TÉCNICO LISBOA

Auriculares com Controlo Ativo de Ruído Sobre Amostrado

André Duarte Barbosa

Dissertação para obtenção do Grau de Mestre em

Engenharia Eletrotécnica e de Computadores

Orientador: Prof. Paulo Alexandre Crisóstomo Lopes

Júri

Presidente: Profª. Teresa Maria Sá Ferreira Vazão Vasques

Orientador: Prof. Paulo Alexandre Crisóstomo Lopes

Vogal: Prof. João Pedro Castilho Pereira Santos Gomes

Novembro 2021

Declaração:

Declaro que o presente documento é um trabalho original da minha autoria e que cumpre todos os requisitos do Código de Conduta e Boas Práticas da Universidade de Lisboa.

Declaration:

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

RESUMO

Um protótipo de auscultadores com Controlo Ativo de Ruído (CAR) com sobre amostragem foi desenvolvido para este projeto. Os sistemas CAR baseiam-se no principio fisíco de que os sinais sonoros no meio ambiente somam-se, e portanto, se existir um ruído e for aplicado o mesmo sinal com oposição de fase e amplitudes equivalentes à da fonte de ruído, resulta num cancelamento. Para a elaboração do protótipo, foram utilizados os auscultadores ATH-ANC1 QuietPoint, sendo que o seu controlador foi removido e outro foi desenvolvido baseado numa ZedBoard que contém FPGA. Adicionalmente, foram necessários conversores de alto desempenho para conseguir suportar a sobre amostragem. Para a utilização de sinais sobre amostrados, técnicas de processamento de sinal foram utilizadas, nomeadamente, a utilização de conversores Sigma-Delta. Assim, foi possível implementar o sistema com uma elevada frequência de amostragem, mas utilizando menos bits por amostra, reduzindo assim a complexidade computacional do sistema. Foram implementados algoritmos FxLMS com *feedforward* e *feedback* com a modelação do caminho secundário realizada em modo *offline.* Para averiguar se a estimação do caminho secudário estava a ser efetuada corretamente, foram realizados vários testes. Por fim, foi examinado o desempenho do sistema FxLMS com *feedback*, para vários tipos de ruído. Verificou-se que para ruídos sinusoidais o sistema conseguia efetuar um cancelamento total, restando apenas ruído ambiente e ruído de altas frequências que não são audíveis pelo ouvido humano. No entanto, para ruído branco, o sistema apenas consegue diminuir cerca de 3% da potência total do sinal.

Palavras-chave: Controlo de Ruído Ativo, Conversores Sigma-Delta, FxLMS, ZedBoard, Controlador *feedforward*, Controlador *Feedback*.

ABSTRACT

A prototype Active Noise Control (ANC) headphone with oversampling was developed in this project. CAR systems are based on the physical principle that sound signals in the environment add up, and therefore if noise exists and the same signal is applied with phase opposition and amplitudes equivalent to that of the noise source, it results in cancellation. To develop the prototype, the ATH-ANC1 QuietPoint headset was used, with its controller removed and another one developed based on a ZedBoard containing FPGA. Additionally, high-performance converters were needed to be able to support oversampling. In order to use oversampled signals, signal processing techniques were used, namely, the use of Sigma-Delta converters. Thus, it was possible to implement the system with a high sampling frequency, yet utilizing fewer bits per sample, hence reducing the computational complexity of the system. FxLMS algorithms with feedforward and feedback were implemented with the secondary path modeling performed in offline mode. To find out if the secondary path estimation was being performed correctly, several tests were performed. Lastly, the performance of the FxLMS system with feedback was examined for various types of noise. It was found that for sinusoidal noise the system was able to perform a full cancellation, leaving only ambient noise and high frequency noise that is not audible to the human ear. However, for white noise, the system can only reduce about 3% of the total signal power.

Keywords: Active Noise Control, Sigma-Delta Converters, FxLMS, ZedBoard, Feedforward Controller, Feedback Controller.

ÍNDICE

LISTA DE FIGURAS

LISTA DE ACRÓNIMOS

AA Anti-Aliasing

ADC Analog to Digital Converter

ASIC Application Specific Integrated Circuit

CAR Controlador Ativo de Ruído

DAC Digital to Analog Converter

DSP Digital Signal Processor

FIR Finite Impulse Response

FPGA Field-Programmable Gate Array

FxLMS Filtered-x Least Mean Square

IIR Infinite Impulse Response

LMS Least Mean Square

LP Low-Pass

LWL Long Word Length

MSE Mean Squared Error

MSPS Megasamples Per Second

NTF Noise Transfer Function

RC Resistor-Capacitor

SDM Sigma-Delta Modulation

SWL Short Word Length

1. INTRODUÇÃO

1.1. VISÃO GERAL

A indústria tecnológica tem crescido exponencialmente nos últimos anos, especialmente no que diz respeito a avanços em processamento digital de sinais [1]. Assim, muitas empresas estão a apostar no desenvolvimento de produtos equipados com redução de ruído, que oferecem uma solução eficaz e de baixo custo para situações em que se verifica um elevado nível de ruído envolvente e em que, consequentemente, é necessária uma melhoria na qualidade do áudio [2]. A técnica utilizada consiste na utilização de uma fonte de cancelamento, que gera um sinal com oposição de fase e amplitudes equivalentes à da fonte de ruído [3], resultando assim num cancelamento (Figura 1). Todavia, é importante reconhecer as limitações subjacentes a esta prática, nomeadamente o facto de que apenas é possível um cancelamento de ruído total num determinado ponto do espaço, e que a sua redução é mais eficiente em baixas frequências, dado que neste caso os comprimentos de onda são grandes [1].

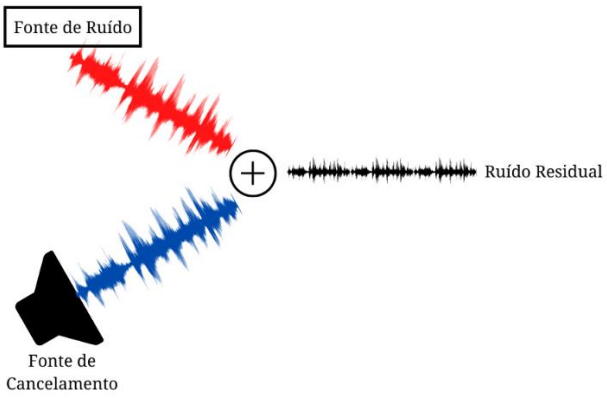


Figura 1 - Representação visual da Redução Ativa do Ruído

O método de cancelamento de ruído ativo já foi aplicado na resolução de uma grande variedade de problemas em várias áreas, como na manufatura e operações industriais [3]. Para este projeto, a técnica de controlo ativo de ruído (CAR) será aplicada a um produto correntemente utilizado no quotidiano dos consumidores, nomeadamente em auscultadores de ouvido. Um microfone de referência será utilizado no interior dos auscultadores para medir o sinal de ruído, sendo este invertido e inserido na fonte de cancelamento, que neste caso será o próprio auricular, de forma a criar uma onda antirruído, como representado na Figura 2.

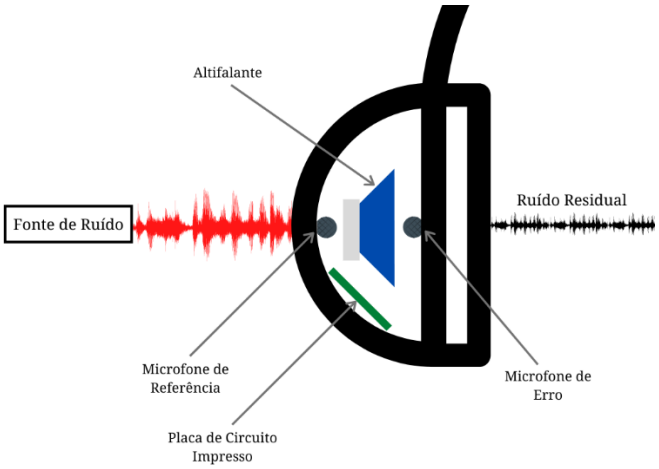


Figura 2 - Estrutura de um Auscultador com CAR

No entanto, é importante considerar que existe uma margem de atraso mínima tanto entre o sinal de referência e o microfone de erro, como entre o som propagado pelo altifalante e o microfone de erro, o que exige que os cálculos relativos ao sinal antirruído tenham de ser executados nesse breve intervalo. Assim, para conseguir uma redução significativa de ruído, é necessário implementar um CAR com o menor atraso possível. Na maioria deste tipo de aplicações, esse atraso é significativo, devido em grande parte à taxa de amostragem tipicamente baixa, aos filtros de *antialising* (AA), e aos filtros de reconstrução (RC) dos *analog to digital converters* (ADC) e dos *digital to analog converters* (DAC). Uma possível contribuição para reduzir o atraso verificado é aumentar a frequência da amostragem, à custa de um aumento significativo da complexidade computacional. Outro recurso incide na utilização de sigma-delta ADCs e DACs, que funcionam em altas frequências de amostragem, porém o seu uso em CAR é limitado devido ao atraso dos filtros AA e RC.

1.2. Motivação e Objetivos

Neste relatório, um estudo inicial sobre controlo ativo de ruído sobre amostrado é apresentado. Este foi essencialmente motivado pelo facto de, apesar de ser uma área já explorada, nunca ter sido utilizado o método descrito em seguida. Para além disso, este projeto surge da constante necessidade de atualização e desenvolvimento da área em questão, bem como da adaptação às necessidades do consumidor, uma vez que, sendo este projeto bem-sucedido, será possível uma redução mais eficaz daquela já existente no mercado, sem que se verifique uma variação de custo final do produto muito elevada.

O principal objetivo desta dissertação é desenvolver um protótipo de auscultadores de ouvido de redução ativa de ruído sobre amostrado. A implementação do controlador será realizada numa placa *ZedBoard* (FPGA) em conjunto com amplificadores, ADCs, e DACs de alta performance.

Neste projeto, ao contrário do habitual para este tipo de controlador, os filtros AA e RC serão removidos do conversores sigma-delta (SDM), utilizando assim os sinais sobre amostrados diretamente no sistema de CAR. Esta proposta permite implementar o sistema de CAR com uma

elevada frequência de amostragem, mas usando sinais formados por palavras de menor dimensão (menos *bits*) por amostra. No protótipo, os conversores SDM serão emulados utilizando ADCs *flash* (convertem o sinal num único ciclo de relógio) de alta frequência ou ADCs *pipeline* e processados na placa FPGA.

Na secção seguinte será explicado de uma forma geral o modo de funcionamento do controlador através de filtragem adaptativa e os algoritmos utilizados, posteriormente será apresentada uma visão geral da tecnologia CAR e suas aplicações, e de seguida será descrita a técnica aplicada ao processamento de sinal com sobre amostragem. Por fim, é efetuada uma descrição do hardware utilizado no projeto, bem como a implementação da filtragem adaptativa e do processamento de sinal em FPGA.

2. FILTRAGEM ADAPTATIVA

A filtragem consiste, de uma forma geral, num processo de remoção de ruído de forma a revelar ou aprimorar informações sobre alguma quantidade de interesse [4]. Ao contrário de uma filtragem comum (não adaptativa), a filtragem adaptativa não consiste num sistema definido com antecedência, mas em vez disso utiliza as informações ao seu redor para estimar o valor dos parâmetros [5], os quais controlam a função de transferência do filtro linear, presente no sistema de filtragem adaptativa. O modo de operação deste tipo de filtragem envolve essencialmente dois processos, nomeadamente o processo de filtragem, em que é produzido um resultado em resposta a uma sequência de dados de entrada, e o processo adaptativo, que fornece um mecanismo de controlo para os parâmetros utilizados no processo antecedente [6].

Frequentemente, as propriedades do sinal não estão disponíveis ou são variáveis, pelo que os parâmetros estatísticos variam com o tempo, tornando difícil atingir a solução ótima [5]. Assim, o filtro utilizado neste processo é autodesenvolvido, na medida em que contém um algoritmo que utiliza feedback, com capacidade de obter um desempenho satisfatório quando não existe um conhecimento completo das características do sinal e do ruído [6]. Isto é conseguido através de um ajuste dos parâmetros do filtro, sendo estes atualizados a cada iteração, e por isso dependentes dos dados de entrada [6].

A estrutura do filtro adaptativo (Figura 3) é composta por um filtro linear digital, cujos coeficientes irão variar com o algoritmo adaptativo, e por sinais de entrada, de saída, de referência, e de erro. O processo adaptativo é controlado pelo sinal de erro, que representa uma medida da adaptação dos parâmetros do filtro, indicando assim o nível de conformidade entre a saída e o sinal de referência, que se trata da resposta desejada do filtro adaptativo [5]. O objetivo principal da filtragem adaptativa é então otimizar o sinal de erro, ou seja, minimizar a diferença entre o sinal desejado e o sinal de saída do filtro, através de uma adaptação constante dos parâmetros presentes no filtro [5].

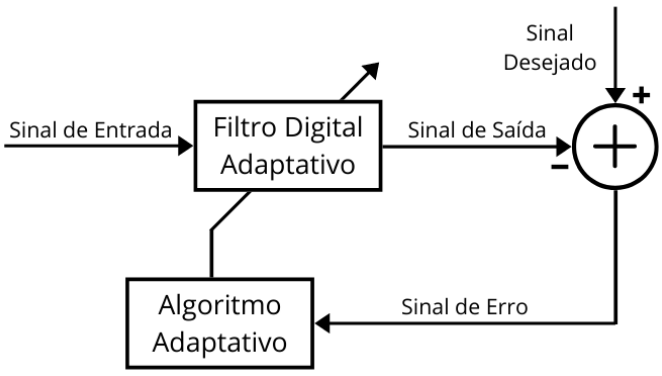


Figura 3 - Estrutura do Filtro Adaptativo

2.1. FILTROS DIGITAIS LINEARES

Com o avanço das técnicas digitais nos últimos anos, os filtros analógicos estão a ser substituídos por filtros digitais, por serem mais baratos, fáceis de construir, e flexíveis [7]. As respostas de frequência dos filtros digitais dependem do valor dos seus coeficientes [8], que modificam ao longo do tempo, provocando uma alteração na resposta em frequência, para que esta vá de encontro ao pretendido.

O filtro linear pode ser realizado usando resposta de impulso finita (FIR) ou resposta de impulso infinita (IIR), no entanto as estruturas de filtros adaptativos mais utilizadas são os FIR, devido à sua estabilidade incondicional e à análise relativamente simples das propriedades desses filtros [5]. Ambos irão ser aprofundados de seguida.

Relativamente aos filtros IIR, a saída do filtro depende das entradas anteriores e das saídas anteriores, apresentando assim a seguinte equação [[8], eq, (1)]:

$$y(n) = \sum_{k=1}^{N-1} a_k * y(n-k) + \sum_{k=0}^{M-1} b_k * x(n-k) \quad (1)$$

O sinal *y(n)* corresponde à saída do filtro no instante de tempo discreto *n*, *x(n - k)* à entrada do filtro atrasada por *k* amostras, *N* é o número de coeficientes de *feedback* (a_k), e *M* é o número de coeficientes de *feedforward* (b_k).

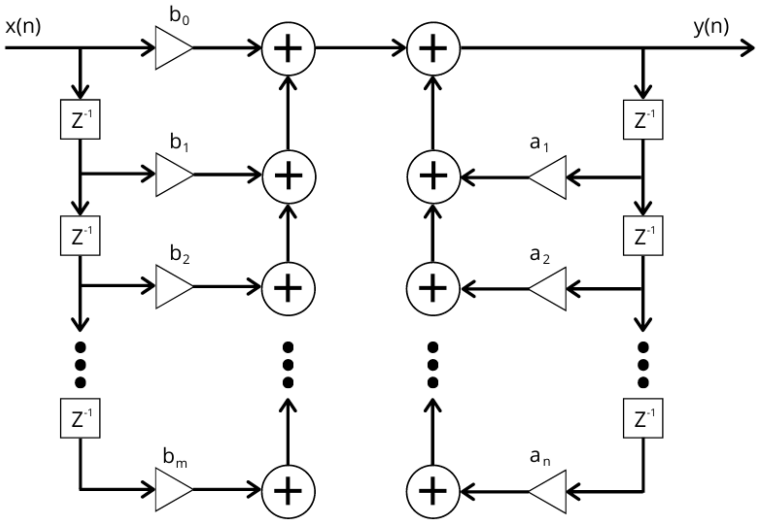


Figura 4 - Estrutura Filtro IIR

Este filtro (Figura 4) normalmente requer um número menor de multiplicações em comparação com os filtros FIR e também podem ser projetados para ter uma resposta de frequência, que é uma versão discreta da resposta de frequência de um filtro analógico [8]. Uma grande desvantagem consiste no facto de que este filtro pode não ser estável se não for desenhado corretamente [8], uma vez que a função de transferência deste filtro apresenta polos, e se estes se encontrarem no semiplano complexo direito, o sistema é instável. Este tipo de filtro é também muito sensível a erros

de quantização de coeficiente que ocorrem devido ao uso de um número finito de bits, para representar os coeficientes de filtro [8].

Uma das maneiras de superar a falta de estabilidade de um filtro digital IIR é projetar um filtro com zeros apenas, sendo que este apresenta uma estrutura não recursiva. Como a memória de tais filtros é limitada, a sua resposta ao impulso é igual a zero fora de algum intervalo de tempo limitado e, por isso, eles são indicados como os filtros FIR [5].

Ao contrário dos filtros IIR, os filtros FIR apenas dependem das entradas anteriores, e apresentam a seguinte equação [[8], eq.(2)]:

$$y(n) = \sum_{k=0}^{M-1} a_k * x(n-k) \tag{2}$$

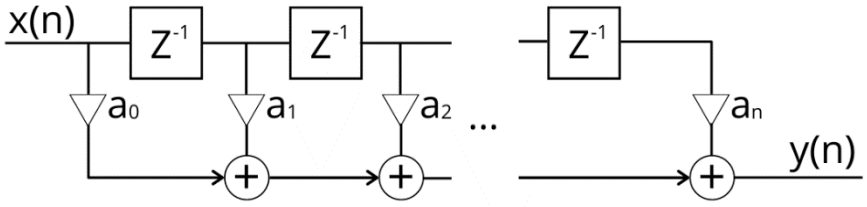


Figura 5 - Forma Direta do Filtro FIR

Uma vez que este tipo de filtro apenas depende das entradas anteriores, a sua função de transferência somente apresenta zeros, o que indica que o filtro é sempre estável. Os filtros FIR também têm uma baixa sensibilidade no que diz respeito aos erros de quantização dos coeficientes. Esta é uma propriedade importante que se deve ter em consideração ao implementar um filtro num circuito integrado [8].

De todas as estruturas dos filtros FIR existentes, a apresentada na Figura 5 é a mais intuitiva. No entanto, existe outra estrutura dos filtros FIR que é a forma transposta apresentada na Figura 6. Em teoria, ambas as estruturas irão apresentar resultados equivalentes, mas quando calculados com precisão finita, pode haver diferenças entre as diferentes implementações.

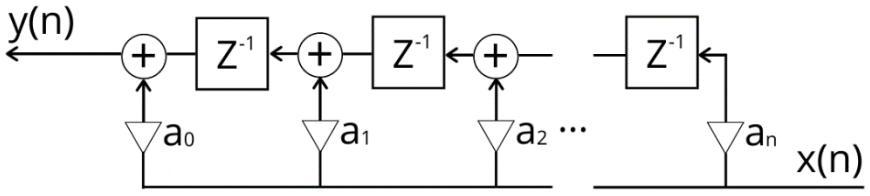


Figura 6 - Forma Transposta do Filtro FIR

Uma grande vantagem com a utilização da forma transposta é que basta uma multiplicação e adição para conseguir obter a saída, podendo-se implementar em paralelo os cálculos dos outros sinais de entradas dos restantes elementos de memória, enquanto que na forma direta tem de se somar *N* os sinais para obter uma saída.

2.2. ALGORITMOS ADAPTATIVOS

O algoritmo adaptativo pode assumir diversas formas, e é geralmente derivado como uma forma de otimização, responsável por minimizar o critério de erro, útil para a tarefa em questão [9]. O objetivo do algoritmo adaptativo é definir os coeficientes do filtro adaptativo, de forma a que a saída minimize uma função objetivo, e assim a saída tender para o sinal desejado [10].

Não existe uma solução única para o problema da filtragem adaptativa. Em vez disso, existem muitas ferramentas representadas por uma variedade de algoritmos iterativos, cada um dos quais oferece características próprias desejáveis. No entanto, existem duas abordagens principais para o desenvolvimento de algoritmos de filtro adaptativo, a Abordagem do Gradiente Estocástico e a Estimativa dos Mínimos Quadrados [6].

O procedimento de mínimos médios quadrados (LMS) utiliza um método de gradiente Estocástico, no qual os coeficientes do filtro FIR, são atualizados com base no sinal de erro instantâneo [5]. Tal como na Figura 3, o sinal de erro no tempo discreto *n* é representado por [[11], eq. (3)(4)]:

$$e(n) = d(n) - y(n) \tag{3}$$

$$e(n) = d(n) - \boldsymbol{X}^T(n)\boldsymbol{W} \ \because y(n) = \ \boldsymbol{X}^T(n)\boldsymbol{W} \tag{4}$$

O parâmetro $d(n)$ corresponde ao sinal desejado, $y(n)$ é o sinal de saída do filtro digital, $\boldsymbol{X}^T(n)$ é o vetor transposto que contém os sinais de entrada, e $\boldsymbol{W}$ é o vetor que contém os coeficientes do filtro, de dimensão N.

