UNIVERSIDADE FEDERAL DE CAMPINA GRANDE

CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Estimação de frequências de sinais digitais baseada no paradigma conexionista.

Dissertação apresentada à Coordenação do Programa de Pós Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, em cumprimento às exigências para obtenção do grau de Mestre em Ciências no Domínio da Engenharia Elétrica.

Alan Vinícius Santos Sá

Orientadores:
Raimundo Carlos Silvério Freire
Jugurta Rosa Montalvão Filho

Campina Grande, PB
2009

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

S111p
2009 Sá, Alan Vinicius Santos.
Estimação de freqüências de sinais digitais baseada no paradigma conexionistae / Alan Vinicius Santos Sá. — Campina Grande, 2009.
52 f. : il.

Dissertação (Mestrado em Engenharia Elétrica) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.
Referências.
Orientadores: Prof. Dr. Raimundo Carlos Silvério Freire, Jugurta Rosa Montalvão Filho.

1. Estimação de Frequências. 2. Conexicionismo. I. Título.

CDU – 621.391(043)

ESTIMAÇÃO DE FREQÜÊNCIA DE SINAIS DIGITAIS BASEADA NO PARADIGMA CONEXIONISTA

ALAN VINÍCIUS SANTOS SÁ

Dissertação Aprovada em 24.07.2009

RAIMUNDO CARLOS SILVÉRIO FREIRE, Dr., UFCG
Orientador

JUGURTA ROSA MONTALVÃO FILHO, Dr., UFS
Orientador (Ausência Justificada)

VINCENT PATRICK MARIE BOURGUET, Dr., Visitante – UFCG
Componente da Banca

ANTONIO AUGUSTO LISBOA DE SOUZA, Dr.
Componente da Banca

BENEDITO ANTONIO LUCIANO, D.Sc., UFCG
Componente da Banca

CAMPINA GRANDE - PB
JULHO - 2009

Dedico esta dissertação a meus pais.

Agradecimentos

Meus sinceros agradecimentos:

- A Deus por me permitir estar aqui;
- Aos meus pais por tudo o que fizeram por mim;
- A minha avó materna pela sabedoria e por sempre me incentivar;
- Aos meus irmãos pelo companheirismo e amizade;
- A minha esposa pela ajuda e compreensão em determinados momentos;
- Aos meus orientadores pela ajuda durante esse período;
- Aos professores que contribuíram para a minha formação;
- A todos os meus colegas que me auxiliaram neste mestrado;
- À CAPES pelo apoio financeiro;
- À CHESF, pelo apoio finaceiro.

"A mente que se abre a uma nova idéia
jamais voltará ao seu tamanho original."

Albert Einstein

Resumo

Neste trabalho é proposto o desenvolvimento e implementação (em *software* e *hardware)* de um sistema, baseado no paradigma conexionista, para a estimação das componentes de frequência de um sinal digital. Os resultados obtidos com a utilização desse sistema são discutidos e comparados com os resultados provenientes de outras abordagens objetivando-se assim uma avaliação do método proposto.

Palavras-chave: estimação de freqüências, conexionismo.

Abstract

In this work, the development and implementation (in software and hardware) of a system based on the connectionist is realized. The goal is to estimate of the frequency components of a digital signal. The results are discussed and compared with the results proceeding from other methods as a means of evaluating the considered method.

Keywords: frequency estimation, connectionist, FPGA.

Sumário

Índice de ilustrações

Capítulo 1 - Introdução

Neste capítulo é feita a introdução ao assunto tema desta dissertação, como também é descrita a organização estrutural do documento.

De uma maneira geral, fenômenos produzem sinais que o caracterizam. Seja o sinal elétrico produzido pelos simples bater do coração, durante um eletrocardiograma, ou a variação do grau de agitação das partículas durante um experimento químico, ou o som produzido durante a fala de uma criança, todos esses sinais portam informações acerca do fenômeno que o produziu.

Tais informações podem estar contidas em parâmetros do sinal como: amplitude, em que, como exemplo, pode-se citar os registros de sinal em um sismograma, cuja amplitude permite inferir a magnitude de um terremoto; fase, em que se pode ilustrar com o princípio de funcionamento de um radar, cuja defasagem entre as ondas emitida e refletida indica a distância da aeronave; freqüência, no caso das notas produzidas por um instrumento musical.

Para um estudo acerca de um determinado fenômeno recorre-se à análise de seus sinais, adequando-os a condições específicas, caso necessário, e buscando-se extrair suas informações, conforme ilustrado na figura 1.1. É neste contexto que surge o conceito de processamento de sinais que trata da representação, transformação e manipulação de sinais, e consequentemente das informações que eles contêm. O processamento pode

ser feito de forma analógica ou digital [1][2].

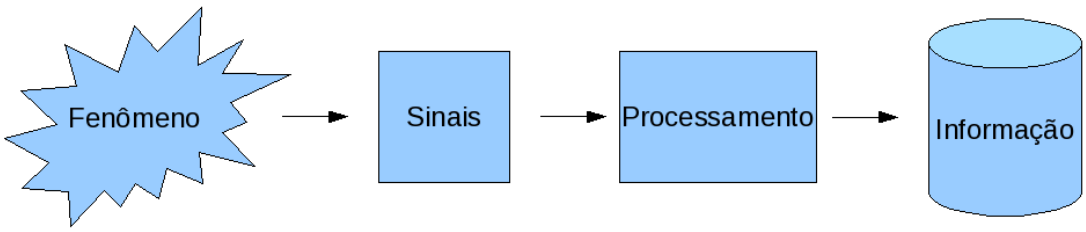


Figura 1.1: Ilustração da extração de informação de um fenômeno.

O Processamento Digital de Sinais – PDS – é uma área que se desenvolveu rapidamente nas últimas décadas, sobretudo por conta do avanço tecnológico nas áreas de computação e fabricação de circuitos integrados. Esses avanços tornaram possível a construção de sistemas digitais sofisticados, cada vez menores, mais rápidos e mais baratos, capazes de realizar tarefas com alto grau de complexidade computacional, além de permitir operações programáveis, se comparados aos sistemas analógicos [2][3].

A associação desses *hardwares* digitais com *softwares* propiciou um grau de flexibilidade ainda maior no projeto de sistemas de processamento da informação. Isso, conjuntamente com as características inerentes à abordagem digital, como por exemplo o fato de o sinal processado estar menos sujeito à degradação e distorções, o que se explica devido à relação sinal ruído, bem como a questão da facilidade no armazenamento dos dados, fez com que onde houvesse disponibilidade de circuitos digitais e os mesmos tivessem velocidade suficiente para o processamento, eles fossem preferíveis em detrimento dos circuitos analógicos [3].

Um outro domínio que tem se destacado e se firmado cada vez mais como área de pesquisa é o Conexionismo. Embora seja uma abordagem recente, de poucas décadas, suas idéias vêm sendo utilizadas em diferentes áreas da ciência. Para citar exemplos, redes conexionistas têm auxiliado

diagnósticos médicos, trabalhos na área da astronomia, gerenciamento de investimentos, diagnósticos em sistemas de controle [5][6][7][8].

O Conexionismo baseia-se em aspectos construtivos da neurofisiologia do cérebro e tenta incorporar suas propriedades funcionais. Portanto, conceitos como conjunto de unidades simples de processamento, estados de ativação, função de saída das unidade de processamento, padrões de conectividade e regras de ativação são características observadas nas Redes Conexionistas [5].

Assim, um sistema fundamentado neste paradigma é basicamente uma rede/conjunto/população de inúmeras unidades simples que são conectadas entre si e orientadas com o objetivo de realizar uma determinada tarefa ou função de maior complexidade. Essa orientação pode ser feita via exemplos – caso de aprendizagem supervisionada – ou pela extração de características comuns da informação disponível na entrada da rede – caso de aprendizagem auto-organizado.

A natureza extremamente paralela das redes conexionistas as fazem potencialmente rápidas para o processamento de determinadas tarefas e adequadas à implementação em tecnologia VLSI (*very-large-scale-integrated)* . Esta mesma peculiaridade, propicia que quando implementada na forma de *hardware*, a rede seja inerentemente tolerante à falhas, isso no sentido de que seu desempenho se degrada suavemente antes de um comprometimento total de sua resposta caso ocorram falhas em suas unidades de processamento [4].

Nesse contexto, busca-se neste trabalho explorar as potencialidades do paradigma conexionista e desenvolver unidades conexionistas que componham uma rede/sistema capaz de realizar a análise espectral de um sinal digital. Adicionalmente, propõe-se que esta implementação seja realizada tanto em *software* quanto *hardware*.

Esta dissertação está organizada da seguinte maneira: no capítulo 2 é

explicado o problema que se deseja resolver, como também é apresentada a abordagem proposta para a sua resolução; no capítulo 3 são discutidos aspectos relacionados à implementação em *hardware* da unidade de processamento da rede; no capítulo 4 são apresentadas abordagens comparativas, como também os resultados obtidos e discussões acerca destes; no capítulo 5 são apresentadas as conclusões e as propostas para trabalhos futuros. Finalizando são apresentados os anexos e são listadas as referências bibliográficas.

Capítulo 2 – Abordagem Proposta

O primeiro tópico deste capítulo trata do objetivo principal deste trabalho, ou seja, a formulação do problema que se deseja resolver e contribuir na sua resolução. Em seguida é apresentada a abordagem proposta para a solução do problema discutindo-se inicialmente aspectos estruturais da rede proposta, para posteriormente serem aprofundados os aspectos de cada unidade que a compõe.

2.1 Formulação do problema

Inicialmente, deve-se considerar um sinal digital $s(n)$ composto por p sinais senoidais puros e limitado em N amostras, conforme representação na equação 2.1.

$$s(n)=\sum_{k=1}^{p} A_k \, sen(\omega_k \cdot n+\phi_k) \quad , \quad n=1,2,\ldots,N \tag{2.1}$$

Em que:

- A_k , ω_k e ϕ_k são amplitude, frequência e fase das componentes senoidais.

O objeto de estudo consiste no sinal digital $s(n)$ acrescido de uma componente ruidosa $r(n)$, de mesmo tamanho (N amostras), obtendo-se assim um sinal digital resultante $x(n)$, dado pela equação 2.2:

$$x(n)=s(n)+r(n) \quad , \quad n=1,2,\ldots,N \tag{2.2}$$

Considerando que os valores de ω_k são desconhecidos, o objetivo deste trabalho consiste em identificá-los. Para isto é proposta a utilização de uma rede composta por inúmeras unidades de processamento interconectadas e que realizem uma pequena parte da tarefa de forma que com a fusão do resultado dos seus processamentos obtenham-se os valores de frequências das componentes do sinal $s(n)$. Na figura 2.1 é apresentada uma ilustração com a ideia geral do trabalho.

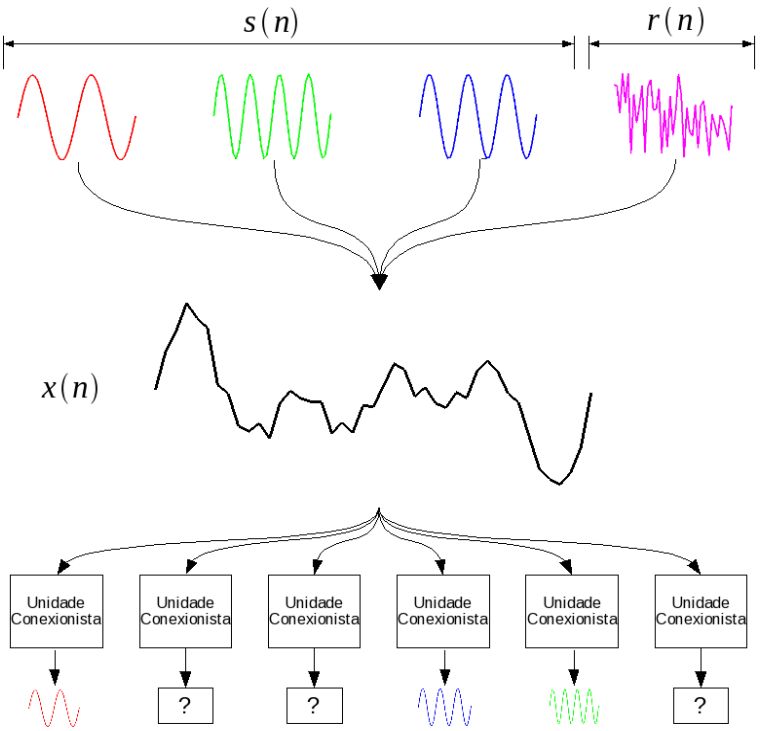


Figura 2.1: Ideia geral proposta no trabalho

2.2 Rede Conexionista

Um dos principais aspectos a ser considerado no desenvolvimento de uma rede é o padrão de interconexão de suas unidades, que juntamente com a disposição das mesmas definem a sua topologia.

