TÉCNICO LISBOA

# Conversor Síncrono de Frequências de Amostragem para Áudio Digital

**Luís Filipe Pires Martins**

Dissertação para obtenção do Grau de Mestre em

**Engenharia Electrotécnica e Computadores**

Orientador(es): Fernando Manuel Duarte Gonçalves

**Júri**

Presidente: Prof. Francisco André Corrêa Alegria
Orientador: Prof. Fernando Manuel Duarte Gonçalves
Vogal: Prof. Gonçalo Nuno Gomes Tavares

**Novembro 2019**

Dedicado à minha família e amigos.

# 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.

# Agradecimentos

Escrever uma tese de mestrado não é fácil, demora muito tempo, é cansativo e parece impossível estar motivado para pesquisar ou para continuar a realizar experiências quando os resultados não são os desejados. Felizmente, torna-se mais fácil se tivermos quem nos ajude.

Gostava de agradecer ao instituto superior técnico por me ter dado um choque de realidade quando sai do secundário, fez-me perceber que era preciso trabalhar para ter resultados e no fundo tornou-me perseverante e determinado. Foi importante também para disponibilizar um ambiente calmo e com tudo o que pudesse necessitar ao meu redor.

Seguidamente, gostava de agradecer sobretudo ao meu orientador Prof. Fernando Gonçalves, teve paciência para todas as minhas dúvidas e esteve sempre disponível a partilhar quer a sua enorme sabedoria pelo tema quer a fornecer conselhos para que a tese decorresse pacificamente.

Adicionalmente, gostava de agradecer aos meus pais por garantirem que acabava o curso e ao meu irmão que de vez em quando partilhava a sua experiência académica.

Finalmente, gostava de agradecer aos meus amigos que me acompanharam ao longo deste tempo todo. Senti que foram fundamentais para conseguir os meus objetivos. Gostava de destacar a Paula Ceita, foi uma fonte de positivismo e boa disposição ajudou para que os momentos maus não fossem tão maus, o André Leite, não conheço ninguém tão genuíno e interessante como ele costumava dar o lado imparcial e realista da situação, o Bernardo Beirão era a minha escapatória para o mundo real e para descomprimir da tese e o André Santos, foi o meu rival durante o curso levando-me a ultrapassar os meus limites.

# Resumo

No trabalho realizado foi implementado um método para realizar a conversão de frequências de amostragem para áudio digital de forma simples, direta e que ocupa poucos recursos numa FPGA. No decorrer deste trabalho foram realizados duas implementações, uma em software e outra em hardware. A implementação em software teve como intuito realizar testes com diferentes parâmetros de uma forma rápida e eficiente. Os parâmetros testados foram a frequência do sinal fundamental, a frequência de amostragem de entrada, a ordem do polinómio interpolador e os rácios de conversão de frequências. Foram também testadas configurações com e sem filtragem. O propósito da segunda implementação era ter um protótipo que pudesse ser implementado numa FPGA e que ocupasse poucos recursos. Usou-se um método direto (polinómio de Lagrange) para realizar a conversão, pois utiliza fórmulas explícitas para obtenção dos coeficientes, não sendo necessário otimização. Adicionalmente, para frequências reduzidas apresentam uma qualidade bastante elevada ideal para áudio digital. Finalmente, pode ser implementado com algoritmos e estruturas eficientes, reduzindo o esforço computacional. Usou-se uma ordem de polinómio elevada, porque constatou-se que os filtros anti-*aliasing* ocupavam muitos recursos, mas não acrescentavam uma melhoria significativa. A implementação em hardware foi validada por simulação, tendo-se obtido taxas de distorção harmónicas mais ruído abaixo de -90 dB na banda de Nyquist para alguns rácios de conversão predefinidos.

**Palavras-chave:** SRC, Lagrange, FPGA

## Abstract

The work carried out intends to find a method to perform the conversion of sampling frequencies for digital audio signals in a simple, straightforward way and that occupies few resources on a FPGA. During this work two implementations were made, one in software and one in hardware. The software implementation aimed to perform tests with different parameters quickly and efficiently. The parameters tested were the fundamental signal frequency, the input sampling frequency, the interpolator polynomial order, and the sample rate ratios. Configurations with and without filtering were also tested. The purpose of the second implementation was to have a prototype that could be deployed on a FPGA and would take up few resources. A direct method (Lagrange polynomial) was used to perform the conversion, because it uses explicit formulas to obtain the coefficients, and there is no need for optimization. Additionally, for low frequencies they feature a very high quality ideal for digital audio. Finally, it can be implemented with efficient algorithms and structures, reducing computational effort. A high polynomial order was used, because it was found that anti-*aliasing* filters were resource intensive, but diddn't add significant improvement. The hardware implementation was validated by simulation and the total harmonic distortion plus noise in the Nyquist band was below -90 dB for some predefined sample rate ratios.



**Keywords:** SRC, Lagrange, FPGA

# Conteúdo

# Lista de Tabelas

# Lista de Figuras

# Lista de Símbolos

### Simbolos gregos

$\Omega$ Frequência analógica.

$\omega$ Frequência angular normalizada.

### Simbolos romanos

$\widehat{h_I}(t)$ Resposta impulsional ideal.

$F$ Frequência de amostragem.

$f_i$ Frequência de amostragem de entrada.

$F_{LP}$ Frequência de corte ideal do filtro passa-baixo.

$s(t)$ Sequência de impulsos de Dirac.

$S_C(j\Omega)$ Transformada Fourier contínua de $s(t)$.

$To$ Período de amostragem de saída.

$u_0$ Resposta unitária a um impulso.

$X(\mathrm{e}^{j\omega})$ Transformada Fourier discreta de $x_c(t)$.

$x[n]$ Sequência de amostras discretas de entrada.

$x_s(t)$ Sinal modulados por amplitude.

$X_C(j\Omega)$ Transformada Fourier contínua de $x_c(t)$.

$x_c(t)$ Sinal contínuo no tempo.

$y[m]$ Sequência de amostras discretas de saída.

### Subscritos

$0$ Instante inicial.

$c, C$ Contínuo no tempo.

$I$ Ideal.

$i$ Entrada.

$LP$ Filtro passa-baixo.

$o$ Saída.

$s$ Modulado por amplitude.

# Lista de Abreviações

| | |
|---|---|
| **AD** | Analógico-Digital. |
| **CIC** | Cascaded-Integrator Comb. |
| **CI** | Circuito Integrado. |
| **DA** | Digital-Analógico. |
| **FFT** | Fast Fourrier Transform. |
| **FIR** | Finite Impulse Response. |
| **FPGA** | Field Programable Gate Array. |
| **IIR** | Infinite Impulse Response. |
| **LS** | Least Square Error. |
| **LUT** | Look Up Table. |
| **LVPT** | Linear e que Varia Periodicamente no Tempo. |
| **MA** | Moving Average. |
| **MF** | Maximal Flatness. |
| **MIPS** | Mega Instruções Por Segundo. |
| **MMC** | Múltiplas Multiplicações Comuns. |
| **NNI** | Nearest Neighbor Interpolation. |
| **O-MOMS** | Optimal Maximal-Order interpolation of Minimal Support. |
| **PMA** | Pulso Modulado por Amplitude. |
| **RAM** | Random Access Memory. |
| **RDS** | Rádio Definido por Software. |
| **ROM** | Read Only Memory. |
| **SFDR** | Spurious Free Dynamic Range. |
| **SNR** | Signal to Noise Ratio. |
| **TDH** | Taxa de Distorção Harmónica. |
| **VLSI** | Very Large Scale Integration. |

# Capítulo 1

# Introdução

## 1.1 Motivação

Devido a razões económicas, de engenharia ou até mesmo históricas existem diferentes frequências de amostragem para diferentes sistemas [1]. Em processamento digital de sinais, usam-se conversores de frequências de amostragem (SRC - *Sample Rate Converter*) sempre que se quiser converter uma frequência de amostragem de entrada noutra frequência de amostragem qualquer [2]. Um exemplo comum na literatura é a conversão de *compact disk* (CD) para *digital audio tape* (DAT), onde um sinal amostrado a 44,1 kSa/s é convertido para 48 kSa/s [3]. Numa perspetiva mais geral, são necessários SRC sempre que é preciso fazer a conversão de formatos de multimédia entre sistemas, pois as frequências de amostragem usadas na gravação tipicamente são diferentes, mas a reprodução do sinal só é possível caso a frequência de amostragem corresponda à frequência de amostragem do sistema que irá fazer a reprodução.

A inevitabilidade dos SRCs surge porque existe dentro do processamento digital de sinais aplicações que requerem este tipo de conversões [4, 5]:

- Sincronização de símbolos em recetores digitais;
- Conversores de frequências arbitrários;
- Processamento de áudio digital;
- Televisão digital;
- Processamento digital de vídeo;
- Processamento digital de imagem;
- Software de rádio.

Na indústria são usadas várias frequências de amostragem, por exemplo, na qualidade das chamadas em telemóveis usa-se 8 kHz, em telecomunicação usa-se 16 kHz, em radiodifusão usa-se 32 kHz, em

leitores de CD usa-se 44,1 kHz, em produtos domésticos usa-se 44,1 kHz e em aplicações profissionais usa-se 48 kHz [6, 7].

Há uma forma tradicional de abordar o problema de converter diferentes frequências de amostragem que consiste no uso de um conversor digital-analógico (DA) juntamente com um filtro ideal e um conversor analógico-digital (AD). Esta solução foi usada no passado. Com o aparecimento de estúdios digitais é comum fazer todo o processamento do sinal digitalmente, por exemplo com a gravação, edição e produção de sinais. As sucessivas conversões realizadas por estes conversores introduzem erros significativos e o custo associado a acrescentar este número excessivo de conversores e filtros é também uma razão para se procurar soluções mais eficientes [8].

As técnicas usadas atualmente envolvem técnicas digitais e evoluíram de forma a serem mais eficientes e complexas.

## 1.2 Objetivos

O objetivo desta tese é implementar um conversor de frequências de amostragem que ocupe poucos recursos num dispositivo reconfigurável (FPGA - *Field Programable Gate Array*) com uma TDH+N - Taxa de Distorção Harmónica mais ruído máxima de -90 dB.

Numa primeira fase será criado um modelo em C do algoritmo, que servirá para ajustar parâmetros do SRC, verificar se a implementação é realizável e observar o comportamento do algoritmo de forma a determinar quais as condições que permitiam extrair o melhor desempenho. Tendo em consideração algumas limitações, como por exemplo, a ordem do polinómio interpolador.

Numa segunda fase será desenvolvida uma descrição em Verilog que realize o controlo dos dados, desde os coeficientes lidos da memória ROM - *Read Only Memory*, até às amostras de entrada do sinal de áudio digital e construí-lo segundo as restrições temporais dos sinais para garantir que de fato as amostras de saída são idênticas às obtidas com o modelo em software.

## 1.3 Estrutura da tese

A tese está dividida em sete capítulos organizados da seguinte forma:

1. Introdução

   Este capítulo explica a existência da diversidade de frequências de amostragem e a necessidade de SRC.

2. Amostragem uniforme e teorema da amostragem

   Este capítulo introduz conceitos fundamentais para a compreensão do processamento digital de sinais, no que concerne ao processamento de áudio digital e conversores de frequência de amostragem. Menciona as bases que suportam a teoria das diferentes técnicas, nomeadamente o conceito de amostragem, o teorema da amostragem e as implicações de não cumprir as restrições impostas. No final é explicado como se faz a reconstrução de sinais.

3. Conversores de taxas de amostragem

   Este capítulo contém o estado da arte acerca de conversores de frequências de amostragem. São referidos artigos que foram relevantes no desenvolvimento deste trabalho, com principal destaque para as implementações e técnicas utilizadas. É explicada a métrica de desempenho da qualidade da conversão usada nesta tese.

4. Implementação em software

   Este capítulo serve para dar ênfase à interface do programa. É importante para entender a sua funcionalidade. Permite por exemplo, gerar uma referência de teste, implementar rapidamente o algoritmo ou testar parâmetros.

5. Implementação em hardware

   Este capítulo serve para apresentar todos os aspetos relacionados com a implementação em hardware. É apresentado o diagrama de blocos, a descrição das entradas e saídas da unidade funcional e indicam-se as restrições do método.

6. Resultados

   Este capítulo serve para apresentar os resultados obtidos, discuti-los com a teoria e compara-los com os métodos atuais.

7. Conclusões

   Este capítulo apresenta considerações finais e sugestões para trabalho futuro.

# Capítulo 2

# Amostragem

O propósito deste capítulo é descrever as bases fundamentais sobre amostragem, abordar o teorema da amostragem, fazer uma análise no domínio do tempo e da frequência, e analisar a amostragem de sinais contínuos e a reconstrução de sinais discretos.

## 2.1 Amostragem uniforme no tempo

Para este capítulo há quatro referências [9–12] que explicam os conceitos de amostragem uniforme, o teorema da amostragem e a reconstrução de sinais. Estes conceitos serão abordados de um modo resumido. Para mais detalhes recomenda-se a leitura destes trabalhos [11, 12].

As representações dos sinais nas etapas da reconstrução e amostragem estão apresentadas na figura 2.1.

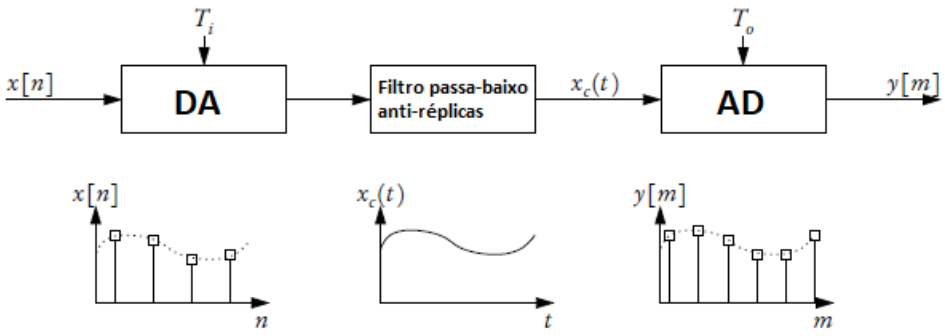



Figura 2.1: Reconstrução e amostragem de um sinal contínuo no tempo (adaptado de [12])

A sequência de amostras $x[n]$ com frequência de amostragem $f_i$ é convertida para um sinal contínuo no tempo $x_c(t)$ com recurso a um conversor DA. O sinal volta a ser amostrado com o auxílio de um conversor AD nos instantes de saída desejados para formar a sequência $y[m]$. Esses instantes são controlados com o período de amostragem de saída $To$.

A conversão de contínuo para discreto pode ser vista como um processo de modulação, exemplificado na figura 2.2. Neste caso considera-se $x_c(t)$ uma função contínua da variável $t$. Está-se interessado na amostragem de $x_c(t)$ a um ritmo uniforme $t = nT, -\infty < n < +\infty$, ou seja uma vez a

cada intervalo de duração $T$. O sinal amostrado designa-se $y[m]$.

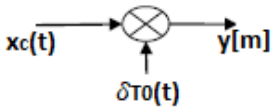



Figura 2.2: Diagrama de blocos para realizar amostragem (adaptado de [10])

O sinal continuo $x_c(t)$ é multiplicado (modulado) por uma sequência de impulsos de Dirac $s(t)$ para dar origem ao sinal pulso modulado por amplitude (PMA), $x_c(t)s(t)$. Nas equações seguintes estão definidos: o sinal discretizado no tempo equação 2.1 e a sequência de impulsos de Dirac equação 2.2:

$$x[n] = \lim_{\epsilon \to 0} \int_{t-nT-\epsilon}^{nT+\epsilon} x_c(t)s(t)\mathrm{d}t\,. \tag{2.1}$$

$$s(t) = \sum_{l=-\infty}^{\infty} \delta_0(t - lT)\,. \tag{2.2}$$

Onde $\delta_0$ é uma função com impulso unitário ideal, $x[n]$ corresponde à área no instante $nT$ que tendo em conta que está a ser multiplicado pelo impulso unitário, cuja área é 1, significa que $x[n]$ corresponde ao valor de $x_c(nT)$, representado na equação 2.3.

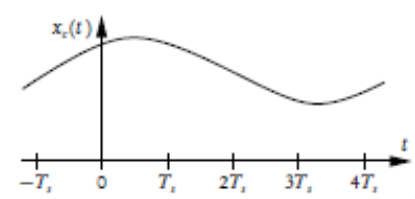
$$x[n] = x_c(nT)\,. \tag{2.3}$$

Importante saber que o impulso de Dirac é um artifício matemático que tem área unitária no instante $t = 0$ e para todos os outros instantes temporais é nulo equação 2.4. Para além disso, existe uma propriedade que refere que qualquer argumento de uma função pode ser definido por um atraso do impulso de Dirac adequado equação 2.5, acabando por justificar as conclusões anteriores. Em termos matemáticos definem-se desta forma:

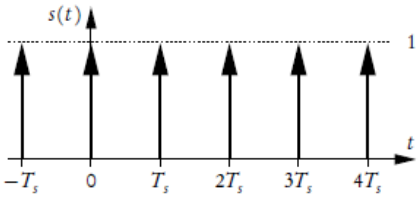
$$\int_{-\infty}^{\infty} f(t)\delta(t)\mathrm{d}t = f(0)\,. \tag{2.4}$$

$$f(\tau) = \int_{-\infty}^{\infty} f(t)\delta(t-\tau)\mathrm{d}t\,. \tag{2.5}$$

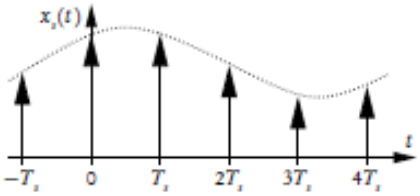
Na figura 2.3 está representado o sinal contínuo do tempo, a sequência de impulsos de Dirac, os pulsos modulados por amplitude e ainda o sinal discreto.



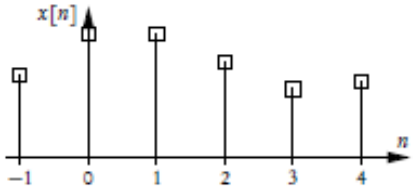

(a) Sinal contínuo




(b) Sequência de impulsos de Dirac




(c) Pulsos modulados por amplitude




(d) Sinal discreto

Figura 2.3: Amostragem de um sinal contínuo no tempo com recurso a modulação, para formar um sinal discreto no tempo (fonte [12]).

### 2.1.1 Transformada Z

Algumas das fórmulas derivadas neste capítulo podem ser mais facilmente obtidas através da aplicação da transformada Z. Esta sub-secção pretende apenas introduzir os conceitos da mesma, apesar de na prática não os utilizar diretamente.

A transformada Z simplifica a descrição do domínio da frequência e a análise de sistemas discretos no tempo, que sofreram uma amostragem. O sinal $x_s(t)$ resulta da multiplicação do sinal $x_c(t)$ por uma sequência de impulsos de Dirac $s(t)$, evidenciado na equação 2.6.

$$\begin{aligned} x_s(t) =& x_c(t) \cdot s(t) \\ =& \sum_{l=0}^{\infty} x[l] \cdot \delta_0(t - lT) \quad \text{se for causal} \end{aligned} \tag{2.6}$$

A transformada de Laplace do sinal $x_c(t)$ é dado pela equação 2.7.

$$\begin{aligned} \mathcal{L}\{x_s(t)\} =& x(0) + x(T)\mathrm{e}^{-sT} + x(2T)\mathrm{e}^{-2sT} + \ldots \\ =& \sum_{l=0}^{\infty} x(lT)\mathrm{e}^{-lsT} \end{aligned} \tag{2.7}$$

Mapeando o semi-plano esquerdo para o interior do círculo unitário (para garantir a estabilidade do sistema), tem-se $z = \mathrm{e}^{sT}$. A transformada Z directa é dada pela equação 2.8 e a transformada Z inversa é dada pela equação 2.9.

$$X(\mathrm{z}) = \sum_{n=0}^{\infty} x(nT)\mathrm{z}^{-n} = \mathcal{Z}\{x(nT)\} \tag{2.8}$$

$$x(nT) = \frac{1}{2\,\pi\,j} \oint_C X(\mathrm{z})\,\mathrm{z}^{n-1}\,\mathrm{dz} = \mathcal{Z}^{-1}\{X(\mathrm{z})\} \tag{2.9}$$

Onde $C$ é o caminho anti-horário fechado contendo a origem e a totalidade da região de convergência. No caso em que a região de convergência é causal, significa que tem que conter todos os polos de $X(\mathrm{z})$.

## 2.2 Amostragem uniforme na frequência

Há duas formas de fazer esta análise, pode-se aplicar a transformada de Fourier contínua no sinal $x_s(t)$ e usar as relações de $x[n]$ e $s(t)$, ou alternativamente aplicar a transformada contínua de Fourier na sequência de impulsos de Dirac $s(t)$ e no sinal contínuo no tempo $x_c(t)$ e aplicar as propriedades das transformadas de Fourier para obter o resultado pretendido, que é a transformada de Fourier contínua dos pulsos modulados por amplitude $x_s(t)$.

Optando por realizar as transformadas nos sinais $x_c(t)$ e $s(t)$, definem-se as suas transformadas de Fourier da seguinte forma:

$$X_C(j\,\Omega) = \int_{-\infty}^{\infty} x_c(t)\mathrm{e}^{-j\,\Omega t}\,\mathrm{d}t\,. \tag{2.10}$$

$$S_C(j\,\Omega) = \int_{-\infty}^{\infty} s(t)\mathrm{e}^{-j\,\Omega t}\,\mathrm{d}t\,. \tag{2.11}$$

Nas expressões, $\Omega$, representa a frequência analógica em rad/s. Completa-se a equação 2.11 com a equação 2.2, obtendo-se a equação 2.12:

$$S_C(j\,\Omega) = \frac{2\pi}{T} \sum_{l=-\infty}^{\infty} \delta_0(\Omega - \frac{2\pi l}{T})\,. \tag{2.12}$$

Manipulando a equação 2.12 com as expressões $F = \frac{1}{T}$, $\Omega = 2\pi f$ e com a variável auxiliar $\Omega_s = 2\pi F$ obtém-se:

$$S_C(j\,\Omega) = \Omega_s \sum_{l=-\infty}^{\infty} \delta_0(\Omega - l\Omega_s)\,. \tag{2.13}$$

A equação 2.13 indica que os impulsos de Dirac para além de serem periódicos no tempo, também o são no domínio da frequência. Tendo neste momento a transformada do sinal contínuo no tempo e a transformada dos impulsos de Dirac é finalmente possível saber qual a transformada de Fourier dos pulsos modulados por amplitude, ou seja, como vai ser a transformada de Fourier do sinal discreto. A modulação ou multiplicação de dois sinais no domínio do tempo, traduzem-se por uma convolução entre duas transformadas na frequência.

$$X_C(j\,\Omega) * S_C(j\,\Omega) = \int_{-\infty}^{\infty} [x_c(t)s(t)]\mathrm{e}^{-j\,\Omega t}\,\mathrm{d}t\,. \tag{2.14}$$

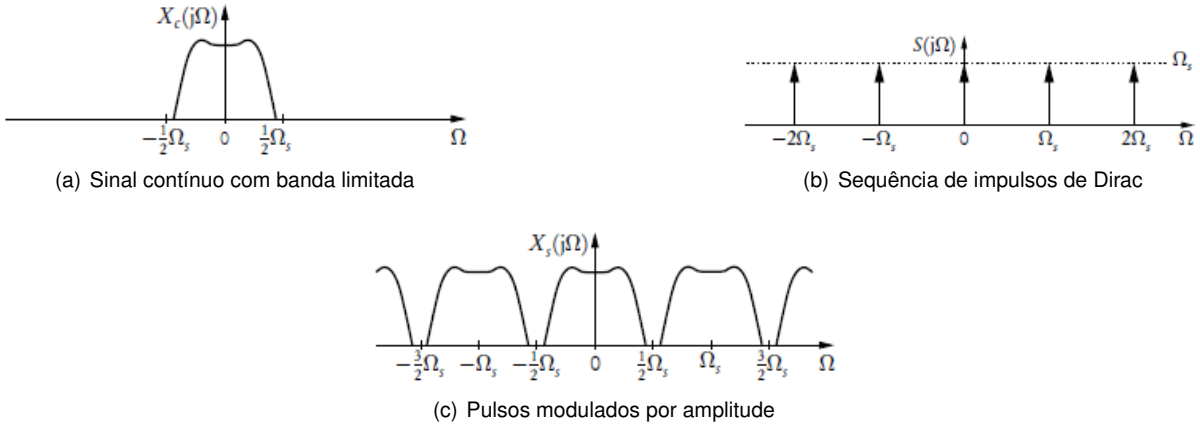
Portanto, tem-se:

$$X_s(j\Omega) = \frac{1}{T_s} \sum_{l=-\infty}^{\infty} X_C(j[\Omega - l\Omega_s]) \,. \tag{2.15}$$

E a representação alternativa:

$$X_s(j\Omega) = \sum_{-\infty}^{\infty} x[n] \mathrm{e}^{-j\Omega T_s n} \,. \tag{2.16}$$

Na figura 2.4 pode-se ver a modulação no domínio da frequência.




(a) Sinal contínuo com banda limitada

(b) Sequência de impulsos de Dirac

(c) Pulsos modulados por amplitude

Figura 2.4: Modulação no domínio da frequência (fonte [12]).

Vai ser útil analisar as propriedades no domínio da frequência do sinal discreto no tempo $x[n]$, cuja informação é conseguida com a transformada de Fourier, existindo a alternativa de se usar a transformada inversa e contínua de Fourier no sinal contínuo no tempo $x_c(t)$ e depois com as equações 2.1 e 2.3 obter a correspondente transformada de Fourier inversa do sinal discreto no tempo $x[n]$.

A transformada discreta de Fourier do sinal discreto do tempo é dado pela equação 2.17.

$$X(\mathrm{e}^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n] \mathrm{e}^{-j\omega n} \,. \tag{2.17}$$

Na equação 2.17, $\omega$ é a frequência angular normalizada e é dada pela equação 2.18.

$$\omega = 2\pi \frac{\Omega}{\Omega_s} = \Omega T_s \,. \tag{2.18}$$

Olhando para as equações 2.16 e 2.17 , chega-se à conclusão que $X(\mathrm{e}^{j\omega})$ é igual a $X_s(j\Omega)$ quando $w = \Omega T_s$, obtendo-se:

$$X(\mathrm{e}^{j\omega}) = \frac{1}{T_s} \sum_{l=-\infty}^{\infty} X_C(j[\frac{w}{T_s} - l\frac{2\pi}{T_s}]) \,. \tag{2.19}$$

Esta equação faz a ligação entre sistemas discretos e contínuos e enuncia que o sinal discreto é composto por translações e escalamento em amplitude do sinal analógico.

## 2.3 Teorema da amostragem

Para perceber esta secção é importante considerar que a partir do sinal contínuo no tempo $x_c(t)$ é sempre possível obter um sinal discreto no tempo $x[n]$ único, o contrário no entanto depende, ou seja tendo a sequência de valores discretos pode não ser possível recuperar o sinal original. As condições para que isso aconteça estão enunciadas no teorema de Nyquist que diz que, se um sinal contínuo $x_c(t)$ tem uma banda limitada na transformada de Fourier $X_C(j\Omega)$, ou seja $|X_C(j\Omega)| = 0$ para $|\Omega| \geq 2\pi F_C$, então $x_c(t)$ pode ser unicamente reconstruido sem erros através de amostras equidistantes $x_c(nT)$, $-\infty < n < \infty$, se $F \geq 2F_C$ onde $F = \frac{1}{T}$ é a frequência de amostragem. Pode-se ilustrar o que sucede se o teorema não for cumprido. Se se definir $\Omega_N$ como sendo a frequência de Nyquist, há três casos possíveis: a frequência angular de amostragem $\Omega_s$ ser maior à frequência de Nyquist $\Omega_s > 2\Omega_N$, pode ser igual $\Omega_s = 2\Omega_N$ ou menor $\Omega_s < 2\Omega_N$. Os primeiros dois cenários correspondem à figura 2.4 c), sendo que no caso de ser igual os espectros encontram-se na fronteira mas não se sobrepõem. Na figura 2.5 representa o terceiro caso, chama-se espelhamento espectral ou *aliasing* e nesse caso a reconstrução não é perfeita, porque algumas componentes de frequência do sinal contínuo vão ser mapeadas para a mesma frequência do sinal discreto.

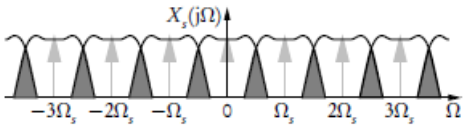



Figura 2.5: Representação do espectro caso o teorema de Nyquist não seja cumprido (fonte [12]).

## 2.4 Reconstrução de sinais

A principal ideia a reter do teorema da amostragem é que qualquer sinal contínuo pode ser recuperado, desde que a frequência de amostragem cumpra os requisitos mínimos. A técnica para reconstruir sinais contínuos no tempo, a partir de sinais discretos denomina-se filtragem. A filtragem serve para recuperar o espectro do sinal $X_C(j\Omega)$ após ter sido realizada a convolução $X_C(j\Omega) * S_C(j\Omega)$. É necessário ainda um conversor DA para se ter $x_c(t)s(t)$ a partir de $x[n]$.

