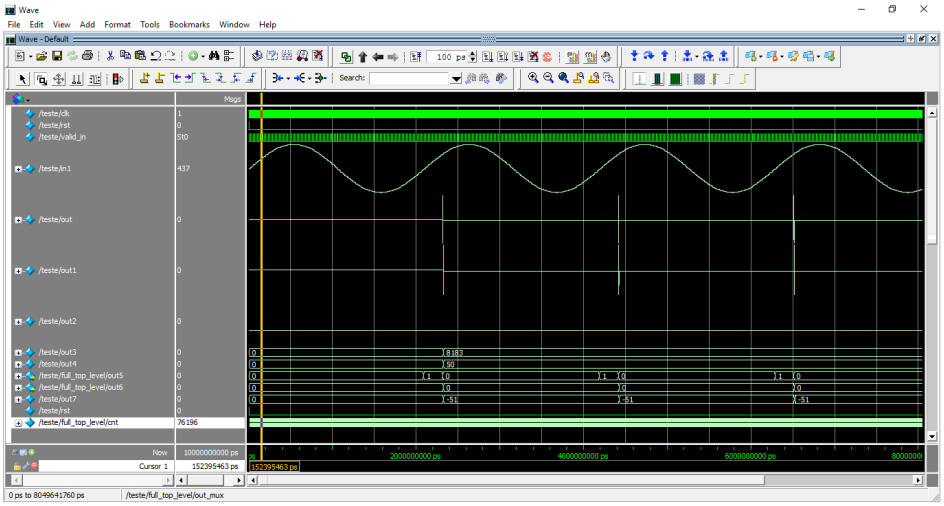

Figura 11: Algoritmo simulando em tempo real no *software* Modelsim

5 RESULTADOS

Neste capítulo serão apresentados os resultados do sistema de compressão proposto, obtidos através da implementação em FPGA. Os mesmos serão comparados com os resultados obtidos via algoritmos implementos no *Software* Matlab.

Para a validação e comparação do desempenho do Sistema de Compressão proposto foram gerados diversos sinais com duração de um ciclo da componente fundamental. Cada sinal de teste é composto por um sinal senoidal com a frequência fundamental de 60 Hz, com 128 pontos por ciclo da componente fundamental, frequência de amostragem igual a 7680Hz. Os sinal intercalados para os testes são:

S1: Sinal formado por componentes harmônicos de cosseno.

S2: Sinal formado por componentes harmônicos de seno.

S3: Sinal com fase diferente de zero.

S4: Sinal com presença de componentes Interharmônicos.

S5: Sinal com afundamento na amplitude.

S6: Sinal com elevação na amplitude.

S7: Sinal com variação exponencial na amplitude de uma componente do sinal.

Algumas métricas foram escolhidas para avaliar a qualidade do sinal reconstruído após a descompressão. São elas: o Erro Médio Quadrático Normalizado, (do inglês, *Normalized Mean Squared Error*) (NMSE), definido em (5.1); a Correlação Cruzada (COR) entre o sinal reconstruído e o sinal original, definida em (5.2); e a Porcentagem de energia mantida (RTE), definida em (5.3)

$$\mathbf{NMSE} = \frac{\|\mathbf{x} - \hat{\mathbf{x}}\|^{\mathbf{2}}}{\|\mathbf{x}\|^{\mathbf{2}}} \tag{5.1}$$

$$\mathbf{COR} = \frac{\mathbf{x^T}.\hat{\mathbf{x}}}{\mathbf{x^T}.\mathbf{x}} \tag{5.2}$$

$$\mathbf{RTE} = \frac{\sum_{n=0}^{N} x[n]^2}{\sum_{n=0}^{N} \hat{x}[n]^2} \tag{5.3}$$

em que, $\mathbf{x}$ representa a notação vetorial para o sinal original $x[n]$, $\hat{\mathbf{x}}$ é a notação vetorial do sinal reconstruído $\hat{x}[n]$ e N é o comprimento do sinal.

Em se tratando de uma aplicação de compressão de sinais, o objetivo é encontrar uma aproximação que seja fiel e ao mesmo tempo compacta, ou seja, que utilize poucos elementos do dicionário. Portanto em conjunto com a qualidade de aproximações, o número de elementos utilizados também deve ser avaliado.

5.1 SINAL FORMADO POR COMPONENTES HARMÔNICOS DE COSSENO

Para o primeiro teste foi escolhido aleatoriamente um sinal formado por uma soma de componentes harmônicas de cosseno, conforme descrito pela Equação (5.4), onde f representa a frequência do sinal, t o tempo e h o componente harmônico. Para este sinal o sistema retornou o número de componentes de Fourier e Wavelet descritos na Tabela 3.

$$x(t) = \sum_{h=1}^{3} \frac{1}{h} cos(2\pi f h t) \tag{5.4}$$

Tabela 3: Número de Componentes de Fourier e Wavelet no caso 1.

Coeficientes de Fourier	3
Coeficientes de Wavelet	0

O sinal reconstruído é mostrado na Figura 12, onde é comparado com o sinal original, e o sinal reconstruído pelo MATLAB. Na tabela 4 são apresentados os valores para as métricas definidas anteriormente.

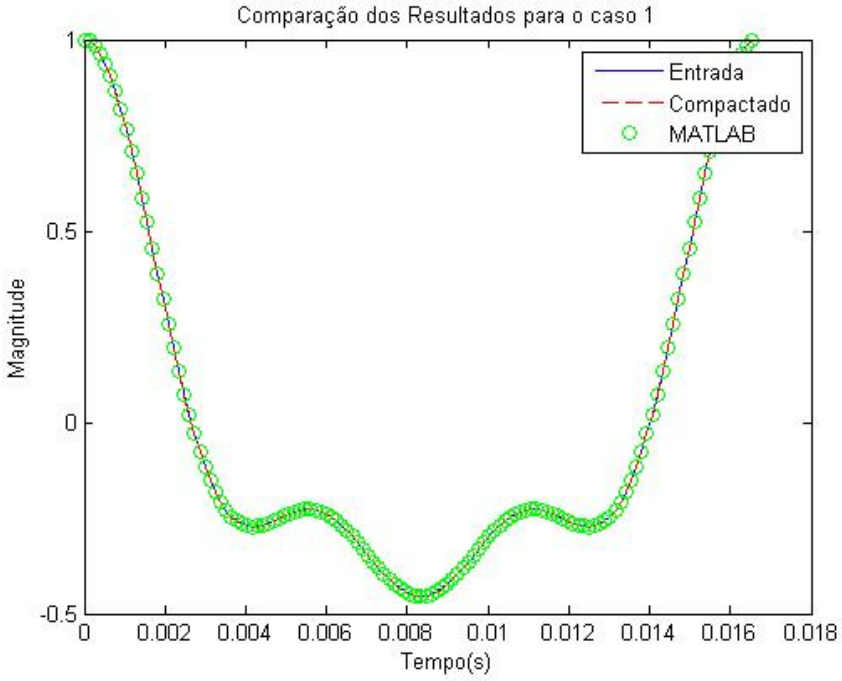

Figura 12: Comparação entre o sinal gerados para o caso 1.

Tabela 4: Resultados métricos para o primeiro caso.