A suscetibilidade ao comprometimento total do processamento caso unidades falhassem foi um fator considerado na escolha da topologia. Como exemplo podemos citar a topologia em árvore, a qual está ilustrada na figura 2.2.a. Nesta estrutura, a falha em um dos nós compromete o processamento dos nós hierarquicamente superiores, não sendo portanto uma solução atrativa.

Uma alternativa seria aumentar o número de conexões entre as unidades de forma que as unidades entre camadas adjacentes fossem totalmente conectadas e a falha de uma unidade pudesse ser compensada pela atuação de outra. Esta opção permite um maior grau de robustez a falhas na medida em que se aumenta o grau de complexidade de conectividade entre suas unidades.

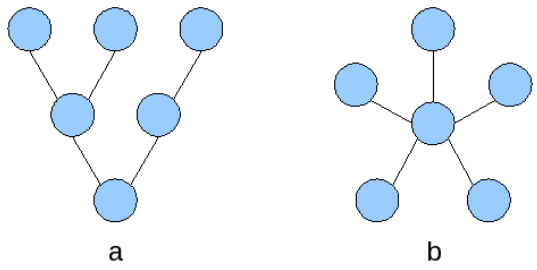


Figura 2.2: Topologias de redes: a) árvore; b) estrela;

Uma outra solução robusta a falhas consistiria em utilizar unidades de processamento que fossem ao máximo independentes umas das outras, proporcionando que o resultado do processamento de um nó não fosse requisito para o processamento de vários outros.

Uma topologia simples, que agrega essa ideia e evita a necessidade de muitas conexões entre suas unidades, é a topologia em estrela. Nela seus nós processam informação de forma paralela e independente apenas enviando a informação para uma unidade diferenciada e centralizada. Esta topologia está ilustrada na figura 2.2.b.

Após a escolha da topologia a ser adotada, foi formulado o sistema cuja ideia básica do funcionamento está ilustrada na figura 2.3. Como entrada do sistema tem-se N amostras do sinal $x(n)$ que é analisado por cada uma das unidades conexionistas (unidades de processamento), as quais atuam individualmente em sub-bandas predeterminadas à procura de componentes harmônicas presentes. Na sequência, valores de frequências estimadas juntamente com um parâmetro de qualidade dessa estimação são repassados à unidade de fusão, cuja função é avaliar esses dados e gerar um vetor de dados contendo as frequências estimadas pela rede.

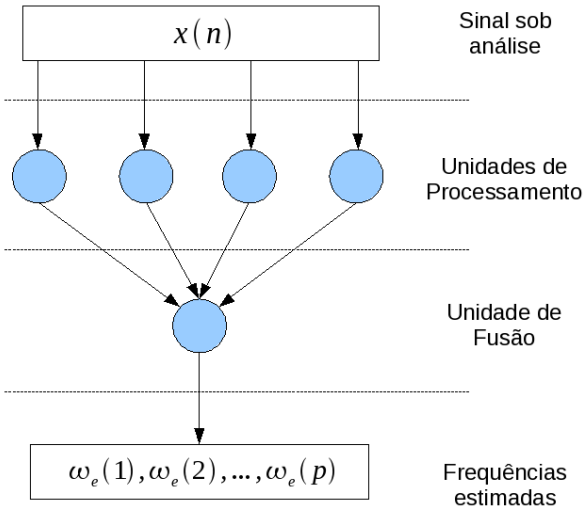


Figura 2.3: Estrutura da rede adotada.

Uma vez que as unidades de processamento atuam em sub-bandas é desejável que cada componente do sinal a ser analizado esteja presente apenas em uma delas. Dessa forma, quanto mais estreitas forem essas sub-

bandas melhor. Entretanto, isso implica a utilização de filtros de ordens elevadas e consequentemente maior complexidade computacional. Assim, a largura da sub-banda e, naturalmente, o número de unidades de processamento utilizadas ficam condicionadas à basicamente duas coisas: às características do tipo de sinal a ser analisado e à complexidade computacional permitida.

Nas sub-seções seguintes são detalhados os aspectos construtivos das unidades conexionistas e da unidade de fusão.

2.2.1 Unidades Conexionistas

As unidades conexionistas são estruturas adaptativas simples cujo objetivo é identificar se há alguma componente harmônica em uma determinada sub-banda de frequências $\Delta\omega$ limitada pelo intervalo de frequências normalizadas $[\omega_{mín}, \omega_{máx}]$ rad/amostra, que são repassados à unidade como parâmetros. Como entrada tem-se o sinal $x(n)$ a ser analisado e na saída dessas unidades tem-se dois valores: ω_e e J, que são respectivamente a frequência estimada pela unidade e um fator de qualidade dessa estimação. Na figura 2.4 são ilustradas as entradas e saídas da unidade.

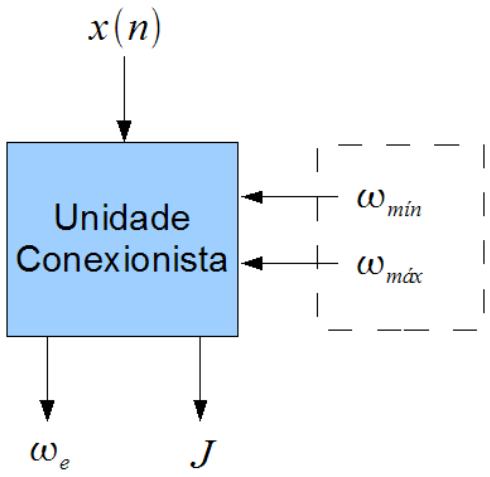


Figura 2.4: Unidade com suas entradas e saídas.

O processamento da unidade se inicia com a filtragem do sinal $x(n)$ utilizando-se um ressonador digital de maneira a diminuir o efeito do ruído ou de outras componentes não pertencentes à sub-banda em questão. Como não se dispõe de nenhuma informação sobre a frequência dominante na faixa de atuação da unidade, a frequência de ressonância do filtro ω_r é inicialmente ajustada para o valor central da sub-banda, de acordo com a equação 2.3:

$$\omega_r = \frac{\omega_{mín} + \omega_{máx}}{2} \tag{2.3}$$

Uma vez que o sinal filtrado $x_f(n)$ é obtido, inicia-se a fase de estimação de sua frequência. Nesta etapa, são selecionados M valores discretos de frequência − pertencentes ao intervalo em questão, $[\omega_{min}, \omega_{max}]$ − definindo-se assim um espaço de busca composto por $\{\omega_m | m = 1,2,...,M\}$. Estes valores serão testados de forma a verificar qual deles está mais próximo do valor da frequência do sinal filtrado.

Esse teste baseia-se na observação de que a autocorrelação normalizada $\Re_{y(n)}(k)$ de índice k de um sinal senoidal digital $y(n)$ com frequência α é aproximadamente igual ao cosseno de k multiplicado por α . Matematicamente, essa relação pode ser expressa pela equação 2.4:

$$\Re_{y(n)}(k) \approx \cos(k \cdot \alpha) \tag{2.4}$$

Em que:

- $\Re_{y(n)}(k)$ é a autocorrelação normalizada com relação a autocorrelação de índice zero, ou seja $R_{y(n)}(k) / R_{y(n)}(0)$;
- k é o índice da autocorrelação.

Dessa maneira o primeiro passo para a estimação da frequência de $x_f(n)$ consiste em extrair os seus valores de autocorrelação. A correlação do

sinal filtrado com ele mesmo é calculada pela equação 2.5:

$$R_{x_f(n)}(k)=\sum_{n=1}^{N} x_f(n)\cdot x_f(n-k) \tag{2.5}$$

Em que:

- $R_{x_f(n)}(k)$ é a autocorrelação do sinal filtrado;
- N é o número de amostras de $x_f(n)$;
- k é o índice da autocorrelação, e $k=0,1,2,...,K$.

Uma vez obtidos os valores da autocorrelação normalizada $\Re_{y(n)}(k)$, busca-se encontrar, para cada um deles, um valor de frequência $\hat{\omega}_e$ dentre os valores de ω_m que minimize o quadrado da diferença entre a autocorrelação normalizada e o cosseno cujo argumento é o índice k da autocorrelação multiplicado pelos valores de frequência ω_m . Matematicamente, essa operação pode ser expressa conforme a equação 2.6:

$$\hat{\omega}_e(k)=\min_{\omega_m}\{[\cos(k\cdot\omega_m)-\Re_{x_f(n)}(k)]^2\} \tag{2.6}$$

Em que:

- k é o índice da autocorrelação, e $k=1,2,...,K$.

Assim, para cada K valor de $\Re_{y(n)}(k)$ um valor de $\hat{\omega}_e$ é obtido, sendo este correspondente ao valor de ω_m que melhor aproximou a relação descrita na equação 2.4. Idealmente, todos esses valores de $\hat{\omega}_e$ deveriam coincidir. Entretanto, na prática, isso não se confirma sobretudo por conta do ruído presente na banda de filtragem do sinal.

Como forma robusta de estimar a frequência é adotado o uso da mediana, conforme equação 2.7. Poderia ter sido utilizada uma outra medida de tendência central, como por exemplo a média, entretanto como se pretende a implementação dessas estruturas em *hardware* busca-se soluções com complexidade computacional mais simples.

Para o cálculo da média seriam necessárias sucessivas somas, determinadas pela quantidade de elementos, seguidas por uma divisão. Já o cálculo da mediana, considerando-se uma quantidade ímpar de elementos, é feito por meio de um algoritmo de ordenação e em seguida seleciona-se o elemento central do conjunto de valores, sendo assim menos dispendioso computacionalmente.

$$\omega_e = median\{\hat{\omega}_e(1), \hat{\omega}_e(2), \cdots, \hat{\omega}_e(K)\} \tag{2.7}$$

Este valor de frequência estimado não é o valor repassado à unidade de fusão. Conforme mencionado anteriormente a unidade de processamento aqui descrita é uma estrutura adaptativa. Portanto, ω_e é passado ao ressonador digital, que parametriza a sua frequência de ressonância para este novo valor e realiza uma nova filtragem no sinal original $x(n)$. Em seguida, repete-se todo o processamento descrito nesta seção, até se encontrar um novo valor para ω_e , o qual novamente será utilizado no ressonador.

Ao final de cada iteração da unidade de processamento é realizado o cálculo do parâmetro J obtido por meio de uma função de custo definida pela equação 2.8. Comparando-se este ao valor obtido na iteração anterior é possível avaliar se a nova frequência estimada está mais próxima do valor de frequência presente no sinal filtrado.

$$J = \frac{1}{K} \sum_{k=1}^{K} [\cos(k \cdot \omega_e) - R_{x_f(n)}(k)]^2 \tag{2.8}$$

Como regra de parada para este procedimento cíclico adotou-se a interrupção da execução após um número fixo i de iterações. Embora uma solução mais sofisticada, que se baseasse na análise de convergência do parâmetro J fosse possível, foi dada a preferência à solução mais simples. Assim, após i ciclos a unidade de processamento repassa então os valores de ω_e e J para a unidade de fusão.

O procedimento de funcionamento da unidade conexionista, descrito nesta subseção, está ilustrado na figura 2.5.

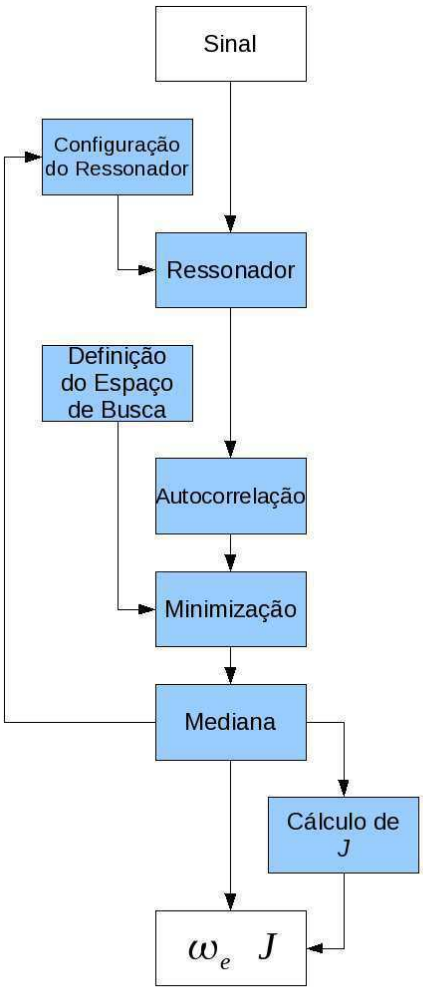


Figura 2.5: Diagrama de Blocos do funcionamento da Unidade Conexionista.