Assim, a função de custo do LMS é representada da seguinte forma [[11], eq.(5)(6)]:

$$C(n) = E[e(n)^2] \tag{5}$$

$$C(n) = E[d(n)] + \boldsymbol{W}^T E[\boldsymbol{X}(n)\boldsymbol{X}^T(n)]\boldsymbol{W} - 2E[d(n)\boldsymbol{X}^T(n)]\boldsymbol{W} \tag{6}$$

Por outras palavras, a função de custo é igual ao valor esperado do erro ao quadrado, sendo esta função de custo chamada erro quadrático médio (MSE). Fica claro, a partir desta expressão, que o erro quadrático médio é uma função quadrática dos coeficientes do filtro quando os componentes de entrada e o sinal de referência são variáveis estocásticas estacionárias [11]. Assim, a representação gráfica da função de custo irá ter a forma idêntica a uma “tijela”, que será apelidada de superfície de desempenho. O ponto mínimo da superfície de desempenho terá os coeficientes que minimizam esta função, não havendo mínimos locais, mas sim apenas um mínimo global.

Muitos processos adaptativos úteis que fazem com que os coeficientes dos filtros digitais tendam para os mínimos da superfície de desempenho, fazem-no por métodos de gradiente [11]. Estes tiram proveito de uma propriedade matemática, em que quando o gradiente de uma função convexa é zero, significa que esse ponto é um mínimo global. Assim, para se encontrar os coeficientes que minimizam

a função de custo, terá de se calcular o gradiente dessa função e igualar a zero, como é representado nas equações seguintes [[11], eq. (7)(8)(9)(10)]:

$$\boldsymbol{R} = E[\boldsymbol{X}(n)\boldsymbol{X^T}(n)] \ \wedge \boldsymbol{P} = E[d(n)\boldsymbol{X}(n)] \ \therefore C(n) = E[d^2(n)] + \boldsymbol{W}^T\boldsymbol{R}\boldsymbol{W} - 2\boldsymbol{P}^T\boldsymbol{W} \tag{7}$$

$$\nabla\, C(n) = \ 2\boldsymbol{R}\boldsymbol{W} - 2\boldsymbol{P} \tag{8}$$

$$\nabla C(n) = 0 \Leftrightarrow 2\boldsymbol{R}\boldsymbol{W}^* - 2\boldsymbol{P} = \boldsymbol{0} \tag{9}$$

$$\boldsymbol{W}^* = \boldsymbol{R}^{-1}\boldsymbol{P} \tag{10}$$

A variável $\boldsymbol{R}$ diz respeito à matriz de auto correlação do sinal de entrada, $\boldsymbol{P}$ ao vetor de correlação cruzada do sinal de entrada e do de referência, e $\boldsymbol{W}^*$ aos coeficientes que minimizam a superfície de desempenho. No entanto, em muitas aplicações, os parâmetros desta superfície de desempenho quadrático são desconhecidos, e uma descrição analítica desta não está disponível. A localização dos pontos na superfície, entretanto, pode ser medida ou estimada calculando a média do erro quadrado ao longo de um período de tempo [11].

2.2.1. Algoritmo LMS

Um método bastante conhecido é *steepest descente,* que é prontamente implementado e provou o seu valor numa ampla variedade de aplicações práticas. É um método de pesquisa de gradiente que também faz com que todos os componentes do vetor de coeficientes sejam alterados a cada iteração [11]. O procedimento de pesquisa de gradiente iterativo pode ser representado algebricamente como [[11], eq. (11)]:

$$\boldsymbol{W}(k+1) = \boldsymbol{W}(k) - \ \mu\, \nabla(k) \tag{11}$$

Em que k é o número da iteração, $\boldsymbol{W}(k)$ um vetor que é constituído pelos coeficientes do filtro no tempo "presente", enquanto $\boldsymbol{W}(k+1)$ é o novo valor dos coeficientes, μ é o passo, e $\nabla(k)$ é o gradiente da função de custo em relação a $\boldsymbol{W}(k)$. No método do gradiente estocástico, $C(k)$ é aproximado por $e^2(k)$, obtendo a seguinte expressão [[11], eq.(12)]:

$$\frac{dC(k)}{d\boldsymbol{W}(k)} = \frac{de^2(k)}{d\boldsymbol{W}(k)} = 2\frac{de(k)}{d\boldsymbol{W}(k)}e(k) = -2e(k)\boldsymbol{X}(k) \ \ \therefore \ \ \boldsymbol{W}(k+1) = \boldsymbol{W}(k) + \ \mu\, e(k)\boldsymbol{X}(k) \tag{12}$$

Para que este algoritmo seja estável, ou seja que convirja para os coeficientes desejados, o tamanho do passo tem de verificar [[6], eq. (13)]:

$$0 < \mu < \frac{2}{\mathrm{trace}(\boldsymbol{R})} \tag{13}$$

O valor $\mathrm{trace}(\boldsymbol{R})$ representa a soma dos elementos da diagonal principal (da esquerda superior para a direita inferior) de $\boldsymbol{R}$ [6] e é igual a N vezes a potência do sinal de entrada. A velocidade de convergência aumenta à medida que o valor do tamanho do passo é aumentado, até tamanhos de passo próximos da metade do valor máximo necessário para operação estável do sistema. Outro

problema corresponde às maiores flutuações dos coeficientes do filtro sobre as suas soluções ótimas em estado estacionário, no entanto se o tamanho do passo for muito baixo, poderá demorar demasiado tempo a convergir [9].

Em suma, o algoritmo LMS é simples, e ainda assim capaz de alcançar um desempenho satisfatório nas condições certas. A sua limitação principal é uma taxa de convergência relativamente lenta [6].

2.2.2. LMS Normalizado (NLMS)

O algoritmo LMS só é estável se o passo for limitado, e este limite do passo é inversamente proporcional à potência do sinal de referência [6].

Uma das principais desvantagens do algoritmo LMS é que é sensível à potência do sinal de entrada x(n), ou seja, o tempo de convergência é proporcional à potencia do sinal de entrada. Isto torna muito difícil (se não impossível) escolher um tamanho do passo que garanta a estabilidade do algoritmo em qualquer cenário [12].

Para se obter uma rápida convergência tem de se ter um tamanho de passo elevado de modo que o sistema não divirja, assim o algoritmo pode se tornar instável muito facilmente se os valores de entrada forem superiores ao esperado. O filtro dos mínimos quadrados médios normalizados (NLMS) é uma variante do algoritmo LMS que resolve este problema através da normalização da potência da entrada. Obtendo assim a seguinte equação:

$$\boldsymbol{W}(k+1) = \boldsymbol{W}(k) + \mu \frac{e(k)\boldsymbol{X}(k)}{\boldsymbol{X}(k)^T \boldsymbol{X}(k)} \tag{14}$$

Este algoritmo é estável desde que 0 < μ < 2 e, claro, $\boldsymbol{X}(k)^T \boldsymbol{X}(k) \neq 0$. A fim de evitar esta última possibilidade, na prática, é normalmente adicionado um valor à potência do sinal de referência, onde este valor é selecionado para ser suficientemente pequeno quando comparado com a potência do sinal de referência [13].

$$\boldsymbol{W}(\boldsymbol{k}+\boldsymbol{1}) = \boldsymbol{W}(\boldsymbol{k}) + \boldsymbol{\mu} \frac{\boldsymbol{e}(\boldsymbol{k})\boldsymbol{X}(\boldsymbol{k})}{\boldsymbol{X}(\boldsymbol{k})^{\boldsymbol{T}} \boldsymbol{X}(\boldsymbol{k}) + \boldsymbol{bias}} \tag{15}$$

Como a potência do sinal de referência não tem impacto no tempo de convergência neste algoritmo logo tem se mais controlo na escolha do tamanho do passo e consequentemente consegue se escolher um tamanho do passo mais adequado para que o este algoritmo convirja mais rápido do que o LMS tradicional. A convergência do algoritmo LSM depende nas estatísticas de sinais de entrada cujo conhecimento, geralmente é bastante limitado.

Embora muitos desenhadores de algoritmos estejam conscientes das vantagens do algoritmo NLMS em relação ao algoritmo LMS, o algoritmo NLMS nem sempre é utilizado na prática. A razão para isto é devido à elevada complexidade computacional da divisão pela potência do sinal de entrada [14].

2.2.3. Leaky LMS

Variantes do algoritmo LMS, tais como a versão leaky LMS, foram propostas para lidar com alguns problemas do algoritmo LMS, como por exemplo, quando não há sinal de entrada ou quando se utiliza um sinal sinusoidal sem ruído como entrada, o algoritmo LMS pode resultar em divergência dos pesos adaptativos. Isto acontece porque as paragens de atualização de peso, ou seja, os aumentos de peso são muito pequenos, e os efeitos de precisão finita podem fazer com que os pesos sem restrições cresçam sem limites, resultando em transbordamento (overflow) durante o processo de atualização de peso. A introdução de fugas no algoritmo LMS estabiliza o sistema, este algoritmo (leaky LMS) baseia-se na adição de um parâmetro à equação de atualização, referido como factor de fuga, que tende a influenciar cada peso do filtro para zero. Outro benefício da utilização do algoritmo leaky LMS diz respeito ao alívio da "paralisação", onde a estimativa do gradiente é demasiado pequena para ajustar os coeficientes do algoritmo devido a um sinal de entrada muito baixo [15].

O valor do factor de fuga é em geral determinado pelo projectista numa base experimental, como um compromisso entre robustez e perda de desempenho do filtro adaptativo, devido à adição de ruído branco. Normalmente um bom ponto de partida é um factor de fuga ligeiramente inferior a 1. Foi demonstrado que o leaky LMS pode ser utilizado para melhorar a estabilidade numa implementação de precisão finita, atenuando os efeitos da excitação não persistente, e reduzir efeitos indesejáveis.

Sabe-se que as fugas resultam em distorções nos coeficientes e, como tal, têm de ser mantidas com valores baixos. O leaky LMS resulta nas seguintes equações, em que o factor de fuga corresponde a φ.

$$C(k) = e(k)^2 + \gamma \boldsymbol{W^T}(k)\boldsymbol{W}(k) \tag{16}$$

$$\frac{dC(k)}{d\boldsymbol{W}(k)} = \frac{d\left(e(k)^2 + \gamma \boldsymbol{W^T}(k)\boldsymbol{W}(k)\right)}{d\boldsymbol{W}(k)} = -2e(k)\boldsymbol{X}(k) + 2\gamma \boldsymbol{W}(k) \tag{17}$$

$$\therefore \quad \boldsymbol{W}(k+1) = \boldsymbol{W}(k) + \mu\, e(k)\boldsymbol{X}(k) - 2\,\mu\,\gamma\, \boldsymbol{W}(k) \iff \tag{18}$$

$$\iff \boldsymbol{W}(k+1) = (1-\varphi)\, \boldsymbol{W}(k) + \mu\, e(k)\boldsymbol{X}(k) \tag{19}$$

Onde $\varphi = 2\,\mu\,\gamma$.

2.2.4. LMS com tamanho do passo adaptativo (VSS LMS)

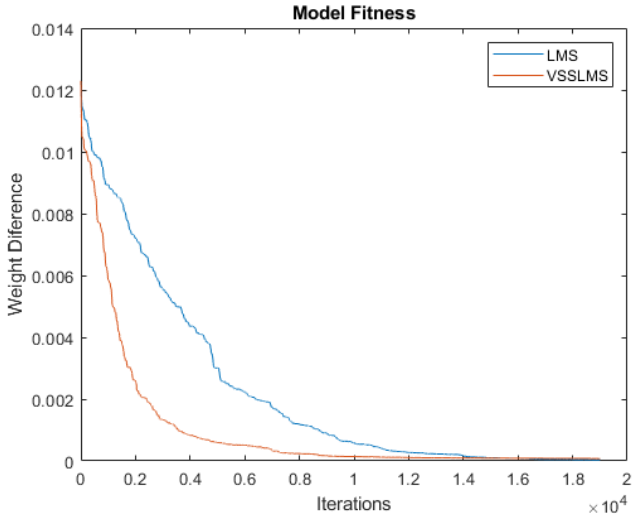
O erro quadrático médio em estado estacionário (MSE), no que diz respeito ao desempenho, é proporcional ao tamanho do passo de adaptação e o tamanho do passo controla a taxa de convergência para os pesos ótimos. Um tamanho do passo pequeno é geralmente escolhido para obter um excesso final de erro pequeno, mas causa uma convergência lenta. Tal escolha não é aceitável em muitos ambientes estatísticos desconhecidos, como por exemplo, as comunicações móveis sem fios e outros.

No algoritmo VSS LMS, o tamanho do passo é grande quando os pesos do LMS estão longe dos pesos ideais. À medida que os pesos se aproximam dos ótimos, o tamanho do passo diminui gradualmente para um valor pequeno, tendo assim um tempo de convergência rápido e um pequeno erro final [16].

O tamanho do passo aumenta ou diminui à medida que o erro quadrático médio aumenta ou diminui, permitindo ao filtro adaptativo acompanhar as mudanças no sistema, bem como produzir um pequeno erro no estado estável. A atualização do tamanho do passo tem a seguinte equação:

$$\mu(n+1) = \alpha\mu(n) + \gamma e^2(n) \tag{20}$$

Este algoritmo é estável desde que 0 < α < 1, γ > 0 e $\mu_{min} < \mu(n) < \mu_{max}$, normalmente, μ_{min} será próximo do valor de μ que seria escolhido para o algoritmo LMS. O valor para μ_{max} é o valor máximo que o sistema pode utilizar sem divergir.



O algoritmo VSS LMS proporciona uma convergência muito mais rápida do que o algoritmo LMS para o mesmo nível de desajustamento, como se pode ver pela Figura X. Este algoritmo existe para reduzir o tradeoff entre o desajuste e a capacidade de rastreio do algoritmo LMS. O algoritmo VSS LMS também reduz a sensibilidade do desajustamento ao nível de não-estacionariedade [17].

3. CANCELAMENTO ATIVO DE RUÍDO

Os sistemas de controle de ruído ativo fornecem atenuação de som ao introduzir uma segunda onda de som gerado eletronicamente no ambiente acústico. Se a amplitude da segunda onda sonora for igual à do primeiro, mas a fase invertida, então as duas serão canceladas [18]. Um dos focos deste projeto é um tipo específico de controlador, que é o mais comum entre pesquisadores e desenvolvedores, e denominado de controlador adaptativo *feedforward*, cuja estrutura é representada na Figura 7. O controlador adaptativo feedback será também um tópico de estudo, representado pela Figura 8.

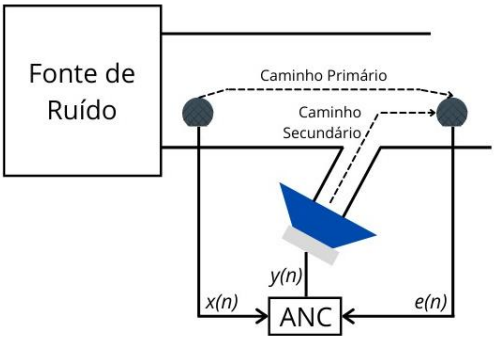


Figura 7 – Sistema *feedforward* CAR

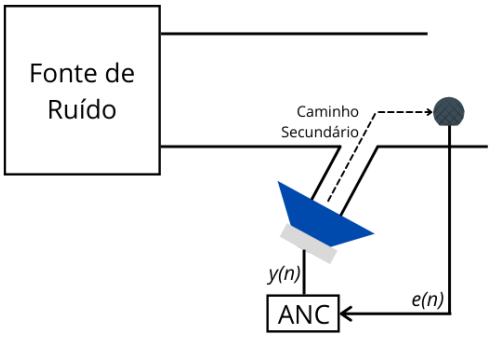


Figura 8 - Sistema *feedback* CAR

3.1. SISTEMA *FEEDFORWARD CAR*

O sinal de ruído recebido no microfone de referência é representado por x(n), y(n) é o sinal de saída do filtro linear, e e(n) o sinal de erro obtido no microfone de erro. O caminho primário, que é composto pelo caminho percorrido pelo sinal de ruído desde o microfone de referência até ao microfone de erro.

Existe ainda um caminho secundário ($\widehat{S'}$), que é composto pelo percurso que o sinal de antirruído tem de percorrer até chegar ao microfone de erro, em conjunto com todos os atrasos e atenuações que o hardware pode introduzir no sistema. Este caminho secundário conduziu à existência de novos algoritmos adaptativos como o algoritmo “Filtered-x LMS" (FxLMS), que é a adaptação para CAR algoritmo LMS [19].

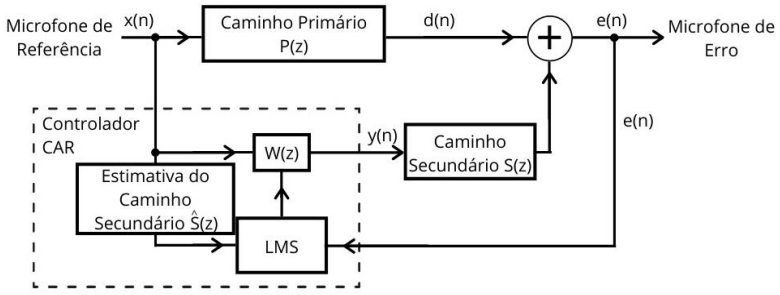


Figura 9 - Diagrama de Blocos FxLMS com *feedforward*

Neste algoritmo, as amostras do sinal de entrada são filtradas pela estimativa da função de transferência do caminho secundário como ilustrado na Figura 9. Este conjunto filtrado de sinais é então multiplicado pelo sinal de erro para produzir a estimativa de gradiente usada para modificar os valores de peso atuais, de forma que os níveis de atenuação de perturbação sejam melhorados [18].

Para poder filtrar as entradas com a função de transferência do caminho secundário, é necessário primeiramente estimá-lo ($\hat{S}$). Tal pode ser feito de duas formas, de modo offline ou online, sendo que no modo offline a estimação ocorre no arranque do sistema, uma vez que o caminho secundário praticamente não varia com o tempo [19].

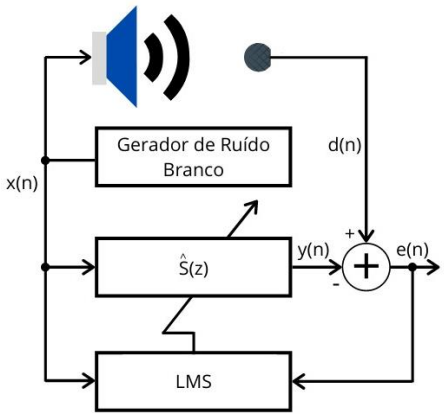


Figura 10 - Estrutura para estimação do caminho secundário

Para efetuar a estimação do caminho secundário no modo *offline*, o sistema de controlo tem de gerar ruído branco e enviar para o auscultador [19], como ilustrado na Figura 10. Assim, o sinal de erro será principalmente o ruído branco depois de passar pelo caminho secundário. Com isto, é possível utilizar o método LMS para estimar o caminho secundário.

Em aplicações em que exista um forte sinal de antirruído, este pode interferir no sinal de referência, que deveria apenas conter o sinal de ruído. Assim, existe uma técnica de cancelamento do caminho de *feedback,* em que esta pode ser incorporada no sistema CAR [19], resultando no diagrama da Figura 11.

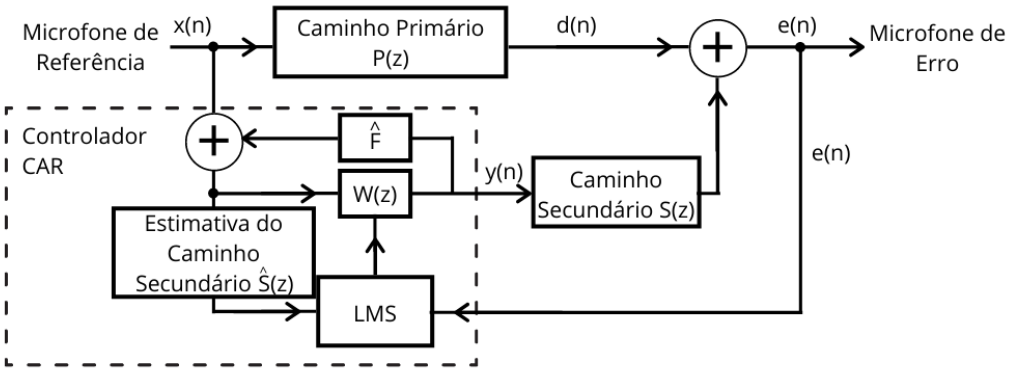


Figura 11 - Diagrama de Blocos FxLMS com Cancelamento do Caminho de Feedback

O filtro do caminho de feedback $\widehat{F'}$ pode ser obtido de uma forma semelhante a $\widehat{S'}$. Adicionalmente, com a utilização de SDMs, que causam ruído nas altas frequências, é necessária uma maior atenção para limitar o ganho de alta frequência de W e $\widehat{F}$, uma vez que estes dois filtros estão num *loop* e o ganho do mesmo precisa de ser menor que um, ou o sistema torna-se instável. No entanto, para ajudar a resolver este problema podem ser inseridos filtros passa-baixo para reduzir a potência do ruído nas altas frequências [20].