Erro Médio Quadrático (NMSE)	0,0160%
Correlação cruzada (COR)	99,9837%
Porcentagem de energia mantida (RTE)	100%

Percebe-se que para este caso o Erro Médio quadrático é baixo e, que os valores de Correlação cruzada e porcentagem de energia mantida estão próximos de 100%, implicando em uma boa fidelidade na reconstrução do sinal de entrada pelo método.

O erro absoluto entre o sinal de entrada e o sinal reconstruído pelo sistema de compressão é mostrado conforme a Figura 13.

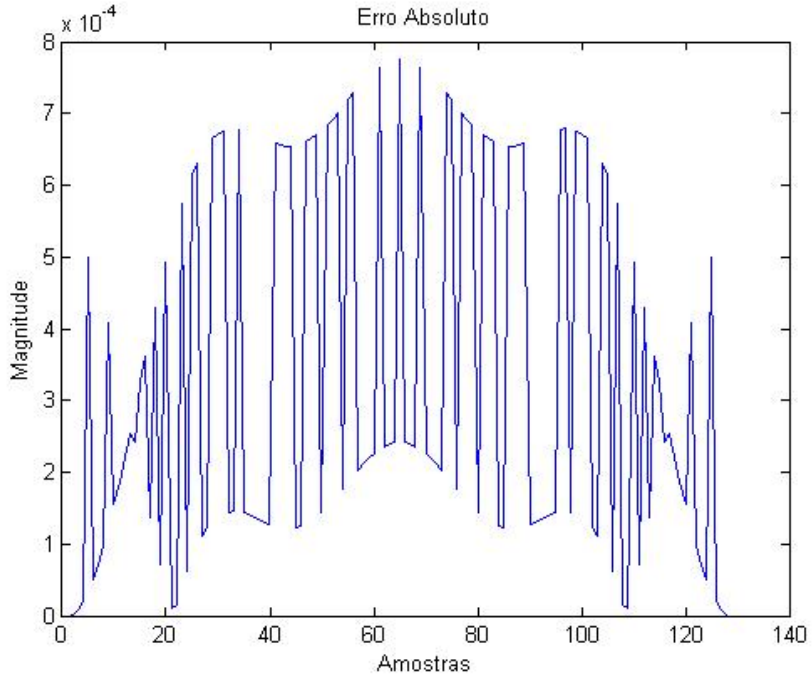

Figura 13: Erro Absoluto entre o sinal de entrada e o sinal reconstruído para o caso 1.

Pode-se perceber que para esta situação, o sinal reconstruído pelo algoritmo, tanto o sinal reconstruído pelo MATLAB, se assemelham perfeitamente com o sinal original, visto que o erro absoluto é muito pequeno (na faixa de 10^{-4}). Isso ocorre porque que o sinal de entrada possui poucas componentes de Fourier, sendo de fácil compressão pelo algoritmo.

5.2 SINAL FORMADO POR COMPONENTES HARMÔNICOS DE SENO

Para o segundo teste foi escolhido aleatoriamente um sinal formado por uma soma de componentes harmônicas de seno, conforme descrito pela Equação (5.5). Para este sinal o sistema retornou o número de componentes de Fourier e Wavelet descritos na Tabela 5.

$$x(t) = \sum_{h=1}^{4} \frac{1}{h} sin(2\pi f(2h-1)t) \tag{5.5}$$

O sinal reconstruído é mostrado na Figura 14, onde é comparado com o sinal original, e o sinal reconstruído pelo MATLAB. Na tabela 6 são apresentados os valores para as métricas definidas anteriormente.

Tabela 5: Número de Componentes de Fourier e Wavelet no caso 2.

Coeficientes de Fourier	4
Coeficientes de Wavelet	0

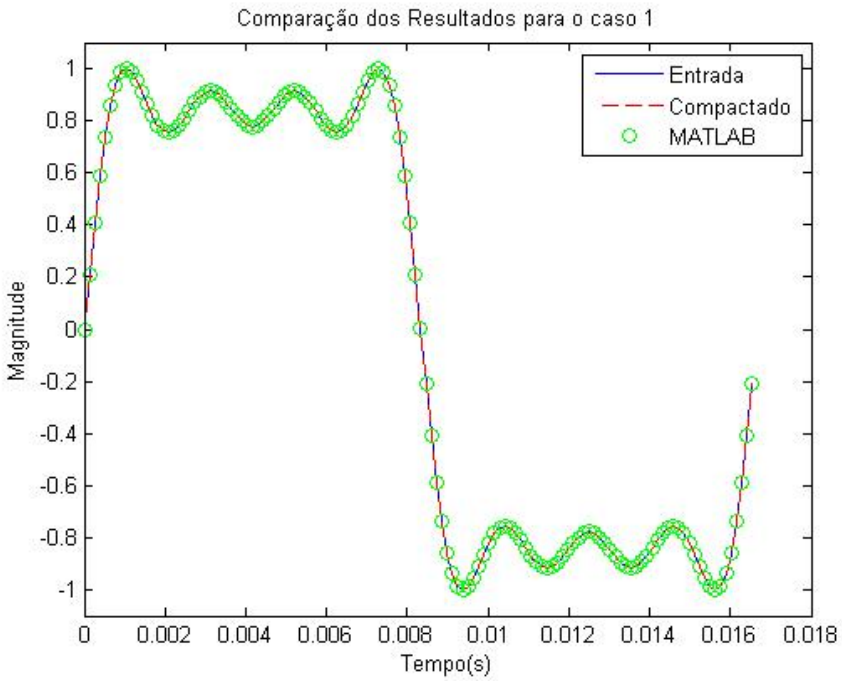

Figura 14: Comparação entre o sinal gerados para o caso 2.

Tabela 6: Resultados métricos para o segundo caso.

Erro Médio Quadrático (NMSE)	0,0098%
Correlação cruzada (COR)	99,9868%
Porcentagem de energia mantida (RTE)	100%

Percebe-se que para este caso o Erro Médio quadrático se manteve baixo, e menor que o caso anterior. Os valores de Correlação cruzada e porcentagem de energia mantida permaneceram próximos de 100%, implicando em uma boa fidelidade na reconstrução do sinal de entrada pelo método.

O erro absoluto entre o sinal de entrada e o sinal reconstruído pelo sistema de compressão é mostrado conforme a Figura 15.

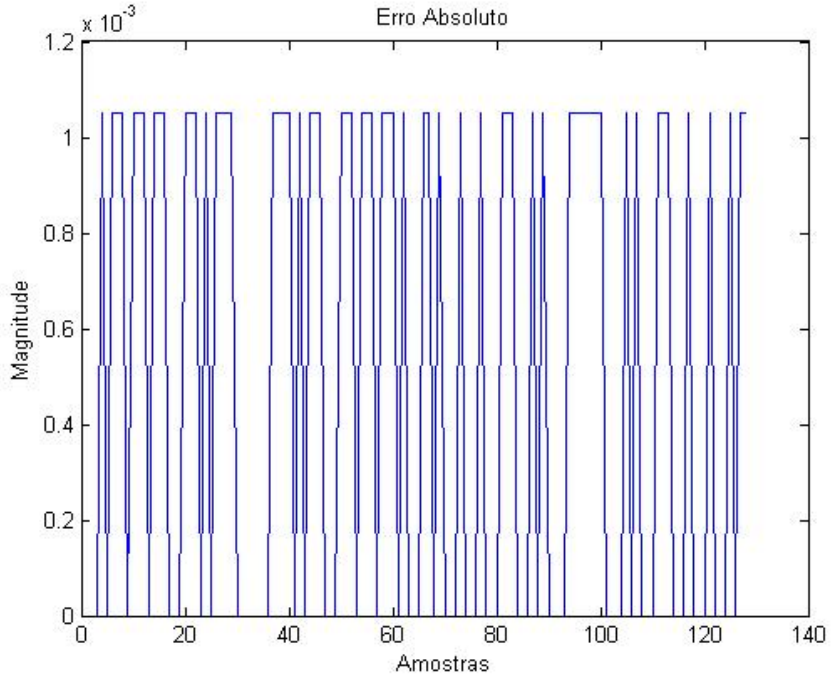

Figura 15: Erro Absoluto entre o sinal de entrada e o sinal reconstruído para o caso 2.