2.2.2 Unidade de Fusão

A unidade de fusão recebe o resultado do processamento, ω_e e J , de cada uma das unidades descritas anteriormente e, a partir deles, estabelece o vetor de saída da rede, contendo as frequências estimadas que formam o sinal de entrada. Seu funcionamento resume-se a avaliar os valores de cada *J* buscando-se a verificação de inferioridade desse valor em relação a um determinado limiar.

Capítulo 3 – Implementação em *hardware*

Neste capítulo são abordados aspectos relacionados à implementação em *hardware* da unidade conexionista do sistema proposto.

Nas últimas décadas, sistemas baseados em FPGA (*Field Programmable Gate Array*) têm se mostrado uma alternativa promissora em relação às soluções baseadas em ASIC (*Application Specific Integrated Circuit*), como também, em relação às que utilizam DSP (*Digital Signal Processors*). E, portanto, têm sido utilizados em uma gama de aplicações, incluindo processamento de sinais, processamento de imagens, processadores de redes e robótica [13,15-17].

O interesse nesse tipo de solução advém da combinação de duas características atrativas: a reconfigurabilidade do *hardware* e a elevada capacidade computacional. Se comparados aos DSP, os FPGA apresentam desempenho superior no processamento da informação, o que não ocorre quando a referência é um ASIC. Sobre este, sua vantagem reside no fato dos FPGA apresentarem um grau de flexibilidade, decorrente de sua reconfigurabilidade, como também um menor ciclo de projeto [14].

Atualmente, há uma grande variedade de FPGA fornecidos por diversos fabricantes como Xilinx, Altera, Atmel, entre outros. Sendo assim, as suas arquiteturas diferem em alguns detalhes, mas conservam uma estrutura

comum, a qual é, basicamente, constituída por blocos lógicos configuráveis (CLB – *Configurable Logic Block*), bloco de entrada e saída (IOB – *Input/Output Block*), e matrizes de chaves de interconexão (*Switch Matrix*). Uma estrutura simplificada é mostrada na figura 3.1. [18]

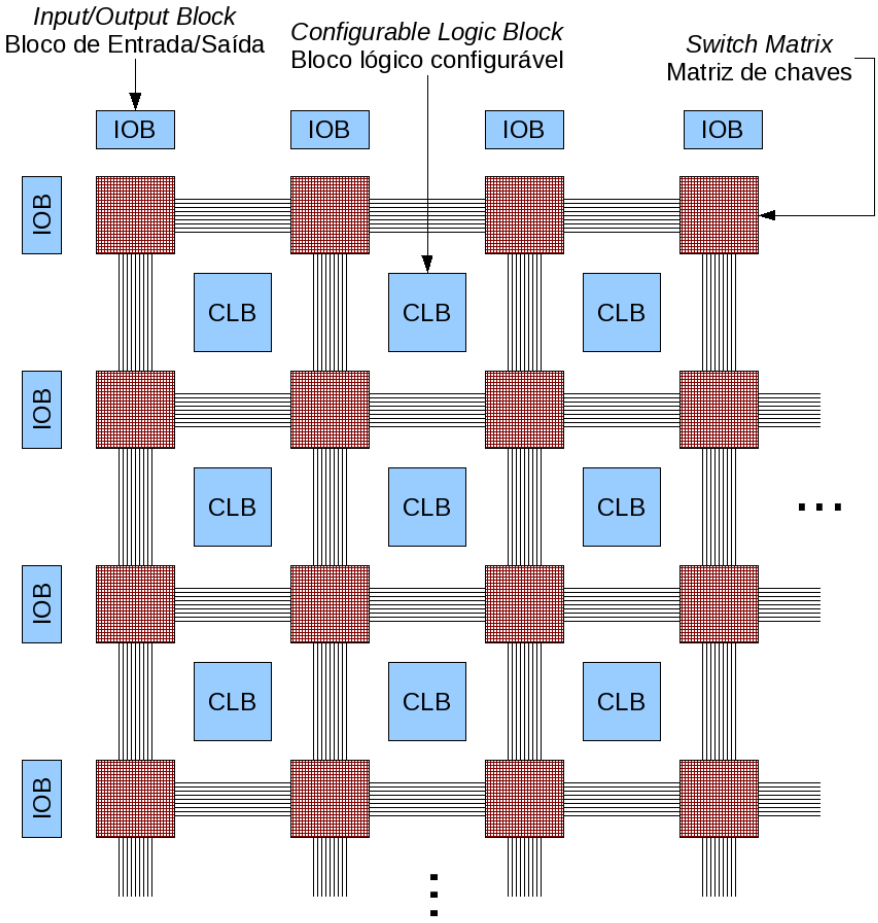


Figura 3.1: Estrutura básica de um FPGA.

Os CLBs formam uma matriz bidimensional, e as chaves de interconexão são organizadas como canais de roteamento horizontal e vertical entre as linhas e colunas dos CLBs. Essas chaves são programáveis e permitem conectar os blocos, inclusive os de entrada e saída, de maneira conveniente, em função das necessidades do projeto. A programação destes dispositivos é

comumente feita por meio da utilização de uma linguagem de descrição de *hardware* (*HDL - Hardware Description Language*).

Dentre as HDLs mais utilizadas no projeto de sistemas digitais se encontram: VHDL (*VHSIC Hardware Description Language*, em que VHSIC significa "*Very High Speed Integrated Circuits*") e o Verilog, sendo esta última a escolhida no desenvolvimento deste trabalho.

3.1 Arquitetura do sistema

A representação de dados, adotada no projeto, segue o padrão IEEE 754 (*Standard for Binary Floating-Point Arithmetic*) [9]. Este padrão estabelece formatos de ponto flutuante que são usados para representar subconjuntos dos números reais. Dentre eles foi adotado o formato “precisão simples”, o qual utiliza 32 bits para a representação do dado, sendo que o bit mais significativo é o bit de sinal; os 8 bits seguintes representam o expoente; e os 23 bits restantes correspondem à mantissa, conforme ilustrado na figura 3.2.

Figura 3.2: Estrutura representativa de um número segundo o formato de precisão simples.

Na figura 3.3 é exibido o diagrama de blocos da implementação desenvolvida. É importante observar que nesta representação, o sinal a ser analisado já consta armazenado em uma memória, ou seja, a etapa de aquisição do sinal não foi coberta neste trabalho. Outra consideração a ser feita é que todos os blocos de circuitos exibidos nessa figura estão conectados a um bloco de controle que administra o *enable* e o *reset* dos circuitos do sistema.

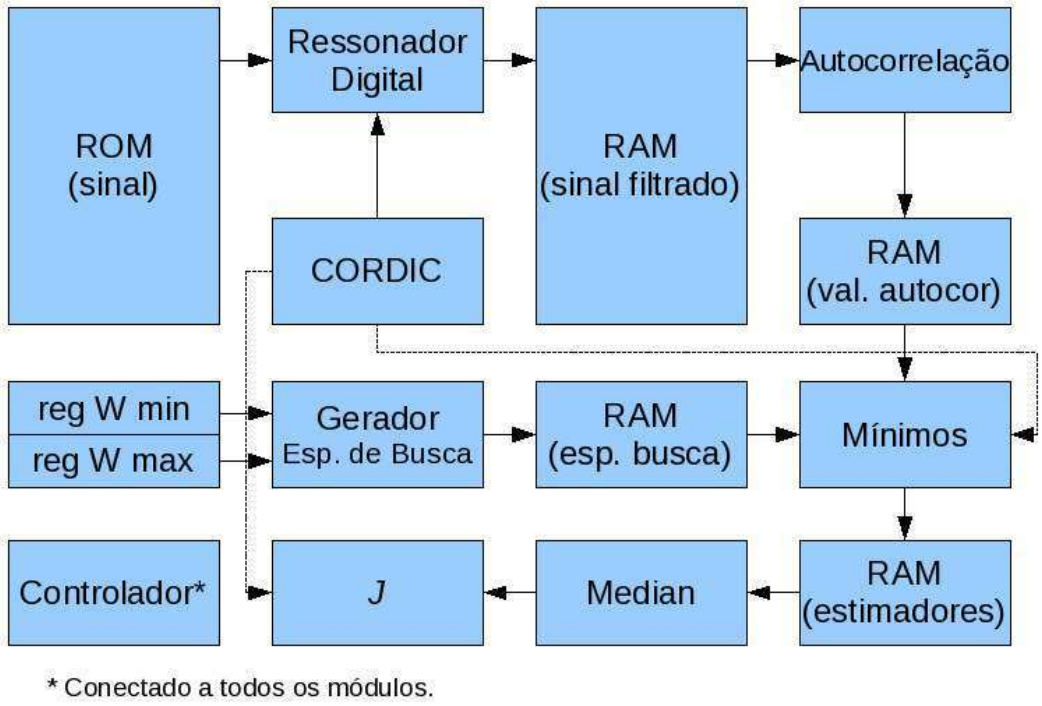


Figura 3.3: Diagrama de blocos da unidade de processamento.

Nas seções seguintes, os aspectos construtivos desses blocos são discutidos detalhadamente.

3.2.1 Ressonador Digital

Uma vez que o sinal analisado encontra-se armazenado, tem-se como etapa seguinte a sua filtragem. Este procedimento é realizado pelo circuito que implementa o Ressonador Digital, e ele está ilustrado na figura 3.4. Inicialmente, aos registradores y_{i-1} e y_{i-2} é atribuído o valor zero, o primeiro valor do sinal a ser analisado é buscado na ROM e armazenado no registrador x_i e é executada a multiplicação entre os registradores $\cos(w)$, oriundo do bloco CORDIC, e o registrador de valor constante $2\mathrm{r}$.

O cálculo de cada ponto do sinal filtrado é realizado basicamente em dois ciclos. No primeiro, o somador realiza a operação entre os registradores x_i e $2\mathrm{r}\cdot\cos(w)\cdot y_{i-1}$. Este resultado é demultiplexado para a segunda entrada do mesmo multiplexador que recebe x_i . No segundo ciclo, esse valor é somado com $-r^2\cdot y_{i-2}$ e assim é obtido o valor de y_i , ponto do sinal filtrado. Dessa forma temos que este circuito implementa a equação 3.1.

$$y_i = x_i + 2\mathrm{r}\cos(w)\cdot y_{i-1} - r^2\cdot y_{i-2} \tag{3.1}$$

Em seguida, os valores dos registradores y_{i-1} e y_{i-2} são atualizado de forma que y_{i-2} recebe o valor de y_{i-1} e este tem o seu valor sobrescrito pelo valor de y_i . Este valor é armazenado na RAM cuja função é armazenar o sinal filtrado.

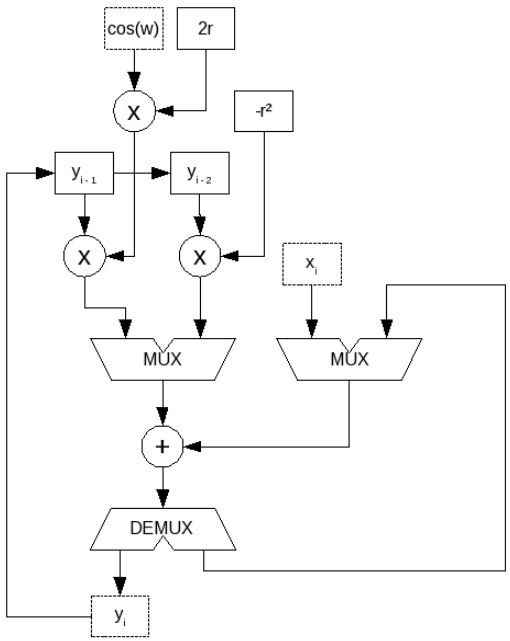


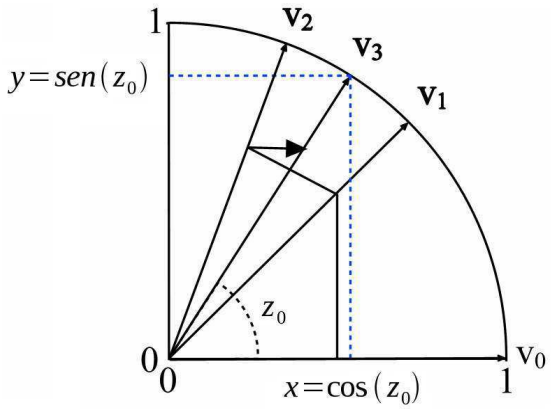
Figura 3.4: Estrutura do Ressonador Digital.

