MÔNICA LOREA MATZENAUER

IMPLEMENTAÇÃO DE ARQUITETURA DEDICADA DE FILTRO ADAPTATIVO EM CODIFICAÇÃO HÍBRIDA UTILIZANDO O ALGORITMO LMS

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre em Ciência da Computação pela Universidade Católica de Pelotas

Orientador: Prof. Dr. Sérgio José Melo de Almeida
Co-orientador: Prof. Dr. Eduardo Antônio César da Costa

Pelotas

2011

MÔNICA LOREA MATZENAUER

IMPLEMENTAÇÃO DE ARQUITETURA DEDICADA DE FILTRO ADAPTATIVO EM CODIFICAÇÃO HÍBRIDA UTILIZANDO O ALGORITMO LMS

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre em Ciência da Computação pela Universidade Católica de Pelotas

Conceito e/ou Nota_____

BANCA EXAMINADORA:

_____
Presidente e Orientador Prof. Dr. Sérgio José Melo de Almeida

_____
Co-orientador Prof. Dr. Eduardo Antônio César da Costa

_____
1º Examinador Prof. Dr. João Batista Martins

_____
2º Examinador Prof. Dr. Lisane Brisolara

Pelotas, 25 de Março de 2011.

"Dói, de tanto medir a distância,
saber que não vou te tocar além da lembrança.
A tua falta é sol sem calor: está aqui, mas se foi.
Virou estrela, a nossa estrela do céu."
(Beto Guedes)

AGRADECIMENTOS

Agradeço à minha mãe, meu exemplo de vida, fonte de inspiração, apoio e ensino diário, que sempre me deu amor e força e me estimulou a dar este grande passo.

Aos meus familiares, por estarem ao meu lado me encorajando nas horas difíceis e me aplaudindo nos momentos de glória, sempre valorizando meus potenciais.

Aos meus amigos, que estiveram sempre presentes, me incentivando, agradeço pelo companheirismo, carinho e compreensão.

Aos meus colegas de laboratório, em especial ao Gustavo e ao João, pela agradável convivência, doando sabedoria, auxílio e sugestões, ajudando na conclusão deste trabalho.

Um especial agradecimento aos meus professores: orientador Prof. Dr. Sérgio José Melo de Almeida e co-orientador Prof. Dr. Eduardo Antônio César da Costa, profissionais corretos e competentes, pelo incentivo e ajuda ao longo desta jornada. Obrigada pela oportunidade de crescimento, aprendizado e pela confiança em mim depositada. Seu apoio, dedicação e amizade foram fundamentais para o desenvolvimento deste trabalho.

Agradeço a todas as pessoas que, direta ou indiretamente, contribuíram para a execução desta dissertação de mestrado.

RESUMO

Este trabalho tem como proposta a implementação de uma arquitetura de *hardware* dedicada para o algoritmo LMS (*Least Mean Square*) de filtragem adaptativa, para o cancelamento de interferências em codificação Híbrida. No esquema utilizado, a partir de um sinal de referência de 60Hz, o algoritmo estima as harmônicas superiores, utilizando esses resultados para o cancelamento da interferência associada ao sinal de interesse. Um dos métodos para a redução da atividade de chaveamento em barramentos de dados que tem sido amplamente utilizado é a codificação de dados. Neste trabalho, a arquitetura de filtragem adaptativa proposta utiliza em seus barramentos de dados a codificação Híbrida, cuja idéia é dividir os operandos em grupos de *m* bits, codificar cada grupo utilizando o código Gray (que habilita reduções na atividade de chaveamento dentro de cada grupo) e utilizar o comportamento do código Binário para propagar o *carry* entre os grupos. Dessa forma, são desenvolvidas arquiteturas otimizadas de circuitos multiplicadores *array* base 2^m na codificação Híbrida para a aplicação na arquitetura dedicada de filtro adaptativo. São implementados circuitos multiplicadores *array* de 18, 23 e 36 bits na codificação Híbrida na base 4 (*m*=2), bem como um caso particular para a base 8 (*m*=3). Essas arquiteturas são implementadas em linguagem de descrição de *hardware*. Os principais resultados mostraram que os multiplicadores Híbridos apresentaram, em alguns casos, menor consumo de potência em relação aos multiplicadores binários. Além disso, foi possível validar e comparar as arquiteturas de filtro adaptativo nas codificações Binária e Híbrida, onde se pôde verificar a eficiência dos filtros para o cancelamento de interferências em ambas as codificações, mostrando-se possível a implementação de um filtro adaptativo em codificação Híbrida.

Palavras-chave: multiplicadores *array*, filtro adaptativo, LMS, codificação híbrida

ABSTRACT

This work proposes the implementation of dedicated hardware architecture for the Least Mean Square (LMS) adaptive filtering algorithm by using Hybrid encoding, whose main goal is to cancel the interferences in the signal of interest. In the used scheme, from a 60Hz reference signal, the algorithm is able to estimate the superior harmonics, using after these results for the cancelling of interferences related to the signal of interest. One of the techniques that is widely used for the switching activity reduction uses signal encoding. In this work, the proposed adaptive filtering architecture uses the Hybrid encoding in its data buses, whose main idea is to split the operands in group of *m*-bits, encode each group using the Gray code (that potentially enables reduction of the switching activity into each group) and propagate the carry between the groups as in the Binary encoding. We developed new Hybrid multipliers for signed multiplication, which uses radix-2^m encoding. The multipliers are applied to the adaptive filtering architecture. We have implemented 18, 23 and 36 bit-width radix-4 Hybrid array multipliers, as well as a particular case for the radix-8 (*m*=3) operation. The main results showed that the Hybrid multipliers are more efficient than the Binary ones, by presenting less power consumption in some cases. Moreover, the implemented adaptive filtering architectures were validated and compared in both Binary and Hybrid encoding. The efficiency of the implemented filters for the cancelling of interferences was proved by using both encoding scheme. By the presented results, we conclude that it could be practicable to implement an adaptive filtering architecture operating on Hybrid encoding.

key-words: array multipliers, adaptive filter, LMS, hybrid coding

LISTA DE ABREVIATURAS E SIGLAS

DSP – *Digital Signal Processing* (Processamento Digital de Sinal)
FPGA – *Field- Programmable Gate Array* (Matriz de portas programáveis em campo)
ASICs – *Application Specific Integrated Circuit* (Circuito Integrado de Aplicação Específica)
LMS – *Least Mean Square Algorithm* (Algoritmo dos mínimos quadrados)
NLMS – *Normalized Least Mean Square Algorithm* (Algoritmo dos mínimos quadrados normalizado)
FIR – *Finite Impulse Response* (Resposta Finita ao Impulso)
IIR – *Infinite Impulse Response* (Resposta Infinita ao Impulso)
VHDL – *Very High speed integrated circuit Hardware Description Language* (Linguagem de descrição de *hardware* para projetos de circuitos de altíssima velocidade)
FFT – *Fast Fourier Transform* (Transformada Rápida de Fourier)
CSA – *Carry Save Adder*
FA – *Full Adder*
HA – *Half Adder*
ECG – Eletrocardiograma
EEG – Eletroencefalograma
MLAEP - *Midlatency Auditory Evoked Potentials* (Potenciais Evocados Auditivos de Média Latência)
ALUT – Blocos Lógicos Combinacionais
EMQ – Erro Médio Quadrático
SIS – *Synthesis Interchange Logic*

LISTA DE TABELAS

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Uma classe de algoritmos que tem merecido especial atenção nos últimos anos, com aspectos de alto desempenho e baixo consumo de potência, é a de processamento digital de sinais (*Digital Signal Processing* – DSP). Esse interesse tem aumentado desde a proliferação de equipamentos portáteis de alto desempenho operados por bateria, tais como telefones celulares, câmeras de vídeo, equipamentos biomédicos, etc. que utilizam internamente vários algoritmos da área DSP, tais como filtros digitais e algoritmos da Transformada Rápida de Fourier (*Fast Fourier Transform* – FFT). Dessa forma, torna-se importante o projeto desses algoritmos em *hardware*, de tal forma que as suas operações se tornem ainda mais eficientes.

Neste trabalho o principal foco é a implementação, em *hardware* dedicado, de filtro adaptativo. As características desejáveis para um filtro adaptativo são a habilidade para operar em um ambiente desconhecido e seguir as variações no tempo dos sinais de entrada. No presente estudo, foi utilizado o algoritmo adaptativo LMS (*Least Mean Square*), que, por sua simplicidade e robustez, é muito utilizado em processamento de sinais (HAYKIN, 1996; MANOLAKIS, 2005).

O problema de implementação de arquiteturas dedicadas de filtros digitais, utilizadas no processamento digital de sinais, tem sido amplamente estudado desde a década passada (MEHENDALE et al., 1995; PARK e KANG, 2001; MEHENDALE et al., 1998; PIEPER, 2008), considerando que a complexidade dos filtros digitais é determinada pelo número de operações aritméticas envolvido. Dentre essas operações, as de multiplicação são as que agregam a maior complexidade. Dessa forma, devido ao fato de os multiplicadores serem circuitos complexos, as suas aplicações em filtros podem acarretar excessiva área, baixo desempenho e elevado consumo de potência desses circuitos, mesmo quando implementados em circuitos *full custom* (CHENDRAKASAN e BRODERSEN, 2000).

O projeto de circuitos multiplicadores visando ao alto desempenho e ao baixo consumo de potência tem sido alvo de intensa pesquisa nos últimos anos (COSTA, 2002; PIEPER, 2008; CALLAWAY e SWARTZLANDER, 1993; COSTA et al., 2001). Em particular, a principal ideia das arquiteturas propostas é manter a mesma regularidade de um multiplicador convencional (NAKAMURA, 1986), reduzindo-se o número de linhas de produtos parciais, que acarreta a redução do

caminho crítico e, consequentemente, o aumento de desempenho e a redução do consumo de potência. Neste trabalho, utiliza-se, na estrutura interna do filtro adaptativo, o circuito multiplicador *array* proposto em Pieper, et al. (2010), que realiza operações de multiplicação na base 4 (multiplicação simultânea de dois bits), além de somadores eficientes do tipo CSA (*Carry Save Adders*) (SOHN, 2004).

Em Costa (2002), foi proposta uma nova arquitetura de circuito multiplicador, que opera em uma codificação diferente da binária (codificação híbrida). Este novo multiplicador foi denominado de multiplicador *array* híbrido na base 2^m (onde *m* representa o número de bits que são multiplicados simultaneamente). A principal vantagem apontada em Costa (2002), para o uso deste novo multiplicador, é a redução da atividade de chaveamento quando há alguma correlação entre os dados nos barramentos, visto que a codificação híbrida agrega o código Gray a cada grupo de *m* bits. A literatura mostra que a codificação Gray consegue reduzir a atividade de transição em torno de 50% em relação à codificação binária em palavras consecutivas (COSTA, 2002).

Neste trabalho de mestrado, propõe-se o uso deste multiplicador híbrido na estrutura interna da arquitetura do filtro adaptativo, de tal forma que o filtro possa operar nessa codificação. Entretanto, além do uso desse multiplicador, propõem-se, no âmbito deste trabalho, novas arquiteturas de circuitos multiplicadores híbridos de tamanhos de palavras diferentes (18, 23 e 36 bits). É importante salientar que o multiplicador proposto em Costa (2002) não era otimizado o suficiente para operar com palavras maiores do que 16 bits. Além disso, o projeto do multiplicador também não previa a operação com palavras ímpares, pois isso agregava um aspecto de irregularidade ao multiplicador. No âmbito deste trabalho de mestrado propõem-se também arquiteturas otimizadas desse multiplicador híbrido com palavras de 23 bits.

A implementação da arquitetura do algoritmo LMS foi realizada a partir da descrição do circuito em linguagem de descrição de *hardware* (VHDL – *Very High Speed Integrated Circuits Hardware Desciption Language*). A arquitetura foi desenvolvida para operações em ponto fixo, pois estas são mais rápidas e oferecem o baixo custo como atrativo, chegando a custar até quatro vezes menos que arquiteturas de ponto flutuante (apesar de menor precisão do que as operações em ponto flutuante) (MANOLAKIS, 2005) .

Visando a validar os resultados obtidos, é também implementado um programa em linguagem C de programação, o qual emula o processamento dos cálculos do sistema adaptativo de filtragem.

A seguir, são apresentados os principais objetivos do trabalho, assim como sua motivação, metodologia e principais contribuições.

1.1 MOTIVAÇÃO

Existem diversas aplicações práticas que não podem ser resolvidas com sucesso utilizando-se filtros digitais fixos, ou porque não possuímos informação suficiente para projetar o filtro com coeficientes fixos ou porque os critérios do projeto mudam durante a operação. Uma grande parte dessas aplicações pode ser resolvida por um tipo especial de filtro chamado de filtro adaptativo. A característica dos filtros adaptativos que os distingue dos demais é que eles podem modificar sua resposta automaticamente para melhorar seu desempenho durante a operação (MANOLAKIS, 2005).

Os sistemas adaptativos são formados por três módulos, representados na figura 1.

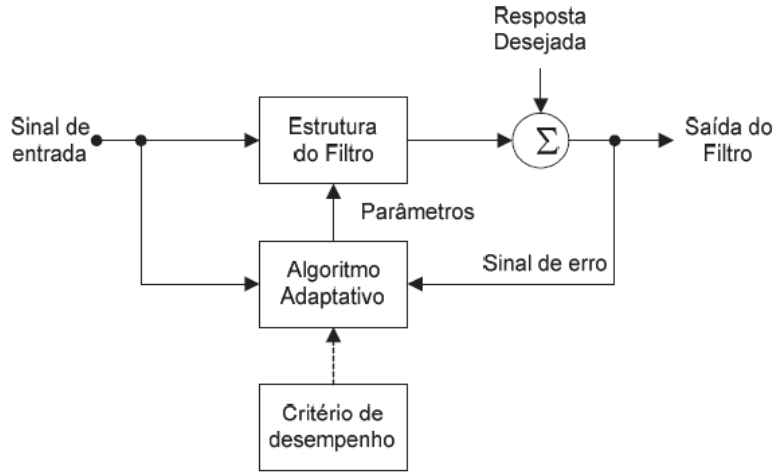

Figura 1: Elementos de um filtro adaptativo

Observa-se, na figura 1, que a estrutura do filtro produz a saída a partir de medições do sinal de entrada. A estrutura é fixa e seus parâmetros são ajustados pelo algoritmo adaptativo. A saída do filtro adaptativo e a resposta desejada são processadas segundo um critério de desempenho, de modo a avaliar a qualidade do

resultado em relação aos requisitos da aplicação. O algoritmo adaptativo utiliza o valor do critério de desempenho, ou alguma função dele, e medições de entrada e da resposta desejada para modificar os parâmetros do filtro, com o objetivo de melhorar seu desempenho.

Diante do exposto, surge a motivação deste trabalho com foco na aplicação de operadores aritméticos eficientes, para a utilização em filtros adaptativos. Essas técnicas devem incluir a redução da atividade de chaveamento nos barramentos de dados a partir da utilização de multiplicadores *array* eficientes em uma codificação diferenciada, a codificação Híbrida (COSTA, 2002).

Neste trabalho utiliza-se o *hardware* de uma arquitetura dedicada de filtragem adaptativa com multiplicadores *array* base-2^m em código híbrido e binário. O algoritmo adaptativo utilizado é o algoritmo LMS.

1.2 OBJETIVOS

O objetivo geral deste trabalho é a implementação de multiplicadores otimizados em codificação Híbrida para aplicação em uma arquitetura dedicada de filtro adaptativo, desenvolvendo, assim, uma arquitetura dedicada de filtro adaptativo híbrida.

O estudo busca explorar a viabilidade de implementação dos operadores propostos em Pieper (2008) em uma diferente codificação, a codificação Híbrida, que foi desenvolvida em Costa (2002), obtendo significantes reduções no consumo de potência, assim como analisar a viabilidade de desenvolver uma arquitetura dedicada de filtro adaptativo híbrida.

Para o projeto do filtro, foi utilizado o algoritmo adaptativo LMS, que apresenta como grande diferencial o fato de não necessitar de operações de média, divisão, raiz quadrada ou diferenciação e ser elegante em sua simplicidade e eficiência (HAYKIN, 1996).

1.3 METODOLOGIA

Para o desenvolvimento e análise dos circuitos implementados, foram utilizadas ferramentas dedicadas. Os circuitos foram descritos em linguagem de

descrição de *hardware* (VHDL) no ambiente Quartus II, da Altera (ALTERA, 2006). Os valores de área, atraso e potência foram obtidos nesta ferramenta.

Foram implementados multiplicadores *array* eficientes, em codificação Híbrida em VHDL, com 18, 23 e 36 bits. Para a comparação de valores de consumo de potência, área e atraso, também foram implementados, em VHDL, multiplicadores *array* eficientes, com 18, 23 e 36 bits, em codificação Binária. Na figura 2, *(a)* e *(b)*, pode ser visto o diagrama de blocos da multiplicação utilizando os multiplicadores *array* eficientes, para codificação Híbrida e Binária, respectivamente.

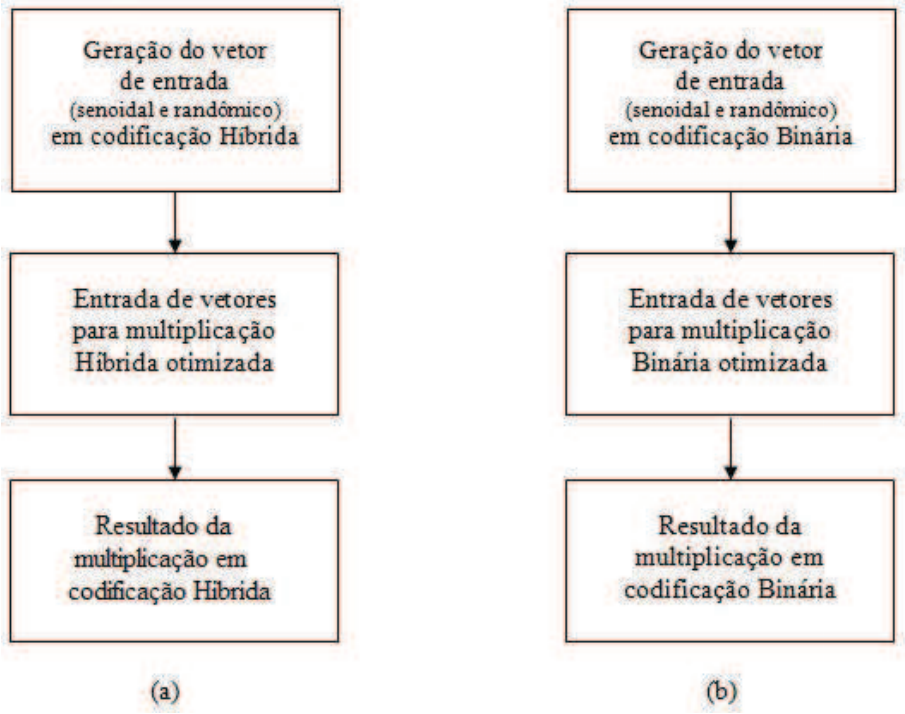

Figura 2: Diagrama de blocos para multiplicação, (a) codificação Híbrida e (b) codificação Binária

O filtro adaptativo utilizando o algoritmo LMS também foi desenvolvido em linguagem de descrição de *hardware*, no ambiente Quartus II, da Altera. Alguns dados necessários para a síntese e a análise do filtro foram desenvolvidos em linguagem *C*, no ambiente MatLab.

A figura 3 mostra o diagrama de blocos dos passos necessários para implementação e verificação deste filtro adaptativo.

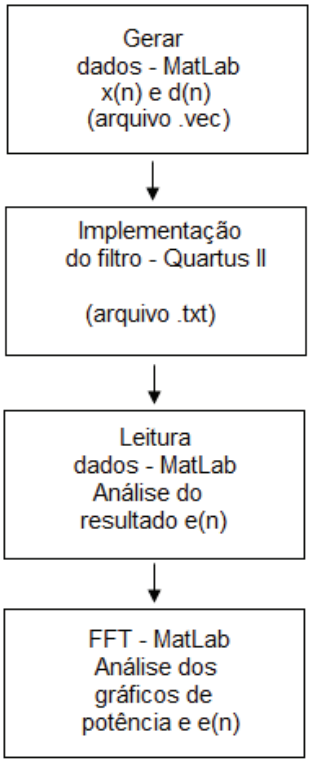

Figura 3: Diagrama de blocos para geração e verificação do filtro adaptativo proposto

Neste trabalho foi desenvolvido o bloco "Estrutura de Filtragem para cancelamento de Interferências" da figura 4, o sinal contaminado e o sinal de interferência em codificação Híbrida, como também um bloco gerador de harmônicas que, juntos, formam a estrutura do filtro adaptativo LMS proposto. Para a comparação de valores de consumo de potência, área e atraso, também foi implementada, em VHDL, esta mesma estrutura em codificação Binária.

O bloco gerador de harmônicas foi obtido através de equações, que são apresentadas mais adiante, neste trabalho.

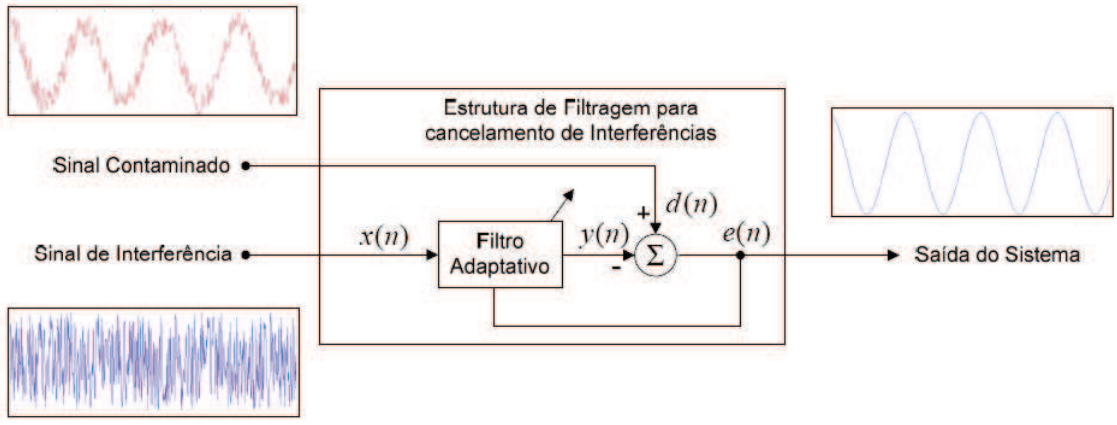

Figura 4: Diagrama de blocos do processo de filtragem

O sistema da figura 4 possui duas entradas, sinal contaminado $d(n)$ e sinal de interferência $x(n)$. As amostras do sinal de interferência são processadas pelo filtro adaptativo, composto por dois coeficientes w_0 e w_1, e, como resposta desse processo de filtragem, uma amostra $y(n)$ é gerada. Este valor de $y(n)$ é então subtraído de uma amostra de $d(n)$; o resultado desta operação é o valor do erro estimado $e(n)$, ou seja, a saída do sistema.

Esse processo se repetirá enquanto houver amostras a serem processadas. O que se espera da estrutura é que, ao final deste processo, a interferência contida no sinal contaminado tenha sido eliminada.

Pode-se dizer que, para que o processo de cancelamento de interferências seja bem sucedido, deve existir alguma relação entre o sinal de interferência $x(n)$ e a interferência contida no sinal sinusoidal que forma o sinal contaminado $d(n)$. Em outras palavras, o trabalho do filtro adaptativo LMS é ajustar os coeficientes w_0 e w_1 de forma a produzir uma saída $y(n)$ mais parecida possível com o ruído contaminador.

