Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia Eletrônica

Aceleração de algoritmos de minimização da métrica ℓ_p de sinais para Compressive Sensing com base em FPGAs

Trabalho de Conclusão de Curso 2

Autor: Amós dos Santos Nunes de Lima
Orientador: Cristiano Jacques Miosso, PhD

Brasília, DF
28 de setembro de 2022

Aceleração de algoritmos de minimização da métrica ℓ_p de sinais para Compressive Sensing com base em FPGAs

Monografia desenvolvida no âmbito da disciplina Trabalho de Conclusão de Curso 2

Amós dos Santos Nunes de Lima

Orientador: Cristiano Jacques Miosso, PhD

Engenharia Eletrônica

Universidade de Brasília

FGA - Faculdade UnB Gama

Setembro, 2022

Desenvolvimento da Universidade de Brasília.

Amós dos Santos Nunes de Lima

Aceleração de algoritmos de minimização da métrica ℓ_p de sinais para Compressive Sensing com base em FPGAs

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Cristiano Jacques Miosso, PhD

Brasília, DF

28 de setembro de 2022

Amós dos Santos Nunes de Lima

Aceleração de algoritmos de minimização da métrica ℓ_p de sinais para Compressive Sensing com base em FPGAs. – Brasília, DF, 28 de setembro de 2022-

47 p. ; 30 cm.

Orientador: Cristiano Jacques Miosso, PhD

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB Faculdade UnB Gama – FGA , 28 de setembro de 2022.

1. Compressive Sensing utilizando FPGAs. 2. Implementação da FFT em Hardware para aceleração de algoritmos. I. Cristiano Jacques Miosso, PhD. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Aceleração de algoritmos de minimização da métrica ℓ_p de sinais para Compressive Sensing com base em FPGAs.

CDU 621.38

Amós dos Santos Nunes de Lima

Aceleração de algoritmos de minimização da métrica ℓ_p de sinais para Compressive Sensing com base em FPGAs

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Trabalho aprovado. Brasília, DF, 28 de setembro de 2022:

Cristiano Jacques Miosso, PhD
Orientador

Vinícius Pinheiro, MSc
Convidado 1

Filipe Emídio Tôrres, MSc
Convidado 2

Brasília, DF
28 de setembro de 2022

Resumo

Os diversos avanços na tecnologia dos sensores de imageamento têm proporcionado uma evolução significativa na captura de figuras singulares, das quais, é possível obter uma gama de informações relevantes para o contexto atual. Por exemplo, imagens de satélites capturadas no domínio multiespectral tem auxiliado autoridades públicas a detectarem diversos crimes ambientais, tais como, atividades de pecuária ilegal, desmatamento e mineração irregular. Um outro exemplo de progresso considerável é constatado na modalidade de imageamento médico. O aumento da resolução e do contraste de imagens anatômicas têm viabilizado o diagnóstico precoce de doenças, de forma a colaborar com os profissionais de saúde, acelerando o início do tratamento apropriado.

Entretanto, principalmente para os exemplos citados, esse sensoriamento moderno possui um alto custo e tempo de captura envolvidos, de maneira que, a demora na aquisição das imagens pode inviabilizar determinadas aplicações. Tomando as máquinas de ressonância magnética (MRI) como exemplo, um simples exame de imagem sagital do encéfalo chega a custar alguns milhares de dólares e durar cerca de algumas horas. O que não é apenas caro e desconfortável, mas também é crítico nos casos de pacientes que não podem ficar muito tempo parados na máquina. Dessa forma, a redução do tempo de captura dessas imagens é de extrema relevância.

Uma técnica consolidada capaz de mitigar o tempo envolvido na aquisição de dados é o *Compressive Sensing* (CS). Como o nome sugere, o CS é uma técnica de processamento capaz de reconstruir sinais com uma taxa de amostragem *sub-Nyquist*, ou a partir de medidas lineares limitadas. O CS se aproveita da existência de domínios transformados em que os sinais a serem reconstruídos são esparsos. Basicamente, as técnicas propostas tentam resolver sistemas lineares indeterminados com a minimização da solução mais esparsa possível. Assim, desde que as condições de esparsidade e incoerência sejam respeitadas, é possível reconstruir sinais com uma alta taxa de confiabilidade utilizando um número reduzido de medidas. No caso de imagens reais, a solução de sistemas lineares indeterminados não é algo simples. Normalmente, é necessário um tempo comparativamente alto e recursos de processamento mais elevados em relação à aquisição de dados em alta densidade.

Um método que tem obtido êxito na redução do tempo de processamento de sinais e

imagens é a implementação acelerada em hardware dedicado com programação em *Field Programmable Gate Array* (FPGA). O dinamismo da programação de circuitos eletrônicos possibilita a aceleração de várias rotinas de *software*, paralelizando as etapas de cálculo, o que é impraticável para um processador normal.

À vista disso, este trabalho propõem um método para realização da implementação em FPGA de um algoritmo de minimização de ℓ_p, que é um dos principais algoritmos para reconstrução de imagem em Compressive Sensing. Substituindo os recursos utilizados para a aquisição de medidas por recursos de processamento acelerados com implementações paralelizadas em hardware.

Neste sentido, foram realizados alguns estudos com imagens reais reconstruídas em *Python*. Nos exemplos gerados foi constatado que cerca de 80% do tempo da reconstrução por CS é gasto com a resolução de sistemas lineares no método indireto. Sendo assim, inicialmente, o trabalho consistirá na implementação e análise do núcleo de processamento da *Fast Fourier Transform* (FFT), Intellectual Property (IP) disponibilizada pela empresa Xilinx, em conjunto com um processador programando em *Pyhton*, de forma que as chamadas de função da FFT serão aceleradas em *hardware*. Será avaliada a eficácia e os ganhos de tempo da abordagem citada.

Depois disso, constatado o aumento da eficiência, também será desenvolvida uma IP na liguagem de descrição de *hardware* VHDL (Very High Speed Integrated Circuits Hardware Description Language) própria para paralelizar a resolução dos sistemas lineares utilizando o método indireto conhecido como *Conjugate Gradient* (CG). A princípio com sinais de pequenas dimensões (para viabilizar a implementação) e, posteriormente, será realizada a validação da implementação, comparando o tempo de processamento gasto em *software* com o processamento em *hardware*.

Finalmente, será feita uma avaliação da viabilidade da implementação do *Compressive Sensing* completo em FPGA utilizando as IPs do *Conjugate Gradient* e da FFT para processamento em *System on a Chip* (SoC), utilizando técnicas de CS avançadas, como a pré-filtragem, para reconstrução de imagens reais.

Palavras-chave: Compressive Sensing, Minimização de ℓ_p, FFT em SoC, FPGA, Conjugate Gradient, Pré-filtragem

Abstract

Compressive Sensing (CS) is a consolidated technique capable of mitigating the time involved in data acquisition. The CS allows the reconstruction of sub-Nyquist sampled signals or from limited linear measurements. However, in the case of satellite images captured in the multispectral domain or magnetic resonance imaging, high time and higher processing resources are required compared to high-density data acquisition.

Given this, the work proposes an analysis for FPGA acceleration of a ℓ_p minimization algorithm implemented in Python. Since, for home computers with commercial memory, the direct method of solving linear systems is unfeasible for reconstructing signals larger than 2^{15}. And, for the indirect method, about 90% of the processing time is spent solving linear systems. With naturally sparse signals in the frequency domain, it was also possible to verify that 41% of the reconstruction spent time is with reindexing and memory access, and about 35% of the reconstruction spent time is with Fourier transforms (direct and inverse).

Finally, using the Pynq-Z2 Development board, with the integrated SoC Zynq-7000, FFT and IFFT FPGA implementations were made to accelerate the CS reconstructions. And, even with the time spent with data transmission and memory access between chips, there was an increase in processing speed.