A fórmula de reconstrução é:

$$x_c(t) = \sum_{n=-\infty}^{\infty} x[n]\widehat{h}(t - nT)\,. \tag{2.20}$$

Assumindo um filtro ideal com frequência de corte $F_{LP}$ e uma resposta impulsiva ideal $\widehat{h_I(t)}$, sendo assumido que a frequência de Nyquist é respeitada, então aplicando a transformada de Fourier inversa a essa resposta na frequência, dá origem à equação 2.21:

$$\widehat{h_I}(t) = \frac{\sin(2\pi F_{LP}t)}{2\pi F_{LP}t}\,. \tag{2.21}$$

A frequência de corte costuma ser definida como:

$$F_{LP} = \frac{F}{2} = \frac{1}{2T}\,. \tag{2.22}$$

Donde resulta que, a fórmula da reconstrução é dada por:

$$x_c(t) = \sum_{n=-\infty}^{\infty} x[n]\,\frac{\sin[\pi(t-nT)/T]}{\pi(t-nT)/T} = \sum_{n=-\infty}^{\infty} x[n]\;\mathrm{sinc}[(t-nT)/T]\,. \tag{2.23}$$

O filtro de reconstrução num determinado instante $t$, corresponde à convolução discreta entre a sequência $x[n]$ e a sequência discreta obtida ao amostrar a resposta contínua do impulso do filtro ideal $\widehat{h_I}(t)$ nos instantes $t - nT$. O filtro ideal passa-baixo pode ser visto como um interpolador do sinal de banda limitada $x[n]$ porque permite calcular qualquer ponto com base nas várias amostras recebidas a uma frequência de amostragem suficientemente elevada. O filtro ideal não é realizável porque seria necessário uma sequência infinita de amostras, em vez disso são usadas aproximações. No entanto o uso de aproximações adiciona erros. Por exemplo, pode haver flutuações nos valores da banda de passagem, pode causar erros de fase (estes erros podem ser compensados com filtros com uma resposta a um impulso simétrica) e podem haver erros provocados por uma atenuação insuficiente das réplicas dos espetros na banda de atenuação.

# Capítulo 3

# Conversores de frequências de amostragem

O capítulo anterior referiu os processos envolventes na conversão de um sinal contínuo no tempo para um sinal discreto no tempo e vice-versa. O propósito deste capítulo serve para explicar os SRCs, começando inicialmente pela apresentação das implementações e técnicas usadas, prosseguindo para a avaliação da qualidade da conversão realizada e finalizando com a fundamentação da solução proposta.

A frequência de amostragem representa o número de amostras recolhidas por segundo de um sinal contínuo no tempo e tem como unidades, Sa/s (*samples per second*). Quando se fala em conversão de frequências de amostragem, significa que se está interessado em utilizar a sequência de amostras discretas no tempo resultantes da amostragem de um sinal contínuo no tempo, que foi amostrado com uma determinada frequência de amostragem de entrada, e com a aplicação de uma técnica arbitrária criar uma nova sequência de amostras discretas equivalentes à amostragem desse sinal contínuo no tempo amostrado à frequência de amostragem de saída. A figura 3.1 ilustra o modelo de conversão.

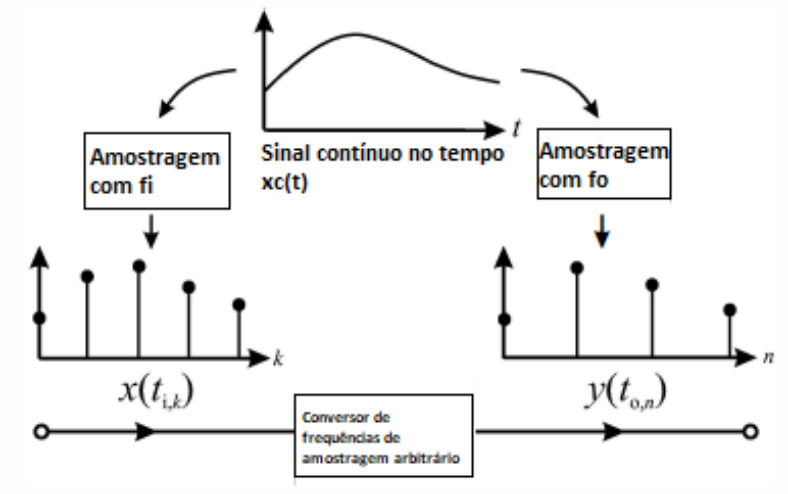


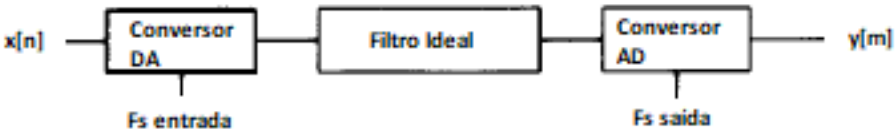

Figura 3.1: Conversão de frequências de amostragem (adaptado de [13])

Há duas classes de conversores: síncronos e assíncronos [8, 14]. Os conversores síncronos são aqueles em que se sabe *a priori* a frequência de amostragem de entrada e o fator de conversão. Por exemplo, se o sinal de entrada tiver uma frequência de amostragem de 48 kSa/s e o fator de conversão for 0,91875, obtém-se na saída um sinal amostrado a 44,1 kSa/s (48 kSa/s $\times$ 0,91875). No caso de conversores assíncronos por oposição aos conversores síncronos, não se conhecem as frequências de amostragem exatas, porque o sinal de relógio pertence a equipamentos diferentes e a referência temporal desses sistemas é diferente, por isso se se medissem as frequências de amostragem no exemplo anterior, as frequências de amostragem não seriam exatamente 44,1 kSa/s e 48 kSa/s.

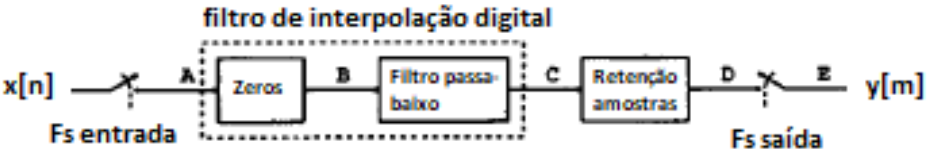
As técnicas de conversão podem ser divididas em dois tipos:

- Abordagem analógica;
- Abordagens digitais.

Na figura 3.2 está apresentado o diagrama de blocos da conversão analógica e um exemplo de um diagrama de blocos para uma conversão digital.




(a) Diagrama de blocos para conversão analógica




(b) Diagrama de blocos para conversão digital

Figura 3.2: Diagrama de blocos para conversão analógica e digital (adaptado de [8]).

## 3.1 Interpretação analógica

Para a conversão analógica é comum utilizar um conversor DA para se reconstruir o sinal contínuo. De seguida, usa-se um filtro passa-baixo para eliminar as réplicas, e o sinal resultante passa por um conversor AD para realizar a amostragem à frequência desejada. Esse processo está ilustrado na figura 3.2 a).

Devido ao filtro $\widehat{h}(t)$, a resposta ao impulso do filtro passa-baixo analógico vai ter uma duração finita e os valores da sequência discreta $y[m]$ no instante $t = mT'$ em que $T'$ é o período de amostragem de $y[m]$ vão ser obtidos através da sequência de amostras $x[n]$.

$$y[m] = x_c(t)|_{t=mT'} = \sum_{n=N1}^{N2} x[n]\widehat{h}(mT' - nT)\,. \tag{3.1}$$

$N1$ e $N2$ correspondem aos limites inferior e superior das amostras para o cálculo de cada valor de $y[m]$.

A sequência discreta x[n] amostra e pesa a resposta impulsiva. Os limites do somatório da equação 3.1 são determinados onde o filtro ideal é diferente de zero. Assumindo que $\widehat{h}(t)$ é nulo para $t < t_1$ e $t > t_2$, podem-se derivar as equações 3.2 e 3.3. Estas equações mostram as relações entre o período de amostragem de entrada ($T$), de saída ($T'$), da amostra $m$ e dos limites do filtro passa-baixo $t_1$ e $t_2$.

$$N1 = \lceil \frac{mT' - t_2}{T} \rceil . \tag{3.2}$$

$$N2 = \lfloor \frac{mT' - t_1}{T} \rfloor . \tag{3.3}$$

Há duas situações peculiares: a primeira situação é quando se têm duas respostas impulsionais de duração diferente para a mesma conversão, se os índices forem pares o conjunto de amostras de $x[n]$ que forma $y[m]$ é o mesmo, mas se os índices forem ímpares então o conjunto de amostras de $x[n]$ que forma $y[m]$ é diferente. A segunda situação envolve os coeficientes do filtro passa-baixo que variam consoante o rácio que se quer converter. Por exemplo, se o objetivo fosse o incremento da frequência de amostragem por um fator de 2, iria haver 2 conjuntos de coeficientes, um para $m$ par e outro para $m$ ímpar, se o rácio fosse um decréscimo por 2 haveria só 1 conjunto de coeficientes que seria usado para calcular a sequência de valores discretos de saída $y[m]$ [11].

Se se considerar que existe uma relação entre as frequências de amostragem em vez de serem arbitrárias, tem-se:

$$\frac{T'}{T} = \frac{M}{L} . \tag{3.4}$$

Se isso se verificar, significa que existem $L$ conjuntos de coeficientes únicos para o filtro passa-baixo analógico que são usados para calcular $y[m]$ a partir da sequência discreta $x[n]$. Se o rácio fosse irracional haveria infinitos conjuntos de coeficientes.

## 3.2 Técnicas de conversão digital

Como se pôde constatar na subsecção anterior o rácio de conversão pode ser representado pela equação 3.4. O que se pretende realizar é um sistema que abdique do uso de andares intermédios analógicos, ou seja o objetivo é realizar as conversões puramente no domínio digital. Para a conversão digital há uma forma comum a todas as conversões digitais, pois todos esses métodos substituem o filtro analógico por um filtro digital de interpolação e usam retenção de amostras para transformar essas amostras discretas num sinal contínuo. As amostras são de seguida amostradas nos instantes desejados. O diagrama de blocos encontra-se presente na figura 3.2 b) [8].

### 3.2.1 Método direto

Para concretizar um conversor que realize essas especificações pode-se proceder da seguinte forma: primeiro, arranja-se a resposta impulsiva do filtro analógico $L$ vezes superior à frequência de amostragem de entrada, de seguida, esse conjunto de $L$ amostras vai corresponder à resposta unitária do filtro digital e essas amostras são usadas também como coeficientes para calcular as amostras de saída da sequência discreta $y[m]$, recordando apenas que existem $L$ conjuntos de coeficientes únicos. Acerca destes sistemas uma propriedade bastante importante que têm é que são lineares e periodicamente variáveis no tempo [11].

### 3.2.2 Decimação

O processo que resulta de reduzir a frequência de amostragem designa-se por decimação. A forma mais fácil de o fazer é guardar uma amostra a cada $M$ amostras e ignorar as restantes, sendo a nova frequência de amostragem igual a: $f_{novo} = \frac{f_{velho}}{M}$. O espectro do sinal vai-se deslocar em torno de $f_{novo}$ e de todos os seus múltiplos por ser periódico. É preciso ter cuidado pois a banda do sinal de entrada tem de ser limitada, porque caso contrário pode ocorrer *aliasing* e perde-se informação, portanto é necessário impor a condição: $f_{novo} > 2B$, onde $B$ representa a banda do sinal. Nalgumas situações pode ser necessário usar um filtro passa-baixo antes de decimar, por exemplo se for necessário usar uma frequência de amostragem menor que $2B$, nesse caso é preciso filtrar primeiro com um filtro passa-baixo e só depois é que se ignoram/descartam as amostras da sequência de entrada. Para reter uma banda $B'$ da banda $B$ do sinal original, a atenuação máxima que o sinal pode ter é até à frequência dada por: $f_{paragem} = f_{novo} - B'$ para evitar que ocorra problemas relacionados com *aliasing* [1].

Recapitulando o procedimento para fazer decimação [10, 15]:

1. O sinal com frequência de amostragem $f_{velho}$ é primeiro filtrado com um filtro passa-baixo para remover a possibilidade de haver *aliasing*;
2. O sinal continua com a frequência de amostragem $f_{velho}$;
3. O sinal agora é decimado, ou seja a cada $M$ amostras guarda-se uma;
4. A frequência de amostragem é agora $f_{novo} = \frac{f_{velho}}{M}$;
5. A banda de interesse do sinal é $B'$, a banda original de interesse é $\frac{f_{velho}}{2M}$;

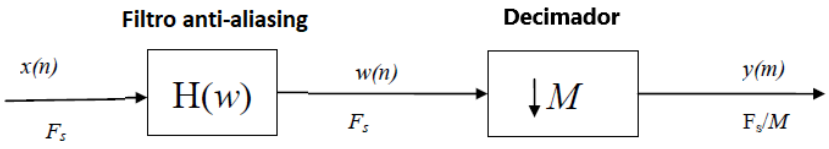



Figura 3.3: Diagrama de blocos para decimação (adaptado de [10])

A figura 3.3 mostra o diagrama de blocos para a decimação e a figura 3.4 mostra as várias etapas dos sinais durante a decimação. Na figura 3.4 a) tem-se a banda limitada do sinal de entrada, na

figura 3.4 b) tem-se as características do filtro passa-baixo para remover as frequências que poderiam provocar *aliasing*, na figura 3.4 c) tem-se o sinal resultante da filtragem e na figura 3.4 d) tem-se a banda do sinal de saída onde se guardou uma amostra a cada $M$ amostras.

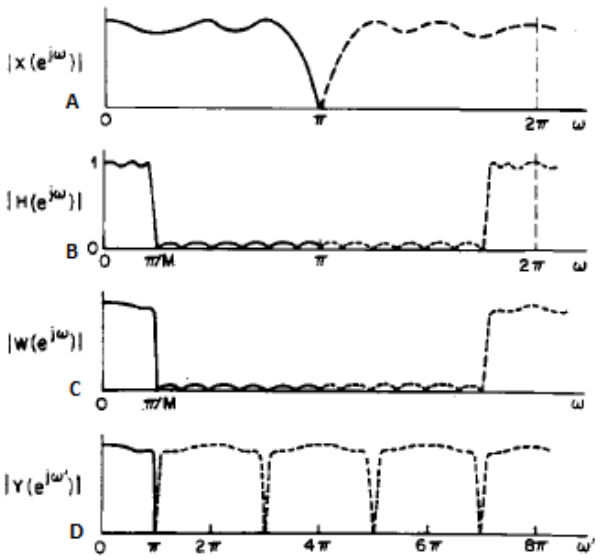
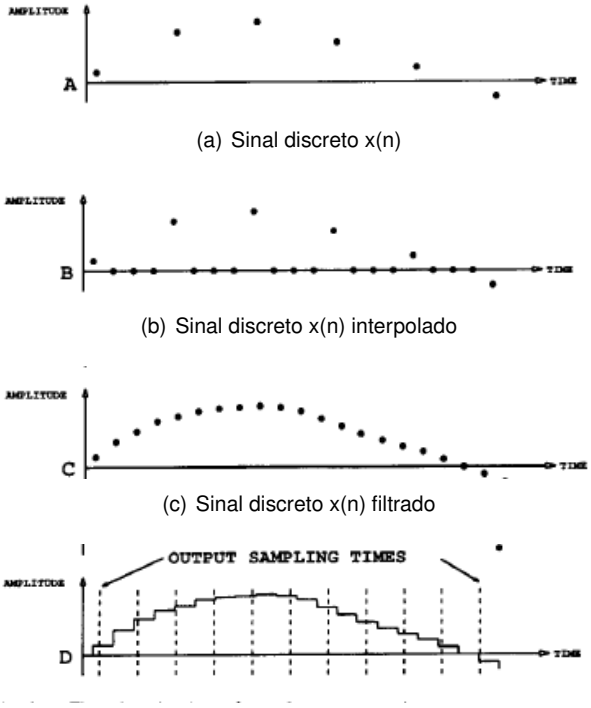



Figura 3.4: Espectros das várias etapas de decimação (fonte [16])

Como consideração final desta técnica, destaca-se o facto de haver imensa perda de informação devido à quantidade de amostras que são descartadas.

### 3.2.3 Interpolação

A figura 3.5 representa a conversão puramente digital do diagrama de blocos b) e mostra as várias etapas temporais dos sinais que passam por um conversor de frequências de amostragem. Na figura 3.5 a) encontra-se o sinal discreto de entrada que se pretende converter, seguidamente são adicionados zeros ao sinal discreto a fim de aumentar a sua frequência de amostragem figura 3.5 b). No entanto este sinal necessita de uma filtragem passa-baixo para se obterem os novos valores (interpolação) correspondentes ao sinal contínuo figura 3.5 c). Resta converter o sinal discreto, que se encontra numa escala temporal mais fina, para um sinal contínuo figura 3.5 d). De um modo geral depois da reconstrução, o sinal de saída é amostrado assincronamente e os valores obtidos na saída correspondem aos instantes temporais mais próximos dos valores produzidos pelo filtro interpolador. Há sempre erro, pois o interruptor que faz a retenção das amostras não fecha exatamente nos instantes ideais, ou seja as amostras de saída ficam ligeiramente afastadas da sua posição relativamente à escala temporal mais fina. A solução para este problema é aumentar a quantidade de amostras ao aumentar o fator de interpolação, pois cria um espaçamento temporal menor e as amostras adjacentes passam a ficar mais juntas [8].




(a) Sinal discreto x(n)

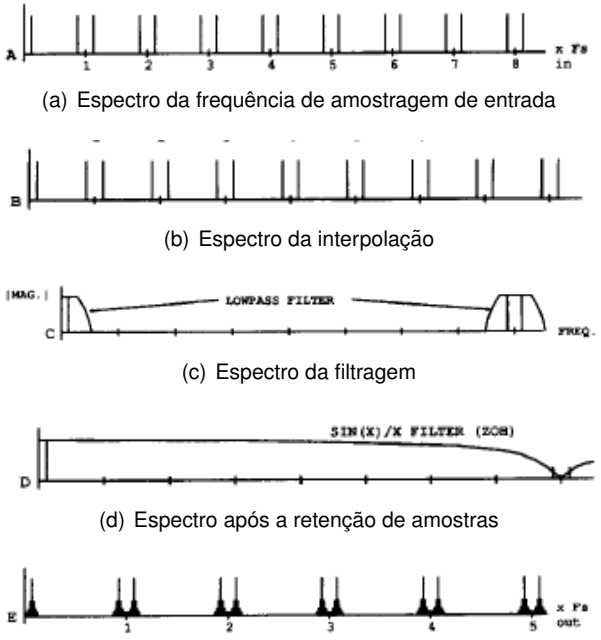
(b) Sinal discreto x(n) interpolado

(c) Sinal discreto x(n) filtrado

(d) Sinal continuo $x_c(t)$

Figura 3.5: Exemplo de interpolação no domínio do tempo (fonte [8]).

A figura 3.6 representa os espectros das etapas referidas anteriormente. A figura 3.6 a) permite ver o espectro da sequência discreta de valores $x[n]$, o aumento da frequência de amostragem com a inclusão de zeros assim como as imagens correspondentes à banda limitada do sinal após a interpolação estão presentes na figura 3.6 b), e na figura 3.6 c) está representado a forma do espectro depois da filtragem. Como no caso anterior, existe retenção de amostras e o seu espectro está representado na figura 3.6 d). Na figura 3.6 e) está representado o espectro da sequência discreta de saída. O filtro de interpolação presente na figura 3.6 c) elimina todas as imagens do sinal exceto as da nova frequência de amostragem e a retenção de amostras atenua as restantes imagens do filtro digital, portanto o fator limitativo vai ser na frequência dos 20 kHz que representa o pior caso (em áudio digital) e vai determinar a qualidade da conversão porque essas componentes de frequência sobrepõem-se a outras posições de frequência após a amostragem.




(a) Espectro da frequência de amostragem de entrada

(b) Espectro da interpolação

(c) Espectro da filtragem

(d) Espectro após a retenção de amostras

(e) Espectro da frequência de amostragem de saída

Figura 3.6: Conversão de frequências de amostragem no domínio da frequência (fonte [8]).

### 3.2.4 Conversão por um fator Racional L/M

Até agora verificou-se como a aumentar a frequência de amostragem por um fator inteiro $L$ ou como diminuir por um fator inteiro $M$. No entanto há situações em que vai ser necessário converter por um fator não inteiro. Nesses cenários é aplicado uma conversão racional $L/M$, onde a frequência de amostragem é primeiro interpolada por um fator inteiro $L$, depois passa por um filtro passa-baixo anti-imagens, prossegue para um filtro passa-baixo anti-*aliasing* e de seguida é decimado por um fator inteiro $M$. Como durante o processamento digital da sequência de valores discreta ambos os filtros são operados à mesma frequência de amostragem podem ser fundidos num só filtro passa-baixo que funciona como filtro anti-imagem e anti-*aliasing*. Ter um só filtro poupa recursos, suprime todas as componentes espectrais acima de $\frac{f_{velho}}{2}$ ou $\frac{f_{velho}}{2}\frac{L}{M}$ e a frequência de corte vai ser $min(\frac{\pi}{L}, \frac{\pi}{M})$. As figuras 3.7 e 3.8 demonstram o diagrama de blocos da conversão racional com e sem a fusão dos filtros passa-baixo. A interpolação precede sempre a decimação, para maximizar a largura da banda do sinal de entrada. No entanto, não passa de um modelo teórico visto ser bastante ineficiente. Por exemplo, se se considerar um rácio de conversão $\frac{5}{4}$ significa que são adicionados quatro zeros entre cada amostra de entrada, aplica-se o filtro e de seguida aproveitam-se $\frac{1}{4}$ desses valores. Para além disso, existe outro defeito que em $\frac{4}{5}$ das vezes o filtro FIR (*Finite Impulse Response* - Resposta Finita a um Impulso) está a realizar a convolução com valores nulos [1, 10, 15, 16].

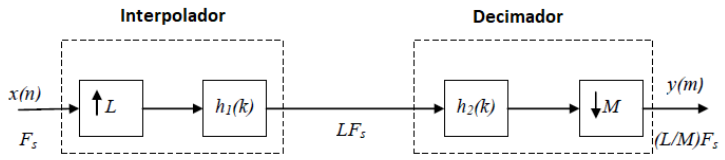



Figura 3.7: Diagrama de blocos para conversão racional de frequências de amostragem (adaptado de [10])

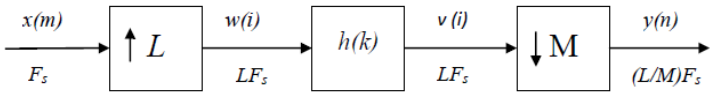



Figura 3.8: Diagrama de blocos para conversão racional de frequências de amostragem (fonte [10])

### 3.2.5 Filtragem com atrasos fracionários

Um filtro de atraso fracionário é um dispositivo que atrasa um sinal discreto por uma fração de uma amostra [17, 18]. Para esta forma de conversão de frequências de amostragem existem 3 formas de classificação de acordo com [3].

- Filtros de atraso fracionário ótimos;
- Filtros de atraso fracionário com uso de janelas de *offset*;
- Filtros polifásicos.

Uma estrutura comummente usada para filtros de interpolação é baseada em filtros FIR com atrasos fracionários. Podem-se usar estruturas de filtros FIR ou IIR (*Infinite Impulse Response* - Resposta Infinita a um Impulso) para implementar filtros com atrasos fracionários [18]. Abordar-se-ão estruturas de filtros FIR, essencialmente por serem mais simples de projetar e analisar. A ideia de filtros com atrasos fracionários está explicada em [19], basicamente reconstrói-se o sinal contínuo de banda limitada e depois de o deslocar realiza-se uma nova amostragem. Um filtro de atrasos fracionários transforma uma sequência discreta de valores no tempo $x[n]$ numa sequência de valores discretos no tempo $y[n, D]$, que aproxima $x[n]$ atrasado por um atraso no tempo $D$ dado em amostras [12]. O parâmetro $D$ geralmente não é um número inteiro. Para controlar os instantes temporais para as amostras de saída, é necessário fazer a síntese de vários filtros FIR com diversos atrasos e os coeficientes têm de ser armazenados numa *look up table* - (LUT). O instante de tempo desejado para a amostra de saída interpolada pode ser controlado ao pesar as amostras de saída dos filtros FIR pelo respetivo intervalo fracionário [20].

Em termos de definições, podemos definir a composição da sequência de amostras de saída e da resposta do filtro ideal como[12]:

$$y[n, D] = x_c(nT_s - DT_s)\,. \tag{3.5}$$

Esta equação é valida se cumprir os requisitos de Nyquist. $x[n]$ é uma sequência discreta de va-

lores amostrados a uma frequência de amostragem $f_s$ com uma correspondência única para um sinal contínuo no tempo com banda limitada $x_c(t)$ e com uma frequência de corte $f_c = \frac{f_s}{2}$.

Usando a fórmula de reconstrução 2.20 obtém-se:

$$\begin{aligned} y[n, D] &= x_c[(n - D)T_s] \\ &= \sum_{l=-\infty}^{\infty} x[l] \operatorname{sinc}\left(\frac{\pi}{T_s}[n - D - l]T_s\right) \\ &= \sum_{l=-\infty}^{\infty} x[l] \operatorname{sinc}(\pi[n - D - l]) \\ &= \sum_{l=-\infty}^{\infty} x[n - l] \operatorname{sinc}(\pi[l - D]) \\ &= x[n] * \widehat{h}(n, D)\,. \end{aligned} \tag{3.6}$$

Numa das etapas usou-se a comutatividade da convolução discreta.

$$f * g = g * f\,. \tag{3.7}$$

$$\sum_{m=-\infty}^{\infty} f[n]g[n - m] = \sum_{m=-\infty}^{\infty} f[n - m]g[n]\,. \tag{3.8}$$

As equações 3.7 e 3.8 mostram que a operação de atraso fracionário é composta pela convolução entre as amostras $x[n]$ e a resposta impulsiva ideal do filtro de atraso fracionário ideal $\widehat{h}(n, D)$ dado por:

$$\widehat{h}(n, D) = \operatorname{sinc}(\pi[n - D])\,. \tag{3.9}$$

Este filtro obtém-se usando o filtro de reconstrução ideal $h_r$ atrasado e amostrado nos instantes normalizados para $T_s = 1$ [12].

#### 3.2.5.1 Classificação de filtros com atrasos fracionários

##### 3.2.5.1.1 Filtros de atraso fracionário ótimos

Nesta categoria estão presentes todos os filtros de atrasos fracionários que foram projetados de forma a minimizar um certo critério de erro, normalmente as formulas usadas para o fazer são: ter o erro da resposta em frequência mais constante possível (*maximal flatness* - MF) e minimização do método dos mínimos quadrados (também designado por *least square error* - LS) ou maior magnitude do erro de aproximação (*minimax*). Os filtros projetados desta forma têm o comportamento mais parecido com o filtro de atraso fracionário ideal para uma determinada ordem do filtro e da aproximação do erro. No entanto, os filtros entre si são projetados independentemente pelo que a resposta global do filtro de atraso fracionário acaba por ter lóbulos largos na banda de atenuação localizados em múltiplos da frequência do sinal de entrada e nas respetivas imagens dessas componentes de frequência. A vantagem é que se a banda do sinal estiver localizada na região de aproximação onde esses filtros

foram projetados, então a resposta de magnitude desse filtro de atraso fracionário vai ser correta e melhor do que se tivesse sido desenhado pelos outros dois métodos de projeto.

#### 3.2.5.1.2 Filtros de atraso fracionário com uso de janelas de *offset*;

Esta categoria de filtros de atraso fracionário possuem uma propriedade bastante importante que é o fato de a janela de *offset* do filtro global corresponder à janela de *offset* que foi usada para fazer o projeto do filtro de atraso fracionário interpolada $L$ vezes. Para além disso é mais projetar filtro porque basta apenas definir as especificações de um segmento do filtro com janelas de *offset* para ter a resposta total do filtro. Comparativamente com a categoria anterior, filtros projetados desta forma não têm lóbulos na banda de atenuação tão largos, pelo que a resposta em magnitude é melhor, devido a terem transições mais suaves. No entanto, individualmente os filtros com janelas de *offset* têm um desempenho pior que os filtros ótimos. O uso desta categoria de filtros de atrasos fracionários prende-se com a banda do sinal, quanto mais banda ocupar maior será o incentivo, porque o comportamento na banda de atenuação é melhor e evita-se o uso de pré-filtros que é uma necessidade da primeira categoria para compensar essa decadência. Outra vantagem deste método é a capacidade para manipular a banda de transição do filtro de interpolação oferecendo maior flexibilidade de projeto. Convém que a janela usada para o projeto do filtro de atraso fracionário seja contínua no tempo para ser mais simples de usar a técnica, caso contrário é mais difícil calcular os coeficientes e a aplicação da janela de *offset* torna-se complicada. Existem algumas soluções para reduzir a complexidade computacional que envolvem o uso de aproximações com partes de polinómios para formar a resposta da janela discreta de *offset* protótipo depois de se ter feito o projeto de alguns filtros de janelas de *offset*. No final, cada um destes polinómios que estão a aproximar os segmentos das janelas vão corresponder às amostras da resposta impulsiva do filtro fracionário e possibilitam o *offset* preciso das janelas no domínio do tempo [3].

#### 3.2.5.1.3 Filtros polifásicos

Para a última categoria basta ter a noção que a decomposição de uma banda $1/L$ de um filtro de interpolação ótimo para $L$ sub-filtros dá origem a $L$ filtros com a banda toda, cada um deles com um atraso $d[m]$ diferente. Estes filtros podem ser guardados em memória para possibilitar a interpolação por um fator $L$. Se o fator de interpolação $L$ for muito elevado então é necessário armazenar respostas impulsionais bastante longas. Para além desta desvantagem, existem mais duas, a primeira está relacionada com a realização do próprio filtro que pode não ser realizável por acumulação de erros e a segunda é que só funciona para o fator de interpolação $L$ definido. Oferece contudo, um grau de liberdade que as outras duas categorias não têm, que é o compromisso entre maiores oscilações na banda de passagem e maior atenuação na banda de atenuação, ou uma banda de passagem mais nivelada e menor atenuação na banda de atenuação. Não é necessário projetar para o fator de interpolação especificado, podendo-se projetar para um fator de interpolação menor e de seguida usar estruturas de Farrow para aproximar a resposta do filtro de atraso fracionário e obter a resposta impulsiva para qualquer atraso fracionário [3].