3.2. SISTEMA *FEEDBACK CAR*

Num sistema feedback CAR o microfone de referência não é utilizado, pelo que não está disponível um sinal de referência. Uma vantagem deste sistema é que evita o problema da realimentação acústica pertencente aos sistemas *feedforward* de dois microfones, que foram discutidos anteriormente, e reduz também o hardware necessário (ADCs, microfones e amplificadores). Uma das desvantagens é que o sinal de referência tem de ser estimado, sendo assim, este sinal pode conter ruído.

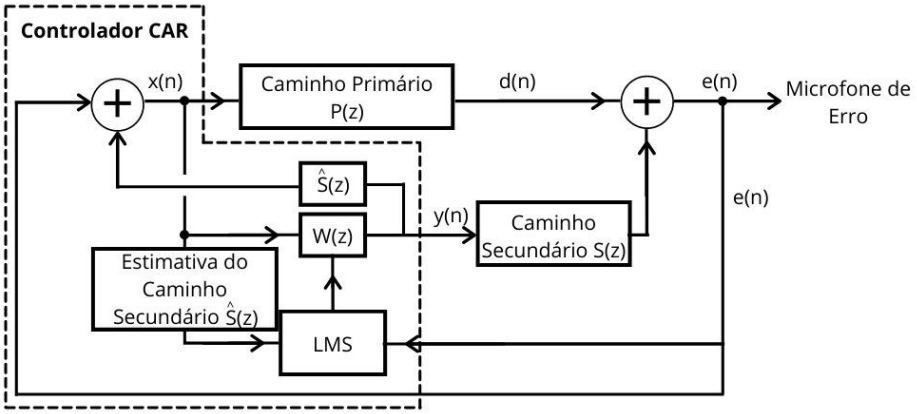


Figura 12 - Diagrama de Blocos FxLMS com *feedback*

Utilizando a topologia da figura 10, mas com um caminho de feedback que é a estimativa do caminho secundário consegue se obter uma estimativa do sinal desejado se $\widehat{S'}$ for aproximadamente igual a $\boldsymbol{S}$ (Figura 12). As seguintes equações demonstram que a afirmação anterior é verdadeira.

$$e(n) = d(n) + z(n) \Leftrightarrow \tag{21}$$

$$\Leftrightarrow e(n) = d(n) + \boldsymbol{X}^T(n)\boldsymbol{WS} \Leftrightarrow \tag{22}$$

Em que Z(n) é y(n) após passar pelo caminho secundário, $\boldsymbol{X}^T(n)$ é o vetor transposto que contém os sinais de entrada, e $\boldsymbol{W}$ é o vetor que contém os coeficientes do filtro, logo:

$$\Leftrightarrow \boldsymbol{x}(n) = \boldsymbol{e}(\boldsymbol{n}) - \boldsymbol{y}(\boldsymbol{n})\widehat{S'} \Leftrightarrow \tag{23}$$

$$\Leftrightarrow \boldsymbol{x}(n) = d(n) + \boldsymbol{X}^T(n)\boldsymbol{WS} - \boldsymbol{X}^T(n)\boldsymbol{W}\widehat{S'} \Leftrightarrow \tag{24}$$

$$\Leftrightarrow \boldsymbol{x}(n) = d(n) \tag{25}$$

O sistema de *feedforward* tem um melhor desempenho uma vez que não faz uma estimativa do sinal de referência, mas sim obtém o sinal desejado a partir do microfone de referência não existindo nenhuma estimativa, que se for mal calculada no sistema feedback o sistema pode não convergir ou até divergir.

4. TÉCNICAS DE PROCESSAMENTO DE SINAL COM SOBRE AMOSTRAGEM

Para que os sistemas CAR sejam capazes de atingir uma quantidade significativa de redução de ruído, é necessário implementar o controlador com pequeno atraso. No entanto, na maioria das aplicações CAR, esse atraso é significativo devido à taxa de amostragem tipicamente baixa, ao *anti-aliasing* (AA) e aos filtros de reconstrução (RC) dos conversores AD e DA [20].

Uma técnica para diminuir esse atraso consiste no aumento da frequência de amostragem, mas com a consequência de um aumento significativo na complexidade computacional, não descorando também que o hardware iria ter um custo superior. A utilização de conversores sigma-delta, apesar de estes funcionarem em altas frequências de amostragem, usam filtros AA e RC, com grande atraso, que torna o seu uso em CAR limitado. Uma forma de ultrapassar este problema seria remover por completo os filtros AA e RC, trabalhando assim com os sinais sobre amostrados no controlador, o que traria novamente o mesmo problema referido anteriormente, isto é, de consequentemente aumentar a complexidade computacional. Contudo, com a utilização dos conversores sigma-delta é possível diminuir o número de *bits* por amostra, e consequentemente reduzir substancialmente a complexidade computacional [7]. Um conversor sigma-delta é constituído normalmente por um SDM, um filtro passa baixo seguido de um processo de subamostragem.

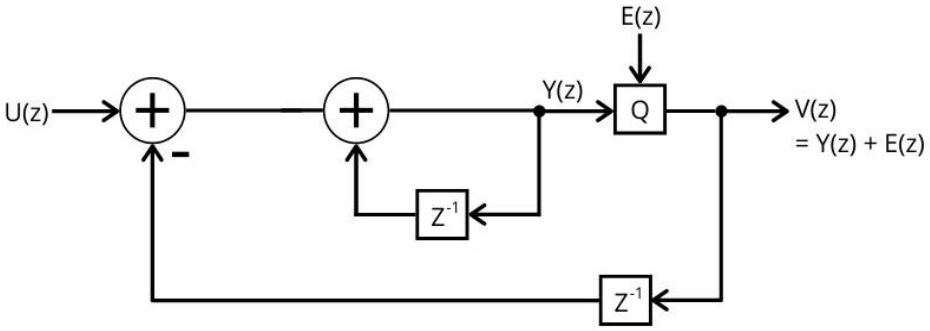


Figura 13 - Diagrama de Blocos de Domínio Z de SDM de primeira ordem

$$V(z) = STF(z)U(z) + NTF(z)E(z) \quad (26)$$

A Figura 13 representa um SDM de primeira ordem, em que neste caso a função de transferência do sinal (STF) é unitária e a função de transferência de ruído (NTF) é de $(1 - z^{-1})$. Para SDM de ordem *P*, a STF continua unitária, mas a NTF é dada como $(1 - z^{-1})^P$. Em relação à equação 26, U(z) representa a entrada do SDM, V(z) a saída do SDM, e E(z) o erro que o quantizador introduz no sistema, em que este quantizador é representado por Q. Este tipo de SDMs tem uma implementação simples, conforme apresentado na figura.

Em relação à escolha do quantizador, o método mais simples e historicamente mais antigo comporta na utilização de quantização de bit único. No entanto, existem grandes vantagens em empregar um quantizador *multi-bit*, uma vez que o erro de quantização é reduzido 6dB por cada bit adicionado na resolução do quantizador, o *loop* de *feedback* torna-se mais linear, pois as variações

do ganho efetivo do quantizador com o seu sinal de entrada são reduzidas, e consequentemente, o NTF pode ser escolhido de forma mais hostil [21].

A aplicação comum dos SDMs é utilizar frequências superiores á da frequência de amostragem do sistema, para diminuir o ruído de quantização. No entanto, para este projeto, como a frequência de amostragem pretendida é bastante elevada, não será necessário os SDMs trabalharem a uma frequência superior, uma vez que entre amostras os seus valores não variam muito na banda de frequências audíveis.

Em relação à estabilidade dos SDMs, se o quantizador for um bit, então é evidente que o *loop* se torna instável e, especificamente, y torna-se ilimitado, se $|u| > 1$. Por exemplo, se $u = 1{,}3$, o *loop* de realimentação tentará desesperadamente equilibrar u, realimentando assim um sinal de -1. Mesmo assim, uma entrada de 0,3 entrará no integrador a cada período de relógio, até que y cresça tanto que o circuito se torne disfuncional. Para quantizadores multi-bit, em que M é o número de passos e λ meio passo do quantizador, se $M \geq 2^P - 1$, então o SDM é estável para todos os *inputs* que respeitam a condição $|u(n)| \leq \lambda(M + 2 - 2^P)$ [21].

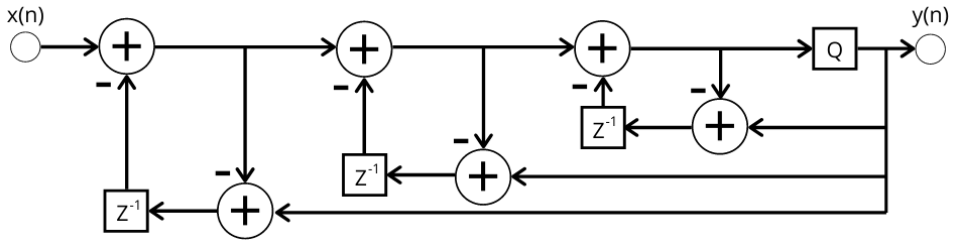


Figura 14 - Bloco SDM de ordem P = 3.

A implementação do SDM representada na Figura 14 tem exatamente as mesmas funções transferência que a implementação representada na Figura 13, mas com a diferença de que esta é um SDM de terceira ordem que necessita de P registos de armazenamento. Ao utilizar diferentes NTF, um menor número de estádios de quantização e menos memória podem ser empregues, porém, a complexidade computacional aumenta [20].

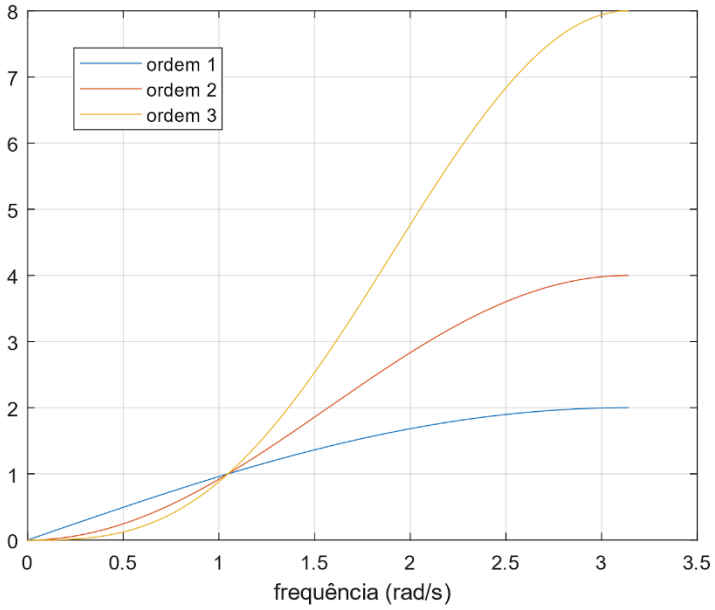


Figura 15 - Curvas de Modelagem de Ruído e Espectro de Ruído em SDM de primeira, segunda e terceira ordem

A NTF do SDM representa uma função de filtro passa-alto, que é retratada pela Figura 15. Esta suprime o erro nas frequências em torno de 0, mas a função NTF também aumenta o erro nas frequências mais altas a partir de aproximadamente $0.3\,f_s$. Com o aumento da ordem do SDM, é possível observar que o erro é suprimido com mais intensidade nas baixas frequência, mas é salientado ainda mais nas altas frequências.

Este erro causado pelo SDM irá alastrar-se para o algoritmo LMS, o que não representa qualquer problema desde que respeite a seguinte condição: $(f_s > K * 2 * f_B)$, em que f_s representa a frequência de amostragem, K é a razão de sobre amostragem, e f_B a frequência do sinal de interesse. A frequência f_B deve ser selecionada maior que a maior frequência audível pelo ouvido humano para sinais acústicos, ou maior que a frequência de corte da planta [20].

5. ESTADO DA ARTE

Um filtro adaptativo pode ser utilizado na modelagem, ou seja, copiar o comportamento de sistemas físicos dinâmicos. Tanto o sistema desconhecido, como o filtro adaptativo são estimulados pela mesma entrada. O filtro adaptativo ajusta-se com o objetivo de fazer com que sua saída seja igual à do sistema desconhecido. Se o filtro adaptativo convergir, o sistema torna-se um modelo do sistema desconhecido. [11]

Esta modelação de sistemas desconhecidos tem muitas aplicações para problemas práticos. Uma das aplicações é na área das comunicações, por exemplo, na transmissão de informação a longas distâncias (antenas). Neste tipo de comunicação, existem vários caminhos (multipath) do transmissor até ao recetor (i.e. reflexões da terra), o que pode provocar interferência e eco no sinal recebido (Figura 16). Uma forma de ultrapassar este problema seria modelar os vários caminhos com um filtro adaptativo e assim no recetor é possível retirar a interferência e eco no sinal recebido [11].

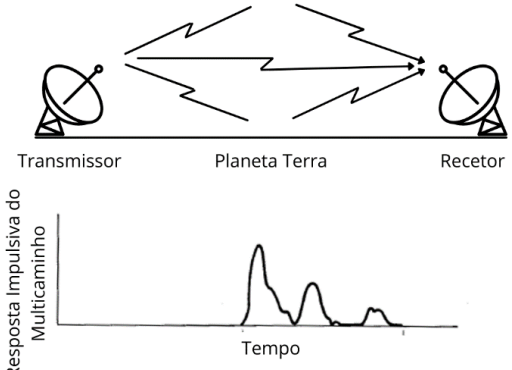


Figura 16 - Sistema de comunicação multicaminhos e a sua resposta impulsiva

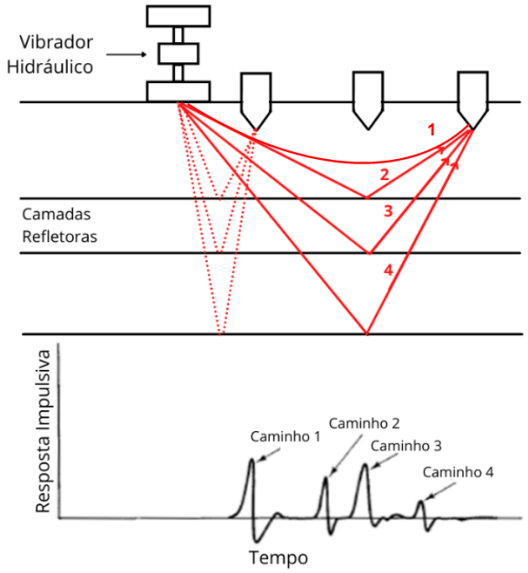


Figura 17 - Exploração geofísica e respetiva resposta sísmica impulsiva

Outra aplicação seria na exploração de petróleo e gás, em que a principal tecnologia utilizada para a deteção destes é a reflexão sísmica, onde é aplicada uma fonte de energia sísmica na superfície da Terra. Como as ondas sísmicas se propagam de uma camada geológica para outra, a mudança de materiais muitas vezes é acompanhada por mudança na impedância sísmica, o que causa reflexões (Figura 17). A deteção de reflexões permite detetar mudanças no material e, assim, detetar camadas na subsuperfície. Assim, medindo os atrasos temporais consegue-se ter uma estimativa da profundidade do plano de reflexão e, portanto, saber se existem potencias fontes de petróleo ou gás. Através da utilização de filtragem adaptativa consegue-se medir o tempo que

demora desde o envio do sinal sísmico até à receção através de reflexões, com uma grande precisão, uma vez que os filtros adaptativos têm uma grande flexibilidade a dados com ruído.

Este método foi primeiramente explorado por R. T. Cloud, no entanto este método foi esquecido durante a evolução da tecnologia, mas com o avanço de técnicas de filtragem adaptativa e software e hardware digital para sua implementação, o método de Cloud's é agora uma ferramenta viável na exploração geológica.[11]

Os filtros *Short Word length* (SWL), em que os sinais e coeficientes de filtro são armazenados utilizando um número reduzido de bits, têm vindo a ser explorados desde o início dos anos 80 [22]. Primeiramente, foram propostos filtros FIR ternários, que tiram proveito de um coeficiente fixo que pode apenas tomar o valor -1, 0, ou 1 [23]. Ainda que os sinais de 1-bit tenham demonstrado uma aplicabilidade em controlo em tempo real, e uma redução significativa da complexidade computacional [24], verificou-se que os filtros adaptativos SWL (como os coeficientes em formato 2-bit), têm uma performance superior do que os restantes (1-bit e ternários) [22]. Assim, é imprescindível que exista um *trade-off* entre a eficiência do *hardware* e performance [25], o que implica que uma implementação SWL produza uma melhor performance, em troco de uma maior complexidade computacional [26].

O sistema sigma-delta, provou não só ser mais eficiente na redução de ruído do que um modelador padrão [27], mas também capaz de obter uma resolução mais elevada comparativamente a circuitos analógicos [28]**,** tendo por isso ganho uma grande popularidade no processamento de sinais de áudio como um método eficaz para construir ADCs e DACs de alta resolução ([24], [21]).

Relativamente à ordem dos SDMs, tornou-se aparente em pesquisas anteriores, que moduladores de ordem superior têm um melhor desempenho em termos de redução de ruído, requisitos de taxa de sobre amostragem mais baixos, e maior flexibilidade no que diz respeito ao movimento do erro de quantização [[28], [29]]. Embora seja vantajoso empregar sistemas de modulação sigma-delta de ordem superior, podem surgir potenciais problemas à medida que o número de estágios aumenta, devido à correspondência de ganho, tolerância de componente, e outras complicações [7].

A implementação dos filtros mencionados anteriormente e outras técnicas baseadas em sigma-delta em FPGA foram examinadas. É reconhecível que este tipo de tecnologia irá fornecer cada vez mais soluções para problemas de processamento de sinal, bem como um mecanismo para trabalhar com variáveis importantes, como a procura crescente por uma melhor performance, dado que são capazes de manter a flexibilidade de soluções baseadas em software, enquanto providenciam elevados níveis de performance [30].

A minimização do ruído dentro de automóveis tem sido um tópico importante de pesquisa na indústria automóvel nas últimas duas décadas. Este problema foi inicialmente abordado através de métodos passivos de cancelamento de ruído, onde foram utilizados tratamentos físicos, como amortecimento estrutural e absorção acústica. No entanto, como os fabricantes de veículos

pretendem ter designs mais económicos e leves, os interiores dos carros tornaram-se mais ruidosos devido ao aumento das vibrações estruturais. Esses campos de ruído são geralmente dominados por baixas frequências (ou seja, 0-500 Hz), portanto, as abordagens convencionais de cancelamento de ruído passivo são menos eficazes. Para resolver este problema, foram desenvolvidos métodos de controlo ativo de ruído (CAR), onde fontes secundárias foram propostas para atenuar o ruído dentro da cabine. Com os modernos sistemas de entretenimento automotivo a fornecer de quatro a seis altifalantes embutidos, a adição de um sistema CAR tem um custo adicional relativamente baixo [31].

O sistema CAR provou ter um grande efeito na redução de ruído quando este é proveniente de um espaço fechado, como por exemplo, máquinas de lavar roupa. Quando o sistema é devidamente implementado, resulta numa redução global do ruído em vez de apenas em zonas locais. Como o ruido produzido tem de passar por uma zona sólida (invólucro da estrutura), para o som se propagar, é necessário que existam vibrações. Desta forma, ao colocar-se atuadores de vibração, em vez de altifalantes, diretamente na estrutura do invólucro, consegue-se controlar mais facilmente as vibrações que a fonte de ruído produz. Estudos recentes indicam que podem ser obtidos níveis muito elevados de redução de ruído, atingindo mais de 10 dB a nível global em todo o espaço da sala. É de mencionar que o nível de redução do ruído pode variar em diferentes locais da sala. Isto deve-se ao facto de serem possíveis níveis de redução de ruído mais elevados em locais onde o ruído primário é mais forte sem controlo. Além disso, vale a pena salientar que o ruído praticamente não é aumentado [32].

As vibrações estruturais devem ser consideradas um tópico importante devido a numerosas razões, tais como a falha da estrutura induzida pela fadiga, desgaste excessivo da maquinaria, e ruído irradiado. A fim de evitar os danos mencionados, o controlo ativo das vibrações tem sido investigado ao longo do tempo e vários algoritmos de controlo têm demonstrado um desempenho excecional na prática. No entanto, a maior parte dos algoritmos utilizados são invariantes no tempo, o que conduz a desafios para a supressão das vibrações de estruturas variáveis no tempo. Na realidade, muitas estruturas de engenharia sujeitas às ameaças de vibrações variam no tempo. Neste caso, estudos indicam que para que o algoritmo tenha um bom desempenho e melhor eficácia, é necessário que a estimação do caminho secundário seja feita de modo online e que o tamanho do passo seja variável [33].

Apesar da extensiva investigação do tema, e de já terem sido implementados diversos auscultadores com diferentes *designs*, tal como auriculares com cancelamento de ruído intra-auricular com o controlador ASIC [34], existem ainda oportunidades de melhoria, nomeadamente na redução de custos, e averiguação de técnicas alternativas de modelação de ruído para reduzir as componentes do sistema [35], principalmente porque existe uma pressão contínua no sentido de obter uma melhor performance com um menor consumo de energia numa menor área [36].