O sinal de eletrocardiograma (ECG) utilizado para exemplos de filtragem foi obtido do banco de sinais fisiológicos "*Physiobank Archive Index*"[1].

1.4 TRABALHOS RELACIONADOS

Em arquiteturas de filtro dedicadas, o circuito multiplicador é o operador que agrega a maior complexidade, contribuindo, assim, de forma significativa para o aumento do consumo de potência nessas arquiteturas. Quando a redução de potência em multiplicadores é levada em consideração, o multiplicador *Booth* tem sido a primeira opção (BOOTH, 1951; GALLAGHER, 1994). Entretanto, na pesquisa de Pieper (2008) foi concluído que o multiplicador *array* base-2^m é mais eficiente do que o multiplicador *Booth*. Assim, foi utilizado neste trabalho este multiplicador *array* para a implementação do filtro adaptativo proposto em codificação Híbrida. Também foram desenvolvidos, neste trabalho, multiplicadores *array* eficientes base-2^m que possuem estruturas regulares de multiplicação, m=2, assim como um caso

[1] Banco de sinais fisiológicos "*Physiobank Archive Index*". Disponível em: <http://physionet.incor.usp.br/physiobank/database/> Acesso em: 20 dez. 2010.

especifico para estruturas irregulares, com *m*=3, em que sua multiplicação ocorre com números ímpares. Ambos foram desenvolvidos em codificação Binária e em codificação Híbrida.

Em Costa e Tavares (2004), foi proposto um algoritmo para o cancelamento de interferências baseado no cancelador adaptativo para uma única frequência (*single-frequency adaptive noise canceller*), introduzido por *Widrow* (1975), que possibilita o cancelamento da interferência de 60Hz presente no sinal bioelétrico, através da subtração de uma versão processada do sinal da rede de alimentação, porém com ação inócua a harmônicas superiores. Assim, este algoritmo proposto por Costa e Tavares (2004) tem como principal característica o cancelamento das harmônicas superiores a 60Hz com um mínimo aumento na complexidade computacional do sistema. Resultados da implementação desse algoritmo em um sistema real de aquisição de sinais biológicos demonstraram o excelente desempenho da estrutura proposta em sinais ECG, EEG e MLAEP. Assim, neste trabalho propõe-se a aplicação de uma estrutura de filtragem dedicada Híbrida em que, a partir de um sinal de referência de 60Hz, o algoritmo estima as harmônicas superiores, utilizando esses resultados para o cancelamento da interferência associada ao sinal de interesse.

Em Iturriet (2008), foi proposta uma implementação em *hardware* dedicado ponto fixo para o algoritmo de filtragem adaptativa LMS em arquiteturas reconfiguráveis baseadas em FPGA, visando ao aumento de desempenho e baixo custo computacional. Com base nos resultados apresentados, a estrutura de *hardware* dedicado do filtro adaptativo LMS proposto apresentou resultados muito semelhantes aos resultados obtidos por *software* do filtro adaptativo LMS, validando, assim, a implementação realizada. Esta dissertação estende o trabalho apresentado em Iturriet (2008), apresentando uma estrutura para o cancelamento de interferências, utilizando a codificação Híbrida.

Em Foster et al. (2006), foram apresentadas duas diferentes arquiteturas dedicadas para o filtro adaptativo LMS (totalmente sequencial e semi-paralela), utilizando operadores aritméticos de baixa potência nas arquiteturas do filtro. Os resultados mostraram que, apesar de os filtros apresentarem maior área utilizando multiplicador *array* binário (*m*=2), estas arquiteturas de filtro podem apresentar uma frequência mais elevada e um menor consumo de potência utilizando esse tipo de multiplicador.

Apesar de alguns trabalhos da literatura abordarem implementações de arquiteturas dedicadas de filtro adaptativo LMS, nenhum desses trabalhos apresentados propõe implementações de arquiteturas dedicadas de filtros adaptativos em codificação Híbrida. Este trabalho visa ao desenvolvimento de um filtro adaptativo utilizando o algoritmo LMS para o cancelamento de interferências, tanto em codificação Binária como em codificação Híbrida.

1.5 CONTRIBUIÇÕES DO TRABALHO

O trabalho desenvolvido propõe a elaboração e a aplicação de multiplicadores *array* otimizados de Pieper (2008), em codificação Híbrida, inseridos em estrutura dedicada de filtro adaptativo LMS. Dessa forma, as contribuições parciais deste trabalho estão focadas nas seguintes etapas:

- Exploração arquitetural de multiplicadores *array* otimizados (arquiteturas com 18, 23 e 36 bits) em código Binário e Híbrido;
- Implementação desses multiplicadores *array* otimizados, com estruturas regulares (18 e 36 bits) em código Binário e Híbrido, em linguagem de descrição de *hardware*;
- Implementação de um multiplicador otimizado com uma estrutura irregular com 23 bits;
- Aplicação dos multiplicadores *array* otimizados, em código Binário e Híbrido, em estrutura dedicada de filtro adaptativo LMS, para o cancelamento de interferências, em linguagem de descrição de *hardware*.

1.5 ESTRUTURA DA DISSERTAÇÃO

No Capítulo 1 é apresentada uma introdução deste trabalho, mostrando as motivações, objetivos, metodologia e trabalhos já apresentados na literatura, assim como as contribuições deste estudo para a comunidade científica. No capítulo 2 são expostos alguns conceitos que envolvem as principais fontes de consumo de potência em circuitos CMOS. No Capítulo 3 é apresentada a teoria de filtragem adaptativa, descrevendo a estrutura típica de filtros adaptativos, bem como o emprego de filtros adaptativos no cancelamento de interferências. Neste capítulo

também é explicada a teoria matemática envolvida na dedução dos algoritmos adaptativos baseados no filtro de Wiener. No Capítulo 4 são apresentadas as principais características dos circuitos multiplicadores *array* base-2^m, e a possível operação em uma diferente codificação, a codificação Híbrida, assim como uma nova estrutura irregular para multiplicadores com largura de bits ímpares, tanto em código Binário, como em código Híbrido. No Capítulo 5 é apresentada a estrutura dedicada do filtro adaptativo LMS implementado como proposta deste trabalho, assim como os blocos que o formam e suas funcionalidades. No Capítulo 6 são apresentados os resultados obtidos com as implementações realizadas nos filtros adaptativos LMS desenvolvidos, mostrando suas curvas de resposta, para a comprovação da eficácia das estruturas atuando no cancelamento de harmônicas em codificação Híbrida e Binária. No Capítulo 7 são apresentadas as conclusões e a perspectiva de trabalho futuro. E, por fim, são listadas as referências utilizadas no auxilio deste trabalho.

2 FONTES DE CONSUMO DE POTÊNCIA EM CIRCUITOS CMOS

Neste capítulo são apresentados alguns conceitos que envolvem as principais fontes de consumo de potência em circuitos CMOS *(Complementary Metal Oxide Semiconductor*).

A estimativa do consumo de potência de um circuito determina quanto desta potência é consumida por operação e/ou quanto calor é dissipado por este circuito. Esses fatores têm uma grande influência em decisões críticas de projeto, tais como capacidade da fonte de alimentação, tempo de vida da bateria e tamanho das linhas de alimentação (WESTE e ESHRAGHIAN, 1994).

A dissipação de potência em um circuito pode ser dividida em três parcelas:

- dissipação estática de potência;
- consumo de potência de curto-circuito;
- dissipação dinâmica de potência.

A parcela estática é determinada em função da corrente de fuga dos transistores e junções *pn* e correntes de *subthreshold*. Idealmente, o consumo de potência estática de um circuito CMOS é nulo. Entretanto, há sempre uma corrente de fuga presente nos circuitos. Essa parcela passa a ser mais significativa nas tecnologias CMOS atuais (valores abaixo de 0,1 μm) (KIM et al., 2003).

Por outro lado, a parcela dinâmica é a parte dominante do consumo de potência, embora a diferença venha se tornando menor em novas tecnologias que envolvem maiores escalas de integração (WILLIAMS et al., 1996). Esta parcela ocorre pelo processo de carga e descarga da capacitância de saída.

Já o consumo de potência de curto-circuito ocorre quando flui uma corrente diretamente da fonte de alimentação para o terra. Isso ocorre quando um circuito CMOS estático é chaveado por um sinal de entrada com tempos de subida e descida não-zero, com os transistores PMOS (*Positive Channel Metal Oxide Semiconductor*) e NMOS (*Negative Channel Metal Oxide Semiconductor*) conduzindo simultaneamente por um curto intervalo de tempo.

Assim, o consumo de potência total em circuitos CMOS pode ser descrito como a equação (1)

$$P_{TOTAL} = P_{SC} + P_{STATIC} + P_{DIN} \quad (1)$$

onde P_{sc} representa a potência de curto-circuito, P_{static} representa a potência estática e P_{din} representa a potência dinâmica.

Neste trabalho a parcela do consumo de potência dinâmica será explorada a partir da atividade de chaveamento nos barramentos de dados de arquiteturas de filtros adaptativos.

A seguir são apresentadas as principais características dessa fonte de consumo de potência em circuitos digitais CMOS.

2.1 POTÊNCIA DINÂMICA

A potência dinâmica ainda é a fonte dominante de consumo de potência, embora a diferença venha se tornando menor em novas tecnologias, como dito anteriormente.

A potência dinâmica consumida ocorre durante a carga e descarga da capacitância de saída. A componente de comutação dinâmica de dissipação de potência (*Pdin*) em uma transição na saída de uma porta carregada por um capacitor C_L é expressa de acordo com a equação (2) (CHANDRAKAN e BRODERSEN, 1995), onde *P* é a potência dinâmica, C_L a capacitância de carga, V_{dd} a tensão de alimentação, *A* a atividade do nó de saída, medida em eventos/segundo para uma carga/descarga completa.

$$P = \frac{1}{2} C_L V^2{}_{dd} A = \frac{1}{2} C_L V^2{}_{dd} af \quad (2)$$

No caso de projetos síncronos, a atividade *A* não é simplesmente *f* (frequência), mas em geral uma probabilidade de atividade normalizada *a* (menor do que 1 para modelo de atraso zero) é computada como função da estatística de entrada e modelos lógicos, pois nem todos os nós mudam em um determinado ciclo de relógio. Se, em um circuito, uma simples transição é realizada a cada ciclo de relógio na taxa f_{clk}, então a potência é dada pela equação (3).

$$P = \frac{1}{2} C_L V^2_{dd} f_{CLK} \qquad (3)$$

Entretanto, existem casos em que a transição do sinal ocorre em diferentes taxas de frequência, tendo-se que considerar o fator de atividade de transição dos nós *a.*

Os sinais espúrios (*glitching*) também influenciam na dissipação de potência dinâmica, o que é característico de sistemas em lógicas complementares em cascata (GOWAN et al., 1998). O atraso de propagação entre portas lógicas pode acarretar sinais espúrios, ou seja, um nó pode ter transições indesejáveis antes de obter o nível lógico correto. Essa atividade de chaveamento extra contribui para a dissipação total de energia com cerca de 20% do valor global em circuitos CMOS (COSTA, 2002).

A figura 5 mostra uma geração e propagação de sinais espúrios em portas lógicas. Nota-se a geração de *glitches* na saída de uma porta, havendo propagação por outra porta.

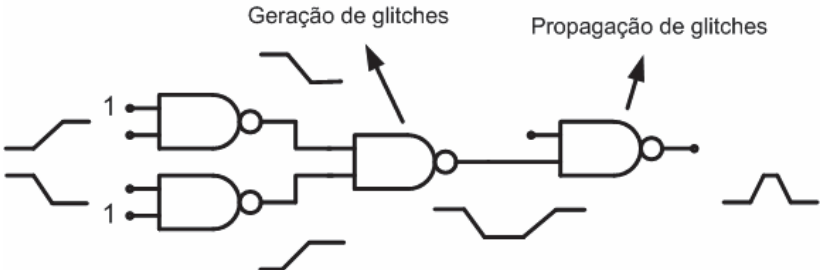

Figura 5: Esquema de contribuição de *glitches* para dissipação de potência

Observa-se que o instante de chaveamento do sinal de entrada de uma porta pode causar níveis lógicos falsos em sua saída, contribuindo para a geração de *glitches.* A propagação desses sinais é uma função da profundidade lógica do circuito, contribuindo fortemente para o aumento do consumo de potência.

2.2 RESUMO

Neste capítulo foram apresentados alguns conceitos que envolvem as principais fontes de consumo de potência em circuitos CMOS, sendo que, neste trabalho, a parcela do consumo de potência dinâmica é explorada a partir da

atividade de chaveamento nos barramentos de dados de arquiteturas de filtros adaptativos.

3 FILTRAGEM ADAPTATIVA

Filtragem se refere ao processo linear de alterar o conteúdo espectral de um sinal de entrada de uma maneira especificada.

Filtros FIR e IIR convencionais são invariantes no tempo, ou seja, efetuam operações lineares no sinal de entrada para gerar um sinal de saída baseado em coeficientes fixos (KUO e LEE, 2001).

Já os filtros adaptativos são variantes no tempo, isto é, características do filtro como largura de banda e resposta espectral variam em função do tempo. A razão para esse comportamento é que os coeficientes do filtro adaptativo são ajustados automaticamente por um algoritmo adaptativo com base nos sinais de entrada.

Esse comportamento é desejado em situações em que as exatas especificações requeridas para o filtro são desconhecidas ou não estacionárias. Neste capítulo são abordados alguns conceitos sobre estruturas de filtragem e o algoritmo adaptativo LMS (*Least Mean Square*). Ao longo deste capítulo, para a representação vetorial e matricial de sinais, são adotadas algumas convenções: letras minúsculas em negrito representam vetores, letras maiúsculas em negrito representam matrizes e letras minúsculas sem negrito representam escalares.

3.1 ESTRUTURAS DE FILTROS ADAPTATIVOS

Um filtro adaptativo consiste de duas partes distintas – um filtro digital para efetuar o processamento desejado do sinal, e um algoritmo adaptativo para ajustar os coeficientes (ou pesos) deste filtro (KUO e LEE, 2001). Uma forma geral de filtro adaptativo está ilustrada na figura 6, onde $d(n)$ é um sinal desejado, $y(n)$ é a saída do filtro digital cuja entrada é $x(n)$, e $e(n)$ corresponde a um sinal de erro, que é a diferença entre a resposta desejada e a saída do filtro. A função do algoritmo adaptativo é ajustar os coeficientes do filtro no sentido de minimizar o valor médio quadrático do erro.

Em geral, existem dois tipos de filtros digitais que podem ser usados na estrutura de um filtro adaptativo: filtros FIR e IIR. Filtros FIR apresentam a vantagem de possuírem uma resposta de fase linear e de serem sempre estáveis, pois,

diferentes dos filtros IIR, não apresentam realimentação da saída no cálculo da mesma. Já os filtros IIR, em comparação com o filtro FIR, apresentam a vantagem de necessitarem menos coeficientes em sua estrutura para executar a mesma tarefa que o filtro FIR.

Pelo motivo de o filtro ser adaptativo, problemas de estabilidade são muito difíceis de gerenciar, o que faz dos filtros adaptativos FIR a escolha mais usada em aplicações em tempo real (KUO e LEE, 2001; HAYKIN, 1996).

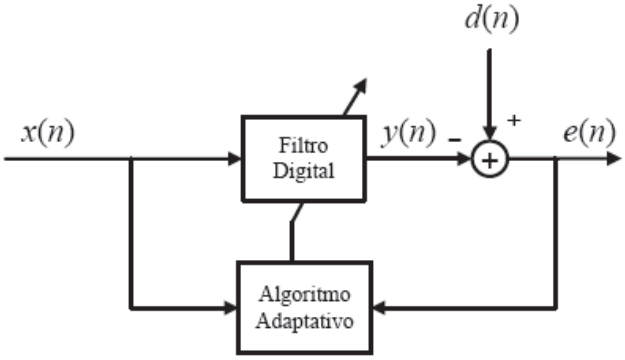

Figura 6: Diagrama de blocos de um filtro adaptativo

Uma estrutura muito utilizada de filtro adaptativo FIR está ilustrada na figura 7.

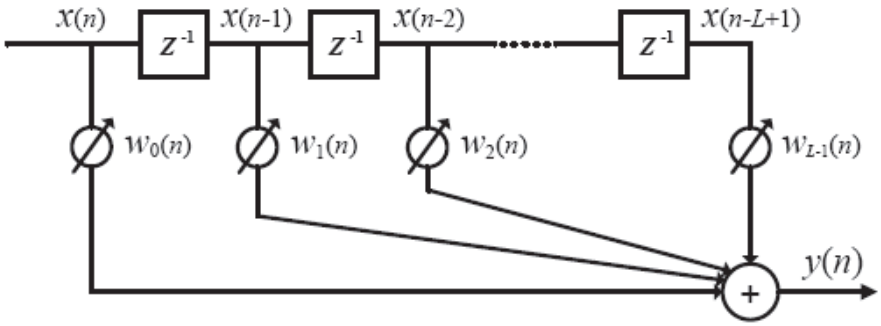

Figura 7: Filtro FIR para filtragem adaptativa

Dado um conjunto de L coeficientes, $w_k(n)$, k = 0,1, ..., L-1, e uma sequência de amostras de entrada, {$x(n)$ $x(n–1)$ ... $x(n–L+1)$}, a saída do filtro é calculada através do processo de convolução

$$y(n) = \sum_{k=0}^{L-1} w_k(k)x(n-k) \qquad (4)$$

onde os coeficientes do filtro $w_k(n)$ são variantes no tempo e ajustados pelo algoritmo adaptativo. O vetor de entrada no instante n é definido como

$$x(n) = [x(n)x(n-1)...x(n-L+1)]^T \quad (5)$$

E o vetor dos coeficientes são definidos como

$$\boldsymbol{w}(n) = [w_0(n)w_1(n)...w_{L-1}(n)]^T \quad (6)$$

Utilizando-se das equações (5) e (6), o sinal de saída *y*(*n*) da equação (4) pode ser expresso usando representação vetorial pela expressão

$$y(n) = \boldsymbol{w}^T(n)\boldsymbol{x}(n) = \boldsymbol{x}^T(n)\boldsymbol{w}(n) \quad (7)$$

A cada nova amostra que chega, as amostras do vetor ***x(n)*** são deslocadas, fazendo com que a amostra mais antiga seja perdida, e a nova amostra tome a posição vazia. Então, a operação de filtragem é realizada novamente e a saída do filtro *y*(*n*) é comparada com a resposta desejada *d*(*n*), resultando no sinal de erro

$$e(n) = d(n) - y(n) = d(n) - \boldsymbol{w}^T(n)\boldsymbol{x}(n) \quad (8)$$

A magnitude do sinal de erro, por sua vez, é usada pelo algoritmo adaptativo para ajustar os coeficientes do filtro.

3.2 APLICAÇÕES

As características desejáveis para um filtro adaptativo são a habilidade para operar em um ambiente desconhecido e seguir as variações no tempo dos sinais de entrada. A diferença essencial entre as diversas aplicações de filtros adaptativos é como os sinais *x*(*n*), *d*(*n*), *y*(*n*) e *e*(*n*) são conectados, existindo, assim, quatro classes básicas de filtros adaptativos: identificação de sistemas, modelagem inversa, predição linear e cancelamento de interferências (KUO e LEE, 2001; GAN e KUO, 2007; MANOLAKIS, 2005). Neste trabalho é explorada a aplicação do filtro adaptativo no cancelamento de interferências.

3.2.1 Cancelamento de interferências

A aplicação de cancelamento de interferências é ilustrada pelo esquema mostrado na figura 8. Nesta estrutura, um sinal corrompido por um ruído aditivo (sinal primário) e uma versão correlacionada do ruído estão disponíveis. O objetivo do filtro adaptativo é produzir uma saída *y* igual ou o mais parecida possível com o

ruído aditivo. Dessa maneira, a saída de erro *e* será igual ou muito próxima ao sinal primário sem interferência (MANOLAKIS, 2005).

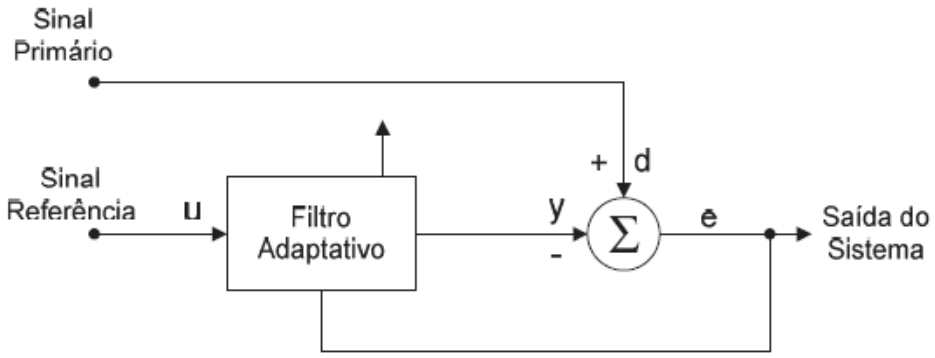

Figura 8: Cancelador de interferências adaptativo

3.3 ALGORITMOS ADAPTATIVOS

O papel do algoritmo adaptativo é ajustar os coeficientes do filtro digital, na tentativa de satisfazer algum critério de desempenho pré-estabelecido. O critério de desempenho mais utilizado na maioria dos algoritmos é baseado na minimização do erro médio quadrático (EMQ) definido como

$$\xi(n) = E[e^2(n)] \tag{9}$$

onde o operador *E*[.] representa o valor esperado para a variável $e^2(n)$, ou seja, o valor médio da variável após diversas realizações do processo (Processo Monte Carlo).

O EMQ definido na equação (9) é uma função dos coeficientes do vetor $\boldsymbol{w}(n)$. A cada iteração do algoritmo, todos os coeficientes do filtro são ajustados, e para cada novo vetor de coeficientes existe um valor correspondente para o EMQ. Elevando *e*(*n*) da equação (8) ao quadrado, obtém-se

$$e^2(n) = d^2(n) + \boldsymbol{w}^T(n)\boldsymbol{x}(n)\boldsymbol{x}^T(n)\boldsymbol{w}(n) - 2d(n)\boldsymbol{x}^T(n)\boldsymbol{w}(n) \tag{10}$$

Substituindo a equação (10) na equação (9), resulta em

$$E[e^2(n)] = E[d^2(n)] + \boldsymbol{w}^T(n)E[\boldsymbol{x}(n)\boldsymbol{x}^T(n)]\boldsymbol{w}(n) - 2E[d(n)\boldsymbol{x}^T(n)]\boldsymbol{w}(n) \tag{11}$$

Pode-se representar a equação (11), definindo uma matriz $\boldsymbol{R}$ de autocorrelação do vetor de amostras do sinal de entrada como

$$\boldsymbol{R} = E[\boldsymbol{x}(n)\boldsymbol{x}^T(n)] = E\begin{bmatrix} x_0^2(n) & x_0(n)x_1(n) & \cdots & x_0(n)x_{L-1}(n) \\ x_1(n)x_0(n) & x_1^2(n) & \cdots & x_1(n)x_{L-1}(n) \\ \vdots & \vdots & \ddots & \vdots \\ x_{L-1}(n)x_0 & x_{L-1}(n)x_1 & \cdots & x_{L-1}^2(n) \end{bmatrix} \quad (12)$$

Os termos da diagonal principal de ***R*** correspondem ao valor esperado para o quadrado dos componentes da entrada *x*(*n*) e podem também ser interpretados como uma estimação da potência do sinal de entrada. Os termos fora da diagonal principal representam a correlação entre as diferentes amostras do vetor de entrada. Para um ruído branco, por exemplo, estes termos seriam nulos, indicando nenhuma correlação entre uma amostra e outra. Também é possível definir um vetor ***p*** de correlação cruzada entre ***x****(n)* e *d*(*n*) como

$$\boldsymbol{p} = E[d(n)\boldsymbol{x}(n)] = E[d(n)x_0(n)\ d(n)x_1(n)\ \ldots\ d(n)x_{L-1}(n)]^T \quad (13)$$

Assim, pode-se expressar o EMQ $\xi(n)$ substituindo (13) e (12) em (11)

$$\xi(n) = E[d^2(n)] + \boldsymbol{w}^T(n)\boldsymbol{R}\boldsymbol{w}(n) - 2\boldsymbol{p}^T\boldsymbol{w}(n) \quad (14)$$

Assumindo que o valor esperado do sinal $E[d^2(n)]$, que a matriz de autocorrelação ***R*** do sinal de referência ***x****(n)* e que o vetor de correlação cruzada ***p*** sejam constantes, pode-se concluir que o EMQ é uma função de segundo grau do vetor de coeficientes ***w****(n)*, com concavidade voltada para cima. E que, portanto, possui um único valor mínimo (MANOLAKIS, 2005).