Keywords: Compressive Sensing, ℓ_p Minimization, FFT in SoC, FPGA, Conjugate Gradient, Pre-filtering

Lista de ilustrações

Figura 1 – Diagrama de resumo para reconstrução de sinais por CS utilizando pré-filtragem. 27
Figura 2 – Placa de Desenvolvimento Pynq-Z2, com SoC Zynq-7000 integrado. (Imagem da Internet) 32
Figura 3 – Comparação de tempo gasto na resolução do sistema linear entre o método direto e o método indireto. 34
Figura 4 – Comparação do erro de 50 reconstruções de uma imagem 2D simples esparsa variando o número de medidas. 36
Figura 5 – Comparação entre um imagem esparsa gerada computacionalmente e sua reconstrução com CS. 36
Figura 6 – Dados do tempo total gasto comparado ao tempo gasto com a resolução do sistema linear para 50 reconstruções de uma imagem 2D simples esparsa. 37
Figura 7 – Dados de reconstruções de uma imagem real com Pré-Filtragem variando o número de trajetórias angulares medidas. 39
Figura 8 – Comparação entre uma imagem real e sua reconstrução com CS. 39
Figura 9 – Comparação do tempo de processamento da FFT em *hardware* e em *software* variando o tamanho do sinal com 10 mil interações. 40
Figura 10 – Comparação do tempo de processamento da IFFT em *hardware* e em *software* variando o tamanho do sinal com 10 mil interações. 41
Figura 11 – Comparação do tempo de processamento da FFT em *hardware* com reconfiguração e em *software* variando o tamanho do sinal com 10 mil interações. 42
Figura 12 – Comparação do tempo de processamento da IFFT em *hardware* com reconfiguração e em *software* variando o tamanho do sinal com 10 mil interações. 42
Figura 13 – Comparação do tempo de processamento de 10 mil reconstruções de CS com a FFT acelerada em *hardware* variando o tamanho do sinal. . . 43
Figura 14 – Diagrama para implementação em FPGA do Conjugate Gradient . . . 47

Lista de tabelas

Tabela 1 – Dados do tempo total gasto comparado ao tempo gasto com a resolução do sistema linear para a reconstrução de uma imagem 2D simples esparsa com 6.15% das medidas. 38

Lista de abreviaturas e siglas

CS	Compressive Sensing
CG	Conjugate Gradient
FPGA	Field-Programmable Gate Array
HDL	Hardware Description Language
VHDL	Very High Speed Integrated Circuits Hardware Description Language
IRLS	Iteratively Reweighted Least Squares
FFT	Fast Fourier Transform
SoC	System on a Chip
IP	Intellectual Property
MRI	Máquina de Ressonância Magnética
CI	Circuito Integrado
MSE	Erro Quadrático Médio

Sumário

1 INTRODUÇÃO 13
1.1 Problemas de imageamento e a necessidade de formação de imagens em tempo hábil para diferentes aplicações 13
1.2 Técnica para formação de imagens a partir de medidas limitadas . . 14
1.2.1 Breve Introdução ao *Compressive Sensing* (CS) 15
1.2.2 A complexidade computacional da reconstrução em CS 15
1.3 Proposta Científica 16
1.4 Objetivos 17
1.4.1 Objetivo Geral 17
1.4.2 Objetivos Específicos 17
1.5 Estrutura do Texto 18

2 EMBASAMENTO TEÓRICO DO *COMPRESSIVE SENSING* . . . 20
2.1 Conceitos formais de *Compressive Sensing* 21
2.2 Minimização de ℓ_2 23
2.3 Minimização de ℓ_p com IRLS 24
2.4 Solução Indireta para sistemas lineares em algorítimo de CS 25
2.5 Método de *Conjugate Gradient* para Soluções Indiretas 25
2.6 Método de Pré-Filtragem em CS 26

3 MATERIAIS E MÉTODOS 28
3.1 Implementação de algoritmos de CS em um computador sequencial pessoal 28
3.1.1 Programas necessários para validação dos algoritmos de minimização em Python 28
3.1.1.1 Gerador de sinais 1D com com representação transformada esparsa 28
3.1.1.2 Gerador da matriz de medidas para imagens esparsas e pré-filtradas 29
3.1.2 Desenvolvimento dos algoritmos para reconstrução em CS 29
3.2 Experimentos Realizados 29
3.2.1 Comparações entre o método direto e indireto para processos estocásticos em domínio 1D 30
3.2.2 Reconstrução por CS de imagens já esparsas no domínio original 30
3.2.3 Reconstrução de uma imagem real com Pré-Filtragem 30

3.3 Implementações em *Hardware* de etapas computacionais utilizadas em CS com pré-filtragem . . . 31
3.3.1 Implementação da FFT em FPGA para acelerar parte da resolução da minimização de ℓ_p em SoC . . . 31

4 RESULTADOS PRELIMINARES E DISCUSSÕES . . . 33
4.1 Comparações entre o método direto e indireto para processos estocásticos em domínio 1D . . . 33
4.2 Reconstrução por CS de imagens já esparsas no domínio original . . . 35
4.2.1 Análise da qualidade de reconstrução por CS em função do número de medidas . . . 35
4.2.2 Análise dos tempos de reconstrução em função do número de medidas . . . 37
4.3 Reconstrução de uma imagem real com Pré-Filtragem . . . 38
4.4 Análise comparativa da FFT e IFFT implementadas em FPGA e no processador ARM . . . 40
4.5 Análise comparativa da FFT e IFFT com reconfiguração implementadas em FPGA e no processador ARM . . . 41
4.6 Análise comparativa da velocidade de processamento de uma reconstrução de CS completa com FFT acelerada em FPGA . . . 43

5 CONCLUSÃO . . . 44

REFERÊNCIAS . . . 45

APÊNDICE A PROPOSTA DE IMPLEMENTAÇÃO DO *CONJUGATE GRADIENT* EM FPGA . . . 47

1 Introdução

1.1 | Problemas de imageamento e a necessidade de formação de imagens em tempo hábil para diferentes aplicações

O evidente progresso na tecnologia de imageamento tem proporcionado inúmeros avanços na captação de dados e informações. A produção de imagens com maior qualidade de contraste e de maior definição permite um melhor reconhecimento dos elementos contidos em um determinado espaço e uma melhor classificação desses elementos. Como exemplo, o aumento da qualidade de imagens médicas tem possibilitado aos profissionais da saúde diagnosticarem mais rápido doenças (WATTJES; STEENWIJK; STANGEL, 2015), dessa forma, acelerando o início do tratamento para os pacientes. Inclusive, ainda na área de imageamento médico, captar imagens com maior definição colabora com a acurácia dos diagnósticos auxiliados por inteligência artificial aprimorando a exatidão dos métodos de classificadores de aprendizagem de máquina (FILHO; SILVA; MIOSSO, 2022).

Uma outra área que pode se beneficiar com aumento da qualidade na aquisição de imagens é a área de imageamento por satélite. Com a ajuda de classificadores de aprendizagem de máquina, o imageamento por satélite tem auxiliado as autoridades de segurança no combate à inúmeras atividades ilegais e riscos ambientais. Por exemplo, na identificação de mineração ilegal, no reconhecimento de extração ilegal de madeira, no combate ao tráfico de armas e drogas, na indicação de princípios de incêndio e de áreas com risco de incêndios, entre outros (KATO et al., 2016) (SHEVCHUK, 2018). De forma que a qualidade das imagens adquiridas é de importância fundamental para uma avaliação e classificação fiel dos fatos em análise.

Via de regra, a qualidade da aquisição de uma imagem está intrinsecamente ligada ao tempo gasto com a captura dessa imagem. Logicamente, quanto mais medidas reais obtidas, maior a qualidade da figura capturada. Entretanto, em determinados casos, principalmente nos casos de ressonância magnética para diagnóstico, tomografia computadorizada para diagnóstico, tomografia para detecção de armas ou bombas em locais

fechados e no âmbito de imageamento por satélite para prevenção de atividades ilegais e riscos ambientais, o tempo de captura das imagens torna-se demasiadamente crítico. (SHEVCHUK, 2018)

É um dos problemas recorrentes na área de imageamento, a necessidade de formação de imagens em tempo hábil. De maneira que, a demora na aquisição das imagens pode inviabilizar determinadas aplicações. Por exemplo, no contexto de reconhecimento de atividades ilícitas e acidentes ambientais por imageamento de satélite, apesar da alta aplicabilidade, no caso da reconstrução de imagens hiperespectrais, demanda-se um alto volume de aquisição de dados, sobretudo quando se buscam imagens com um grande nível de detalhamento (resolução espacial). Para obter resultados nesse grau de qualidade, exige-se um elevado custo computacional, aumentando o tempo relativo para a classificação de dados, e também uma banda larga de transmissão, para transportar as medidas adquiridas a um centro com grande capacidade de processamento. Determinadas atividades ilegais, principalmente quando estão começando, antes que atinjam uma escala de grande prejuízo, necessitam de um grande nível de resolução espacial. Isto normalmente exige uma quantidade grande de medidas, que, por sua vez, demandam recursos de aquisição e reconstrução. Além disso, a complicação na demora de aquisição de dados em alta densidade é incrementada, principalmente, ao considerar a complexidade envolvida em mapear um determinado local percorrendo uma órbita para acompanhar o movimento de rotação da terra. Por isso, a agilidade na captura e classificação das imagens é de grande importância, de forma a permitir uma resposta eficaz das autoridades responsáveis.

Um outro exemplo, agora no contexto das máquinas de ressonância magnética (MRI), um simples exame de imagem sagital do encéfalo chega a custar alguns milhares de dólares e durar cerca de algumas horas. O que não é apenas caro e desconfortável, mas também é crítico nos casos de pacientes que não podem ficar muito tempo inseridos no equipamento. Esse exemplo se expande para o caso da tomografia computadorizada para diagnóstico que, incrementado ao desconforto, está a incidência de raios ionizantes que, em grandes doses, pode ser cancerígeno para o paciente (GARCIA; FRANCO; MIOSSO, 2022). Portanto, de forma geral, a redução do tempo de captura nesses exames de imagem é de extrema relevância.