6. METODOLOGIA

Como referido anteriormente, neste projeto será implementado um protótipo de auscultadores com cancelamento de ruído sobre amostrado, a partir de uns auscultadores analógicos com CAR. O CAR do protótipo será implementado numa *ZedBoard* FPGA, em que os sinais dos microfones terão de passar por ADCs de alto desempenho, e o sinal de saída do CAR terá de passar por um DAC, também de alto desempenho, para poder ser transmitido pelos auscultadores.

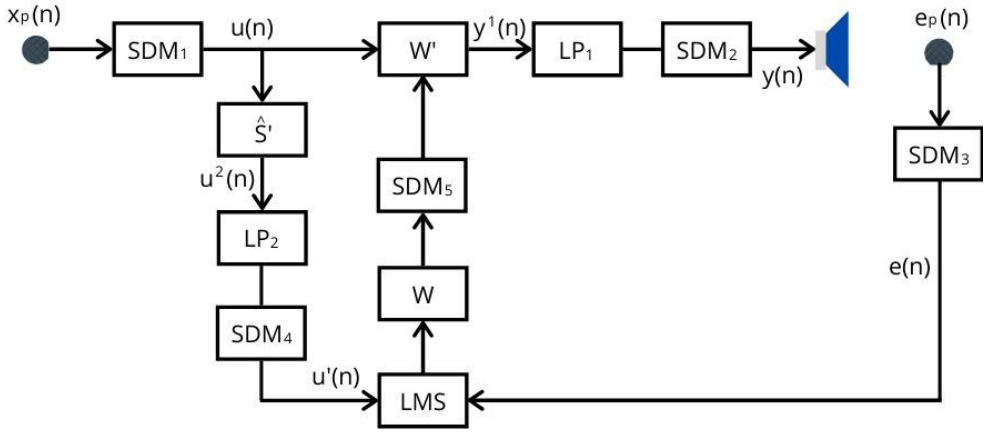


Figura 18 - Sistema CAR com algoritmo FxLMS *feedforward* e SDMs

Todos os blocos presentes no diagrama da Figura 18 serão implementados na FPGA. Os sinais $x_p(n)$ e $e_p(n)$ são medidos pelos sensores de referência e de erro após a sua passagem pelos ADCs, e o sinal $y(n)$ é a entrada para os auscultadores, depois de passar pelos DACs.

Os blocos SDM serão utilizados para reduzir o número de bits em geral de B bits (por exemplo 16 bits) para C bits (por exemplo 5 bits), baixando assim a complexidade computacional, não introduzindo erro nas baixas frequências, mas sim nas altas, não afetando, todavia, o bom funcionamento do sistema. O bloco SDM_5 converte W, que é o resultado da atualização dos coeficientes do filtro FIR (B bits), em W', que corresponde aos coeficientes usados no filtro FIR, mas com um menor número de bits (C bits). Para reduzir a complexidade computacional, a conversão é efetuada apenas um coeficiente em cada período de amostragem.

Para o bloco W', como este é a resposta da convolução de W com SDM_5, então iria ter de conter um número de coeficientes igual aos de W (L coeficientes) mais a ordem do SDM_5. No entanto, ao realizar-se a passagem dos coeficientes W por SDM_5 apenas se obtém L coeficientes, o que indica que ocorre truncação. Assim para o SDM_5 aplicou-se um método diferente do que para os restantes blocos SDM, em que neste se utilizou correção de truncação, uma vez que se não existisse esta correção, os valores de erro de truncação poderiam ser demasiado elevados para este projeto. Portanto, para que exista um menor erro de truncação, os últimos P (ordem dos $SDMs$) coeficientes de W não são quantizados. É de notar que este método implica que W' terá B bits para os últimos P coeficientes, e por isso não irá aumentar significativamente o custo computacional.

O filtro $\widehat{S'}$ representado utilizando apenas C bits, é obtido por quantização da estimativa de $\hat{S}$. Esta estimação acontece sempre no arranque do sistema com a utilização do método offline, como referido anteriormente. Como $\hat{S}$ tem um número limitado de coeficientes, este número tem de ser bem apurado, uma vez que a estimação tem de apresentar uma resposta em frequência e amplitude muito semelhantes ao caminho secundário real, portanto, se o número de coeficientes for baixo, pode não se conseguir uma boa estimativa do caminho secundário, e se forem demasiados verifica-se uma utilização desnecessária de recursos.

O número de coeficientes depende também da taxa de amostragem do sistema, pois existe um trade off em relação à taxa de amostragem e ao tamanho do filtro. Se a taxa de amostragem for muito elevada, o número de coeficientes também tem de ser elevado para se conseguir uma boa estimativa, no entanto, podem não existir recursos suficientes para uma boa estimação.

No caso de $u^2(n)$ e $y^1(n)$, uma vez que estas sinais já foram processados por SDMs, estes são a entrada dos blocos SDM_2 e SDM_4, nesse caso, o ruído de quantização de alta frequência também dita o nível de entrada máximo. Assim, os filtros dos blocos LP (filtro passa baixo) reduzem o ruído de quantização de alta frequência antes do SDM_2 e SDM_4, permitindo assim reduzir os seus níveis de quantização.

Outro sistema que irá ser testado é o FxLMS feedback (Figura 19), que não necessita da utilização de um microfone de referência, exigindo assim menos hardware.

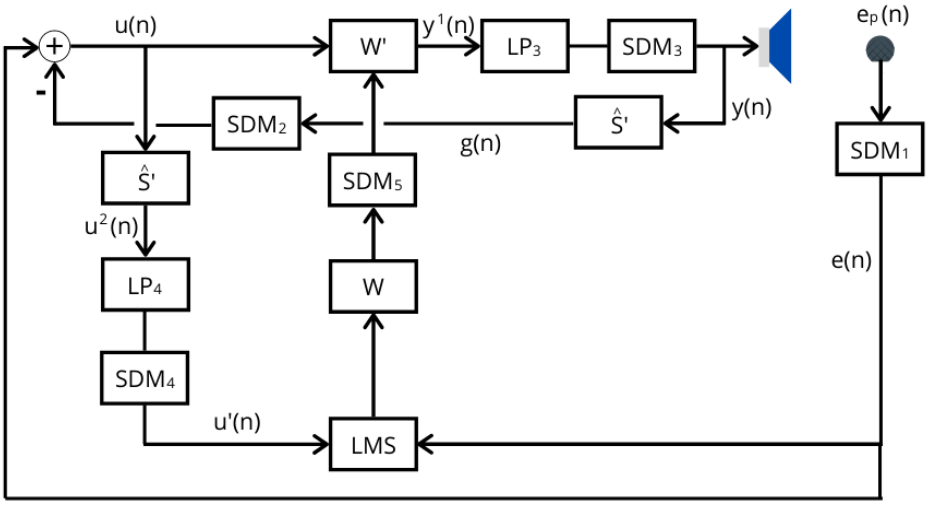


Figura 19 - Sistema CAR com algoritmo FxLMS *feedback* e SDMs

Este método tira partido da utilização do cancelamento do caminho de feedback para conseguir obter um sinal de referência. Para tal, é necessário que $\hat{F}$ seja igual a $\hat{S}$', assim se a estimação do caminho secundário for bem sucedida, então pela equação [$\boldsymbol{u}(n) = d(n) + \boldsymbol{X}^T(n)\boldsymbol{WS} - \boldsymbol{X}^T(n)\boldsymbol{W}\widehat{S'}$], os restantes termos são aproximadamente iguais, por isso cancelam-se, obtendo $\boldsymbol{u}(n) = d(n)$.

Neste caso, o caminho de cancelamento de feedback não tem um bloco LP antes do SDM, visto que se este bloco fosse colocado, estar-se-ia a alterar os coeficientes do filtro S^', não se obtendo assim um sinal de referência correto para toda a banda de frequências.

Uma vez que o caminho de cancelamento de feedback se encontra num caminho fechado, o ganho deste caminho não pode ser superior a 1, se não o sistema diverge. Este problema só ocorre se DF*W' > 1, em que DF é a resposta em frequência do erro de estimação de $\hat{F}$ (=$\hat{S}$'). Desde que $\hat{F}$ (=$\hat{S}$') seja estimado corretamente, o caminho de realimentação é cancelado donde o caminho fechado tem ganho zero.

Uma desvantagem para este método é ter de se estimar o sinal de referência enquanto o sistema *feedforward* utiliza hardware para o obter, o que implica que se o caminho secundário for de apenas um atraso de D amostras, então o sinal antirruído terá de ter um avanço de D amostras de forma a compensar o atraso do caminho secundário, então para que ocorra cancelamento tem de se verificar o seguinte: y(n) = d(n+D) ⟺ y(n)*S=d(n)). Isto significa que é necessário que o sinal antirruído esteja D amostras à frente, e tal só é possível à custa de um aumento de ruído de alta frequência, ou seja, para se reduzir o ruído nas baixas frequências tem de se aumentar nas altas, este efeito é conhecido como waterbed effect.

No caso de ruído branco, este efeito pode resultar num aumento da potência total do ruído, no entanto se o ruido for nas baixas frequências, como é normalmente devido à filtragem passiva dos auscultadores e dos microfones, é possível reduzir a potência total de ruído, reduzindo o ruído nas baixas frequências e aumentando nas altas.

No sistema de *feedforward* estes problemas não ocorrem, uma vez que o caminho primário terá um maior atraso em relação ao caminho secundário, logo o sinal de referência estará mais que D amostras atrás em comparação com o sinal de erro. Assim, apenas é necessário que o sinal antirruído tenha um atraso relativamente ao sinal de referência e não um adiantamento.

6.1. IMPLEMENTAÇÃO EM FPGA DO SISTEMA FXLMS

Como referido anteriormente, o sistema CAR será implementado na totalidade utilizando uma FPGA, onde os blocos SDMs serão emulados, utilizando a mesma lógica presente na Figura 14, mas para uma ordem variável, e com um número de registos equivalente à ordem do SDM (P) definida. O bloco de SDM primeiro calcula a sua saída com os registos calculados anteriormente e só posteriormente os registos são atualizados. Inicialmente foi desenvolvido código para a implementação do bloco SDM na placa FPGA, apresentado no Anexo 1.

O sinal de referência será processado por um ADC para que a FPGA consiga interpretar este sinal de uma forma discreta, o que também ocorre para o sinal de erro. Com o sinal de referência discreto ($u(n)$), e com um número de amostras anteriores iguais à dimensão do filtro (L), consegue-se obter um resultado do filtro FIR. Estas amostras anteriores estão sempre a ser atualizadas em cada ciclo de relógio. Uma vez que os coeficientes do filtro FIR, e o sinal de referência discreto são representados com C bits (menor número de bits), assim, o resultado do filtro FIR ($y^1(n)$) terá pelo

menos duas vezes C bits. Portanto, para reduzir este número de bits terá de passar por outro SDM, no entanto para reduzir os seus níveis de quantização necessita de passar anteriormente por um bloco LP. A saída deste SDM (y(n)) é o sinal antirruído na forma discreta, que por sua vez passa por um DAC para ser transmitido pelo auscultador.

No que diz respeito à atualização dos coeficientes do filtro FIR, o sinal $u(n)$, terá de ser processado pela estimativa do caminho secundário ($\hat{S}$), e como o sinal $u(n)$ e os coeficientes de $\hat{S}$ são ambos representados usando C bits, então o resultado do processamento é de duas vezes C bits, tal como ocorre com o sinal $y^1(n)$, e por esse motivo, emprega-se a mesma solução de utilizar um LP e de seguida um SDM, obtendo assim o sinal $u'(n)$ representado por C bits. Com os sinais $u'(n)$ até $u'(n-L)$, e o sinal de erro discreto ($e(n)$), é possível fazer a atualização dos coeficientes W, que por sua vez são processados por um SDM para atualizar os coeficientes em W'.

Fazendo este processo iterativamente, obtém-se o algoritmo FxLMS, em que sempre que ocorre alguma alteração no caminho primário, este consegue se adaptar para obter o mínimo ruído possível no microfone de erro.

7. IMPLEMENTAÇÃO EM HARDWARE

Neste capítulo será feita uma descrição de todo o sistema, com o hardware e software desenvolvidos para o projeto, tendo as opções de design justificadas. O hardware utilizado foi um circuito analógico desenvolvido por um ex-aluno do Instituto Superior Técnico, que realizou um projeto na mesma área, uma ZedBoard e conversores. O software implementado foi maioritariamente o descrito no tópico 6. Adicionalmente, serão apresentados neste capítulo os resultados da implementação efetuada.

7.1. HARDWARE

O hardware deste sistema CAR é composto por auscultadores, que contêm um par de microfones e altifalantes, um circuito analógico para amplificar os sinais destes, uma placa de desenvolvimento ZedBoard e conversores AD e DA para conseguir interpretar estes sinais.

Para este protótipo não foram desenvolvidas placas de circuito impresso, e como este projeto foi implementado com uma frequência elevada para os vários componentes, verificou-se uma limitação no que diz respeito à interferência entre sinais. Este problema deveu-se ao facto dos cabos utilizados não serem os mais adequados para altas frequências, e foi ultrapassado eficazmente com a utilização de cabos com o menor comprimento possível.

7.1.1. ZedBoard

A ZedBoard é um kit de desenvolvimento completo para designers interessados em soluções de alta performance, pois a placa contém todas as interfaces e funções de suporte necessárias para permitir uma ampla gama de aplicações.

As características de expansão que a placa fornece permitem uma rápida prototipagem e desenvolvimento para prova de conceitos. Vários conectores de expansão expõem o sistema de processamento para fácil acesso do utilizador.

Esta placa é dividida em duas partes, o Processing System (PS) e a Programable Logic (PL). O PS contém um Advanced RISC Machine (ARM) que apresenta dois *cores*, que trabalham a uma frequência de 33MHz, que podem ser programados individualmente e de forma a comunicarem entre si, bem como muitos outros componentes, como apresentado no Anexo 4. Esta parte da ZedBoard irá ser apenas utilizada para a extração de dados, pois o processamento irá ser totalmente efetuado na parte PL. A FPGA é o tipo de lógica programável que está presente na ZedBoard e contem um número limitado de recursos (Figura 20).

Device Name	Part Number	Progammable Logic Cells	Look-Up Tables (LUTs)	Flip-Flops	Block RAM	DSP Slices
Z-7020	XC7Z020	85K	53 200	106 400	4,9 Mb	220

Figura 20 - Recursos disponíveis pela ZedBoard

Esta não é a única limitação em relação à FPGA. Quando a complexidade da solução desenhada é muito elevada, isto resulta numa grande dificuldade na colocação dos recursos, dado que alguns

dos recursos têm de estar adjacentes, criando assim blocos. Mesmo que os recursos sejam suficientes para a solução, a colocação dos recursos na FPGA pode não ser possível devido às restrições.

VHDL ou "VHSIC Hardware Description Language" (Linguagem de descrição de hardware VHSIC "Very High Speed Integrated Circuits") é a linguagem utilizada para facilitar o desenho de circuitos digitais em FPGA. Esta foi a linguagem empregue ao longo deste projeto, já que permite que o comportamento do sistema seja descrito (modelado) e verificado (simulado) antes que as ferramentas de síntese traduzam o projeto em hardware real.

A escolha da FPGA em relação a outras opções deveu-se ao facto deste projeto necessitar de um alto valor para a frequência de amostragem e da necessidade de muitos recursos. Comparando com DSPs (Digital signal processor), para taxas elevadas de amostragem, este pode ter dificuldade em capturar, processar e emitir os dados sem qualquer perda. Isto deve-se aos muitos recursos partilhados, barramentos e mesmo ao núcleo dentro do processador. A FPGA, contudo, pode dedicar recursos a cada função.

7.1.2. Auscultadores

Para este projeto foram utilizados os Auscultadores ATH-ANC1 QuietPoint® (Figura 21) que contêm um circuito com cancelamento ativo de ruído, que neste caso não será utilizado. Ao utilizar estes auscultadores o projeto é simplificado por um lado, porque o dimensionamento do microfone, altifalante e acústicas do ar já foi executado, no entanto, por este motivo, existe a desvantagem da performance ser limitada.

Figura 21 - Auscultadores ATH-ANC1 QuietPoint

Estes auscultadores apenas contêm dentro de cada cápsula um altifalante e um microfone de erro, não contendo por isso o microfone de referência, o que proporciona apenas que seja utilizado o sistema feedback. No entanto, para se conseguir testar o sistema *feedforward*, pode-se utilizar o microfone de erro de uma das cápsulas como microfone de referência e a outra cápsula para cancelamento de ruído.

7.1.3. Circuito de Amplificação

O circuito de amplificação dos sinais do microfone e do altifalante, cujo diagrama de blocos está apresentado no Anexo 5, foi previamente desenvolvido por um ex-aluno que explorou a mesma área.

No que diz respeito ao sinal do microfone, este tem uma amplitude máxima de aproximadamente 26,4mV, antes de ser amplificado. É adicionado o valor de 2.5V a este sinal para que o amplificador fique sempre em funcionamento, uma vez que este não consegue amplificar tensões menores que 0V. Com a utilização de um potenciómetro é possível que o ganho deste sinal varie, tendo assim controlo na amplitude do sinal do microfone após amplificado, sendo que este sinal está centrado em 2.5V.

Em relação ao sinal enviado para o altifalante, este não pode ser enviado diretamente de um dispositivo que não tenha capacidades para fornecer correntes altas, isto porque a impedância do altifalante é de 130 ohms, como descrito na especificação dos auscultadores, logo é necessário um amplificador de potência de áudio. Da mesma forma anteriormente descrita, utilizou-se um potenciómetro de 200k ohms para controlar o ganho deste amplificador. Com a topologia usada existe uma limitação na banda de frequência útil, estando esta entre 19.4Hz e 16.2kHz com o potenciómetro a 200KHz.

7.1.4. Conversores AD e DA

Como o sistema vai trabalhar com frequências de amostragem elevadas, é necessário que os conversores sejam capazes de trabalhar a essas frequências. Primeiramente foi utilizado o TLC5540 Evaluation Module como conversor analógico para digital, que consegue funcionar a uma frequência máxima bastante elevada de 40MHz. No entanto, para este projeto apenas foi necessária uma frequência máxima de aproximadamente 1MHz, pelo que não foi utilizada toda a potencialidade do ADC. Este ADC converte um sinal analógico, de amplitude máxima de 1.14V centrado em 0V, num sinal digital de 8 bits. Como referido anteriormente, o sinal do microfone, depois de ampliado, terá uma amplitude máxima que varia a posição do potenciómetro, no entanto, o sinal está centrado em 2.5V. Assim, terá de se colocar um condensador entre a saída do amplificador e a entrada do ADC, para extrair a componente DC deste sinal, centrando-o a 0V. Variando o potenciómetro, é possível obter um sinal de amplitude máxima de 1.14V centrado em 0V para obter a máxima resolução do ADC.

Uma limitação deste ADC é que apresenta apenas um canal, o que significa que só se consegue amostrar um sinal, não sendo por isso possível testar o sistema de *feedforward* com este ADC, pois necessita da amostragem do microfone de referência e de erro.

Assim, no decorrer deste projeto, foi requisitado outro ADC que possuísse pelo menos dois canais, surgindo então o ADS8363EVM. Este apresenta dois canais, que possuem a capacidade de tirar amostragens em simultâneo, e conseguem amostrar sinais a uma frequência máxima de 1MHz, obtendo sinais digitais de 16 bits.

Este ADC consegue ser configurado para que os sinais a serem amostrados sejam do tipo diferencial, isto é, consegue fazer a diferença entre dois sinais, o que é útil para eliminar ruído eletromagnético que os cabos a altas frequências possam inserir.

Os sinais analógicos a serem amostrados necessitam de variar entre 0 e 5V, portanto, utilizando o modo diferencial, uma das entradas do ADC está ligada à saída do amplificador, em que esta varia entre 0 e 5V, e outra a 2.5V, obtendo assim um sinal digital que varia entre -2.5V e 2.5V.

Uma desvantagem na utilização deste novo ADC é que existe um protocolo de comunicação (serial interface) e comparativamente ao primeiro ADC este amostrava o sinal a cada ciclo de relógio. Portanto para o novo ADC foi necessário desenvolver código na ZedBoard para se conseguir obter o sinal amostrado.

Em relação aos conversores de digital para analógico, primeiramente, foi utilizado o MAX5216PMB1, que fornece o hardware necessário para fazer a interface do MAX5216 DAC de 16 bits com qualquer sistema que utilize Pmod, neste caso a ZedBoard contem o sistema Pmod, sendo por isso uma boa escolha para o projeto.

Este conversor recebe palavras de 16 bits e converte num sinal analógico com referência 2.5V. Uma limitação deste DAC é o facto de apresentar uma taxa de variação (Slew Rate) de 0.5V/us, o que implica que se fosse aplicado um sinal retangular a variar entre 0 e 2.5V, a frequência máxima seria de 200kHz. Porém, este problema não é critico, devido ao facto de o ouvido do ser humano apenas captar frequências até 20kHz. Outra limitação é o tempo de estabilização (Settling Time), pois para um sinal que varie entre 0 até 1.25V, demora 18us até que o sinal estabilize e quando não está estabilizado o sinal pode apresentar não linearidades.

Para que estes problemas não influenciassem o decorrer deste projeto, requisitou-se o DAC8814EVM, que apresenta uma resposta mais rápida em comparação com o DAC descrito anteriormente. Este conversor oferece uma excelente linearidade (± 1 bit menos significativo), baixo ruído e estabilização rápida (0,3us típ. para ± 0,1% da saída de escala completa).