Para aplicações que precisem de filtros de atrasos fracionários tipicamente usam-se uma destas 3 abordagens:

- Guardam-se alguns conjuntos de coeficientes para diferentes valores de atraso fracionário $d$ e escolhem-se os mais apropriados para cada caso durante a execução;
- Calculam-se os coeficientes do filtro de interpolação enquanto se realiza a filtragem;
- Usam-se estruturas de implementação especiais para atrasos fracionários que variem no tempo.

#### 3.2.5.2 Abordagens para a implementação de filtros com atrasos fracionários

##### 3.2.5.2.1 Filtros de atraso fracionário baseados em interpolação de Lagrange

O uso da interpolação de Lagrange é usada frequentemente na filtragem com atrasos fracionários, porque possui fórmulas explícitas para o calculo dos coeficientes. A interpolação de Lagrange é especialmente relevante para sinais onde o alcance de frequência é pequeno comparativamente com a frequência de Nyquist, porque possui a propriedade de ser o mais plano possível na banda de passagem, o que significa que as primeiras derivadas são nulas para uma frequência predefinida assim como o erro de interpolação. Para aplicações que necessitem de mais banda não é aconselhável, porque o aumento da ordem no filtro não aumenta a frequência útil mas aumenta a complexidade associada [18].

A fórmula para obtenção dos coeficientes do polinómio de Lagrange é obtida por:

$$h[n,d] = \prod_{k=0,k\neq n}^{N} \frac{d-k}{n-k}\, n = 0,1,2,\ldots,N\,. \tag{3.10}$$

##### 3.2.5.2.2 Implementação de estruturas baseadas em sobre-amostragem e interpolação de Lagrange

Para contrariar os erros de interpolação de Lagrange para frequências superiores aumenta-se a frequência de amostragem de entrada por um fator inteiro e usa-se posteriormente o interpolador de Lagrange. Desta forma consegue-se reduzir as especificações do erro do interpolador de Lagrange para uma fração da frequência de Nyquist [18]. O bloco que expande a frequência de amostragem é eficientemente implementado por uma estrutura polifásica (representada na figura 3.9) e apesar da frequência de amostragem de entrada ter sido aumentada por um fator $L$ o interpolador de Lagrange realiza a estimativa ao ritmo original.

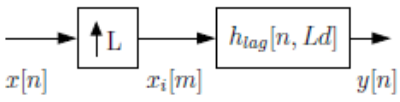



Figura 3.9: Representação de um diagrama de blocos com sobre-amostragem + interpolação de Lagrange (fonte [18])

#### 3.2.5.2.3 A estrutura de Farrow para filtragem de atrasos fracionários

A estrutura de Farrow é uma estrutura eficiente para intervalos fracionários que variem e combina o calculo dos coeficientes do filtro com a própria filtragem [18]. A ideia principal desta estrutura é aproximar a resposta impulsiva global do filtro através de um polinómio com uma ordem mais baixa e composto por várias partes que aproximam cada amostra diferente do filtro de atraso fracionário. Deste modo é mais fácil incorporar as vantagens de usar um método de projeto direto para as especificações do filtro de interpolação e simultaneamente fazer a conversão para qualquer rácio desejado. Pode-se ainda manipular a frequência de corte, as ondulações da banda de passagem e a atenuação na banda de atenuação. A única desvantagem é que para especificações diferentes do filtro é preciso voltar a calcular os coeficientes para a estrutura [3].

A estrutura de Farrow consiste no uso de filtros FIR fixos em paralelo sendo que cada coeficiente dos filtros $h[n, d]$ é dado por um polinómio em $d$, que é o intervalo fracionário. Para além disso, o controlo de $d$ é mais fácil durante a execução do que numa implementação correspondente já guardada em memória, pois a resolução de $d$ depende apenas da resolução da aritmética e não da capacidade de memória [21–23].

A saída do filtro vai ser obtida pelos valores das saídas de $M + 1$ sub-filtros. A multiplicação por $d$ na estrutura de Farrow corresponde à amostragem do sinal reconstruido $ya$ no modelo analógico.

Prosseguindo para a definição da estrutura de Farrow, tem-se segundo [20]: que se o sistema for modelado por um sistema híbrido analógico/digital (representado na figura 3.10), onde o sinal é reconstruido por um conversor DA, $y_a(t)$, a partir de um filtro de reconstrução, $h_a(t)$, e depois amostrado a $t_l = (n_l + d_l)T_e$ chega-se às amostras interpoladas de saída, $y(l)$, dadas pela equação 3.11.

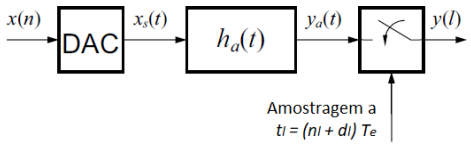



Figura 3.10: Modelo híbrido analógico/digital, Adaptado de: [20]

$$y[l] = y_a(t_l) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} x(n_l - k)\, h_a((k + d_l)T_e)\,. \tag{3.11}$$

A equação 3.11 é obtida ao assumir que o filtro de reconstrução $h_a$ é nulo fora do intervalo $-\frac{N\,T_e}{2} \leq t \leq \frac{N\,T_e}{2}$.

$$h_a((k + d_l)T_e) = \sum_{m=0}^{M} \widehat{c}_m(k)\, d_l^m\,. \tag{3.12}$$

A equação 3.12 é característica de um filtro de interpolação polinomial. A resposta impulsiva $h_a(t)$ é formada por polinómios a cada período de amostragem de entrada $T_e$. O $k$ é dado por $k = -\frac{N}{2}, -\frac{N}{2} + 1, \ldots, \frac{N}{2}$ e o intervalo fracionário varia entre $d = \in [0, 1]$. Os termos $\widehat{c}_m(k)$ são os coeficientes dos

polinómios para o intervalo $k$ de duração $T_e$ e onde $M$ é o grau do polinómio.

A equação 3.13 resulta da substituição de 3.12 em 3.11.

$$y[l] = y_a(t_l) = \sum_{m=0}^{M} v_m(n_l)\, d_l^m \,. \tag{3.13}$$

Onde,

$$v_m(n_l) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} x(n_l - k)\, \widehat{c}_m(k) \,. \tag{3.14}$$

são as amostras de saída de $M+1$ filtros paralelos FIR com uma ordem $N$ e com coeficientes $\widehat{c}_m(k)$. As funções de transferência se forem causais são dadas pela equação 3.15.

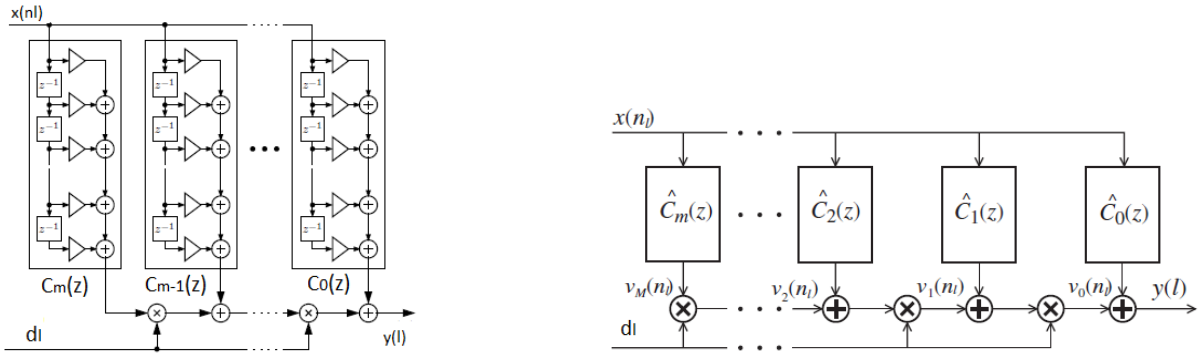
$$\widehat{C}_m(z) = \sum_{k=0}^{N+1} \widehat{c}_m(k - \frac{N}{2})\, z^{-k} \,. \tag{3.15}$$

A complexidade de computação para além de depender do número de sub-filtros, depende também da ordem do polinómio e é dada por $(M+1)(N+1)$ somas e $(M+2)(N+2)$ multiplicações. Pode-se alterar ligeiramente a estrutura de modo a aproveitar simetrias e chega-se à estrutura modificada de Farrow. A estrutura modificada de Farrow obtém-se obrigando o intervalo fracionário a ser simétrico em relação a zero $d \in [-1/2, 1/2]$ pondo os filtros FIR com fase linear e reduzindo a complexidade para $(M+2)N$ adições e $(M+2)(N/2)$ multiplicações [18].

Em relação a estruturas de Farrow, existem as seguintes:

- Estrutura de Farrow (representada na figura 3.11);
- Estrutura de Farrow modificada (representada na figura 3.12);
- Estrutura de Farrow transposta;
- Estrutura de Farrow prolongada.

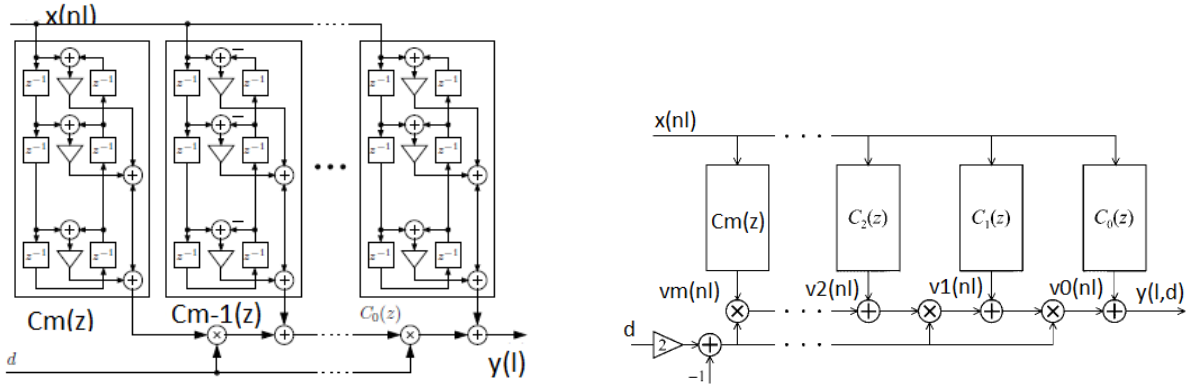
Destas, a estrutura de Farrow modificada e transposta são as mais usadas. A estrutura de Farrow modificada é usada para interpolação, devido a ter boas propriedades anti-imagem e a estrutura de Farrow transposta é usada para decimação, devido a ter boas propriedades anti-aliasing. Ocasionalmente é usada a estrutura de Farrow prolongada tanto para interpolação como para decimação [20, 22].




(a) Estrutura de Farrow detalhada

(b) Estrutura de Farrow

Figura 3.11: Estrutura de Farrow, Adaptado de:[20, 22].




(a) Estrutura de Farrow modificada detalhada

(b) Estrutura de Farrow modificada

Figura 3.12: Estrutura de Farrow modificada, Adaptado de:[22, 23] .

### 3.2.6 Conversão de frequência de amostragem assíncrona

A maneira alternativa de fazer a conversão de frequências de amostragem é transformar a frequência de amostragem de entrada numa frequência de amostragem de saída com recurso a compressores e expansores de frequências. Com esta técnica, as principais precauções envolvem a atenuação das imagens do sinal bem como o *aliasing*. Os conversores assíncronos têm maior relevo se o rácio for formado por dois inteiros muito grandes, se o rácio variar no tempo ou se o rácio não for formado por um rácio de números inteiros. Os conversores assíncronos são uma generalização dos conversores síncronos [18].

#### 3.2.6.1 Abordagens para conversões de frequência de amostragem assíncronas

##### 3.2.6.1.1 Modelo híbrido / Discreto no tempo

Para o modelo híbrido usa-se um conversor DA e um filtro de reconstrução, onde um sinal discreto do tempo $x[n]$ com uma determinada frequência de amostragem de entrada $f_i$ é convertido para um sinal contínuo no tempo, para obter o sinal discreto de saída amostra-se com a frequência de amostragem $f_o$. Para este tipo de sistemas consegue-se defini-los completamente a partir da resposta impulsiva do filtro $h(t)$. Referir brevemente que pode ocorrer erros com as imagens do sinal ou artefactos de

*aliasing*. Os erros de imagem são causados porque a banda de atenuação não é perfeita e tem alguma amplitude, sendo que quando se faz a amostragem as imagens do sinal podem aparecer no espectro do sinal discreto. Os artefactos de *aliasing* aparecem porque a energia da banda base do sinal pode estar com *aliasing* e aparece espelhada para o espectro do sinal discreto.

#### 3.2.6.1.2 Conversores assíncronos de frequências de amostragem baseados em interpolação polinomial no domínio do tempo

Nesta subsecção de conversores de frequências assíncronos estão presentes interpoladores analógicos, mais concretamente polinómios de interpolação de Lagrange ou as B-splines. Sobre estes interpoladores a principal característica é que as bandas de atenuação apresentam lóbulos largos que decrescem lentamente com o incremento da ordem do filtro.

#### 3.2.6.1.3 Combinação de um conversor de frequências de amostragem racional com interpolação polinomial

Normalmente na combinação de um conversor racional com interpolador polinomial costuma-se usar um bloco para aumentar a frequência de amostragem de entrada e aplicar de seguida o filtro passa-baixo para realizar a interpolação polinomial no domínio do tempo com o auxílio, por exemplo do método de Lagrange ou de B-spline. A principal vantagem de usar esta configuração é que as especificações que o filtro tem que cumprir passam a ser menos exigentes e além disso também relaxa a complexidade computacional.

#### 3.2.6.1.4 Filtros baseados em interpolação polinomial usando a estrutura de Farrow

Tal como na filtragem de atrasos fracionários também a conversão assíncrona de frequências de amostragem utiliza a estrutura de Farrow e todas as suas variantes, a diferença está na função de custo. Para uma implementação de interpolação polinomial, a resposta impulsiva é dada por partes de polinómios para formar uma resposta global que aproxima a resposta em frequência ideal contínua do filtro [18].

## 3.3 Filtragem

O intuito desta secção é discutir que métodos de filtragem digital existem e abordar algumas estruturas que aplicam eficientemente as técnicas abordadas anteriormente. Para a conversão de frequências é possível usar filtros FIR ou IIR, no entanto na maior parte dos casos os filtros FIR de fase linear são os escolhidos.

### 3.3.1 Resposta finita a um impulso

Na maioria das situações um filtro FIR é preferível a um filtro IIR porque [10, 24]:

1. A resposta em frequência de um filtro FIR é estritamente linear. No caso de um filtro IIR isso não se verifica. Para além disso as não-linearidades de um filtro IIR são inversamente proporcionais à sua seletividade;

2. Para um filtro FIR é possível usar *Fast Fourrier Transform* (FFT), enquanto que para um filtro IIR isso não é possível;

3. A estrutura de um filtro FIR não é recursiva ao contrário de um filtro IIR. Na prática significa que os erros de aritmética de precisão finita são pequenos para os filtros FIR, enquanto que para os filtros IIR por serem recursivos podem ocorrer oscilações parasitas;

4. São mais fáceis de implementar e têm fase linear na banda de frequência de interesse, assim a relação entrada-saída dos conversores racionais não é distorcida.

Existem várias técnicas para a implementação de filtros FIR. Estes filtros são ótimos no que diz respeito à banda de transição entre a banda de passagem e a banda de atenuação para especificações de ondulações e frequências de corte. Podem também ser projetados para ter ondulações mínimas na banda de passagem e atenuação, e com uma banda estreita para a banda de transição sendo o *trade-off* uma resposta impulsiva mais longa. Deste modo comprova-se que os erros provocados pelas não-linearidades da fase podem ser anulados e os erros de distorção de amplitude podem ser reduzidos consideravelmente. Por outro lado, geralmente os filtros FIR precisam de mais cálculos para atingir a mesma precisão de uma resposta de amplitude que um filtro IIR [2]. A última observação é que os filtros FIR são estáveis e não possuem estados internos que dependam de saídas prévias do sistema, o que causa o seguinte [12]:

1. O número de amostras atrasadas podem ser partilhadas por vários filtros;

2. As especificações do filtro podem ser alteradas durante a execução sem causar erros transitórios.

### 3.3.2 Comb

O filtro passa-baixo FIR mais simples de se implementar é o filtro de média móvel (habitualmente conhecido como *moving average* - MA) também designado por filtros comb [25]. Os filtros comb costumam ser usados nos primeiros andares de decimação por não realizarem multiplicações e por não ser necessário armazenar coeficientes. Contudo existem dois problemas dos filtros comb:

1. Os filtros comb trabalham à frequência máxima de amostragem antes da decimação ocorrer, gastando muita energia;

2. A resposta em frequência do filtro comb não satisfaz as especificações.

Pode-se usar uma estrutura recursiva chamada *cascated-integrator comb* - (CIC) formada por um integrador, um bloco para reduzir a frequência de amostragem e um diferenciador para melhorar as características da resposta em frequência. As partes principais desta configuração encontram-se em cascata. O integrador funciona à frequência de amostragem maior e, para uma ordem de filtro e factores de decimação elevados, ocupa uma área grande e tem um consumo de energia elevado. O andar

de decimação funciona à frequência de amostragem menor. Os filtros CIC são uma classe de filtros digitais FIR eficientes em termos de hardware e de fase linear, e possuem o mesmo número de andares de integradores e de filtros comb. A resposta em frequência pode ser afinada escolhendo o equilíbrio entre o número de pares de integradores em cascata e de filtros comb. Na banda de passagem não é plano o que pode ser um inconveniente nalguns cenários, mas pode ser minimizado ao usar filtros de compensação. Se o número de andares for grande, os filtros CIC introduzem uma queda na banda de passagem que depende do rácio de decimação [26].

### 3.3.3 Resposta Infinita a um Impulso

Os filtros IIR não conseguem obter uma fase completamente linear. Os filtros IIR têm características de amplitude excecionalmente boas, mas com alguns erros de interpolação devido a não-linearidades. Em termos de complexidade computacional usam menos recursos que os filtros FIR, por terem uma implementação recursiva [2]. Os filtros IIR podem ser implementados como um filtro passa-tudo perfeito. São mais difíceis de controlar, são mais subjetiveis a ter erros transitórios e são mais propensos a terem problemas de instabilidade [12, 18].

### 3.3.4 Polifásicos

Esta estrutura existe para otimizar as operações realizadas pelos filtros e para melhorar a complexidade computacional. Com esta estrutura evitam-se as multiplicações por zero na interpolação e a necessidade de descartar amostras na decimação. Nesta estrutura metem-se em paralelo vários filtros FIR alcançando-se deste modo uma decomposição polifásica. A função de transferência é dividida em $L$ ou $M$ sub-filtros, (designados por componentes polifásicos) e no final somam-se essas parcelas todas para se ter a resposta global da função de transferência [10]. A estrutura polifásica pode ser aplicada diretamente para filtros FIR. No caso dos filtros IIR é necessária uma transformação para ser equivalente e só depois se pode realizar a decomposição polifásica [27].

A definição de filtros polifásicos apresentada é a de [10], para um filtro $H(Z)$ de ordem $N$ e as suas respectivas decomposições $E_0(Z)$ e $E_1(Z)$:

$$H(Z) = h(0) + h(1)Z^{-1} + h(2)Z^{-2} + h(3)Z^{-3} + \ldots + h(N-2)Z^{-(N-2)} + h(N-1)Z^{-(N-1)}\,. \quad (3.16)$$

$$\begin{aligned} H(Z) =& h(0) + h(2)Z^{-2} + \ldots + h(N-2)Z^{-(N-2)} \\ &+ Z^{-1}(h(1) + h(3)Z^{-2} + \ldots + h(N-1)Z^{-(N-2)})\,. \end{aligned} \quad (3.17)$$

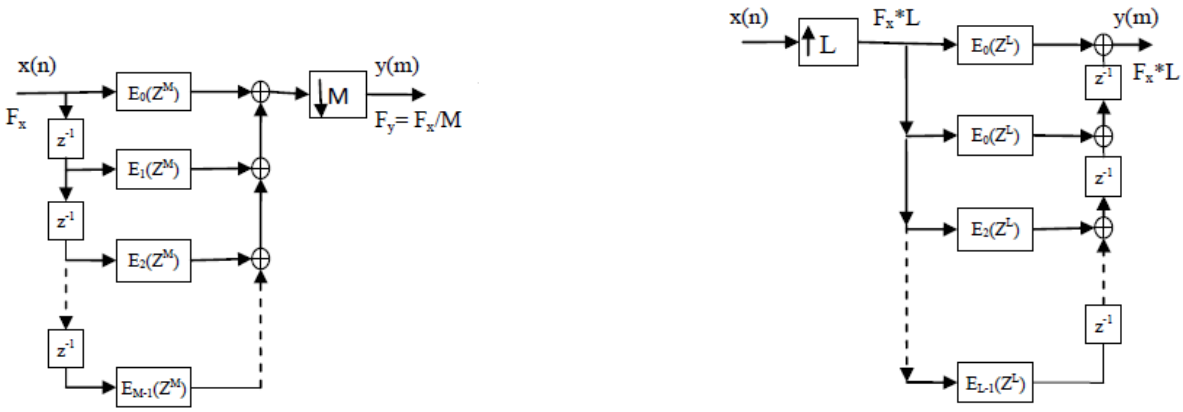
$$E_0(Z^2) = h(0) + h(2)Z^{-2} + \ldots + h(N-2)Z^{-(N-2)}\,. \quad (3.18)$$

$$E_1(Z^2) = h(1) + h(3)Z^{-2} + \ldots + h(N-1)Z^{-(N-2)}\,. \tag{3.19}$$

Portanto, pode-se decompor um filtro mais complexo em sub-filtros mais simples. Por exemplo a equação 3.16 pode ser re-escrita da seguinte forma:

$$H(Z) = E_0(Z^2) + Z^{-1}E_1(Z^2)\,. \tag{3.20}$$

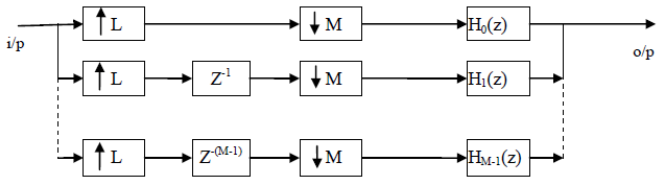
Com estas estruturas podem-se realizar cálculos mais eficientemente para decimação e interpolação figura 3.13 ou para uma implementação racional $\frac{L}{M}$ figura 3.14:



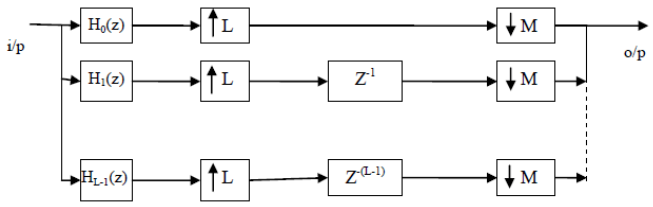

(a) Decimação polinomial.

(b) Interpolação polinomial.

Figura 3.13: Estruturas polifásicas para decimação e interpolação (fonte [10]).




(a) Estrutura polifásica racional $M > L$




(b) Estrutura polifásica racional $L > M$

Figura 3.14: Estrutura polifásica racional (fonte [10]).

## 3.4 Conversores síncronos

Esta subsecção apresenta diversas técnicas relacionadas com conversão síncrona. O foco dos autores nas modificações dos conversores síncronos são a simetria dos coeficientes dos filtros, os filtros

utilizados para a filtragem ou a estrutura dos próprios filtros. Noutros casos, podem ser aplicados abordagens menos conhecidas como o armazenamento dos dados ou aplicação de transformadas para a realização da filtragem.

**Rabiner [28],Crochiere and Rabiner [11], Schafer and Rabiner [16]** Estes trabalhos foram agrupados devido às semelhanças entre eles. Têm autores comuns, pelo que a distinção está nos objetivos. Estão explicados todos os aspetos, desde a fase introdutória onde se definem os conceitos de processamento digital de sinais e de conversores de frequências de amostragem, até à parte mais complexa onde se analisam os filtros de interpolação, as suas estruturas e a possibilidade de haver mais andares no sistema. Pode ser visto como o ponto de referência para perceber a teoria que engloba sistemas com mais de um sinal de relógio e as técnicas digitais utilizadas.

**Bi and Li [29]** Este artigo apresenta um método alternativo de fazer uma conversão síncrona ou racional, como os autores descrevem, ao manipular a transformada discreta do cosseno. Para além disso, afirmam que a solução proposta é superior em termos de complexidade computacional e erros de desempenho na conversão de frequências de amostragem. Concluem dizendo que conseguem uma precisão muito elevada de conversão, porque usam um processo de filtragem com uma banda de passagem ideal e uma banda de transição nula. Acrescentam mais dois tópicos, o primeiro é que os erros de conversão são minimizados com operações de sobreposição e o segundo é que com algoritmos de transformadas rápidas reduz-se bastante a complexidade computacional, quando comparado com os métodos mais eficientes no domínio do tempo com filtragem passa-baixo e implementação baseada em várias fases.

**Djordje Babic [21]** Este artigo apresenta uma forma para reduzir a complexidade da implementação de um conversor de frequências de amostragem, quando se tem um rácio de dois números inteiros muito grandes, usando filtros de interpolação baseados em polinómios. Para além disso, encontra a relação entre os vários filtros de interpolação baseados em polinómios (estrutura de Farrow e as suas modificações) e modelos de filtros com várias fases FIR. Acerca de conversores de frequências de amostragem por um rácio racional, para qualquer filtro de interpolação baseado em polinómios com uma ordem arbitrária existe sempre um filtro FIR equivalente invariante no tempo que pode ser usado para combater os problemas de espelhamento espectral no modelo analógico/digital. Sobre as estruturas de Farrow e as suas modificações refere apenas que podem ser consideradas uma alternativa viável à implementação de filtros lineares que variam periodicamente no tempo (LVPT). Termina, fazendo observações acerca da aplicabilidade dessas relações, em por exemplo projeto de filtros, redução de complexidade de implementação e ainda análise da distorção da resposta.

**Takahashi and Daisuke Takago [30]** Este artigo apresenta uma técnica alternativa de conversão síncrona por um rácio $L/M$ em que tanto $L$ como $M$ são números compostos (inteiros positivos obtidos através do produto de dois inteiros positivos mais pequenos), originando uma estrutura com vários andares de conversão em que cada andar é formado por um um segmento onde se aumenta a frequência de amostragem e outro onde se diminui. Esses andares realizam conversões pequenas com as características do filtro passa-baixo relaxadas, ou seja, a banda de transição não é tão abrupta, reduzindo a complexidade de computação.

**Bi [31]** Este artigo apresenta requisitos para atrasos no sistema e mostra como poupar, por um fator $L$, o número de dados que são precisos armazenar. Derivam duas regras de transformação para minimizar os tais atrasos. Apresentam um modelo realizado em hardware para demonstrar a utilidade da abordagem proposta. É especialmente útil para abordagens de multiplexagem no tempo, porque faz com que sejam precisas menos operações aritméticas comparativamente com uma implementação direta ou uma estrutura polifásica.

**Gustafsson and Johansson [32]** Este artigo mostra que conversões racionais baseadas em filtros FIR podem ser substituídas por uma matriz constante. Reduzem a complexidade aritmética quando comparada com blocos separados de múltiplas multiplicações comuns (MMC). Mostram que o número de registos varia entre a forma direta e a forma direta transposta de filtros FIR. A estrutura com o número mínimo de registos depende se a frequência de amostragem aumentar ou diminuir.

**Russell [33]** Este artigo mostra que conversões racionais podem ser implementadas eficientemente com filtros IIR, usando filtros polifásicos para aproximar a resposta em frequência de um filtro passa-baixo ideal. Analisam separadamente os custos de implementação para a parte recursiva e não recursiva e chegam à conclusão que é possível ter um ganho $LM/(L+M-1)$ na parte recursiva, mas só se L e M forem maiores que 1. Reconhecem que a abordagem não é ótima, mas que nalguns cenários pode ser útil, pois é mais fácil de projetar este tipo de filtros.

**Robert Bregović and Lim [24]** Este artigo apresenta uma estrutura eficiente para implementar um filtro FIR com fase linear e uma ordem arbitrária $N$ para conversão de frequências de amostragem. Os autores tentaram aproveitar ao máximo a simetria dos coeficientes da fase linear do filtro e tentaram reduzir o número de atrasos. Para a realização tiveram em conta que na interpolação a entrada deixa de ser zero a cada $L$ amostras e no caso da decimação a saída é avaliada a cada $M$ amostras. Desta forma, conseguem reduzir as multiplicações por amostra de saída por um fator de dois comparativamente com os métodos tradicionais polifásicos. Fornecem ainda umas diretrizes para implementar corretamente um filtro FIR com fase linear e uma ordem $N$ para um rácio arbitrário $L/M$ e finalizam com uma análise à complexidade da implementação.

Destacaram cinco pontos de interesse:

1. A abordagem proposta funciona melhor para filtros com maior ordem;
2. Pode ser implementado diretamente em sistemas com múltiplos andares;
3. Pode ser combinado com o produto de uma matriz constante [32];
4. É melhor usar a abordagem proposta do que uma implementação polifásica, para qualquer valor de $L, M, N$, tendo atenção que $L, M$ não podem ser muito elevados sob pena de o conversor de frequências de amostragem racional ser irrealizável;
5. Convém escolher uma ordem $N$ que divida sensivelmente a matriz de entrada-saída ao meio.