1.2 | Técnica para formação de imagens a partir de medidas limitadas

1.2.1 | Breve Introdução ao *Compressive Sensing* (CS)

Uma técnica consolidada capaz de mitigar o tempo envolvido na aquisição de dados é o *Compressive Sensing* (CS). Como o nome sugere, o CS é uma técnica de processamento capaz de reconstruir sinais com uma taxa de amostragem *sub-Nyquist*, ou a partir de medidas lineares limitadas. O CS se aproveita da existência de domínios transformados em que os sinais a serem reconstruídos são esparsos (CANDÈS; WAKIN, 2008). Basicamente, as técnicas propostas tentam resolver sistemas lineares indeterminados com a minimização da solução mais esparsa possível. Assim, desde que as condições de esparsidade e incoerência sejam respeitadas, é possível reconstruir sinais com uma alta taxa de confiabilidade utilizando um número reduzido de medidas.

Para tanto, considera-se um sistema subdeterminado que pode ser representado por $b = Mx$, onde b é o vetor de medidas, M é a matriz que modela o processo de medição (denominada matriz de medidas) e x é o sinal a ser reconstruído. Sabe-se, da álgebra linear, que um sistema subdeterminado ou não possui solução, ou possui infinitas soluções. Entretanto, impondo condições de minimização ao sistema é possível encontrar soluções dentro desse espaço de condições.

A primeira proposta teórica de CS foi com o método de reconstrução por ℓ_0. Este método equivale a testar recursivamente, um a um, os valores mais esparsos de $\hat{x}$ (sinal a ser reconstruído no domínio transformado esparsificante) que solucionam as condições de igualdade no sistema linear para reconstruir o sinal real. Dessa forma, a ℓ_0 é capaz de solucionar o sistema com o limite teórico de medidas, no entanto o tempo gasto nesse método de recursividade cresce fatorialmente com a resolução do sinal. Um outro método capaz de reduzir o tempo computacional gasto na reconstrução da imagem é a minimização por ℓ_p. No caso da ℓ_p, as infinitas soluções do sistema indeterminado são reduzidas a uma curva de conjunto de valores com dimensão igual ao número de medidas. Sendo que, para $p = 2$, esta minimização possui uma solução fechada (método de mínimos quadrados) dada por $\hat{x} = A^H(AA^H)^{-1}b$. Onde A^H é o hermitiano da matriz de reconstrução. Estes conceitos ficarão mais claros ao longo do desenvolvimento desse trabalho.

1.2.2 | A complexidade computacional da reconstrução em CS

A solução de sistemas lineares indeterminados não é algo simples. Normalmente, é necessário um tempo comparativamente alto e recursos de processamento mais elevados em relação à aquisição de dados em alta densidade. Esta exigência computacional pode inviabilizar determinados tipos de aplicações. Por exemplo, o método de reconstrução por ℓ_0 é viável apenas para reconstruções de pequenos sinais. Pelo fato da recursividade do método crescer fatorialmente, uma reconstrução simples de uma imagem de tomografia computadorizada pode demorar alguns milhares de anos para ser finalizada.

Para o caso das técnicas de ℓ_p, normalmente utilizadas com $p = 1$, é possível reduzir o tempo de processamento envolvido durante a reconstrução por CS para uma complexidade polinomial. No entanto, para isso, é necessário comprometer o nível de confiabilidade da reconstrução ou aumentar o número de amostras captadas. Normalmente demanda-se um número 3 a 6 vezes maior de medidas amostradas, em comparação ao limite teórico, para uma reconstrução fiel. E mesmo para esses métodos de resolução, ainda é preciso considerar a limitação de memória computacional na solução dos sistemas lineares utilizando o método direto, o qual a matriz de reconstrução deve ficar completamente armazenada na memória. Sendo necessário, para aplicações de imageamento, utilizar métodos indiretos de solução de sistemas lineares, como é o caso do *Conjugate Gradient* (CG). Nessa situação, ainda sim, a reconstrução de imagens reais pode durar alguns minutos, dependendo do tamanho da imagem.

Na conjuntura dos exemplos apresentados, principalmente no imageamento de satélites, os dados são adquiridos com uma alta frequência, ou seja, é necessário que esse conjunto de dados seja processado, tanto para reconstrução como para classificação, antes da aquisição de novos dados, para que se mantenha o fluxo de processamento. Portanto, deseja-se obter resultados com um alto nível de confiabilidade e com a maior rapidez possível. Assim deseja-se que, não apenas a captura, mas também o processamento dos dados sejam o mais rápido possível.

1.3 | Proposta Científica

Diante dos problemas levantados, este trabalho propõe avaliar quais etapas de reconstrução em CS são mais críticas do ponto de vista do tempo computacional e quais etapas são passíveis de implementação em arquiteturas paralelas. Esta análise será realizada, inicialmente, utilizando um algoritmo de minimização em ℓ_p implementado em *software* para reconstrução em CS. Este algoritmo consiste na reponderação de mínimos quadrados realizando reconstruções de ℓ_2, que é, na prática, a resolução de vários sistemas lineares consecutivos (MIOSSO et al., 2009). Por isso, o estudo será realizado para ambos os métodos de solução de sistemas lineares, a saber, direto e indireto. Esses métodos são fundamentalmente diferentes; assim, será examinada qual é a etapa mais crítica de cada método e qual método de solução possui um maior ganho com a implementação com arquiteturas paralelas.

Além disso, o trabalho propõe avaliar implementações em FPGA que abordam as etapas mais críticas do tempo computacional de cada método. Também serão desenvolvidas implementações dedicadas para paralelizar a resolução dos sistemas lineares. A princípio, essas implementações utilizaram sinais de pequenas dimensões (para viabilizar a implementação) e, posteriormente, será realizada a validação da implementação, comparando

o tempo de processamento gasto em *software* com o processamento em *hardware*. Finalmente, serão avaliados ainda os ganhos de tempo de processamento proporcionados pelas implementações em FPGA para CS, com base, tanto em simulações, como em implementação com uma arquitetura que combina o uso de processador ARM para as operações sequenciais com FPGAs para as operações paralelizadas.

1.4 | Objetivos

1.4.1 | Objetivo Geral

O objetivo dessa pesquisa é avaliar a viabilidade e os ganhos de processamento para implementações de reconstrução por *Compressive Sensing* em FPGA. Essa análise poderá contribuir para a viabilização de aplicações que necessitem de um alto nível de confiabilidade e rapidez durante a captura e processamento dos dados. Além disso, irão gerar conhecimento de novos métodos de implementações possíveis dentro da área do CS.

1.4.2 | Objetivos Específicos

Os objetivos específicos estão descritos a seguir.

1. Criar um entendimento dos método de reconstrução de sinais em CS, bem como o estado da arte desses métodos.
2. Desenvolver, junto ao grupo de pesquisa, algoritmos dos métodos de minimização de CS em Python para solução e validação de reconstruções possíveis.
3. Avaliar as diferenças computacionais entre os métodos direto e indireto de resolução de sistemas lineares.
4. Criar um entendimento no método de pré-filtragem para reconstruções em CS.
5. Avaliar quais são os gargalos de processamento do método direto e indireto da reconstrução de sinais em CS.
6. Propor soluções em *software* e em *hardware* de como acelerar os algoritmos de CS.
7. Implementar em um SoC um algoritmo funcional de CS programado em Python e cujas etapas estão aceleradas em implementações de FPGA.
8. Avaliar e comparar os tempos gastos para implementações em *hardware* e em *software*.
9. Quantificar os ganhos de processamento das implementações aceleradas em FPGA.

10. Propor soluções e próximos passos para a pesquisa.

1.5 | Estrutura do Texto

O restante do texto está organizado da seguinte forma, o Capítulo 2 apresenta uma fundamentação teórica sobre as principais técnicas e o estado da arte dessas técnicas utilizadas para a reconstrução de imagens com base em medidas limitadas via técnicas de Compressive Sensing. Nesse capítulo será realizada uma introdução ao CS e, posteriormente, será apresentada a minimização de LP, que é uma das abordagens que permite a reconstrução com a melhor qualidade a partir de uma quantidade reduzida de medidas. (MIOSSO et al., 2009) Sendo que, para o entendimento da LP, é necessário desenvolver uma apresentação formal da minimização L2 clássica. Está minimização não chega a ser uma solução para CS, entretanto é um passo para o entendimento da técnica de IRLS (*Iteratively Reweighted Least Squares*).