Uma maneira de determinar o ponto de mínimo da superfície do erro quadrático é tomar a derivada parcial da equação (14) em relação a cada coeficiente do vetor ***w****(n)* e igualar a zero

$$\nabla\xi(n) = \frac{\partial\xi}{\partial w} = \left[\frac{\partial\xi}{\partial w_0}\frac{\partial\xi}{\partial w_0}\cdots\frac{\partial\xi}{\partial w_0}\right]^T = 2\boldsymbol{R}\boldsymbol{w} - 2\boldsymbol{p} \quad (15)$$

onde o símbolo ∇ representa o gradiente.

Para que a equação acima forneça o valor mínimo do erro médio quadrático, o vetor de coeficientes ***w*** deverá ter seu valor ótimo, $\boldsymbol{w}^o$, o que acontece quando o gradiente da equação (15) é igual a zero

$$\nabla\xi(n) = 0 = 2\boldsymbol{R}\boldsymbol{w}^o - 2\boldsymbol{p} \quad (16)$$

Assumindo que a matriz de autocorrelação admite inversa, obtém-se

$$\boldsymbol{w}^o = \boldsymbol{R}^{-1}\boldsymbol{p} \quad (17)$$

A equação (17) é conhecida como solução de Wiener, e fornece o vetor de coeficientes ótimo. O grande problema desta solução é que exige a inversão da matriz de autocorrelação, o que envolve grande esforço computacional. Dessa forma, utilizam-se na prática métodos iterativos baseados no gradiente, que consistem em dar pequenos passos em direção ao ponto mínimo da superfície do EMQ, sem que seja necessária a inversão da matriz $\boldsymbol{R}$. Um desses métodos é conhecido como *steepest descent*, ou gradiente descendente, numa tradução livre, e consiste em fazer com que os coeficientes do filtro se aproximem da solução de Wiener, seguindo a direção oposta do gradiente expresso na equação (16), a partir de pequenos passos de adaptação. Dessa forma, pode-se expressar o algoritmo como

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \frac{\mu}{2}\left(-\nabla \xi(n)\right) \tag{18}$$

e, por fim, substituindo (16) em (18), obtém-se

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu[\boldsymbol{p} - \boldsymbol{R}\boldsymbol{w}(n)] \tag{19}$$

onde μ representa o passo de adaptação, ou fator de convergência, que determina a estabilidade e a taxa de convergência do algoritmo. Quanto maior for o valor de μ, maior será a velocidade de convergência. Entretanto, como será visto a seguir, este valor é limitado em virtude de poder não permitir que os coeficientes do filtro adaptativo não convirjam adequadamente, ocasionando um erro elevado na fase de regime permanente.

3.3.1 Algoritmo LMS

O método do gradiente descendente, descrito anteriormente, assume o conhecimento exato do vetor gradiente a cada iteração, a partir do conhecimento da matriz de autocorrelação $\boldsymbol{R}$ e do vetor de correlação cruzada $\mathbf{p}$, o que se torna inviável em muitas aplicações práticas. Para contornar esse problema, Widrow e Hoff, no ano de 1960, desenvolveram um algoritmo chamado *Least Mean Square* (LMS), que usa o valor instantâneo do erro quadrático, $e^2(n)$, para estimar o EMQ (KUO e LEE, 2001). Assim sendo, define-se

$$\hat{\xi}(n) = e^2(n) \tag{20}$$

Com isso, o gradiente usado pelo LMS fica

$$\nabla \hat{\xi}(n) = \nabla e^2(n) = 2[\nabla e(n)]e(n) \quad (21)$$

Uma vez que $e(n) = d(n) - \mathbf{w}^T(n)\mathbf{x}(n)$, então $\nabla e(n) = -\mathbf{x}(n)$ e, portanto, a estimação do gradiente torna-se

$$\nabla \hat{\xi}(n) = -2\mathbf{x}(n)e(n) \quad (22)$$

Substituindo esta estimação do gradiente em (18), obtém-se

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\mathbf{x}(n)e(n) \quad (23)$$

Esta é a equação de atualização dos coeficientes do conhecido algoritmo LMS. É uma equação simples, que não requer radiciações, médias, ou derivadas, apenas multiplicações e somas, operações que podem ser facilmente implementadas em DSPs e ASICs ou FPGAs.

Assim como no método do gradiente descendente, a velocidade de convergência do LMS é proporcional ao valor de μ. Contudo, é necessário limitar o valor de μ para garantir a estabilidade, de modo que satisfaça à seguinte equação

$$0 < \mu < \frac{2}{LP_x} \quad (24)$$

onde L é o número de coeficientes do filtro e Px é a potência do vetor de referência $\mathbf{x}(n)$, ou seja, o maior autovalor da matriz de autocorrelação $\mathbf{R}$. O algoritmo LMS converge em média para a solução de Wiener e fica oscilando em torno dela ao atingir o regime permanente. Essa oscilação depende do passo de adaptação μ; quanto maior seu valor, maior é a oscilação.

A equação (24) fornece importantes informações a respeito de como escolher um valor adequado para μ, tais como:

1 – Uma vez que o máximo valor de μ é inversamente proporcional a L, um passo de adaptação menor é necessário para filtros de ordem muito grande;

2 – Para um passo de adaptação fixo, o algoritmo LMS é dependente da potência do sinal de referência, isto é, sinais de pequena magnitude necessitam de um μ maior e sinais de magnitude elevada requerem um μ menor para alcançarem a máxima velocidade de convergência.

Para finalizar, pode-se resumir e enumerar os passos para a realização do algoritmo LMS da seguinte forma:

1. Cálculo da saída do filtro

$$y(n) = \mathbf{w}^T(n)\mathbf{x}(n)$$

2. Cálculo do erro

$$e(n) = d(n) - y(n)$$

3. Atualização dos coeficientes

$$\boldsymbol{w}(n + 1) = \boldsymbol{w}(n) + \mu \boldsymbol{x}(n)e(n)$$

3.4 RESUMO

Este capítulo apresentou uma introdução à filtragem adaptativa, a classes de operação onde os filtros podem ser utilizados e aos elementos que compõem um filtro adaptativo. Para o entendimento do trabalho, foi demonstrado todo desenvolvimento matemático dos filtros e do algoritmo adaptativo LMS.

4 MULTIPLICADORES OTIMIZADOS

Neste capítulo são descritos os principais aspectos dos multiplicadores utilizados nas arquiteturas dedicadas do algoritmo LMS de filtragem adaptativa. Foram desenvolvidas duas arquiteturas para este algoritmo, sendo uma na codificação binária e a outra na codificação híbrida. Para a arquitetura que opera na codificação binária, foi utilizado, como base, o multiplicador *array* otimizado proposto em Pieper (2008). Para a arquitetura que opera na codificação híbrida, foram propostas novas arquiteturas de circuitos multiplicadores, que têm como base o multiplicador híbrido proposto em Costa (2002). A seguir são apresentadas as principais características destes circuitos multiplicadores.

4.1 MULTIPLICADOR ARRAY BASE-2^M

O multiplicador *array* na base 2^m proposto em Pieper (2008) utiliza como base o multiplicador *array* proposto em Costa (2002), que tem como principal característica a redução do número de linhas de produtos parciais.

Um circuito multiplicador convencional (HWANG, 1979), que realiza a operação de multiplicação bit a bit (m=1), apresenta como principal característica a regularidade e a fácil implementação, visto que a sua estrutura é composta apenas por portas *AND* (que realizam a operação de multiplicação propriamente dita) e circuitos somadores (que agrupam os resultados parciais das multiplicações e somas anteriores). Esta estrutura pode ser vista na Figura 9 para um exemplo de multiplicador de 4 bits.

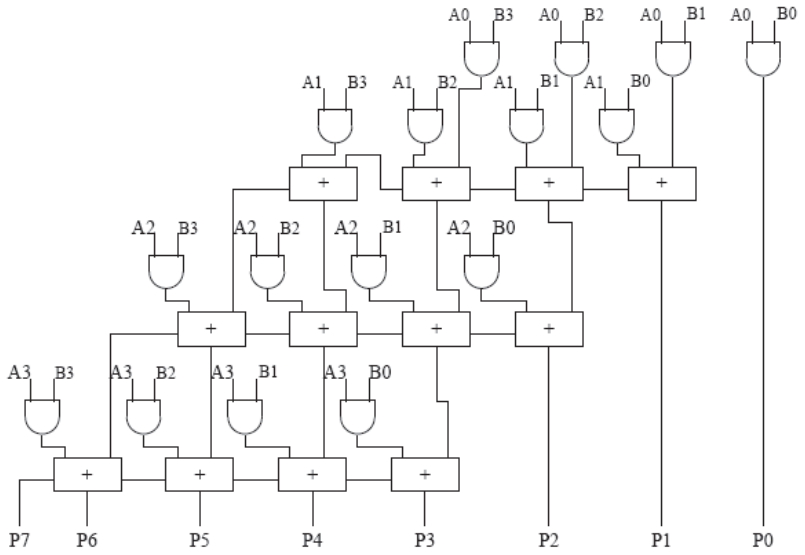

Figura 9: Estrutura de um multiplicador *array* convencional (*m*=1) de 4 bits
Fonte: COSTA, 2002.

Na estrutura mostrada na figura 9, pode-se observar o longo caminho crítico da entrada até a saída do circuito, o que compromete o seu desempenho (visto que o atraso característico será dado pelo somatório das inúmeras portas lógicas presentes no caminho crítico) e o consumo de potência (visto que há uma intensa atividade de *glitching* ao longo da estrutura).

Em Costa (2002), foi proposta uma nova estrutura de circuito multiplicador *array* na base 2^m em complemento de 2, que mantém a regularidade do multiplicador convencional, e que reduz o número de linhas de produtos parciais (visando à redução da profundidade lógica e, consequentemente, à redução do caminho crítico). Esta estrutura pode ser vista na Figura 10, para operandos com largura de bits *W*=8 usando base-16 (*m*=4).

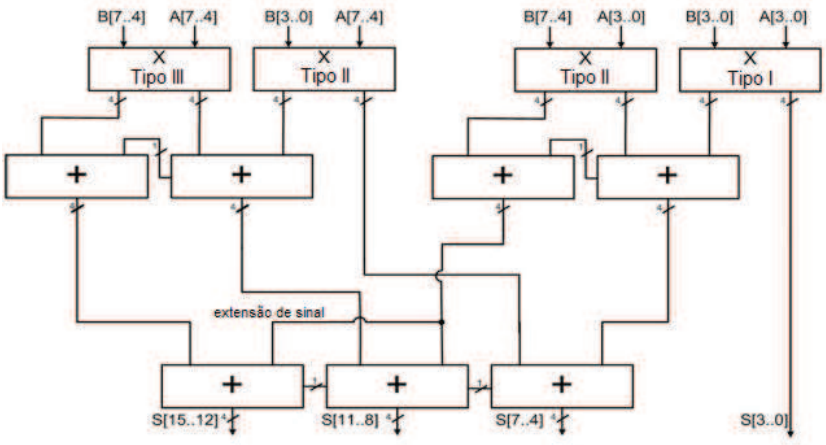

Figura 10: Arquitetura do multiplicador *array* (*m*=4) binário de 8 bits.
Fonte: proposto em Costa (2002)

Para a arquitetura mostrada na figura 10, três tipos de módulos são necessários. Os módulos do tipo I são módulos que efetuam operações de multiplicação sem sinal. Os módulos tipo II realizam a multiplicação de um valor sem sinal com um valor em complemento de 2. Finalmente, o módulo tipo III opera com dois valores com sinais. Somente o módulo tipo III é necessário para qualquer tipo de multiplicador, pois este módulo sempre realiza as operações de multiplicação entre os *m* bits mais significativos. Entretanto, para as outras operações de multiplicação, que envolvem operações dos bits mais significativos com os bits menos significativos, são necessários $(2\frac{W}{m}-2)$ módulos do tipo II e $(\frac{W}{m}-1)^2$ módulos do tipo I.

O multiplicador apresentado na figura 10 exibe um bom compromisso entre regularidade e redução do número de linhas de somas de produtos parciais. Para uma operação na base 4, por exemplo, os circuitos dedicados de multiplicação Tipo I, Tipo II e Tipo III são facilmente obtidos pela minimização lógica (usando mapa de *Karnaugh*), visto que, na base 4, os módulos de multiplicação apresentam 4 entradas (visto que na base 4 => *m*=2, ou seja, dois bits são multiplicados simultaneamente). Entretanto, para operações na base 16 (como o exemplo mostrado na Figura 10), torna-se difícil a minimização dos módulos de multiplicação (visto que na base 16 => *m*=4, quatro bits são multiplicados simultaneamente e os módulos de multiplicação apresentam 8 entradas).

Para esses módulos de multiplicação dedicados *m*=4, a estratégia utilizada em Costa (2002) foi a geração de um arquivo PLA (*Programmable Logic Array*) com todas as combinações binárias possíveis para o multiplicador (combinações das entradas binárias e resultados binários) e síntese automática gerada pela ferramenta SIS (*Synthesis Interchange Logic*) (SENTOVICH, 1992). A ferramenta gera automaticamente os módulos dedicados de multiplicação mapeados para um arquivo de portas lógicas (*mcnc.genlib*). Foi observado em Costa (2002) que a base 16 (*m*=4) é o limite de circuito que a ferramenta SIS pode sintetizar e otimizar os circuitos a partir do arquivo PLA (devido ao complexo algoritmo de otimização utilizado pela ferramenta). Entretanto, visando a superar tal limite, em Pieper (2008) são propostos novos blocos dedicados de módulos de multiplicação tipos I, II e III para uma multiplicação na base 2^m.

4.1.1 Multiplicador array BASE-2^m proposto em Pieper (2008)

O multiplicador *array* proposto em Pieper (2008) tem como base dois aspectos: i) o uso de multiplicadores dedicados na base 4, independente da base de operação do multiplicador (devido ao fato de os multiplicadores na base 4 serem fáceis de ser implementados, pois utilizam poucas portas lógicas), e ii) o uso de somadores CSA - *Carry Save Adder* (KIM, 1998) (pelo fato de estes somadores serem mais eficientes na propagação do *carry*, o que torna mais rápida a operação de soma, além de fazerem a soma simultânea de três operandos).

A Figura 11 mostra um exemplo de multiplicação de dois números sem sinal (Tipo I) na base 16 (*m*=4), com a respectiva indicação das somas dos produtos parciais gerados por cada bloco de multiplicação na base 4, onde se pode observar que, usando o somador CSA, três números podem ser somados simultaneamente. Como pode ser observado pelas linhas pontilhadas da Figura 11(b), o caminho crítico do circuito é composto por um bloco de multiplicação *m*=2, um bloco de soma CSA, um meio somador e um somador completo. O fato de usar somador CSA faz com que haja um bloco a menos de soma no caminho crítico em relação à técnica proposta anteriormente na literatura (COSTA, 2002).

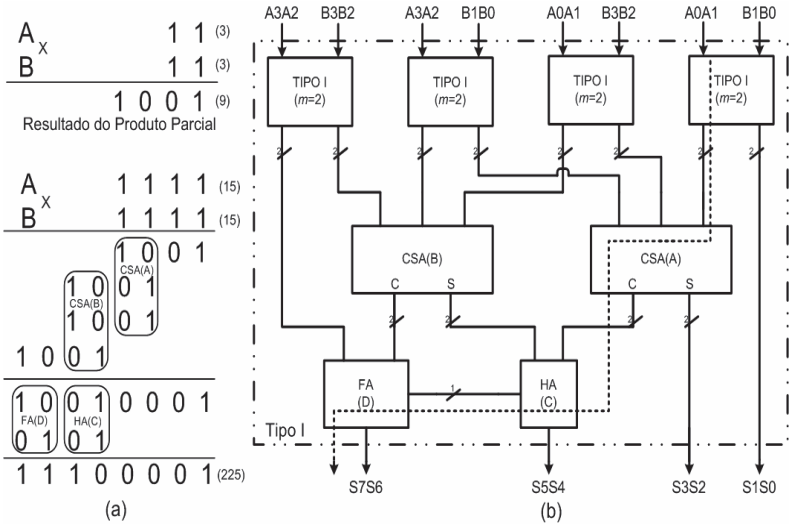

Figura 11: Bloco de multiplicação *m*=4 Tipo I composto por blocos *m*=2 e somador CSA
Fonte: Pieper, 2008.

De acordo com Pieper (2008), a estrutura regular apresentada pelos novos blocos de multiplicação dedicados na base 16 possibilita uma fácil extensão para multiplicações na base 256. A figura 12 mostra exemplos de multiplicação nas bases 16 (*m*=4) e 256 (*m*=8) para blocos de multiplicação do tipo III (operações em

complemento de 2). Como pode ser visto nos exemplos, a partir do bloco *m*=4, pode-se facilmente estender para a estrutura *m*=8, visto que esta é composta pelo mesmo número de somadores CSA (*Carry Save Adder*) do bloco *m*=4.

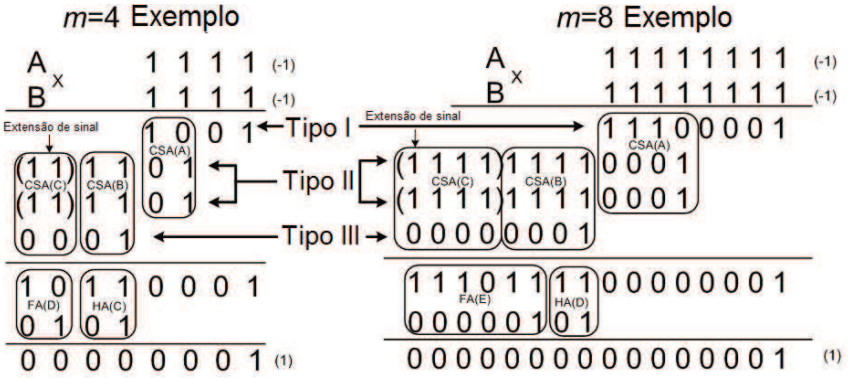

Figura 12: Exemplos de multiplicação tipo III nas bases 16 e 256
Fonte: Pieper, 2008.

A arquitetura para o exemplo de multiplicação do bloco *m*=8 tipo III é ilustrada na figura 13. Observa-se, pelas linhas pontilhadas, o caminho crítico do circuito que é composto apenas por um bloco de multiplicação do tipo I (com poucas portas lógicas), um somador CSA, um meio somador (HA – *Half Adder*) e um somador completo (FA – *Full Adder*). Esse aspecto torna esses multiplicadores atrativos para aplicações que envolvam aspectos de aumento de desempenho e redução do consumo de potência, como pode ser visto na tabela 1, onde são apresentados os resultados de comparação entre os multiplicadores *array* original (COSTA, 2002) e otimizado (PIEPER, 2008).

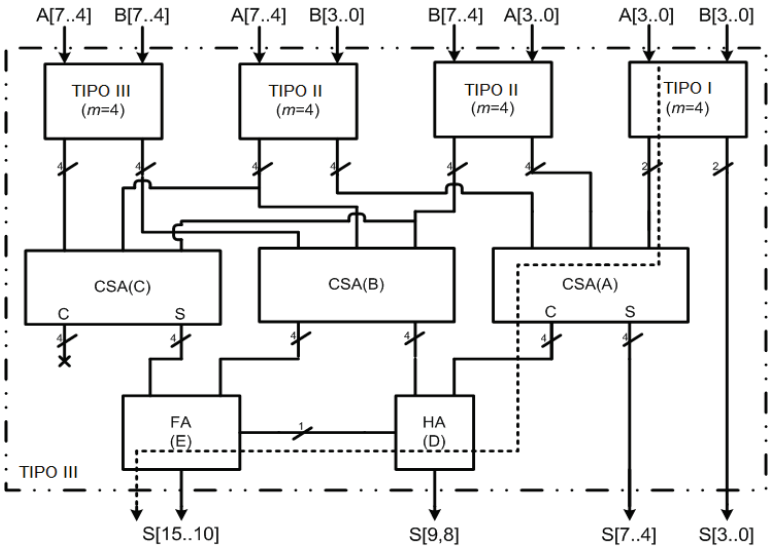

Figura 13: Estrutura base 256 tipo III
Fonte: Pieper, 2008

Tabela 1: Resultados para multiplicadores *m*=4

		Área (literais)		Atraso (ns)		Potência (W)	
16 bits	Original (COSTA,2002)	14983		206,2		0,22925	
	Otimizado (PIEPER, 2008)	5102	-65,95(%)	205,70	-0,24(%)	0,15812	-31,03(%)
32 bits	Original (COSTA,2002)	61935		431,4		1,4635	
	Otimizado (PIEPER, 2008)	20166	-67,44(%)	430,90	-0,12(%)	1,126	-23,06(%)
64 bits	Original (COSTA,2002)	251551		881,3		10,3773	
	Otimizado (PIEPER, 2008)	79862	-68,25(%)	881,30	-0,06(%)	8,8869	-14,36(%)

De acordo com os valores mostrados na tabela 1, podem-se observar os ganhos obtidos com o multiplicador *array* proposto em Pieper (2008). Os principais motivos para os ganhos se devem às estruturas mais simples dos blocos dedicados de multiplicação, que utilizam somadores eficientes do tipo CSA ao longo da arquitetura e blocos mais simples de multiplicação (*m*=2) como base. Dessa forma, esses circuitos multiplicadores são utilizados neste trabalho como base para as novas estruturas de multiplicadores na codificação Híbrida propostas.

Porém, vale ressaltar que, em Pieper (2008), não foram implementados multiplicadores *array* binários com estrutura irregular com números ímpares de bits. O trabalho proposto aborda a implementação de multiplicadores *array* a uma estrutura regular, assim como a uma estrutura irregular para uma aplicação com número ímpar de bits, tanto em codificação binária como em codificação Híbrida.

4.1.2 Multiplicador *array* Híbrido BASE-2^m proposto neste estudo

Assim como são realizadas as multiplicações binárias com multiplicadores Binários do tipo *array*, uma operação de multiplicação pode ser realizada em uma codificação diferente. Um esquema de codificação denominado codificação Híbrida foi proposto em Costa (2002), em que uma multiplicação de *m* bits pode ser realizada de forma regular. A seguir são mostradas as principais características da codificação Híbrida, bem como a nova estrutura proposta no âmbito deste trabalho.

4.1.2.1 Código Híbrido

A ideia do código Híbrido é de dividir os operandos em grupos de *m* bits, codificar cada grupo utilizando o código Gray e utilizar o comportamento do código Binário para propagar o *carry* entre os grupos. Assim, o número de transições em cada grupo pode ser reduzido e uma estrutura regular pode ser obtida, onde os grupos menos significativos do resultado dependem somente dos grupos menos significativos dos operadores.