Pode-se perceber que para esta situação, o sinal reconstruído pelo algoritmo, tanto o quanto o sinal reconstruído pelo MATLAB, se assemelham perfeitamente com o sinal original, visto que o erro absoluto é muito pequeno (na faixa de 10^{-3}). Isso ocorre porque que o sinal de entrada possui poucas componentes de Fourier, sendo de fácil compressão pelo algoritmo.

5.3 SINAL COM FASE DIFERENTE DE ZERO

Para o terceiro teste escolheu-se um sinal que seria necessário utilizar tanto componentes de seno quanto de cosseno para comprimir o sinal, por esta razão gerou-se uma soma de componentes harmônicas de seno com fase, conforme descrito pela Equação (5.6). Para este sinal o sistema retornou o número de componentes de Fourier e Wavelet descritos na Tabela 7.

$$x(t) = sen(2\pi ft + \frac{\pi}{3}) + 0{,}5.sen(6\pi ft + \frac{\pi}{6}) + 0{,}2.sen(10\pi ft + \frac{\pi}{4}) \qquad (5.6)$$

O sinal reconstruído é mostrado na Figura 16, onde é comparado com o sinal original, e o sinal reconstruído pelo MATLAB. Na tabela 8 são apresentados os valores para as métricas definidas anteriormente.

Tabela 7: Número de Componentes de Fourier e Wavelet no caso 3.

Coeficientes de Fourier	6
Coeficientes de Wavelet	0

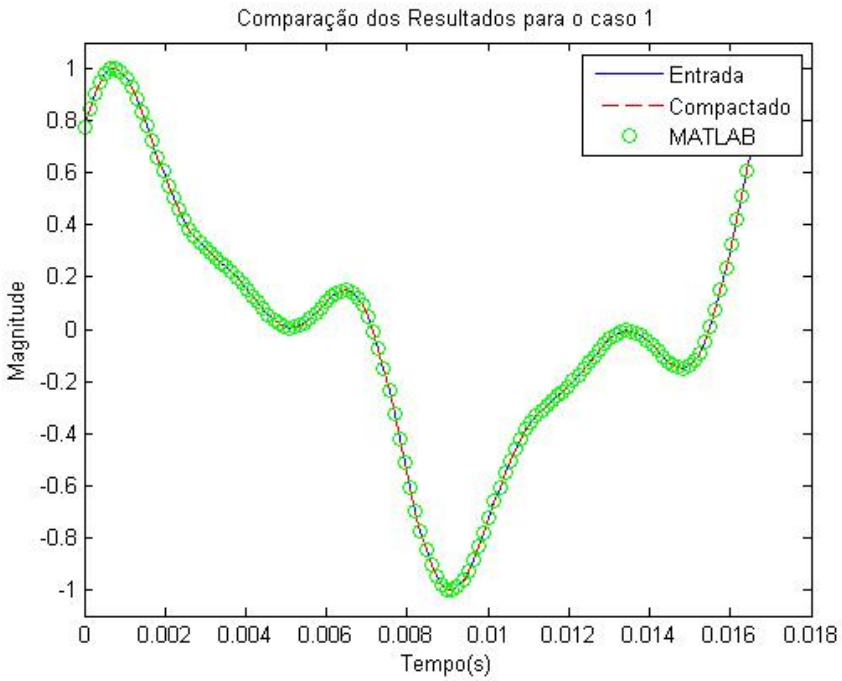

Figura 16: Comparação entre o sinal gerados para o caso 3.

Tabela 8: Resultados métricos para o terceiro caso.

Erro Médio Quadrático (NMSE)	0,1321%
Correlação cruzada (COR)	99,9857%
Porcentagem de energia mantida (RTE)	100%

Percebe-se que para este caso o Erro Médio quadrático se manteve baixo, porém com um maior erro em relação aos casos anteriores. Os valores de Correlação cruzada e porcentagem de energia mantida permaneceram próximos de 100%, implicando em uma boa fidelidade na reconstrução do sinal de entrada pelo método.

O erro absoluto entre o sinal de entrada e o sinal reconstruído pelo sistema de compressão é mostrado conforme a Figura 17.

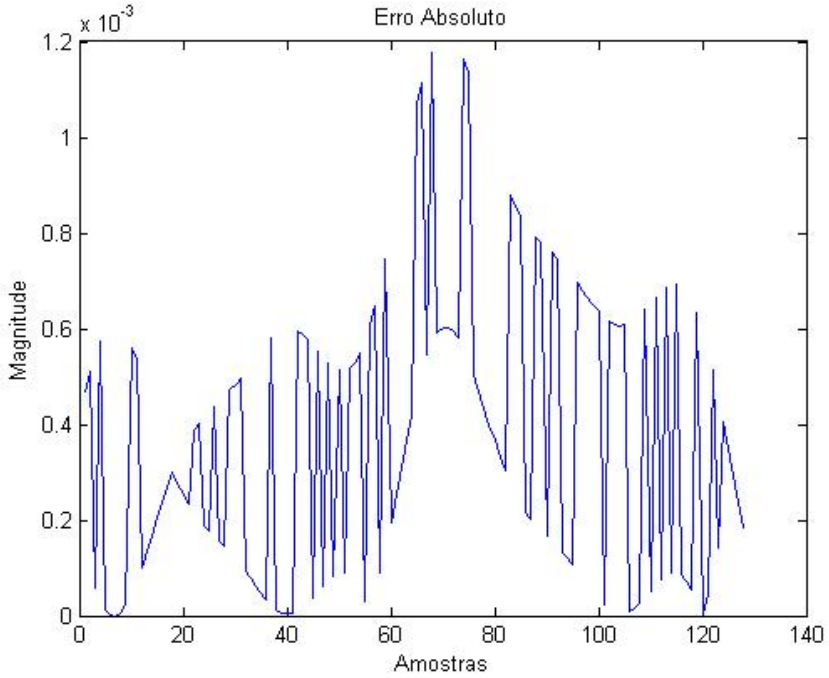

Figura 17: Erro Absoluto entre o sinal de entrada e o sinal reconstruído para o caso 3.

Pode-se perceber que para esta situação, o sinal reconstruído pelo algoritmo, tanto o quanto o sinal reconstruído pelo MATLAB, se assemelham ao sinal original, neste caso o erro absoluto ainda é muito pequeno (na faixa de 10^{-3}). Neste caso o algoritmo de compressão precisou utilizar mais componentes do que o número de harmônicos do sinal de entrada, isto ocorre por causa da fase ser diferente de zero. Para representar um sinal com fase diferente de zero e necessário utilizar tanto componentes de seno quanto de cosseno. Para este caso o sistema se mostrou eficiente no uso destes componentes, visto que os sinais encontrados são semelhantes.