**Hsiao [34]** Este artigo apresenta uma extensão do que normalmente é feito para conversões eficientes de rácios inteiros que consiste no uso de um vetor de filtros polifásicos. Neste modo de fazer a conversão, o lado da frequência de amostragem elevada é decomposto para os seus filtros polifásicos onde

posteriormente pode ser movido para frequências de amostragem reduzidas sem alterar as funções dos filtros. O autor deste trabalho refere que a complexidade de um filtro FIR é reduzida por um fator igual à frequência de amostragem. Refere ainda que para sistemas com rácios $L/M$ onde há um interpolador que realiza a interpolação $1 : L$, seguido de um decimador que realiza a decimação $M : 1$ existem 3 frequências de operação $F$, $LF$, ou $F/M$, mas que se usar um vetor de filtros polifásicos a funcionar a uma frequência de amostragem de $LF$, pode ser implementado tanto na entrada como na saída a uma frequência de amostragem reduzida. Conclui comparando o modelo proposto por uma implementação direta e por uma matriz de filtros polifásicos. Reduz a computação por um fator $LM$, e por um fator $M$ (ou $L$) respetivamente, enquanto a relação entrada-saída for assegurada.

**Robert Bregovic and Lim [35]** Este artigo é semelhante ao trabalho [24], mas com algumas diferenças. Apresenta uma forma de implementação de filtros FIR com fase linear para rácios arbitrários $L/M$ em que os pressupostos assumidos foram os mesmos: coeficientes simétricos, amostras diferentes de zero na entrada a cada $L$ amostras, e amostras que só precisam de ser avaliadas na saída a cada $M$ amostras. Considerando esses pressupostos conseguiram obter uma estrutura eficiente para filtros FIR com fase linear.

**Vandita Singh [36]** Neste artigo aproveitam a simetria dos coeficientes da fase linear dos filtros FIR para implementar uma estrutura eficiente para conversão racional $L/M$ de frequências de amostragem, onde $L$ é um fator de escala para aumentar a frequência e $M$ é idêntico mas para a diminuir. Com o uso desta técnica conseguem diminuir o número de multiplicadores comparativamente com um implementação polifásica tradicional, porque por cada amostra de saída são precisas menos multiplicações.

**Vandita Singh [37]** Este artigo apresenta uma estrutura eficiente para conversão racional $L/M$ de frequências de amostragem, ao introduzir um filtro FIR entre o bloco que aumenta a frequência e o bloco que a diminui, evitando o espelhamento espectral e as imagens da banda do sinal que se quer converter. Os autores aproveitam a simetria dos coeficientes para reduzir o número de multiplicações por amostra de saída. A proposta apresentada reduz a quantidade das multiplicações comparativamente com uma implementação polifásica ou com a técnica de [24] e avaliam a complexidade da implementação e eficiência em relação a outras técnicas.

**Nagrale [38]** Este artigo apresenta uma técnica que melhora o sistema de decimação. Separam esse fator em dois inteiros que representam andares na estrutura no sistema e que são colocados nas regiões onde a frequência de amostragem é mínima. Com esta técnica consegue-se reduzir o armazenamento dos dados e a carga de computação. Consegue-se também ocupar menos área e o consumo de energia é menor, devido às vantagens apresentadas anteriormente, satisfazendo os requisitos exigidos para Rádio Definido por Software (RDS).

**Dolecek [39]** Este artigo apresenta uma modificação aos filtros Cascaded-Integrator-Comb (CIC) para fazer conversões de frequências de amostragem racionais para RDS. Este trabalho também faz as mesmas alterações ao sistema de decimação de [38], enquanto que o filtro é baseado na forma triangular da resposta ao impulso de um filtro CIC, juntamente com a expansão do filtro de cosseno e o filtro de compensação baseado no seno. A conversão é eficiente porque utilizam só adições e

subtrações e a melhoria na banda de passagem, que melhora o *Signal to Noise Ratio* (SNR), torna o método proposto atrativo para aplicações de RDS.

**Dolecek and Mitra [40]** Este artigo apresenta uma conversão de frequências de amostragem racional $L/M$, onde tanto $M$ como $L$ são números mutuamente primos, com um filtro CIC baseado na forma triangular e a expansão do filtro de cosseno. O filtro de expansão é escolhido de forma a atenuar os efeitos do espelhamento espectral. Existem dois andares de decimação. Existem três estruturas compostas por integradores ou filtros comb que oferecem menos complexidade e melhor desempenho que o trabalho [39].

**Chunduri SreenivasaRao and VenkataRao [14]** Este artigo apresenta uma forma alternativa de fazer a conversão de frequências de amostragem. Em vez de fazer a análise no domínio do tempo com o uso típico de filtros polifásico, apresenta um processo de como proceder se se optar por uma abordagem no domínio da frequência, onde se pode poupar alguns recursos de computação. Recorrem às propriedades da FFT para resolver alguns problemas de implementação. Fazem ainda um balanço entre os resultados das simulações, chegando à conclusão que as diferenças da taxa de distorção harmónica (TDH) e SNR são negligenciáveis..

## 3.5 Conversores assíncronos

Tal como foi feito no capítulo anterior irá ser feita uma revisão do estado da arte sobre conversores de frequências de amostragem assíncronos e que contribuíram para o entendimento e desenvolvimento deste trabalho.

**Adams and Kwan [8]** Este artigo apresenta um conversor de frequências de amostragem assíncrono. Os autores apresentam primeiro a teoria por detrás destes conversores recorrendo a um modelo de processamento de sinal baseado em amostras de entrada altamente interpoladas. Os autores também facultam uma técnica para as medições do instante de chegada do sinal do relógio serem mais precisas recorrendo a de uma malha de fase síncrona. Esta solução tem o benefício acrescido de reduzir o jitter, o que permite selecionar o filtro polifásico correto para as amostras de saída. Outro aspeto que esta solução permite é a variação da frequência de amostragem por um rácio de 2:1 para qualquer um dos sentidos. Concluem afirmando que é adequado para uma implementação *Very Large Scale Integration* (VLSI) com um desempenho de 16-bits, usando quantidades modestas de *Random Access Memory* (RAM), de *Read Only Memory* (ROM) e de hardware do filtro digital.

**Medan and Shvadron [41]** Este artigo apresenta uma forma de fazer a conversão entre a frequência de amostragem de entrada e a frequência de amostragem de saída mesmo que os sinais de relógio desses sistemas estejam dessincronizados. A relevância do algoritmo está no facto de o sinal ser produzido num determinado sítio e ter que ser produzido noutro equipamento. Nessas situações, um pequeno desvio no sinal de relógio pode causar erros de sincronização. O algoritmo é simples e consome menos de um mega instruções por segundo (MIPS).

**Pallab Midya and Schooler [42]** Este artigo apresenta uma necessidade de fazer a sincronização dos dados de entrada com o relógio interno do sistema, o qual não tem muito jitter e é responsável

por gerar a modulação por largura de pulso (MLP) de saída. Para isso é necessário um conversor assíncrono de frequências. A forma como o autor deste trabalho o implementou foi com sinais sobre-amostrados e dessa forma conseguiu-se reduzir a quantidade de dados para armazenar e a computação deixa de ser tão exigente. Fizeram numa placa de circuito integrado (CI) o amplificador digital e o conversor de frequências de amostragem. O funcionamento do amplificador digital não é limitado pelas frequências de amostragem.

**Katsutnata and Hamada [43]** Este artigo apresenta os resultados da aplicação de um processador de sinais digitais na conversão de frequências de amostragem para áudio. Discutem a construção por software e ainda o processamento de sinais. As medições foram realizadas sobre uma referência obtida por um gerador de um seno digital e ainda usando um filtro rejeita banda. Para a realização da abordagem proposta, tiveram que usar dois processadores digitais de sinal. Para além disso, a interpolação é feita ao longo de três andares e há um andar onde usam um filtro FIR de interpolação. Concluem, dizendo que segundo as medições realizadas o método tem um SNR de 16-bits.

**Lagadec et al. [7]** Este artigo apresenta uma forma de fazer uma conversão arbitrária de frequências de amostragem baseado em vários andares de filtros interpoladores. O controlo é feito no domínio do tempo com sinais externos provenientes do sinal de relógio que está a servir para a amostragem. Conseguem obter um desempenho de 16-bits mesmo com algumas oscilações e rácios de conversão que variam no tempo.

**Russell and Beckmann [44]** Este artigo apresenta um método inovador de fazer uma conversão arbitrária de frequências de amostragem onde simulam uma conversão contínua por uma conversão discreta. Utilizam uma estrutura paralela onde cada ramo contém um filtro que varia no tempo e conjunto com um filtro recursivo. A técnica é eficiente em termos de computação e ocupa apenas uma fração da memória dos métodos tradicionais, porque os coeficientes são calculados recursivamente o que elimina a necessidade de os armazenar em memória. Destacam que o método requer pouca memória e nalguns casos é mais eficiente que os métodos tradicionais. Os filtros são facilmente projetados usando métodos convencionais, por exemplo, Butterworth ou Chebychev. Referem que, se os coeficientes calculados recursivamente tiverem algum erro, por exemplo, de arredondamento esses erros são propagados. Para além disso, os filtros não têm fase linear, ou seja introduzem atrasos de grupo.

**Ramstad [45]** Este artigo apresenta dois métodos para fazer uma conversão arbitrária de frequências de amostragem dependendo se o rácio é fixo ou se as frequências de amostragem têm ligeiras flutuações. No primeiro caso baseiam-se numa implementação baseada na transformada do impulso-invariante de um filtro analógico. No segundo caso, combinam filtros de interpolação FIR para conversões inteiras ou racionais com interpoladores de Lagrange de ordem baixa. Ambas as propostas sugeridas podem ser facilmente aplicáveis tanto em software como em hardware.

**Babic and Renfors [46]** Este artigo apresenta uma forma eficiente de reduzir as imagens e os efeitos de espelhamento, sendo essas as características que diferenciam este método para os restantes. Apresentam ainda duas estruturas equivalentes, a primeira é baseada na estrutura de Farrow modificada e a segunda é baseada na estrutura modificada de Farrow transposta. Ao utilizar estas estruturas

conseguem passar a maioria das operações para uma frequência de amostragem inferior.

**Yong-jian Xu and Shen [47]** Este artigo apresenta uma arquitetura de um filtro de interpolação polifásico modificado para um conversor de frequências de amostragem arbitrário através do endereçamento flexível dos coeficientes, permitindo desta forma reduzir o número de coeficientes guardados em memória. Usando uma implementação em série conseguem reduzir o número de somadores e de multiplicadores. Os resultados mostram que é possível controlar a atenuação na banda de atenuação, ao variar a quantidade de vezes que o filtro passa-baixo é interpolado.

**Ramstad [48]** Este artigo apresenta uma forma de fazer a interface entre dois sistemas com diferentes frequências de amostragem e fá-lo independentemente do rácio resultante, podendo mesmo ser irracional ou variante no tempo. Refere que para realizar um rácio irracional, os coeficientes do filtro digital não vão ser periódicos e que existem duas soluções viáveis para este problema. A primeira solução é pré-calcular os coeficientes todos. Esta solução é válida porque a resolução necessária é finita. De notar que é necessária uma memória muito grande para armazenar os coeficientes. A segunda solução implica calcular os coeficientes durante a execução com filtros FIR e IIR. A grande desvantagem desta solução é que é computacionalmente intensiva para o sistema. Existem, implementações que são uma combinação desses casos mais extremos. Essas implementações também são apresentadas e discutidas. O que o autor sugere são implementações baseadas em filtros de reconstrução analógica onde os coeficientes derivados dos filtros digitais são funções da distância entre a entrada e saída ou, alternativamente, interpoladores digitais usados juntamente com esquemas de interpolação analógica para descobrir os valores desejados entre as amostras de saída dos interpoladores digitais.

**S. Cucchi and Sicuranza [49]** Este artigo revê o trabalho [48] nomeadamente a estrutura híbrida com a amostragem a ser feita pela função B-spline. Usam um circuito para fazer a conversão em tempo real das frequências tipicamente usadas em áudio digital.

**Brunel Happi Tietche and Denby [50]** Este artigo apresenta um circuito numa FPGA para amostragem de rácios arbitrários para bandas de sinal situadas nas baixas frequências para as muito altas frequências de amostragem, sendo direcionado para aplicações de RDS. É adequado para implementações em FPGA que tenham restrições de relógio, porque o método permite controlar o *Spurious Free Dynamic Range* (SFDR), enquanto cria uma interface entre o domínio do relógio da entrada e saída sem precisar de sinais de relógio adicionais.

**Franck and Brandenburg [51]** Este artigo propõe um método de projeto para conversores de frequências de amostragem assíncronos que permite otimizar toda a estrutura do filtro. Incluem também uma descrição analítica da resposta contínua do sistema e uma forma de minimizar o erro na perspetiva de Chebyshev. Em relação às normas de erro, filtros projetados desta forma demonstram menos erros que métodos atualmente existentes. Em relação às melhorias dependem de vários fatores como a ordem do filtro protótipo, a banda do sinal, o rácio de interpolação e a ordem do interpolador. Os filtros protótipos são implementações de filtros eletrónicos. São usados como modelo base, para produzir implementações de filtros modificados para aplicações específicas.

**Franck [52]** Este artigo é a continuação do trabalho [51], onde identificaram que uma das falhas nesse método é que o método de Lagrange não funcionava muito bem como filtro de amostragem

contínuo no tempo, o que por sua vez afetava negativamente a qualidade. O artigo define um objetivo para filtros de amostragem contínuos no tempo, adaptados tanto para sobre-amostragem como para o esquema de otimização global. Conseguem implementações com complexidade reduzida e melhor qualidade para diversas especificações de projeto com um método chamado otimização da atenuação da banda de imagens.

**Franck [12]** Este artigo apresenta algoritmos e implementações para SRCs.

Fizeram-no tentando cumprir três objetivos:

1. Introduzem representações analíticas para a resposta em frequência contínua de várias estruturas de conversores assíncronos. Derivaram expressões para o polinómio interpolador de Lagrange, estruturas de Farrow e estruturas formadas por sobre-amostragem por números inteiros e amostragem contínua no tempo. Evidenciaram as propriedades e comportamentos de cada estrutura, e como são refletidas na implementação;

2. Determinam os coeficientes com uma escolha propositada dos critérios de implementação para as estruturas. Particularizam para uma implementação que produza resultados ótimos, escolhendo normas de erro e restrições rigorosas;

3. A variedade de estruturas para implementação e os seus parâmetros dificultam a escolha de um algoritmo de SRC para uma dada aplicação. Portanto, é feita uma análise do desempenho e uma comparação das diferentes estruturas.

**Vesma and Saramaki [53]** Este artigo apresenta uma técnica que permite fazer síntese de filtros interpoladores baseados em polinómios com uma frequência arbitrária qualquer. Conseguem, portanto, criar um filtro interpolador tão facilmente como se tivessem que fazer o projeto de um filtro FIR. Com a técnica proposta podem ter várias bandas de passagem e de atenuação e para cada banda conseguem definir a amplitude e os pesos desejados.

**Løkken [13]** Este trabalho apresenta os princípios básicos de conversões assíncronas de frequências de amostragem e como os modelar. Fazem uma revisão dos conceitos de sobre-amostragem, sub-amostragem, atrasos fracionários e prosseguem para o conversor assíncrono. Explicam como se faz a conversão totalmente assíncrona quando se tem um sinal de relógio de amostragem separado do sinal de relógio de saída.

**Dick [54]** Este artigo descreve o interpolador e mecanismos de controlo que mantêm o rácio de interpolação adequado, quando limitado pela tolerância da frequência da FPGA e de desvios dos sinais de relógio. Descrevem a implementação do processo da amostragem arbitrária numa FPGA. É feita uma síntese de uma técnica baseada num interpolador polifásico. As técnicas utilizadas para a amostragem arbitrária foram, interpolação pelo vizinho mais próximo (*Nearest Neighbor Interpolation* - NNI) e interpolação com dois vizinhos. Descrevem processos de filtragem e estabelecem os detalhes para o plano de controlo para obter desempenho elevado e conversões flexíveis.

**M. Jorgovanovic and Popovic [55]** Este artigo apresenta uma forma de baixar uma frequência de amostragem arbitrária numa FPGA. No fundo faz a decimação do sinal de entrada para estar de acordo com as exigências da frequência de saída. A ideia deste trabalho é semelhante à do trabalho [54], com

a diferença que neste caso diminui a frequência de amostragem em vez de aumentá-la. Permite lidar bem com pequenas variações das frequências de amostragem de entrada e saída e segue a tendência de outros trabalho de fazer a implementação numa FPGA, porque consome poucos recursos e tem uma excelente resposta dos filtros.

**Long and Torres [56]** Este artigo apresenta um VLSI com alta taxa de transferência e estruturas paralelas para conversores arbitrários de frequências. Conversores de alta taxa de transferência são usados para dar flexibilidade e suporte para acomodar as várias frequências usadas em RDS. As estruturas paralelas são pertinentes para sinais de relógio mais baixos para aumentar a sua taxa de transferência. A estrutura de Farrow possui vantagens para este tipo de implementações e possibilitam a diminuição da complexidade ao usar filtros paralelos FIR. As estruturas apresentadas neste artigo podem ser implementadas em FPGA ou VLSI.

## 3.6 Validação da qualidade do conversor

A taxa de distorção harmónica mais ruído (TDH+N) foi a métrica escolhida para avaliar o desempenho do sistema, pois é a métrica mais usada para medir a distorção de sinais de áudio. Mede o ruído e a distorção adicionados a um sinal por um equipamento qualquer. É determinada ao aplicar um único sinal sinusoidal (com conteúdo harmónico conhecido) na entrada do circuito de teste, sendo determinada a distorção. De seguida, é filtrada a frequência do sinal usada no teste por um filtro rejeita-banda, passando o sinal resultante por uma série de filtros ajustados para a banda de interesse (20 Hz - 20 kHz). O que resta é ruído e as harmónicas geradas pelo equipamento. Sendo medido o valor eficaz desse sinal composto. A definição usada para a TDH+N está na equação 3.21.

$$\mathrm{TDH+N} = \frac{\sum_{n=2}^{\infty} \mathrm{harmonicas} + \mathrm{ruido}}{\mathrm{fundamental}} \,. \tag{3.21}$$

## 3.7 Solução proposta

A solução proposta é simples e baseia-se em filtros interpoladores de Lagrange com uma estrutura de filtros FIR. A implementação é puramente digital e utiliza o método direto.

O polinómio de Lagrange é uma forma de interpolação polinomial utilizada para análise numérica. Dado um conjunto de pontos de uma função $y_k$ para $K+1$ pontos $x_k$ é obtido um polinómio de ordem K que passa por esses pontos $(x_k, y_k)$ com $k = 0, 1, \ldots, K$. Este polinómio permite interpolar pontos arbitrários no intervalo $[x_0, x_K]$. A interpolação de Lagrange é uma classe de interpolação polinomial, baseada nos polinómios de Lagrange [12].

Obtém-se o interpolador de Lagrange pela combinação linear dos polinómios de Lagrange, a equação 3.22 mostra como:

$$L(x) := \sum_{j=0}^{k} y_j \, l_j(x) \,. \tag{3.22}$$

Os polinómios de Lagrange obtêm-se com a equação 3.23:

$$l_j(x) := \prod_{i=0, j\neq i}^{k} \frac{(x - x_i)}{(x_j - x_i)} \,. \tag{3.23}$$

As propriedades com interesse prático são:

- Os coeficientes do polinómio podem ser calculados em forma fechada (com fórmulas explícitas) e portanto não é preciso otimização [57];
- Os polinómios interpoladores de Lagrange de ordem baixa são eficientes com o hardware que usam e têm bom desempenho com elevada atenuação para sinais de banda estreita (ou seja uma banda afastada da frequência de Nyquist) [18, 58].

  Para frequências reduzidas estes polinómios apresentam uma qualidade bastante elevada, tornando-os uma escolha ideal para aplicações de áudio digital [12];
- O polinómio interpolador de Lagrange pode ser implementado com algoritmos e estruturas eficientes, reduzindo o esforço computacional do sistema [51];
- É simples de fazer o projeto do filtro e para além disso oferece várias opções de projeto para conversores de frequências de amostragem racionais;
- Usam uma estrutura com dois andares (sobre-amostragem mais Lagrange) melhora a atenuação das imagens e a qualidade para sinais com banda larga [51, 52].

As desvantagens aparecem sob a forma de:

- Se a banda do sinal for muito larga (uma banda próxima da frequência mínima de Nyquist) a qualidade do polinómio interpolador de Lagrange começa a deteriorar-se [18, 58];
- Resposta em frequência é fixa e só depende de um parâmetro, a ordem do filtro interpolador. Não é possível otimizar o método no domínio da frequência [57];
- Os dois andares (sobre-amostragem mais Lagrange) são projetados individualmente provocando uma perda de desempenho [51].

Especificamente o trabalho [58] refere que para se usar o polinómio interpolador de Lagrange corretamente é preciso usar polinómios de ordem elevada, senão os erros para altas frequências (acima de metade da frequência de Nyquist) são elevados. A consequência de se aumentar a ordem é que acarreta problemas numéricos. Pode-se diminuir o erro ao adicionar amostras extra.

O trabalho [52] refere que existem outros polinómios de interpolação com características relevantes e úteis, como por exemplo a B-spline ou O-MOMS (*Optimal maximal-order interpolation of minimal support*). No caso da B-spline, refere que comparativamente com o polinómio interpolador de Lagrange atenua mais as imagens e que a banda de passagem é mais abrupta. O trabalho [59] refere que para a mesma ordem de outros polinómios interpoladores oferece maior gama de banda e causa menor distorção devido a *aliasing*.

A razão pela qual se optou pelo interpolador de Lagrange em vez do interpolador de B-spline, foi essencialmente porque há mais algoritmos e estruturas eficientes para Lagrange, é mais utilizado na literatura, a sua implementação é simples e porque para frequências baixas tem uma qualidade ótima para áudio digital.

A razão de se usar o método direto em vez do método racional para conversões de frequências de amostragem síncronas é devido a ser mais eficiente. Foi apresentado em 3.2.4 que uma grande parte das multiplicações feitas durante a convolução com os coeficientes do filtro FIR era por zero e que só uma fração dos valores é que seriam aproveitados devido à decimação. Com o método direto isso não acontece, porque é feita uma reconstrução do sinal analógico com um filtro ideal e os coeficientes desse filtro são usados para o filtro digital. A convolução é aplicada diretamente entre a sequência de amostras de entrada e os coeficientes do filtro FIR. O cálculo dos coeficientes pode ser realizado de duas formas: em tempo-real ou pré-calculado [45, 60]. A primeira abordagem exige muito esforço computacional para obter os coeficientes, enquanto que a segunda abordagem precisa de uma memória para armazenar os coeficientes, tendo também a desvantagem de só conseguir fazer conversões para os rácios armazenados.

Para a realização deste trabalho optou-se pela segunda opção para poupar recursos no dispositivo. O procedimento usado foi o seguinte:

1. Pré-calcularam-se os coeficientes para alguns rácios. Esse cálculo foi feito com o uso de um filtro FIR analógico de interpolação polinomial de Lagrange no domínio do tempo;

2. Com a resposta impulsiva do filtro analógico projetado obtiveram-se $N$ coeficientes correspondentes à resposta impulsiva do filtro digital usado para a interpolação 'real' do conversor de frequências de amostragem.

   O valor de $N$ corresponde ao único parâmetro configurável desta implementação referido em 3.2.6.1.2 que é a ordem do polinómio interpolador de Lagrange;

3. Como foi explicado em 3.1, para o rácio de frequências de amostragem $\frac{L}{M}$ existem $L$ conjuntos únicos definidos pelo fator de interpolação. Assim, armazenaram-se numa ROM $L$ conjuntos únicos de coeficientes formados por $N$ coeficientes para os rácios de conversão (0,4; 0,5; 0,6666; 0,75 ;2; 3; 4);

4. Para realizar a decimação foi aplicado um offset de $M$ na posição de memória a cada $L$ iterações.

5. Para um rácio de conversão projetado com esta implementação, a complexidade está toda na forma em como se endereçam os coeficientes e idealmente convém que esses rácios sejam dados por dois números inteiros mutuamente primos não muito elevados, para reduzir o número de coeficientes armazenados em memória.

O valor de $N$ foi escolhido tendo em conta o programa desenvolvido em software e que permitiu avaliar diferentes configurações para a ordem do polinómio. Juntamente com o programa, houve o compromisso entre ter uma ordem maior, oferecendo menor erro nas aproximações mas uma memória maior

ou escolher uma ordem menor, aumentando o erro mas diminuindo a complexidade computacional [58].

# Capítulo 4

# Implementação em software

Este capítulo serve para explicar o programa de software desenvolvido. Foi criado porque era necessário uma ferramenta que pudesse possibilitar uma rápida implementação do polinómio interpolador de Lagrange para se testar alguns parâmetros para a sua posterior implementação em hardware. Essas alterações permitiram ver como o desempenho do interpolador era afetado. Serviu também para experimentar algumas configurações e compreender de uma forma mais generalizada como se comportava o polinómio, por exemplo ao variar a ordem ou ao usar diferentes rácios de conversão e de frequências de amostragem. Adicionalmente, o programa possibilitou a criação de uma referência de comparação para a conversão realizada em hardware, servindo como uma validação da funcionalidade desejada. O programa foi realizado em vírgula fixa de modo a corresponder à implementação em hardware.

## 4.1 Descrição do algoritmo

O código que foi realizado encontra-se dividido em dois programas. O primeiro programa $sinewave.c$ é responsável por criar os dados de entrada correspondentes à amostragem de uma sinusoide para uma dada frequência de entrada. É também este programa que cria a referência, ou seja, o resultado esperado após a conversão.

O segundo programa $lagrange.c$ implementa a solução proposta. Este programa atua com rácios de conversão de frequências de amostragem fracionários ou inteiros. Permite também alterar a ordem do polinómio. Os valores que usa como entrada são provenientes do $stdin$ do terminal e são gerados pelo programa $sinewave.c$. A aritmética está na forma de vírgula fixa para corresponder à posterior implementação em FPGA.

É usado um programa externo executado em $Octave$ para visualizar as formas de onda do sinal reconstruido e para determinar a TDH+N presente nesse sinal. Neste trabalho o termo pontos e amostras são usados de forma intercambiável, assim como o termo polinómio interpolador de Lagrange e filtro interpolador.

### 4.1.1 Sinewave

O programa $sinewave.c$ tem como argumentos: o rácio com que se quer fazer a conversão $src\_ratio$, a frequência fundamental do sinal $freq\_signal$, a frequência com que esse sinal é amostrado $freq\_sampling$ e o nome do ficheiro onde é realizada a escrita dos dados pelo programa $sinewave.c$. A execução começa com o cálculo do número total de pontos $nbr\_points$ criados quando o sinal $freq\_signal$ é amostrado a $freq\_sampling$. O número de pontos depende de 3 variáveis, o número de períodos $NBR\_PERIODS$ (é uma constante e o seu valor padrão são 1000 períodos). Depende ainda da frequência de amostragem e da frequência do sinal, de acordo com a equação 4.1:

$$nbr\_points = NBR\_PERIODS \times \frac{freq\_sampling}{freq\_signal}\,. \tag{4.1}$$

Existem ainda outras duas constantes neste programa, a amplitude do sinal $AMPL$ (o seu valor padrão é 0.5) e ainda o número total de bits $NBITS$ (o seu valor padrão é 24 e corresponde ao número de bits usado em áudio digital). Estas duas constantes fazem parte da expressão usada para determinar o valor das amostras de entrada. É criado um ciclo que percorre $nbr\_points$ e a cada iteração do ciclo cria um novo valor que é usado para o polinómio interpolador de Lagrange, o segmento de código que determina os valores é:

$$floor(AMPL * sin(2 * M\_PI * \frac{freq\_signal * i}{freq\_sampling}) * (pow(2, NBITS) - 1))$$

A função $floor$ serve para fazer o arredondamento para baixo do argumento para o inteiro mais próximo. O termo $(pow(2, NBITS) - 1))$ transforma o valor das amostras para a notação de vírgula fixa. Neste caso em particular há um bit para o sinal, nenhum bit para a parte inteira (ou seja o valor inteiro do número correspondente é sempre 0) e os restantes 23 bits servem para representar a parte fracionária, logo tem imensa resolução para representar a parte decimal. Esse foi o motivo para se escolher esse formato, ou seja, a maximização do número de casas decimais armazenadas, que é importante sobretudo para os cálculos intermédios a fim de usar a maior precisão computacional disponível. Pode-se adotar esta representação porque a amplitude do sinal contínuo varia entre $[-0.5; 0.5]$ devido à constante $AMPL$. No entanto, está presente nesta notação um fator de compensação por 2 para que a variação das amplitudes do sinal contínuo passe a ser $[-1, 1]$.