7.2. SOFTWARE

O desenvolvimento de todo o software para este projeto foi efetuado em três aplicações: Vivado, Vitis, em que ambos são pertencentes da Xilinx, e MatLab. Estes ambientes de desenvolvimento da Xilinx incluem todos os recursos necessários para síntese e análise de designs VHDL e programação dos ARMs e FPGA. A plataforma Vivado serve para o desenvolvimento do código VHDL que será programado na FPGA. A aplicação MatLab irá realizar o processamento dos dados para conseguir obter de forma quantitativa se o sistema está a funcionar corretamente e também avaliar o desempenho do sistema.

7.2.1. Filtro Passa Alto

Foi desenvolvido um filtro digital passa alto, devido às leituras dos sinais dos microfones não estarem absolutamente centradas em zero, assim, com uma frequência de corte abaixo de 5Hz, é possível remover a componente DC, sem que interfira com a integridade dos sinais dos microfones. A abordagem para o desenho deste filtro foi desenvolver um filtro passa baixo e depois remover este sinal ao sinal do microfone, obtendo assim um sinal com atenuação nas baixas frequências. O mesmo efeito teria sido obtido ao desenhar apenas um filtro passa alto.

Um filtro passa baixo apresenta a seguinte transformada de Laplace:

$$H(s) = \frac{\alpha}{(\alpha - 1)s + 1} \tag{27}$$

Assim para um α igual a 2^{-15} e uma frequência de amostragem de 1MHz a frequência de corte é:

$$f_c = \frac{\alpha}{(1-\alpha)2\pi * \frac{1}{f_s}} \Leftrightarrow f_c = 4.8572\ Hz \tag{28}$$

Como se pode ver pela Figura 22, que é a resposta em frequência deste filtro, o sinal atenua por 3 decibéis quando está numa frequência de 4.85Hz como era espectável.

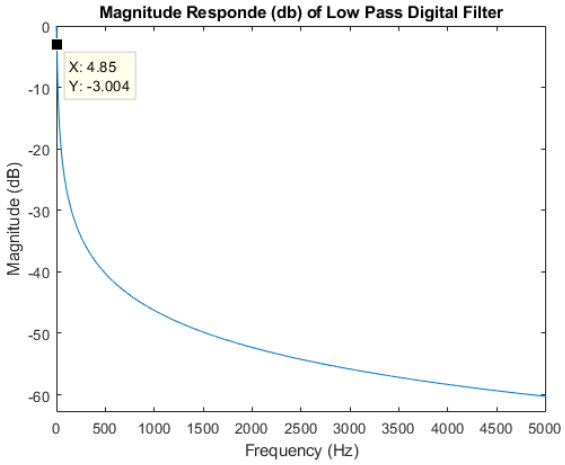


Figura 22 - A resposta em frequência de ganho-magnitude de um filtro passa baixo

Foram efetuados testes a este filtro passa alto e concluiu-se que tem um bom desempenho, embora demore algum tempo a estabilizar.

7.2.2. Estimação do caminho secundário

Para o sistema CAR, uma vez que é necessário estimar o caminho secundário, primeiramente, com o modo offline, foi efetuado o cálculo deste. Para tal, definiu-se um valor de 10M samples para a estimação, assim, para uma frequência de amostragem de 1MHz esta estimação demora 10 segundos. Nesta estimação, os coeficientes têm de ter uma resposta em frequência muito semelhante à do caminho secundário real para toda a banda de frequências, pelo que não se pode utilizar SDMs. Para obter uma boa resolução, optou-se por utilizar palavras de 16 bits, o que aumenta o número de recursos utilizados e a complexidade computacional. Para diminuir a complexidade computacional da utilização da estimação do caminho secundário no sistema CAR, realizou-se uma diminuição de bits através de arredondamento, passando a utilizar apenas 5 bits, em substituição da utilização de coeficientes com 16 bits, não causando grande disparidade na resposta em frequência.

Para gerar o ruído branco a ser enviado para o altifalante, foi implementado um linear-feedback Shift register (LFSR), cujo bit de entrada é uma função linear do seu estado anterior. Uma vez que o registo tem um número finito de estados possíveis, deve eventualmente entrar num ciclo de repetição.

No entanto, um LFSR com uma função de feedback bem selecionada pode produzir uma sequência de bits aparentemente aleatória por ter um ciclo muito longo. Assim, foi utilizado um Shift register de 16 bits, tendo um período de 65535 amostras, o que significa que após este número de amostras os valores voltam a repetir-se.

Este algoritmo consegue produzir uma sequência de valores pseudo-aleatórios, resolvendo iterativamente a expressão recursiva:

$$psrn_u = psrn\_u(14\ downto\ 0)\ \&\ (psrn\_u(15)\ xor\ psrn\_u(14)\ xor\ psrn\_u(12)\ xor\ psrn\_u(3));$$

Para a atualização dos coeficientes foi utilizado o algoritmo LMS, no entanto, como os sinais acústicos se somam no ar, então a atualização dos coeficientes muda, partindo da equação 11 e, como $e(n) = d(n) + y(n) = d(n) + \boldsymbol{X}^T(n)\boldsymbol{W}$ então:

$$\frac{dC(k)}{d\boldsymbol{W}(k)} = \frac{de^2(k)}{d\boldsymbol{W}(k)} = 2\frac{de(k)}{d\boldsymbol{W}(k)}e(k) = 2e(k)\boldsymbol{X}(k)\ \ \therefore\ \ \boldsymbol{W}(k+1) = \boldsymbol{W}(k) - \mu\, e(k)\boldsymbol{X}(k) \qquad (29)$$

Observa-se assim que apenas muda um sinal de soma para subtração. O tamanho do passo e o número de coeficientes para a estimação do caminho secundário foram determinados a experimentalmente para minimizar o erro.

7.2.3. Algoritmo FxLMS

Após a estimação do caminho secundário, o sistema fica em espera antes de começar a realizar o cancelamento de ruido, para certificar que o sinal anteriormente enviado pelo altifalante não é captado pelo microfone. Para tal, o sistema fica durante mil de amostras à espera, e nesse intervalo realiza o arredondamento dos coeficientes da estimação do caminho secundário, e coloca-se o sinal do altifalante a zero.

A linguagem VHDL causa atrasos em todos os sinais utilizados, isto é, quando se coloca um valor num sinal, este só aparece num próximo ciclo de relógio e, portanto, tem de se ter em consideração os atrasos inseridos no sistema para que os sinais na atualização dos coeficientes estejam no mesmo instante temporal. No sistema feedback tem também de se ter em atenção à estimação do sinal de referência, porque ao mínimo atraso de um dos sinais, o sinal de referência já não está correto.

Os filtros passa baixo (blocos LP), que estão antes dos SDMs, são apenas filtros de média móvel. Optou-se por utilizar este tipo de filtro, uma vez que necessitam de poucos recursos e não têm muita complexidade computacional. Utilizando um filtro de média móvel de dimensão 4, apenas são precisas 4 somas e um shift do resultado das somas para a direita de 2 bits, que teria o mesmo efeito se efetuasse o resultado da soma a dividir por 4. O objetivo da utilização dos blocos LP é reduzir o erro inserido nas altas frequências pelos SDMs. Assim, os SDMs de segunda ordem, utilizados neste projeto, têm o efeito apresentado na Figura 23 antes e após passarem pelos blocos SDM. É possível observar que com o filtro de média móvel, o erro inserido pelos SDMs é sempre atenuado para toda a banda de frequências.

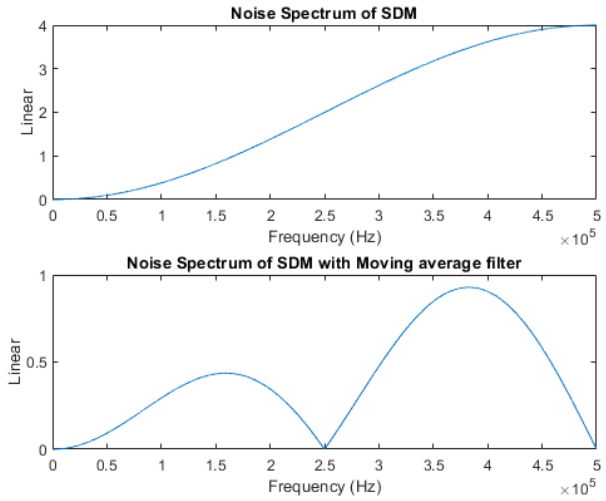


Figura 23 - Espectro de ruído do SDM antes e depois de passar pelo filtro de média móvel

Uma melhoria que se fez no cálculo da saída dos filtros FIR foi utilizar o paralelismo permitido pela linguagem VHDL. Logo, em vez de se realizar constantemente uma multiplicação de uma entrada por um coeficiente e subsequente soma a um acumulador, foi realizado um filtro FIR com redução. Ou seja, realizam-se todas as multiplicações, colocam-se num vetor, e após este estar totalmente preenchido, executam-se somas dois a dois em paralelo, criando assim uma árvore de somas, como a apresentada a título de exemplo na Figura 24. Uma desvantagem deste método é que o número de coeficientes tem de ser uma potência de base 2 (i.e. $2^6 = 64, 2^7 = 128, 2^9 = 512, \ldots$).

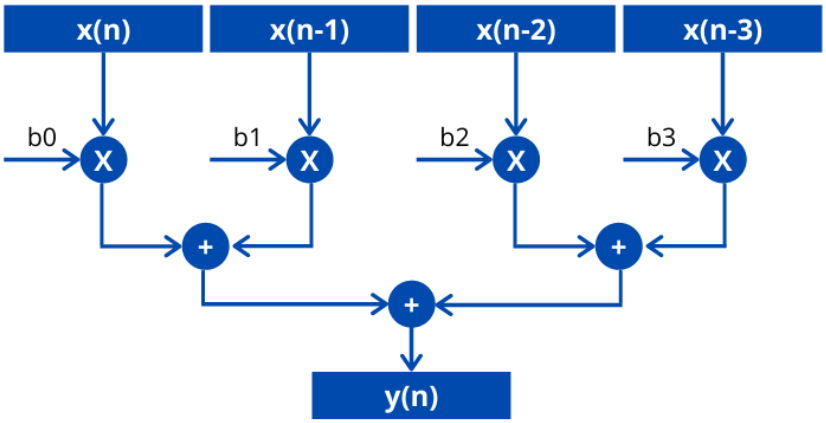


Figura 24 - Estrutura do filtro FIR com redução

Para a atualização dos coeficientes, utilizou-se o leaky LMS devido às vantagens referidas anteriormente. É de salientar que os tamanhos dos filtros foram variando ao longo do projeto, uma vez que estes dependem da frequência de amostragem e dos recursos disponíveis. Em relação ao tamanho do passo e ao fator de leakage, ambos foram determinados experimentalmente para obter melhores resultados.

7.2.4. Ferramentas de Desenvolvimento

O Vivado é a ferramenta onde se escreve o código VHDL para posteriormente se poder programar a FPGA. Nesta aplicação o código produzido é sintetizado, implementado e gera um bitstream para programar a FPGA.

Esta plataforma permite a criação de vários blocos (IP - intellectual property), e através do IP integrator, é possível efetuar a ligação rápida de vários IPs. Na criação de um IP é possível correr simulações para testar o bom funcionamento deste. Para tal, além do código desenvolvido para o IP, será também necessário desenvolver código para a realização do banco de ensaio (Test Bench), permitindo assim gerar sinais de entrada e averiguar se os sinais de saída estão de acordo com o expectável. Neste projeto foram utilizados vários IPs e muitos deles da biblioteca predefinida do Vivado, ficando assim com o modelo de blocos apresentado no Anexo 6.

Para conseguir analisar se o funcionamento do sistema CAR vai de encontro ao esperado, é necessário observar os sinais produzidos pelo sistema. Para tal, começou-se por efetuar a comunicação entre a FPGA e o ARM (ZYNQ), tirando partido dos GPIOs que são IPs da biblioteca do Vivado.

Para gerar os sinais de clock às frequências pretendidas utilizou-se o Clocking Wizzard, que também pertence a biblioteca do Vivado. Este só produz o clock para o DAC e para o ADC, uma vez que, como não consegue valores de frequência abaixo de 5MHz e como a frequência de amostragem do sistema CAR que se pretende utilizar é de 1Mhz, teve de ser criado um IP que recebe uma frequência elevada e a divide por um valor à escolha. Uma restrição deste IP é que os clocks de saída começam com uma frequência mais elevada à pretendida e só após alguns ciclos é que estabiliza para a frequência desejada. Portanto, não é possível utilizar a frequência máxima para as comunicações com o ADC, uma vez que os primeiros ciclos de relógio que estão a uma frequência mais elevada fazem com que o ADC não funcione corretamente.

A ferramenta Vitis é onde se programa o ARM com código em linguagem C para conseguir obter os sinais vindos da FPGA. Uma limitação desta via de comunicação é que, como o ARM trabalha a uma frequência de 33MHz e a FPGA a uma frequência diferente, nenhum dos dados recebidos no ARM vai estar exatamente à frequência da FPGA.

Para realizar o código em linguagem C utilizou-se uma biblioteca pré-definida, que contém todas as funções dos GPIOs, e por esse motivo, o código desenvolvido está apenas a correr um loop durante um valor de amostras fixo e, durante este loop, o ARM fica a retirar dados dos GPIOs e coloca na memória (DDR) da ZedBoard. Após a extração dos dados para a memória, através de uma funcionalidade do Vitis, esvaziam-se partes da memória para ficheiros binários, contendo cada um destes um sinal produzido pelo sistema CAR.

Por último, o software MatLab foi aplicado para análise de dados e visualização dos sinais, produzindo os gráficos obtidos nos capítulos seguintes.

8. IMPLEMENTAÇÃO PRELIMINAR

Numa fase inicial, uma implementação mais simples do que a apresentada na Figura 18 do sistema CAR em FPGA foi efetuada (Figura 25), de forma a entender a viabilidade do sistema, ou seja, se o erro introduzido pelos SDMs não interfere com a convergência dos coeficientes do CAR. Todos os sinais utilizados nesta implementação variam apenas de -1 a 1, sendo que o primeiro bit representa o sinal, e os restantes a parte fracionária em complemento para dois. Para obter os sinais externos, recorreu-se a uma simulação (Anexo 3) que consiste na criação de sinais de referência com 19 bits (LWL), e de sinais de erro que comportam a diferença entre o sinal desejado, que é o resultado da aplicação de um caminho primário simulado aos sinais de referência, e a saída do filtro FIR.

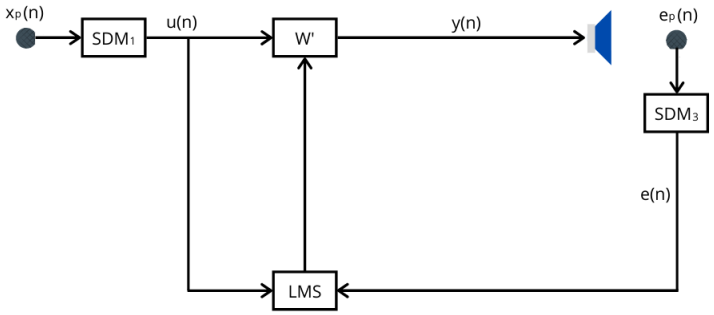


Figura 25 - Implementação Preliminar do CAR em FPGA

Os blocos SDM foram implementados utilizando o código desenvolvido em FPGA, com uma ordem de segundo grau (ver Anexo 1), pelo que são necessários apenas 2 registos de memória. Estes recebem sinais de entrada de 19 bits, e têm como resultado uma saída de 5 bits (SWL).

Na implementação preliminar do sistema CAR em FPGA (Figura 25), os coeficientes do filtro FIR são representados usando 19 bits e a dimensão do filtro é composta por 256 coeficientes, pelo que o resultado do sinal de referência após filtragem no FIR (y(n)) é de 24 bits, que consiste numa dimensão superior à requerida, e por isso numa elevada complexidade computacional. Esta questão será mitigada na implementação final, através da utilização de outro bloco SDM, que reduz o número de bits representativos do sinal w'(n), e consequentemente numa diminuição da complexidade computacional. Adicionalmente, para a atualização dos coeficientes do filtro FIR, foi aplicado o algoritmo LMS, com um tamanho de passo de valor 2^{-9}, representado por 10 bits para facilitar os cálculos desta fase.

8.1. RESULTADOS PRELIMINARES DE SIMULAÇÕES

Os resultados da implementação preliminar obtida a partir do desenvolvimento do código apresentado no Anexo 2, fornecem uma perspetiva positiva para a implementação final. Os dados conseguidos desta implementação foram extraídos para um ficheiro de texto, e posteriormente analisados utilizando um *script* desenvolvido em MatLab.

O caminho primário simulado foi constituído principalmente por um atraso de 128 amostras, uma atenuação de 0,8 e, para além disso, foi colocado ruído em torno de 0 para o resto do caminho. Como é possível observar na Figura 26, os coeficientes do filtro FIR convergem para os coeficientes reais do caminho primário. Assim, é possível afirmar que o sinal de erro tende para 0, e por isso existe cancelamento de ruído.

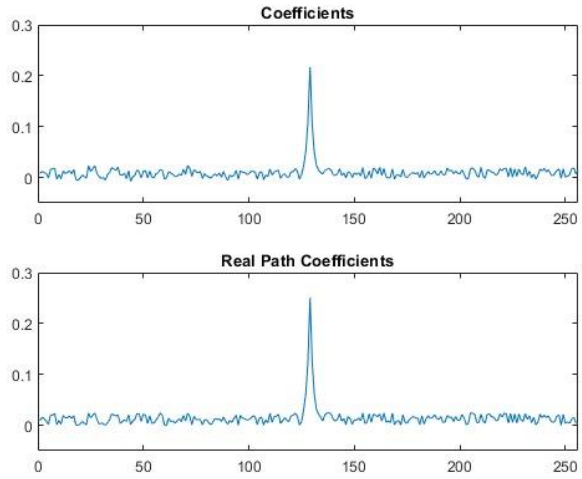


Figura 26 - Coeficientes Obtidos e Reais do Caminho Primário

Relativamente ao ruído introduzido pelo SDM no sistema, confirma-se que o ruído está presente principalmente nas altas frequências (Figura 27).

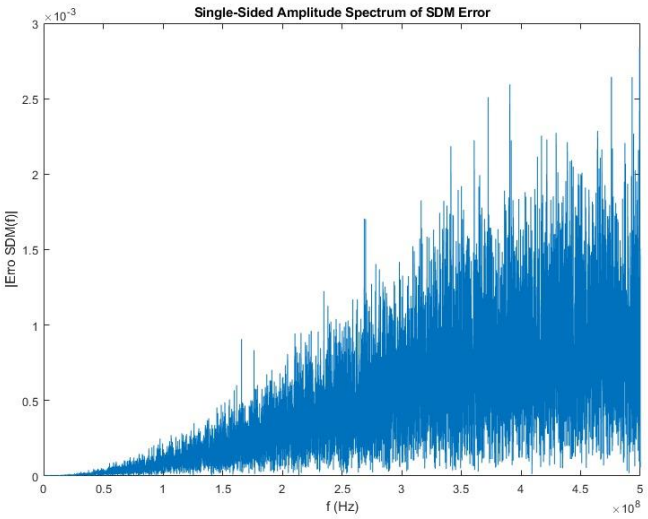


Figura 27 – Transformada de Fourier do Erro inserido pelo SDM

Com o ruído introduzido pelos SDMs, o sistema CAR também é afetado, mas o erro nos coeficientes do filtro também ocorre principalmente nas altas frequências (Figura 28). Uma vez que o ouvido humano capta apenas frequências de 16 Hz até 20 kHz [37], e que o erro só ocorre a partir de

aproximadamente $0{,}3 * f_s$ Hz para uma frequência de amostragem de f_s, é possível concluir que o erro não será audível.

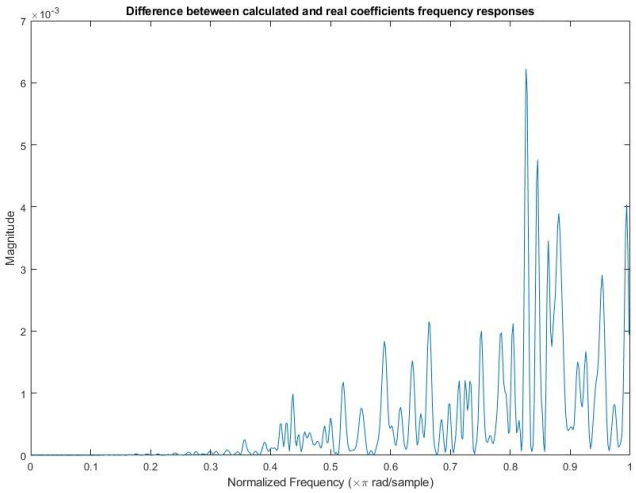


Figura 28 – Transformada de Fourier da Diferença da Resposta em Frequência dos Coeficientes Obtidos e Reais

9. RESULTADOS

Neste capítulo são apresentados os resultados de várias simulações e testes realizados no hardware para compreender as capacidades e as limitações do sistema. Todos os testes foram realizados com o ambiente o mais semelhante à realidade possível, com ruído ambiente e reflexos sonoros de vários objetos. Para estes testes, um auricular é suficiente para avaliar o desempenho do sistema, e para este caso foi utilizado o esquerdo, arbitrariamente.