3.2.2 CORDIC

A função cosseno é utilizada nessa estrutura de processamento por diversos blocos. Uma possível solução para a implementação desta função em *hardware* seria a utilização de tabelas com ângulos e seus respectivos cossenos. Entretanto, como nesta aplicação é necessário um alto grau de precisão, seria necessário a utilização de tabela com muitos elementos, sendo preciso comprometer parte da memória com o armazenamento desses valores, como também desenvolver algoritmos de busca eficientes.

A solução adotada foi construir um bloco de processamento cuja finalidade era calcular a função trigonométrica cosseno e este bloco foi projetado baseado no algoritmo CORDIC (*COordinate Rotation DIgital Computer*). Inicialmente este algoritmo foi desenvolvido por VOLDER [10] para cálculos de rotação de vetores em um plano. Posteriormente generalizações foram feitas por WALTHER [11] para rotações em sistemas circulares, lineares e hiperbólicos [12].

A operação deste algoritmo no modo de rotação se inicia com a passagem de um ângulo z_0 como argumento ao algoritmo e este rotaciona um vetor (x_0, y_0) por esse ângulo. Fazendo-se com que $x_0=1$ e $y_0=0$ tem-se, após a sua rotação, que a projeção do vetor resultante (x_i, y_i) no eixo x conterá informações acerca do valor do cosseno de z_0 bem como a projeção no eixo y conterá informação acerca do seno. Na figura 3.5 tem-se o procedimento ilustrado.



A rotação completa é realizada por meio de uma sequência de rotações curtas e conhecidas. Dessa forma, a cada iteração do algoritmo o vetor é rotacionado, através do acréscimo ou diminuição no acumulador do ângulo z_0 . Quanto maior o número de iterações mais exato será o valor do $\cos(z_0)$. As equações do CORDIC que regem este modo de operação são:

$$\begin{aligned} x_{i+1} &= x_i - y_i \cdot d_i \cdot \tan(\alpha_i) \\ y_{i+1} &= y_i + x_i \cdot d_i \cdot \tan(\alpha_i) \\ z_{i+1} &= z_i - d_i \cdot \alpha_i \end{aligned} \tag{3.2}$$

Em que:

- $d_i = -1$ se $z_i < 0$, $+1$ de outra maneira;
- α_i são valores de ângulos que produzem as rotações curtas.

Os valores de α , bem como os de sua tangente, são armazenados em memória. A cada iteração o endereço das ROMs são atualizados e um novo valor para (x_i, y_i) é produzido. Valores estes que serão utilizados na próxima iteração. Além do critério “número de iterações”, a monitoração do ângulo residual pode ser uma regra de parada para este algoritmo, entretanto, a utilização de um *hardware* adicional para esta tarefa não é atrativa neste caso

e o algoritmo tem como regra de parada a primeira opção. Na figura 3.6 está ilustrada a estrutura do circuito projetado.

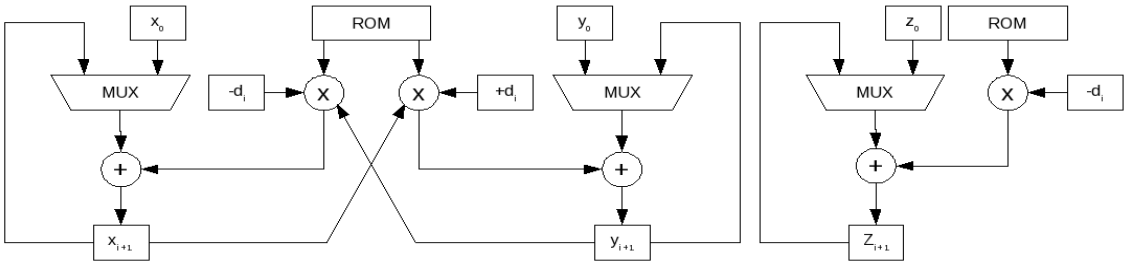


Figura 3.6: Arquitetura do CORDIC.

3.2.3 Autocorrelação

Após o sinal ser filtrado e armazenado em uma memória, esta é acessada pelo bloco chamado Autocorrelação, cuja função é calcular os valores de autocorrelação do sinal. Sua estrutura está ilustrada na figura 3.7.

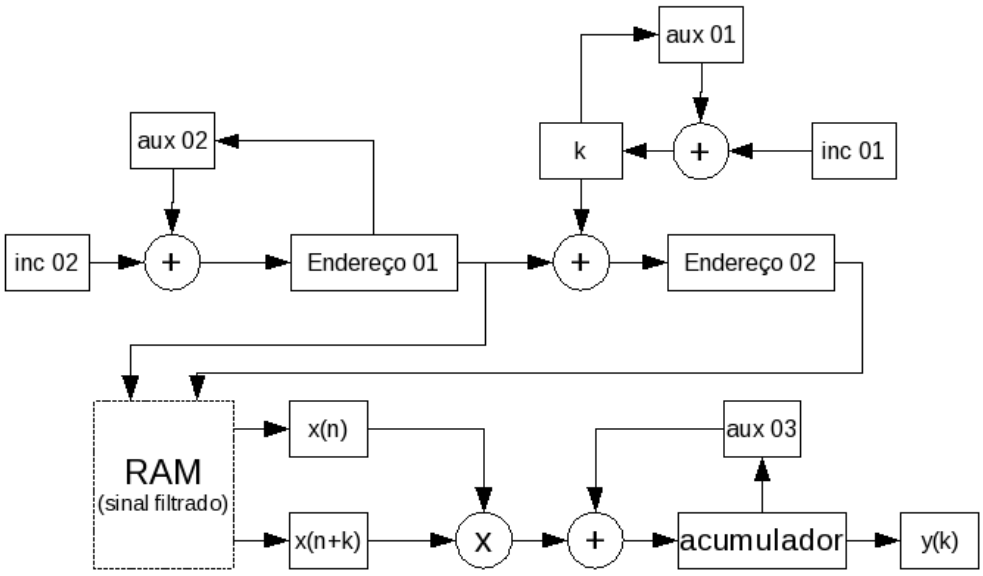


Figura 3.7: Estrutura do bloco autocorrelação.

Este bloco pode ser dividido em basicamente duas partes. A primeira delas tem como objetivo fornecer os endereços de memória que serão acessados para se buscar informação na RAM que armazena o sinal filtrado. A segunda realiza o cálculo da autocorrelação propriamente dita.

No caso da autocorrelação de índice zero, cada elemento do sinal é multiplicado por ele mesmo e estes valores são somados de forma a se obter $y(0)$ portanto, os registradores “Endereço 01” e “Endereço 02” devem apresentar o mesmo valor. Entretanto, no caso da autocorrelação de índice 1, cada elemento do sinal é multiplicado pelo seu seguinte. Portanto, neste último e nos demais casos, se faz necessária uma estrutura em *hardware* que ajuste os endereços de leitura na RAM em função do índice de autocorrelação.

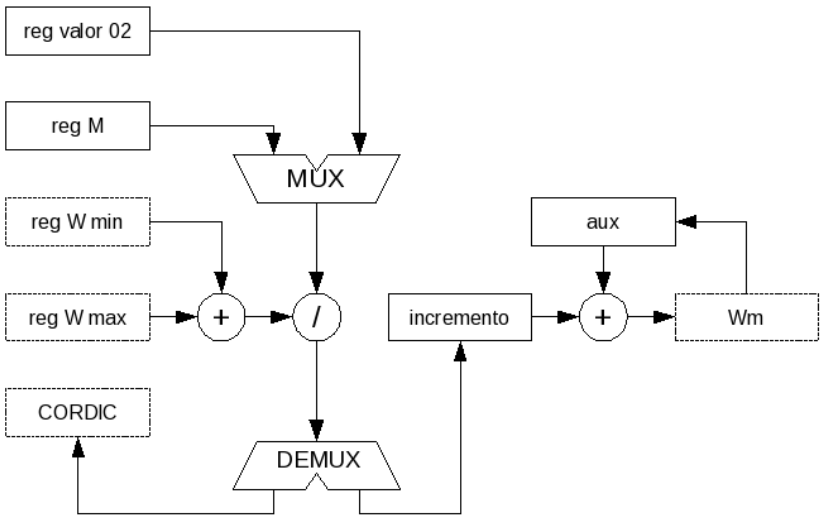
No início do processamento realizado por esse bloco, tem-se que os valores dos registrador k , “Endereço 01” e “Endereço 02” são nulos. Assim, os registradores $x(n)$ e $x(n+k)$ recebem o primeiro dado do sinal filtrado, o qual está armazenado na primeira posição da memória. Em seguida, são multiplicados entre si e o resultado é adicionado ao valor contido em “aux03”, que neste momento é nulo. Essa soma é guardada no acumulador e repassada ao registrador auxiliar “aux 03” para ser utilizada posteriormente.

Após a atualização do registrador “aux 03”, incrementa-se o “Endereço 01” e recalcula-se o “Endereço 02” por meio da soma do primeiro com o valor do registrador k , que nesse caso ainda é nulo e conseqüentemente os valores dos dois registradores de endereço serão idênticos, entretanto, diferentemente do ciclo anterior, eles apontarão para a segunda posição da memória e ao fim desse ciclo o valor do acumulador será idêntico à soma dos quadrados dos dois primeiros valores da memória que guarda o sinal filtrado. Seguindo essa operação até o final do sinal teremos no acumulador o valor da correlação de índice zero o qual é passado para a gravação em uma nova memória para ser utilizado posteriormente.

Ao término de cada cálculo de $y(k)$, o registrador “aux 01” recebe o valor do registrador k e acrescenta o valor unitário armazenado no registrador “inc” armazenando este novo valor no registrador k . O que permite o ajuste dos endereços de leitura na RAM em função do índice de autocorrelação.

3.2.4 Gerador do Espaço de Busca

O bloco Gerador do Espaço de Busca fornece dados para dois outros blocos: o CORDIC e o Estimação. Para o primeiro ele fornece a informação da frequência de ajuste inicial do ressonador digital, e para o segundo ele fornece os M valores de frequência que serão testados de forma a se verificar quais deles mais se aproximam do valor de frequência do sinal filtrado. Sua arquitetura está ilustrada na figura 3.8.



A sua primeira função é realizada por meio da soma dos registradores “reg W min” e “reg W max” seguida pela divisão cujo divisor é o valor armazenado em “reg valor 02”. O demultiplexador direciona este resultado ao bloco externo CORDIC.

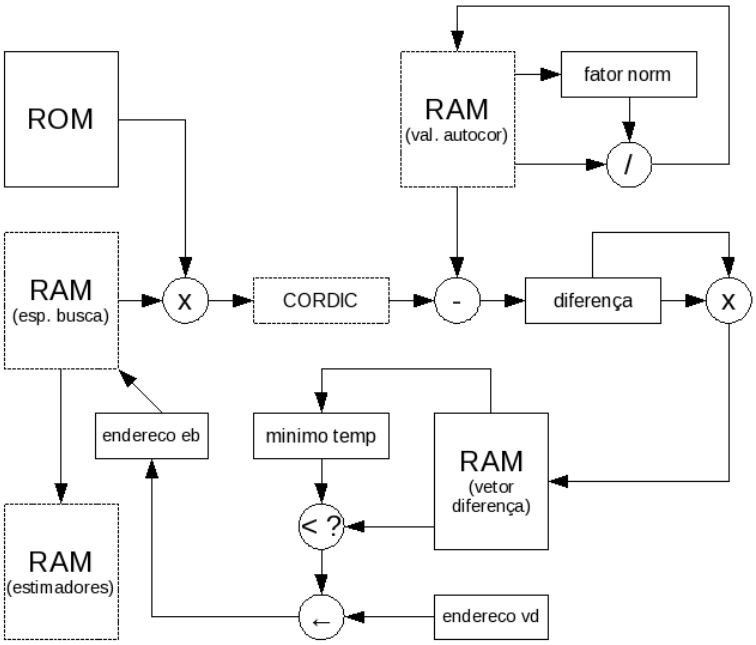
Em sua segunda etapa de processamento este bloco calcula o valor do registrador “incremento”, que será utilizado para calcular os valores das frequências formadoras do espaço de busca. Inicialmente, a unidade de controle altera a função de soma para subtração e o valor de “reg W max” é diminuído do valor de “reg W min”. Em seguida, esse valor é dividido pelo “reg M” e armazenado no registrador incremento.