As posições relativas das amostras correspondentes à amostragem na entrada são obtidas a partir do rácio $freq\_signal * i/freq\_sampling$ e é atribuído o respetivo valor do sinal contínuo. Todos estes valores são posteriormente encaminhados para o $stdout$ do terminal, enquanto o programa $lagrange.c$ fica em modo de espera até os receber. Este programa não é muito extenso mas está dividido em duas partes: a primeira parte gera as amostras para a implementação, e a segunda parte cria a referência. A variável $nbr\_points$ é reaproveitada para a determinação da quantidade de pontos para a referência e depende da frequência de amostragem de saída e de entrada. Essa relação está presente no argumento de entrada $src\_ratio$ que se obtém com 4.2:

$$src\_ratio = \frac{freq\_sampling\_out}{freq\_sampling}\,. \tag{4.2}$$

Onde $freq\_sampling\_out$ é a frequência de amostragem de saída, a quantidade de pontos para a referência obtém-se com 4.3:

$$nbr\_points = nbr\_points \times src\_ratio\,. \tag{4.3}$$

Os valores discretos para a referência são obtidos de uma forma semelhante ao ciclo anterior, diferindo no número de amostras obtidas $nbr\_points$ e na posição relativa das amostras $\frac{freq\_signal*i}{freq\_sampling\_out}$. As amostras deixam de ser obtidas com a frequência de amostragem de entrada e em vez disso passam a ser obtidas com a frequência de amostragem de saída conforme o segmento de código:

$$floor(AMPL * sin(2 * M\_PI * \frac{freq\_signal * i}{freq\_sampling\_out}) * (pow(2, NBITS) - 1))$$

Os dois primeiros valores gerado por este programa correspondem ao rácio de conversão $src\_ratio$ e ao número de pontos gerados pela amostragem de entrada $nbr\_points$ e são exportados para o $stdout$ do terminal. Este procedimento permite a passagem de parâmetros do gerador de dados $sinewave.c$ para o conversor $lagrage.c$.

### 4.1.2 Lagrange

O programa de $lagrange.c$ é fundamental pois é neste programa que está contido o algoritmo responsável pelas estimativas de interpolação do sinal reconstruido. Este programa possui apenas um argumento que corresponde à ordem do polinómio $order$. A ordem do polinómio indica a quantidade de pontos que se podem guardar para a realização da interpolação. Como este programa serviu essencialmente para se ter uma implementação rápida do polinómio interpolador de Lagrange não se procurou maximizar a otimização. Nesse sentido gerou-se a única constante deste programa $MAX\_ORDER$ para alocar espaço fixo de memória para os vetores de dados ao limitar a ordem máxima permitida. Desta forma, evitou-se a alocação de memória dinâmica que é mais complexa de se programar. Por sua vez, a ordem do polinómio $order$ limita a dimensão dos vetores, impedindo-os de ultrapassar $MAX\_ORDER$ para impedir que ocorram erros de execução. Atribuem-se os valores do rácio de conversão $src\_ratio$ e do número de pontos amostrados $nbr\_points$ através da leitura dos dois primeiros valores do $stdin$ do terminal que foram gerados pelo programa $sinewave.c$. Com o argumento $order$ define-se a posição relativa inicial da amostra de saída a estimar $xval$. Esta variável é um número inteiro e é incrementada por uma unidade a cada iteração do ciclo que percorre as posições relativas das amostras de saída. Indica a posição relativa da amostra de saída que está atualmente a ser estimada pelo polinómio interpolador de Lagrange. A sua localização inicial depende da paridade da ordem do filtro interpolador:

Se for par:

$$xval = ((\frac{order}{2}) - 1) \times src\_ratio;$$

Ou, se for ímpar:

$$xval = (\frac{order}{2}) \times src\_ratio;$$

Este programa tem um ciclo principal em que a cada iteração $k$ faz a correspondência das posições relativas das amostras de entrada para a escala das posições relativas na saída atribuindo-lhes o valor das amostras adequadamente. Esta correspondência depende do fator de conversão $src\_ratio$, mais concretamente se é maior ou menor que 1. Existe no entanto um passo comum para garantir que existem pontos suficientes para se poder começar a estimar os valores. Consiste em preencher um vetor $y[n]$ com os pontos amostrados do programa $sinewave.c$ nas posições relativas das amostras de entrada $x[n]$. A variável $n$ endereça os vetores. É inicializada a 0 no começo do programa e por cada amostra adicionada é incrementada por uma unidade. Reinicia o seu valor para o valor original sempre que atingir o limite do vetor $order$. O pseudo-código apresentado na figura 4.1 pretende sintetizar o funcionamento do código antes do inicio do algoritmo da conversão de frequências de amostragem.

```
1) Este programa recebe os dados gerados pelo programa sinewave.c.


2) Verifica se a memória RAM já guardou amostras suficientes?
   Sim?
    Prossegue para o algoritmo
   Não?
    Continua a guardar amostras
```

Figura 4.1: Pseudo-código pré-algoritmo de conversão.

Se o fator de conversão for menor que 1 é mais complicado e envolve mais procedimentos. Numa primeira fase é verificado quantas amostras da sequência discreta de entrada se tem que acrescentar, representado pela variável $offset$. A forma como é feita essa verificação é através de duas variáveis: o $increment$ serve como referência para consultar quantos pontos foram usados e o $src\_ratio$ representa a relação entre as frequências de amostragem. Há um ciclo que incrementa duas variáveis: $increment$ e $offset$ por uma unidade até a condição $((int)(src\_ratio * increment) == k)$ ser satisfeita. No fundo o fundamento para essa condição é sabendo que $xval$ é incrementado por uma unidade por cada iteração $k$ do ciclo principal, o número de pontos a serem adicionados têm de manter essa relação de distância em relação a $xval$ que corresponde ao valor central do vetor de dados. A razão para o porquê de se manter a mesma relação de distância é para aumentar a qualidade da estimativa da interpolação, pois nessa situação tem-se a mesma quantidade de pontos antes e depois da amostra a ser estimada. $xval$ é incrementado por uma unidade, porque representa a posição relativa da amostra no sinal contínuo que é amostrado com a frequência de amostragem de saída.

A próxima fase consiste na atualização dos vetores, $x[n]$ e $y[n]$. Existe um ciclo responsável por este processo. Para a quantidade de amostras especificadas em $offset$, são lidas do $stdin$ do terminal

a mesma quantidade de amostras. Iguala-se o valor dessas amostras ao vetor $y[n]$ e faz-se a equivalência da posição relativa das amostras para o vetor $x[n]$. Os valores do vetor $x[n]$ são obtidos com $elem\_counter * src\_ratio$ onde $elem\_counter$ é uma variável auxiliar equivalente a $increment$ atrasado de uma amostra. Adicionalmente, a variável $elem\_counter$ indica quando começa a estimação dos valores das amostras e é comum a qualquer rácio de conversão. Pode acontecer que durante o ciclo de atualização dos vetores a variável $elem\_counter$ indique que os vetores de dados $y[n]$ e $x[n]$ estão preenchidos. Nesse caso é iniciado o processo de estimação, onde é interrompido o ciclo e é guardado na variável $buffer$ a quantidade de amostras restantes para adição prosseguindo-se a estimação dos valores das amostras indicadas pela posição relativa da amostra de saída $xval$.

Na iteração seguinte, se houver amostras para acrescentar entra num ciclo para adicionar as amostras em falta, caso contrário fica impossibilitado de entrar nesse ciclo até à conclusão do programa e prossegue o funcionamento normal do programa com a execução dos ciclos de verificação de amostras a adicionar e do ciclo de atualização dos vetores. A forma como é realizada a inclusão de novas amostras é exatamente igual à do ciclo de atualização dos vetores, com a diferença de percorrer $buffer$ amostras e não $offset$ amostras. O pseudo-código apresentado na figura 4.2 pretende sintetizar o funcionamento do código, se o rácio de conversão for menor que um.

```
3) Se o rácio de conversão for menor que 1
        Verifica se acabou de utilizar um conjunto único de coeficientes
        Sim?
            Mostra o resultado da amostra de saída para esse conjunto único de coeficientes
            Guarda novas amostras na memória RAM
            Verifica se guardou amostras suficientes
            Sim?
                Verifica se falta usar algum conjunto único de coeficientes
                Sim?
                    Utiliza um conjunto único de coeficientes diferente
                Não?
                    Reinicia o ciclo dos conjuntos únicos de coeficientes
                Prossegue para o cálculo do valor da amostra de saída
            Não?
                Continua a guardar amostras na memória RAM
        Não?
            Continua a calcular a amostra de saída
```

Figura 4.2: Pseudo-código para rácios de conversão menores que um.

Para rácios de conversão superiores a 1 o processo de conversão é mais simples. O cuidado principal está presente na decisão do momento adequado para a adição de novas amostras e permitir que o segmento de código estime continuamente o valor das amostras enquanto for possível. Portanto, o segmento de código que controla, a cada iteração $k$, se é necessário acrescentar mais amostras é: `if`$(k == lim)$. A variável auxiliar $lim$ é obtida através de $lim = (int)elem\_counter * src\_ratio$; e existe porque o valor da iteração $k$ é uma variável do tipo inteiro. É usada para antecipar a altura da escrita da nova amostra nos vetores $y[n]$ e $x[n]$. Enquanto o valor da iteração $k$ não for igual a $lim$ o algoritmo entra num modo em que está continuamente a calcular novas amostras. Chama-se a atenção que neste caso não há a necessidade de interromper nenhum ciclo com a variável $elem\_counter$.

Há uma condição que tem de ser cumprida em primeiro lugar, que é garantir que existem pontos suficientes para começar a estimar `if`$(elem\_counter >= order)$. Nesta condição o $elem\_counter$ é uma variável que indica o número de elementos adicionados e assim que os vetores $y[n]$ e $x[n]$ tenham os

seus $order$ endereços todos preenchidos, pode-se iniciar a estimação. $elem\_counter$ é incrementado sempre que uma nova amostra é armazenada.

O algoritmo de estimação das amostras usa o polinómio interpolador de Lagrange de 3.22 e está dividido em 3 partes. A primeira parte é responsável pelo produtório, a segunda parte é responsável pela transformação para vírgula fixa e a terceira parte é responsável por acumular os resultados do produto das amostras pelos coeficientes do polinómio interpolador de Lagrange. A variável que armazena os valores das amostras estimadas é a variável $est$. Existem dois ciclos, o ciclo interior é usado para a determinação dos coeficientes do filtro interpolador através do seguinte segmento de código $coef\_real = coef\_real * (xval - x[j])/(x[i] - x[j])$;. Aqui, $coef\_real$ é uma variável local com precisão $double$ e que funciona como acumulador do resultado do produtório. É definida como uma variável do tipo $double$, pois nestas operações é necessário maior precisão aritmética e variáveis deste tipo têm mais bits para a representação de números atingindo o objetivo de ter maior rigor nos cálculos. De notar que a variável da iteração $i$ e $j$ são diferentes. A variável de iteração $j$ percorre o ciclo interior que calcula os coeficientes do polinómio, enquanto que a variável de iteração $i$ é usada para endereçar o vetor de posições relativas das amostras $x[i]$ no ciclo interior, para percorrer o ciclo exterior e para endereçar o vetor das amostras $y[i]$. Após o ciclo que calcula os coeficientes do polinómio ter terminado segue-se a transformação para o formato de vírgula fixa com o seguinte segmento de código: $coef = floor(coef\_real * (pow(2, 22)))$. $coef$ representa a variável $coef\_real$ convertida para o formato de vírgula fixa, a multiplicação pela parcela $(pow(2, 22))$ (frequentemente representado por $\mathtt{Q}_{22}$) significa que tem 1 bit para o sinal, 22 bits para representar a parte fracionária e 1 bit para representar o valor do inteiro correspondente. A razão de se necessitar de um bit para a parte inteira é devido a $coef\_real$ ter que representar o número 1. Com o valor do coeficiente da parcela $i$ resta multiplicá-lo pela amostra $y[i]$ e guardar o resultado da operação no acumulador $est$ com o segmento de código $est = est + (((long)coef * y[i]) >> 22)$. Como a variável $coef$ e a amostra guardada em $y[i]$ estão com formatos em vírgula fixa diferentes, $\mathtt{Q}_{22}$ e $\mathtt{Q}_{23}$ respetivamente, o resultado desta operação vai ser em $\mathtt{Q}_{22}$, obtido com o operador $>> 22$. Com o ciclo de estimação das amostras terminado prossegue-se o incremento da variável responsável pela posição relativa das amostras de saída $xval$ por uma unidade e passa-se para a próxima iteração do ciclo principal $k$. O pseudo-código apresentado na figura 4.3 pretende sintetizar o funcionamento do código, se o rácio de conversão for maior que um.

```
4) Senão
        Verifica se acabou de utilizar um conjunto único de coeficientes
        Sim?
            Mostra o resultado da amostra de saída para esse conjunto único de coeficientes
            Verifica se falta usar algum conjunto único de coeficientes
            Sim?
                Utiliza um conjunto único de coeficientes diferente
            Não?
                Reinicia o ciclo dos conjuntos únicos de coeficientes
                Guarda uma nova amostra na memória RAM
            Prossegue para o cálculo do valor da amostra de saída
        Não?
            Continua a calcular a amostra de saída
```

Figura 4.3: Pseudo-código para rácios de conversão maiores que um.

### 4.1.3 Octave

Por último usou-se o programa auxiliar $my\_read\_test\_out.m$. Este programa foi usado sobretudo para se ter acesso aos valores da TDH+N. No entanto, houve um conjunto de características que auxiliaram a entender se o programa era coerente com os resultados esperados. Este programa quando executado imprime 4 figuras. A primeira figura mostra o conjunto de pontos estimados pelo programa $lagrange.c$ com base nas amostras geradas pelo programa $sinewave.c$. A segunda figura mostra o conjunto de pontos que o programa $my\_read\_test\_out.m$ usa para os cálculos da FFT. A terceira figura mostra o espectro da FFT unilateral e a quarta figura mostra uma representação da FFT, mas com legendas nos eixos e com as escalas de frequência e de potência adequadas. Este programa reverte a representação em vírgula fixa ao dividir as amostras por um fator $(pow(2, 23))$ obtendo o resultado original do produto das amostras pelos coeficientes.

De notar que este programa, para calcular a potência do ruído, descarta a componente DC e a harmónica fundamental, assim como uma janela de frequências em torno dessas duas frequências.

## 4.2 Modo de utilização

Para utilizar os programas desenvolvidos é preciso abrir um terminal e definir os parâmetros das funções. O programa $sinewave.c$ usa os argumentos, $src\_ratio$, $freq\_signal$, $freq\_sampling$ e o ficheiro para escrita, enquanto o programa $lagrange.c$ usa apenas, $order$. Para os executar é necessário o operador de $pipeline\ (|)$ onde os valores produzidos pelo primeiro programa $sinewave.c$ vão servir de entrada para o segundo programa $lagrange.c$. Por exemplo, para converter um sinal de 1 kHz amostrado a 48 kHz, para o mesmo sinal amostrado a 96 kHz (fator de conversão igual a 2) e uma ordem para o polinómio interpolador de Lagrange igual a 16, a instrução a usar é:

$$./sinewave\,2\,1000\,48000\,REF\,|\,./lagrange\,16\,>\,OUT$$

Nesta instrução os argumentos `REF` e `OUT` representam respetivamente o ficheiro de escrita para a referência e o ficheiro de escrita para os valores estimados pelo filtro interpolador. Para utilizar o programa auxiliar $my\_read\_test\_out.m$ são precisos 6 argumentos:

- O primeiro parâmetro serve para definir a frequência de amostragem do sinal (em Hertz);
- O segundo parâmetro é o canal que vai ser analisado;
- O terceiro parâmetro é o número total de canais;
- O quarto parâmetro é o ficheiro de dados;
- O quinto parâmetro é a frequência fundamental do sinal;
- O sexto parâmetro é o número de períodos.

Usando o exemplo anterior, como a frequência de amostragem de saída é 96 kHz, o ficheiro de dados com os valores estimados estão armazenados no ficheiro `OUT` e a frequência fundamental do sinal é 1 kHz, a instrução a executar é:

$$my\_read\_test\_out(96000, 1, 1, "OUT", 1000, 500)$$

O último parâmetro é usado para a determinação da quantidade de pontos para a FFT, melhorando a representação do espectro conforme se aumenta este parâmetro. Pode ser interessante comparar as figuras das amostras estimadas pelo programa $lagrange.c$ com as amostras ideais, para o fazer basta substituir o parâmetro do ficheiro de dados para `REF`.

# Capítulo 5

# Implementação em hardware

Este capítulo descreve a implementação em Verilog do algoritmo de conversão de frequências de amostragem descrito no capítulo 4, justificando as opções tomadas relativamente à implementação em hardware.

Após a descrição e entendimento do sistema de conversão de frequências de amostragem, segue-se a apresentação das limitações do sistema. Por exemplo, as restrições impostas à relação $\frac{L}{M}$ para haver periodicidade nos coeficientes e a impossibilidade de os calcular em tempo real devido às exigências de frequência.

No final, são apresentados os diagramas de blocos, é dada uma explicação dos sinais do ponto de vista do utilizador e é apresentado o relatório de utilização dos recursos ocupados pela implementação em FPGA.

## 5.1 Solução proposta

O circuito desenvolvido e sintetizado em FPGA tem essencialmente 4 módulos cada um com funções específicas e distintas. Começando pelas memórias: uma memória só de leitura ROM onde estão armazenados os coeficientes do filtro interpolador para os seguintes rácios de conversão: 0,4; 0,5; 0,6666; 0,75; 2; 3; 4, que podem alternativamente ser representados na forma de fração irredutível $(\frac{2}{5}; \frac{1}{2}; \frac{2}{3}; \frac{3}{4}; \frac{2}{1}; \frac{3}{1}; \frac{4}{1})$. Nesta forma os fatores de interpolação ($L$) são obtidos diretamente através do numerador (2, 1, 2, 3, 2, 3, 4), não esquecendo que os fatores de interpolação correspondem à multiplicidade de conjuntos únicos de coeficientes. A forma como os coeficientes estão organizados internamente na ROM é por ordem crescente de rácio de conversão. Por exemplo, se a ordem do polinómio interpolador de Lagrange for 10, tem-se a seguinte configuração interna de memória: os $2 \times 10 = 20$ primeiros coeficientes são para o rácio de conversão 0,4; os próximos $1 \times 10 = 10$ coeficientes são para o rácio de conversão 0,5 e assim sucessivamente até se chegar ao último rácio de conversão (4), onde são guardados $4 \times 10 = 40$ coeficientes. No total a ROM teria 170 coeficientes.

A outra memória é uma memória RAM e permite armazenar as amostras de entrada. A memória que se usou nesta implementação usa dois sinais de relógio:

- Um sinal de relógio $clka$ a funcionar à frequência de amostragem de entrada para escrita de novas amostras;
- Um sinal de relógio $clkb$ a funcionar à frequência obtida através do produto entre a ordem do filtro interpolador e a frequência de amostragem de entrada, o qual é usado para processamento dos dados.

De seguida, tem-se a máquina de estados finita ($FSM.v$) responsável pelo controlo dos dados. Esta máquina de estados tem 3 estados: `IDLE`, `READ` e `FULL`. A máquina de estados serve para gerar os endereços das memórias RAM e ROM e fornecer sinais de controlo ao circuito. Fica no estado `IDLE` sempre que a memória RAM tenha uma quantidade insuficiente de amostras para realizar a estimação. Assim que a quantidade de amostras armazenadas na memória RAM seja satisfatória, prossegue para o estado `FULL`. Este estado indica que já foram estimadas todas as amostras de saída possíveis com as amostras de entrada armazenadas na memória RAM. Para passar para o próximo estado é preciso guardar na memória RAM mais amostras. Entra-se no estado `READ` quando essa condição é verificada. Enquanto permanecer no estado `READ`, significa que o processo de estimação das amostras de saída ainda não está concluído. Lê os coeficientes da memória ROM e as amostras de entrada da memória RAM, realizando o produto desses valores. O resultado dessa operação é somado ao valor de um acumulador. O valor final desse acumulador corresponde ao valor da amostra de saída que se está a estimar.

O último módulo ($Circuito.v$) junta as memórias RAM e ROM com o módulo de controlo de dados ($FSM.v$), formando o circuito de conversão de frequências de amostragem. O módulo $Circuito.v$ faz o endereçamento da memória RAM para escrita das amostras de entrada. Difere dos endereços fornecidos pela máquina de estados, porque na máquina de estados são fornecidos os endereços da memória RAM para a leitura das amostras de entrada. Os sinais de relógio que são usados para a atribuição destes endereços também são diferentes. Para os endereços onde as amostras de entrada são escritas usam-se os flancos positivos do sinal de relógio $clka$ e para os endereços onde as amostras de entrada são lidas usam-se os flancos positivos do sinal de relógio $clkb$. Este módulo usa os sinais de controlo fornecidos pela máquina de estados para efetuar a operação de estimação das amostras de saída, equivalente ao segmento de código $est = est + (((long)coef * y[i]) >> 22)$ da implementação em software.

## 5.2 Módulos

As sub-secções seguintes descrevem as unidades funcionais enumeradas anteriormente, indicando em cada caso as entradas e saídas, bem como os diagramas temporais. Os diagramas de blocos detalhados estão representados no apêndice A.

### 5.2.1 ROM

A memória ROM ilustrada na figura 5.1, tem dois portos de entrada, o porto $clk$ para o sinal de relógio e o porto $addr$ para o endereço de leitura e tem um porto de saída $data$ que corresponde aos coeficientes do filtro interpolador. Analisando os sinais repara-se que o endereço de leitura tem 15 bits. Estes servem para endereçar os coeficientes que foram pré-calculados através da implementação em software. No total estão guardados $\texttt{TOTAL\_COEFFICIENTS} = 8704$ coeficientes. Os detalhes acerca dos motivos da escolha para esta quantidade de coeficientes são dados futuramente no capítulo 6. O porto $data$ tem $\texttt{ROM\_WIDTH} = 24$ bits, usados comummente para áudio digital. Aprofundando o conteúdo e lógica combinatória do módulo, existe um vetor $rom\_data$ que simula a memória RAM, este vetor tem uma dimensão de `TOTAL_COEFFICIENTS` linhas, e cada linha tem `ROM_WIDTH` bits. Tanto `TOTAL_COEFFICIENTS` como `ROM_WIDTH` são parâmetros do módulo. Adicionalmente existe outro parâmetro `INIT_FILE` para indicar o ficheiro que contém os valores para inicializar a ROM. A forma como é feita a inicialização é com o segmento de código $readmemh(\texttt{INIT\_FILE}, rom\_data)$ onde faz a leitura e atribuição dos valores dos coeficientes no vetor $rom\_data$ na unidade de tempo igual a 0. Estes parâmetros servem para poder configurar a arquitetura do módulo oferecendo flexibilidade ao projeto. Conectado ao porto $clk$ está o sinal de relógio de processamento de dados $clkb$. A cada flanco ascendente deste sinal é lido do porto $addr$ da memória $rom\_data$ um coeficiente $data$. Foi usado o ficheiro $rom\_data\_512.mem$ para fazer a inicialização dos coeficientes na memória ROM.

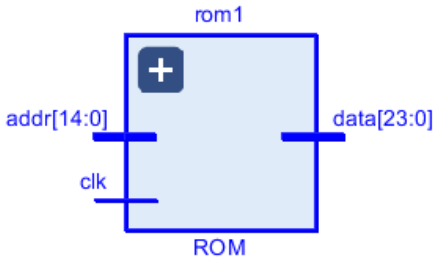
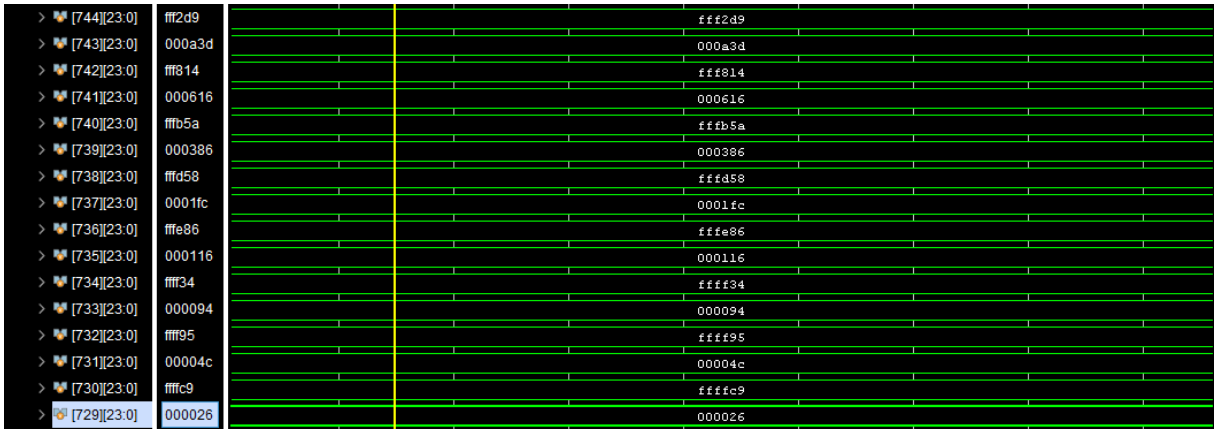



Figura 5.1: Módulo ROM.

O porto de saída $data$ está conectado ao sinal $out\_coef$ e os portos de entrada $clk$ e o $addr$ estão conectados aos sinais $clkb$ e $index2$ respetivamente na arquitetura de maior nível. Para os diagramas temporais em 5.2 representaram-se os sinais da arquitetura de maior nível. O valor dos coeficientes $out\_coef$ aparecem com um atraso de 1 ciclo de relógio em relação ao endereço de leitura, porque demora um ciclo de relógio para ler o endereço do sinal $index2$ na memória ROM.


(a) Endereços da ROM




(b) Sub-conjunto de coeficientes da ROM


(c) Sinal de saída da ROM

Figura 5.2: Diagrama temporal para a ROM.

### 5.2.2 RAM

Este módulo pertence à biblioteca de exemplos da $Xilinx$ e está representado na figura 5.3. Tal como no módulo anterior, este módulo é parametrizável. Os parâmetros que são permitidos modificar são: `RAM_WIDTH`, `RAM_DEPTH`, `RAM_PERFORMANCE` e `INIT_FILE`. O parâmetro `RAM_WIDTH` é comum a todos os módulos e o valor padrão é sempre 24 bits, porque representa a quantidade de bits usado em áudio digital. O parâmetro `RAM_DEPTH` define a quantidade de posições de memória desta memória. O parâmetro `RAM_PERFORMANCE` representa os modos de funcionamento. Pode funcionar em `LOW_LATENCY` ou em `HIGH_PERFORMANCE`. Se estiver no modo `LOW_LATENCY` o resultado demora um ciclo de relógio a aparecer no porto $doutb$, mas os dados têm um maior atraso face ao flanco do relógio. Se estiver no modo `HIGH_PERFORMANCE`, o resultado aparece dois ciclos de relógio depois, mas diminui o atraso dos dados face ao flanco do relógio. Nesta implementação adotou-se o modo `HIGH_PERFORMANCE`.

As características dos portos estão apresentadas na tabela 5.1:

O porto $enb$, permite a leitura dos conteúdos de memória através do segmento de código $ram\_data <=$ `BRAM`$[addrb]$. `BRAM` é um sinal equivalente a uma memória RAM com `RAM_DEPTH` posições de memória e com capacidade para guardar amostras com `RAM_WIDTH` bits. O sinal $ram\_data$ é um sinal auxiliar e a sua funcionalidade varia consoante o modo de funcionamento da RAM. Se a RAM estiver no modo `LOW_LATENCY` a saída deste módulo é diretamente $ram\_data$, se estiver no modo `HIGH_PERFORMANCE` o sinal $ram\_data$ é usado como propagação do valor da amostra. O porto $regceb$ serve para atualizar o valor do sinal $doutb\_reg$ que é posteriormente encaminhado para o porto de saída $doutb$. O porto de entrada $rstb$ coloca na saída deste módulo um valor nulo, no entanto não altera o conteúdo de memória da RAM.

O parâmetro `INIT_FILE` é usado caso se tenha um ficheiro com valores para inicializar a RAM. Se

| Porto | Direção | Descrição |
|---|---|---|
| addra | entrada | Endereça as posições de memória para escrita. |
| addrb | entrada | Endereça as posições de memória para leitura. |
| clka | entrada | Ligação ao sinal de relógio de escrita. |
| clkb | entrada | Ligação ao sinal de relógio de leitura. |
| dina | entrada | Armazena em memória a amostra de entrada. |
| enb | entrada | Permite a leitura dos conteúdos de memória. |
| regceb | entrada | Atualiza o sinal de saída. |
| rstb | entrada | Coloca na saída deste módulo um valor nulo. |
| wea | entrada | Autoriza a escrita na memória RAM. |
| doutb | saída | Mostra o resultado da estimação. |

Tabela 5.1: Descrição dos portos do módulo RAM.

não for disponibilizado um ficheiro, a RAM é inicializada com todas as posições de memória a 0 na unidade de tempo igual a 0. Foi o que se fez nesta implementação. A escrita dos valores das amostras para a memória RAM é realizada no módulo $Circuito.v$.

Este módulo e os restantes usam a função $clogb2$ para calcular o valor do logaritmo do argumento especificado. Usou-se esta função porque a quantidade de bits a usar para alguns portos depende de parâmetros, por essa razão não é possível definir a dimensão do *bus* de dados *a priori*. Esta função divide o argumento sucessivamente por dois até o resultado dessa divisão deixar de ser positivo, retornando o valor resultante.

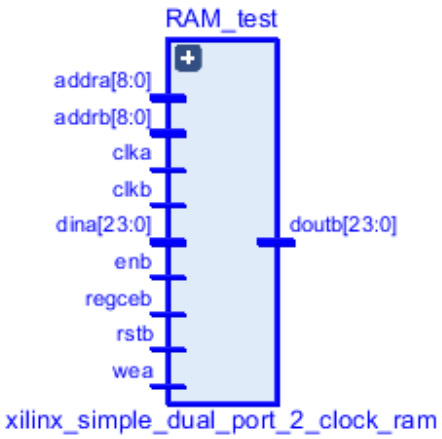
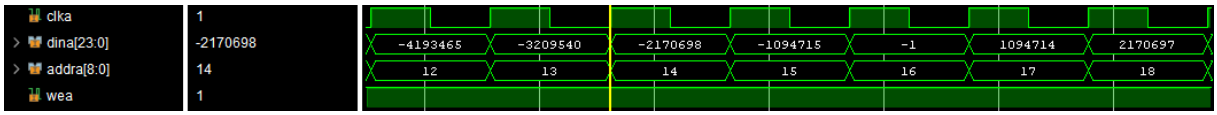


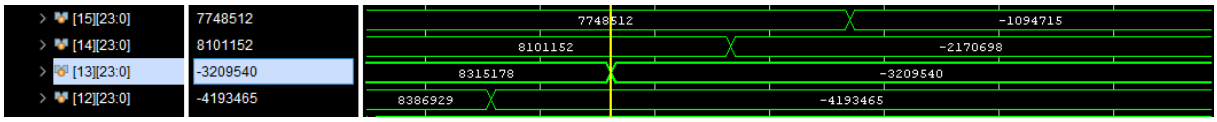

Figura 5.3: Módulo RAM.