A tabela 2 mostra as representações dos códigos Binário, Híbrido (*m*=2) e Gray para números de 4 bits, em complemento de 2. Pode ser observado que o código Híbrido apresenta um compromisso entre a mínima dependência das entradas de dados apresentada pelo código Binário e a característica de baixa atividade de chaveamento apresentada pelo código Gray.

Tabela 2: Representações dos códigos Binário, Híbrido (*m*=2) e Gray em complemento de 2

Decimal	Binário	Híbrido	Gray
0	0000	0000	0000
1	0001	0001	0001
2	0010	0011	0011
3	0011	0010	0010
4	0100	0100	0110
5	0101	0101	0111
6	0110	0111	0101
7	0111	0110	0100
-8	1000	1100	1100
-7	1001	1101	1101
-6	1010	1111	1111
-5	1011	1110	1110
-4	1100	1000	1010
-3	1101	1001	1011
-2	1110	1011	1001
-1	1111	1010	1000

Fonte: Costa, 2002

A tabela 3 apresenta o número de transições dos códigos Binário, Híbrido e Gray. Como pode ser observado, o código Híbrido é uma situação intermediária entre os códigos Binário e Gray em termos de número de transições. Para situações em que a capacitância chaveada no barramento de dados é significativa e em que os dados apresentam um alto grau de correlação, a utilização do código Híbrido

pode reduzir o consumo de potência em cerca de um terço em relação ao código Binário.

Tabela 3: Número de transições dos códigos Binário, Híbrido e Gray

Número de Bits	Número de Transições			Diferença	
	Binário	Gray	Híbrido (*m*=2)	Hib→Bin	Hib→Gray
***n* = 4 bits** (0, 1,..15,0)	30	16	20	-33,3%	+25%
***n* = 8 bits** (0, 1,..255,0)	510	256	340	-33,3%	+32,8%
***n* = 16 bits** (0, 1,..65535,0)	131070	65536	87380	-33,3%	+33,3%

Fonte: Costa, 2002

Assim, o cálculo geral do número de transições, em sequências de contagem de números, é dado de acordo com as equações (25), (26) e (27) para os códigos Binário, Gray e Híbrido respectivamente (COSTA, 2002).

$$n°\_trans.(Bin) = 2^{n+1} - 2 \tag{25}$$

$$n°\_trans.(Gray) = 2^{n} \tag{26}$$

$$n°\_trans.(Hib) = 2^{m}\frac{2^{n}-1}{2^{m}-1} \tag{27}$$

Para o caso particular de *m*=2, o cálculo do número de transições para o código Híbrido é dado pela equação (28) (COSTA, 2002).

$$n°\_trans\_Hib(m=2) = \frac{4}{3}\left(2^{n}-1\right) \tag{28}$$

Além da redução do número de transições, o código Híbrido possibilita a geração de circuitos somadores e multiplicadores com estruturas regulares, como poderá ser visto no próximo capítulo.

Outra vantagem apresentada pelo código Híbrido, com *m*=2, é a simplicidade na mudança de representação para o código Binário, como mostra a figura 14. Assim, o processo de codificação e decodificação dos dados utiliza baixa complexidade com apenas uma porta XOR ligada a cada grupo de *m* = 2 bits.

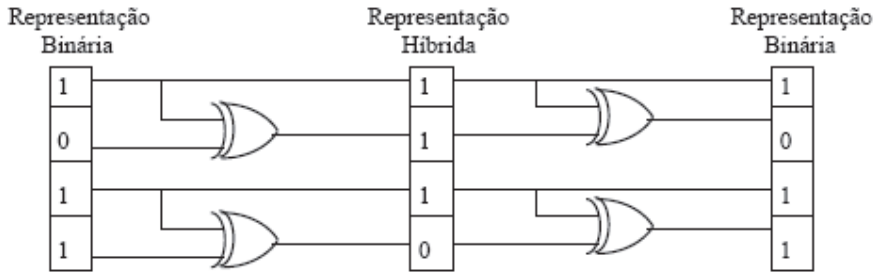

Figura 14: Conversão entre os códigos Binário e Híbrido
Fonte: Costa, 2002.

4.1.2.2 Multiplicador Híbrido na Base 4

A figura 15 mostra um exemplo de multiplicação de 4 bits na codificação Híbrida na base 4 (*m*=2). Pode-se observar, no exemplo da figura 15, que três tipos de multiplicação são realizados, sendo uma multiplicação do Tipo I, que realiza uma multiplicação sem sinal (11 x 11 = 2 x 2), duas multiplicações do Tipo II, que realizam multiplicações de um termo sem signal por um termo com sinal (11 x 11 = -2 x 2 ou 2 x -2) e uma multiplicação do Tipo III, que realiza a multiplicação de dois termos com sinal (11 x 11 = -2 x -2). Deve-se também observar, na figura, que a extensão do sinal é realizada pelo termo (10), que representa -1 na codificação Híbrida.

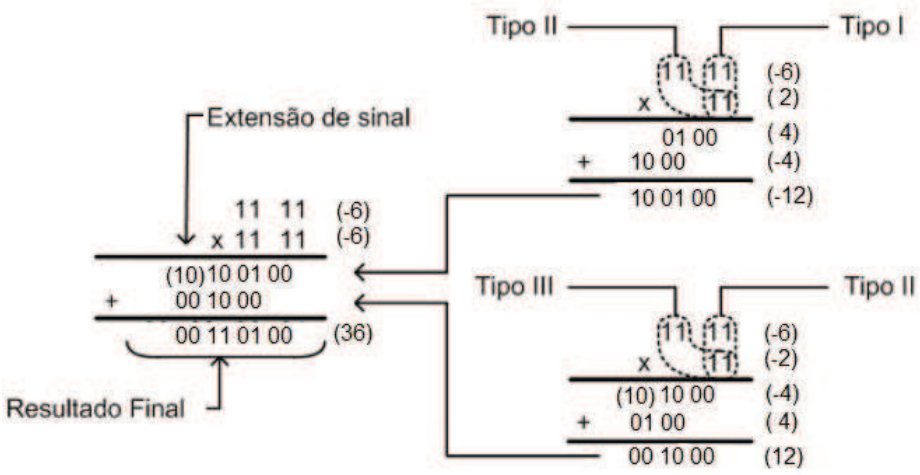

Figura 15: Exemplo de multiplicação numérica de 4 bits na base 4 (*m*=2) na codificação híbrida

A arquitetura de um multiplicador Híbrido pode ser vista na figura 16, para operação com largura de bits *W*=8, base-4 (*m*=2) em complemento de 2.

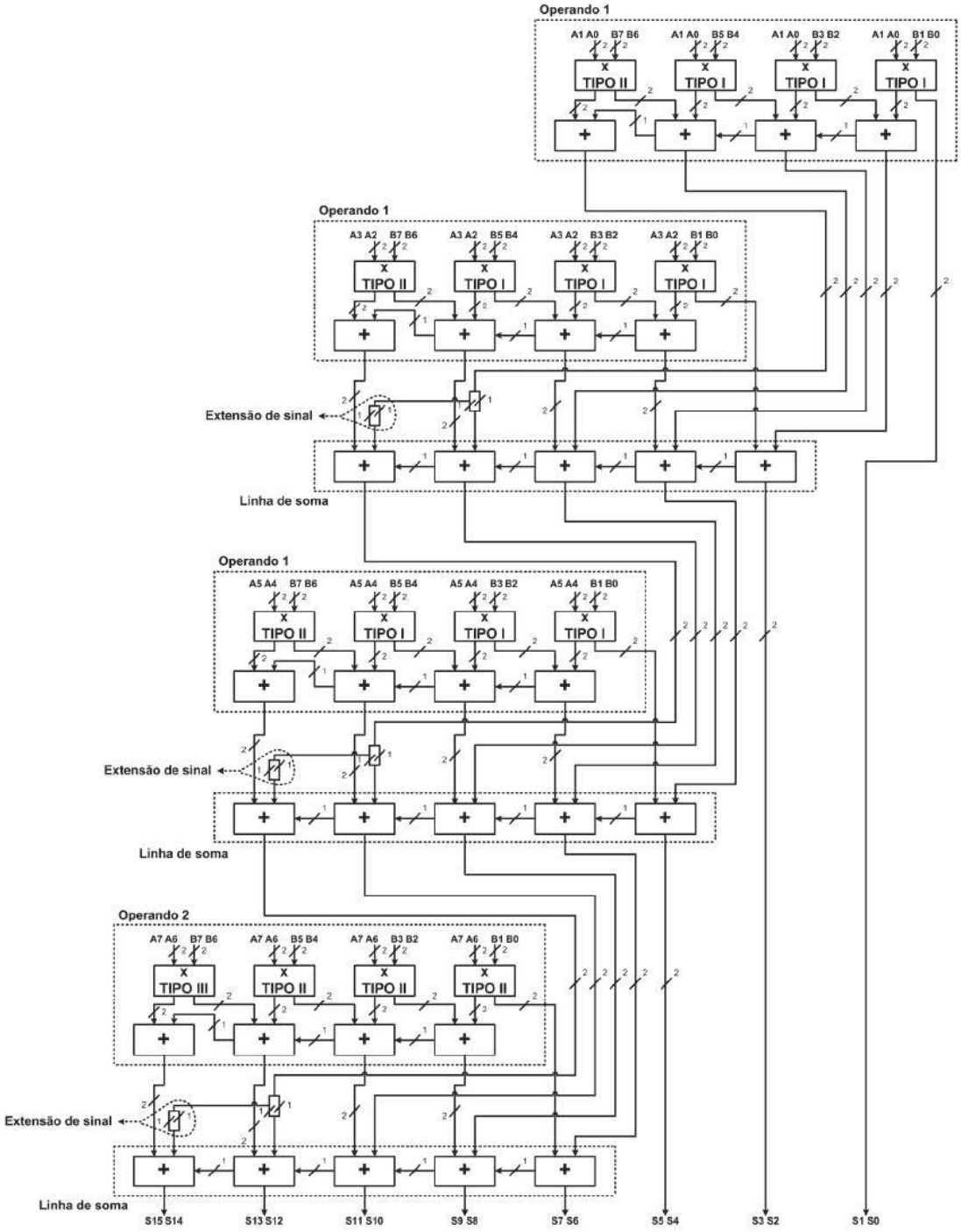

Figura 16: Arquitetura do multiplicador *array* híbrido base-4 (*m*=2) em complemento de 2
Fonte: Pieper, 2008.

Como pode ser visto na figura 16, o multiplicador Híbrido *array* apresenta uma estrutura regular, com os elementos básicos sendo utilizados para realizar os produtos parciais. Os dois bits menos significativos são obtidos imediatamente após a primeira multiplicação. Os demais bits são obtidos pela soma dos termos dos produtos parciais. Pode ser observado também que se utiliza uma linha de circuitos

somadores responsáveis pelas somas dos termos dos produtos parciais. Para uma arquitetura em código Híbrido de *W* bits, serão utilizados $\left(\frac{W}{2}\right)-1$ de linhas de somadores.

Com os módulos regulares de multiplicação, tornou-se possível a implementação de multiplicadores *array* eficientes, *m*=2, com 18 e 36 bits (que são necessários para a utilização na arquitetura do filtro adaptativo LMS). No próximo capítulo deste trabalho, são apresentadas as análises desses circuitos. Entretanto, além dos multiplicadores regulares, a arquitetura do filtro adaptativo também necessita de uma estrutura de multiplicação com 23 bits. Para este tipo de multiplicador, torna-se necessária a implementação de uma estrutura irregular de multiplicação, cuja arquitetura foi desenvolvida neste trabalho, e é mostrada a seguir.

4.1.2.3 Multiplicador Híbrido Proposto neste estudo

Para um multiplicador de 23 bits, cujo número de bits não é divisível por 2, e não podem ser aplicadas multiplicações em grupos regulares de multiplicação com *m*=2, foi necessária a aplicação de módulos de multiplicação diferenciados.

A estratégia utilizada foi a de criar novos módulos para os bits mais significativos, com multiplicações simultâneas de 3 bits (base 8 - *m*=3) em complemento de 2 e módulos com multiplicações de 3 bits por 2 bits (3:2), em que apenas o grupo de 3 bits é em complemento de 2.

Esse módulo 3:2 é constituído de:

- um módulo Tipo I, já visto anteriormente, que multiplica dois grupos numéricos sem levar em consideração o sinal,

- um módulo 2:1, com uma multiplicação de um grupo de 2 bits, sem sinal, por apenas 1 bit em complemento de 2, além de circuitos somadores.

O módulo *m*=3 é constituído de um módulo Tipo I e outros dois módulos 2:1, além de circuitos somadores.

Para a implementação de uma arquitetura do multiplicador *array* otimizado Híbrido 2:1, são utilizados elementos básicos, como somadores e

multiplicadores, que podem ser vistos na figura 17. As multiplicações possuem entrada em código Híbrido e saída em código Híbrido.

Figura 17: Exemplos de multiplicações numéricas em código híbrido.

A figura 18 mostra a arquitetura para os exemplos da figura 17. Como pode ser visto na figura 18, são necessários dois módulos de multiplicação, um do tipo I e outro 2:1 e circuitos somadores.

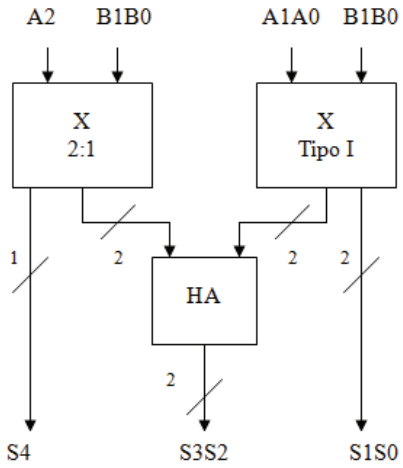

Figura 18: Bloco de multiplicação 3:2

Através da estrutura da figura 18, é possível observar a multiplicação de um grupo de 3 bits (*m*=3) em complemento de 2, por um grupo de 2 bits (*m*=2), sem levar o sinal em consideração, em codificação Híbrida. Assim, essa mesma proposta é utilizada no multiplicador de 23 bits, para quando os 3 bits mais significativos de uma palavra estão sendo multiplicados pelos bits menos significativos, em grupos de 2 bits (*m*=2), de uma segunda palavra.

Para implementar uma arquitetura do multiplicador *array* otimizado Híbrido *m*=3, são utilizados elementos básicos como somadores e multiplicadores, que podem ser vistos na figura 19. As multiplicações possuem entrada em código Híbrido e saída em código Híbrido.

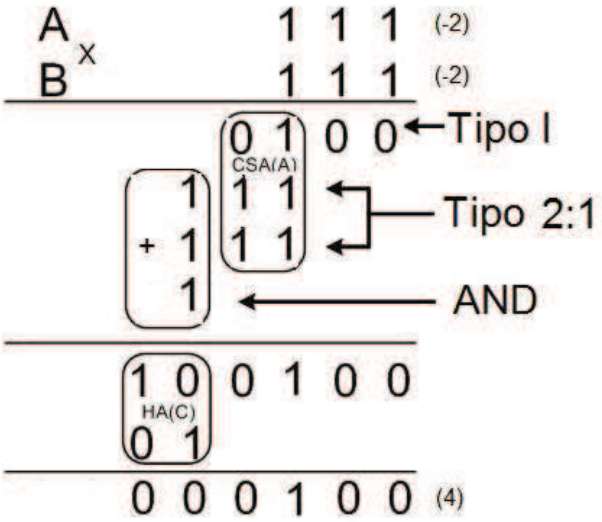

Figura 19: Exemplo de multiplicação numérica *m*=3 em código híbrido

No exemplo da figura 19, pode ser visto que, na multiplicação de dois valores em complemento de 2, se torna necessário um módulo Tipo I e dois módulos 2:1, além de circuitos somadores. Um somador do tipo *Carry Save* (CSA) é utilizado para acelerar o processamento das somas parciais.

Através dessa estrutura da figura 20, é possível observar a multiplicação de um grupo de 3 bits (*m*=3) por outro grupo de 3 bits (*m*=3) em complemento de 2, em codificação Híbrida.

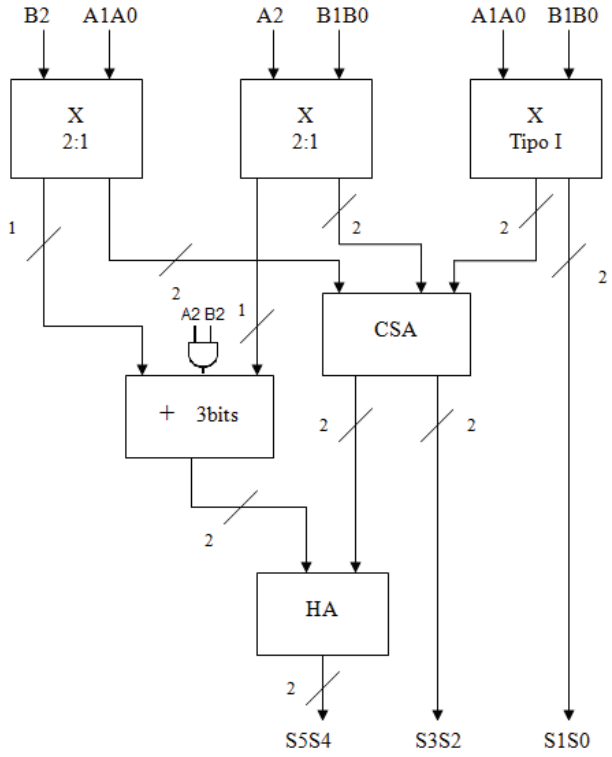

Figura 20: Bloco de multiplicação 3:3

Assim, essa mesma proposta é utilizada no multiplicador de 23 bits, para quando os 3 bits mais significativos de uma palavra estão sendo multiplicados pelos 3 bits mais significativos de uma segunda palavra.

A figura 21 apresenta um exemplo de multiplicação com largura de bits *W*=5 em código Híbrido, e com resultado em código Híbrido.

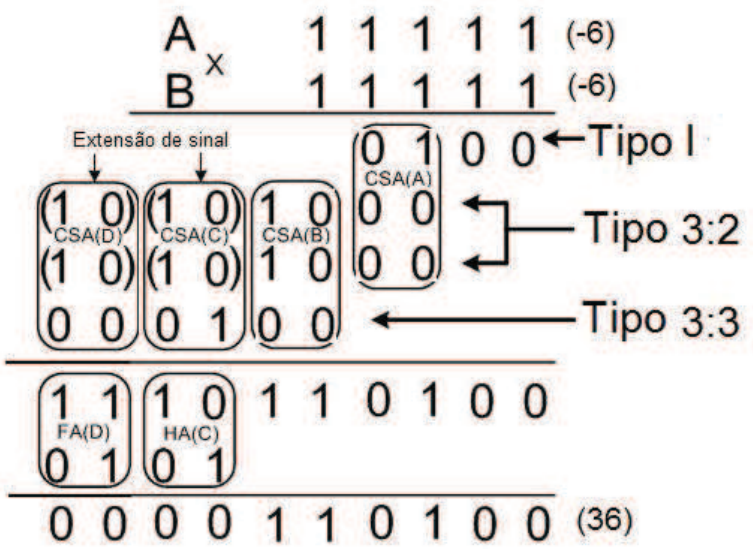

Figura 21: Exemplo de multiplicação numérica de dois números, ímpares, em código híbrido

Como apresentado anteriormente, e podendo ser observado na figura 21, para uma arquitetura de multiplicador *array* otimizado Híbrido com uma largura de bits ímpar, são necessários diferentes módulos de multiplicação para os 3 bits mais significativos, os quais, neste trabalho, foram implementados para um multiplicador *array* otimizado Híbrido de 23 bits.

4.2 RESULTADOS OBTIDOS COM OS MULTIPLICADORES *ARRAY*

Os circuitos multiplicadores, nas codificações Binária e Híbrida, foram implementados em linguagem de descrição de *hardware*, no ambiente Quartus II, da Altera.

Para os testes e a síntese dos circuitos implementados, foi utilizado o dispositivo '*Cyclone* II EP2C70F896I8'. A área é dada em blocos lógicos (*combinational* ALUT), o atraso em *ns* (nano segundos) e o consumo de potência por amostra em *mW* (mili *Watts*).

Para a análise de potência, área e atraso, os testes foram realizados com sinais senoidal e randômico nas entradas dos circuitos. Esses estímulos de entrada foram obtidos através de um programa desenvolvido em linguagem *C* de programação, no ambiente MatLab, com 10000 amostras.

A tabela 4 apresenta os resultados obtidos para os multiplicadores *array* otimizados de 18, 23 e 36 bits, em códigos Binário e Híbrido, para estímulos de entrada senoidal e randômico.

Tabela 4: Resultados de área, atraso e potência dos multiplicadores array binário e híbrido

Multiplicador		Área (*ALUT*)	Atraso (*ns*)	Potência (*mW*)	
				senoidal	randômico
18 bits	Binário	866	68,43	7,18	9,05
	Híbrido	940	66,17	6,53	7,82
23 bits	Binário	1643	79,84	16,64	18,44
	Híbrido	1743	69,65	16,84	18,70
36 bits	Binário	3521	136,58	51,37	59,75
	Híbrido	3829	142,05	48,22	56,25

Quanto a valores de área e atraso, os multiplicadores *array* em codificação Binária mostraram-se mais eficazes, exceto pelo valor de atraso dos multiplicadores de 18 e 23 bits em codificação Híbrida, que obtiveram uma redução de 3,3% e 12,7% respectivamente. Para estruturas com um maior número de bits é necessária uma maior quantidade de módulos de multiplicação, tornando-se inevitável o uso de mais portas lógicas *xor* para a conversão entre os códigos binário e híbrido, acarretando, assim, um maior caminho crítico levando a um maior valor de atraso na arquitetura.

Como já era esperado, os multiplicadores com estímulo de entrada senoidal obtiveram um consumo de potência maior do que os multiplicadores com estímulo de entrada randômico. Isto se dá devido ao fato de que, para uma entrada senoidal, seus dados têm uma correlação maior do que nos dados com entrada randômica (COSTA, 2002).

Os resultados mostraram a eficiência na redução de potência para multiplicadores *array* na estrutura Híbrida com estruturas regulares (18 e 36 bits) em relação aos multiplicadores Binários. Entretanto, na multiplicação com um número ímpar de bits, 23 bits, já não se pode dizer o mesmo, pois a multiplicação em código

Binário obteve um consumo de potência menor. Isso comprova a base teórica da codificação Híbrida, que apresenta eficiência na redução da atividade de transição para estruturas regulares, com valores de *m* maiores ou iguais a 2. Entretanto, mesmo com valores de multiplicação de 23 bits (onde são usadas estruturas de multiplicação irregulares *m*=3), os valores dos multiplicadores Binário e Híbrido são praticamente os mesmos, como pode ser visto na tabela 4.

Embora haja ambientes mais apropriados para a estimativa dos valores de potência, para uma primeira análise dos resultados foi utilizada a ferramenta *PowerPlay Power Analyzer Tool* do Quartus II, da Altera. Apesar de esta ferramenta possuir um aspecto de reconfiguração interna dos componentes FPGA que causam um consumo de potência extra, os resultados obtidos mostram que os multiplicadores Híbridos são promissores e podem ser utilizados em aplicações que demandem redução de potência (como os filtros digitais).

4.3 RESUMO

Este capítulo apresentou as principais características dos circuitos multiplicadores *array* base-2^m, e a possível operação em uma diferente codificação, a codificação Híbrida. Também foi apresentada uma nova estrutura irregular para multiplicadores com largura de bits ímpares, tanto em código Binário, como em código Híbrido. Assim, como proposta deste trabalho, foram implementados multiplicadores *array* eficientes Binários e Híbridos, para multiplicações com 18, 23 e 36 bits. Os códigos em VHDL encontram-se no cd-rom, em anexo.