O primeiro valor de frequência corresponde ao valor contido em “reg W min”, assim o registrador “Wm” é atualizado com este valor, que em seguida é repassado ao registrador auxiliar e para uma memória externa a esse bloco. Os valores de frequência seguintes são calculados por meio do processamento cíclico entre os registradores “incremento”, “aux” e “Wm”.

Nesse processamento os registradores “incremento” e “aux” são somados e seu valor é armazenado em “Wm”, o qual é repassado para a memória, bem como ao registrador auxiliar o qual será somado novamente ao incremento para gerar um novo valor de frequência.

3.2.5 Mínimos

Este bloco de processamento é a implementação em *hardware* que desempenha a função matemática descrita pela equação 2.6, ou seja, extrai os valores de frequência, pertencentes ao espaço de busca, que apresentam os menores quadrados de diferença entre os seus cossenos e os valores de autocorrelação normalizada do sinal. Sua arquitetura está ilustrada na figura 3.9.



A primeira etapa do processamento é normalizar os fatores de autocorrelação do sinal, pois o processamento do bloco autocorrelação restringe-se apenas ao cálculo de tais valores. A normalização é feita da seguinte forma: o registrador “fator norm” é atualizado com o primeiro dado armazenado na memória “val autocor”, que em termos práticos representa o valor da autocorrelação de índice zero do sinal filtrado. Em seguida, os valores armazenados nessa memória são divididos pelo fator de normalização e este resultado é armazenado na mesma posição de onde se foi extraído o dividendo. Portanto, ao fim desse processamento a memória passa a conter os valores normalizados.

Este bloco faz uso de uma memória ROM cujos valores armazenados tem uma correspondência direta com a posição da memória em que eles se encontram. Especificando melhor essa correspondência tem-se que na posição 0 de memória está contido o valor 0 em representação de ponto flutuante; a posição 1 conterá o valor 1 e assim sucessivamente.

A segunda etapa do processamento tem como objetivo preencher a RAM

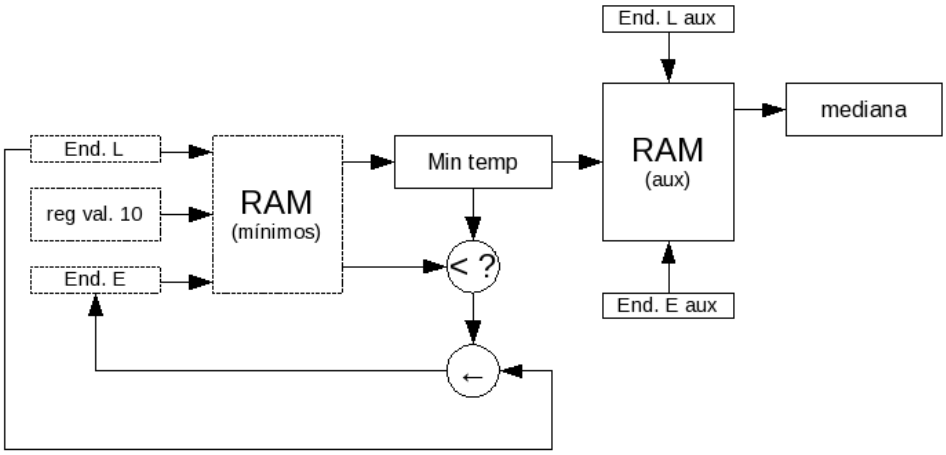
(vetor diferença), a qual armazenará os valores dos quadrados da diferença entre o cosseno das frequência do espaço de busca e a autocorrelação normalizada de determinado índice. Uma vez que para esse determinado índice a memória denominada “vetor diferença” é preenchida, o objetivo seguinte é localizar a posição dessa memória que contém o menor valor.

Localizado esse endereço basta utilizá-lo na leitura da RAM (esp. busca) para se descobrir qual a frequência do espaço de busca que gerou a menor distorção em relação ao índice de autocorrelação em questão e este valor de frequência é passado a uma nova memória denominada RAM (mínimos) a qual será acessada pelo bloco que calcula a mediana.

3.2.6 Median

O bloco que calcula a mediana é, na verdade, uma estrutura de *hardware* baseada em um algoritmo de ordenação de dados. Isso porque, uma vez que se considera um conjunto de *p* amostras, sendo *p* um número ímpar, a mediana desse conjunto é a amostra [(p – 1)/2 + 1].

Na figura 3.10 está ilustrada a arquitetura desse bloco. Inicialmente, o registrador que contém o endereço de leitura da RAM (mínimos) aponta para a primeira posição desta memória, ou seja, seu valor é zero. O dado armazenado nesta posição é repassado ao registrador “Min temp” e então se inicia o processo para a localização do menor valor contido nessa RAM.



Cada vez que o endereço de leitura “End. L” é incrementado pelo sistema de controle um novo valor da RAM (mínimos) é comparado com o valor armazenado em “Min temp”. Caso este valor seja menor, “Min temp” é atualizado com ele e o registrador do endereço de escrita “End. E” recebe o endereço onde esse valor menor está armazenado.

Ao se completar a leitura da RAM, ter-se-á o menor valor da mesma armazenado no registrador “Min temp” e o seu endereço no registrador de escrita “End. E”. O valor de “Min temp” é repassado para a primeira posição da memória “aux” e o endereço “End E aux” será incrementado para que no próximo ciclo a memória esteja pronta para o novo recebimento de dados.

Uma vez que o menor valor foi localizado e salvo em uma outra memória se faz necessário anular a posição de memória que ele se encontrava, pois o circuito continuará a sua busca pelo menor valor no próximo ciclo. A solução adotada para contornar esse problema consiste em atribuir um valor de ordem muito superior, com relação aos valores em questão, à posição do menor valor encontrado. Assim, o conteúdo do registrador “reg val. 10”, predefinidamente elevado, é gravado na RAM (mínimos) na posição indicada pelo “End. E”.

Uma vez que a posição média da RAM (aux) é ocupada o processamento é finalizado. E este valor corresponde a mediana dos valores processados.

3.2.7 J

Após o cálculo da frequência estimada do sinal, esta é utilizada para o cálculo do parâmetro *J*, o qual fornece uma indicação da similaridade entre o valor encontrado pelo sistema e valor da frequência do sinal filtrado. Seu funcionamento é bastante próximo ao descrito no bloco Mínimos, pois baseia-se também no quadrado da diferença entre o cosseno da frequência estimada e o valor de autocorrelação normalizado do sinal. Sua estrutura está ilustrada na figura 3.11.

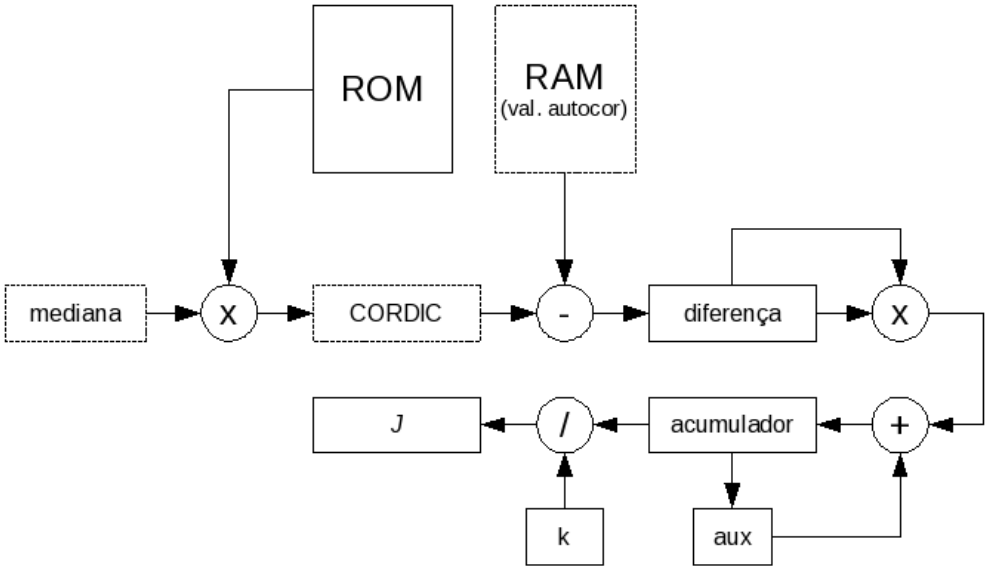


Figura 3.11: Arquitetura do bloco J.

Uma vez que esse processamento se inicia, os valores de autocorrelação armazenados já estão normalizados. O registrador “mediana” fornece o valor de frequência estimada pelo sistema, o qual é multiplicado pelo valor apontado na ROM durante o ciclo de operação. O resultado dessa multiplicação é repassado ao bloco CORDIC que calcula o cosseno, o qual é comparado com

o valor de autocorrelação apontado na RAM (val. Autocor) durante este ciclo. Em seguida é calculada a diferença entre eles e o resultado é passado ao registrador “diferença”.

O valor armazenado nesse registrador é multiplicado por ele mesmo e em seguida é somado ao valor contido em “aux”, em que considerando-se o primeiro ciclo de operação será nulo. O resultado dessa soma é armazenado no registrador “acumulador”, o qual ao começo de um novo ciclo de operação será repassado ao registrador “aux”.Na continuação, o valor do endereço das memória é incrementado e se inicia um novo ciclo de operação, o qual é finalizado após um novo cálculo ser armazenado no “acumulador”.

Uma vez que todos valores de autocorrelação tenham sido utilizados nos cálculos e o “acumulador” tenha sido atualizado é executada a divisão, em que o acumulador é o dividendo e o registrador “k” é o divisor. Este é o resultado armazenado no registrador “J” e corresponde ao parâmetro de mesmo nome.

Capítulo 4 – Resultados e Discussões

Neste capítulo são apresentadas as abordagens comparativas que serviram de referência para a avaliação de desempenho da abordagem proposta. Em seguida são apresentados os resultados e discussões são feitas acerca dos dados apurados.

4.1 Abordagens Comparativas

Na seção 4.1.1 é descrita a utilização da predição linear para estimação da frequência e a seção 4.1.2 é dedicada a descrição do algoritmo MUSIC.

4.1.1 Predição Linear

A técnica da predição linear direta consiste em obter um valor de $x(n)$ por meio da combinação linear ponderada de valores passados $x(n-1), x(n-2), \ldots, x(n-p)$. Dessa forma tem-se que:

$$\hat{x}(n) = -\sum_{k=1}^{p} a_p(k)\, x(n-k) \qquad (4.1)$$

Em que:

- $\hat{x}(n)$ é o valor linearmente predito de $x(n)$;

- $a_p(k)$ representa os pesos da combinação linear, ou seja são os coeficientes de predição;
- p é a ordem do preditor.

Assim, considerando um sinal $x(n)$ com N amostras e um preditor de 2ª ordem é possível obter os coeficientes de predição por meio da resolução do sistema da equação 4.2:

$$\begin{bmatrix} x(1) & x(2) \\ x(2) & x(3) \\ \vdots & \vdots \\ x(N-2) & x(N-1) \end{bmatrix} \begin{bmatrix} a_p(1) \\ a_p(2) \end{bmatrix} = \begin{bmatrix} x(3) \\ x(4) \\ \vdots \\ x(N) \end{bmatrix} \tag{4.2}$$

Sabe-se que um sinal senoidal pode ser gerado conforme uma equação diferença, equação 4.3, cujos coeficientes a_1 e a_2 são respectivamente $2\cos(\omega)$ e -1 . Portanto, considerando um sinal senoidal, tem-se que uma maneira relativamente simples de extrair a informação de frequência é resolver o sistema da equação 4.2, encontrar o coeficiente a_1 e a partir deste extrair o valor de ω .

$$x(n) = a_1 x(n-1) + a_2 x(n-2) \tag{4.3}$$

4.1.2 MUSIC

O algoritmo MUSIC, acrônimo de (M*ultiple Signal Classification*), é um

método de estimação da frequência baseado no subespaço do ruído. Este método utiliza-se da decomposição da matriz de autocorrelação de um sinal caracterizado por p componentes senoidais acrescidas de ruído branco [3].

Essa decomposição visa separar os autovetores em dois conjuntos. O primeiro deles, dado por $\{v_i, i=1,\ldots,p\}$ são os autovetores principais, pertencentes ao subespaço do sinal, enquanto que o segundo conjunto é formado por autovetores ortogonais aos principais, $\{v_i, i=p+1,\ldots,M\}$ e são tidos como pertencentes ao subespaço do ruído [3].