Tal como se fez para a ROM, os diagramas temporais indicam os nomes dos sinais conectados na RAM na arquitetura de maior nível. Só o porto da RAM $wea$ é que não está conectado a um sinal com o mesmo nome, estando ligado ao sinal $wea\_ram$. Se $rst = 1$, então o valor do sinal $wea\_ram$ é zero, ou seja não há escrita na RAM. Caso contrário é atribuído o valor do sinal do porto $wea$ do circuito. Relembra-se que por a RAM estar no modo de funcionamento `HIGH_PERFORMANCE`, a leitura do conteúdo de memória no endereço especificado aparece na saída dois ciclos de $clkb$ depois. Para a escrita é preciso esperar um ciclo de $clka$ para armazenar a amostra na memória RAM. Os diagramas temporais

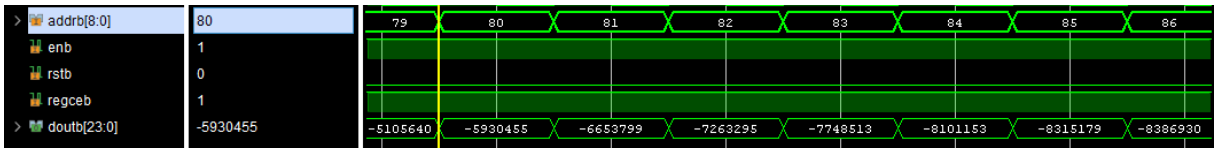
para a RAM podem ser visualizados na figura 5.4.



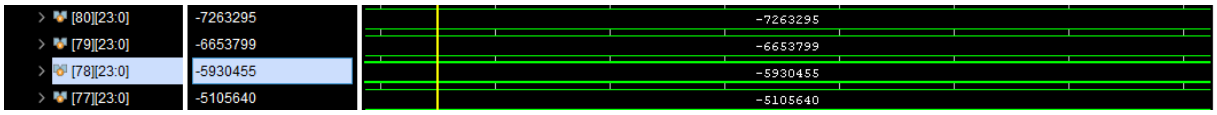

(a) Sinais para escrita da RAM




(b) Parte do conteúdo de memória para o exemplo de escrita da RAM




(c) Sinais para leitura da RAM




(d) Parte do conteúdo de memória para o exemplo de leitura da RAM

Figura 5.4: Diagrama temporal para a RAM.

### 5.2.3 FSM

O módulo que implementa a máquina de estados está representado na figura 5.5. São utilizados dois parâmetros já definidos anteriormente: `RAM_DEPTH` para a dimensão da memória RAM, `TOTAL_COEFFICIENTS` para a dimensão da memória ROM.

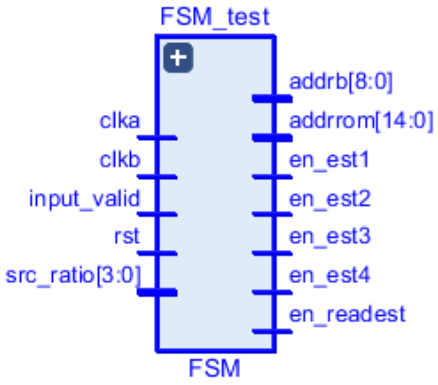



Figura 5.5: Módulo FSM.

A descrição dos portos de entrada e de saída está ilustrada na tabela 5.2:

| Porto | Direção | Descrição |
|---|---|---|
| clka | entrada | Ligação ao sinal de relógio de escrita. |
| clkb | entrada | Ligação ao sinal de relógio de leitura. |
| input_valid | entrada | Autoriza a escrita na memória RAM. |
| rst | entrada | Reinicia o circuito ao seu estado inicial. |
| src_ratio | entrada | Indica o fator de conversão pretendido. |
| addrb | saída | Fornece os endereços para a memória RAM. |
| addrrom | saída | Fornece os endereços para a memória ROM. |
| en_est1 | saída | Indicação do final do primeiro conjunto único de coeficientes. |
| en_est1 | saída | Indicação do final do segundo conjunto único de coeficientes. |
| en_est1 | saída | Indicação do final do terceiro conjunto único de coeficientes. |
| en_est1 | saída | Indicação do final do quarto conjunto único de coeficientes. |
| en_readest | saída | Indica que se encontra em modo de leitura. |

Tabela 5.2: Descrição dos portos do módulo FSM.

Prosseguindo para o número de bits usados nestes portos: para o porto $src\_ratio$ usaram-se 4 bits possibilitando a representação de 16 rácios de conversão. Nesta implementação usaram-se apenas 7, significando que 3 bits bastavam ($2^3 = 8$). Os rácios de conversão extra servem para uma expansão futura de rácios de conversão. O número de bits para o porto $addrrom$ é obtido através da utilização da função $clogb2$ para determinar quantos bits são necessários para endereçar a memória ROM. Da mesma forma, o número de bits para o porto $addrb$ é obtido através da utilização da função $clogb2$ para determinar quantos bits são necessários para endereçar a memória RAM.

Neste módulo há 3 parâmetros locais para definir os estados da máquina de estados finita:

$$localparam\ \texttt{IDLE} = 0, localparam\ \texttt{READ} = 1 e localparam\ \texttt{FULL} = 2$$

. O diagrama de estados está presente na figura 5.6:

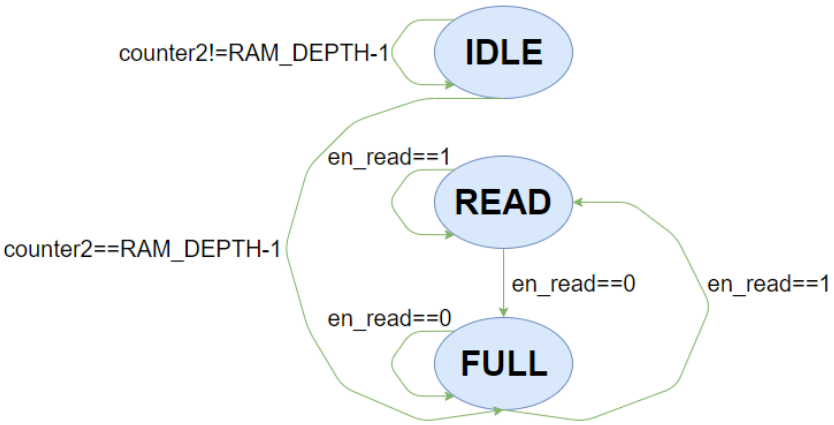



Figura 5.6: Diagrama de estados.

Os sinais $counter2$ e $en\_read$ serão explicados brevemente.

Antes, será feita uma apresentação das sub-secções deste módulo. A primeira fase calcula quantos coeficientes no total ($lenght$) são usados para o rácio de conversão especificado. Relembra-se que na secção 5.1 foi explicado que o número total de coeficientes dependia do fator de interpolação, pois este fator indicava a quantidade de conjuntos únicos de coeficientes. A tabela 5.3 mostra os rácios de conversão, a palavra binária atribuída e a quantidade de conjuntos únicos de coeficientes para esse fator.

| Rácio de conversão | Palavra binária | Fator de interpolação |
|---|---|---|
| 0,4 | 0000 | 2 |
| 0,5 | 0001 | 1 |
| 0,6666... | 0010 | 2 |
| 0,75 | 0011 | 3 |
| 2 | 0100 | 2 |
| 3 | 0101 | 3 |
| 4 | 0110 | 4 |

Tabela 5.3: Rácio de conversão, palavra binária e fator de interpolação.

Por exemplo, para o rácio de conversão 0,75 e 3, o número de coeficientes totais usados é dado pelo seguinte código:

```
4'b0011, 4'b0101: begin
          lenght <= RAM_DEPTH * 3;
        end
```

Se for usado um rácio de conversão inválido, o valor padrão que é atribuído é RAM_DEPTH coeficientes.

A segunda fase implica a definição da lógica da máquina de estados, ou seja a forma como atualiza o novo estado. Esta atualização é feita ao ritmo do relógio de processamento de dados. Se houver um sinal positivo no porto $rst$, transita para o estado IDLE. Se não ocorrer essa ativação, o estado atual $state$ passa para o próximo estado $nxt\_state$.

O terceiro passo conta as amostras armazenadas na memória RAM com o auxílio do sinal interno $counter2$. Existem os sinais internos $counter1$ e $counter$, sendo usado o sinal $counter2$ para este módulo estar sincronizado com os restantes. O sinal $counter2$ é formado através do atraso do sinal $counter1$ por um ciclo do relógio $clka$. Do mesmo modo, o sinal $counter1$ é formado através do atraso do sinal $counter$ por um ciclo do relógio $clka$. Se for ativado o estado inicial do circuito com o sinal conectado ao porto $rst$, o valor dos contadores é 0. Caso contrário, se o valor das amostras for válido, se houver um flanco positivo do sinal de relógio $clka$ e se não tiverem sido armazenadas amostras suficientes na memória RAM é incrementado o valor do contador $counter$ sendo este sinal propagado a cada ciclo ascendente do relógio $clka$ para os restantes contadores. Quando tiver guardado o número de amostras necessárias na memória RAM, é inibido o incremento do sinal interno $counter$ parando a contagem dos contadores.

A quarta fase implica saber a quantidade de amostras a armazenar na memória RAM após a geração da amostra de saída. A quantidade de pontos a adicionar é fundamental para a qualidade da estimação das novas amostras, pois garante que o valor que se quer estimar está situado sensivelmente a 'meio' da memória RAM, ou seja tem tantas amostras antes como depois da sua posição de memória, aumentando consideravelmente a qualidade da estimação. Recorreu-se à implementação em software para determinar a quantidade de amostras a acrescentar após usar um conjunto único de coeficientes. A tabela 5.4 mostra que para diferentes rácios de conversão existem padrões de amostras a adicionar que podem ser explorados numa implementação em hardware:

| Rácio de conversão | Padrão de adição de amostras |
|---|---|
| 0,4 | 3, 3, 2, 3, 2, . . . , 3, 2 |
| 0,5 | 2, 2, . . . , 2, 2 |
| 0,6666. . . | 2, 1, 2, 1, . . . , 2, 1 |
| 0,75 | 2, 1, 1, 2, 1, 1, . . . , 2, 1, 1 |
| 2, 3, 4 | 1, 1, . . . , 1, 1 |
| | 1, 2, . . . , 1, 2 |

Tabela 5.4: Sequência de amostras a adicionar para estimação de amostras de saída.

Em hardware implementou-se uma sub-secção que imita a sequência de amostras a adicionar para cada rácio de conversão. Os sinais usados foram: $sample\_counter$ para fazer a contagem das amostras adicionadas, $nbr\_sample1$ e $nbr\_sample2$ para indicar a quantidade de amostras a adicionar e $flag\_sample1$ e $flag\_sample2$ para controlar o fluxo de execução do programa. Em caso de reinicio do circuito com a ativação do sinal $rst$, os valores dos sinais $flag\_sample1$, $flag\_sample2$ e $sample\_counter$ ficam nulos e os valores dos sinais $nbr\_sample1$ e $nbr\_sample2$ correspondem às quantidades das amostras a adicionar, as quais estão indicadas na tabela 5.5:

| Rácio de conversão | Número de amostras (Reinício) |
|---|---|
| 0,4 | $nbr\_sample1 <= 3;$<br>$nbr\_sample2 <= 2;$ |
| 0,5 | $nbr\_sample1 <= 2;$<br>$nbr\_sample2 <=' bx;$ |
| 0,6666...; 0,75 | $nbr\_sample1 <= 2;$<br>$nbr\_sample2 <= 1;$ |
| 2; 3; 4 | $nbr\_sample1 <= 1;$<br>$nbr\_sample2 <= 2;$ |

Tabela 5.5: Atribuição da quantidade de amostras após o reinicio do circuito.

Se o rácio de conversão não for nenhum dos especificados, os valores dos sinais $flag\_sample1$, $flag\_sample2$ e $sample\_counter$ são nulos. Acerca da tabela 5.5, no fator de conversão igual a 0,5 o sinal $nbr\_sample2$ tem um valor igual a $'bx$ que significa que não interessa, enquanto que os rácios de conversão 2, 3 e 4 têm a particularidade do sinal $nbr\_sample2 <= 2$. Adotou-se essa estratégia para criar um impulso na implementação. Esses impulsos são necessários para iniciar o algoritmo. Se o sinal fosse $nbr\_sample2 <= 1$, significava que o valor era constante, e nesse caso seria impossível detetar novos impulsos. Não detetando esses impulsos, não seriam guardadas novas amostras na memória RAM, e por consequência não seriam calculadas novas amostras de saída.

Assumindo que estão armazenadas amostras suficientes na memória RAM e que se tem um rácio de conversão válido, o que acontece é o seguinte: verifica-se se a quantidade de amostras adicionadas na memória RAM após a estimação do valor da amostra de saída é suficiente (com o segmento de código $(sample\_counter == nbr\_sample1)$). Se não for continua a adicioná-las. Caso sejam suficientes, verifica-se o valor do sinal de controlo $flag\_sample1$. Se for nulo, significa que está no primeiro valor do padrão de adição de amostras. Muda-se o seu valor lógico para 1 para indicar que tem que mudar para o próximo elemento da sequência do padrão adição de amostras e reinicia a contagem do contador do número de amostras adicionadas $sample\_counter$ para 1. Se for igual a 1, está no segundo ou no terceiro elemento, e é preciso verificar o valor do sinal lógico $flag\_sample2$. Se o valor do sinal lógico $flag\_sample2$ for nulo, significa que está no segundo elemento da sequência do padrão adição de amostras. Muda-se o seu valor lógico para 1 para indicar que tem que mudar para o último elemento da sequência e reinicia a contagem do contador do número de amostras adicionadas $sample\_counter$

para 1. O sinal lógico $flag\_sample2$ só é necessário para padrões de adição de amostras com 3 elementos. Nesta implementação abrange os rácios de conversão 0,4 e 0,75. Se o valor do sinal lógico $flag\_sample2$ for igual a 1, significa que está no último elemento do padrão adição de amostras. A atribuição do valor dos sinais $flag\_sample1$ e $flag\_sample2$ depende da sequência de adição de amostras. Por último, é reiniciada a contagem do contador do número de amostras $sample\_counter$ para 1.

Por exemplo, para o rácio de conversão igual a 0,75 a adição de amostras é dado pelo seguinte código:

```
4'b0011: begin
    if ((sample_counter == nbr_sample1) && !flag_sample1)
      begin
        sample_counter <= 1;
        flag_sample1 <= 1;
      end
    else if ((sample_counter == nbr_sample2) && flag_sample1 && !flag_sample2)
      begin
        sample_counter <= 1;
        flag_sample2 <= 1;
      end
    else if ((sample_counter == nbr_sample2) && flag_sample1 && flag_sample2)
      begin
        sample_counter <= 1;
        flag_sample1 <= 0;
        flag_sample2 <= 0;
      end
  end
```

A quinta etapa cria sinais lógicos no módulo $FSM.v$ para alertar que é possível, com as amostras armazenadas na memória RAM, estimar novas amostras de saída. Os sinais lógicos em questão são: $start0$, $start1$, $start2$, $start3$, $start4$, $start5$, $start6$. Como não se guardam os valores destes sinais, em hardware são definidos como ligações físicas. A vantagem em relação a registos é que não precisam de esperar um ciclo do sinal de relógio para atualizar o seu valor. A função destes sinais é verificar continuamente a validade das condições da etapa quatro. Existem 7 sinais destes porque são verificadas 7 condições de começo do algoritmo. Há 6 condições relacionadas com o padrão de adição de amostras e uma condição relacionada com o momento em que a memória RAM tem, pela primeira vez, as amostras necessárias para efetuar a primeira estimativa (quando sai do estado `IDLE`).

Os registos auxiliares, $start0d1$, $start1d1$, $start2d1$, $start3d1$, $start4d1$, $start5d1$ e $start6d1$ são formados através do atraso dos sinais $start0$, $start1$, $start2$, $start3$, $start4$, $start5$ e $start6$ por um ciclo de relógio $clkb$. Estes registos têm como única função ajudar na criação de um impulso.

Os impulsos estão nos sinais lógicos: $start0p$, $start1p$, $start2p$, $start3p$, $start4p$, $start5p$ e $start6p$ e são formados através da atribuição do valor lógico '1' aos sinais lógicos $start0$, $start1$, $start2$, $start3$, $start4$, $start5$ e $start6$ e o valor lógico '0' aos registos $start0d1$, $start1d1$, $start2d1$, $start3d1$, $start4d1$, $start5d1$ e $start6d1$. Estes impulsos indicam que o próximo ciclo de relógio de processamento de dados ($clkb$) começa a ler valores da memória RAM e da memória ROM. Implementou-se desta forma para que o sincronismo com os restantes sinais fosse feito diretamente evitando desta forma lógica extra.

Dependendo do rácio de conversão, são usadas diferentes combinações destes impulsos, porque como foi explicado anteriormente estes sinais correspondem às condições responsáveis pela adição de amostras e a navegação entre essas condições é específica ao rácio de conversão como pode ser

consultado na tabela 5.4. No entanto, todos os rácios de conversão usam o sinal $start0p$, pois este sinal marca o momento em que a memória RAM fica preenchida pela primeira vez e permite iniciar o algoritmo para estimar o valor da primeira amostra de saída. O sinal $en\_start$ tem para todos os rácios de conversão a lógica para o início ao processo de leitura das memórias RAM e ROM, como se pode ver no segmento de código:

```
assign en_start = (src_ratio == 4'b0000)? start0p || start1p || start2p || start3p :
                  (src_ratio == 4'b0001)? start0p || start4p :
                  (src_ratio == 4'b0010)? start0p || start1p || start5p :
                  (src_ratio == 4'b0100)? start0p || start1p || start5p :
                  (src_ratio == 4'b0101)? start0p || start1p || start5p :
                  (src_ratio == 4'b0110)? start0p || start1p || start5p :
                  (src_ratio == 4'b0011)? start0p || start1p || start3p || start6p :
                  0;
```

O sexto passo consiste em criar o sinal lógico $en\_read$ para controlar os estados `READ` e `FULL`. Existe um registo auxiliar designado por $read\_counter$ para fazer a contagem da quantidade de elementos que já foram lidos. Há sempre a necessidade de explicar o que acontece se houver um sinal de reinício do circuito. Neste caso atribui-se $en\_read = 0$ e reinicia-se o contador $read\_counter$ a 0.

A largura do sinal lógico $en\_read$ depende se o fator de conversão do porto $src\_ratio$ é maior ou menor que 1. Se o sinal conectado ao porto $src\_ratio$ for menor que 1, o registo auxiliar $read\_counter$ conta até `RAM_DEPTH`. Se por outro lado for maior que 1, o registo auxiliar $read\_counter$ conta até $lenght$. Enquanto o contador $read\_counter$ não for igual ao limite da contagem, significa que ainda existem valores nas memórias para serem lidos e por essa razão o $en\_read = 1$. Quando o contador atingir o último elemento significa que já leu todos os elementos que podia, portanto o valor lógico de $en\_read$ fica a 0. A razão de existir a distinção com o valor do rácio de conversão tem sobretudo a ver com a forma como os coeficientes são utilizados.

No caso em que o rácio de conversão é menor que 1, são usados `RAM_DEPTH` coeficientes de cada vez para cada conjunto único. Por exemplo, para o rácio de conversão 0,5 quando se preenche pela primeira vez a memória RAM com o número suficiente de amostras de entrada é realizada a estimativa da primeira amostra de saída com o primeiro conjunto único de coeficientes composto por `RAM_DEPTH` coeficientes. De seguida são escritas duas novas amostras de entrada na memória RAM e é usado o segundo conjunto único de coeficientes composto por `RAM_DEPTH` coeficientes para estimar a segunda amostra de saída. Para a terceira amostra de saída são escritas duas novas amostras de entrada na memória RAM e é usado novamente o primeiro conjunto único de coeficientes, repetindo este processo continuamente.

No caso em que o rácio de conversão é maior que 1, são usados $lenght$ coeficientes, correspondendo aos coeficientes de todos os conjuntos únicos de coeficientes. Por exemplo, para o rácio de conversão 2, quando se preenche pela primeira vez a memória RAM com o número suficiente de amostras de entrada é realizada a estimativa da primeira amostra de saída com o primeiro conjunto único de coeficientes composto por `RAM_DEPTH` coeficientes. De seguida é usado o segundo conjunto único de coeficientes composto por `RAM_DEPTH` coeficientes para realizar a estimativa da segunda amostra de saída sem escrever nada na memória RAM, ou seja percorre $lenght$ elementos de uma só vez. De seguida é escrita na memória RAM uma nova amostra. Para a terceira amostra de saída usa-se o

primeiro conjunto único de coeficientes, e para a quarta amostra de saída usa-se o segundo conjunto único de coeficientes, repetindo este processo continuamente.

O sinal que marca o início da contagem é $en\_start$. Se o contador $read\_counter$ chegar ao valor limite da contagem é atribuído o valor $'bx$ ou seja não interessa o seu valor. Caso contrário é incrementado o seu valor.

Há um contador semelhante a $read\_counter$ mas dedicado unicamente aos coeficientes da memória ROM. Também é um registo auxiliar no sentido que os seus valores são usados para lógica de endereçamento. Esse registo auxiliar é o sinal $coef$. Tal como o contador $read\_counter$, se houver um sinal de reinício o seu valor passa a ser nulo. Caso contrário, se houverem elementos suficientes na memória RAM podem acontecer dois cenários: no primeiro o contador $coef$ leu os $lenght$ coeficientes todos e nesse caso reinicia o valor do contador a 0, no segundo cenário o sinal $en\_read$ está ativo e como estão a ser lidos coeficientes da memória ROM o contador $coef$ tem de ser incrementado. Em todas as outras situações o contador $coef$ mantém o seu valor.

A sétima etapa corresponde aos endereços da memória RAM. Essencialmente há dois registos $addr$ e $addr1$. O registo $addr$ é um contador interno responsável pela lógica de geração de endereços, enquanto o registo $addr1$ corresponde ao registo $addr$ atrasado de um ciclo de relógio $clkb$. O registo $addr1$ é encaminhado para o porto de saída $addrb$ se o estado da máquina de estados for `READ`. Caso contrário o valor desse porto não interessa. Se $rst = 1$, o valor do contador é $addr <= 0$. Caso contrário, o contador pode ser incrementado com algumas condições. Se o sinal $en\_start = 1$ e se estiver no estado `FULL`, reinicia-se o valor do contador com $addr <= 0$. Se a RAM estiver preenchida e se estiver no estado `READ`, é incrementado o valor do contador. Se leu as amostras todas da RAM, coloca-se o valor do contador a $addr <=' bx$.

A oitava etapa faz o correto endereçamento da memória ROM. Foram abordados na secção 5.1 alguns detalhes da configuração dos coeficientes na memória ROM. A ideia é ter uma única memória ROM com todos os coeficientes dos rácios de conversão previamente definidos. Os coeficientes ocupam posições sequenciais de memória por ordem crescente do rácio de conversão. No entanto, há algumas situações onde é preciso controlar o ponteiro dos endereços de memória. Essas situações vão depender do fator de interpolação, do fator de decimação e da quantidade de elementos lidos das memórias. Quanto ao fator de interpolação, fornece a quantidade de conjuntos únicos de coeficientes. O fator de decimação (M) é dado pelo denominador da forma em fração irredutível, que para os rácios de conversão (0,4; 0,5; 0,6666; 0,75; 2; 3; 4) são (5, 2, 3, 4, 1, 1, 1). Representam o *offset* dos coeficientes na memória ROM, após a utilização de um conjunto único de coeficientes. Será apresentado um exemplo futuramente, quando forem explicados os restantes sinais de endereçamento. Existem os registos $offset1$, $offset2$, $offset3$ e $offset4$ que apontam para os novos endereços iniciais dos conjuntos únicos de coeficientes; e o registo $index$ para percorrer os endereços da memória ROM. Há 4 registos $offset$ porque existem 4 conjuntos únicos de coeficientes para esta implementação, devido ao rácio de conversão 4. Existem ainda os sinais lógicos $en\_offset1$, $en\_offset2$ e $en\_offset3$ cuja função envolve sinalizar o momento em que há o risco de aceder a posições de memória que não correspondem às posições de memória do intervalo de coeficientes desse conjunto único de coeficientes. Estes

sinais lógicos são exclusivos para rácios de conversão menores que 1, e são 3 sinais destes porque no máximo existe esse risco para os 3 conjuntos únicos de coeficientes usados no rácio de conversão 0,75. A razão para a exclusividade está relacionada com a implementação, mais concretamente com a forma como se definiram as condições, em que para rácios de conversão maiores que 1 usou-se uma igualdade, e para rácios menores que 1 usou-se uma inequação.

A configuração da memória ROM em caso de reinício do circuito está apresentado na tabela 5.6.

| Rácio de conversão | offset1 | offset2 | offset3 | offset4 | index |
|---|---|---|---|---|---|
| 0,4 | 0 | RAM_DEPTH | | | 0 |
| 0,5 | 3×RAM_DEPTH-2 | | | | 2×RAM_DEPTH |
| 0,6666... | 3×RAM_DEPTH | 4×RAM_DEPTH | | | 3×RAM_DEPTH |
| 0,75 | 5×RAM_DEPTH | 6×RAM_DEPTH | 7×RAM_DEPTH | | 5×RAM_DEPTH |
| 2 | 8×RAM_DEPTH | 9×RAM_DEPTH | | | 8×RAM_DEPTH |
| 3 | 10×RAM_DEPTH | 11×RAM_DEPTH | 12×RAM_DEPTH | | 10×RAM_DEPTH |
| 4 | 13×RAM_DEPTH | 14×RAM_DEPTH | 15×RAM_DEPTH | 16×RAM_DEPTH | 13×RAM_DEPTH |

Tabela 5.6: Configuração das posições da memória ROM em caso de reinício do circuito.

Na tabela 5.6, as células vazias representam casos não permitidos. Com a tabela 5.6 também é possível ver onde começa e acabam os conjuntos únicos de coeficientes. O primeiro conjunto único de coeficientes tem início em $offset1$ e termina em $offset2 - 1$. O segundo conjunto único de coeficientes tem início em $offset2$ e termina em $offset3 - 1$. O terceiro conjunto único de coeficientes tem início em $offset3$ e termina em $offset4 - 1$ e o quarto conjunto único de coeficientes tem início em $offset4$ e termina em $offset4 + \texttt{RAM\_DEPTH} - 1$. Por exemplo para o rácio de conversão 4, o primeiro conjunto único de coeficientes está compreendido nas posições $6656 \rightarrow 7167$, o segundo conjunto único de coeficientes está compreendido nas posições $7168 \rightarrow 7679$, o terceiro conjunto único de coeficientes está compreendido nas posições $7680 \rightarrow 8191$ e o quarto conjunto único de coeficientes está compreendido nas posições $8192 \rightarrow 8703$.

Se for colocado um rácio de conversão inválido, é atribuído o valor 0 aos registos da oitava etapa. Em todos os rácios de conversão, caso a memória RAM tenha amostras de entrada suficientes guardadas e se a máquina de estados estiver no estado de leitura READ, prossegue para o próximo coeficiente da memória ROM ao incrementar por uma unidade o registo $index$. De seguida, é verificado se já leu um conjunto único de coeficientes, se sim ajusta o endereço inicial do sinal $offset$ correspondente a esse conjunto único de coeficientes ao deslocar uma quantidade de posições de memória dadas pelo fator de decimação. No entanto, há o risco de entrar em posições de memória que tenham coeficientes de outros conjuntos únicos de coeficientes. Nesse caso, em vez de deslocar essas posições de memória é feita uma reinicialização do sinal $offset$ através da continuação dessa deslocação para o topo da pilha de endereços desse conjunto único de coeficientes. No final, é atribuído ao sinal $index$ o novo endereço inicial de leitura desse conjunto único de coeficientes.

O último pormenor está relacionado com os endereços para a memória ROM. Se estiverem a ser lidas amostras desta memória, significa que o sinal $index$ e o contador $coef$ estão a ser incrementados por uma unidade por cada amostra lida. Pode acontecer que o contador $coef$ indique que ainda faltam

ler coeficientes desse conjunto, mas o valor do sinal $index$ esteja no limite das posições de memória para esse conjunto único de coeficientes. Nesse caso, o $index$ é reinicializado para o fundo da pilha de endereços desse conjunto único de coeficientes.