5 IMPLEMENTAÇÃO DA ARQUITETURA DEDICADA DE FILTRO ADAPTATIVO PARA CANCELAMENTO DE HARMÔNICAS

No âmbito deste trabalho, além dos circuitos multiplicadores propostos, foi também implementada uma arquitetura dedicada de filtro adaptativo, utilizando o algoritmo LMS e uma estrutura dedicada para geração de harmônicas a partir de um sinal referência senoidal com frequência de 60 hertz (Hz).

A estrutura proposta utiliza o algoritmo LMS de apenas dois coeficientes, pois se trabalha com um sinal de entrada senoidal (HAYKIN, 1996). Essa implementação visa à praticidade, visto que o filtro LMS é replicado para o cancelamento de harmônicas.

A partir da implementação dos multiplicadores *array* Híbridos, descritos anteriormente, pôde-se também desenvolver uma arquitetura de filtro adaptativo dedicada em codificação Híbrida, cujos resultados serão apresentados no capítulo 6.

A figura 22 representa o diagrama de blocos da estrutura desenvolvida. A estrutura foi proposta a partir de um cancelador de harmônicas apresentado em Costa e Tavares (2009), implementado em DSP (*Digital Signal Processor*), trabalhando com sinais em ponto flutuante. A nova arquitetura apresentada neste trabalho foi adaptada para a utilização em ponto fixo, devido ao fato de os demais circuitos utilizados (multiplicadores, somadores, subtratores) trabalharem em ponto fixo, como também com o intuito de tornar a arquitetura mais rápida e oferecer o baixo custo como atrativo (MANOLAKIS, 2005).

As interferências sobre a amplitude de harmônicas superiores podem ser realizadas através de relações trigonométricas e da amplitude da componente fundamental. Assim, as estimativas das harmônicas de ordem superior podem ser obtidas diretamente a partir das amostras do sinal de 60Hz.

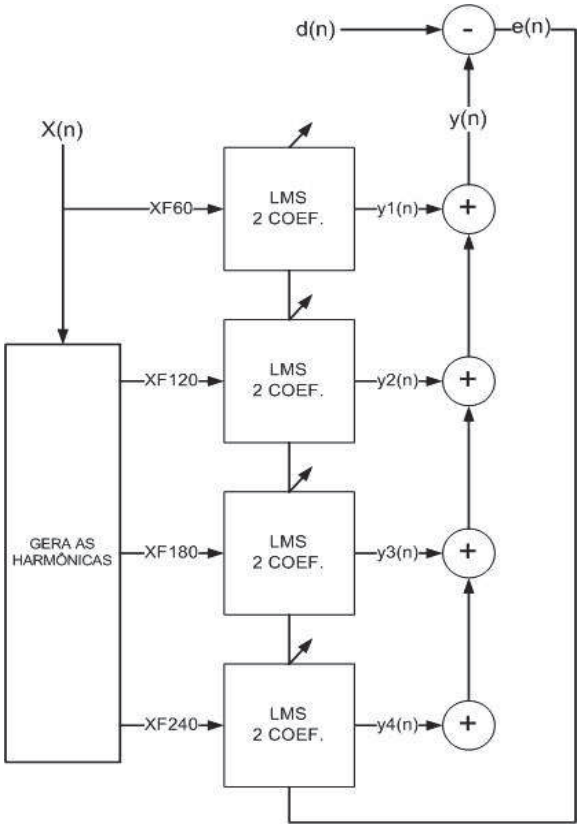

Figura 22: Estrutura do projeto para filtragem adaptativa desenvovido

Pode ser visto que o filtro desenvolvido é composto de um bloco gerador de harmônicas e blocos do filtro adaptativo LMS, além de circuitos somadores e subtratores.

A partir do sinal de referência de 60Hz, o algoritmo realiza a estimativa das harmônicas superiores, utilizando esses resultados para o cancelamento da interferência associada ao sinal de interesse. Mais adiante são apresentadas as equações que modelam a geração de sinais que representam as possíveis harmônicas presentes no sinal desejado (eletrocardiograma, por exemplo).

Inicialmente foi criado um programa, desenvolvido no ambiente MatLab em linguagem *C* de programação, que gera o sinal de interferência *x(n)* em códigos Binário e Híbrido. Esse programa também gera o sinal desejado, *d(n)*, que se refere ao sinal de ECG, previamente obtido, contaminado com sinais sinusoidal com frequências de 60Hz, 120Hz, 180Hz e 240Hz (harmônicas), em códigos Binário e Híbrido.

O sinal de entrada *x(n)* é um sinal sinusoidal com frequência 60Hz. Usando como base o sinal *x(n)*, são gerados sinais sinusoidais com frequências de 120Hz, 180Hz e 240Hz, que representam as primeiras harmônicas do sinal *x(n)*. A

seguir, esses sinais harmônicos são aplicados em filtros adaptativos cuja estrutura é caracterizada pelo algoritmo LMS. Cada filtro tem seus coeficientes ajustados conforme equação de adaptação do algoritmo LMS. O somatório das saídas de cada filtro, $y_i(n)$, é então subtraída do sinal desejado $d(n)$, produzindo um sinal de erro $e(n)$. A eliminação das harmônicas no sinal desejado é representada pelo sinal de erro. Isso acontece quando os filtros adaptativos têm seus coeficientes ajustados corretamente.

A estrutura implementada foi desenvolvida no ambiente Quartus II da empresa Altera (ALTERA, 2006), utilizando-se a linguagem de descrição de *hardware* VHDL *(Very High Speed Integrated Circuits Hardware Description Language).*

Nas próximas seções deste capítulo, é apresentado o funcionamento dos blocos do gerador de harmônicas e o do filtro adaptativo LMS.

5.1 GERADOR DE HARMÔNICAS

O programa para a geração de harmônicas produz sinais sinusoidais com frequências de 120Hz, 180Hz e 240Hz com cada amostra representada em 18 bits. Devido ao fato de os filtros utilizando o algoritmo LMS implementados usarem apenas 2 coeficientes, o filtro com 16 bits não teve o sinal desejado livre de interferências, pois não foi capaz de rastrear e eliminar as componentes do sinal de interferência até níveis considerados adequados para o projeto. Tendo como objetivo utilizar o algoritmo LMS com apenas 2 coeficientes, verificou-se que ao aumentar a entrada do filtro para 18 bits o algoritmo eliminou a interferência até níveis ideais de projeto.

Para não utilizar um maior número de multiplicadores, nem multiplicadores com maior palavra binária, e para minimizar o erro de “truncagem”, o sinal de interferência $x(n)$ de 60Hz tem suas amostras representadas em 16 bits. Esta é uma estratégia utilizada devido ao fato de que as equações para a geração das harmônicas superiores apresentam operações de multiplicação, permitindo, assim, serem estas representadas apenas através do uso de deslocamentos (deslocamentos de 1 e 2 bits para a esquerda).

Assim, o sinal de interferência *x(n)* de 60Hz é processado através da arquitetura, para produzir as demais harmônicas. Na figura 23 está apresentada a estrutura do bloco que gera as harmônicas de interferências para entrada nos blocos LMS.

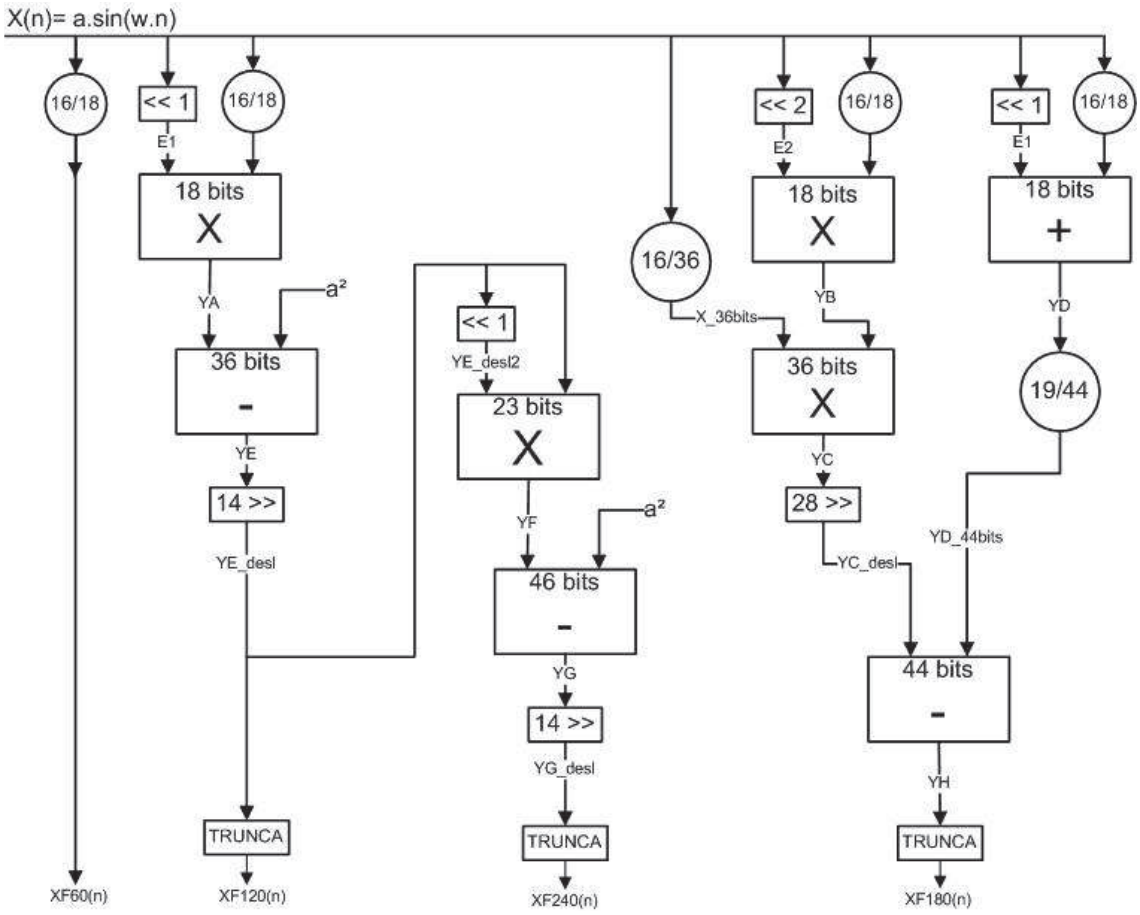

Figura 23: Estrutura do bloco gerador de harmônicas

A estrutura do bloco gerador de harmônicas é obtida através das equações (29), (30) e (31), que foram modificadas para cálculos em ponto fixo para esta arquitetura, a partir das equações apresentadas em Costa e Tavares (2006).

$$XF120 = \frac{2x^2 - a^2}{a} \tag{29}$$

$$XF180 = \frac{4x^3 - 3xa^2}{a^2} \tag{30}$$

$$XF240 = \frac{2(XF120)^2 - a^2}{a} \tag{31}$$

A amplitude do sinal de entrada é dada por *a*, sendo que, para a simulação desta estrutura, é utilizado um sinal de referência com amplitude de valor

decimal de -16384 a +16383, o qual equivale a 2^{14}, evitando, assim, a saturação deste sinal.

Quando é necessária a divisão pelo valor da amplitude nas equações para a obtenção das harmônicas superiores, foi utilizada a mesma técnica de deslocamento de sinal, desta vez de 14 bits para a direita, para obter uma divisão, sem a necessidade de utilizar um operador divisor.

Da mesma maneira, deslocando 28 bits para a direita, tem-se uma divisão pelo valor da amplitude elevada ao quadrado (a^2), necessária para a obtenção da harmônica de 180Hz.

Também foram realizadas concatenações dos sinais para algumas operações de multiplicação e subtração.

Os módulos “trunca” realizam a “truncagem” dos sinais, para que estes apresentem uma largura de dados de 18 bits, limitando os bits menos significativos e para, posteriormente, serem utilizados nos blocos dos filtros adaptativos LMS. Esta “truncagem” é realizada após os valores das harmônicas superiores serem obtidos por equações adequadas à amplitude do sinal de referência, visando minimizar os erros das operações.

Portanto, pode-se desenvolver uma arquitetura para a geração de harmônicas de frequência 120HZ, 180Hz e 240Hz de ponto fixo através dessas operações. Tendo isso em vista e a partir da implementação dos multiplicadores *array* Híbridos, pode-se também desenvolver essa mesma arquitetura de geração de harmônicas em codificação Híbrida.

5.2 FILTRO ADAPTATIVO LMS

Depois de gerados o sinal de entrada *x(n)*, assim como as harmônicas superiores, esses sinais são aplicados no filtro adaptativo LMS, que foi implementado em linguagem de descrição de *hardware* em código Binário e Híbrido.

A estrutura do filtro adaptativo usando o algoritmo LMS é mostrada na figura 24. As amostras do sinal $XF_i(n)$ são processadas pelo filtro adaptativo, composto por dois coeficientes w_o e w_1, (indicados na figura 24) e, como resposta desse processo de filtragem, uma amostra $y_i(n)$ é gerada. Este valor de $y_i(n)$ é,

então, subtraído de uma amostra de *d*(*n*). O resultado dessa operação é o valor do erro estimado *e*(*n*), ou seja, a saída do sistema.

Assim, um valor de erro *e(n)* é determinado comparando-se o resultado calculado *y(n)* com o valor desejado *d(n)*. Este valor de erro, ponderado por um passo de adaptação *μ* e pela entrada *x(n)*, é utilizado para o cálculo da correção dos coeficientes e dará origem ao coeficiente para próxima convolução. O passo de adaptação também determina a velocidade com que os coeficientes se adaptam, garantindo a convergência do sinal.

Esse processo se repetirá enquanto houver amostras a serem processadas. O que se espera da estrutura é que, ao final desse processo, a interferência contida no sinal contaminado tenha sido eliminada. Para tal, todos os dados devem ser manipulados como números de ponto fixo.

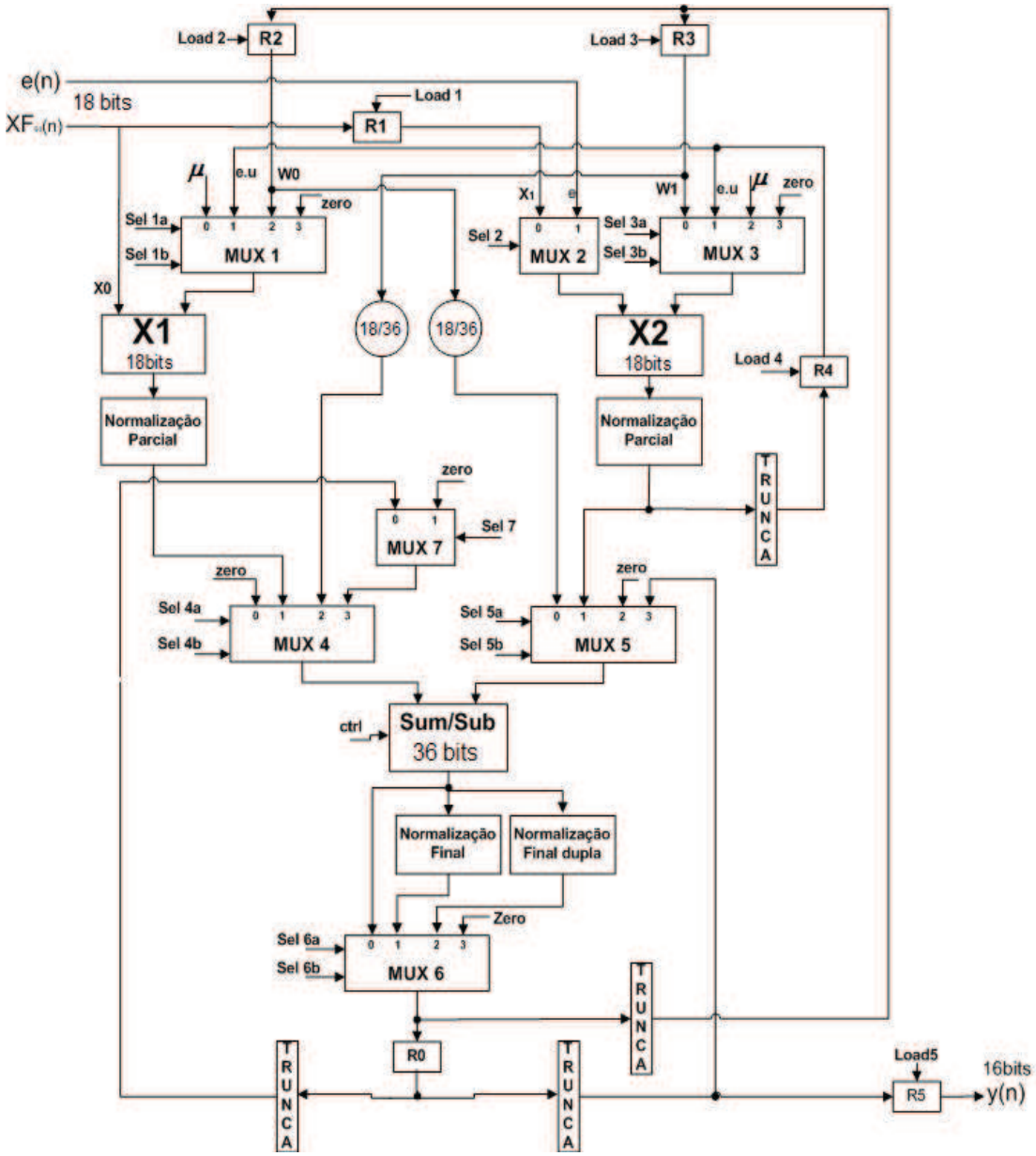

Figura 24: Estrutura do filtro adaptativo LMS

Na figura 24 pode-se observar a estrutura dos blocos LMS, onde o sinal de erro *e(n)* é capturado externamente, e, na saída, é apresentado o sinal da filtragem *y(n)*.

O filtro adaptativo implementado neste trabalho utiliza dois coeficientes para a etapa de filtragem. Para isso, utilizam-se dois circuitos multiplicadores em paralelo, o que possibilita a execução da equação de filtragem (equação (7)) em apenas um ciclo de relógio. Os circuitos multiplicadores utilizados nesta arquitetura foram implementados nas codificações Binária e Híbrida, e representam um circuito *array* de 18 bits que opera na base 2^m, com m=2, em complemento de 2. Esses circuitos multiplicadores são mostrados como blocos **X1** e **X2** na figura 24.

Para a realização de somas e subtrações, visto que estas operações são necessárias para o cálculo do erro, equação (8), filtragem, equação (7) e atualização dos coeficientes, equação (23), utiliza-se o bloco ***Sum/*Sub**, encontrado na própria biblioteca da ferramenta Quartus II, da Altera, mostrado na figura 24. Esse bloco apresenta, nas suas entradas, palavras de 36 bits (proporcionados por multiplexadores), pois as saídas dos circuitos multiplicadores **X1** e **X2** apresentam os dobros do número de entradas. A máquina de estados, que é responsável por indicar quando é necessária uma soma ou subtração, está omitida na figura 24 para a simplificação do desenho.

A utilização de circuitos multiplexadores, representados como **MUX** na figura 24, tem como objetivo possibilitar o reúso de um mesmo circuito aritmético em ciclos de relógio diferentes. A seleção dos sinais de controle para os multiplexadores está prevista na máquina de estados.

Os circuitos registradores com sinal de carga, **R1**, **R2**, **R3**, **R4** e **R5**, mostrados na figura 24, permitem a retenção temporária de valores previamente calculados e que serão utilizados nos cálculos posteriores. O registrador **R0** é um registrador sem sinal de carga, logo depende apenas do sinal de relógio para reter informações.

Devido aos problemas de convergência apresentados pelo algoritmo LMS, relacionados à sua sensibilidade à potência do sinal de entrada (HAYKIN, 2002), não conseguindo eliminar a interferência presente no sinal contaminado *d(n)*, foi feita uma compensação no passo de adaptação para sua normalização, através da inserção de módulos normalizadores, utilizando o formato Q15 (ITURRIET, 2008).

A idéia do formato Q15 consiste em transformar os valores utilizados nos cálculos para este formato. A conversão de um valor para o formato Q15 consiste na divisão deste mesmo valor por um fator normalizador. Nesse caso, o fator normalizador é o número $2^{15} = 32768$.

Na figura 25, encontra-se um exemplo do formato Q15, com um passo de adaptação (μ) com valor igual a 50, normalizando os números para uma faixa de valores de 16 bits.

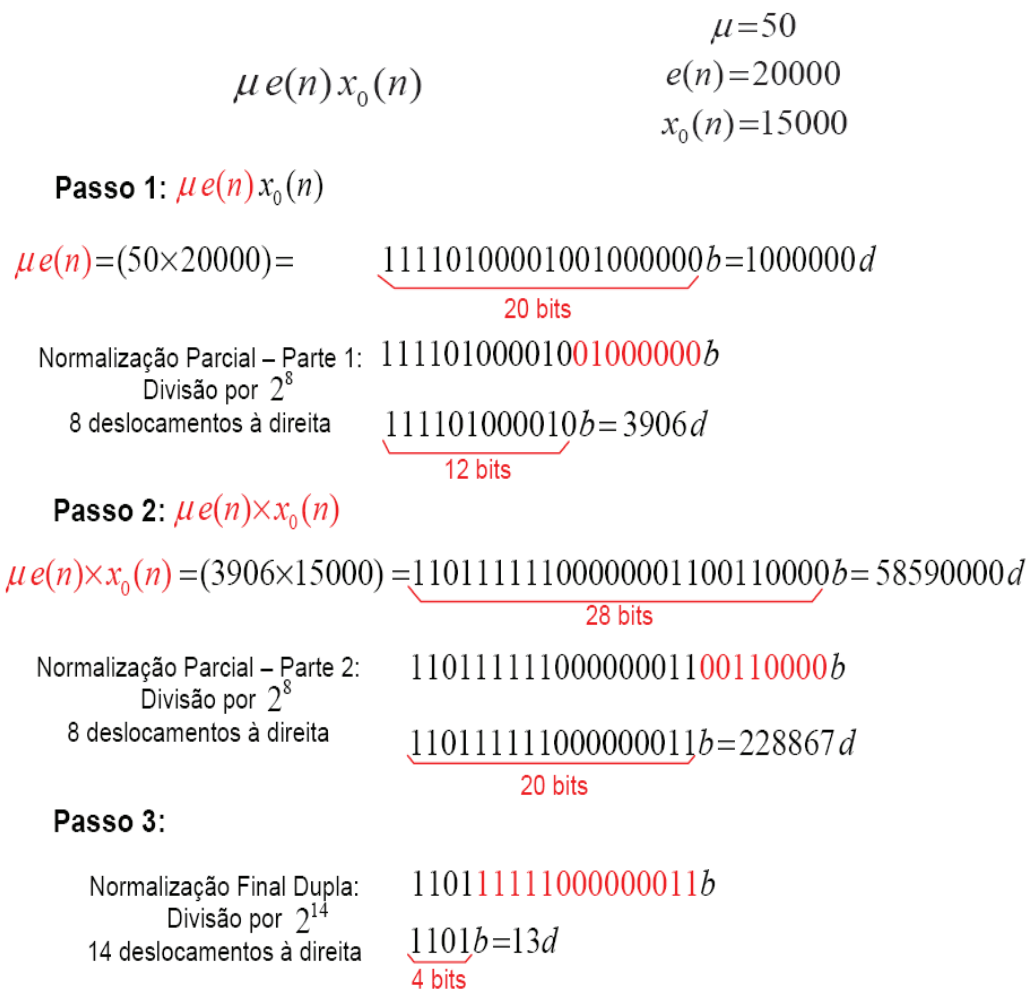

Figura 25: Exemplo de normalização Q15

Assim, esses módulos normalizadores inseridos foram divididos em dois blocos: Normalização Parcial e Normalização Final (COSTA, 1997). Essas normalizações são necessárias para colocar os valores no formato Q15.