O capítulo apresenta ainda uma descrição do problema de solução indireta de sistema lineares. Uma vez que, no problema de imageamento, os sistemas lineares envolvidos possuem um número grande de dimensões, de maneira que o tamanho da memória RAM comumente disponível nos sistema computacionais modernos não é suficiente para armazenar as matrizes de operação. Assim, a solução indireta vem como uma forma de abordar uma solução numérica desses sistemas lineares sem a necessidade de armazenar a cada momento a matriz do sistema. Um método muito conhecido para resolução de sistemas lineares de forma indireta é o Conjugate Gradient, que também será apresentado nesse capítulo. Finalmente é apresentado o método de pré-filtragem para reconstrução de imagens com CS. Nesse método, a transformada esparsificante T é equivalente a identidade, já que as medidas são filtradas com filtros passa-altas no domínio da frequência antes de serem reconstruídas, por conseguinte, as medidas ficam esparsas. E depois é realizada uma composição da imagem a partir das reconstruções em alta frequência e com as medidas em baixa-frequência.

O capítulo 3 irá abordar os métodos utilizados nesse trabalho. Inicialmente será apresentado o desenvolvimento em Python dos algoritmos de CS e de funções necessárias para os experimentos. Depois será feita uma explicação dos experimentos e o objetivo de cada experimento realizado.

O capítulo 4 apresenta dos resultados preliminares obtidos com os experimentos realizados. Primeiro é realizada uma comparação entre o método direto e o método inverso de solução da minimização de ℓ_p avaliando o tempo gasto com as operações e a qualidade dos sinais reconstruídos. Depois será apresentado o resultado de outros dois experimentos, ambos para avaliação de como a quantidade de medidas disponíveis afeta no tempo e na

qualidade da reconstrução. Um experimento será uma uma imagem esparsa e o outro com uma imagem real, porém utilizando a técnica de pré-filtragem para CS. E ainda serão apresentados os resultados obtidos da implementação de partes do CS em FPGA.

Finalmente, o Capítulo 5 irá apresentar as conclusões que podem ser retiradas desse tempo de desenvolvimento do trabalho e quais são os próximos passos para pesquisas futuras.

2 Embasamento Teórico do *Compressive Sensing*

A maior parte dos sinais adquiridos é amostrada a partir de sinais analógicos a uma taxa de amostragem fixa f_s. Na maioria dos casos reais, a taxa de amostragem utilizada na captação dos sinais é igual ou superior a $2f_m$, onde f_m é a maior frequência do sinal. Isso se dá pelo fato do entendimento de amostragem do Teorema de Nyquist—Shannon (SHANNON, 1948). Esse Teorema, também conhecido como Teorema da amostragem, prova que, caso a frequência de amostragem seja o dobro da maior frequência do sinal ou maior, a recuperação de um sinal analógico amostrado, a partir apenas das amostras, pode ser feita teoricamente sem erro. A não observância dessa condição pode gerar um fenômeno conhecido com *aliasing*, em que há uma distorção do sinal amostrado podendo impossibilitar a identificação correta do sinal real.

Após a aquisição correta dos sinais, tipicamente, antes de serem armazenados ou transmitidos, esses sinais são comprimidos. Ou seja, a maior parte dos coeficientes transformados captados são desconsiderados durante o processo de envio de dados. Isso se dá devido a uma estratégia bastante utilizada que é a compressão por transformada. Nesses casos, apesar do sinal estar representado inicialmente por N amostras, existe uma transformada conhecida do vetor x capaz de concentrar as informações em menos coeficientes. Para que isso seja possível, deve existir uma matriz de transformação $T_{[N\times N]}$ tal que

$$\hat{x}_{[N\times 1]} = T_{[N\times N]} \cdot x_{[N\times 1]}, \tag{2.1}$$

onde $\hat{x}$ é um vetor transformado que preserva as informações de x, ou seja, a matriz T é inversível de forma que existe T^{-1} tal que $T \cdot T^{-1}$ é a identidade.

Para efeito de compressão por transformada, é necessário que o vetor $\hat{x}$ seja esparso, ou seja, a maior parte dos coeficientes de $\hat{x}$ é nula ou, na prática, próxima de zero. Dessa forma, é possível reconstruir um sinal $x_{[N\times 1]}$, com uma alta taxa de confiabilidade, conhecendo apenas os valores não-nulos de $\hat{x}$, as posições corretas de cada coeficiente não-nulo e a transformada inversa T^{-1} apropriada.

Considerando que, de fato, é possível representar informações de N amostras de um sinal em n_c coeficientes, com $n_c << N$, surge uma pergunta: — Não há um desperdício na

aquisição? Será que não é possível, principalmente para aplicações em que o sensoriamento é custoso, reduzir a quantidade de amostras a uma faixa *sub-Nyquist* de forma a reconstruir um sinal perfeitamente com $n_c << N$ amostras? Nesse sentido é proposta a área de estudo do *Compressive Sensing*.

2.1 | Conceitos formais de *Compressive Sensing*

O Compressive Sensing objetiva medir poucos coeficientes, porém suficientes, que contenham, teoricamente, toda a informação do sinal amostrado. No entanto, de maneira direta, não há um método geral para amostrar exatamente os n_c coeficientes do sinal transformado comprimido desejado $\hat{x}_r$. Tendo isso em vista, a área de CS propôs um conjunto de teorias e métodos capazes de calcular, teoricamente sem erro, um sinal x de tamanho N a partir de l medidas lineares de x, com $n_c < l << N$. Para tanto, semelhantemente à Equação 2.1, utiliza-se o fato de que o sinal possui uma representação esparsa $\hat{x}$ em um domínio conhecido, porém, nesse caso, $x_{[N\times 1]}$ não é conhecido. A informação coletada do sinal, é portanto representada pelo vetor de medidas

$$b_{[l\times 1]} = M_{[l\times N]} \cdot x_{[N\times 1]}, \tag{2.2}$$

em que x é o sinal a ser reconstruído, o vetor b é denominado vetor de medidas conhecido e contém l componentes e a matriz M é a matriz de medidas proveniente e intrínseca ao equipamento ou sensor utilizado na amostragem. A Equação 2.2, por si só, é um sistema subdeterminado, ou seja, ou ela admite um número infinito soluções, ou não possui solução. (SHILOV, 2012) Nesse caso específico, é sabido que esse sistema possui solução, uma vez que M é uma matriz conhecida e x é um sinal real que pode ser medido. Logo, o objetivo é calcular a solução mais esparsa em um domínio transformado que satisfaça essa equação.

Contudo, não é possível medir diretamente os n_c componentes do sinal. Por isso, a ideia sugerida é de encontrar os η coeficientes não-nulos na representação transformada do sinal. Caso seja conhecido o vetor $\hat{x}_r$ construído apenas com os componentes η não-nulos do sinal transformado, é viável construir um sistema sobredeterminado cuja solução é possível, desde que as condições de esparsidade e incoerência sejam respeitadas, tal que $P \cdot b = \hat{x}_r$, e, consequentemente, é possível encontrar o vetor x desejado. Assim, idealmente, busca-se a solução mais esparsa minimizando o vetor x em um domínio transformado esparcificante. A ideia geral pode ser representada pelo problema de otimização

$$\hat{x}_r = \text{argmin}_{\hat{x}} \|\hat{x}\|_0. \tag{2.3}$$

A Equação 2.3 representa, na prática, a aquisição dos valores não-nulos do vetor transformado. Entretanto, o resultado dessa minimização, na forma como foi posta, é apenas

um vetor preenchido de zeros, o que não é o desejado. Esta função necessita de uma condicional do vetor transformado. Partindo da Equação 2.2, considerando que $T^{-1}\hat{x} = x$, sem perdas de informações, é obtido o sistema

$$b = MT^{-1}\hat{x}. \tag{2.4}$$

Dessa maneira, deseja-se encontrar o vetor $\hat{x}_r$ com o mínimo de componentes não-nulas que satisfaça a condição da Equação 2.4. Assim, a solução ideal é dada por,

$$\begin{aligned} \hat{x}_r &= \text{argmin}_{\hat{x}_r} \|\hat{x}\|_0 \\ \text{tal que, } & MT^{-1}\hat{x}_r = b. \end{aligned} \tag{2.5}$$