5.4 SINAL COM INTERHARMÔNICO

O quarto teste é formado por dois componentes senoidais harmônicos, somados a um componente senoidal interharmônico, conforme descrito pela Equação (5.7). Para este sinal o sistema retornou o número de componentes de Fourier e Wavelet descritos na Tabela 9.

$$x(t) = sen(2\pi ft) + 0{,}1.sen(6\pi ft) + 0{,}05.sen(2.5{,}3.\pi ft) \tag{5.7}$$

Tabela 9: Número de Componentes de Fourier e Wavelet no caso 4.

Coeficientes de Fourier	6
Coeficientes de Wavelet	2

O sinal reconstruído é mostrado na Figura 18, onde é comparado com o sinal original, e o sinal reconstruído pelo MATLAB. Na tabela 10 são apresentados os valores para as métricas definidas anteriormente.

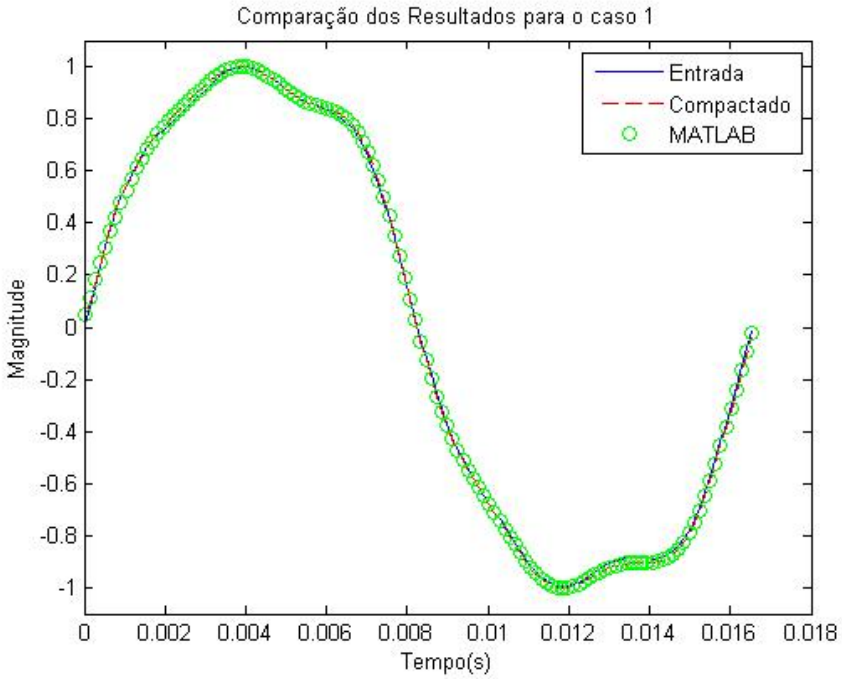

Figura 18: Comparação entre o sinal gerados para o caso 4.

Tabela 10: Resultados métricos para o quarto caso.

Erro Médio Quadrático (NMSE)	1,0148%
Correlação cruzada (COR)	99,6321%
Porcentagem de energia mantida (RTE)	99,2519%

Percebe-se que para este caso o Erro Médio quadrático aumentou em relação aos casos anteriores. Este fato se deve pela presença de um componente fora do dicionário de Fourier, o interharmônico. Em contra partida os valores de Correlação cruzada e porcentagem de energia mantida continuam próximos a 100%, mostrando uma boa eficiente do sistema desenvolvido.

O erro absoluto entre o sinal de entrada e o sinal reconstruído pelo sistema de compressão é mostrado conforme a Figura 19.

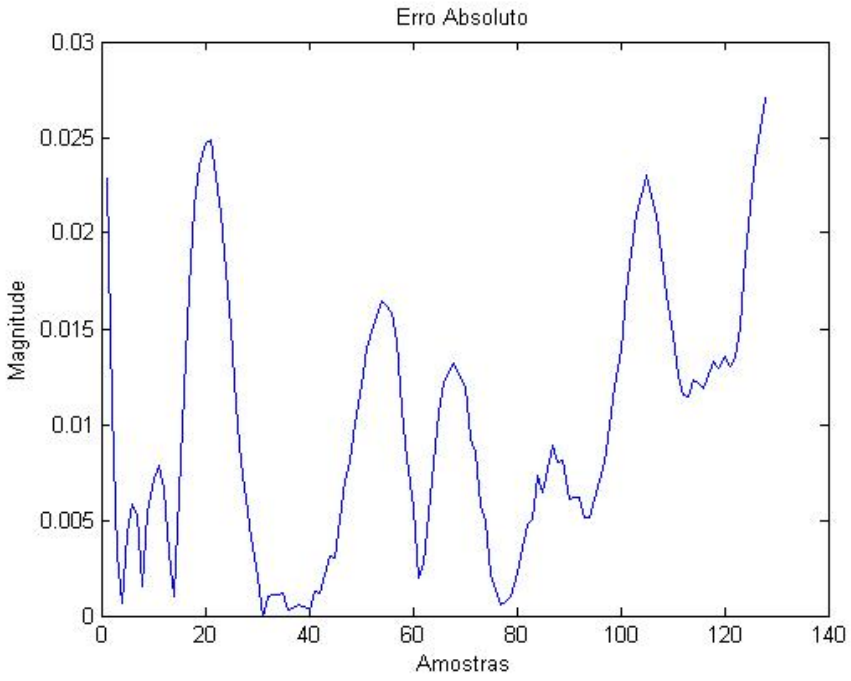

Figura 19: Erro Absoluto entre o sinal de entrada e o sinal reconstruído para o caso 4.

Percebe-se que para este caso, o sinal reconstruído e o sinal reconstruído pelo MATLAB possuem um maior erro absoluto (na faixa de 0,02) em relação ao sinal de entrada, essas diferenças ocorrem devido à limitação de que o sinal reconstruído só pode utilizar apenas 8 (oito) coeficientes do dicionário. Um diferença em relação aos outros casos é a presença de componentes de Wavelet para representar o sinal escolhido.

5.5 SINAL COM AFUNDAMENTO NA AMPLITUDE

Para este quinto caso o sinal de teste gerado é constituído por uma componente senoidal que apresenta uma variação de amplitude. A amplitude do sinal varia em degrau, com uma variação negativa de 40% do seu valor original. Após atingir este valor a amplitude permanece constante durante um terço do período do sinal, retornando em degrau ao seu valor original. Este sinal é descrito pela Equação (5.8) e o afundamento em degrau na amplitude do sinal de entrada, é apresentado na Figura 20. Para este sinal o sistema retornou o número de componentes de Fourier e Wavelet descritos na Tabela 11.

$$x(t) = A(t).sen(2\pi ft) \tag{5.8}$$

Tabela 11: Número de Componentes de Fourier e Wavelet no caso 5.