Nas secções 9.1 e 9.2 serão realizados testes para a estimação do caminho secundário, e as restantes secções serão focadas no desempenho do sistema FxLMS com *feedback*. O sistema passou por um processo de mudança de vários parâmetros até se encontrar o que continha melhor performance. Para avaliar o desempenho do sistema, diferentes ruídos exteriores foram utilizados, primeiramente um único tom sinusoidal para várias frequências e, por último, um ruído branco gaussiano.

9.1. SIMULAÇÃO DA ESTIMAÇÃO DO CAMINHO SECUNDÁRIO

Para realizar a estimação do caminho secundário optou-se por usar o modo offline, como referido anteriormente. Assim, é importante avaliar o desempenho do sistema ao fazer este procedimento. Para tal, primeiramente, foi testado o algoritmo realizando simulações, com os seguintes parâmetros: $\mu_{sp} = 7 * 2^{-11}; L_{sp} = 128; F_s = 1MHz$.

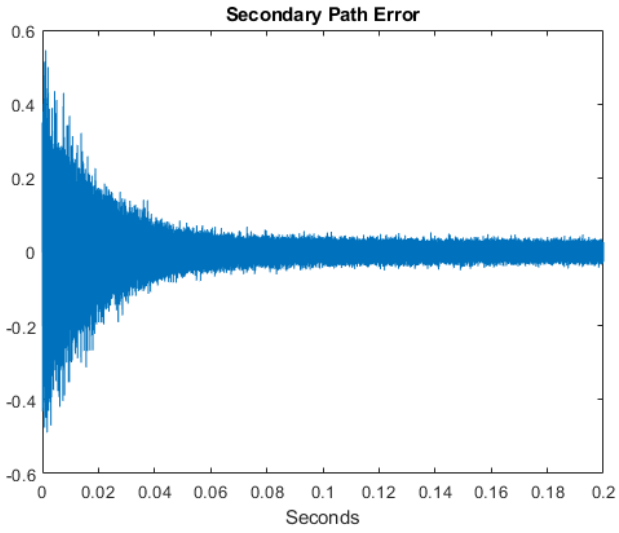


Figura 29 - Erro da estimação do caminho secundário simulado

Como se pode ver pela Figura 29, o erro converge para zero, o que significa que os coeficientes estão a convergir corretamente. No entanto, como estamos a utilizar um número finito de bits para cada palavra (16 bits) e como o tamanho do passo é relativamente elevado, vai existir bastante ruido residual. A escolha de um tamanho de passo tão elevado deve-se ao facto de que, se for muito pequeno, a convergência do sistema demora muito tempo, e como se está a testar o sistema numa simulação, o tempo que demora a convergir é muito superior ao real.

A estimação do caminho secundário não tem de corresponder exatamente ao real, mas o erro de fase não pode ser superior a 90º, pois caso contrário o sistema fica propenso a instabilidade [19]. Assim, para se avaliar se a estimação foi bem efetuada, realizou-se a diferença da resposta em frequência dos coeficientes reais da simulação e dos coeficientes estimados antes de serem arredondados (16 bits) e seguidamente com 5 bits, como se pode ver na Figura 30.

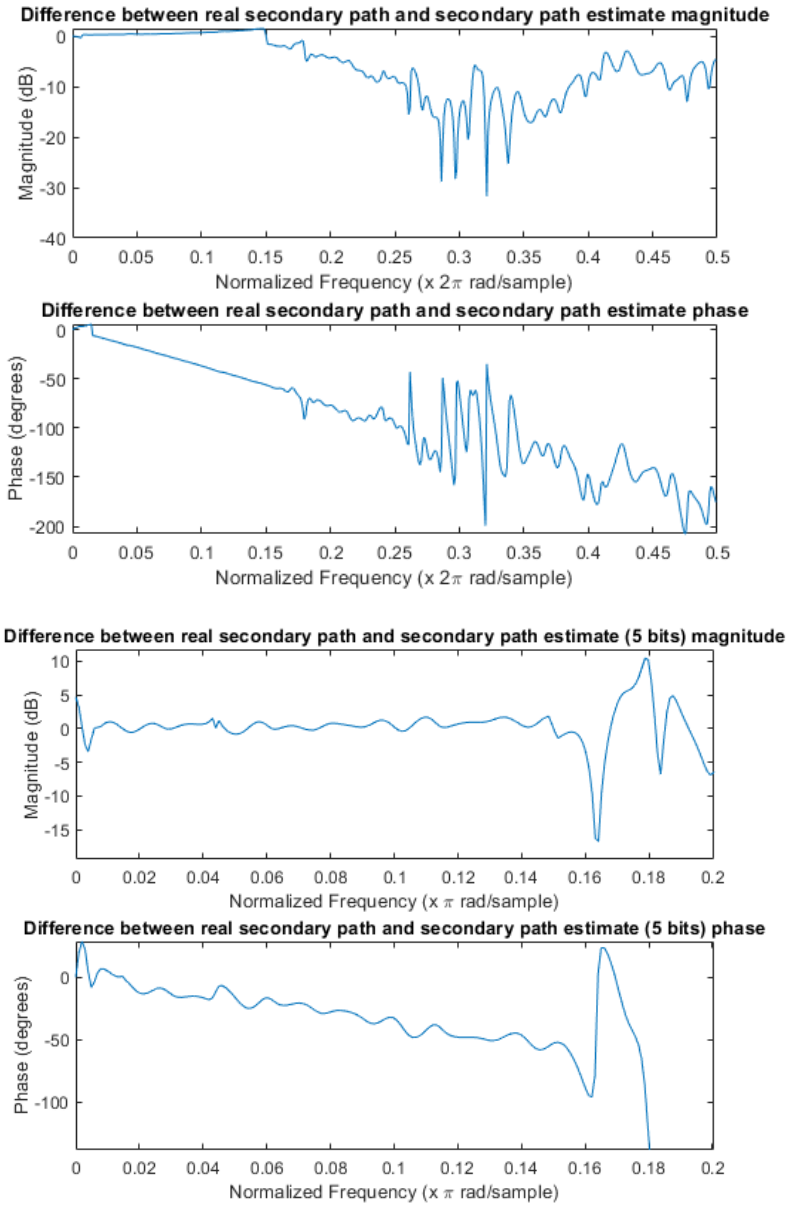


Figura 30 - resposta em frequência da diferença entre o caminho secundário real e o caminho secundário estimado para 16 bits e 5 bits respetivamente

Com estas diferenças é possível observar que a partir de aproximadamente $0.15\ F_s$, o erro da fase é superior a 90º para ambos os casos, o que significa que para o bom funcionamento do sistema FxLMS não se pode ultrapassar essa frequência. Em relação ao erro de magnitude, este apenas interfere no ritmo de convergência, pois quanto maior for o erro maior é o tempo de convergência.

9.2. Estimação do Caminho Secundário em Hardware

Para se ter uma ideia do caminho secundário que se pretende que o algoritmo estime, foi primeiramente feito um pequeno programa para apenas enviar para o auscultador ruído branco e para ler os valores do microfone. Assim, executou-se o algoritmo LMS em MatLab (palavras com 64 bits) para estimar o caminho secundário para duas frequências de amostragem diferentes (100KHz e 500KHzHz), e também com os auscultadores colocados fora da cabeça e na cabeça. Após a estimação ser feita, obteve-se a Figura 31 ,em que se observa os coeficientes estimados normalizados.

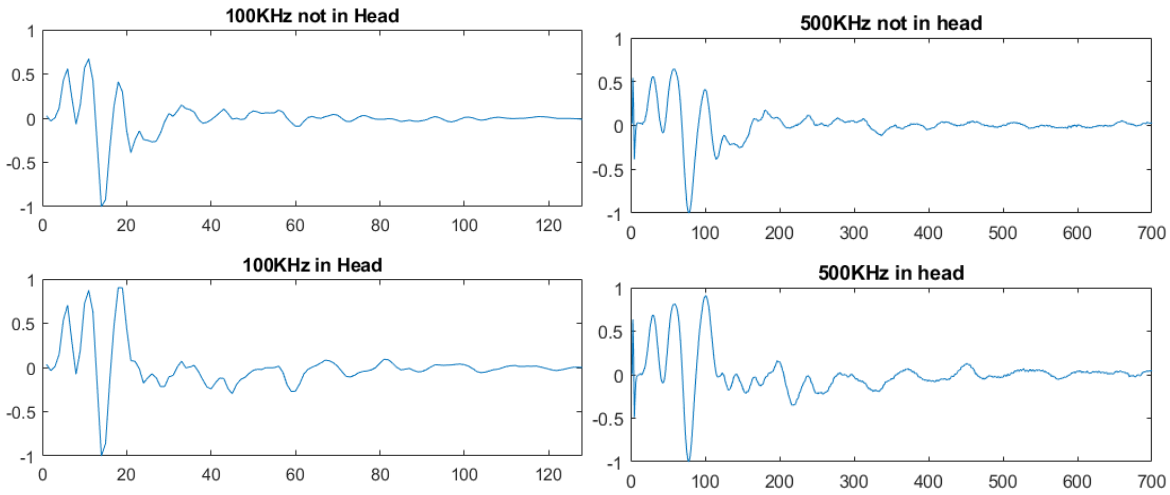


Figura 31 - Estimação do caminho secundário em MatLab para várias configurações

Como se pode observar, os coeficientes não variam muito para os testes efetuados. A grande diferença é que para as frequências de amostragem mais elevadas é necessário um número elevado de coeficientes, mas em contrapartida esta estimação tem uma maior resolução.

Após se ter uma ideia dos valores dos coeficientes do caminho secundário, testou-se o algoritmo desenvolvido em VHDL para a estimação do caminho secundário. Com isto, observou-se que o caminho secundário estimado (5 bits) era constituído maioritariamente por zeros, o que não correspondia ao expectável Figura 31. Este problema deve-se ao facto de os coeficientes de 16 bits terem valores muito reduzidos e ao arredondar para 5 bits ficarem a zero. Como o caminho secundário é controlado pelos ganhos do auscultador e do microfone, estes foram aumentados, e consequentemente obtiveram coeficientes com valores mais elevados, o que mesmo assim não foi suficiente. Outra forma utilizada para que os coeficientes tivessem valores mais elevados foi multiplicar o sinal recebido do microfone por 4, assim os coeficientes ficam 4 vezes maiores do que o caminho secundário real, o que tem de ser compensado ao utilizar esta estimação no sistema FxLMS. Outra questão que teve de se ter em atenção foi em relação ao ganho do caminho secundário, no qual teve de se ir alterando o ganho do auscultador e do microfone para que o ganho do caminho secundário esteja o mais perto de 0dB, para que os sinais no sistema FxLMS não saturem.

Neste caso, não é possível colocar todos os ganhos do caminho secundário perto de zero uma vez que existe grande disparidade entre eles. Assim ao aumentar o ganho do microfone ou do altifalante, todos os ganhos iriam aumentar o que significa que para esta configuração não é possível colocar todos os ganhos a 0dB.

Para avaliar o desempenho da estimação do caminho secundário, foram realizadas 10 estimações para visualizar a variação na resposta em frequência das estimativas. Para estes testes, os parâmetros utilizados foram os seguintes: $\mu_{sp} = 2^{-11}; L_{sp} = 128; F_s = 100KHz$. Foi utilizada uma frequência de amostragem mais baixa que o pretendido, uma vez que para uma frequência maior teria de se ter um maior numero de coeficientes e a ZedBoard não tem recursos suficientes para ter mais que 128 coeficientes em conjunto com o sistema FxLMS. Como a frequência de amostragem é mais baixa que o suposto, então o número de amostras para estimar o caminho secundário foi reduzido para 3M de amostras, demorando assim aproximadamente 30 segundos cada teste.

Para calcular a variação na resposta em frequência das estimativas, pode assumir-se que as variações das estimativas se devem a mudanças lentas no caminho secundário, pelo que a diferença máxima entre as fases das estimativas é uma boa aproximação do erro de fase. Este erro não deve ultrapassar os 90º, pois se passar significa que o sistema para a dada frequência pode ficar instável, como referido anteriormente.

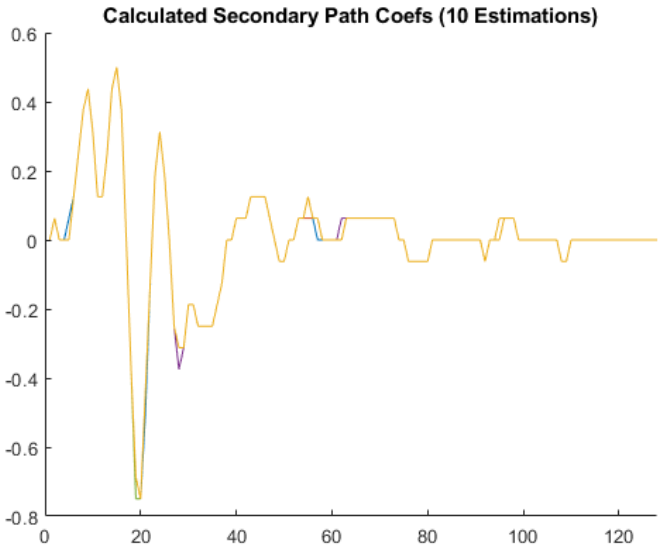


Figura 32 - Várias estimações do caminho secundário em *Hardware*

Como se pode ver pela Figura 32, os coeficientes das estimações têm poucas variações. No entanto, uma pequena diferença nos coeficientes é suficiente para criar uma desigualdade na resposta em frequência.

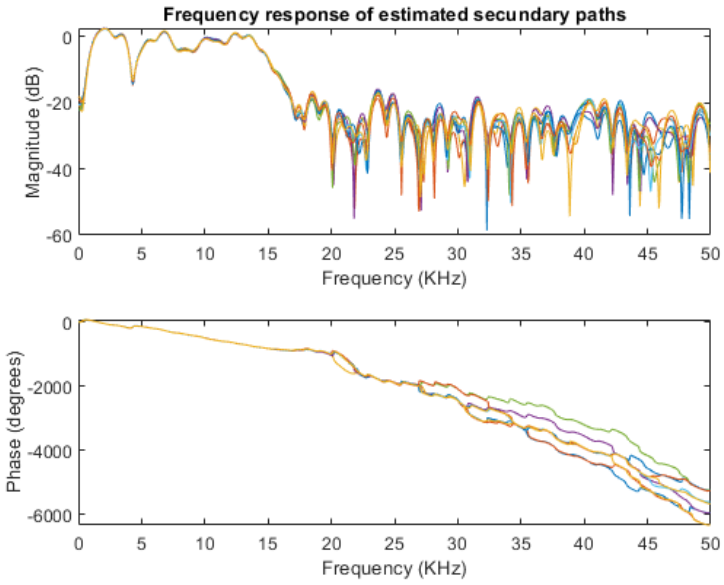


Figura 33 - Resposta em frequência das várias estimações do caminho secundário em *Hardware*

Na Figura 33 é possível observar a resposta em frequência das várias estimações, e conclui-se que a maior variação é nas altas frequências. Para obter uma melhor perceção de quando existe um grande erro na magnitude e um erro de fase maior que 90º foi realizada a diferença máxima entre as respostas em frequência das estimativas Figura 34.

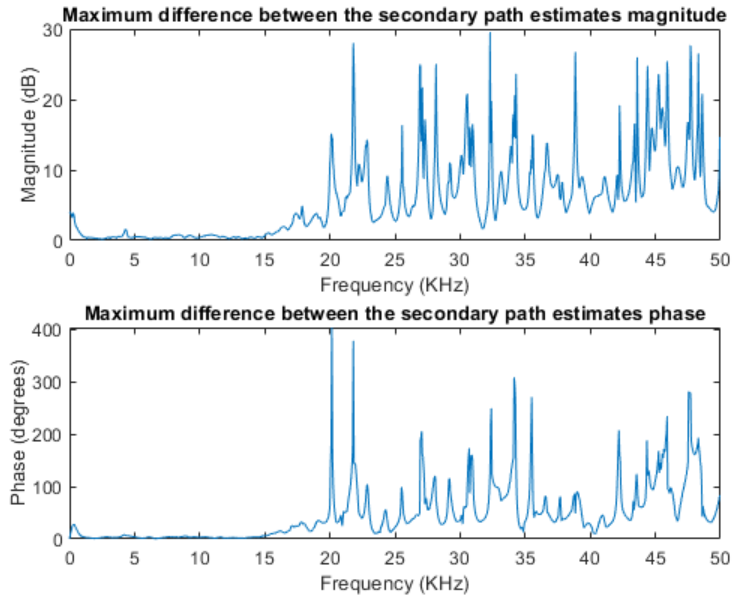


Figura 34 - Diferença máxima entre as respostas em frequência das estimativas do caminho secundário

Com esta diferença é possível concluir que a partir de 20KHz o sistema pode ter um erro de fase maior que 90º. Como visto anteriormente nas simulações, o erro que a estimação provoca é maioritariamente a partir de $0.15 * F_s = 15KHz$, para $F_s = 100KHz$, pelo que é possível concluir com grande certeza que a partir de 20KHz o sistema pode ficar instável.

9.3. SIMULAÇÃO DO ALGORITMO FXLMS FEEDBACK

Para se ter uma ideia do comportamento do sistema, foram efetuadas algumas simulações. Para se obter resultados o mais próximo do real utilizou-se o caminho secundário previamente estimado, e também se adicionou ruído branco no sinal de erro, como é espectável num ambiente real.

O esperado para os coeficientes do filtro W' é que tenham uma resposta em frequência inversa em comparação com o caminho secundário, isto é, $\boldsymbol{W'}(s) = \frac{1}{\boldsymbol{S}(s)}$, uma vez que o sinal de referência é visto no microfone de erro, assim o que o altifalante tem de enviar é o sinal de referência após passar pelo inverso do caminho secundário.

Como referido anteriormente, o ganho ideal do caminho secundário para todas as frequências deveria de ser o mais perto de 0db, no entanto para esta configuração não é possível. Assim, para o caminho secundário estimado anteriormente, fontes de ruído a produzir sinais com frequências abaixo de 1KHz e acima de 15KHz irão causar que o filtro W' tenham ganhos elevados para essas frequências. Para uma fonte de ruido a produzir uma sinusoide de 100Hz, o espectável do filtro W' é que tenha um ganho de aproximadamente 20db (x10 linear), e consequentemente, o sinal de referência não pode ter uma amplitude maior que 0.1, uma vez que ao passar pelo filtro W' ficaria com amplitude superior a 1, e não é possível visto que este sistema apenas trabalha com valores entre -1 e 1.

Outra situação, que neste caso não ocorre, seria se o caminho secundário tivesse um ganho elevado, assim o sinal $u^2(n)$ da Figura 19 poderia saturar se o sinal de referência for elevado, o mesmo acontece para o sinal $g(n)$.

Nas simulações sequentes irão se utilizar os seguintes parâmetros: $\mu_{pp} = 15 * 2^{-11}; L_{pp} = 64;\ \varphi_{pp} = 2^{-15}; F_s = 100KHz$ (μ_{pp}:Tamanho do passo de FxLMS, L_{pp}: Número de coeficientes de W', φ_{pp}: Fator Leakage de FxLMS). Para uma fonte de ruido a produzir uma sinusoide com frequência de 100Hz o sistema apresenta o comportamento da Figura 35.

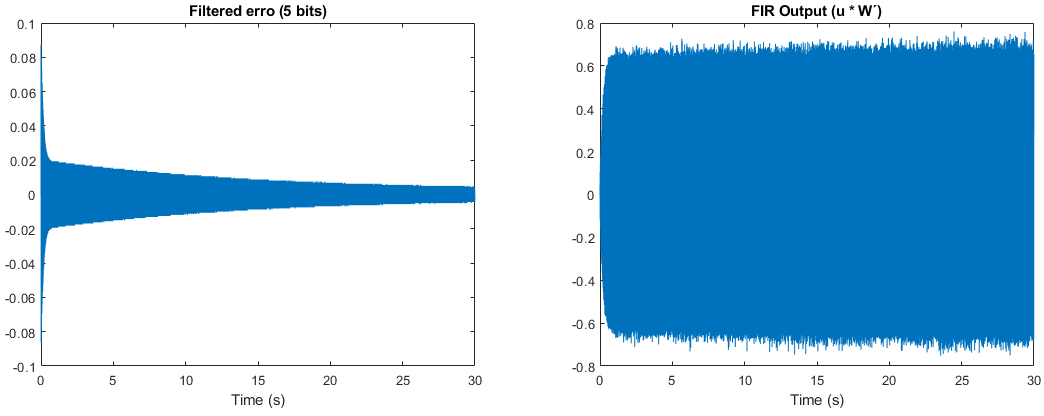


Figura 35 - Sinal de erro e sinal da saída do filtro W' do sistema FxLMS *feedback* para um ruído sinusoidal de 100Hz

Nesta simulação foi utilizada a sinusoide com 0.1 de amplitude. Como se pode observar, o tempo que demora a convergir é maior que 30 segundos uma vez que ainda não tinha convergido totalmente. Isto deve-se ao facto de o sinal de referência ser muito pequeno, no entanto, não se pode aumentar muito mais que assim a saída do filtro W' satura, como referido anteriormente.

Para uma fonte de ruido a produzir uma sinusoide com frequência de 3KHz, o sistema converge muito mais rapidamente como se pode observar na Figura 36.