Com o propósito de simplificar a notação, doravante, a expressão MT^{-1} será substituída pela letra A na representação dessas matrizes. Nesse caso, é construído um sistema reduzido sobredeterminado apresentado na equação

$$A_{r\ [l\times\eta]} \cdot \hat{x}_{r\ [\eta\times 1]} = b_{[l\times 1]}, \tag{2.6}$$

em que A_r é uma submatriz reduzida de A formada por η colunas, $\hat{x}_r$ é o vetor formado apenas pelos η elementos não-nulos de $\hat{x}_r$ e b é o vetor de medidas obtido. Como η é menor que l, esse sistema é de fato subdeterminado. Em geral, esse sistema não admite solução, teoricamente, apenas uma combinação de posições irá zerar $A_r\hat{x}_r = b$ como solução. Para verificar que o sistema admite solução, é possível encontrar $\hat{x}_r$, tal que, a norma da ℓ_2 do sistema é mínima,

$$\hat{x}_r = \text{argmin}_{\hat{x}_r} \|A_r\hat{x}_r - b\|_2. \tag{2.7}$$

A Equação 2.7 possui solução fechada (BOYD; BOYD; VANDENBERGHE, 2004) igual a

$$\hat{x}_r = \left(A_r^{\mathcal{H}} A_r\right)^{-1} A_r^{\mathcal{H}} b, \tag{2.8}$$

em que $A_r^{\mathcal{H}}$ é o hermitiano da matriz reduzida de A. Para chegar nessa equação fechada utiliza-se um artifício popular com os multiplicadores de Lagrange (NOCEDAL; WRIGHT, 1999). Logo, o sistema da Equação 2.7 tem solução em forma fechada, portanto o sistema pode ser resolvido sempre que for compatível. Com base nessa solução em forma fechada, é possível resolver o sistema maior da Equação 2.5, de minimização de ℓ_0, trabalhando com todas as combinações na forma reduzida. Assim, basta encontrar os componentes do vetor $\hat{x}_r$, as posições de cada componente não-nulo, criar o vetor $\hat{x}_{[N\times 1]}$

com os componentes nas posições apropriadas e, finalmente, aplicar a transformada inversa T^{-1} para obter o sinal $x_{[N\times 1]}$ de alta densidade.

Esta solução, apesar de ser conceitualmente simples, na prática, possui uma única forma de resolvê-la diretamente, no caso geral. De forma que é necessário testar todas as combinações de posições dos elementos não-nulos e, para cada combinação, avaliar se o sistema reduzido respeita a condição imposta da solução do sistema. A única forma conhecida do algoritmo dessa solução possui uma complexidade combinacional, ou seja, o tempo de processamento necessário cresce fatorialmente com o tamanho do sinal a ser reconstruído. Como um exemplo, um cálculo simples mostra que uma reconstrução de uma imagem de tamanho 512x512, que necessite de aproximadamente 1 milissegundo de processamento em cada combinação — um tempo comparativamente baixo, pode levar um tempo de processamento da ordem de 10^8 anos — esse tempo é superior à idade do universo.

Tendo isso em mente, apesar da solução apresentada ser a resolução mais simples e direta do problema, outras formas de minimização do problema para reconstrução de sinais em CS foram propostas.

2.2 | Minimização de ℓ_2

Como há infinitas soluções para o sistema $A\hat{x} = b$, uma outra maneira de chegar a uma dessas soluções pode ser sugerida. A solução mais simples para o problema, do ponto de vista computacional, é utilizar a minimização por norma ℓ_2. Como dito anteriormente, essa norma tem uma solução fechada que pode ser processada em poucos milissegundos por um computador normal, até mesmo para imagens de tamanhos típicos. O problema da Equação 2.5 pode ser reescrito na forma da minimização de ℓ_2

$$\begin{aligned} \hat{x} &= \text{argmin}_{\hat{x}} \|\hat{x}\|_2^2 \\ &\text{tal que, } A\hat{x} = b, \end{aligned} \tag{2.9}$$

em que a norma da ℓ_2 equivale a

$$\|\hat{x}\|_2 = \sqrt{|\hat{x}_1|^2 + |\hat{x}_2|^2 + |\hat{x}_3|^2 + \cdots + |\hat{x}_N|^2}. \tag{2.10}$$

A utilização de multiplicadores de Lagrange conduz a seguinte solução em forma fechada para o problema da Equação 2.9 (NOCEDAL; WRIGHT, 1999).

$$\hat{x} = A^{\mathcal{H}} \left(AA^{\mathcal{H}}\right)^{-1} b \tag{2.11}$$

2.3 | Minimização de ℓ_p com IRLS

De forma semelhante à minimização de ℓ_2, que tem forma fechada de resolução, é possível propor outros maneiras de minimização para a solução. Comprometendo alguns elementos como processamento ou convexidade, Por exemplo, é possível reconstruir sinais com transformada esparsa com a minimização da ℓ_1, que possui uma solução numérica para o problema (DONOHO, 2006). A principal minimização utilizada nesse trabalho é a métrica da norma ℓ_p. No caso da minimização da ℓ_p a Equação 2.5 pode ser reescrita na forma,

$$\begin{aligned} \hat{x} &= \text{argmin}_{\hat{x}} \|\hat{x}\|_p^p \\ &\text{tal que, } A\hat{x} = b, \end{aligned} \tag{2.12}$$

onde, $0 < p < 1$. De modo que,

$$\|\hat{x}\|_p = \sqrt[p]{|\hat{x}_1|^p + |\hat{x}_2|^p + |\hat{x}_3|^p + \cdots + |\hat{x}_N|^p}. \tag{2.13}$$

Como indica a literatura, este problema converge para uma solução utilizando o método de mínimos quadrados reponderados.

$$\begin{aligned} &\min_{\hat{x}} \frac{1}{2} \sum_{k=1}^{N} w_k^{p-2} \hat{x}_k^2 \\ &\text{tal que, } A\hat{x} = b, \end{aligned} \tag{2.14}$$

onde w_k é o parâmetro de reponderação (MIOSSO, 2010). Essa aproximação corresponde a uma minimização quadrática que possui forma fechada. De maneira que é possível reconstruir sinais verdadeiramente esparsos que convergem para a minimização da ℓ_p a partir de N iterações da minimização da ℓ_2. Portanto, a Equação 2.11 tem uma solução mais esparsa se reescrita na seguinte forma,

$$\hat{x}^{(m)} = Q^{(m)} A^{\mathcal{H}} \left(A Q^{(m)} A^{\mathcal{H}} \right)^{-1} b \tag{2.15}$$

Em que Q é a matriz diagonal formada pelos elementos,

$$q_k = \left| \hat{x}_k^{(m-1)} \right|^{2-p} \tag{2.16}$$

Resolvendo a Equação 2.14 para $m = 1, 2, \cdots$, até a convergência.

2.4 | Solução Indireta para sistemas lineares em algorítimo de CS

O algoritmo apresentado na Equação 2.14 é um processo iterativo que envolve a resolução de sistemas lineares. Cada sistema linear possui uma matriz de descrição que pode possuir um tamanha superior à memória típica de sistemas computacionais comuns, mesmo no problema de reconstrução de imagens relativamente pequenas. Por exemplo, para a reconstrução de uma imagem de tamanho 512×512, que não é um imagem particularmente grande para os padrões atuais, utilizando o método direto, se faz necessária uma memória RAM de algumas centenas de gigabytes. O que inviabiliza a implementação direta da solução desse sistema.

Para tratar esse problema, existem os chamados métodos indiretos. Eles permitem a resolução dos sistemas lineares sem a necessidade de armazenar explicitamente a matriz na memória RAM. Ao contrário, é implementado uma função que realiza um produto matriz vezes vetor, e a partir desse produto um algoritmo iterativo resolve esse sistema linear em vários passos. O sistema iterativo dispensa a matriz armazenada em memoria RAM, substituindo os cálculos com a matriz por sequências de multiplicações matriz—vetor. Para este estudo, optou-se por utilizar o método de solução indireta conhecido por *Conjugate Gradient*.

2.5 | Método de *Conjugate Gradient* para Soluções Indiretas

O *Conjugate Gradient* é um método iterativo para resolução de sistemas lineares onde não é possível, ou muito custoso, armazenar na memória as matrizes para resolução do sistema. Este método iterativo resolver sistemas lineares tal que, $My = b$, onde b é um vetor conhecido e M é uma função, que quando chamada, é capaz de retornar um $M(y)$, para qualquer y genérico.

De forma resumida, o método consiste na geração de um vetor de resíduos tal que, $r = M(y) - b$. Deseja-se minimizar esse resíduo até que, dentro de uma tolerância, ele seja próximo ao vetor nulo. A resposta ao sistema será dada por,

$$y_{n+1} = y_n - \frac{s_n}{p_n^{\mathcal{H}} \times M(p_n)} \tag{2.17}$$

tal que,

$$p_{n+1} = -r_n + \frac{s_{n+1}}{s_n} \times p_n \tag{2.18}$$

onde,

$$s_n = r_n^{\mathcal{H}} \times r_n \tag{2.19}$$

O algoritmo ira convergir mesmo que o valor inicial de y seja um vetor nulo. Com algumas iterações já é possível obter resoluções satisfatórias para o sistema linear, sem a necessidade de armazenamento de toda a matriz. O grande desafio está em ser capaz de descrever corretamente uma função que possa representar corretamente a matriz em questão. O algoritmo que foi implementado em Python é demostrado no quadro a seguir. A descrição do algoritmo foi adaptada do livro *Numerical Optimization* do Nocedal (NOCEDAL; WRIGHT, 1999).

```
    def mycg(M, b, maxiter=0, tol=1e-6):
        # Solves the linear equation My = b, where y ...
        # is the unknown an M is n x n matrix.
        N = len(b)
        if maxiter == 0:
            maxiter = 2 * N
        y = np.zeros((N, )) # Or np.random.randn(N, 1)
        # Start Conjugate Gradient Method
        r = M(y) - b
        p = -r
        rsold = hermitian(r) @ r
        for i in range(maxiter):
            Sp = M(p)
            alpha = rsold / (hermitian(p) @ Sp)
            y = y + alpha * p
            r = r + alpha * Sp
            rsnew = hermitian(r) @ r
            if (np.sqrt(rsnew) < tol):
                  break
            p = - r + (rsnew / rsold) * p
            rsold = rsnew
        return y
```

Listing 2.1 – Algoritmo para o *Conjugate Gradient* implementado em Python.