Coeficientes de Fourier	4
Coeficientes de Wavelet	4

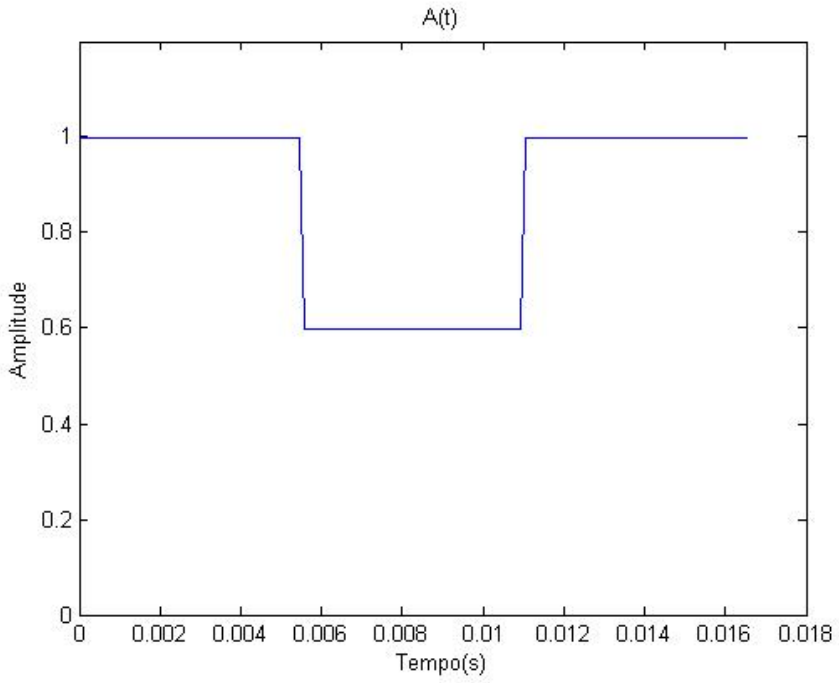

Figura 20: Afundamento de amplitude para o sinal de entrada no caso 5.

O sinal reconstruído é mostrado na Figura 21, onde é comparado com o sinal original, e o sinal reconstruído pelo MATLAB. Na tabela 12 são apresentados os valores para as métricas definidas anteriormente.

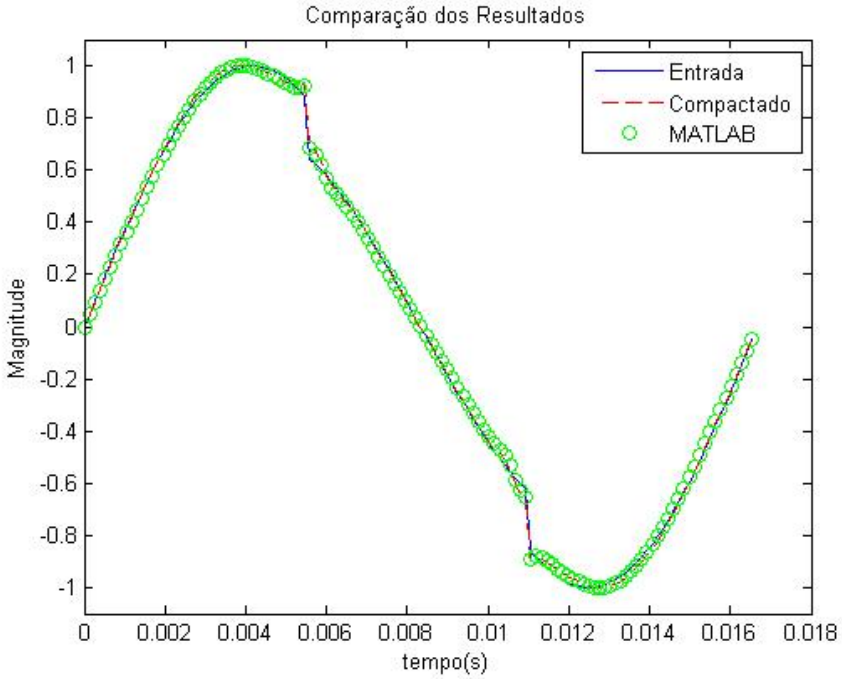

Figura 21: Comparação entre o sinal gerados para o caso 5.

Tabela 12: Resultados métricos para o quinto caso.

Erro Médio Quadrático (NMSE)	0,2426%
Correlação cruzada (COR)	99,7594%
Porcentagem de energia mantida (RTE)	99,4726%

Percebe-se que para este caso o Erro Médio quadrático é menor que o erro do caso anterior. Outro fato a se ressaltar, são os valores de Correlação cruzada e porcentagem de energia, que se mantem próximas ao valor ideal, mostrando-se eficaz na compactação deste tipo de distúrbio no sinal de entrada.

O erro absoluto entre o sinal de entrada e o sinal reconstruído pelo sistema de compressão é mostrado conforme a Figura 22.

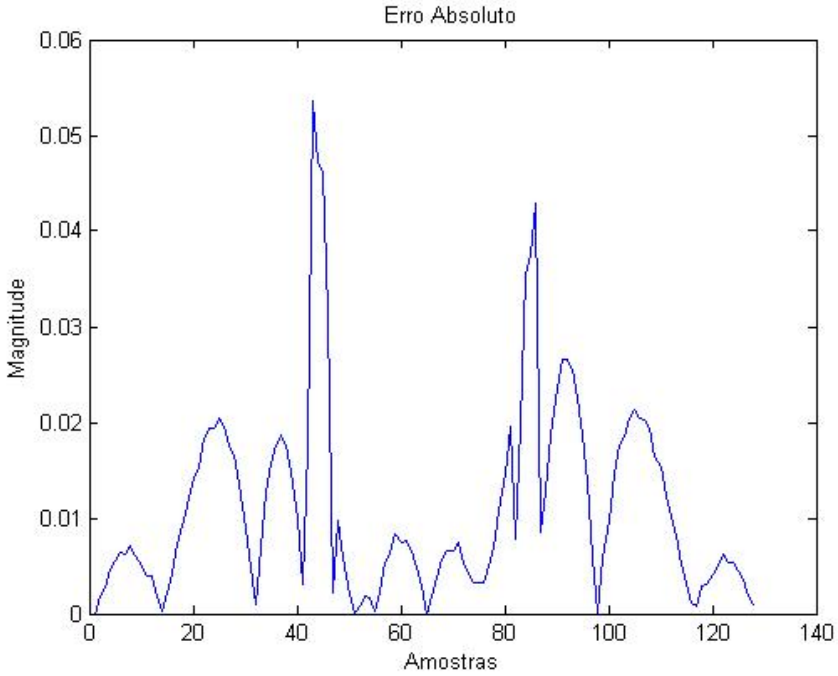

Figura 22: Erro Absoluto entre o sinal de entrada e o sinal reconstruído para o caso 5.

É evidente que ao se observar a transição de amplitude do sinal, ocorre pequenas diferenças em relação ao sinal original, assim como no caso anterior essas diferenças ocorrem devido a limitação de que o sinal reconstruído só pode utilizar 8(oito) coeficientes do dicionário para representar o sinal.

5.6 SINAL COM ELEVAÇÃO NA AMPLITUDE

Para este sexto caso o sinal de teste gerado é constituído por uma componente senoidal, que apresenta uma variação de amplitude. A amplitude do sinal varia em degrau, com uma variação positiva de 20% do seu valor original. Após atingir este valor a amplitude permanece constante durante um terço do período do sinal, retornando em degrau ao seu valor original. Este sinal é descrito pela Equação (5.9) e a elevação em degrau na amplitude do sinal de entrada, é apresentado na Figura 23. Para este sinal o sistema retornou o número de componentes de Fourier e Wavelet descritos na Tabela 13.

$$x(t) = A(t).sen(2\pi ft) \tag{5.9}$$

Tabela 13: Número de Componentes de Fourier e Wavelet no caso 6.