Para descrever esse método, parte-se do seguinte cálculo espectral ponderado:

$$P(f)=\sum_{k=p+1}^{M}\omega_k|\boldsymbol{s}^{\boldsymbol{H}}(f)\boldsymbol{v}_{\boldsymbol{k}}|^2 \qquad (4.4)$$

Em que:

- $\{v_k, k=p+1,\ldots,M\}$ são os autovetores no sub-espaço do ruído;
- ω_k É o conjunto de pesos positivos;
- $s(f)$ é o vetor senoidal complexo $s(f)=[1, e^{j2\pi f}, e^{j4\pi f}, \ldots, e^{j2\pi(M-1)f}]$;
- $s^H(f)$ é o hermitiano do vetor $s(f)$.

Assim, para qualquer uma das p componentes de frequência do sinal senoidal, ou seja $f=f_i$, em que $i=1,2,\ldots,p$, tem-se que $P(f_i)=0$.

O algoritmo MUSIC consiste no cálculo do inverso de $P(f)$ utilizando o

vetor peso ω_k com todos os seus valores iguais a um. O gráfico obtido a partir do inverso de $P(f)$ terá como formato característico picos posicionados sobre as componentes de frequências do sinal, dessa forma podemos expressar matematicamente como:

$$P_{MUSIC}(f)=\frac{1}{\sum_{k=p+1}^{M} |\boldsymbol{s}^{\boldsymbol{H}}(f)\boldsymbol{v}_{\boldsymbol{k}}|^2} \quad , \quad \sum_{k=p+1}^{M} |\boldsymbol{s}^{\boldsymbol{H}}(f)\boldsymbol{v}_{\boldsymbol{k}}|^2 \neq 0 \qquad (4.5)$$

4.2 Resultados e Discussões

A implementação em *software* do sistema foi realizada utilizando-se o Scilab [25]. Uma vez criados os módulos correspondentes às unidades de processamento, alguns testes de desempenho foram realizados de forma a se avaliar o sistema proposto.

O primeiro aspecto a ser abordado foi a influência do número de índices de autocorrelação na estimação da frequência. Naturalmente, quanto mais índices de autocorrelação fossem utilizados nos cálculos menor seria o erro de estimação obtido. Assim, de forma a quantificar essa relação foi realizada uma simulação cujo resultado está ilustrado na figura 4.1.

Objetivando-se isolar o efeito decorrente da utilização de uma determinada quantidade de índices de autocorrelação, foram fixados os níveis de ruído e o número de amostras do sinal a ser analisado. A relação sinal ruído utilizada neste experimento foi de 20dB e foram utilizadas 1000 amostras do sinal. Então, um sinal gerado a partir de uma única componente harmônica era fornecido à unidade de processamento, e esta estimava a sua frequência diversas vezes utilizando-se de diferentes quantidades de índices de autocorrelação. Precisamente, este procedimento foi realizado utilizando de 2

até 30 índices (conforme o eixo das abcissas - figura 4.1). Estes valores de frequência estimados eram então comparados com o valor da componente harmônica utilizada na geração do sinal e assim obtiveram-se os erro de estimação.

Na sequência, novos sinais eram gerados, suas frequências eram estimadas e os erros de estimação obtidos a partir da utilização de determinada quantidade de índices eram armazenados para que ao fim do processo fosse extraída a média dos erros calculados.

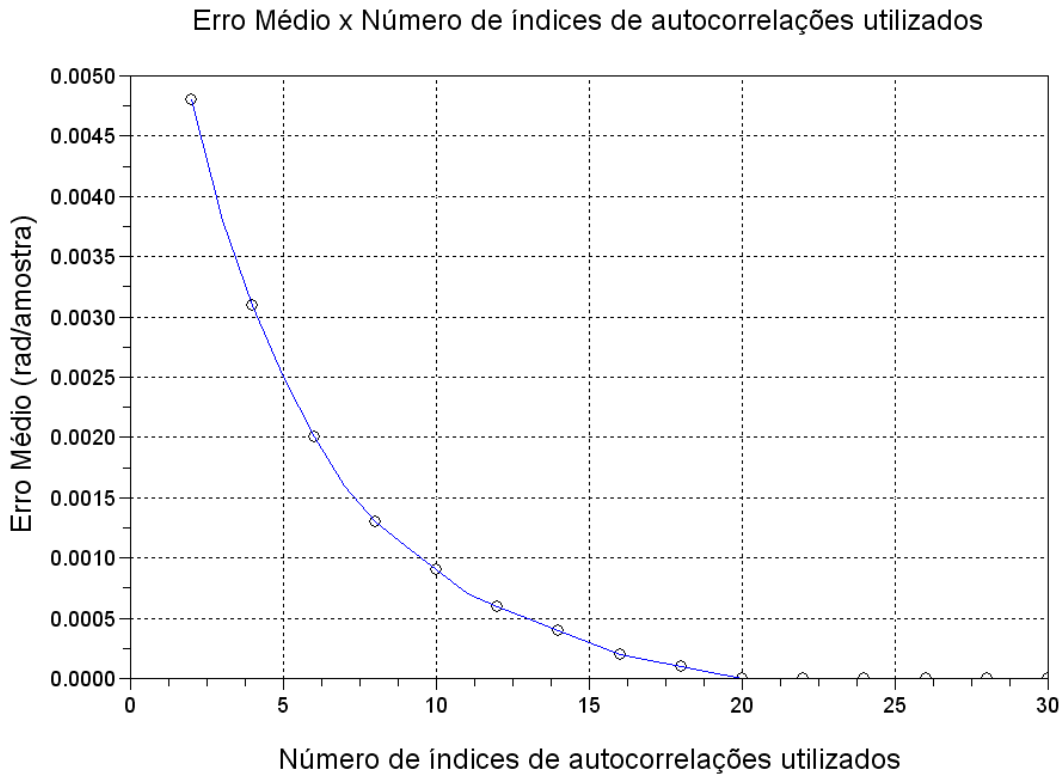


Figura 4.1: Avaliação do Erro médio com relação ao número de índices de autocorrelação utilizados.

Deste gráfico obtido pode-se verificar que a utilização de 20 índices de autocorrelação, considerando a relação sinal ruído e os números de amostras do sinal pré-estabelecidos anteriormente, propicia um erro médio muito próximo de zero.

Uma segunda avaliação foi realizada com o objetivo de verificar a influência entre o número de amostras de sinal utilizadas e o erro de estimação. Para tal um procedimento similar ao da primeira análise foi realizado.

Nesta simulação foram fixados a quantidade de índices de autocorrelação, especificamente 20 índices, como também os níveis de ruído – SNR igual a 20 dB. Na sequência, sinais foram gerados com diferentes tamanhos e suas frequências foram estimadas produzindo erros diferentes, estes relacionados ao número de amostras utilizadas nos cálculos. Após vários sinais serem avaliados e vários erros obtidos, estes foram utilizados para se encontrar o erro médio produzido. Esta relação está ilustrada na figura 4.2.

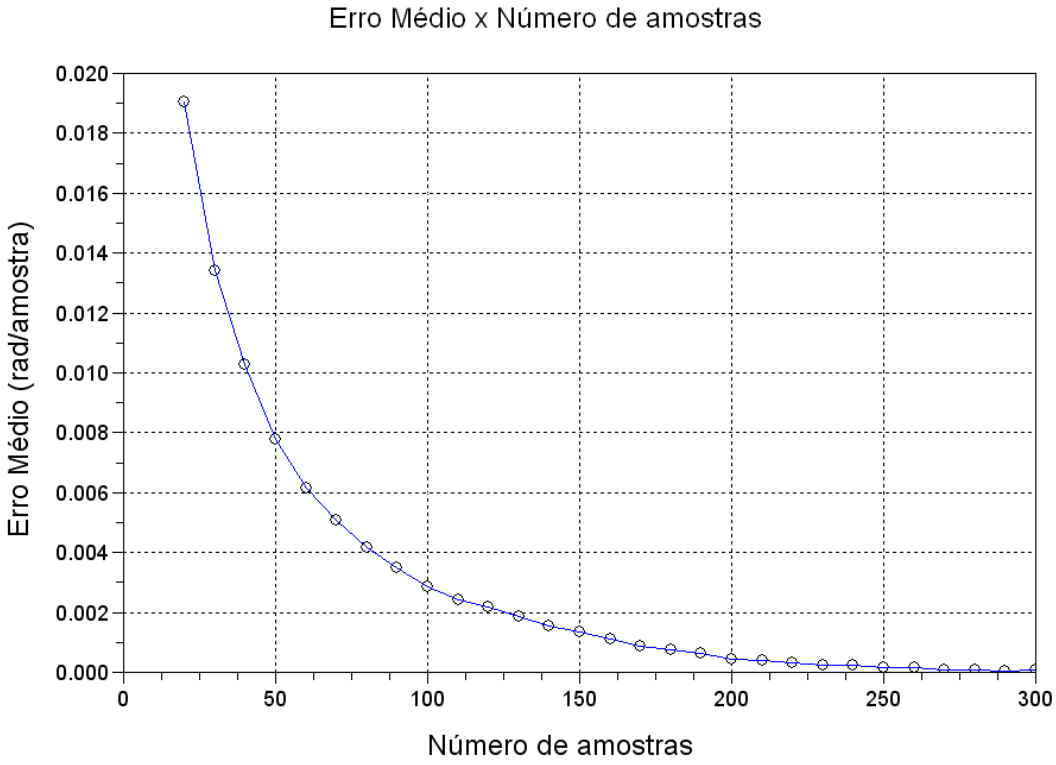


Figura 4.2: Avaliação do Erro médio com relação ao número de amostras utilizadas.

Uma terceira avaliação de desempenho realizada teve como objetivo

observar a robustez do método com relação ao ruído. Para isso foi feita uma simulação, na qual, ruído foi adicionado a um sinal de uma única componente harmônica, obedecendo a uma determinada SNR e em seguida este sinal "corrompido" era processado e sua frequência era estimada. Este valor era comparado ao valor da frequência do sinal puro, obtendo-se assim o erro da estimação.

Para um determinado valor da SNR esse procedimento foi realizado diversas vezes e na sequência foi extraído o erro médio de estimação. Nesta mesma situação, de forma a comparar os desempenhos, os sinais foram processados utilizando-se o MUSIC e o preditor linear. Os resultados obtidos estão ilustrados na figura 4.3.

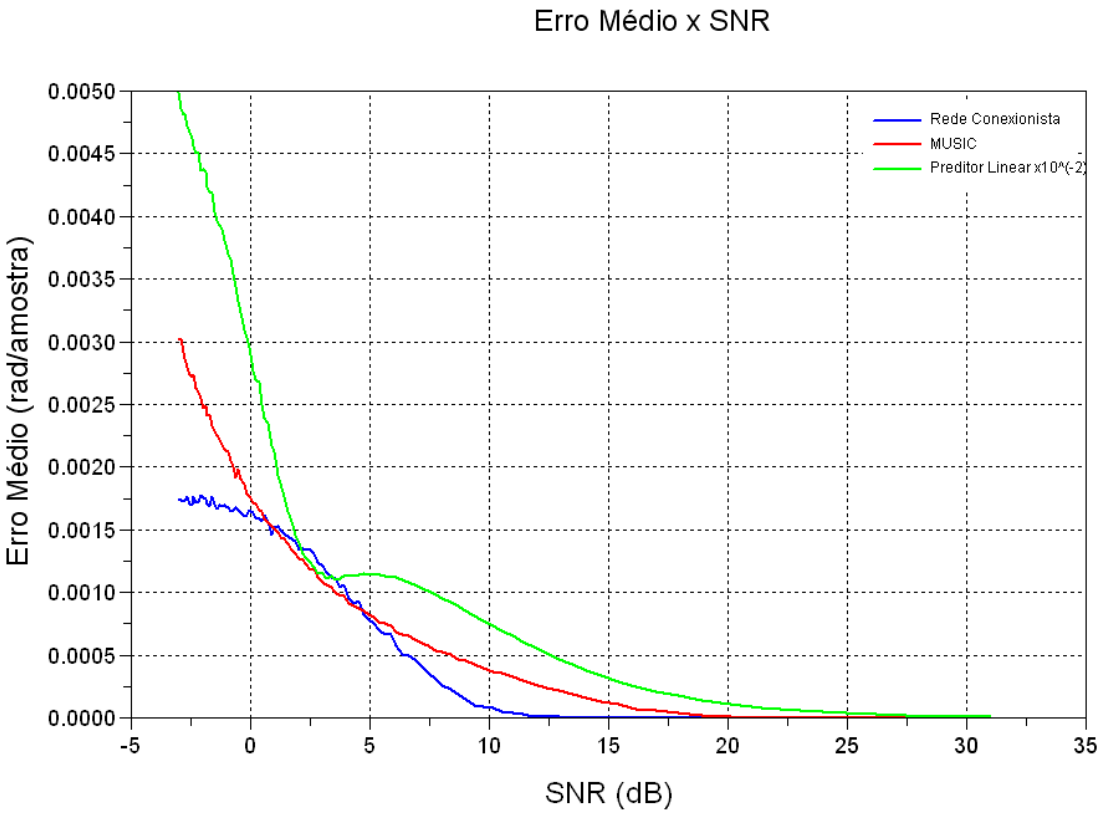


Figura 4.3: Avaliação do Erro médio com relação à SNR.