Voltando ao exemplo anterior, é descrito o endereçamento da ROM, esclarecendo a função do fator de decimação. Relembrando que, para os rácios de conversão 2, 3 e 4, desloca-se apenas uma posição dos coeficientes na memória ROM:

- Após calcular a primeira amostra de saída, o valor do registo é $offset1 <= 7167$. O $index$, para o primeiro conjunto único de coeficientes, endereça as posições $7167$ e $6656 \rightarrow 7166$;
- Após calcular a segunda amostra de saída, o valor do registo é $offset2 <= 7679$. O $index$, para o segundo conjunto único de coeficientes, endereça as posições $7679$ e $7168 \rightarrow 7678$;
- Após calcular a terceira amostra de saída, o valor do registo é $offset3 <= 8191$. O $index$, para o terceiro conjunto único de coeficientes, endereça as posições $8191$ e $7680 \rightarrow 8190$;
- Após calcular a quarta amostra de saída, o valor do registo é $offset4 <= 8703$. O $index$, para o quarto conjunto único de coeficientes, endereça as posições $8703$ e $8192 \rightarrow 8702$;
- Após calcular a quinta amostra de saída, o valor do registo é $offset1 <= 7166$. O $index$, para o primeiro conjunto único de coeficientes, endereça as posições $7166$, $7167$ e $6656 \rightarrow 7165$;

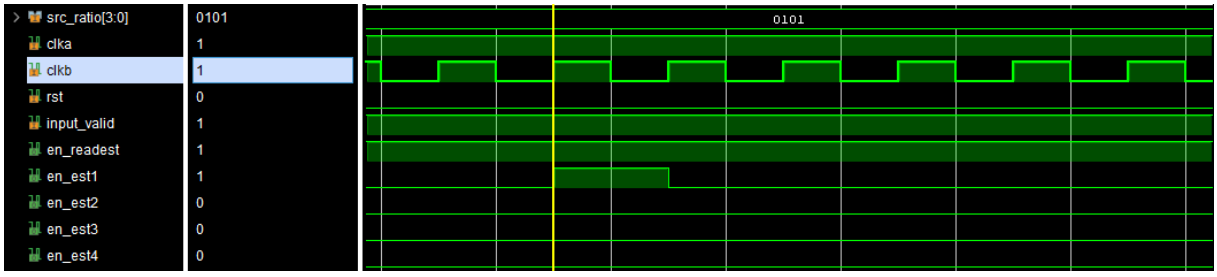
Explicando com mais detalhe o exemplo anterior, cada conjunto único de coeficientes comporta-se como uma memória circular. Quando é feito uma iteração de um conjunto único de coeficientes, os registos $offset$ indicam a posição em memória do primeiro coeficiente para esse conjunto único. A posição para onde apontam, depende do fator de decimação. Pois, esse fator representa o deslocamento em torno da posição atual dos registos $offset$. Para fatores de conversão inteiros e maiores que um, esse deslocamento é sempre de uma posição de memória. Usando, o primeiro conjunto de coeficientes como exemplo, após $rst = 1$ o valor do registo é $offset <= 6656$ e o valor do registo $index$ endereça os $RAM_DEPTH$ coeficientes seguintes. No total, $index$ terá acedido às posições da ROM $6656 \rightarrow 7167$. O deslocamento de uma posição nos endereços da ROM, faz com que se aceda ao terceiro conjunto único de coeficientes do fator de conversão 3. Portanto, é sinalizado que é necessário fazer um ajuste nos endereços, para permanecer nos limites do primeiro conjunto único de coeficientes. Neste caso, o valor usado é $offset1 <= 7167$ e representa o limite superior deste conjunto único de coeficientes. Os $RAM_DEPTH - 1$ coeficientes seguintes, usam as restantes posições de memória $6656 \rightarrow 7166$. Após calcular a quinta amostra de saída, o decremento do registo $offset1$ não endereça posições inválidas de memória, portanto é atualizado o valor para $offset1 <= 7166$. Os $RAM_DEPTH - 2$ coeficientes seguintes, usam as restantes posições de memória. O próximo coeficiente está na posição de memória seguinte $7167$, enquanto que os restantes por violarem os limites máximos (teriam que ocupar as posições de memória do segundo conjunto único de coeficientes) estão situados nos endereços $6656 \rightarrow 7165$. Este raciocínio pode ser transposto para qualquer rácio de conversão e para qualquer conjunto único de coeficientes.

A nona fase são os sinais de controlo para o circuito. Em caso de reinício do circuito os valores dos registos, $en\_est1$, $en\_est2$, $en\_est3$ e $en\_est4$ ficam com o valor 0.

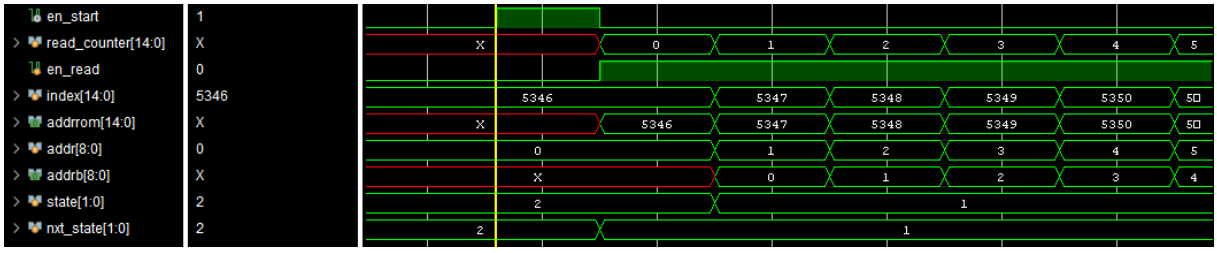
- O registo $en\_est1$ sinaliza que foram lidos os elementos do primeiro conjunto único de coeficientes;
- O registo $en\_est2$ sinaliza que foram lidos os elementos do segundo conjunto único de coeficientes;
- O registo $en\_est3$ sinaliza que foram lidos os elementos do terceiro conjunto único de coeficientes;
- O registo $en\_est4$ sinaliza que foram lidos os elementos do quarto conjunto único de coeficientes.

São atualizados a cada flanco ascendente do sinal de relógio $clkb$. O porto $en\_readest$ só fica ativo se a máquina de estados estiver no estado `READ`.

Por último, a atualização dos estados é feita quando há amostras de entrada suficientes na memória RAM ou quando o sinal $en\_read$ muda de valor. Se $en\_read = 1$, a máquina de estados muda para o estado `READ` ou permanece no estado `READ`. Se $en\_read = 0$, muda o estado para `FULL` ou permanece no estado `FULL`. A figura 5.6 mostra as transições entre cada estado da máquina de estados. Na figura 5.7 está o diagrama temporal para alguns sinais deste módulo para um rácio de conversão igual a 3.




(a) Sinais de controlo




(b) Sinais de endereço e de estados

Figura 5.7: Diagrama temporal para o módulo FSM.v.

### 5.2.4 Circuito

O último módulo da implementação em hardware é o $Circuito.v$. A sua instanciação está ilustrada na figura 5.8.

Este módulo tem 6 portos de entrada: $clka$, $clkb$, $wea$, $dina$, $src\_ratio$ e $rst$ e 2 portos de saída: $valid\_est$ e $out\_est$. Os portos de entrada já foram explicados portanto o foco vai ser os portos de saída. O porto $valid\_est$ indica a validade da amostra de saída e o porto $out\_est$ mostra o resultado da estimação.

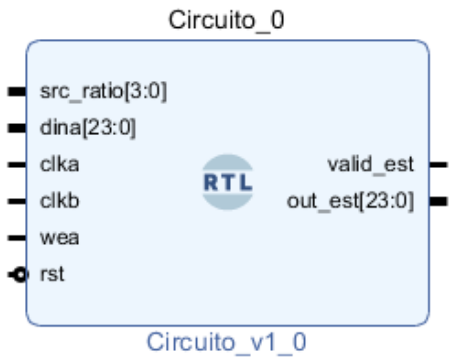



Figura 5.8: Módulo Circuito.

Para os endereços internos gerados para endereçar a memória RAM são criados dois registos: $i\_data$ e $i\_data1$. O registo $i\_data1$ é obtido pelo atraso do registo $i\_data$ por um ciclo do sinal de relógio $clka$. O endereço de escrita para a RAM é dado pelo registo $i\_data1$. O registo $i\_data$ é um contador interno. Se for permitida a escrita para a memória RAM, sinalizado através de $wea = 1$, é incrementado o valor do contador $i\_data$ e é propagado o seu valor para $i\_data1$. É reiniciado o contador $i\_data$ se tiver ocorrido uma escrita no último endereço da memória RAM.

Para os endereços da memória ROM estarem sincronizados com as amostras lidas da memória RAM é necessário atrasar o sinal de endereçamento de posições de memória $index$ por dois ciclos de relógio. O registo $index1$ é obtido atrasando o registo $index$ por um ciclo do sinal de relógio $clkb$, e o registo $index2$ é obtido atrasando o registo $index1$ por um ciclo do sinal de relógio $clkb$.

A aritmética consiste em verificar em primeiro lugar se está em modo de leitura através do sinal lógico $en\_readest2$. Este sinal é obtido através do atraso do sinal $en\_readest$ por dois ciclos do sinal de relógio $clkb$. Os atrasos dos sinais lógicos de controlo $en\_est1$, $en\_est2$, $en\_est3$, $en\_est4$ e $en\_readest$ da máquina de estados são uma consequência do modo de funcionamento da memória RAM. Se $en\_readest2 = 0$ é reiniciado o registo auxiliar $est$ com o valor 0. O registo auxiliar $est$ armazena os resultados dos cálculos efetuados durante a aritmética da estimativa das amostras de saída. De seguida, são verificados os sinais lógicos $en\_est1d3$, $en\_est2d3$, $en\_est3d3$ e $en\_est4d3$. Estes sinais são obtidos pelo atraso dos sinais lógicos $en\_est1$, $en\_est2$, $en\_est3$, $en\_est4$ por três ciclos do sinal de relógio $clkb$. Estes sinais lógicos têm como função indicar que foi lido um conjunto de coeficientes. No caso de algum destes sinais estar com o valor lógico '1' é realizada a multiplicação do valor da amostra de entrada pelo respetivo coeficiente, descartando a acumulação dos resultados anteriores.

Para a indicação que o resultado das estimativas está pronto foram criados impulsos com base nos sinais lógicos $en\_est1d3$, $en\_est2d3$, $en\_est3d3$ e $en\_est4d3$, e $en\_est1d4$, $en\_est2d4$, $en\_est3d4$ e $en\_est4d4$. O primeiro conjunto de sinais lógicos tem de ter o valor lógico '1', e o segundo conjunto de sinais lógicos tem de ter o valor lógico '0'. No entanto, o segundo conjunto de sinais lógicos é obtido através da propagação dos valores dos sinais lógicos do primeiro conjunto de sinais lógicos por um ciclo do sinal de relógio $clkb$. O sinal $valid\_est$ deteta se o valor lógico de algum destes impulsos é '1'.

Só quando o sinal $valid\_est$ indicar que o resultado da amostra de saída está pronto é que o resul-

tado aparece no porto $out\_est$ do módulo $Circuito.v$ através da ligação ao sinal $est$. Na figura 5.9 está o diagrama temporal para alguns sinais deste módulo para um rácio de conversão igual a 0,5.

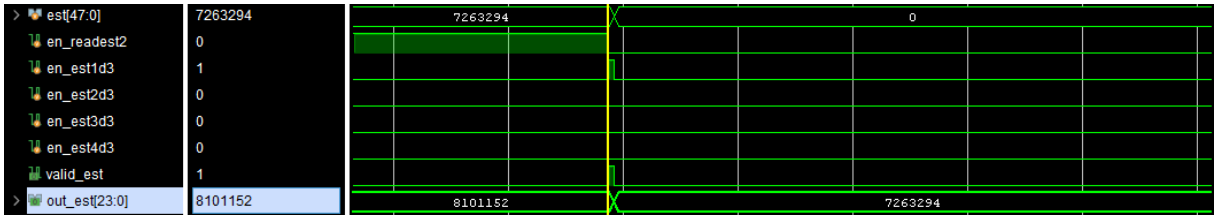



Figura 5.9: Diagrama temporal para o módulo Circuito.v.

Na figura 5.10, é apresentado o diagrama de blocos simplificado do circuito que implementa o SRC com base nos módulos descritos anteriormente.

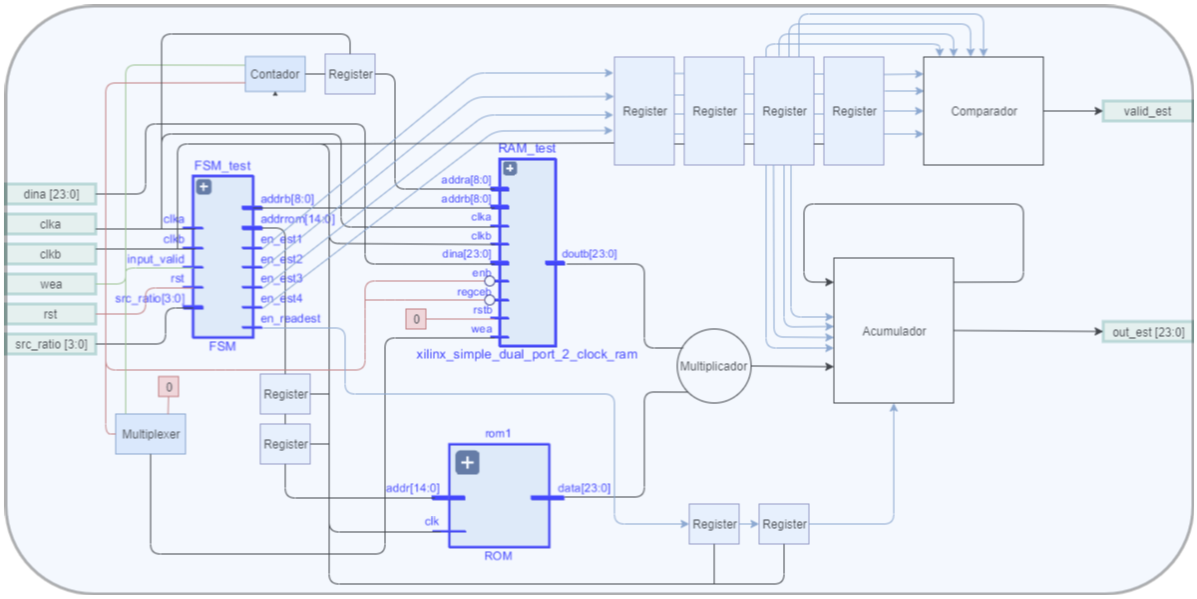



Figura 5.10: Diagrama de blocos do módulo Circuito.v.

### 5.2.5 Testbench

Foi desenvolvido um módulo de teste, $tb\_Circuito.v$, para gerar os sinais de entrada e aplicá-los no módulo $Circuito.v$, de forma a obter a resposta do conversor de frequências de amostragem. Adicionalmente, cria também um ficheiro de saída que contem as amostras de saída estimadas, o que facilita a comparação com os resultados produzidos pela implementação em software.

## 5.3 Implementação em FPGA

A figura 5.11 mostra a quantidade de recursos utilizados para o dispositivo ”XC7A12T”. Esta FPGA é a mais pequena da família ”Artix-7”. A figura 5.12, mostra os mesmos resultados mas em forma percentual. Portanto, tendo em conta os recursos indicados cumpriu-se o objetivo de usar uma implementação em hardware que ocupasse poucos recursos.

| Resource | Utilization | Available | Utilization % |
|---|---|---|---|
| LUT | 834 | 8000 | 10.43 |
| LUTRAM | 4 | 5000 | 0.08 |
| FF | 263 | 16000 | 1.64 |
| BRAM | 12.50 | 20 | 62.50 |
| DSP | 2 | 40 | 5.00 |
| IO | 57 | 112 | 50.89 |

Figura 5.11: Recursos usados na FPGA.

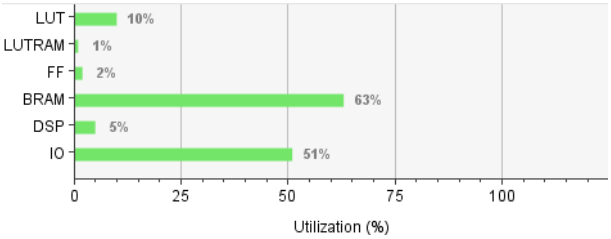



Figura 5.12: Recursos percentuais usados na FPGA.

Foi calculado a latência do circuito. A primeira iteração precisa de 516 ciclos do sinal de relógio $clkb$, as restantes iterações necessitam apenas de 512 ciclos de relógio. Esse número é consistente com a quantidade de amostras e de coeficientes que têm de ser lidos das memórias RAM e ROM.

# Capítulo 6

# Resultados

Antes de se apresentarem os resultados experimentais é conveniente definir o valor mínimo teórico que a TDH+N pode atingir. Há um limite de -144 dB devido à quantização com 24 bits. Ao calcular a TDH+N esse erro provocado pela quantização é contabilizado como ruído.

Começando por um etapa muito inicial na implementação onde só se tinham rácios de conversão inteiros maiores que 1, foi realizado um varrimento das frequências do sinal de entrada para várias ordens do filtro interpolador e testaram-se várias frequências de amostragem de entrada e rácios de conversão. A figura 6.1 mostra algumas configurações para frequências de amostragem típicas (44,1 kHz e 48 kHz) e rácios de conversão 2 e 4.

| | FS in | 48,000 | | | | | |
|---|---|---|---|---|---|---|---|
| | ratio | 2 | | | | | |
| | FS out | 96,000 | | | | | |
| | order of polynomial | 16 | 32 | 64 | 128 | 256 | 512 |
| signal freq | 1000 | -125.7 | -120.79 | -116.24 | -111.55 | -106.36 | -100.99 |
| | 5000 | -125.93 | -120.72 | -116.48 | -111.49 | -106.39 | -100.99 |
| | 10000 | -87.398 | -120.4 | -116.42 | -111.8 | -106.68 | -101.32 |
| | 15000 | -41.505 | -69.791 | -115.57 | -111.95 | -106.89 | -101.48 |
| | 20000 | -15.076 | -22.508 | -34.768 | -56.624 | -97.445 | -101.98 |

(a) Variação para uma frequência de amostragem de 48kHz e rácio de conversão 2

| | FS in | 44,100 | | | | | |
|---|---|---|---|---|---|---|---|
| | ratio | 4 | | | | | |
| | FS out | 176,400 | | | | | |
| | order of polynomial | 16 | 32 | 64 | 128 | 256 | 512 |
| signal freq | 1000 | -125.15 | -119.98 | -115.17 | -110.32 | -105.12 | -99.748 |
| | 5000 | -125.4 | -120.01 | -115.29 | -110.33 | -105.12 | -99.749 |
| | 10000 | -79.112 | -119.8 | -115.21 | -110.25 | -105.1 | -99.754 |
| | 15000 | -34.409 | -55.297 | -94.529 | -110.87 | -105.7 | -100.37 |
| | 20000 | -19.033 | -22.564 | -27.924 | -36.466 | -50.961 | -91.08 |

(b) Variação para uma frequência de amostragem de 44.1kHz e rácio de conversão 4

Figura 6.1: Primeira fase de resultados obtidos.

A análise da figura 6.1 permite constatar que à medida que a frequência do sinal se vai aproximando da frequência de Nyquist, $\frac{F_s}{2}$, cada vez mais se notam os efeitos de *aliasing* e por conseguinte aumento da TDH+N. Por outro lado, para frequências do sinal baixas (por exemplo, 1 kHz) nota-se que o aumento da ordem do filtro conduz a resultados piores. Este fenómeno deve-se ao facto do polinómio de Lagrange, produzir resultados imprevisíveis entre as amostras, que são mais notórios quando a

ordem do polinómio é elevada, pois pode ter grandes oscilações entre amostras consecutivas.

Numa fase inicial procurou-se melhorar a qualidade dos resultados usando um filtro anti-*aliasing*, implementando com recurso a um filtro FIR. Com este intuito desenvolveram-se dois programas em $Mathematica$, um deles para comprovar os efeitos da filtragem num sinal interpolado e o outro programa para calcular os coeficientes do filtro passa-baixo FIR. Normalmente para evitar *aliasing* o filtro deve cortar em $\frac{F_s}{2}$, ou seja, $\pi$, mas como o filtro não é ideal, vai haver um pouco de *aliasing*. Estabelecendo uma dada frequência mínima de entrada (por exemplo 44,1 kHz) então é possível cortar um pouco mais cedo. Tentou-se manipular as bandas de passagem e de transição de forma a ter melhor TDH+N nas amostras de saída geradas. A figura 6.2 mostra a TDH+N obtida com a utilização de filtros digitais FIR com a função de anti-*aliasing*. Nesta fase da implementação já era possível realizar a conversão para qualquer rácio de conversão $\frac{L}{M}$.

| SEM FILTRO | | | order of polynomial | |
|---|---|---|---|---|
| FS in | FS out | src_ratio | 256 | 512 |
| 44,100 | 96,000 | 2.176871 | -39.826 | -66.3 |
| 44,100 | 48,000 | 1.088435 | -39.978 | -68.528 |
| signal freq | 20,000 | | | |
| Number of Periods | 250 | | | |

(a) Tabela de referência

| | | | order of polynomial | |
|---|---|---|---|---|
| FS in | FS out | src_ratio | 256 | 512 |
| 44,100 | 96,000 | 2.176871 | -61.96 | -89.085 |
| 44,100 | 48,000 | 1.088435 | -2.7994 | -30.062 |
| signal freq | 20,000 | | | |
| Number of Periods | 250 | | | |
| a=EquiripplеFilterKernel[{{{0,0.46 Pi},{0.46 Pi+0.04,Pi}},{1,0}},128] | | | | |

(b) Configuração 1

| | | | order of polynomial | |
|---|---|---|---|---|
| FS in | FS out | src_ratio | 256 | 512 |
| 44,100 | 96,000 | 2.176871 | -61.385 | -88.356 |
| 44,100 | 48,000 | 1.088435 | -38.338 | -66.298 |
| signal freq | 20,000 | | | |
| Number of Periods | 250 | | | |
| a=EquiripplеFilterKernel[{{{0,0.45 Pi},{0.45 Pi+0.05,Pi}},{1,0}},128] | | | | |

(c) Configuração 2

Figura 6.2: TDH+N com filtros anti-aliasing digitais.

É visível que a configuração 2 melhora a TDH+N face à referência (figura 6.2 (a)) relativamente ao $src\_ratio$ igual a 2,176871, no entanto piora um pouco para o $src\_ratio$ igual a 1,088435. Usaram-se estes rácios porque na altura, eram os únicos rácios que tinham uma TDH+N superior a -90 dB, que era o valor máximo definido como meta a atingir para ter uma boa qualidade de áudio. Relativamente às funções *EquiripplеFilterKernel* servem para criar um filtro com oscilações iguais na banda de passagem e os parâmetros que recebe são a localização da banda de passagem, a localização da banda de atenuação, os valores nessas bandas e a ordem do filtro.

Decidiu-se não usar o filtro anti-*aliasing* e usar uma ordem mais elevada. O filtro permitia baixar a ordem do polinómio, mas implicava mais hardware na sua implementação. Eliminando o filtro, basta haver o hardware para implementar o polinómio de Lagrange e usar uma memória um pouco maior

para guardar mais coeficientes.

A implementação final em software produziu os resultados da figura 6.3.

| SEM FILTRO (PRODUTO) | | | order of polynomial | | | | |
|---|---|---|---|---|---|---|---|
| FS in | FS out | src_ratio | 128 | 256 | 512 | 768 | 1024 |
| 44,100 | 96,000 | 2.176871 | -76.65 | -76.64 | -76.63 | -76.63 | -76.62 |
| 48,000 | 96,000 | 2.000000 | -111.55 | -106.35 | -100.98 | -97.82 | -96.10 |
| 44,100 | 48,000 | 1.088435 | -76.64 | -76.63 | -76.62 | -76.62 | -76.62 |
| 48,000 | 44,100 | 0.918750 | -76.07 | -76.07 | -76.07 | -76.07 | -76.07 |
| 96,000 | 48,000 | 0.500000 | -143.84 | -143.85 | -143.84 | -143.86 | -143.85 |
| 96,000 | 44,100 | 0.459375 | -76.02 | -76.02 | -76.02 | -76.02 | -76.02 |
| number periods | | | signal freq | | | | |
| 250 | | | 1,000 | | | | |

(a) Frequência de sinal de 1 kHz.

| | | | order of polynomial | | | | |
|---|---|---|---|---|---|---|---|
| FS in | FS out | src_ratio | 128 | 256 | 512 | 768 | 1024 |
| 44,100 | 96,000 | 2.176871 | -25.48 | -39.81 | -66.19 | -80.68 | -81.08 |
| 48,000 | 96,000 | 2.000000 | -56.62 | -97.46 | -102.00 | -98.77 | -98.28 |
| 44,100 | 48,000 | 1.088435 | -25.69 | -39.87 | -66.45 | -80.77 | -81.11 |
| 48,000 | 44,100 | 0.918750 | -56.62 | -80.05 | -80.14 | -80.15 | -80.29 |
| 96,000 | 48,000 | 0.500000 | -128.58 | -128.58 | -128.58 | -128.18 | -129.05 |
| 96,000 | 44,100 | 0.459375 | -80.14 | -80.14 | -80.14 | -80.14 | -80.14 |
| number periods | | | signal freq | | | | |
| 500 | | | 20,000 | | | | |

(b) Frequência de sinal de 20 kHz.

Figura 6.3: Resultados finais.

Da análise da figura, constata-se que independentemente da frequência do sinal para os rácios de conversão 0,5 e 2 as TDH+N são sempre inferiores a -90 dB se a ordem do polinómio for maior ou igual a 256. Para a frequência de sinal de 1 kHz, excluindo esses dois rácios de conversão, nenhum rácio de conversão conseguiu atingir o patamar dos -90 dB, ficando-se pela gama dos -76 dB. Para a frequência de sinal de 20 kHz a TDH + N melhora com a ordem do polinómio. Optou-se por escolher a ordem de polinómio igual a 512, pois consegue-se um compromisso entre quantidade de coeficientes armazenados e a qualidade de conversão. Com esta frequência de sinal e ordem de polinómio, com exceção do rácio de conversão 0,5 e 2, os rácios de conversão 2,176871 e 1,088435 ficam na gama dos -66 dB e os rácios de conversão 0,918750 e 0,459375 ficam na gama dos -80 dB. As causas para a fraca qualidade podem ser atribuídas à natureza oscilatória de uma aproximação polinomial com uma ordem muito elevada. Esse efeito designa-se por fenómeno de Runge. Nos casos em que é igual a 0,5 e 2 há pontos que coincidem, daí a qualidade aumentar.

A figura 6.4 mostra um excerto das amostras estimadas pela implementação em software comparativamente com as amostras estimadas pela implementação em hardware, para uma frequência de sinal de 1 kHz e um rácio de conversão de 0,4.

Como se pode verificar as amostras da implementação em hardware ou coincidem ou têm valores bastante próximos da implementação em software diferindo por um ou dois dígitos menos significativos.

Resta apenas obter os resultados da TDH+N das amostras estimadas pela implementação em hardware e compará-las com a implementação em software. As figuras 6.5 mostram vários resultados da análise da TDH+N para diferentes taxas de conversão, diferentes frequências do sinal de entrada e diferentes ordens do polinómio.

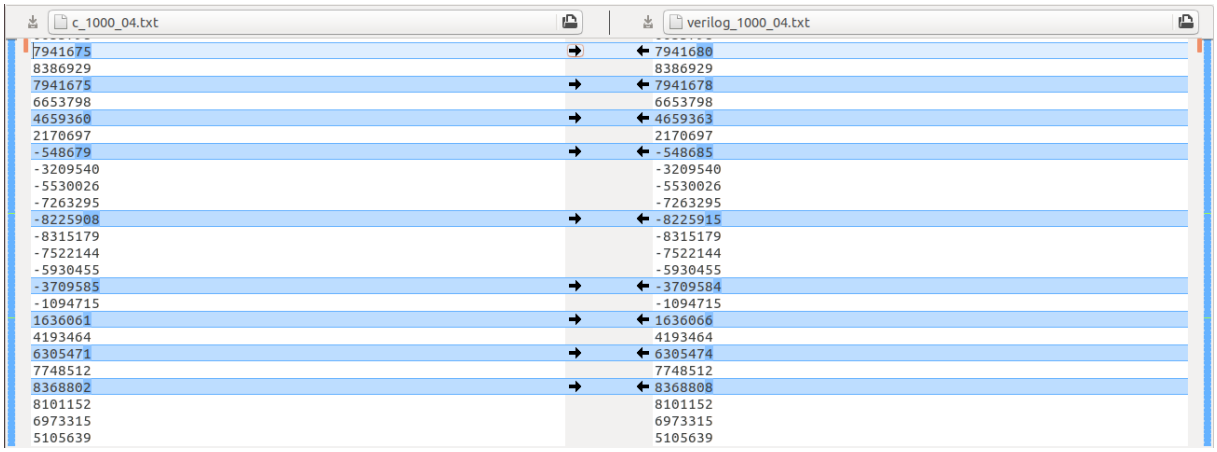



Figura 6.4: Amostras estimadas por ambas as implementações.

Os resultados da implementação em hardware são melhores que os resultados da implementação em software. Essa ocorrência acontece notoriamente para $src\_ratio = 0.6666\ldots$ como é um número com dízimas infinitas, pode acontecer que a implementação em hardware lide melhor com esse caso devido a arredondamentos. Os outros desvios entre software e hardware devem-se a imprecisões no cálculo da TDH+N. Nas figura 6.5 as células estão vazias para o rácio de conversão igual a 0,4 e frequência de sinal igual a 20 kHz, porque se está a violar a frequência de Nyquist. O caso mais notável ocorre para o rácio de conversão 0,6666... em que há uma diferença de -42 dB para uma frequência de sinal igual a 1 kHz e uma ordem de polinómio igual a 512.

Relativamente ao acréscimo da ordem do polinómio de 512 para 768 não se nota uma melhoria significativa, havendo até casos em que há um ligeiro aumento da TDH+N.