Coeficientes de Fourier	6
Coeficientes de Wavelet	2

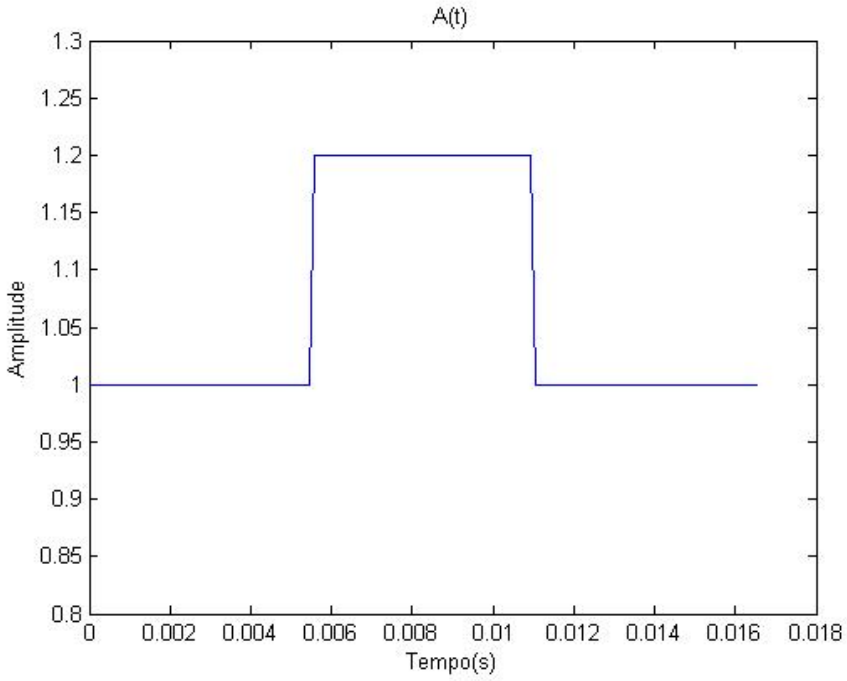

Figura 23: Afundamento de amplitude para o sinal de entrada no caso 6.

O sinal reconstruído é mostrado na Figura 24, onde é comparado com o sinal original, e o sinal reconstruído pelo MATLAB. Na tabela 14 são apresentados os valores para as métricas definidas anteriormente.

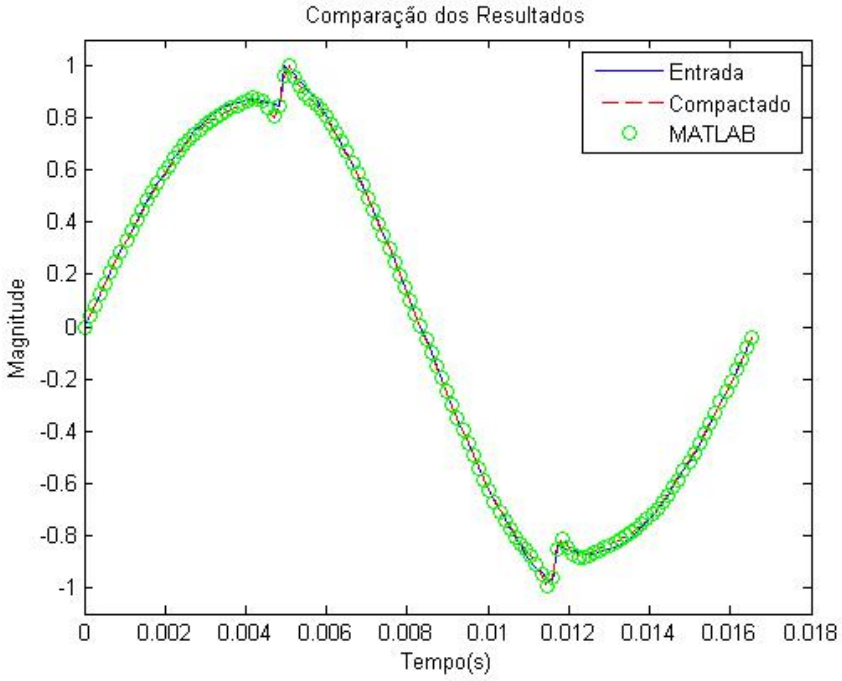

Figura 24: Comparação entre o sinal gerados para o caso 6.

Tabela 14: Resultados métricos para o sexto caso.

Erro Médio Quadrático (NMSE)	0,3002%
Correlação cruzada (COR)	99,92%
Porcentagem de energia mantida (RTE)	100%

Percebe-se que para este caso o Erro Médio quadrático aumento em relação ao caso de afundamento de amplitude. Porém, ainda o valor do erro se mostra baixo, indicando que o sistema é eficaz na compactação deste tipo de distúrbio no sinal de entrada. Outro fato a se ressaltar, são os valores de Correlação cruzada e porcentagem de energia, que se mantem próximas ao valor ideal.

O erro absoluto entre o sinal de entrada e o sinal reconstruído pelo sistema de compressão é mostrado conforme a Figura 25.

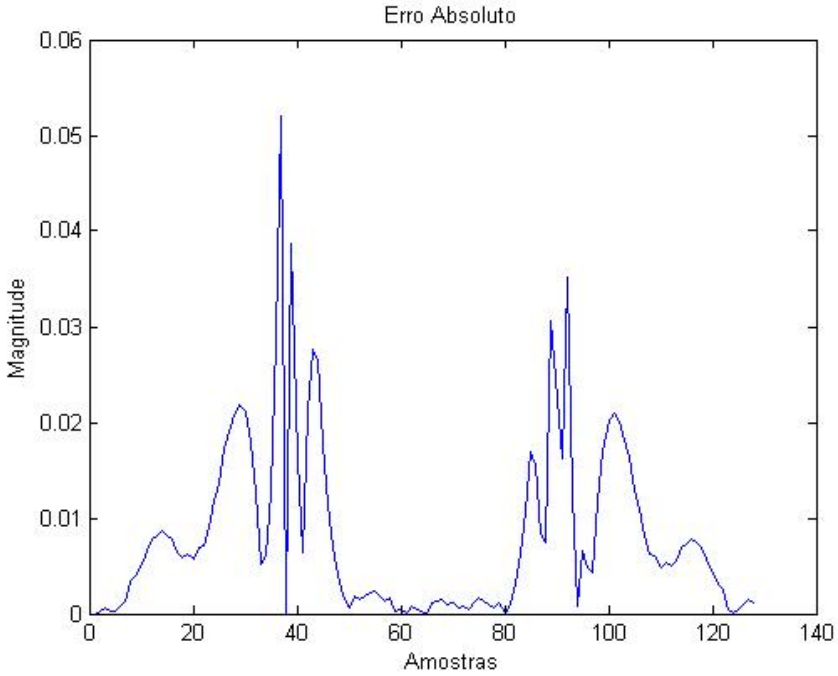

Figura 25: Erro Absoluto entre o sinal de entrada e o sinal reconstruído para o caso 6.

É visualmente evidente que ao se observar a transição de amplitude do sinal, ocorre pequenas diferenças em relação ao sinal original, assim como no caso anterior essas diferenças ocorrem devido a limitação de que o sinal reconstruído só pode utilizar 8(oito) coeficientes do dicionário para representar o sinal.