A figura 26 representa o diagrama de blocos de funcionamento da estrutura do filtro adaptativo LMS. Os arquivos *.vec* representam os vetores com as amostras dos sinais *x(n)* e *d(n)* com valores do sinal de referência de 60Hz e sinal desejado, que é o sinal de referência de 60Hz com a presença de interferências, respectivamente.

Primeiramente são zerados os dados de entrada, as amostra de *x(n)* e os coeficientes do filtro encontram-se em valor zero. Logo, uma primeira amostra *x(n)* é processada, passando pelo processo de filtragem, onde se obtém um valor para *y(n)*. Então, são realizados os cálculos e atualização dos coeficientes. Por fim, um *buffer* circular desloca a amostra antiga de *x(n)*, para adquirir um novo valor para

este dado de entrada *x(n)* e recomeçar novamente a filtragem. Este processo se finalizará assim que a saída de erro se torne igual ou muito próxima ao sinal primário sem interferência.

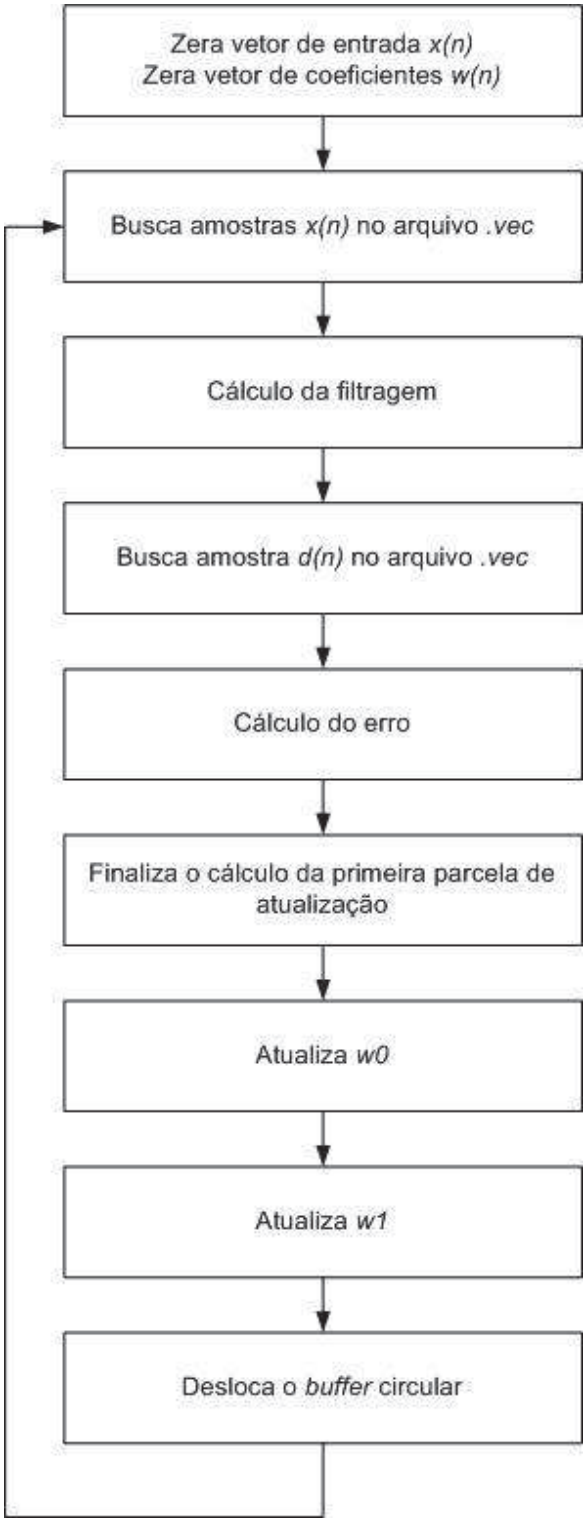

Figura 26: Diagrama de blocos da estrutura do filtro adaptativo LMS

Na arquitetura do filtro adaptativo LMS, o processamento completo de uma amostra é realizado em 7 ciclos de relógio, cujos passos podem ser vistos na tabela 5.

Tabela 5: Tabela com a descrição do processamento em cada ciclo de relógio

Ciclos	Cálculo	Definição
1°	$w_0(n).\ x_0(n) + w_1(n).\ x_1(n)$	Armazena o *y(n)* na saída.
2°	$e(n) = d(n) - y(n)$	Espera o sinal *e(n)* da subtração externa.
3°	$\mu.e(n)$	Multiplica o erro pelo o passo de adaptação.
4°	$w_1(n) + \mu.e(n).\ x_1(n)$	Calcula o próximo coeficiente.
5°	$w_1(n+1)$	Atualiza o valor.
6°	$w_0(n) + \mu.e(n).\ x_0(n)$	Calcula o próximo coeficiente.
7°	$w_0(n+1)$	Atualiza o valor.

Como pode ser observado na tabela 5, para cada amostra passar pelo processo de filtragem e ser concluída, são necessários 7 ciclos de relógio. A definição mais detalhada do processamento em cada um destes ciclos de relógio é apresentada a seguir:

- Ciclo 1 – Cálculo da saída do filtro *y(n)*

O processo do cálculo da filtragem é realizado utilizando os dois circuitos multiplicadores **X1** e **X2** simultaneamente. Quando uma amostra é inserida na posição x_0, o registrador **R1** retém esta amostra enquanto o processamento é executado. Antes de uma nova amostra ser inserida na posição x_0, o registrador **R1** é habilitado, permitindo a passagem desta amostra para a posição x_1.

Os valores resultantes das operações de multiplicação recebem a primeira etapa de normalização e, então, são somados. O resultado da soma em 36 bits recebe a etapa de normalização final e permanece retido no registrador **R0** até o próximo ciclo de relógio.

- Ciclo 2 – Cálculo do erro *e(n)*

Para o cálculo de erro, o valor de $y(n)$ passa pelo bloco “trunca”, que tem como objetivo limitar $y(n)$ nos 18 bits menos significativos, pois, no processo de normalização descrito anteriormente, os bits são deslocados para a direita, ou seja, a maior parte da representação do sinal processado está nos bits menos

significativos. Uma amostra $d(n)$ deve ser carregada, e o circuito ***Sum/Sub*** é habilitado pela máquina de estados para operar como um subtrator.

- Ciclo 3 – Cálculo do $\mu.e(n)$

Esta é a primeira parte do cálculo da parcela de atualização. Trata-se de uma etapa comum aos dois coeficientes, sendo calculada pela estrutura apenas uma vez. Neste ciclo de relógio, o único circuito aritmético utilizado é o multiplicador **X2** e, como resultado, é gerado um valor de 36 bits que, então, é normalizado e truncado. Importante observar que, neste ciclo, apenas uma normalização parcial é executada.

- Ciclo 4 – Cálculo do próximo coeficiente w_1

Para o cálculo da parcela de atualização do coeficiente w_1 mostrado na equação (23), os circuitos aritméticos utilizados são o multiplicador **X2** e o circuito ***Sum/Sub,*** operando como somador. Finalizada a multiplicação, uma nova normalização parcial é executada, totalizando duas normalizações parciais consecutivas. Para não interferir no resultado das operações, duas normalizações finais devem ser executadas, e, para isso, foi criado o bloco com o nome de "normalização final dupla". A soma com o valor zero (representado por **zero** na figura 22) foi desenvolvida de forma que o resultado não sofra nenhuma interferência quando a normalização final dupla for executada.

- Ciclo 5 – Atualização de w_1

Finalizado o cálculo da parcela de atualização do coeficiente w_1, resta agora somar esta parcela com o valor atual de w_1**.** O resultado dessa soma é truncado em 18 bits e, então, atualizado. O único circuito aritmético utilizado na realização do processo de atualização é o circuito somador. O bloco "trunca" neste cálculo garante que nenhum coeficiente assuma um valor maior do que 18 bits.

- Ciclo 6 – Cálculo do próximo coeficiente w_0

A estrutura utilizada neste ciclo é muito semelhante à utilizada no ciclo 4. A única diferença de implementação é que, neste ciclo, o multiplicador utilizado é o circuito **X1**. O procedimento de normalização também é o mesmo mostrado para o ciclo 4.

- Ciclo 7 – Atualização de w_0

Neste último ciclo de relógio, o coeficiente w_0 é finalmente atualizado. A máquina de estados retorna ao primeiro ciclo, e novas amostras $x(n)$ e $d(n)$ podem ser carregadas.

5.3 ARQUITETURA EM CODIFICAÇÃO HÍBRIDA

A partir do desenvolvimento do filtro adaptativo utilizando o algoritmo LMS para o cancelamento de interferências em codificação Binária, pôde ser implementado, para o mesmo fim, um filtro adaptativo LMS dedicado em codificação Híbrida. Para isso, foram utilizados os multiplicadores *array* base-2^m otimizados, descritos anteriormente neste trabalho, assim como outros operadores aritméticos, também já vistos através das figuras 22 e 23.

A estrutura utilizada para geração do filtro adaptativo LMS dedicado em codificação Híbrida é a mesma utilizada para o filtro em codificação Binária, exceto pela aplicação dos operadores desenvolvidos em codificação Híbrida e portas lógicas XOR ao longo do circuito, que realizam a função de codificadores/decodificadores. O sinal de entrada *x(n)* e o sinal contaminado *d(n)* também estão nesta codificação Híbrida, assim como os valores do passo de adaptação (μ) e da amplitude do sinal de entrada (a).

A figura 27 apresenta o bloco gerador de harmônicas. Para esta arquitetura operar em codificação Híbrida, os multiplicadores otimizados de 18, 23 e 36 bits utilizados operam em codificação Híbrida. As entradas desses multiplicadores estão em codificação Híbrida, porém apresentam uma saída Binária, para que os dados de saída possam ser utilizados pelos blocos somadores/subtratores. Isso ocorre, pois, como apresentado em Costa (2002); os somadores Híbridos não são uma boa alternativa quando se trata de ganho em consumo de potência. Estes operadores apresentaram altos valores no consumo de potência, área e atraso quando comparados a estruturas de somadores em codificação Binária. Dessa forma, os circuitos somadores/subtratores utlizados na arquitetura operam na codificação Binária.

Então, como em meio à estrutura existem outras operações que são realizadas em codificação Binária, são necessárias conversões da codificação

Binária para a codificação Híbrida e vice-versa. Logo, foram desenvolvidos conversores para realizarem estas operações. Na figura 27, encontram-se localizados os conversores implementados, representados por portas XOR, para viabilizar o correto funcionamento da estrutura.

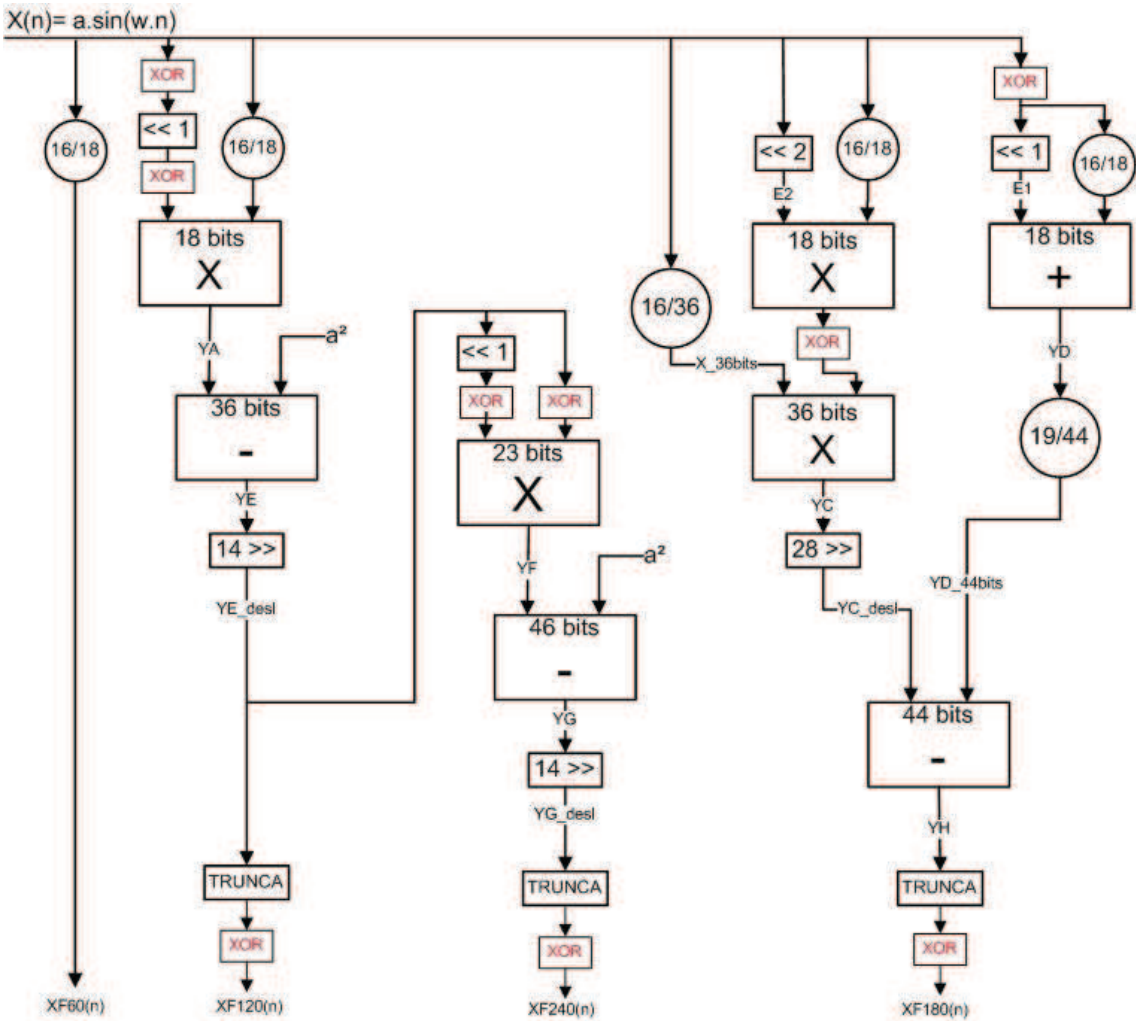

Figura 27: Bloco gerador de harmônicas com aplicação de conversores

Observa-se que as saídas do sistema, sinal de referência e suas harmônicas estão em codificação Híbrida, para, posteriormente, ser aplicados na arquitetura do filtro LMS.

A figura 28 apresenta a estrutura do filtro LMS utilizado. Os multiplicadores otimizados de 18 bits utilizados estão operando em codificação Híbrida. E, assim como para o bloco gerador de harmônicas, também foi necessária a aplicação de conversores Binário para Híbrido e Híbrido para Binário. Os locais de

implementação destes conversores estão representados na figura 28 pelas portas XOR.

Devido ao aumento no caminho crítico do circuito em codificação Híbrida, pela aplicação dos diversos conversores, consequentemente ocorrerá um acréscimo no consumo de potência desta arquitetura, como será visto posteriormente.

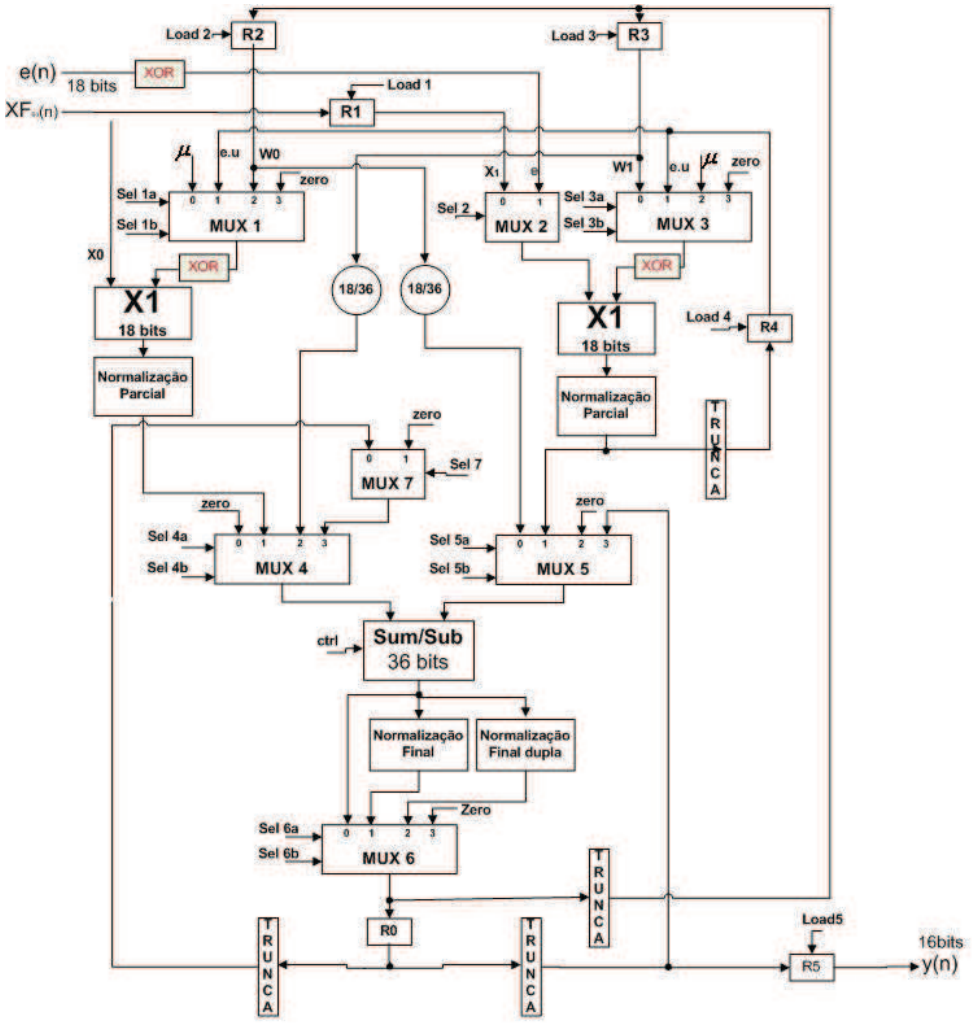

Figura 28: Bloco do filtro LMS com aplicação de conversores

5.4 RESUMO

Este capítulo apresentou a estrutura dedicada do filtro adaptativo LMS implementado como proposta deste trabalho, assim como os blocos que o formam e suas funcionalidades, operando em codificação Binária e Híbrida. Foram também

detalhadas as operações de cálculo realizadas em cada ciclo de relógio para o processo de filtragem.

6 VALIDAÇÃO DAS ARQUITETURAS DE FILTROS ADAPTATIVOS LMS NAS CODIFICAÇÕES BINÁRIA E HÍBRIDA

Foram implementadas em linguagem de descrição de *hardware*, no ambiente Quartus II, da Altera, arquiteturas dedicas de filtros adaptativos LMS em código Binário e Híbrido.

Para os testes, foi utilizado o dispositivo '*cyclone* II EP2C70F896I8'. A área é dada em blocos lógicos (*combinational* ALUT), o atraso em *ns* (nano segundos) e o consumo de potência em *mW* (mili *Watts*).

Para a análise de potência, área e atraso, os testes foram realizados com um sinal de entrada bioelétrico, sendo que, no caso deste trabalho, está sendo utilizado um sinal de ECG. Esse estímulo de entrada foi obtido através de um programa desenvolvido em linguagem *C* de programação, no ambiente MatLab, em código Binário e Híbrido.

Portanto, o filtro apresentado tem como característica o cancelamento das harmônicas de 60Hz e superiores, com uma complexidade computacional mínima tanto em codificação Binária, como também em uma codificação diferenciada, a codificação Híbrida.

Para realizar as simulações da estrutura, foram utilizadas as seguintes configurações:

- número de amostras = 10.000
- sinal de interferência *x(n)* = 60 Hz
- frequência de amostragem = 1k Hz
- frequência de relógio = 7k Hz
- passo de convergência (μ) = 80
- sinal desejado *d(n)* = sinal de ECG contaminado com sinais sinusoidais com frequências 60Hz, 120Hz (1° harmônica), 180Hz (2° harmônica) e 240Hz (3° harmônica).

Através dos programas gerados no ambiente MatLab, foi possível a verificação dos dados, em gráficos, dos sinais de interferência, sinal desejado e resultados em potência em dB, permitindo, assim, observar a qualidade do filtro para o seu propósito de cancelamento de interferências de um sinal bioelétrico.

A seguir, são apresentados os gráficos, comprovando o satisfatório funcionamento da arquitetura dedicada de filtro adaptativo LMS implementado, tanto em codificação Binária como em codificação Híbrida.

6.1 FILTRO ADAPTATIVO LMS DEDICADO EM CODIFICAÇÃO BINÁRIA

A figura 29 mostra o sinal de entrada senoidal do filtro adaptativo *x(n)* e a figura 30 mostra o sinal contaminado *d(n)* em codificação Binária.