2.6 | Método de Pré-Filtragem em CS

Por fim, foi realizado ainda uma avaliação do método de pré-filtragem para reconstrução em *Compressive Sensing* (MIOSSO; BORRIES; PIERLUISSI, 2009). De maneira

geral, o método proposto consiste na utilização de filtros esparsificantes antes de realizar a reconstrução de um sinal. De forma que, utilizando N filtros e, consequentemente, N reconstruções, seja possível realizar a composição desses vetores reconstruídos para obter o sinal de alta densidade ao final.

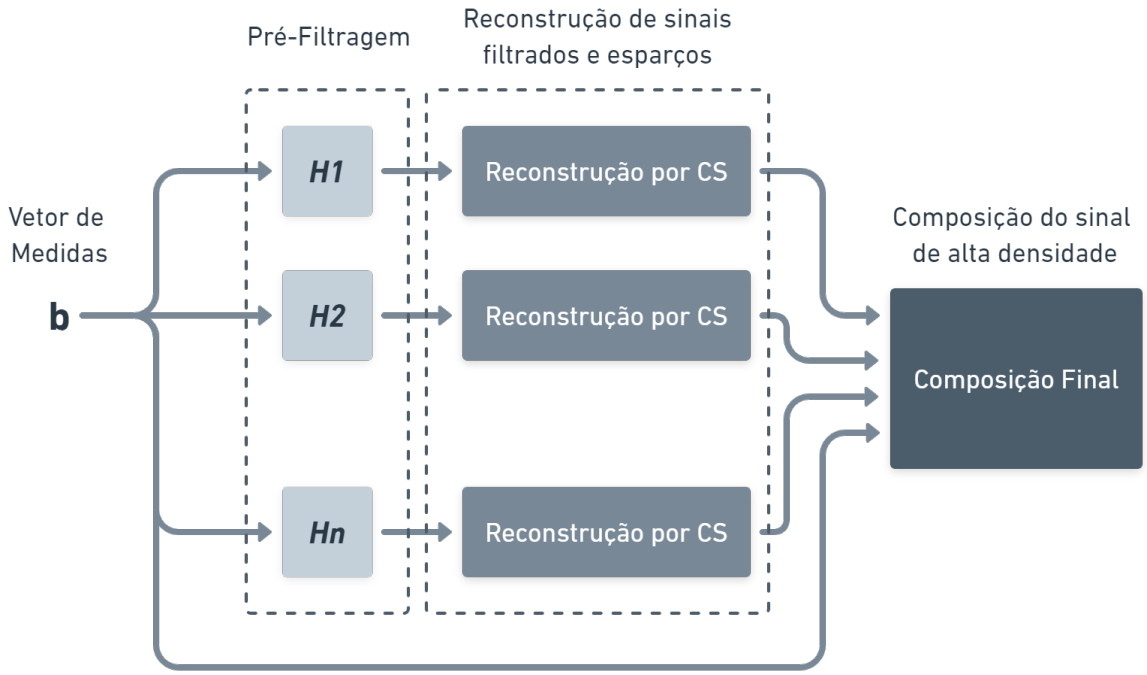


Figura 1 – Diagrama de resumo para reconstrução de sinais por CS utilizando pré-filtragem.

Por exemplo, considere o conjunto dos filtros de Haar bidimensionais. Famosos na área de processamento de sinais para detecção de bordas, diagonais, verticais e horizontais. Os filtros são compostos pelas matrizes de diferenças,

$$h_1 = \begin{vmatrix} 1 & -1 \\ -1 & 1 \end{vmatrix}, \; h_2 = \begin{vmatrix} 1 & -1 \\ 1 & -1 \end{vmatrix}, \; h_3 = \begin{vmatrix} 1 & 1 \\ -1 & -1 \end{vmatrix}. \tag{2.20}$$

Neste caso, o conjunto de filtros de Haar é um filtro passa-altas que irá realçar as componentes de alta frequência do sinal em todas as direções. Dessa forma, é possível a reconstrução de uma versão filtrada de um sinal $\hat{x}_n$ a partir de um número de medidas limitadas b_n. Assim, na reconstrução, utilizando as componentes de baixa frequência medidas, é possível reconstruir uma versão satisfatória do $\hat{x}$ desejado.

3 Materiais e Métodos

3.1 | Implementação de algoritmos de CS em um computador sequencial pessoal

Apresentadas as principais técnicas para reconstrução de sinais com medidas limitadas por Compressive Sensing, foram desenvolvidos algoritmos computacionais, baseados na teoria, capazes de reconstruir sinais e imagens a partir de uma quantidade reduzida de amostras. Foram desenvolvidos programas para minimização de ℓ_0, ℓ_2 e ℓ_p utilizando a linguagem de programação Python. Vale ressaltar também que boa parte dos algoritmos de minimização já estavam implementados em versões para MatLab. E que a tradução dos programas para Python é fruto de um trabalho conjunto do grupo de pesquisa do qual esse trabalho faz parte. A tradução desses algoritmos para Python é de extrema relevância para esse trabalho, uma vez que, não apenas auxilia no aprendizado e fixação dos conceitos teóricos, mas também, possibilita a implementação do programa em um processador ARM de tecnologia SoC para aceleração de etapas em FPGA, esse tópico será melhor abordado na Seção 3.3. Os programas são executados na versão do Python3 e utilizam as bibliotecas disponíveis do pacote Anaconda para Python.

3.1.1 | Programas necessários para validação dos algoritmos de minimização em Python

3.1.1.1 | Gerador de sinais 1D com com representação transformada esparsa

Assim, o primeiro programa desenvolvido foi um gerador de sinais aleatórios com representação transformada esparsa. A função desenvolvida recebe um tamanho de matriz esparsificante ou a própria matriz e recebe um valor para η componentes não-nulos de $\hat{x}_r$. A função retorna a matriz esparcificante T, o vetor x, bem como sua versão no domínio transformado $\hat{x}$ e as posições de cada componente não-nulo. Essa função é importante para validação de cada algoritmo construído. Este programa será utilizados em todos os experimentos realizados com sinais estocásticos de domínio 1D. No caso desses sinais,

ainda é gerada uma matriz aleatória M, para que sejam realizadas amostras l de Mx para a geração do vetor b e da matriz A. Tendo assim, todas os vetores e matrizes necessárias para a validação do algoritmo.

3.1.1.2 | Gerador da matriz de medidas para imagens esparsas e pré-filtradas

No caso dos experimentos realizados com imagens simples já esparsas e com imagens pré-filtradas, a transformada esparsificante T é igual a identidade, de forma que $A = M$ e $x = \hat{x}$. Assim, basta apenas construir a matriz M ou, no caso do método indireto, a função $M(x)$. Para simular a matriz de medidas de uma MRI, foi desenvolvida uma função que gera amostras da FFT bidimensional do sinal em trajetórias angulares. Logo a função de $A^{\mathcal{H}}(x)$ é definida como a versão inversa da FFT bidimensional.

3.1.2 | Desenvolvimento dos algoritmos para reconstrução em CS

Foi desenvolvida também, em Python, uma versão própria do Conjugate Gradient de cálculo recursivo para resolução dos sistemas lineares pelo método indireto, seguindo o padrão disponível na bibliografia (NOCEDAL; WRIGHT, 1999).

Assim, finalmente foi implementada a minimização de ℓ_p para realização dos experimentos. A função está implementada com técnicas de IRLS (MIOSSO et al., 2009). Para o método direto, é utilizada a função de resolução de sistemas lineares `linalg.solve()` disponível na biblioteca `Numpy`. E no caso do método indireto, a função de Conjungate Grandient criada `my_cg()` está sendo utilizada.