5.7 SINAL COM VARIAÇÃO EXPONENCIAL NA AMPLITUDE DE UMA COMPONENTE DO SINAL

Para este sétimo caso o sinal de teste gerado é constituído por uma componente senoidal fundamental, somada com outra componente senoidal, onde a amplitude apresenta uma variação exponencial. A amplitude da componente de vigésimo primeiro harmônico varia de forma exponencial, simulando uma sinal transitório da rede elétrica. Após atingir o valor de 7,5% da amplitude da componente fundamental, o crescimento exponencial é interrompido. Durante 0,002s permanece constante, e depois deste intervalo de tempo, retorna a zero, na forma de uma exponencial decrescente. Este sinal é descrito pela Equação (5.10), e a variação exponencial na amplitude desta componente é apresentado na Figura 26. Para este sinal o sistema retornou o número de componentes de Fourier e Wavelet descritos na Tabela 15.

$$x(t) = sen(2\pi ft) + 0{,}075u(t).sen(2\pi.21.ft) \tag{5.10}$$

Tabela 15: Número de Componentes de Fourier e Wavelet no caso 7.

Coeficientes de Fourier	4
Coeficientes de Wavelet	4

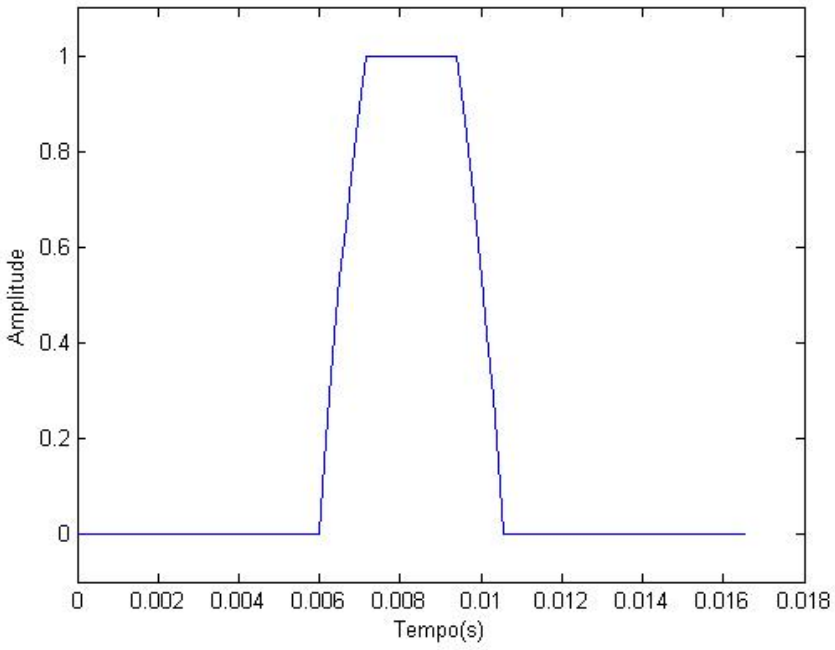

Figura 26: Variação na amplitude do vigésimo primeiro harmônico do sinal de entrada no caso 7.

O sinal reconstruído é mostrado na Figura 27, onde é comparado com o sinal original, e o sinal reconstruído pelo MATLAB. Na tabela 16 são apresentados os valores para as métricas definidas anteriormente.

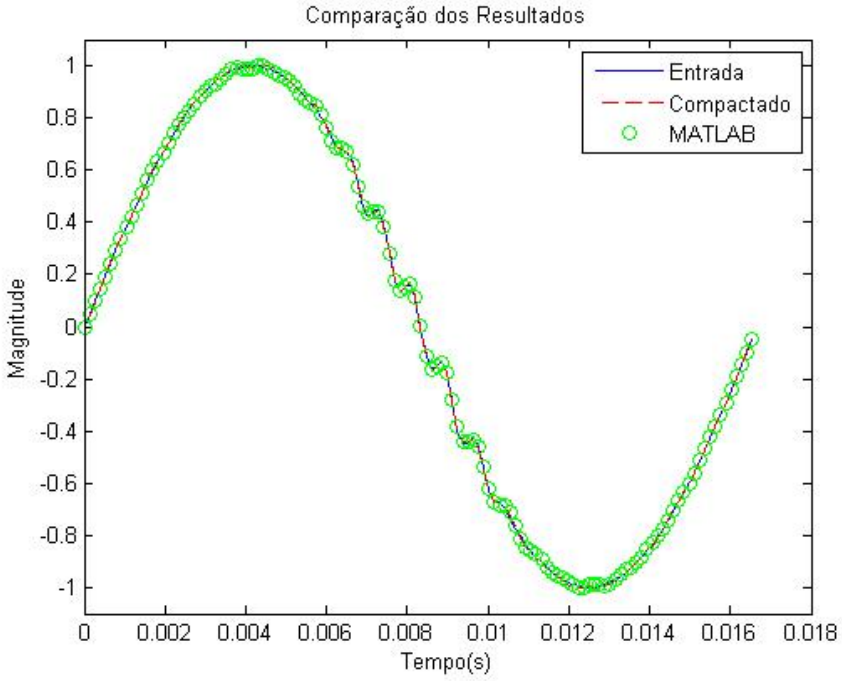

Figura 27: Comparação entre o sinal gerados para o caso 7.

Tabela 16: Resultados métricos para o sétimo caso.

Erro Médio Quadrático (NMSE)	0,3165%
Correlação cruzada (COR)	99,9723%
Porcentagem de energia mantida (RTE)	100%

Percebe-se que para este caso, um pequeno Erro Médio quadrático, um pouco superior aos casos de variação em rampa. Como foi visto, nos casos anteriores os valores de Correlação cruzada e Porcentagem de energia se mantiveram próximas de 100%, implicando em uma boa eficiência do sistema proposto.

O erro absoluto entre o sinal de entrada e o sinal reconstruído pelo sistema de compressão é mostrado conforme a Figura 28.

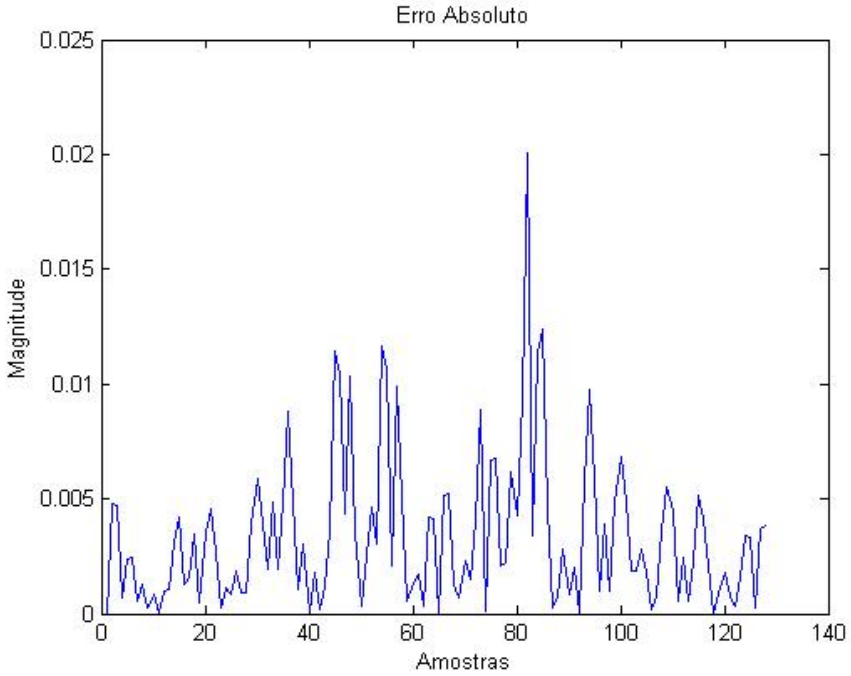

Figura 28: Erro Absoluto entre o sinal de entrada e o sinal reconstruído para o caso 7.