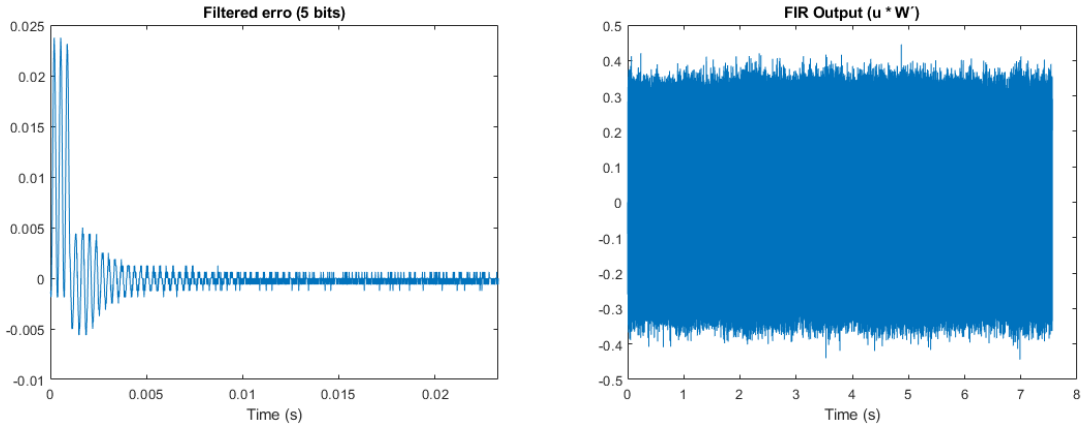


Figura 36 - Sinal de erro e sinal da saída do filtro W' do sistema FxLMS feedback para um ruído sinusoidal de 3KHz

Com estes resultados é possível observar que o sistema converge em apenas 0.01 segundos para um sinal de referência de 0.25 de amplitude. Neste caso poder-se-ia aumentar mais o sinal de referência uma vez que a saída do filtro W' não esta no limiar de saturação.

Para esta simulação a convergência foi rápida uma vez que o caminho secundário para esta frequência tem ganho aproximadamente 0db, o que significa que basicamente basta inverter o sinal de referência e enviar para o altifalante para que exista cancelamento quase total.

Os sinais de erro observados nas figuras anteriores foram filtrados por um filtro passa baixo para se conseguir observar a convergência uma vez que existe erro nas altas frequências.

É expetável que num cenário real o sistema não tenha o mesmo desempenho que nas simulações uma vez que existem outros fatores em consideração como ruído ambiente, erros de leitura do ADC, entre outros.

9.4. ALGORITMO FXLMS FEEDBACK EM HARDWARE

Como análise preliminar da performance do sistema, começou-se por observar o ruído ambiente que é captado pelo microfone de erro (Figura 37), de forma a examinar possíveis anomalias.

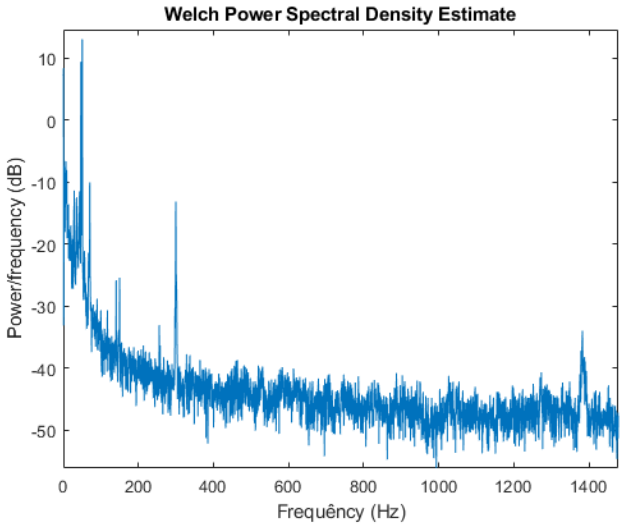


Figura 37 - Ruído ambiente captado pelo microfone de erro

Como se pode ver o ruído ambiente está maioritariamente nas baixas frequências e apresenta dois picos, um a 300Hz com potência de -13dB, correspondente à ventoinha do computador, e outro a 50Hz com potência de 13 dB que advém da tensão da rede do edifício que funciona a 50Hz e causa interferência eletromagnética no sistema. Esta interferência pode causar problemas no sistema, uma vez que não é possível cancelar com sinais sonoros.

Assim, para reduzir esta interferência, a frequência de corte do filtro passa alto anteriormente desenhado foi alterada para 88Hz, filtrando assim grande parte da interferência como se pode ver na Figura 38.

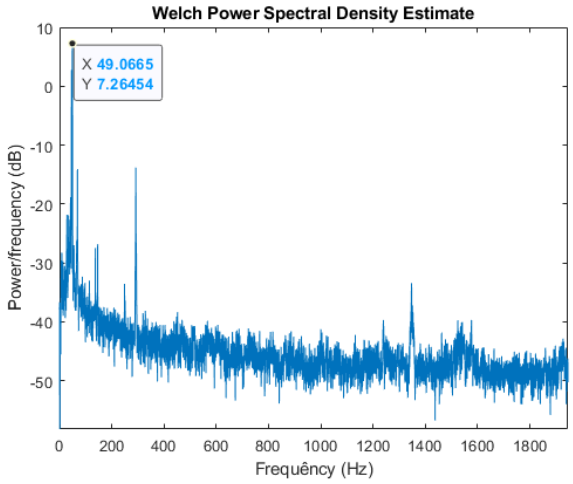


Figura 38 - Ruído ambiente que é captado pelo microfone de erro depois da alteração do filtro

A interferência visualizada não está exatamente nos 50Hz, porque teve de se efetuar uma estimativa da frequência de amostragem do ARM, como referido anteriormente, este trabalha a uma frequência de 33MH. No entanto, não é sabido quantos ciclos de relógio são necessários para extrair todas as amostras vindas do sistema FxLMS. Para se ter uma noção desta frequência de

amostragem, colocaram-se várias sinusoides no microfone de erro com frequência conhecida e estimou-se uma frequência de amostragem de aproximadamente 210KHz.

Para se conseguir obter resultados quantitativos do desempenho do sistema FxLMS foi utilizado um interruptor da ZedBoard para desativar e ativar o sistema sempre que for conveniente. Assim, é possível medir o sinal recebido no microfone de erro sem que o sistema esteja ativo, conseguindo desta forma comparar os valores e obter a quantidade de cancelamento. Adicionalmente, foi inserida uma propriedade em que com a desativação do sistema, os coeficientes de W' colocam-se todos a zero. Esta particularidade foi bastante útil para obter resultados, uma vez que quando o sistema ficava instável tinha de se reiniciar o sistema e calcular de novo os coeficientes do caminho secundário, e desta forma basta apenas alternar o interruptor duas vezes.

Os parâmetros utilizados para os resultados apresentados de seguida foram:

- SWL (C bits): 5 bits
- LWL (R bits): 16 bits
- SDM: 2º ordem
- Dimensão LP: 4
- Dimensão W: 128 coeficientes
- Dimensão $\hat{S}$: 128 coeficientes
- Número de amostras para estimação de S: 10 Mil
- Valor de Leakage: 2^{-15}
- Tamanho do passo na estimação de S: $5 * 2^{-11}$
- Tamanho do passo do sistema FxLMS: $3 * 2^{-11}$
- Frequência de amostragem: 100KHz

Para estes parâmetros, a ZedBoard tem uma utilização de recursos de 72%, o que indica que ainda existem recursos disponíveis para utilizar no sistema. No entanto, como foi referido anteriormente, a partir de cerca de 80% de recursos utilizados estes já não conseguem ser colocados na ZedBoard. E por este motivo, não se colocou uma frequência de amostragem mais elevada, pois seria necessário aumentar a dimensão de $\hat{S}$ para que o caminho secundário fosse bem estimado, e os recursos utilizados não caberiam na ZedBoard.

O número de amostras e o tamanho do passo na estimação do caminho secundário foram mais elevados do que o esperado, uma vez que, feitos alguns testes, observou-se que com as configurações utilizadas em 9.2, o caminho poderia ficar mal estimado, devido a excesso de ruído ambiente. Assim, aumentou-se não só o número de amostras, demorando assim 100 segundos para a estimação, mas também o tamanho do passo, para que convirja mais rapidamente.

Ao ligar o sistema, a estimação do caminho secundário é a primeira rotina a ser corrida, e se esta for bem realizada o sistema inicia o FxLMS. Como o auxílio de um altifalante auxiliar foi gerada uma

sinusoide com uma frequência de 3KHz e observou-se o sinal de erro para ver se o sistema estava a convergir corretamente.

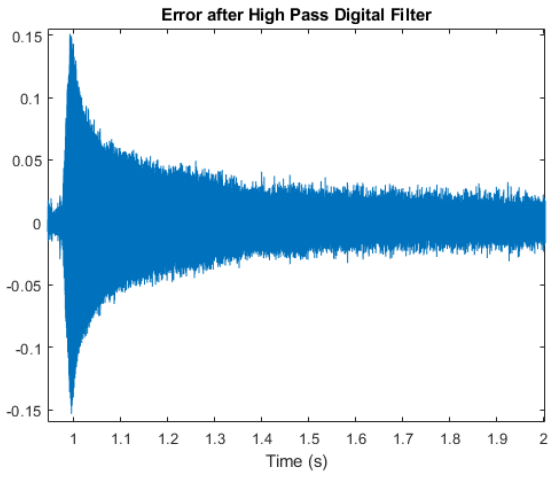


Figura 39 - Sinal de erro do sistema FxLMS *feedback* para um ruído sinusoidal de 3KHz

Assim, observando a Figura 39, é possível concluir que o sistema está a realizar cancelamento de ruído. Demorou cerca de 1 segundo para que convergisse totalmente, restando apenas ruído de alta frequência, na qual não é audível.

Para um ruído sinusoidal, o espectável para os coeficientes do filtro W é que também sejam uma sinusoide, no entanto, esta sinusoide vai conter bastante ruído de alta frequência uma vez que o sistema no geral tem bastante erro nas altas frequências.

Após se verificar que o sistema estava a funcionar corretamente fez se uma avaliação da quantidade de cancelamento de ruído sinusoidal para várias frequências, para tal, é necessário realizar a diferença da potencia do sinal na frequência do ruido quando o sistema está desativado e ativado. Começou se por analisar as baixas frequências.

Ao realizar estas medições observou-se que o sistema saturava muito facilmente com ruídos de potência elevada. Faz sentido tal acontecer, uma vez que como visto em 9.2, o sistema tem um caminho secundário que nas baixas frequências tem grande atenuação, então, o filtro W' tem de ter um ganho elevado para estas frequências. Mas mesmo para sinais com potência baixa saturava, e após alguma análise percebeu-se que o problema era derivado da amplificação não só do sinal a se querer cancelar, mas também da interferência eletromagnética (50Hz) e ambos os sinais em conjunto era suficiente para saturar a saída do filtro W' (Figura 40).

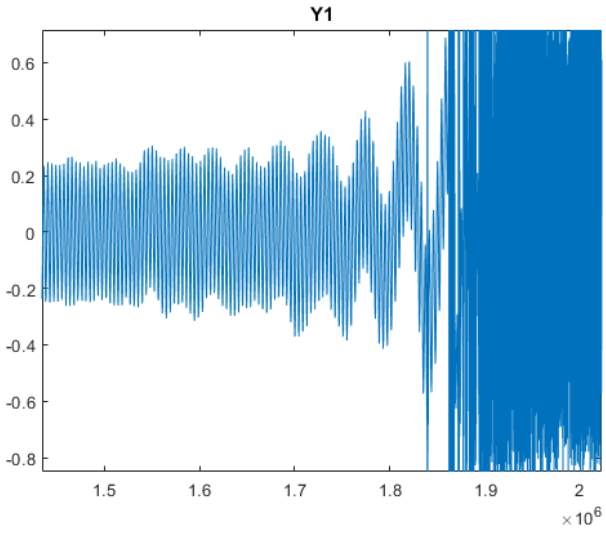


Figura 40 - Saída do filtro W' a entrar em saturar

Para ultrapassar este problema teve de se diminuir o ganho que o filtro W' ficava para as baixas frequências, para tal foi necessário alterar os ganhos do caminho secundário. Assim teve de se diminuir a multiplicação do sinal de erro para 2 em vez de 4, como especificado em 9.2, para se conseguir aumentar o ganho do microfone e do altifalante sem que os coeficientes saturassem. Obteve-se assim a nova resposta em frequência da estimação do caminho secundário na Figura 41.

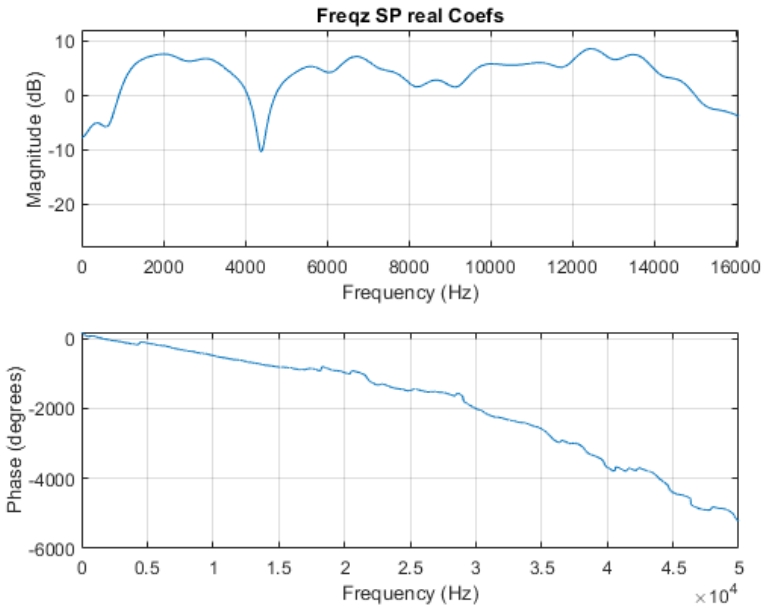


Figura 41 - Resposta em frequência da estimação do caminho secundário

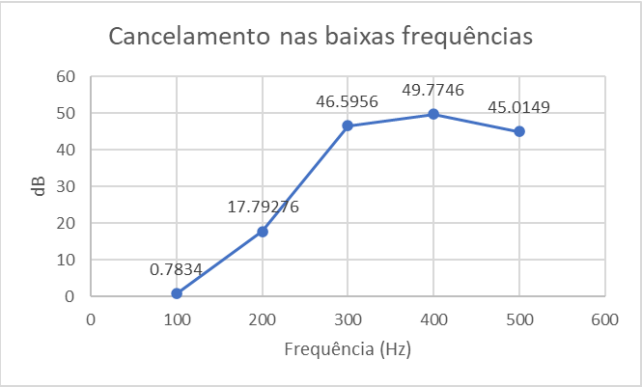


Figura 42 - Desempenho do sistema para ruídos de baixa frequência

Observando a Figura 42, é possível concluir que o sistema funciona com uma maior eficácia para ruídos com frequência superior a 300Hz, isto pode ser devido ao facto de que como se está a utilizar um filtro passa alto, para filtrar o sinal vindo do microfone, com frequência de corte de 88Hz, o caminho secundário para as baixas frequências fique mal estimado devido a complexidade da estimação de um filtro passa alto.

Para frequências superiores a 300Hz é possível verificar que o sistema consegue cancelar mais de 45dB, e olhando para o espetro do sinal de erro é possível verificar que o ruído foi totalmente cancelado na frequência da sinusoide, uma vez que nem existe rasto da existência da dada frequência. Todas as medições dos valores de cancelamento foi apenas às dadas frequências e não do ruído total.

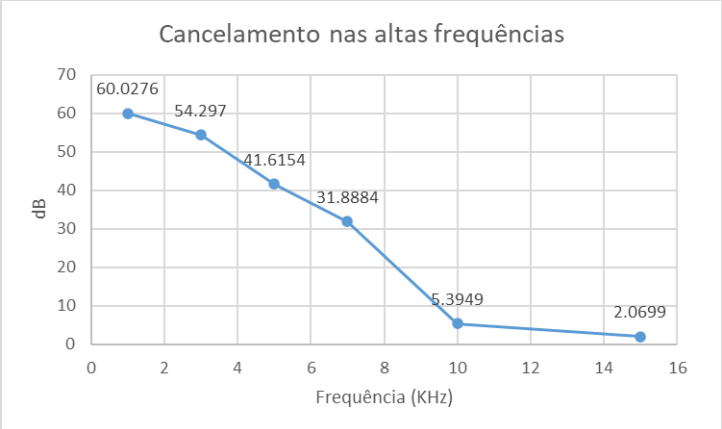


Figura 43 - Desempenho do sistema para ruídos de alta frequência

Para se verificar até que frequências o sistema tinha um bom desempenho foi efetuado a Figura 43 que indica quanto decibéis o sistema consegue cancelar a uma dada frequência. Assim, é possível concluir que o sistema tem uma boa performance até a uma frequência de 7KHz. O que indica que o sistema consegue cancelar com uma boa performance de 300Hz até 7KHz. O expectável seria o sistema conseguir cancelar até 20KHz, no entanto, como a frequência de amostragem está a apenas 100KHz em vez de 1.4MHz que era a frequência de amostragem originalmente pretendida, então os SDMs têm um maior ruido de quantização, uma vez que quanto maior a frequência de trabalho do

SDM menor é o ruido de quantização nas baixas frequências. Outra hipótese para que o sistema a partir dos 10KHz já não conseguir cancelar quase nada seria devido à má estimação do caminho secundário, como visto nas simulações do tópico 9.1, a partir de 0.15*Fs o erro de fase é superior a 90º o que pode indicar instabilidade, o que neste caso não ocorre, mas como já existe grande erro não consegue cancelar.

Como visto, o sistema consegue-se cancelar com bastante eficácia sinusoides de frequências a variar entre os 300Hz e os 7KHz, no entanto, não se sabe qual o comportamento do sistema para ruido complexos nomeadamente para ruido branco. Portanto, colocou-se o altifalante a produzir ruído brando e observou-se o comportamento do sinal de erro com o sistema desativado e ativado, como se pode ver na Figura 44.

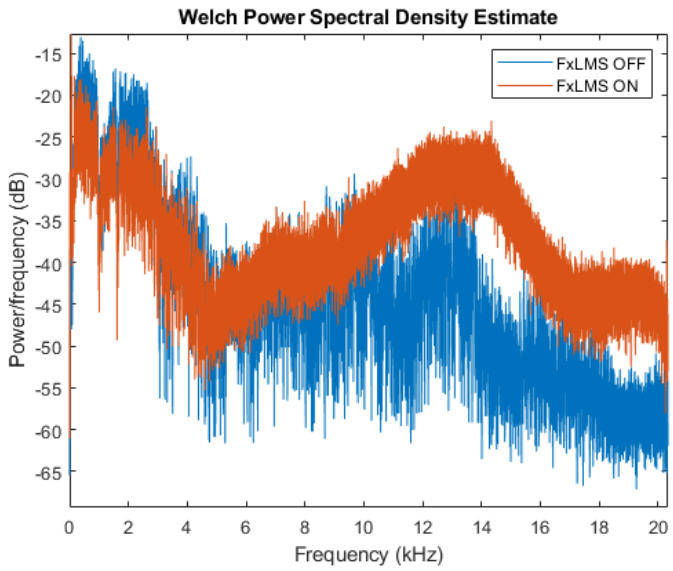


Figura 44 - Densidade espectral de potência do ruído branco com o sistema ativado e desativado

Ao analisar a figura que contem o espetro do sinal com o sistema desativo e ativado, observa-se que a partir de aproximadamente 5KHz o sinal de ruido é superior quando o sistema está ativado, no entanto, para frequências mais baixas o sistema consegue diminuir o sinal de ruído. Para frequências mais elevadas a 20KHz, frequência máxima audível, o sistema piora substancialmente o sinal, o que não há problema uma vez que para essas frequências o som já não é captado pelo ouvido humano.

Como a Figura 44, apresenta valores numa escala logarítmica (dB) não é possível ter grande perceção se o sistema dentro da banda audível se diminui o ruído ou aumenta, assim calculando a variância de ambos os sinais, obtém-se que o sistema diminui por aproximadamente 3% a potência do ruído, o que não é muito, mas era o espectável para o sistema FxLMS com *feedback*.

Foram feitos testes para frequências de amostragem mais elevadas, mas mesmo para frequências de 200KHz a estimação do caminho secundário ficava com demasiados erros devido a número baixo de coeficientes tendo assim uma pior performance do que frequência de amostragem a 100KHz.

10. CONCLUSÃO

O objetivo deste projeto era implementar um sistema de auscultadores CAR sobre amostrado com uma placa de desenvolvimento FPGA, a ZedBoard alojando uma FPGA e um ARM com dois *cores*. Foi utilizado também um circuito analógico para amplificar os sinais de entrada/saída de/para os conversores AD e DA, que por sua vez geram sinais de/para a ZedBoard. Este sistema foi então utilizado para experimentar várias configurações de sistema, a fim de alcançar uma boa atenuação, estabilidade e robustez.

Em primeiro lugar, efetuaram-se as comunicações entre os conversores e a ZedBoard e o seu *software* foi desenvolvido. Durante este processo, o conhecimento dos sistemas eletrónicos e protocolos de comunicação foram reforçados com estes aspetos práticos. Com o desenvolvimento de software, foi obtida uma nova perspetiva sobre o modo de funcionamento da FPGA. Os algoritmos não apresentaram quaisquer problemas de software, tendo sido por isso bem implementados.