Para o caso da reconstrução da imagem pré-filtrada, estão sendo utilizados os filtros clássicos de Haar. São um conjunto de 3 filtros passa-alta. Para a pré-filtragem, cada filtro irá destacar as bordas em 3 direções: horizontal, vertical e diagonal. As medidas do sinal são multiplicadas no domínio da frequência, o que equivale à convolução no domínio do tempo (VETTERLI; KOVAČEVIĆ; GOYAL, 2014). Assim, cada parcela do vetor de medidas b filtrado, por conseguinte esparso, é reconstruído utilizando a minimização de ℓ_p. Depois é realizada a composição da figura, com as componentes de alta frequência de cada espectro reconstruídas e com as componentes de baixa frequência no espectro medidas.

3.2 | Experimentos Realizados

3.2.1 | Comparações entre o método direto e indireto para processos estocásticos em domínio 1D

Para esse experimento, foi utilizada a função geradora de sinais 1D com representação transformada esparsa. Nesse experimento foram avaliados os tempos de reconstrução de sinais estocásticos amostrados com l medidas para o método direto e indireto de solução da minimização de ℓ_p para CS. Foi também avaliado o tempo gasto especificamente com a resolução do sistema linear para ambos os métodos, e por fim, é indicado o MSE (Erro Quadrático Médio) para cada reconstrução.

Este experimento tem o objetivo de compreender como o aumento do tamanho do sinal a ser reconstruído afeta o tempo e a qualidade da reconstrução por minimização de ℓ_p em CS. Para tanto, foi gerado uma série de sinais aleatórios para serem reconstruídos. Para todas as reconstruções, foi utilizado 10% de amostras do sinal original, ou seja, $l = 0.1N$. Tanto para o método direto, como para o método indireto o sinal começou com um tamanho de $N = 100$, e recebeu incrementos de 250 amostras para cada nova reconstrução. Os testes de método direto e indireto foram rodados simultaneamente em um mesmo processador Intel Core i3.

3.2.2 | Reconstrução por CS de imagens já esparsas no domínio original

Para este experimento, foi utilizado uma imagem exemplo clássica para ser reconstruída. Foi utilizado uma versão esparsa do fantoma de Shepp-Logan, gerada pela função `phantom.phantom()` do pacote Anaconda. A imagem gerada tem o tamanho 512x512. Também foi utilizado o gerador de matriz de medidas para simular as medidas obtidas por MRI. Todas as reconstruções desse experimento utilizam o método indireto, pois é inviável reconstruir uma imagem desse tamanho utilizando o método direto, considerando um tamanho de 4Gb de memória RAM.

Neste experimento, o objetivo é entender como a quantidade de medidas do vetor b amostrado influencia no tempo e na qualidade da reconstrução de uma imagem naturalmente esparsa. Para tanto, a primeira reconstrução iniciou-se com 5 trajetórias angulares, equivalente a 0.97% de N e houve um acréscimo de 5 trajetórias angulares a cada nova reconstrução.

3.2.3 | Reconstrução de uma imagem real com Pré-Filtragem

Finalmente, esse experimento objetiva a reconstrução de uma imagem real utilizando técnicas de pré-filtragem para avaliação do tempo e da qualidade da reconstrução de

imagens por pré-filtragem. A imagem foi amostrada utilizando o gerador de matriz de medidas de MRI, e as medidas obtidas foram filtradas no domínio da frequência, para criação da composição da figura. Nesse caso foram realizadas 12 reconstruções, com um incremento de 25 novas trajetórias angulares no processo de medidas. De forma que a primeira reconstrução iniciou-se com 25 trajetórias angulares, equivalente a 9.48% de N e a última teve 300 trajetória angulares, cerca de 81.31% de N.

3.3 | Implementações em *Hardware* de etapas computacionais utilizadas em CS com pré-filtragem

Após os estudos de caso realizados, verificou-se que o maior tempo de processamento gasto, no caso do método indireto, é na resolução dos sistemas lineares. Dessa maneira, é possível utilizar a implementação em hardware para acelerar a resolução dos algoritmos de CS. Já foi verificado os ganhos das implementações das operações de multiplicação em paralelo em FPGAs no caso de resolução pelo método direto para imagens multiespectrais (NASCIMENTO; VéSTIAS, 2021). E também foram verificados avanços no cálculo do método de mínimos quadrados acelerados com implementações em *hardware* (DONG; LI; WANG, 2013). Esse trabalho focará na resolução do método indireto utilizando implementações da FFT para acelerar as reconstruções.

3.3.1 | Implementação da FFT em FPGA para acelerar parte da resolução da minimização de ℓ_p em SoC

Foi escolhida a placa de desenvolvimento Pynq-Z2. A placa é um projeto *open-source* e possui um SoC Zynq-7000 integrado da Xilinx. Esse é um CI (Circuito Integrado) que possui um processador ARM com dois núcleos de processamento Cortex-A9 MPCore^{TM}, e também, uma FPGA da família Artix-7 da Xilinx (XILINX, 2018). A placa possui também um ambiente no Jupyter Notebook para desenvolvimento de programas em Python. Assim, é possível aproveitar os benefícios da implementação em *hardware* e a facilidade da programação em Python no processador dentro do SoC.

Dessa forma, foi realizada a implementação da FFT em FPGA utilizando o módulo da Transformada de Fourier discreta implementada em HDL (*Hardware Description Language*) copilado disponibilizada pela Xilinx (IP) utilizado o programa Vivado. O programa é utilizado para gerar o arquivo de *bitstrem* (Formato de arquivo de descrição de hardware) que será executado na placa de desenvolvimento. Os algoritmos de CS descritos

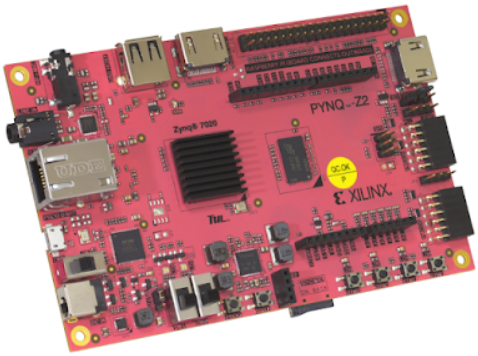


Figura 2 – Placa de Desenvolvimento Pynq-Z2, com SoC Zynq-7000 integrado. (Imagem da Internet)

foram copiados para dentro do ambiente Jupyter Notebook da Pynq-Z2, e foi utilizado o protocolo de comunicação AXI para troca de dados entre o processador e o bloco de IP da FFT implementado na FPGA para substituir o processamento da FFT de *software* para *hardware*.

Antes de, efetivamente, começar as reconstruções em CS com a FFT acelerada em FPGA, foi realizado um estudo para análise dos ganhos possíveis dessa abordagem, e também análise de áreas para acelerar o melhorar o algoritmo. Para tanto, inicialmente, foi realizada uma comparação entre o tempo gasto e o erro da FFT e da IFFT implementadas em FPGA, e também, dentro do processador ARM. Para a comparação, foram gerados sinais aleatórios de tamanho $N = 2^6$ até $N = 2^{14}$ analisando a média do resultado para 10 mil interações cada.

A fim de aproveitar o máximo da placa de desenvolvimento, foi realizada a implementação de uma memória para reconfiguração do módulo. De forma que com a mesma quantidade de silício, foi possível realizar a tanto a implementação da FFT, como da IFFT. Apesar desta técnica possibilitar a reconstrução de sinais com maiores tamanhos, houve um comprometimento com o tempo de processamento. Por isso também foi feita a análise do tempo gasto para a implementação com reconfiguração do módulo. O método de comparação foi o mesmo realizado do experimento anterior.

Por fim, utilizando a técnica da FFT e IFFT reconfigurável, foi possível implementar o algoritmo de CS para resolução do sistema com método indireto da métrica ℓ_p. Foi utilizada a transformada de Fourier como a matriz de medidas de um sinal esparso do domínio da frequência. Assim, foi utilizada a FPGA para acelerar para acelerar o processamento das transformações inversas e diretas durante a reconstrução. Foi realizada uma comparação entre o tempo e erro da reconstrução realizada do processador e da reconstrução acelerada em *hardware*.

4 Resultados Preliminares e Discussões

Esse trabalho aborda o uso implementações em *hardware* para acelerar e viabilizar a aplicação de técnicas de CS para imageamento com uso de arquiteturas paralelas. Sendo assim, um dos aspectos de investigação necessário é determinar quais etapas de uma reconstrução por CS são mais críticas em termos do tempo gasto de uma reconstrução em computação serial. De forma a compreender qual etapa do processamento irá usufruir de maior ganho com paralelização em *hardware*.

Nesse sentido, os resultados aqui apresentados se iniciam pela comparação de tempos de reconstrução em diferentes circunstâncias. Primeiro para sinais de domínio 1D usando método direto e indireto, depois para sinais de domínio 2D já esparsos e, por fim, para imagens não-esparsas utilizando o método de pré-filtragem para reconstrução em CS. Os resultados preliminares obtidos foram gerados em *Python* para reconstruções de CS com minimização por ℓp. Note que os métodos para produção de cada gráfico estão descritos no Capítulo 3 desse documento.