É possível notar pequenas diferenças com o sinal de entrada. Assim como nos casos anteriores, essas diferenças ocorrem devido à limitação de que o sinal reconstruído só pode utilizar apenas 8(oito) componentes do dicionário para representar o sinal de entrada.

6 CONCLUSÕES FINAIS

6.1 CONCLUSÕES

O presente trabalho apresentou um estudo e implementação de uma técnica de representação esparsa em dicionários redundantes, o algoritmo Matching Pursuit(MP), para avaliar a possibilidade de substituir bloco tradicionais de Compressão com Perdas. Um dicionário composto pela união de duas bases ortogonais (Wavelet e Fourier) foi utilizado e o algoritmo de busca foi implementado e seu desempenho foi comparados quanto compressão e reconstrução de diversos sinais.

O dicionário proposto foi pensado não somente para proporcionar uma elevada taxa de compressão e boa qualidade da reconstrução, mas também para permitir uma rápida execução do algoritmo de busca, que depende do número de elementos contidos no dicionário. Com a união das bases de Fourier e Wavelet conseguiu-se resultados interessantes, visto o dicionário é composto por 228 elementos. Além disso, às características das duas bases são complementares, a base de Fourier representa bem, utilizando poucos elementos, os sinais estacionários no tempo enquanto que a base de wavelet representa melhor os sinais transitórios. Com isso então, conseguiu-se uma boa qualidade da representação utilizando poucos elementos do dicionário.

Ao longo deste trabalho a plataforma FPGA foi abordada dando-se enfoque tanto nos recursos de código HDL desenvolvidos, quanto nos blocos de hardware que realizam diversas tarefas dentro sistema proposto.

Uma das técnicas em destaque na questão de otimização de lógica do DLP, foi o método utilizando o processador embarcado. Ele permitiu a flexibilidade de se implementar qualquer algoritmo utilizando o mesmo hardware e ainda permitiu a facilidade de ser programado utilizando-se de um subconjunto de linguagem C.

Foi mostrado também que o sistema proposto mesmo que com um alto custo computacional, é capaz de ser implementado em tempo real, com algumas limitações, mas ainda assim, é capaz de reconstruir os sinais com alta qualidade.

Além da implementação, testes foram feitos para verificar o desempenho do mesmo. Para esses testes, utilizou-se diversos sinais, tendo em vista que um sinal mais complexo poderia expor de forma mais clara os pontos fortes e fracos do sistema.

O desempenho do sistema proposto foi testado e comparado com o desempenho de um algoritmo desenvolvido no Software MATALAB. Os resultados do sistema proposto são no mínimo equivalentes aos do MATLAB, o que motiva na continuação da pesquisa e da otimização do sistema desenvolvido. Com respeito à compressão, o sistema é adequado para a aplicação em cenários de smart grids, haja visto sua habilidade de comprimir qualquer sinal, e não somente os sinais livres de distúrbios.

Analisando os resultados obtidos percebeu-se que o algoritmo apresentou resultados satisfatórios, mostrando ser atrativo para aplicações de compressão de sinais elétricos, apesar de apresentar complexidade computacional mais elevada.

Enfim, conclui-se que sistema construído o foi capaz de cumprir com os objetivos que lhe foram propostos a realizar. Mostrou-se eficaz, fornecendo bons níveis de compressão aos sinais testados, e principalmente, mantendo um bom nível de reconstrução. Sendo uma ferramenta muito útil para resolver diversos problemas de armazenamento de sinais advindos de sistemas elétricos de potência.

6.2 TRABALHOS FUTUROS

Como trabalhos futuros propõe-se:

i Desenvolver um protótipo funcional do sistema desenvolvido.

ii Estudar diferentes algoritmos de representação esparsa em dicionários redundantes, e comparar suas implementações com o algoritmo estudado neste trabalho.

iii Implementar em FPGA diferentes técnicas de representação esparsa em dicionários redundante.

iv Utilizar a técnica de representação esparsa em dicionários redundantes não só para comprimir os sinais, mas também para detectar e classificar os distúrbios.

REFERÊNCIAS

AHARON, M.; ELAD, M.; BRUCKSTEIN, A. *rmk*-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, IEEE, v. 54, n. 11, p. 4311–4322, 2006.

DONOHO, D. L. et al. Data compression and harmonic analysis. *IEEE Transactions on Information Theory*, IEEE, v. 44, n. 6, p. 2435–2476, 1998.

ELAD, M.; AHARON, M. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image processing*, IEEE, v. 15, n. 12, p. 3736–3745, 2010.

ENGAN, K.; AASE, S. O.; HUSOY, J. H. Method of optimal directions for frame design. In: IEEE. *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*. 1999. v. 5, p. 2443–2446.

GABOR, D. Theory of communication. part 1: The analysis of information. *Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering*, IET, v. 93, n. 26, p. 429–441, 1946.

GOODWIN, M. M.; VETTERLI, M. Matching pursuit and atomic signal models based on recursive filter banks. *IEEE Transactions on Signal Processing*, IEEE, v. 47, n. 7, p. 1890–1902, 1999.

GRIBONVAL, R.; BACRY, E. Harmonic decomposition of audio signals with matching pursuit. *IEEE Transactions on Signal Processing*, IEEE, v. 51, n. 1, p. 101–111, 2003.

JAGGI, S. et al. High resolution pursuit for feature extraction. *Applied and Computational Harmonic Analysis*, Elsevier, v. 5, n. 4, p. 428–449, 1998.

KAPISCH, E. B. Detecção e compressão de distúrbios elétricos baseadas em plataforma fpga. Universidade Federal de Juiz de Fora (UFJF), 2015.

KREUTZ-DELGADO, K.; RAO, B. Focuss-based dictionary learning algorithms. In: *Proc. SPIE*. 2000. v. 4119, p. 459–473.

KRIM, H. et al. On denoising and best signal representation. *IEEE transactions on information theory*, IEEE, v. 45, n. 7, p. 2225–2238, 1999.

LEEUWEN, J. *Handbook of theoretical computer science.* : Elsevier, 1990.

LESAGE, S. et al. Learning unions of orthonormal bases with thresholded singular value decomposition. In: IEEE. *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*. 2005. v. 5, p. v–293.

LEWICKI, M. S.; OLSHAUSEN, B. A. Probabilistic framework for the adaptation and comparison of image codes. *JOSA A*, Optical Society of America, v. 16, n. 7, p. 1587–1601, 1999.

MALLAT, S. G.; ZHANG, Z. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on signal processing*, IEEE, v. 41, n. 12, p. 3397–3415, 1993.

VERA-CANDEAS, P. et al. Transient modeling by matching pursuits with a wavelet dictionary for parametric audio coding. *IEEE Signal Processing Letters*, IEEE, v. 11, n. 3, p. 349–352, 2004.