Uma primeira observação a ser feita sobre os resultados ilustrados na figura 4.3 é que a utilização da abordagem proposta permitiu que o erro médio

da estimação fosse praticamente anulado a um valor da SNR inferior em relação às outras abordagens.

No intervalo de [0,5] dB, valores da SNR, os desempenhos da abordagem proposta e do algoritmo MUSIC foram similares, tendo este último apresentado um desempenho ligeiramente superior. Uma consideração a ser feita é que na utilização do algoritmo MUSIC, a quantidade de componentes harmônicas presentes no sinal analisado foi previamente indicada, contrapondo-se à abordagem conexionista, na qual esse procedimento não foi realizado.

Computacionalmente, o MUSIC é mais complexo que a rede proposta, uma vez que são necessárias a manipulação da matriz de autocorrelação e a decomposição desta para a extração dos seus autovetores, para então se estabelecer a equação 4.5 que é utilizada no cálculo da estimação das frequências.

Comparativamente ao método proposto, a utilização de preditores lineares também requer um custo computacional superior, pois envolvem cálculos com inversão das matrizes dos sinais. Mesmo assim o desempenho obtido com a utilização dessa abordagem foi inferior com relação às outras.

Uma segunda avaliação da robustez com relação ao ruído está ilustrada na figura 4.4. Nela foram utilizados os mesmos procedimentos e sinais da avaliação anterior, diferenciando apenas o intervalo de valores da SNR a serem utilizados. Para esta avaliação a abordagem proposta foi comparada apenas com o método MUSIC, pois com esses valores de SNR, o sinal corrompido gerava matrizes não inversíveis, inviabilizando a utilização dos preditores.

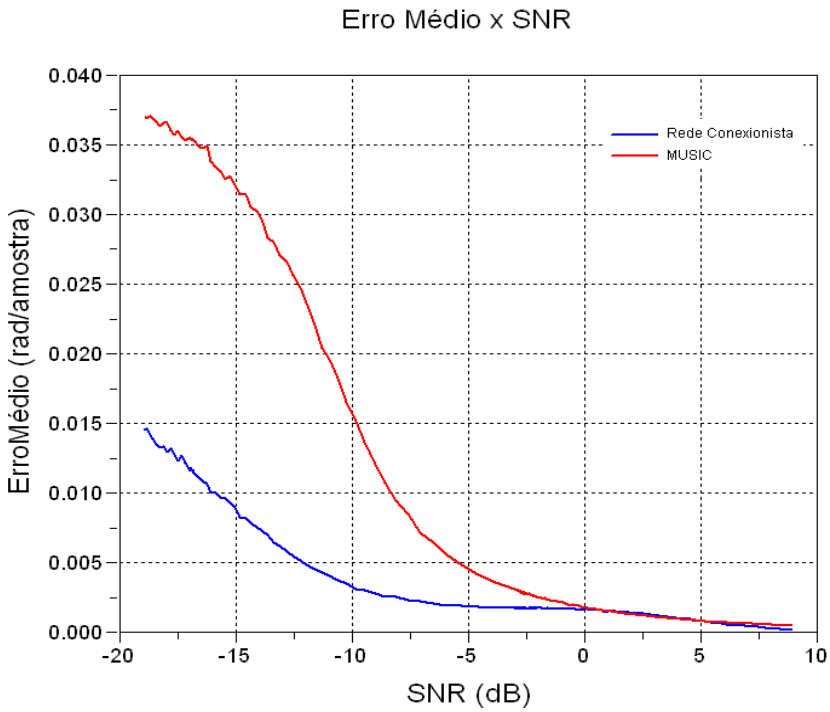


Figura 4.4: Avaliação do Erro médio com relação à SNR.

Pode-se observar que para esses níveis de SNR o desempenho da rede conexionista é superior ao desempenho do MUSIC. Sendo estes resultados obtidos com uma menor carga computacional e consequentemente, em menos tempo de processamento.

Capítulo 5 – Conclusões e Trabalhos Futuros

Neste capítulo são apresentadas as conclusões extraídas a partir deste trabalho de dissertação como também são propostos trabalhos a serem realizados a partir do que foi desenvolvido.

5.1 Conclusões

Neste trabalho foi proposto o desenvolvimento de uma rede conexionista capaz de estimar componentes de frequências contidas em um sinal digital. Inicialmente a rede foi implementada, testada e aprimorada em *software* para que em seguida fosse implementada em *hardware* por meio da utilização de um FPGA.

Por meio dos resultados obtidos pôde-se verificar sua robustez ao ruído, relativa simplicidade computacional de implementação e redução no tempo de processamento, quando comparada aos outros métodos, atestando-se assim um bom desempenho na estimação das frequências componentes do sinal analisado.

Este trabalho desenvolvido pode ser aplicado no projeto de pesquisa e desenvolvimento da CHESF, entitulado "Pára-raios a ZnO: Monitoramento remoto e ferramenta de apoio ao diagnóstico de falhas."

Especificamente, na ferramenta de apoio ao diagnóstico de falhas em

pára-raios de óxido de zinco, uma etapa do processamento da informação requer a estimação das componentes harmônicas da corrente de fuga do pára-raio, assim este método proposto pode ser utilizado para a realização dessa tarefa.

5.2 Trabalhos Futuros

A partir do projeto desenvolvido é possível fazer algumas considerações com relação aos possíveis trabalhos futuros.

Neste trabalho foi proposto o desenvolvimento de uma rede conexionista capaz de estimar componentes de frequências contidas em um sinal digital. Propõe-se como um dos trabalhos futuros o aprimoramento das unidades de processamento para que sejam capazes de realizar estimação de amplitude e fase dos sinais.

Como outra possibilidade, uma vez que grande parte do *hardware* está descrito em Verilog, é a adaptação dos blocos para se tornarem o mais independente possível da plataforma de testes do FPGA de forma permitir a fabricação de ASIC.

Anexo 1 - Ressonador Digital

Um ressonador digital é um filtro passa-faixa especial, com um par de pólos complexos conjugados localizados próximo do círculo unitário como mostrado na figura A1.1.(a). A magnitude da resposta em frequência está ilustrada na figura A1.1.(b). A posição angular dos pólos determina a frequência de ressonância do filtro. O nome ressonador refere-se ao fato que o filtro tem uma larga magnitude de resposta, isto é, ele ressona nas adjacências onde o pólo está localizado. Ressonadores digitais são úteis em muitas aplicações, incluindo síntese de voz e filtragens passa-faixa simples.

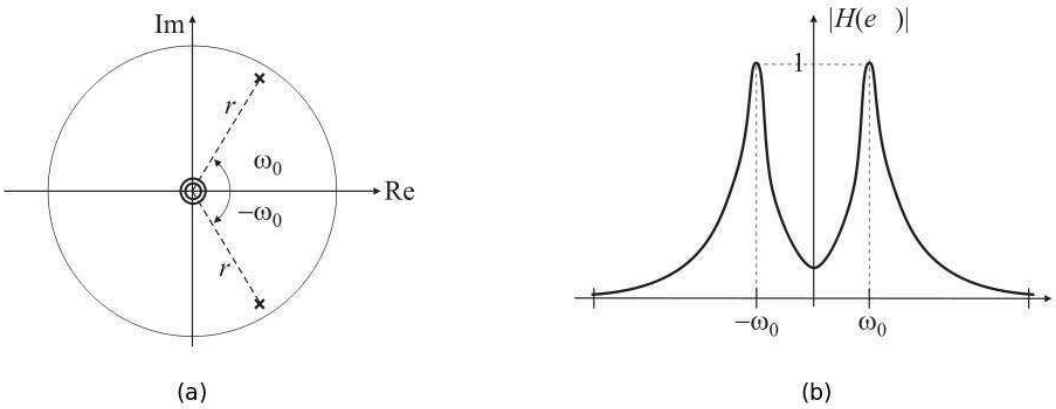


Figura A1.1: a) Localização do par de pólos de um Ressonador Digital;
b) Magnitude da resposta em frequência do Ressonador Digital.

No projeto de um ressonador digital com um pico de ressonância em $\omega = \omega_0$ seleciona-se os pólos complexos conjugados:

$$p_{1,2} = r\,\mathrm{e}^{\pm j\omega_0} \qquad 0 < r < 1 \tag{A1.1}$$

Ademais, pode-se selecionar dois zeros. Embora haja várias possibilidades de escolha, dois casos são de interesse em especial. Um escolha é colocar os zeros na origem e a outra é colocá-los em $z=\pm 1$ Esta escolha eliminam completamente a resposta do filtro para as frequências $\omega=0$ e $\omega=\pi$ e é útil em muitas aplicações práticas.

A função de sistema do ressonador digital com zeros na origem é dada pela equação :

$$H(z)=\frac{b_0}{(1-r\,\mathrm{e}^{+j\omega_0}z^{-1})(1-r\,\mathrm{e}^{-j\omega_0}z^{-1})} \qquad \text{(A1.2)}$$

Manipulando algebricamente a equação A1.2, tem-se:

$$H(z)=\frac{b_0}{1-(2\mathrm{r}\cos\omega_0)\,z^{-1}+r^2z^{-2}} \qquad \text{(A1.3)}$$

Uma vez que $|H(\omega_0)|$ tem seu pico em $\omega=\omega_0$ pode-se selecionar o ganho b_0 de maneira que $|H(\omega_0)|=1$. Da equação A1.2 pode-se obter:

$$H(\omega_0)=\frac{b_0}{(1-r\,\mathrm{e}^{+j\omega_0}\mathrm{e}^{-j\omega_0})(1-r\,\mathrm{e}^{-j\omega_0}\mathrm{e}^{-j\omega_0})} \qquad \text{(A1.4)}$$

Manipulando algebricamente a equação A1.4, tem-se:

$$H(\omega_0)=\frac{b_0}{(1-r)(1-r\mathrm{e}^{-2j\omega_0})} \tag{A1.5}$$

Igualando o módulo dessa função a unidade, obtem-se o fator de normalização desejado:

$$b_0=(1-r\sqrt{1+r^2-2\mathrm{r}\cos 2\,\omega_0}) \tag{A1.6}$$

Na figura A2.2 tem-se ilustrado o comportamento da magnitude de frequência do Ressonador Digital à medida em que se aproxima os seus pólos do cículo unitário.

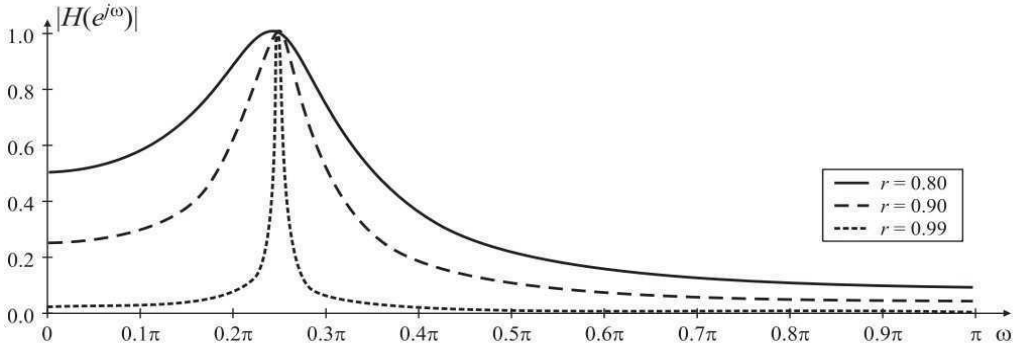


Figura A1.2: Resposta em frequência de um Ressonador Digital com zeros na origem.