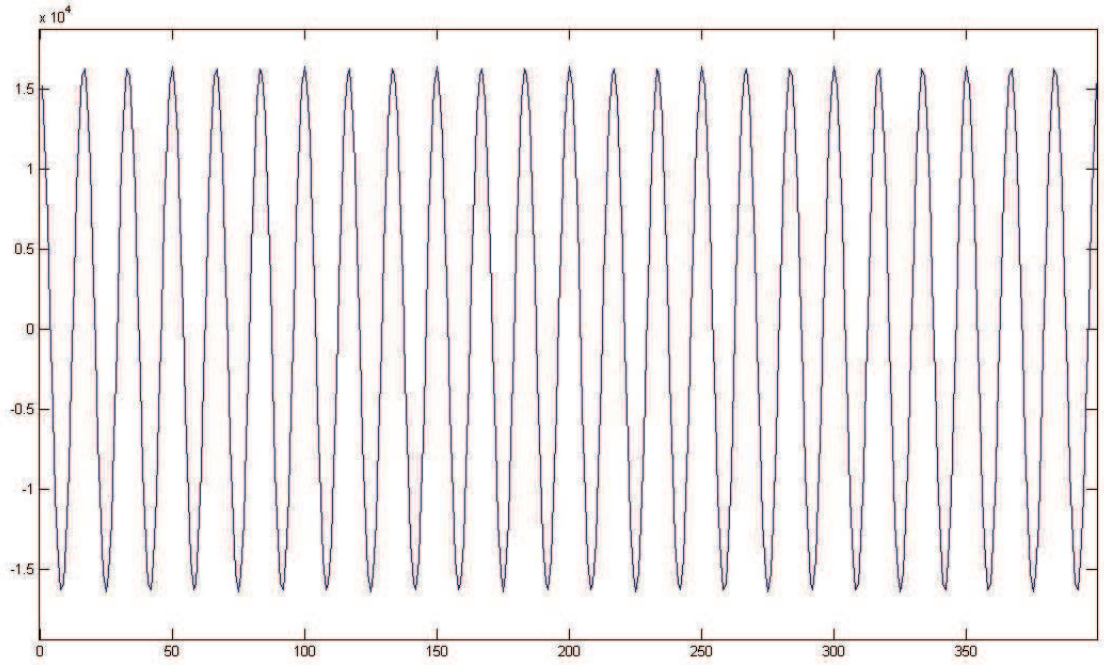

Figura 29: Sinal de interferência *x(n)* (sinusoidal 1k Hz)

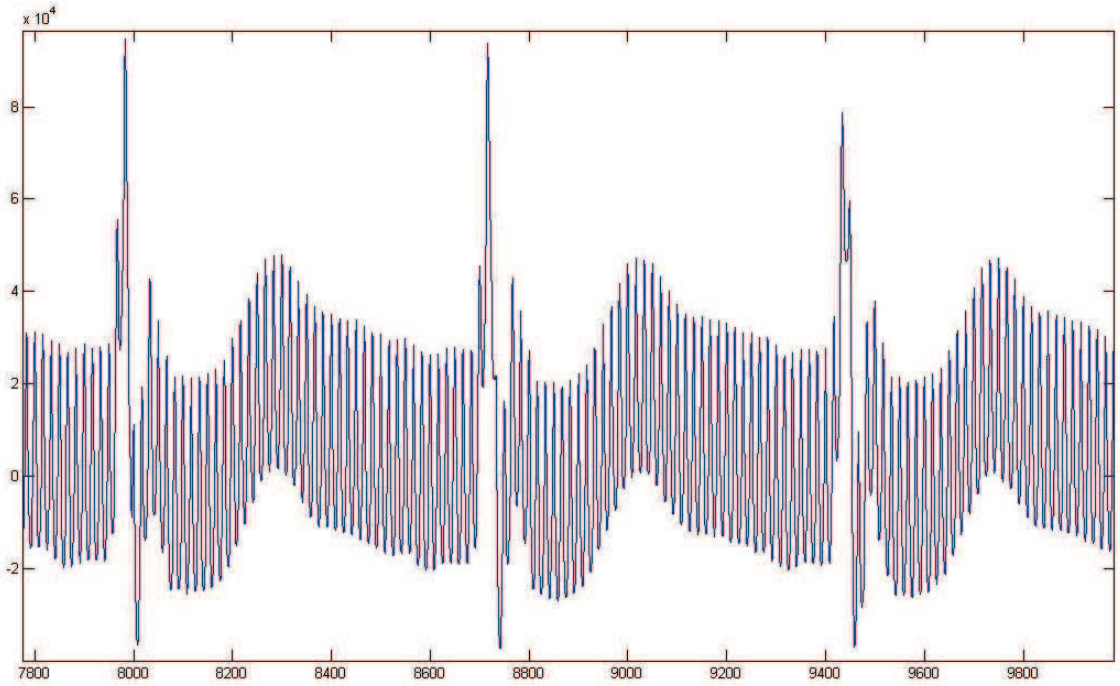

Figura 30: Sinal contaminado *d(n)*

As amostras do sinal *x*(*n*) são processadas pelo filtro adaptativo, e, como resposta desse processo de filtragem, uma amostra *y*(*n*) é gerada. Este valor de *y*(*n*) é então subtraído de uma amostra de *d*(*n*). O resultado dessa operação é o valor do erro estimado *e*(*n*), ou seja, a saída do sistema.

A figura 31 apresenta as 10000 amostras do sinal *e(n)* resultantes do processo de filtragem. Pode-se observar que o sinal se encontra livre de interferências. A figura 32 mostra as últimas amostras do processo antes e depois da filtragem, este livre de interferências.

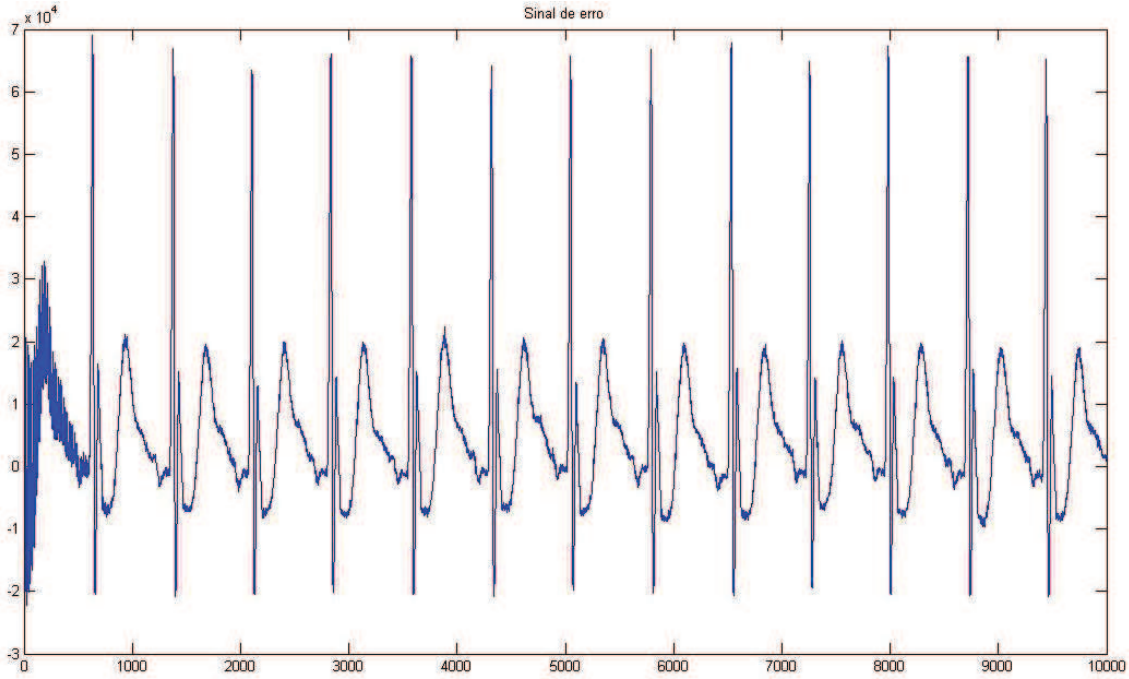

Figura 31: Sinal de saída do filtro adaptativo *e(n)*

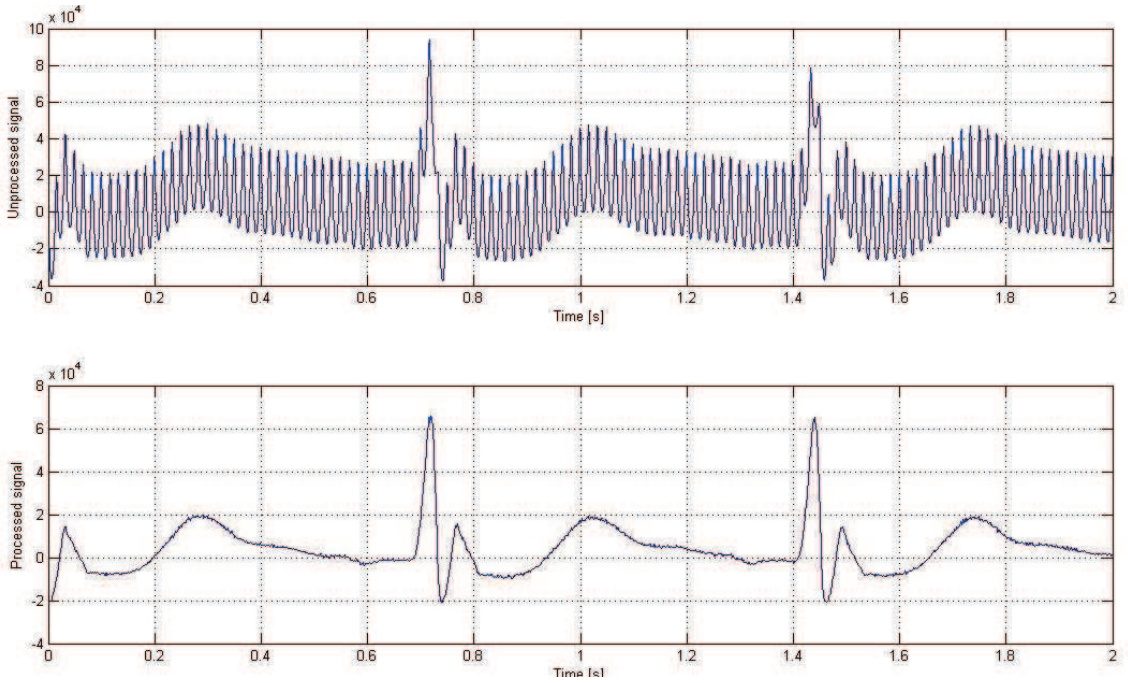

Figura 32: Últimas amostras do sinal antes da filtragem e do sinal após a filtragem (*e(n)*)

O gráfico de potência, da figura a seguir, demonstra as frequências contidas nos sinais de *e(n)* não processado e *e(n)* processado.

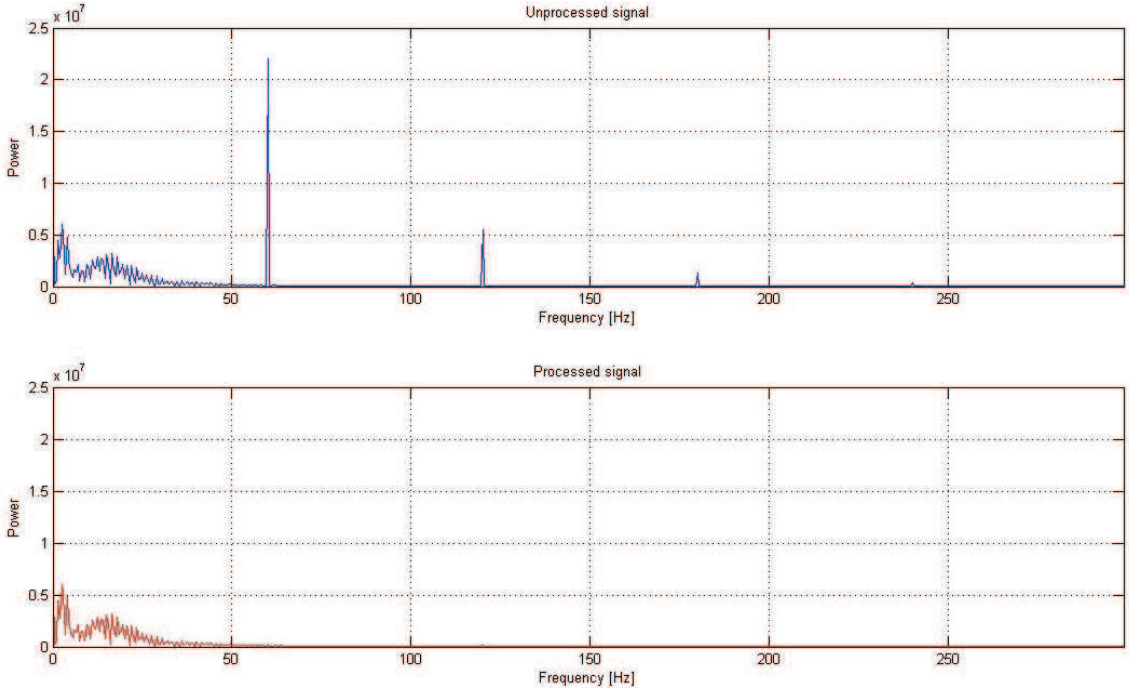

Figura 33: Gráfico de potência do sinal *e(n)* processado e não processado

Através dessa figura 33, pode-se observar que o sinal *e(n)* possui frequências em 60Hz, 120Hz, 180Hz e 240Hz. Essas frequências são filtradas e minimizadas, tornando-se praticamente nulas após o processo de filtragem descrito anteriormente.

Para fins de verificação, foi desenvolvido um programa no ambiente MatLab para o cálculo do consumo de potência desta arquitetura, dado em *dB*, em que se pode observar que as frequências de interferência foram eliminadas do sinal bioelétrico.

Na figura 34, pode ser vista a potência em *dB* do sinal *e(n)* não processado e do sinal *e(n)* processado.

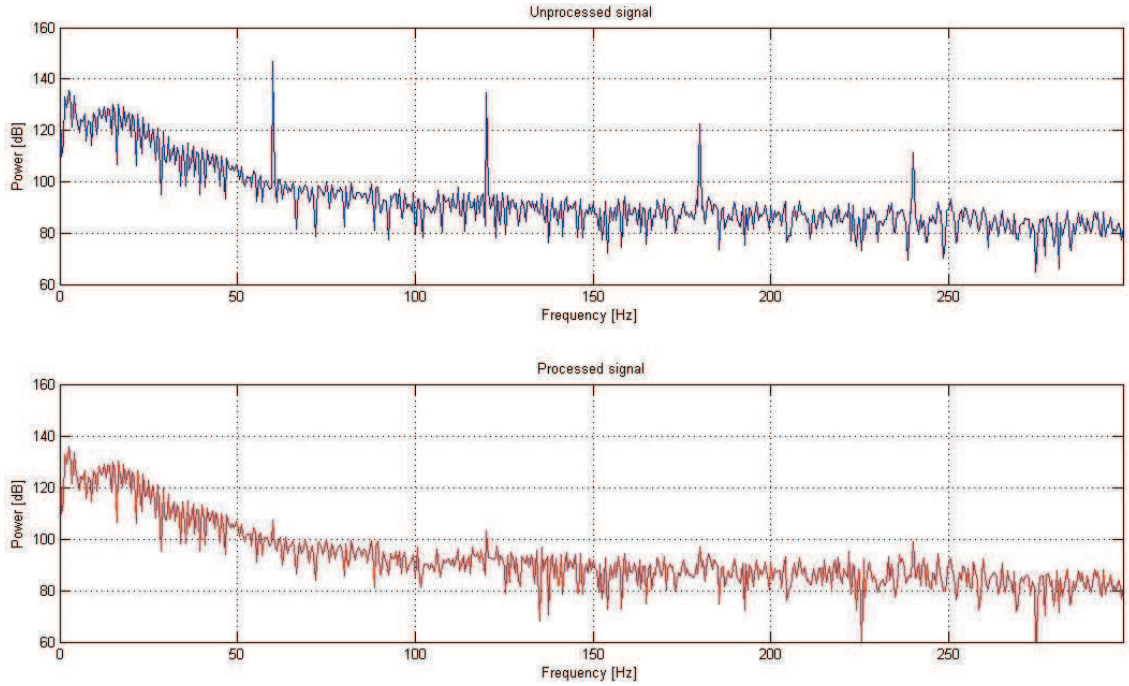

Figura 34: Potência em *dB* do sinal *e(n)* não processado e do sinal *e(n)* processado

Observa-se que o sinal processado não apresenta as harmônicas em 60Hz, 120Hz, 180Hz e 240Hz, que o sinal não processado apresenta. Assim, pode-se conferir a eficácia do filtro implementado, em código Binário, para a eliminação dessas frequências.

6.2 FILTRO ADAPTATIVO LMS DEDICADO EM CODIFICAÇÃO HÍBRIDA

A figura 35 mostra o sinal de entrada senoidal do filtro adaptativo *x(n)* e a figura 36 mostra o sinal contaminado *d(n)* em codificação Híbrida.

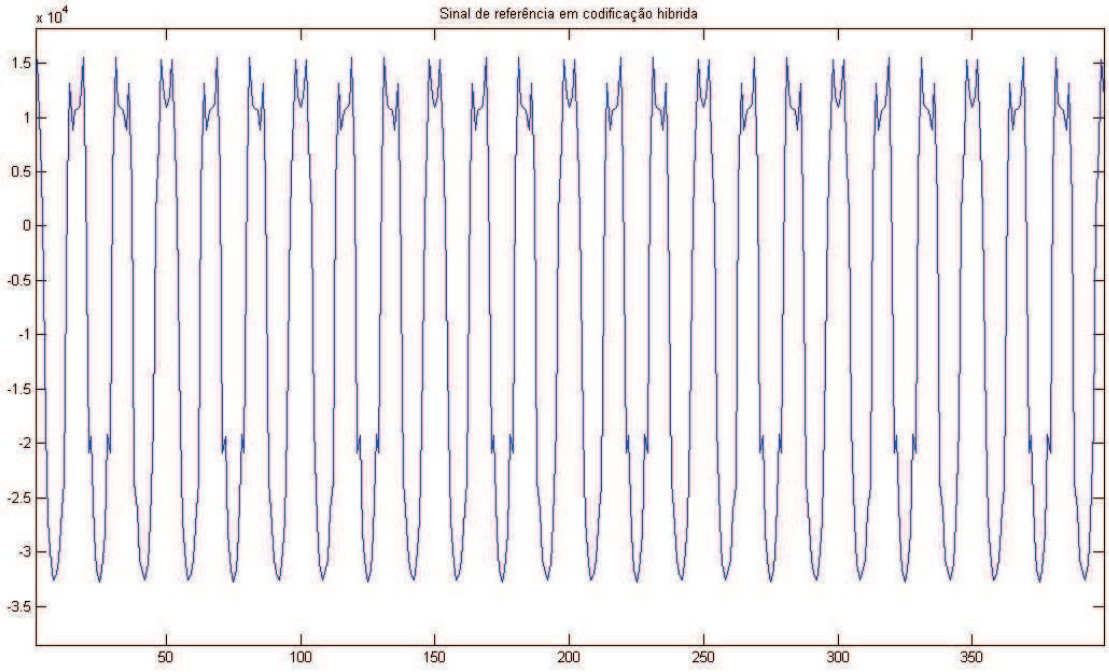

Figura 35: Sinal de interferência *x(n)* (sinusoidal híbrido 1k Hz)

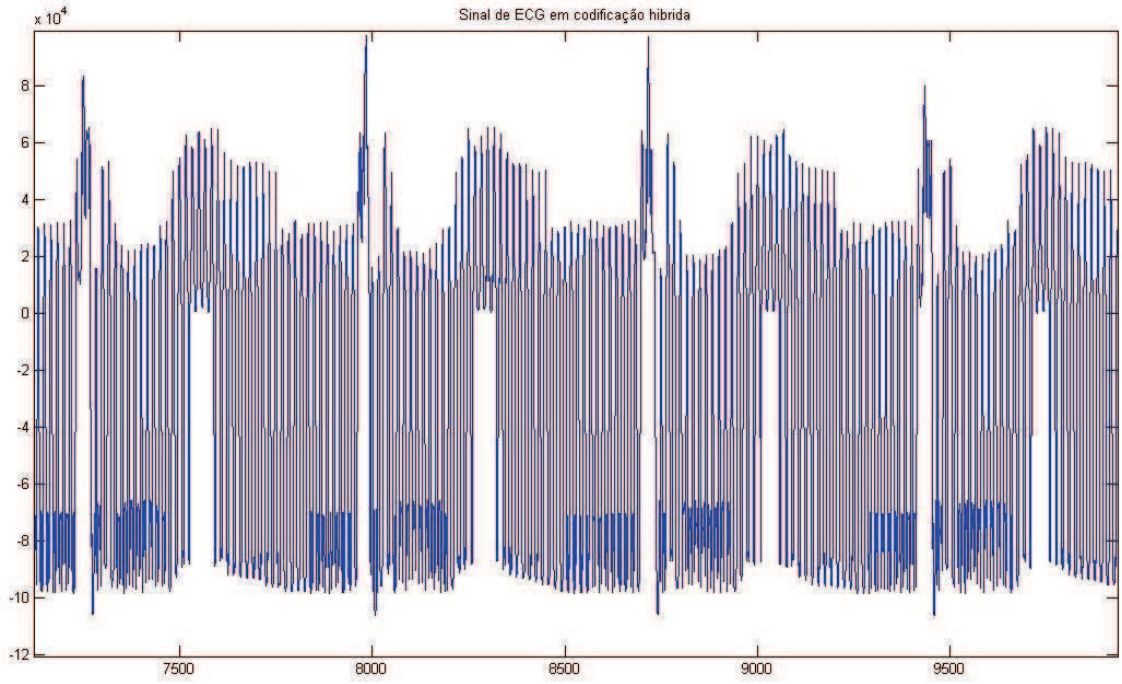

Figura 36: Sinal contaminado *d(n)* híbrido

O gráfico da figura 36 mostra-se mais contaminado quando comparado ao gráfico da figura 30 (sinal em codificação Binária) devido ao fato de que o sinal de ECG está representado com dados em código Híbrido, assim como as interferências do sinal, que também se encontram em codificação Híbrida.

Na tabela 6 são apresentadas, como exemplo ilustrativo, as primeiras amostras do sinal *d(n)* em codificação Binária e em codificação Híbrida. Estas amostras são utilizadas para o cálculo do erro *e(n)* na arquitetura desenvolvida. Pode-se observar pela tabela que os valores do sinal *d*(n) na codificação Híbrida são convertidos para esta codificação, mantendo-se inicialmente o bit mais significativo da palavra binária, e utilizando-se uma lógica XOR a cada grupo de dois bits da palavra binária. Isto ocorre pelo fato de estarmos utilizando, neste trabalho, a codificação Híbrida na base 4, ou seja *m*=2.

Tabela 6: Primeiras amostras dos vetores *d(n)* em codificação binária e híbrida

***d(n)* − codificação Binária**	***d(n)* − codificação Híbrida**
000100100110101101	000100110111111001
000010001110011110	000011001011011011
111111100110110001	101010110111100001
111101100010011110	101001110011011011
111100000100000101	101000000100000101

As amostras do sinal *x*(*n*) (em codificação Híbrida) são processadas pelo filtro adaptativo, e, como resposta desse processo de filtragem, uma amostra *y*(*n*) é gerada. Este valor de *y*(*n*) é então subtraído de uma amostra de *d*(*n*). O resultado dessa operação é o valor do erro estimado *e*(*n*), ou seja, a saída do sistema.

Aplicando a conversão de Binário para Híbrido, obtêm-se todas as amostras do sinal contaminado *d(n)* em codificação Híbrida. A partir da sequência *d(n)* convertida, é realizada a operação de subtração com o sinal *y(n)* de saída, resultando no valor estimado do erro *e(n)*.

A figura 37 mostra as 10000 amostras do sinal *e(n)* resultantes do processo de filtragem. Pode-se observar que o sinal se encontra livre de interferências. A figura 38 mostra as últimas amostras do processo antes e depois da filtragem, este livre de interferência.

Para a obtenção dos gráficos apresentados nas figuras 37 e 38, foi utilizado um leitor de dados desenvolvido em linguagem MatLab, o qual converte os dados em codificação Híbrida para codificação Binária. Essa operação tem, como objetivo, permitir uma melhor visualização do sinal de saída gerado.