Para concluir, é de realçar que na implementação em hardware se obteve uma TDH+N sempre inferior a -95 dB, o que já é suficiente para muitas aplicações reais.

| | | | Código | |
|---|---|---|---|---|
| FS in | FS out | src_ratio | Software ( C ) | Hardware ( Verilog ) |
| 48,000 | 19,200 | 0.4 | -101.03 | -101.04 |
| 48,000 | 24,000 | 0.5 | -141.16 | -141.16 |
| 48,000 | 32,000 | 0.6667 | -79.47 | -121.65 |
| 48,000 | 36,000 | 0.75 | -98.57 | -98.57 |
| 48,000 | 96,000 | 2 | -101.05 | -101.05 |
| 48,000 | 144,000 | 3 | -98.54 | -98.54 |
| 48,000 | 192,000 | 4 | -100.02 | -100.02 |
| number periods | | | signal freq | order of polynomial |
| 250 | | | 1,000 | 512 |

(a) Frequência de sinal de 1 kHz e ordem de polinómio de 512.

| | | | Código | |
|---|---|---|---|---|
| FS in | FS out | src_ratio | Software ( C ) | Hardware ( Verilog ) |
| 48,000 | 19,200 | 0.4 | | |
| 48,000 | 24,000 | 0.5 | -122.48 | -122.48 |
| 48,000 | 32,000 | 0.6667 | -80.18 | -98.32 |
| 48,000 | 36,000 | 0.75 | -99.62 | -99.63 |
| 48,000 | 96,000 | 2 | -102.03 | -102.03 |
| 48,000 | 144,000 | 3 | -99.50 | -99.50 |
| 48,000 | 192,000 | 4 | -101.07 | -101.07 |
| number periods | | | signal freq | order of polynomial |
| 500 | | | 20,000 | 512 |

(b) Frequência de sinal de 20 kHz e ordem de polinómio de 512.

| | | | Código | |
|---|---|---|---|---|
| FS in | FS out | src_ratio | Software ( C ) | Hardware ( Verilog ) |
| 48,000 | 19,200 | 0.4 | -97.86 | -98.04 |
| 48,000 | 24,000 | 0.5 | -141.15 | -141.15 |
| 48,000 | 32,000 | 0.6667 | -79.50 | -116.51 |
| 48,000 | 36,000 | 0.75 | -95.42 | -95.57 |
| 48,000 | 96,000 | 2 | -97.88 | -98.06 |
| 48,000 | 144,000 | 3 | -95.39 | -95.57 |
| 48,000 | 192,000 | 4 | -96.92 | -97.10 |
| number periods | | | signal freq | order of polynomial |
| 250 | | | 1,000 | 768 |

(c) Frequência de sinal de 1 kHz e ordem de polinómio de 768.

| | | | Código | |
|---|---|---|---|---|
| FS in | FS out | src_ratio | Software ( C ) | Hardware ( Verilog ) |
| 48,000 | 19,200 | 0.4 | | |
| 48,000 | 24,000 | 0.5 | -122.97 | -122.97 |
| 48,000 | 32,000 | 0.6667 | -80.43 | -98.74 |
| 48,000 | 36,000 | 0.75 | -96.52 | -96.57 |
| 48,000 | 96,000 | 2 | -98.82 | -99.05 |
| 48,000 | 144,000 | 3 | -96.35 | -96.57 |
| 48,000 | 192,000 | 4 | -97.85 | -98.07 |
| number periods | | | signal freq | order of polynomial |
| 500 | | | 20,000 | 768 |

(d) Frequência de sinal de 20 kHz e ordem de polinómio de 768.

Figura 6.5: Resultados da implementação em hardware.

# Capítulo 7

# Conclusão

A realização deste trabalho tinha como objetivo implementar um conversor de frequências de amostragem que ocupasse poucos recursos e obtivesse uma TDH+N máxima de -90 dB.

Para cumprir o objetivo proposto foram criados criadas duas descrições do algoritmo. Criou-se um programa em C para ajustar os parâmetros do projeto e testar as melhores configurações. Posteriormente desenvolveu-se uma descrição em Verilog para realizar a implementação. Na implementação em Verilog foram usados quatro módulos: uma memória ROM, uma memória RAM, uma máquina de estados FSM e o circuito de topo.

Os resultados obtidos para a implementação em software, mostram que existem alguns erros de arredondamento. Nos restantes casos, cumpre sempre os requisitos da TDH+N máxima de -90 dB. Para $src\_ratio = 0.5$ atinge -141 dB, perto do limite mínimo teórico da TDH+N que são -144 dB.

Os resultados obtidos para a implementação em hardware, mostram que a TDH+N máxima é sempre atingida, sendo o valor máximo de -95 dB. A TDH+N mínima que se obteve com esta implementação foi -141 dB, sendo este valor próximo do limite mínimo teórico.

Uma das desvantagens de se usar o método de interpolação de Lagrange, deve-se ao facto dos coeficientes terem de ser pré-calculados e, dependendo dos rácios de conversão $\frac{L}{M}$, pode ser necessário usar uma ROM com grande capacidade.

No geral, mostrou-se que os recursos ocupados em FPGA são reduzidos, cumprindo o objetivo.

Como trabalho futuro seria interessante experimentar outro método de interpolação, por exemplo B-spline. No caso de se pretender manter a interpolação de Lagrange, poder-se-á recorrer a uma estrutura polifásica para o filtro FIR, o que poderia introduzir algumas melhorias. Outra técnica a explorar seria a sobre-amostragem, visto que se trata de uma técnica recorrentemente utilizada.

# Referências


[1] M. Forster. Sample rate conversion in digital signal processors. Master's thesis, Graz University of Technology, 2014.

[2] R. W. Schafer and L. R. Rabiner. A digital signal processing approach to interpolation. *Procedings of the IEEE*, 61(6):692–702, June 1973.

[3] M. Blok. On sample rate conversion based on variable fractional delay filters. *International Journal of Computer Science and Applications*, 10:98–116, 01 2013.

[4] J. Vesma. Frequency-domain approach to polynomial-based interpolation and the farrow structure. *IEEE Transactions on Circuits and Systems — II: Analog and Digital Signal Processing*, 47(3): 206–209, Mar. 2000.

[5] G. Evangelista. Design of digital systems for arbitrary sampling rate conversion. *Signal Processing*, 83:377–387, 02 2003. doi: 10.1016/S0165-1684(02)00421-8.

[6] S. Park, G. Hillman, and R. Robles. A novel structure for real-time digital sample-rate converters with finite precision error analysis. In *[Proceedings] ICASSP 91: 1991 International Conference on Acoustics, Speech, and Signal Processing*, pages 3613–3616 vol.5, April 1991. doi: 10.1109/ICASSP.1991.151056.

[7] R. Lagadec, D. Pelloni, and D. Weiss. A 2-channel, 16-bit digital sampling frequency converter for professional digital audio. In *ICASSP '82. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 7, pages 93–96, May 1982. doi: 10.1109/ICASSP.1982.1171661.

[8] R. Adams and T. Kwan. A stereo asynchronous digital sample-rate converter for digital audio. *IEICE TRANSACTIONS on Electronics*, E77-C(5):811–818, May 1994. ISSN 0916-8516.

[9] F. Overney. Synchronization of sampling-based measuring systems. *IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT*, 63(1):89–95, January 2014.

[10] J. Singh. A novel architecture for sample rate converter using fir filters. Master's thesis, Thapar University, July 2015.

[11] R. E. Crochiere and L. R. Rabiner. *Multirate Digital Signal Processing*. Signal Processing Series. Prentice-Hall, Prentice Hall, Inc., Englewood Cliffs, NJ, 1983.

[12] A. Franck. Efficient algorithms for arbitrary sample rate conversion with application to wave field synthesis, 2012.

[13] I. Løkken. The ups and downs of arbitrary sample rate conversion, 12/4-05.

[14] A. A. Chunduri SreenivasaRao and D. VenkataRao. Synchronous sampling rate conversion using frequency domain based techniques. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 6(4):13–28, August 2013.

[15] F. M. Rothacher. *Sample-Rate Conversion: Algorithms and VLSI Implementation*. PhD thesis, Swiss Federal Institue of Technology Zurich, 1995.

[16] R. W. Schafer and L. R. Rabiner. Interpolation and a decimation of digital signals-tutorial review. *Procedings of the IEEE*, 69(3):300–331, 1981.

[17] R. M. A. Ponce and G. Jovanovic-Dolecek. A multirate approach to design of fractional delay filters, 1999.

[18] U. R. Andreas Franck, Karlheinz Brandenburg. Efficient delay interpolation for wave field synthesis. *Audio Engineering Society Convention Paper Presented at the 125th Convention*, page 1–17, October 2008.

[19] T. I. Laakso, V. Valimaki, M. Karjalainen, and U. K. Laine. Splitting the unit delay [fir/all pass filters design]. *IEEE Signal Processing Magazine*, 13(1):30–60, Jan 1996. doi: 10.1109/79.482137.

[20] J. Vesma and T. Saramaki. Polynomial-based interpolation filters—part i: Filter synthesis. *Circuits Systems Signal Processing*, 26(2):115–146, Apr 2007.

[21] M. R. Djordje Babic, Vesa Lehtinen. Discrete time modeling of polynomial-based interpolation filters in rational sampling rate conversion, 2003.

[22] D. Babic. Polynomial-based filters in bandpass interpolation and sampling rate conversion. In *5th workshop on spectral methods and multirate signal processing SMMSP*, 2006.

[23] J. Vesma. A frequency-domain approach to polynomial-based interpolation and the farrow structure. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 47(3): 206–209, 2000.

[24] T. S. Robert Bregović, Ya JunYu and Y. C. Lim. Implementation of linear-phase fir filters for a rational sampling-rate conversion utilizing the coefficient symmetry. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS*, 58(3):548–561, March 2011.

[25] G. Jovanovic-Dolecek and S. K. Mitra. A new two-stage sharpened comb decimator, 2005.

[26] V. Awasthi and K. Raj. Power performance analysis of compensated cascaded integrator comb (cic) filter in optimum computing. In *2012 1st International Conference on Power and Energy in NERIST (ICPEN)*, pages 1–6, Dec 2012. doi: 10.1109/ICPEN.2012.6492323.

[27] H. Göckler. Polyphase realization of fractional sample rate converters. In *ECCTD*, volume 99, pages 405–408, 1999.

[28] L. R. Rabiner. Digital techniques for changing the sampling rate of a signal. In *Audio Engineering Society Conference: 1st International Conference: Digital Audio*, Jun 1982. URL `http://www.aes.org/e-lib/browse.cfm?elib=3413`.

[29] G. Bi and S. Li. Sampling rate conversion based on dct ii/iii, 2013.

[30] N. Takahashi and S. M. Daisuke Takago. Efficient multistage implementation of rational sampling rate converter, 2013.

[31] G. Bi. Minimization of delay requirements for rational sampling rate alternating systems, 1991.

[32] O. Gustafsson and H. Johansson. Efficient implementation of fir filter based rational sampling rate converters using constant matrix multiplication, 2006.

[33] A. I. Russell. Efficient rational sampling rate alteration using iir filters. *IEEE SIGNAL PROCESSING LETTERS*, 7(1):6–7, January 2000.

[34] C.-C. Hsiao. Polyphase filter matrix for rational sampling rate conversions, 1987.

[35] Y. J. Y. Robert Bregovic, Tapio Saramaki and Y. C. Lim. An efficient implementation of linear-phase fir filters for a rational sampling rate conversion, 2006.

[36] R. M. Vandita Singh. Rational sampling rate converter design analysis using symmetric technique. *International Journal of Computer Trends and Technology (IJCTT)*, 27(2):116–120, September 2015. ISSN 231-2803.

[37] R. M. Vandita Singh. Rational sampling rate converter using coefficient symmetry. *International Journal of Computer Applications*, 129(4):1–7, November 2015. ISSN 0975–8887.

[38] N. R. Nagrale. Improved method to design rational decimation system for software defined radio: Part i - algorithm. *Robotics and Automotive Mechanics Conference*, page 85–89, October 2010. ISSN 978-0-7695-4204-1. doi: 10.1109.

[39] G. J. Dolecek. Modified cic filter for rational sample rate conversion, 2007.

[40] G. J. Dolecek and S. K. Mitra. Stepped triangular cic filter for rational sample rate conversion, 2006.

[41] Y. Medan and U. Shvadron. Asynchronous rate conversion, 1997.

[42] B. R. Pallab Midya and T. Schooler. Asynchronous sample rate converter for digital audio amplifiers. *Audio Engineering Society Convention Paper 6863*, 5(8):1–7, October 2006.

[43] Y. Katsutnata and O. Hamada. An audio sampling frequency conversion using digital signal processors, 1986.

[44] A. I. Russell and P. E. Beckmann. Efficient arbitrary sampling rate conversion with recursive calculation of coefficients. *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, 50(4):854–865, Apr 2002.

[45] T. A. Ramstad. Sample rate conversion by arbitrary ratios, 1982.

[46] D. Babic and M. Renfors. Power efficient structure for conversion between arbitrary sampling rates. *IEEE SIGNAL PROCESSING LETTERS*, 12(1):1–4, January 2005.

[47] H.-y. W. Yong-jian Xu and Z. Shen. Modified polyphase filter for arbitrary sampling rate conversion, 2010.

[48] T. Ramstad. Digital methods for conversion between arbitrary sampling frequencies. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(3):577–591, June 1984. ISSN 0096-3518. doi: 10.1109/TASSP.1984.1164362.

[49] G. P. S. Cucchi, F. Desinan and G. Sicuranza. Dsp implementation of arbitrary sampling frequency conversion for high quality sound applications, 1991.

[50] O. R. Brunel Happi Tietche and B. Denby. A practical fpga-based architecture for arbitrary-ratio sample rate conversion, 2013.

[51] A. Franck and K. Brandenburg. An overall optimization method for arbitrary sample rate converters based on integer rate src and lagrange interpolation. In *2009 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 301–304, Oct 2009. doi: 10.1109/ASPAA.2009.5346472.

[52] A. Franck. Arbitrary sample rate conversion with resampling filters optimized for combination with oversampling. In *2011 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 149–152, Oct 2011.

[53] J. Vesma and T. Saramaki. Interpolation filters with arbitrary frequency response for all digital receivers, 1996.

[54] C. Dick. Options for arbitrary resamplers in fpga-based modulators, 2004.

[55] G. K. M. Jorgovanovic, M. Pajic and J. Popovic. Fpga design of arbitrary down-sampler. *PROC. 26th INTERNATIONAL CONFERENCE ON MICROELECTRONICS (MIEL 2008)*, 11(14), May 2008.

[56] J. P. Long and J. A. Torres. High throughput farrow re-samplers utilizing reduced complexity fir filters, 2012.

[57] J. V. Jaakko Ketola and K. Halonen. Synchronization of fractional interval counters in non-integer ratio sample rate converters, 2003.

[58] A. Tarczynski, W. Kozinski, and G. D. Cain. Sampling rate conversion using fractional-sample delay. In *Proceedings of ICASSP '94. IEEE International Conference on Acoustics, Speech and Signal Processing*, volume iii, pages III/285–III/288 vol.3, April 1994. doi: 10.1109/ICASSP.1994.390042.

[59] H.-y. W. Yong-jian Xu and Z. Shen. A low-complexity b-spline based digital sample rate conversion circuit architecture, 2005.

[60] F. M. G. Lars Erup and R. A. Harris. Interpolation in digital modems-part 11: Implementation and performance. *IEEE TRANSACTIONS ON COMMUNICATIONS,*, 41(6):998–1008, June 1993.

# Anexo A

# Diagrama de blocos

Este capítulo apresenta os módulos apresentados em 5.2 com mais detalhe. Não foi possível apresentar as figuras durante a explicação do capítulo 5 devido às dimensões dos diagramas de blocos e a sua complexidade poderia causar mais confusão, logo optou-se por colocar em apêndice. A exceção é o módulo $FSM.v$ a razão para a sua exclusão é o excesso de figuras extra que se teriam de por para ver a arquitetura toda.

## A.1 Circuito

Nesta secção estão apresentados os vários segmentos de figuras que compõem a implementação em hardware do circuito (figuras A.8 até A.2).

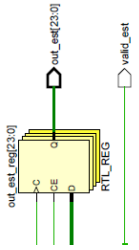



Figura A.1: Circuito (parte 8)

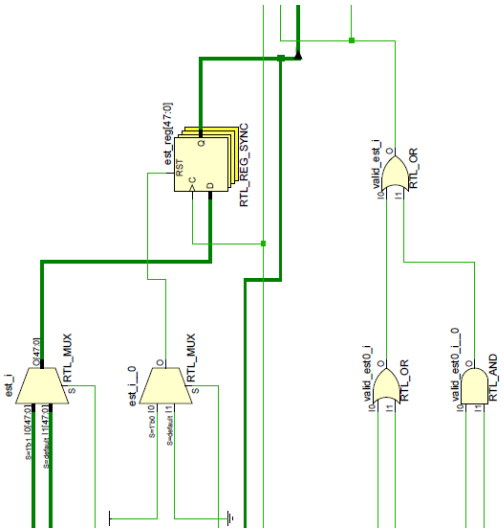



Figura A.2: Circuito (parte 7)

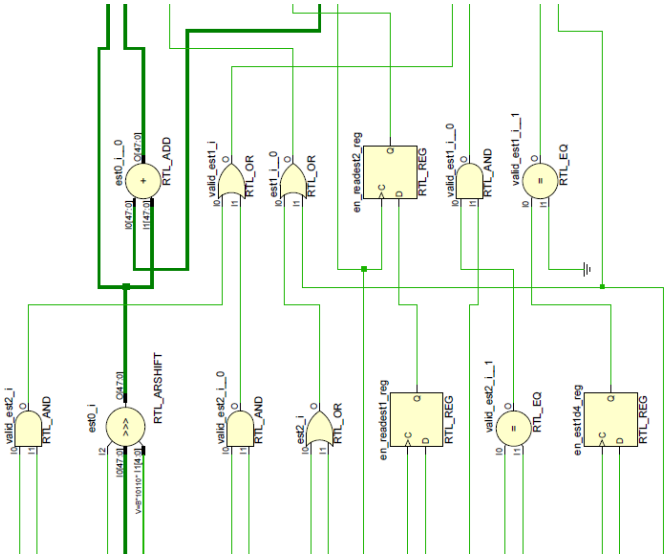



Figura A.3: Circuito (parte 6)

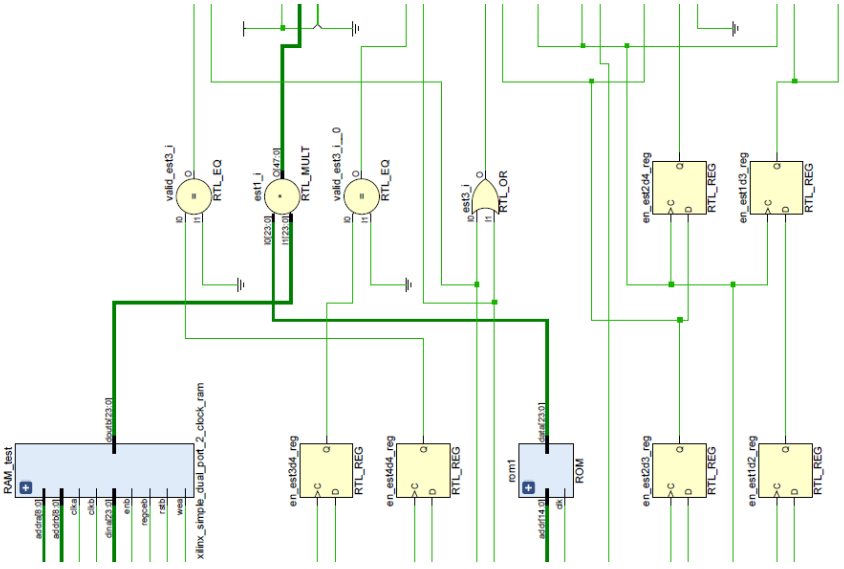



Figura A.4: Circuito (parte 5)

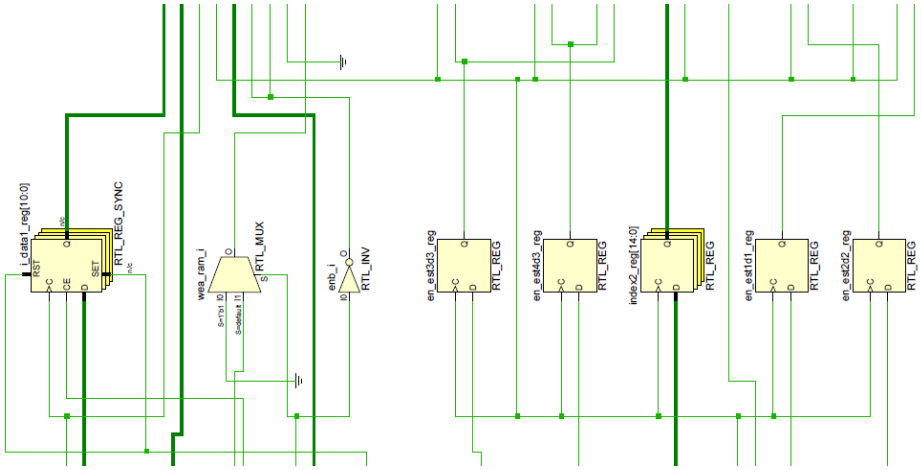



Figura A.5: Circuito (parte 4)

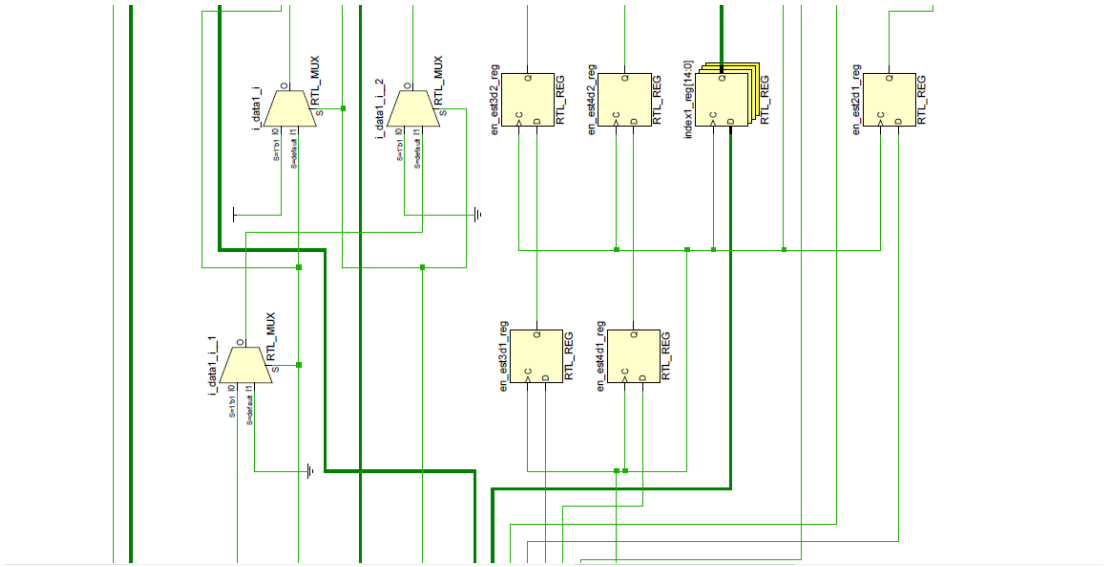



Figura A.6: Circuito (parte 3)

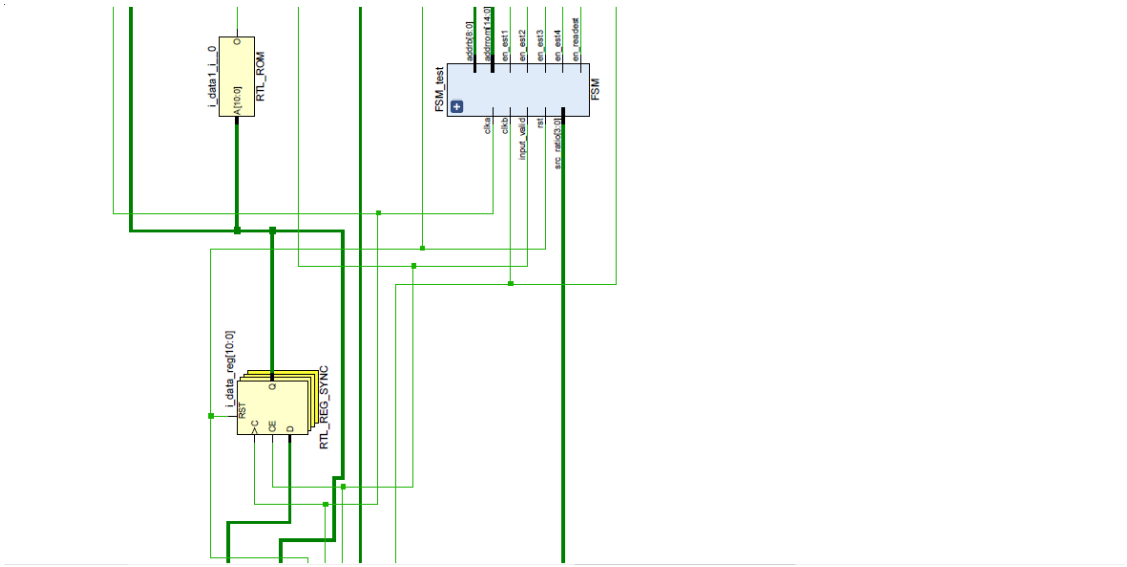



Figura A.7: Circuito (parte 2)

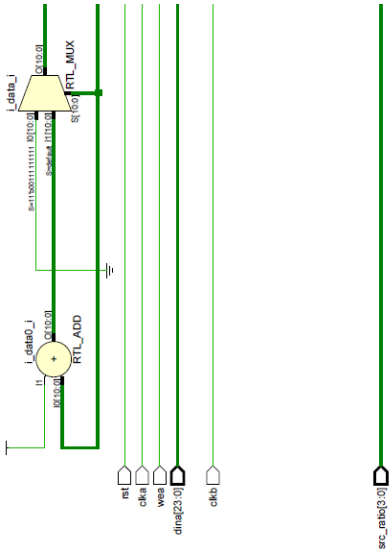



Figura A.8: Circuito (parte 1)

## A.2 RAM

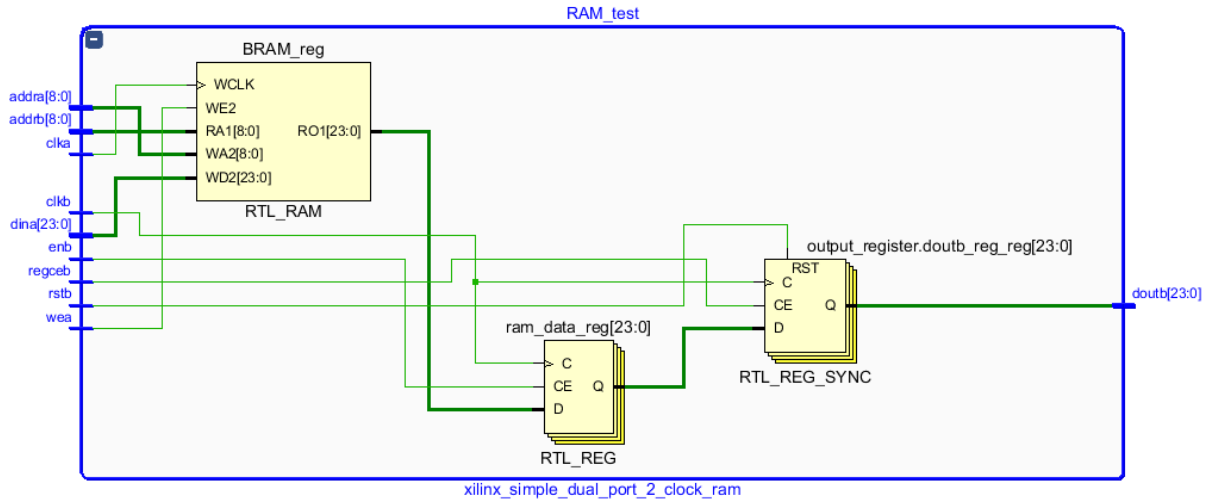



Figura A.9: Módulo RAM expandido.

## A.3 ROM

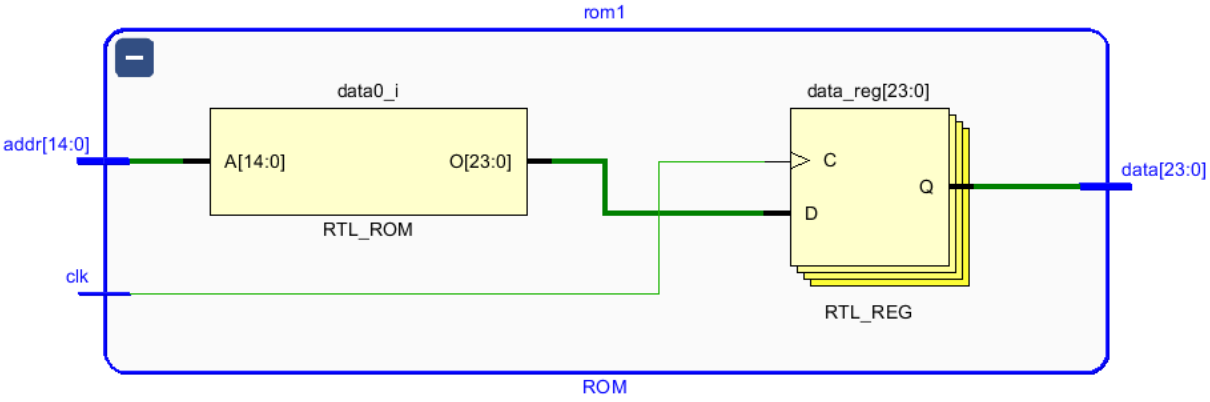



Figura A.10: Módulo ROM expandido.