Anexo 2 – Código dos Programas em Scilab

Descrição da função do ressonador digital:

```
// Universidade Federal de Campina Grande
// Centro de Engenharia Elétrica e Informática
// Pós-Graduação em Engenharia Elétrica
// Laboratório de Instrumentação e Metrologias Científicas
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

function y=filtro_ressonador(x,w)
   r=0.99;

   // Inicializa as memórias do filtro:
   y2=0;
   y1=0;

   // Procedimento de filtragem
  for n=1:length(x)
    y(n)=x(n)+2*r*cos(w)*y2-r*r*y1;
    y1=y2;
    y2=y(n);
  end
endfunction
```

Descrição da função que calcula a autocorrelação:

```
// Universidade Federal de Campina Grande
// Centro de Engenharia Elétrica e Informática
// Pós-Graduação em Engenharia Elétrica
// Laboratório de Instrumentação e Metrologias Científicas
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

function r=autocovar(y,N)
  y=y-mean(y);
  r(1)=sum(y.*y);
  L=length(y);

  for n=2:N
    r(n)=sum(y(1:$-n+1).*y(n:$));
    r(n)=r(n)/r(1);
  end
  r(1)=1;
endfunction
```

Descrição da função MUSIC:

```
// Universidade Federal de Campina Grande
// Centro de Engenharia Elétrica e Informática
// Pós-Graduação em Engenharia Elétrica
// Laboratório de Instrumentação e Metrologias Científicas
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

function [w_music]=MUSIC(M,p,x)

  // Gerando a matrix de autocorrelação:
    N=length(x);
    x=x-mean(x);

    for n=1:M
      X(n:N+n-1,n)=x(:);
    end
    Rxx=(1/N)*X'*X;

  // Extraindo os autovalores e autovetores
    [autovetores,autovalores] = spec(Rxx);
    [lambda,ind_lambda]=sort(diag(autovalores));

  // MUSIC
      N= autovetores(:,ind_lambda((2*p+1):M)); //autovetores associados aos
menores autovalores

  w_music = [];
  escala=0:0.0001:%pi;
  k=1;
  for w=0.65:0.001:0.75
    s=exp(%i*(0:size(N,1)-1)*w);
    P(k)=1/sum(abs(s*N).^2);
    freqAng(k)=w;
    k=k+1;
  end

Paux=P;
margem=round(length(freqAng)*0.03/2); // 3% do range de freq.
[val,pos]=max(Paux);

for k=1:p
  w_music(k) = freqAng(pos);

  ini = pos-margem;
  if ini < 1
    ini=1;
  end
```

```
  fim = pos+margem;
  if fim > length(freqAng)
    fim=length(freqAng);
  end

  Paux(ini:fim)=0;
  [val,pos]=max(Paux);

end // do if

endfunction
```

Descrição da função que utiliza os preditores lineares:

```
// Universidade Federal de Campina Grande
// Centro de Engenharia Elétrica e Informática
// Pós-Graduação em Engenharia Elétrica
// Laboratório de Instrumentação e Metrologias Científicas
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

function [r,w,perdido]=preditor_linear(s_f)

  perdido = %F;
  s_f = s_f(:);
  M = [s_f(1:length(s_f)-2) s_f(2:length(s_f)-1)];
  P = [s_f(3:length(s_f))];
  N=M'*M; // Matriz 2 x 2
  D=N(1,1)*N(2,2)-N(1,2)*N(2,1);
  if D < 1e-6
    r=1;
    w=0; // pois a frequência do sinal é aprox. nula (sinal DC)
  else
    invN = (1/D)*[N(2,2) -N(1,2); -N(2,1) N(1,1)];
    ab = invN*M'*P;
    if ab(1) >= 0
      r = 1;
      w = 0;
      perdido = %T;
    else
      r = sqrt(-ab(1));
      teste = ab(2)/(2*r);
      if (abs(teste)<=1)
         w = acos(ab(2)/(2*r));
      else
```

```
        w = 0;
        perdido = %T;
      end
    end

  end
endfunction
```

Descrição da unidade de processamento:

```
// Universidade Federal de Campina Grande
// Centro de Engenharia Elétrica e Informática
// Pós-Graduação em Engenharia Elétrica
// Laboratório de Instrumentação e Metrologias Científicas
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

function [we,SCD]=dec(s,wmin,wmax)

  // Filtra o sinal:
  we=(wmin+wmax)/2;
  y=filtro_ressonador(s,(wmin+wmax)/2);


  N=10;

  r=autocovar(y,N);
  // Definição dos valores discretos de frequência onde a busca será feita
      w=wmin:(wmax-wmin)/20:wmax;

  // Encontra N-1 frequências que minimizam cada uma das funções de custo:
      for n=1:(length(r)-1)
        Jaux=(cos(n*w)-r(n+1)).^2;
        [val,pos]=min(Jaux);
         we(n)=w(pos);
      end
  // Escolhe a frequência mediana como melhor estimador hipotético:
    we=median(we);

  // Testa se esta escolha reduz a distorção em relação à autocorrelação ideal
    SCD=mean(  ( cos((1:N-1)'*we)-r((2:N)) ).^2  );


// De forma cíclica,m o sinal original é filtrado, buscando uma ressonância
// em we, e o we é re-estimado
// Isso corresponde a testar a hipótese de existência de uma componente
```

```
// harmônica em we, através de um ressonador.

for cont=1:3                  // 3 Ciclos -> Valor arbitrário escolhido empiricamente
    y=filtro_ressonador(s,we);
    r=autocovar(y);
        for n=1:(length(r)-1)
          Jaux=(cos(n*w)-r(n+1)).^2;
          [val,pos]=min(Jaux);
           we(n)=w(pos);
        end
    we=median(we);

    SCD=mean(  ( cos((1:N-1)'*we)-r((2:N)) ).^2  );

end

if (we==w(1) | we==w($)) // Mínimo encontrado nas bordas do intervalo
  SCD = 1;
end

endfunction
```

Descrição da função que forma a rede:

```
// Universidade Federal de Campina Grande
// Centro de Engenharia Elétrica e Informática
// Pós-Graduação em Engenharia Elétrica
// Laboratório de Instrumentação e Metrologias Científicas
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

function [wdetectados]=rede(s)
// Escolhas arbitrárias de inicialização:
   dw=0.01;                           //Largura (banda) do domínio de cada unidade
   wmin=0.05:0.005:(%pi/2 - dw);  // Inicialização determinística de posições
   wmax=wmin+dw;
   Limiar = 0.001;

// Entrega o sinal para análise independente de cada unidade:
for k=1:length(wmin)
  [we(k),SCD(k)]=dec(s,wmin(k),wmax(k));

end

// Retorna apenas os harmônicos detectados com SCD acima do limiar.
wdetectados=we(find(SCD<Limiar));
```

```
// As freq. detectadas (em rad) são ordenadas e retornadas:
wdetectados=sort(wdetectados);
wdetectados=wdetectados($:-1:1);
endfunction
```

Descrição do ambiente de simulação da REDE:

```
// Universidade Federal de Campina Grande
// Centro de Engenharia Elétrica e Informática
// Pós-Graduação em Engenharia Elétrica
// Laboratório de Instrumentação e Metrologias Científicas
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

clear;
clc;
clf();
tic();

chdir('/home/alanvss/Desktop/workdir_diss'); //comentar caso rode em windows
exec('filtro.sci');
exec('autocovar.sci');
exec('dec.sci');
exec('music.sci');
exec('rede.sci');
// ---------------------------------------------------------------------------------------

w = [0.55 0.6 0.85];
n = 200;
s = [];
for i=1:length(w)
  s = s + sin(w(i)*[0:n]);
end


Ar = 0.5; //amplitude do ruido;
sr = s + Ar*rand(s,'normal');


[wdetectados]=rede(sr);

liminf = 0.025;
limsup = 1.575;
dlim = 0.05;
//liminf = 0.005;
```

```
//limsup = 1.555;
//dlim = 0.01;

classes = [liminf:dlim:limsup];

classeso = [];

for i=1:(length(classes)-1)
  for j=1:length(wdetectados)
    if(wdetectados(j) > classes(i))
      if(wdetectados(j) < classes(i+1))
        classeso = [classeso ((classes(i)+ classes(i+1))/2)];
      end
    end
  end
end
freqs = tabul(classeso)
plot2d(freqs(1:$,1),freqs(1:$,2),-2,rect=[0,0,1.6,20]);
xgrid;
scf(1);
plot(s);
plot(sr,'r');


f = toc()
```

Referências

[1] OPPENHEIM, A. V.; WILLSKY, A. S.; HAMID, S.: “Signal and Systems”, 2nd edition. New Jersey: Prentice-Hall Inc, 1999.

[2] OPPENHEIM, A.V.; SCHAFER, R.W.; BUCK J.R.: “Discrete-Time Signal Processing”, 2nd edition. New Jersey: Prentice-Hall Inc, 1997.

[3] PROAKIS, J. G.; MANOLAKIS D. G.: “Digital Signal Processing: Principles, Algorithms and Applications”, 3rd edition. New Jersey: Prentice-Hall Inc, 1995.

[4] HAYKIN, S.: “Neural Networks: A Comprehensive Foundation”, 2nd edition. New Jersey: Prentice-Hall, 1998.

[5] MEDLER, D. A.: “A Brief History of Connectionism”, Neural Computing Surveys, 1998.

[6] BILGIN, G.; ALTUN, O.: "Cardiac Problem Diagnosis with Statistical Neural Networks and Performance Evaluation by ROC Analysis," Applied Electronics, 2006. AE 2006. International Conference on Applied Electronics, vol., no., pp. 15-18, 6-7 Sept. 2006.

[7] WHITE, H.: “Economic prediction using neural networks: The case of IBM daily stock returns.” In Proceedings of the IEEE International Conference on Neural Networks, pp 451–459, San Diego, 1988.

[8] KING, S. Y.; HWANG, J. N.: “Neural network architectures for robotic applications,” Robotics and Automation, IEEE Transactions on Robotics and Automation, vol.5, no.5, pp.641-657, Oct 1989.

[9] IEEE Standard #754-2008.

[10] VOLDER, J. E.: “The CORDIC trigonometric computing technique.” , IRE

Transactions on electronic computers, pp. 330-334, September 1959.

[11] WALTHER, J. S.: A unified algorithm for elementary functions. Proceedings of the AFIPS Spring Joint Computer Conference, pp. 379-385, 1971.

[12] Van der Kolk, K.J.; Lee, J.A.; Deprettere, E.F.A., "A fast-rotations based floating point vectoring algorithm," Signal Processing Systems, 1999. SiPS 99. 1999 IEEE Workshop on Signal Processing Systems, vol., no., pp.231-240, 1999.

[13] Ray Andrak, ”A survey of CORDIC algorithms for FPGA based computers”, Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays, pp. 191-200, 1998.

[14] Minghua He; Chong Chen; Xueying Zhang, "FPGA Implementation of a Recursive Rank One Updating Matrix Inversion Algorithm for Constrained MPC," Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on Intelligent Control and Automation, vol.1, no., pp.733-737.

[15] R. Ackner, T. Kailath, ”The Schur algorithm for matrix-valued meromorphic functions”, SIAM J. Matrix Anal. Appl., Vol. 15, pp. 140-150, 1994.

[16] Stephen D. Brown, Robert J. Francis, and Jonathan Rose. Field-Programmable Gate Arrays. Boston: Kluwer Academic Publishers, 1992.

[17] B.Schoner, J. Villasenor, S. Molloy and R. Jain, ”Techniques for FPGA implementation of video compression systems”, Proceedings of the 1995 ACM third international symposium on Field programmable gate arrays, February 1995.

[18] Dick, C.; Harris, F., "FPGA signal processing using sigma-delta modulation," Signal Processing Magazine, IEEE , vol.17, no.1, pp.20-35, Jan 2000.

[19] Waxman, R.; Aylor, J.H.; Marschner, E., "The VHSIC hardware description language (IEEE standard 1076): language features revisited," Compcon Spring '88. Thirty-Third IEEE Computer Society International Conference, Digest of Papers , pp.310-315, 29 Feb-3 Mar 1988.

[20] Ashenden, P.J., "Modeling digital systems using VHDL," Potentials, IEEE , vol.

17, no.2, pp.27-30, Apr/May 1998.

[21] ASHENDEN, P. J. The VHDL cookbook. Dept. Computer Science, University of Adelaide, South Australia. July 1990.

[22] Midorikawa, E. T., "Uma Introdução às Linguagens de Descrição de Hardware", EPUSP, 2007.

[23] HYDE, D. C. Handbook on Verilog HDL. Computer Science Department, Bucknell University. August 1995.

[24] THOMAS D. E.; MORRBY P. R.: “The Verilog Description Language”,2nd edition. New York: Springer, 2008.