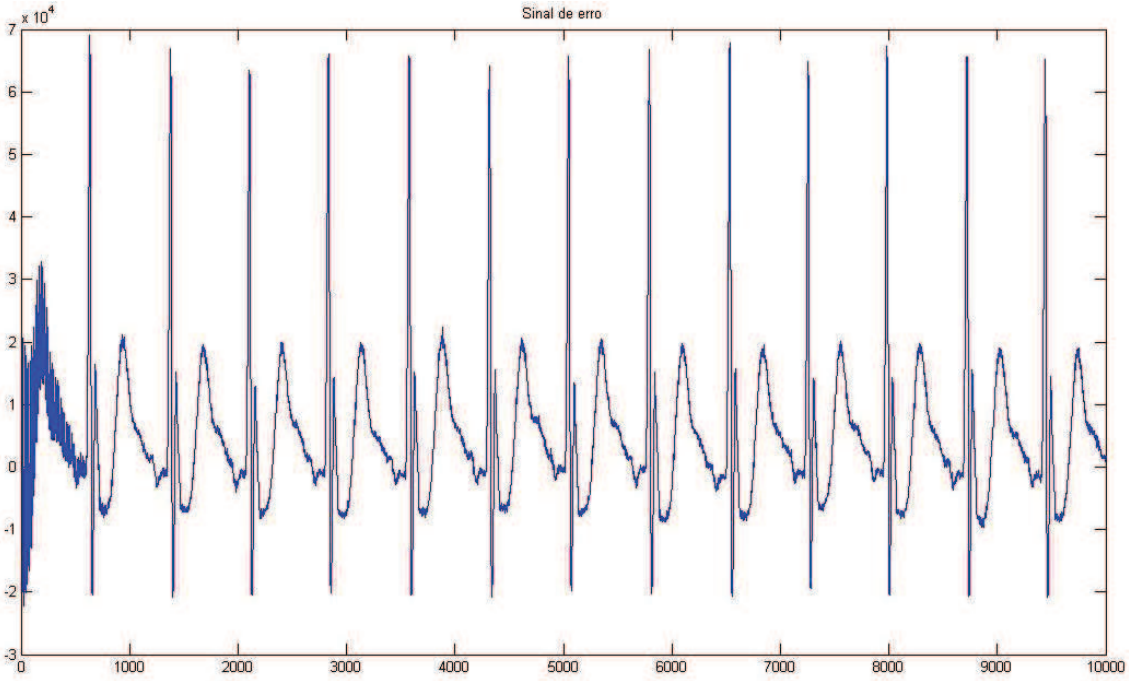

Figura 37: Sinal de saída do filtro adaptativo *e(n)*

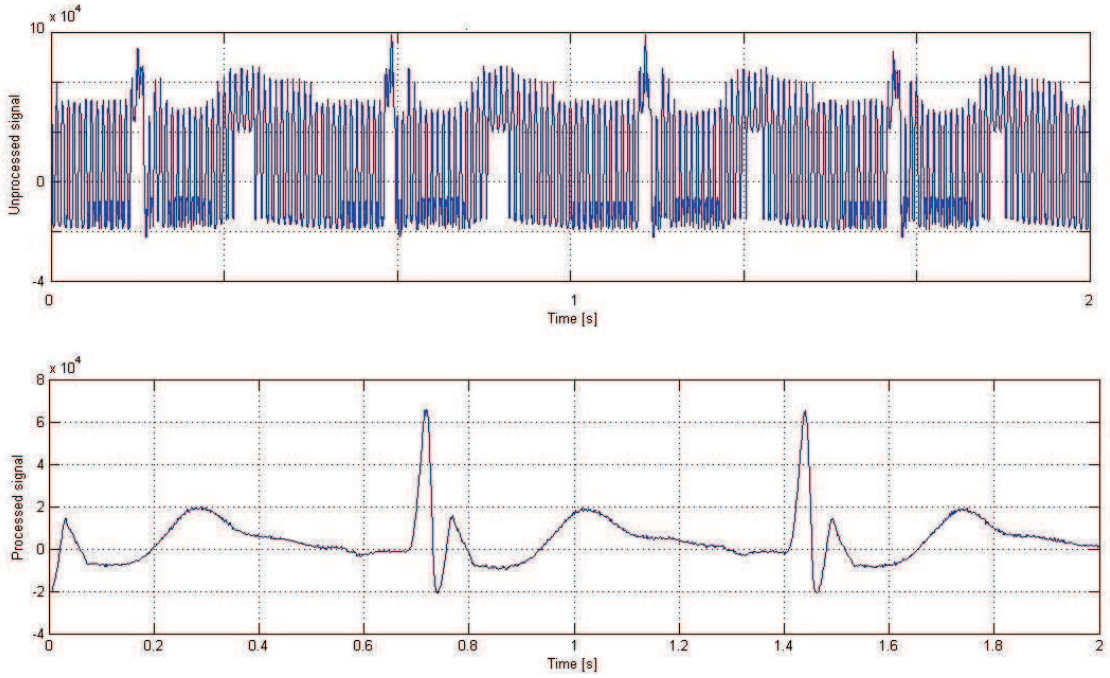

Figura 38: Últimas amostras do sinal antes da filtragem e do sinal após a filtragem *e(n)*

O gráfico de potência, da figura a seguir, demonstra as frequências contidas nos sinais de *e(n)* não processado e *e(n)* processado.

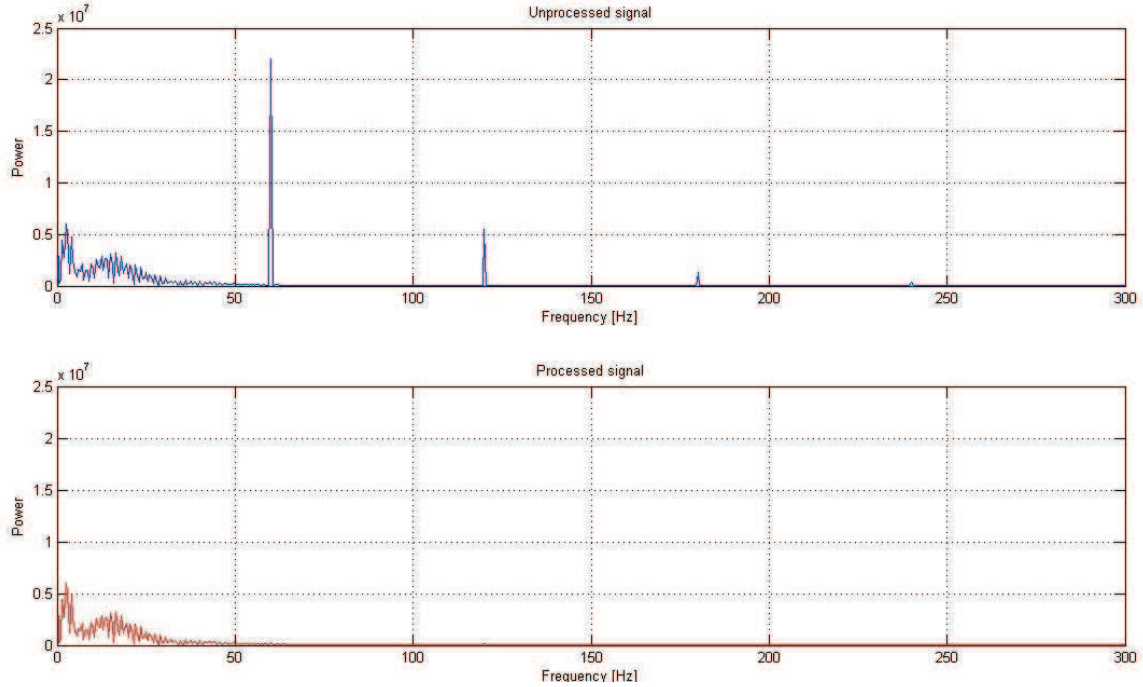

Figura 39: Gráfico de potência do sinal *e(n)* processado e não processado

Através dessa figura 39, pode-se observar que o sinal *e(n)* possui frequências em 60Hz, 120Hz, 180Hz e 240Hz. Essas frequências são filtradas e minimizadas, tornando-se praticamente nulas após o processo de filtragem descrito anteriormente.

Para fins de verificação, foi desenvolvido um programa no ambiente MatLab para o cálculo do consumo de potência desta arquitetura, dado em *dB*, em que se pode observar que as frequências de interferência foram eliminadas do sinal bioelétrico.

Na figura 40, pode ser vista a potência em *dB* do sinal *e(n)* não processado e do sinal *e(n)* processado.

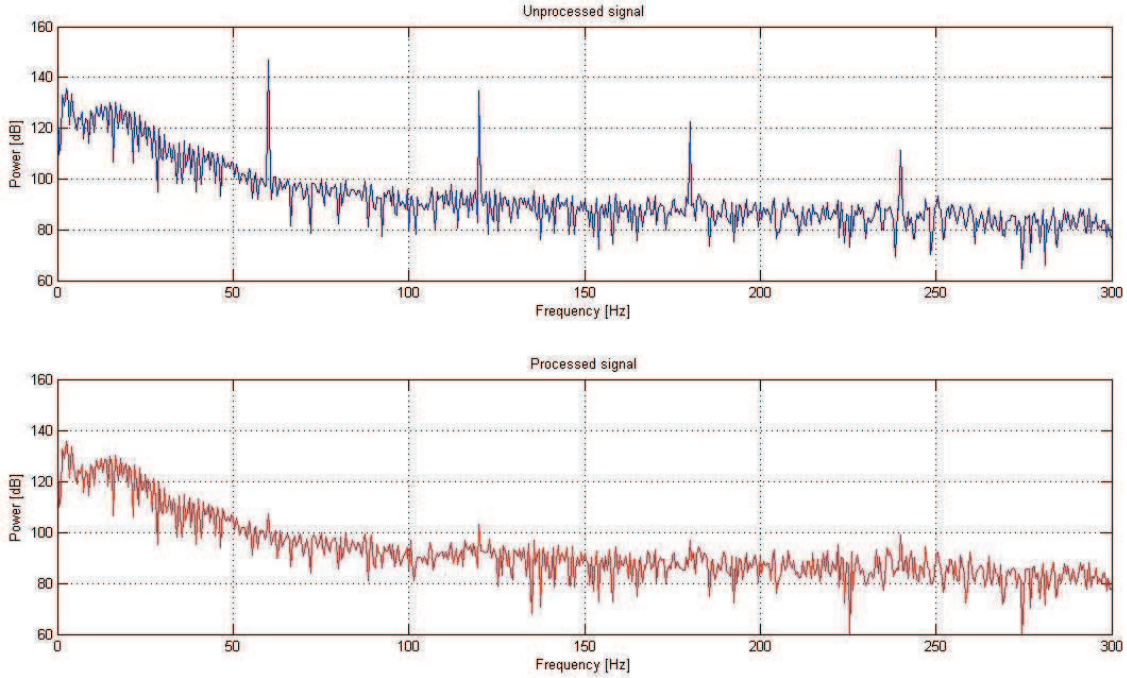

Figura 40: Potência em *dB* do sinal *e(n)* não processado e do sinal *e(n)* processado

Observa-se que o sinal processado não apresenta as harmônicas em 60Hz, 120Hz, 180Hz e 240Hz, que o sinal não processado apresenta. Assim, pode-se conferir a eficiência do filtro implementado para a eliminação das interferências nestas frequências em uma codificação diferenciada, a codificação Híbrida.

6.3 ANÁLISE EM CONSUMO DE POTÊNCIA DAS ARQUITETURAS IMPLEMENTADAS

Os filtros dedicados implementados obtiveram resultados satisfatórios quando implementados em codificação Binária, bem como em codificação Híbrida. Comparando-se as curvas apresentadas nas figuras dos itens 6.1 e 6.2, pode ser observado que as duas implementações apresentaram comportamentos semelhantes no final do processo de adaptação, tornando praticamente nulas as interferências contidas no sinal desejado.

Foram calculados também valores de consumo de potência dessas arquiteturas, que são apresentados em *mW* (mili *Watts*) na tabela 7.

Tabela 7: Resultados de consumo de potência e número de transições obtidos dos filtros adaptativos LMS implementados em códigos Híbrido e Binário

Filtro adaptativo LMS	Potência (*mW*)	N° transições
Binário	0,16	2322511986
Híbrido	0,13	2117244430

Como a frequência de relógio utilizada para essas estruturas é relativamente baixa (7k Hz), os valores de potência apresentados também são reduzidos. Isso ocorre devido ao fato mostrado anteriormente (capítulo 2), que o consumo de potência está diretamente relacionado com a frequência de funcionamento do circuito.

O filtro desenvolvido em código Híbrido apresentou consumo de potência menor, quando comparado ao filtro desenvolvido em código Binário. Mesmo com um pequeno aumento do caminho crítico nas arquiteturas implementadas em codificação Híbrida, devido à aplicação de conversores necessários para realizar a conversão Binário para Híbrido e Híbrido para Binário, estas estruturas apresentaram um valor menor de consumo de potência. Isto ocorre devido à maior correlação entre os dados com a codificação Híbrida, nos barramentos de dados, visto que o código Híbrido representa um compromisso entre a mínima dependência das entradas de dados apresentada pelo código Binário e a característica de baixa atividade de chaveamento apresentada pelo código Gray. Logo, observa-se que, para sistemas onde a capacitância chaveada no barramento de dados é significativa (como é o caso da arquitetura de filtro adaptativo), e onde os dados apresentam um alto grau de correlação (como é o caso de um sinal senoidal em codificação Híbrida), a utilização do código Híbrido pode reduzir a atividade de chaveamento, como pode ser visto na tabela 7, onde o número de transições na codificação Híbrida é menor em relação à codificação Binária. Logo, a menor atividade de transição reflete no menor consumo de potência da arquitetura do filtro adaptativo na codificação Híbrida, em relação à arquitetura na codificação Binária (como pode ser visto nos resultados apresentados na tabela 7).

Ambas as arquiteturas dedicadas de filtro adaptativo utilizando o algoritmo LMS em codificação Binária e Híbrida mostraram-se eficazes para o processo de cancelamento de interferências proposto. A partir do sinal de referência de 60Hz, o

algoritmo estima as harmônicas superiores, utilizando esses resultados para o cancelamento da interferência associada ao sinal de interesse.

Este estudo é uma investigação que abre linhas de pesquisa nessa área arquitetural de filtragem adaptativa com uma codificação diferenciada. Uma estrutura operando totalmente em codificação Híbrida também pode trazer melhorias no desempenho com a redução do caminho crítico, reduzindo o atraso e, consequentemente, o consumo de potência. Para tal, deve-se investigar, como trabalho futuro, a possibilidade da implementação de operadores aritméticos somadores/subtratores operando na codificação Híbrida de forma eficiente (com maior desempenho e menor consumo de potência em relação aos somadores/subtratores binários).

6.4 RESUMO

Este capítulo apresentou as curvas de resposta obtidas para a validação do funcionamento das arquiteturas dos filtros adaptativos implementados neste trabalho, em codificação Binária e em codificação Híbrida. Pôde-se verificar a eficiência dos filtros para o cancelamento de interferências em ambas as codificações, mostrando-se possível a implementação de um filtro adaptativo com um menor valor de consumo de potência em codificação Híbrida.

7 CONCLUSÕES

O trabalho apresentou a implementação de arquiteturas de *hardware* dedicado para um algoritmo de filtragem adaptativa em codificação Híbrida. Para o desenvolvimento do filtro, foi escolhido o filtro adaptativo LMS por sua eficácia, robustez e simplicidade de implementação. Também existem outros algoritmos, da mesma família, que são derivações do algoritmo LMS, por exemplo, NLMS, Leaky LMS, entre outros. Isso significa que, ao implementar uma arquitetura eficiente do algoritmo adaptativo LMS, se abrem novas perspectivas para implementações destes outros algoritmos citados, com apenas algumas pequenas modificações.

Para o funcionamento da arquitetura do filtro na codificação Híbrida, foi necessário o desenvolvimento de circuitos multiplicadores *array* operando nesta codificação. O trabalho apresenta implementações de multiplicadores *array* otimizados, em linguagem de descrição de *hardware*, apresentados no meio científico por Costa (2002) e Pieper (2008), os quais otimizam os blocos dedicados de multiplicação. Assim, permitem operações de multiplicação na base 2^m, o que possibilita a redução de número de linhas de produtos parciais da estrutura. Neste trabalho utiliza-se m=2 e, para um caso específico, m=3.

Para o caso m=3, foram desenvolvidas novas arquiteturas de multiplicadores com estruturas irregulares, ou seja, estruturas que realizam multiplicações de números ímpares de maneira otimizada.

Esses multiplicadores implementados são apresentados, além da codificação Binária, em uma codificação diferenciada, a codificação Híbrida. Esta codificação representa um compromisso entre a mínima dependência das entradas de dados apresentada pelo código Binário e a característica de baixa atividade de chaveamento apresentada pelo código Gray. Assim, o código Híbrido é uma situação intermediária entre os códigos Binário e Gray em termos de número de transições.

Os resultados mostraram eficiência na redução do consumo de potência dos multiplicadores Híbridos apenas para multiplicadores com estruturas regulares. Para os multiplicadores *array* otimizados de 18 e 36 bits, a operação em codificação Híbrida mostrou melhor desempenho. No caso de 23 bits, o multiplicador atuando em codificação Binária obteve melhores resultados.

Os multiplicadores desenvolvidos foram aplicados nas arquiteturas dos filtros adaptativos, o que possibilitou que os filtros adaptativos LMS fossem implementados nas codificações Binária e Híbrida, mostrando-se eficientes para ambas as estruturas no processo de cancelamento de interferências proposto. A partir do sinal de referência de 60Hz, o algoritmo estima as harmônicas superiores, utilizando esses resultados para o cancelamento da interferência associada ao sinal de interesse.

O filtro adaptativo utilizando o algoritmo LMS em codificação Híbrida (proposta deste trabalho) apresentou um consumo de potência inferior (18,75%) ao filtro em codificação Binária, devido à correlação entre os dados nos barramentos de dados da estrutura. Este é um resultado significativo, pois se torna possível uma implementação de uma arquitetura de filtro adaptativo operando em codificação Híbrida, através da aplicação de multiplicadores otimizados e dados de entrada nesta mesma codificação Híbrida, com um reduzido valor de consumo de potência quando comparado à arquitetura de filtro adaptativo operando em codificação Binária.

7.1 TRABALHOS FUTUROS

Como principal trabalho futuro, pretende-se realizar testes do consumo de potência, destas arquiteturas apresentadas neste trabalho, em um ambiente mais adequado para esta finalidade. Pretende-se obter novos valores, para estas arquiteturas, na ferramenta comercial da *Cadence* (RTL *Compiler*) que, além de ser mais precisa, também é mais bem aceita no meio acadêmico.

Para isso, são necessários, além dos arquivos implementados e disponíveis no cd-rom, anexo neste trabalho, vetores de entradas *testbanch*, que também já foram desenvolvidos. Tinha-se como ideia inicial apresentar esses resultados de imediato, mas não foi possível esse desenvolvimento para o presente trabalho[2].

Outro objetivo, como trabalho futuro, é o projeto de uma arquitetura de filtro adaptativo utilizando o algoritmo LMS, proposto neste trabalho, totalmente Híbrida. Para tal, torna-se necessário o desenvolvimento de circuitos

[2] Não foi possível dar continuidade ao trabalho com a ferramenta da *Cadence* por razões completamente alheias à vontade do pesquisador.

somadores/subtradores que operem na codificação Híbrida para aplicar na estrutura apresentada. A aplicação da técnica de *pipelining* nas arquiteturas de multiplicadores apresentada neste trabalho é outra meta a ser buscada, bem como a aplicação na estrutura interna do filtro adaptativo. A forma *pipelined* é implementada com a inserção de registradores ao longo da estrutura para reduzir a atividade de chaveamento desnecessária, aumentando assim o desempenho dos circuitos.

REFERÊNCIAS

ALTERA. ***Quartus II Version 6.0 Handbook*.** [S.1.]: Altera Inc., 2006.

BOOTH, A. ***A Signed Binary Multiplication Technique***. J. Mech. And Applied Mathematics, (4): 236-240, 1951.

CALLAWAY, T.; SWARTZLANDER, E. ***Optimizing Multipliers for WSI.*** In Annual IEEE International Conference on Wafer Scale Integration. 1993.

CHANDRAKASAN, A.; BRODERSEN, R. ***Low Power Digital CMOS Design****.* Kluwer Academic Publishers, 1995.

COSTA, E. A. C. ***Operadores Aritméticos de Baixo Consumo para Arquiteturas de Circuitos DSP.*** 2002. Tese (Doutorado em Ciência da Computação) — Programa de Pós-Graduação em Computação - UFRGS, Porto Alegre, RS.

COSTA, E.; MONTEIRO, J.; BAMPI, S. ***A New Architecture for Signed Radix-2m Pure Array Multiplier***. IEEE International Conference on Circuit Design, ICCD, September 2002. 2002a.

COSTA E.; MONTEIRO J.; BAMPI, S. ***Power Efficient Arithmetic Operand Encoding***. In Proceedings Symposium on Integrated Circuits and Systems Design, 2001.

COSTA, M. H. ***Implementação de um sistema digital para controle ativo de Ruído***. Atividade de laboratório. UFSC. Florianópolis, 1997.

COSTA, M. H.; TAVARES, M. C. ***Algoritmo Adaptativo para o Cancelamento da Interferência da Rede de Alimentação em Sinais Bioelétricos***. IIICLAEB'2004. 2004.

FOSTER, D.; ARDIZZONE, N.; FONSECA, M.; ALMEIDA, S.; COSTA, E. ***Arquiteturas Dedicadas de Filtro Adaptativo LMS de Baixa Potência***. XXI Congresso de Iniciação Científica e Tecnológica em Engenharia – CRICTE, 2006.

GALLAGHER, W.; SWARTZLANDER, E. ***High Radix Booth Multipliers Using Reduced Area Adder Trees****.* In Twenty-Eighth Asilomar Conference on Signals, Systems and Computers, volume 1, pages 545-549, 1994.

GAN, Woon S.; KUO, Sen M. ***Embedded Signal Processing with the Micro Signal Architecture.*** John Wiley & Sons, 2007.

GOWAN, M. K.; BIRO, L. L.; JACKSON, D. B. Power ***Considerations in the Desing of the Alpha 21264 Microprocessor.*** Design Automation Conference, p. 726-731, Jun. 1998.

HAYKIN, Simon. ***Adaptive Filter Theory****.* 3 ed, Prentice-Hall, 1996.

ITTURRIET, F. ***Implementação de um Hardware ponto fixo para o algoritmo LMS***. 2008. Trabalho final de graduação – Engenharia Elétrica/Eletrônica, UCPel, Pelotas/RS.

KIM, N. S.; BLAAUW, D.; MUDGE, T. ***Leakage Power Optimization Techniques for Ultra Deep Sub-Micron Multi-Level Caches***, International Conference in Computer Aided Design. ICCAD-2003. p.627-632, Nov 2003.

KUO, Sen M.; LEE, Bob H. ***Real-Time Digital Signal Processing: Implementations, Applications, and Experiments with the TMS320C55X.*** John Wiley & Sons, 2001.

MANOLAKIS, D. G. ***Statistical and Adaptive Signal Processing***. Artech House, 2005.

MEHENDALE, M.; SHERLEKAR, S.; VENKATESH, G. ***Techniques for Low Power Realization of FIR Filters****.* In Design Automation Conference, DAC, 3(3), pages 404-416, September, 1995.

MEHENDALE, M.; SHERLEKAR, S.; VENKATESH, G. ***Low-Power Realization of FIR Filters on Programmable DSP's***. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 6(4):546-553, December 1998.

NAKAMURA, S. ***Algorithm for Interative Array Multiplication****.* IEEE Transactions on Computers, Washington, v.35, n.8, p. 713-719, Out. 1986.

PARHAMI, B. ***Computer Arithmetic – Algorithms and Hardware Designs***. Oxford University Press. 2000.

PARK, I-C.; KANG, H-J., ***Digital Filter Synthesis Based on Minimal Signed Digit Representation****.* In Design Automation Conference, pag. 468-473, 2001

PIEPER, L. Z. ***Circuitos Multiplicadores Array de Baixo Consumo de Potência Aplicados a Filtros Adaptativos.*** Tese — Mestrado em Ciência da Computação - UCPel, Pelotas, RS. 2008.

PIEPER, L.; COSTA, E.; BAMPI, S.; MONTEIRO, J., ***Efficient Dedicated Multiplication Blocks for 2´s Complement Radix-16 and Radix-256 Array Multipliers***, Journal of Computers. Outubro, 2010.

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

WESTE, N.; ESHRAGHIAN, K. ***Principles of CMOS VLSI Desing***. 2nd Ed. Boston: Addison-Wesley, 1994.

WIDROW, B., GLOVER-JR., J.R., MCCOOL, J.M., et al., ***"Adaptive Noise Cancelling: Principles and Applications",*** Proceedings of the IEEE. 63(12):1692-1716, 1975.

WILLIAMS, T. W. DENNARD, R. H.; KAPUR, R. MERCER, M. R.; MALY, M. ***Iddq test: sensitivity analysis of scaling***, Test Conference, Washington, DC – USA, p. 786 – 792, Oct 1996.

ANEXO