O FxLMS foi o único algoritmo a ser desenvolvido. O sistema de auscultadores com cancelamento ativo de ruído, com configurações de feedback, mostrou uma atenuação satisfatória. No entanto, teoricamente, os sistemas *feedforward* teriam um melhor desempenho do que os sistemas de *feedback*, mas mesmo assim ambos os sistemas têm uma melhor atenuação em comparação com alguns controladores de estrutura fixa. Contudo, o sistema revelou problemas de instabilidade relacionados com o erro de modelação das estimativas do caminho secundário e a saturação de sinais.

Para mitigar estes problemas teria sido útil a utilização de um controlador de ganho automático (AGC). Para o microfone, a vantagem seria colocar o ganho do caminho secundário uniforme e o mais perto de 0dB. Assim, seria possível reduzir os níveis de som que o altifalante necessitava para ter o ganho anteriormente descrito, o que tornaria possível testar o sistema colocado nos ouvidos e não apenas colocados em cima de uma mesa. Em relação às saturações dos sinais, um AGC digital iria beneficiar o sistema colocando sempre o sinal de referência para qualquer frequência a uma determinada amplitude. Desta forma, com estas duas melhorias, o sistema com o algoritmo FxLMS iria ter aproximadamente a mesma amplitude em todos os sinais, exceto o sinal de erro.

Como trabalho futuro, é recomendado testar este sistema para outros algoritmos, de forma a perceber se a sua performance é melhorada. Adicionalmente, sugere-se a utilização de um maior número de recursos da FPGA, para que seja possível aumentar a frequência de amostragem, tendo um número de coeficientes mais elevado para a estimação do caminho secundário. Uma forma de melhorar a estabilidade do sistema, seria através do aumento do número de bits dos coeficientes do caminho secundário, pois isso faria com que a resolução da estimação fosse mais apurada.

No caso de ser dada continuidade a este projeto, aconselha-se como passo seguinte desenhar uma placa de circuito impresso (PCB), com todos os elementos que foram utilizados, de forma a reduzir as interferências não só do ambiente, mas também entre componentes. Posteriormente, seria ideal a utilização de um sinal de referência binário, isto é, que apresenta apenas valores de 1 ou -1,

para a modelação do caminho secundário, permitindo assim poupar recursos, pois não existiria necessidade de realizar multiplicações, e também a possibilidade de colocar filtros mais compridos. Com o objetivo de economizar ainda mais hardware, sem perder performance, o número de bits dos sinais poderia ser reduzido a apenas 2 bits, mantendo a dimensão dos coeficientes dos filtros com 5 bit.

Os resultados obtidos indicaram que este sistema de auscultadores CAR pode ser considerado para comercialização. No entanto, os desempenhos de estabilidade têm de ser melhorados antes de chegarem ao utilizador final. Neste projeto, foi criado um protótipo, que necessita de alguma investigação futura e melhorias para poder ser vendido no mercado utilizando toda a sua potencialidade ao menor custo possível.

11. BIBLIOGRAFIA

[1] S. J. Elliott, “Down with noise [active noise control],” *IEEE spectrum,* vol. 36, nº 6, pp. 54-61, 1999.

[2] C. Y. Chang, A. Siswanto, C. Y. Ho, T. K. Yeh, Y. R. Chen e S. M. Kuo, “Listening in a noisy environment,” *IEEE Consumer Eletronics Magazine,* vol. 5, nº 4, pp. 34-43, 2016.

[3] S. M. Kuo e D. R. Morgan, “Active noise control: a tutorial review,” *Proceedings of the IEEE ,* vol. 87, nº 6, pp. 943-973, 1999.

[4] A. Zaknich, Principles of adaptive filters and self-learning systems, Springer Science & Business Media, 2005.

[5] B. Kovacevic, Z. Banjac e M. Milosavljevic, Adaptive Digital Filters, Springer-Verlag, 2013.

[6] S. Haykin, Adaptive Filter Theory, 3ª ed., Prentice-Hall, 1996.

[7] P. W. Wong e R. M. Gray, “FIR Filters with Sigma-Delta Modulation Encoding,” *IEEE Transactions on Acoustics, Speech, and Signal Processing,* vol. 38, nº 6, pp. 979-990, 1990.

[8] L. Litwin, “FIR and IIR digital filters,” *IEEE Potentials,* vol. 19, nº 4, pp. 28-31, 2000.

[9] V. K. Madisetti e D. B. Williams, Digital Signal Processing Handbook, Atlanta, Georgia: CRC Press, 1999.

[10] P. S. R. Diniz, Adaptive Filtering Algorithms and Practical Implementation, Rio de Janeiro, Brazil: Springer, 2013.

[11] B. Widrow e S. D. Stearns, Adaptive Signal Processing, New Jersey: Prentice-Hall, 1985.

[12] S. S. Haykin, Adaptive Filter Theory, Prentice Hall, 2002.

[13] P. A. C. Lopes, G. Tavares e J. B. Gerald, “A NEW TYPE OF NORMALIZED LMS ALGORITHM BASED ON THE KALMAN FILTER,” *IEEE Xplore,* vol. 3, p. 4, 2007.

[14] D. T. M. Slock, “On the Convergence Behavior of the LMS and the Normalized LMS Algorithms,” *IEEE TRANSACTIONS ON SIGNAL PROCESSING,* vol. 41, nº 9, p. 15, 1993.

[15] K. Mayyas e T. Aboulnasr, “Leaky LMS algorithm: MSE analysis for Gaussian data,” *IEEE Transactions on Signal Processing,* vol. 45, nº 4, pp. 927 - 934, 1997.

[16] H. C. Woo, “Variable Step Size LMS Algorithm using Squared Error and Autocorrelation of Error,” *Procedia ENgineering,* vol. 41, pp. 47-52, 2012.

[17] R. H. Kwong e J. E. W., “A varible step size LMS algorithm,” *IEEE Transactions on Signal Processing,* vol. 40, pp. 1633-1642, 1992.

[18] S. D. Snyder, Active Noise Control Primer, South Australia: Springer, 2000.

[19] S. M. Kuo e D. R. Morgan, Active Noise Control Systems: Algorithms and DSP Implementations, New York: Wiley-Interscience, 1996.

[20] P. A. C. Lopes e J. A. B. Gerald, “Low Delay Short Word Length Sigma Delta Active Noise Control,” *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS,* vol. 68, nº 9, 2021.

[21] R. Schreier e G. C. Temes, Understanding delta-sigma data converters, IEEE Press, 2004.

[22] T. Memon, P. Beckett e A. Z. Sadik, “Sigma-Delta Modulation Based Digital Filter Design Techniques in FPGA,” *ISRN Eletronics,* vol. 2012, pp. 1-10, 2012.

[23] T. Memon, P. Beckett e Z. M. Hussain, “Design and Implementation of a Ternary FIR Filter using Sigma Delta Modulation,” *International Symposium on Computing, Communication, and Control,* vol. 1, 2009.

[24] X. Wu e R. Goodall, “One-bit processing for real-time control,” *European Control Conference,* pp. 3347-3352, 2003.

[25] A. C. Thompson, P. O'Shea, Z. M. Hussain e B. R. Steele, “Efficient single-bit ternary digital filtering using sigma-delta modulator,” *IEEE Signal Processing Letters,* vol. 11, nº 2, pp. 164-166, 2004.

[26] A. Z. Sadik e Z. M. Hussain, “Short word-length LMS filtering,” *International Symposium on Signal Processing and Its Applications,* pp. 1-4, 2007.

[27] D. A. Johns e D. M. Lewis, “Design and analysis of delta-sigma based IIR filters,” *IEEE Transactions on Circuirs and Systems II: Analog and Digital Signal Processing,* vol. 40, nº 4, pp. 233-240, 1993.

[28] S. S. Abeysekera e K. P. Padhi, “Design of multiplier free FIR filters using a LADF sigma-delta modulator,” *IEEE International Symposium on Circuits and Systems,* vol. 2, pp. 65-68, 2000.

[29] C.-L. Chen e A. N. Wilson, “Higher order sigma-delta modulaiton encoding for the design of efficient multiplierless FIR filters with powers-of-two coefficients,” *IEEE International Symposium on Circuits and Systems,* vol. 4, pp. 2361-2364, 1997.

[30] C. Dick e F. Harris, “FPGA signal processing using sigma-delta modulation,” *IEEE Signal Processing Magazine,* vol. 17, nº 1, pp. 20-35, 2000.

[31] P. N. Samarasinghe, W. Zhang e T. D. Abhayapala, “Recent Advances in Active Noise Control Inside Automobile Cabins: Toward quieter cars,” *IEEE Signal Processing Magazine,* vol. 33, nº 6, pp. 61 - 73, 2016.

[32] K. Mazur, S. Wrona e M. Pawelczyk, “Active noise control for a washing machine,” *Applied Acoustics,* vol. 146, pp. 89-95, 2019.

[33] W. Niu, “Adaptive vibration suppression of time-varyung structures with enhanced FxLMS algorithm,” *Mechanical Systems and Signal Processing,* vol. 118, pp. 93-107, 2019.

[34] H. S. Vu e K. H. Chen, “A low-power broad-bandwidth noise cancellation vlsi circuit design for in-ear heaphones,” *IEEE Transactions on Very Large Scale Integration Systems,* vol. 24, nº 6, pp. 2013-2025, 2016.

[35] P. N. Samarasinghe, W. Zhang e T. D. Abhayapala, “Recent Advances in Active Noise Control Inside Automobile Cabins: Toward quieter cars,” *IEEE Signal Processing Magazine,* vol. 33, nº 6, pp. 61-73, 2016.

[36] T. Memon, P. Beckett e A. Sadik, “Power-area-performance characteristics of FPGA-based sigma-delta FIR filters,” *Journal of Signal Processing Systems,* vol. 70, nº 3, 2013.

[37] J. Blauert, Spatial hearing: the psychophysics of human sound localization, MIT press, 1997.

[38] D. H. Crawford e R. W. Stewart, “Adaptive IIR filtered-v algorithms for active noise control,” *Acoustical Society of America,* vol. 101, nº 4, pp. 2097-2103, 1997.

[39] Texas Instruments, “TLC5540/TLC5510/TLC5510A Evaluation Module, User's Guide,” 2002.

[40] M. Integrated, “MAX5214/MAX5216 14-/16-Bit, Low-Power, Buffered Output, Rail-to-Rail DACs with SPI Interface,” 2013.

12. ANEXOS

Anexo 1 - Código SDM em VHDL

```
procedure SDM_block (x: in word; signal m: inout vector(0 to P-1); variable output: inout short_word; constant quantize: in boolean) is
    variable z: word;
begin
    z := x;
    for i in 0 to P-1 loop
        z := z - m(i);
    end loop;
    z := z + 2 ** (R - C - 1);
    output := z(R-1 downto R-C); -- quantization
    z := x;
    -- Update memory
    for i in 0 to P-1 loop
        z := z - m(i);
        m(i) <= (output & (1 to R-C => '0')) - z;
    end loop;
end SDM_block;
```

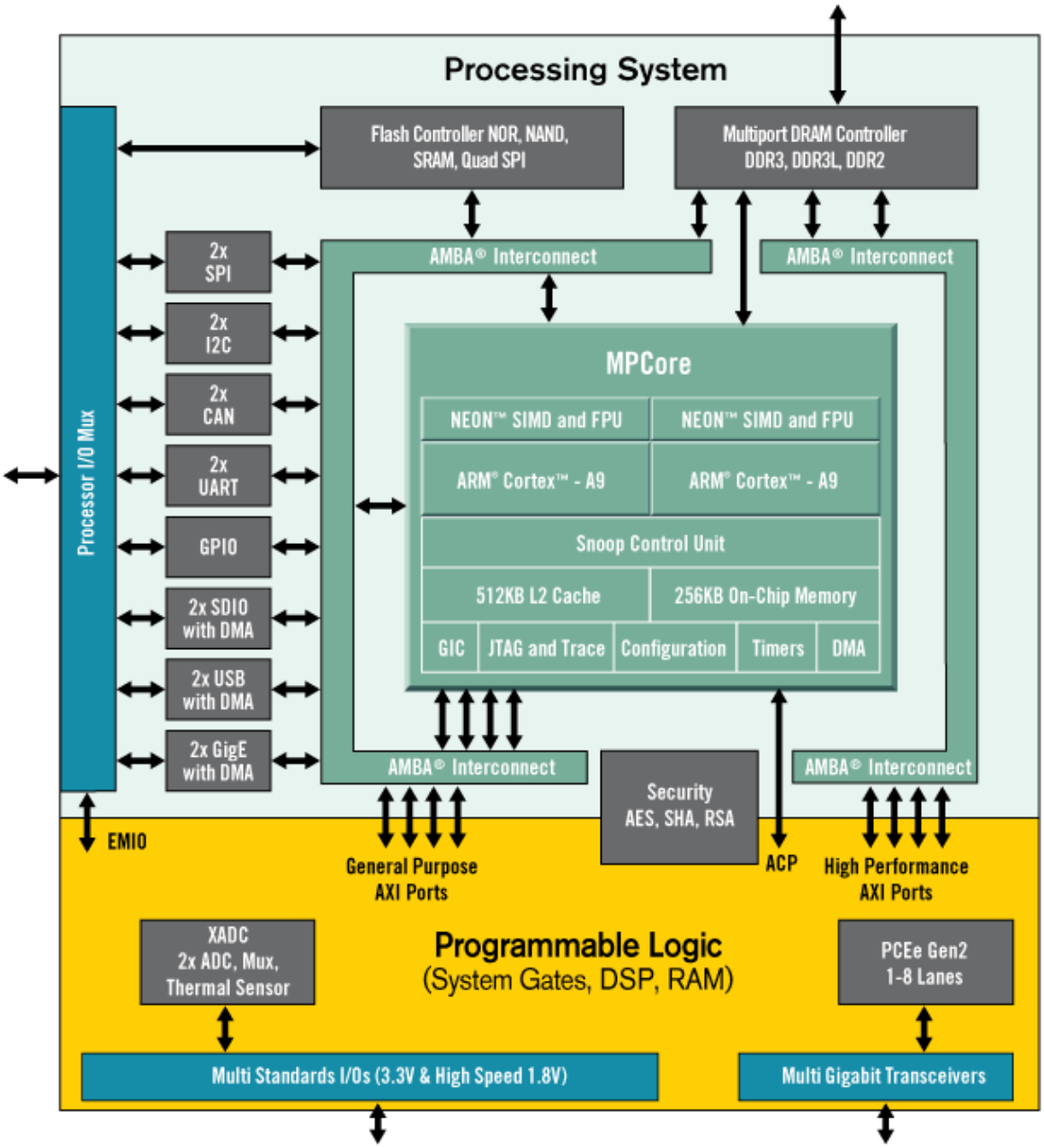
Anexo 2 - Código LMS em VHDL

```
process(clock) --Process wakes up when clock changes value
        variable erro: word;
        variable u_SDM: short_word := (others => '0');
        variable erro_SDM: short_word := (others => '0');
    begin
      if rising_edge(clock) then
        SDM_block(u & (1 to R-16 => '0'), SDM_m_u, u_SDM, True);
        for i in 0 to L-3 loop
          w_buffer(i) <= w_buffer(i+1) + w_coeffs(i+1)*u_SDM;
        end loop;
        w_buffer(L-2) <= w_coeffs(L-1)*u_SDM;
        output <= (w_buffer(0) + w_coeffs(0)*u_SDM) + 2 ** (C-2);  -- round
        -- buffer
        for i in 1 to L-1 loop
          LMS_buffer(i) <= LMS_buffer(i-1);
        end loop;
        LMS_buffer(0) <= u_SDM;
        -- Erro microfone:
        erro := d  & (1 to R-16 => '0') - output(C+R-2 downto C-1);
        erro_signal <= erro;
        SDM_block(erro, SDM_m_e, erro_SDM, True);
        -- Update weights
        for i in 0 to L-1 loop
          w_coeffs(i) <= w_coeffs(i) + step_size*LMS_times(erro_SDM, LMS_buffer(i));
        end loop;
      end if;
    end process;
```

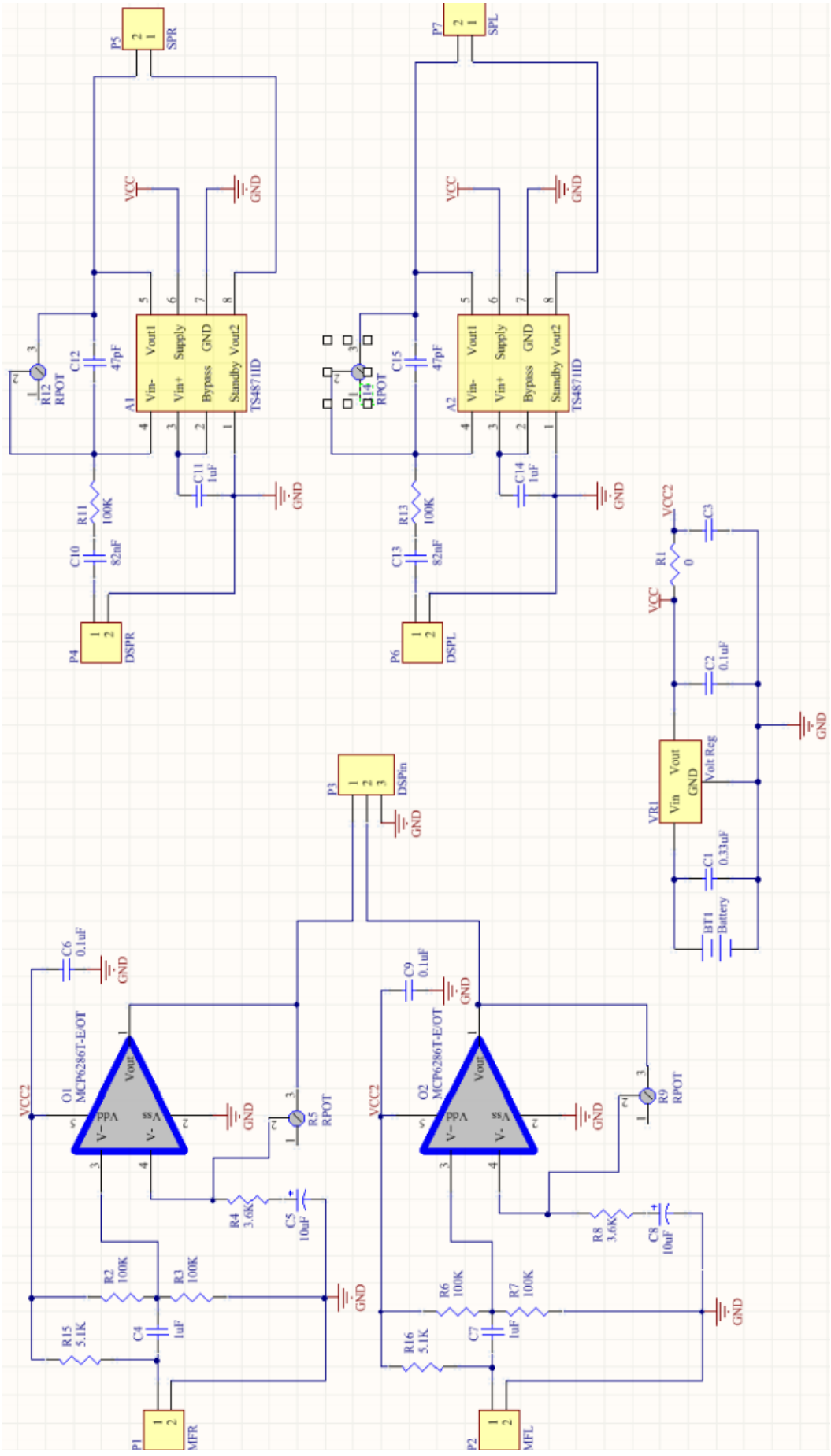
Anexo 3 - Código Simulação VHDL

```
process(clock)
        variable aux: real;
        variable pPathOut: signed(31 downto 0);
        -- Primary Path array, only used for simulation
        variable pPath: data_vector_type(0 to L-1) := (others => x"0000");
        variable inputArray: data_vector_type(0 to L-1) := (others => x"0000");
        variable seed1 : positive := 47387;
        variable seed2 : positive := 27273;
    begin
        if rising_edge(clock) then
            uniform(seed1, seed2, aux);
            pPath(128) := x"4000"; --atraso de 128 samples e atenuação de 50%
            u <= to_signed(integer((2.0**15 - 1.0) * aux - 2.0**14 + 1.0), 16); --Passar para 16 bits
            --Fill input array
            for j in 0 to L-2 loop
                inputArray(L-1-j) := inputArray(L-2-j);
            end loop;
            inputArray(0) := u;
            --Simulation of Output after primary Path
            pPathOut := (others=>'0');
            for k in 0 to L-1 loop --vector multiplication
                pPathOut := pPathOut + pPath(k)*inputArray(k);  --Apply primary path to inputArray
            end loop;
            if pPathOut(14) = '1' then
                d <= pPathOut(30 downto 15) + 1;
            else
                d <= pPathOut(30 downto 15);
            end if;
        end if;
    end process;
```

Anexo 4 - Diagrama de blocos da ZedBoard



Anexo 5 - Esquemático da PCB de amplificação de sinais



Anexo 6 - Diagrama de blocos do sistema geral desenvolvido no Vivado