4.1 | Comparações entre o método direto e indireto para processos estocásticos em domínio 1D

Inicialmente, a fim de identificar qual é o maior esforço computacional para o processamento de cada método de resolução da minimização por ℓp (direto e inverso), foram realizadas dezenas de reconstruções de cada tipo em um mesmo processador. Para tanto, foi gerado um sinal aleatório com transformada esparsa para cada uma das reconstruções, de forma que, para cada nova reconstrução, o tamanho do sinal gerado recebia um incremento de 250 amostras. Para os dois tipos de reconstrução, o sinal começou com um tamanho de 100 amostras e o critério de parada foi a impossibilidade computacional do método direto, uma vez que o programa não permitia alocar mais memória para as opera-

ções. Os testes de ambos os tipos de reconstruções foram realizados simultaneamente, em uma mesma janela de tempo. A Figura 3 exprime três gráficos com dados obtidos durante as reconstruções.

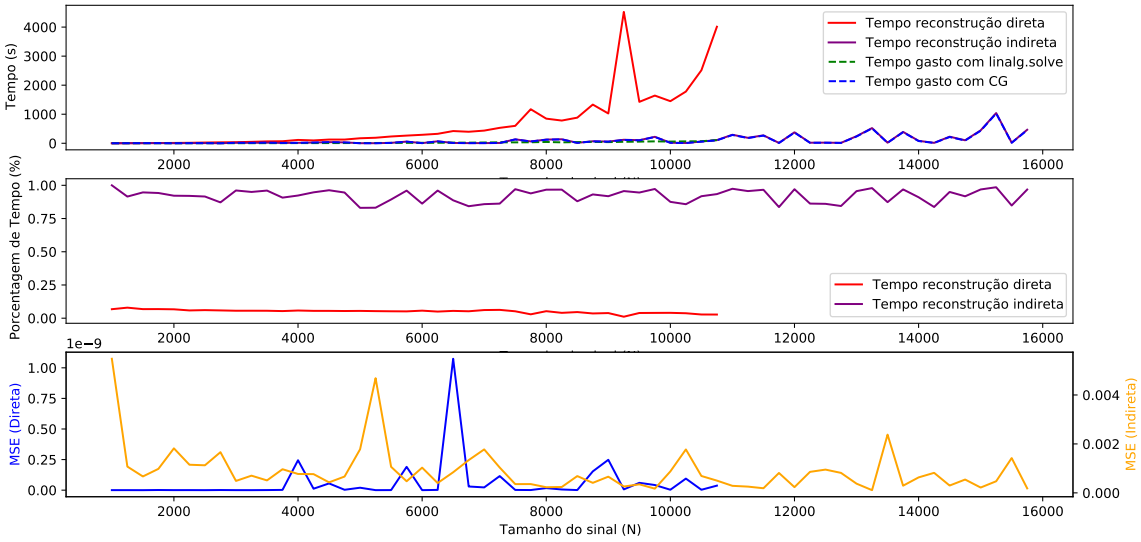


Figura 3 – Comparação de tempo gasto na resolução do sistema linear entre o método direto e o método indireto.

Como está indicado na legenda dos eixos verticais, o primeiro gráfico exibe uma relação entre o tamanho do sinal a ser reconstruído e o tempo gasto na reconstrução em segundos. Foi estratificado também qual é o tempo gasto com a função de resolução do sistema linear de cada caso. A porcentagem desse tempo em relação ao tempo total de reconstrução é apresentada no segundo gráfico. Finalmente, no último gráfico da Figura 3, é indicado o Erro Quadrático Médio (MSE) das reconstruções.

Verifique que, em uma mesma quantidade de tempo, foi possível realizar mais reconstruções com o método indireto do que o com o método direto, e com sinais de tamanhos superiores. Como já foi comentado, isso se dá pelo fato da necessidade do método direto de alocar toda a matriz de transformação na memória, isso fica claro ao perceber que a porcentagem gasta na resolução do sistema linear e irrisória em comparação ao tempo total de alocação da matriz de transformação.

Note também que, apesar do MSE do método direto ser evidentemente menor que o do método indireto, e que, em um primeiro momento, para sinais pequenos, a reconstrução por método direto também é mais rápida. Esse método se demonstra impraticável para sinais maiores que alguns milhares de amostras, que é o caso da maioria dos sinais reais, principalmente na área de imageamento.

Nessa mesma linha, constatou-se que o tempo gasto com a reconstrução cresce linearmente com o tamanho do sinal, possibilitando, em um tempo plausível, a reconstrução

de imagens reais, por exemplo. E também, que o tempo gasto com o *Conjugate Gradient* no método indireto é bem significativo, em média, 92% do tempo total gasto nas reconstruções é para o CG, no caso dos sinais gerados aleatoriamente.

Tendo em vista os dados apresentados acima, fica evidente a necessidade de acelerar o algorítimo de resolução do sistema linear para o método indireto. De modo que, implementações em *hardware* que acelerem a resolução do método indireto podem contribuir com a generalização e com o aumento de aplicações possíveis para reconstrução de sinais com *Compressive Sensing*.

4.2 | Reconstrução por CS de imagens já esparsas no domínio original

4.2.1 | Análise da qualidade de reconstrução por CS em função do número de medidas

Após os resultados obtidos na seção anterior, foi proposto um experimento semelhante com uma imagem de MRI esparsa de tamanho 512x512 gerada computacionalmente. Nesse caso, só é possível utilizar a reconstrução por método indireto, uma vez que o sinal é demasiadamente grande para a reconstrução com método direto. Neste teste, foram realizadas 50 reconstruções por *Compressive Sensing* utilizando a minimização por ℓp no método indireto.

Como descrito no Capitulo 3, a máquina de ressonância magnética captura as medidas em uma trajetória angular e a imagem é construída a partir da composição de vários ângulos dessas medidas. Para simular a aquisição de dados de uma máquina de MRI, uma imagem esparsa foi gerada e amostrada nas posições da trajetória de cada medida angular. Para fins de identificação dos limites da reconstrução por CS, a primeira reconstrução iniciou-se com apenas 5 trajetórias angulares, o que equivale a 0.97% das medidas necessárias para construção da imagem pelo método clássico. Foi realizado um incremento de 5 trajetórias angulares a cada nova reconstrução, para avaliar o MSE das reconstruções em comparação à imagem original. Os resultados desse experimento estão apresentados na Figura 4.

No primeiro gráfico da Figura 4 é exibido o Erro Médio Quadrático das reconstruções. Perceba que para um número muito pequeno de amostras o erro da reconstrução é significativo, indicando a limitação da técnica de reconstrução por CS. Entretanto, rapidamente, o gráfico converge para um valor aceitável de erro, viabilizando a reconstrução de uma imagem de MRI com um baixo nível de amostras. Com menos de 10% de amostras, para

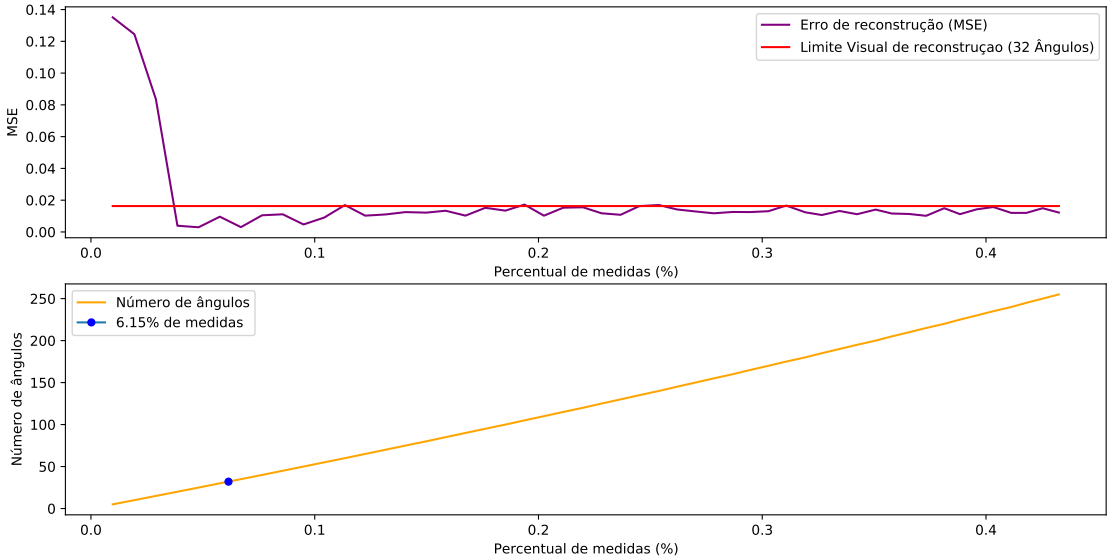
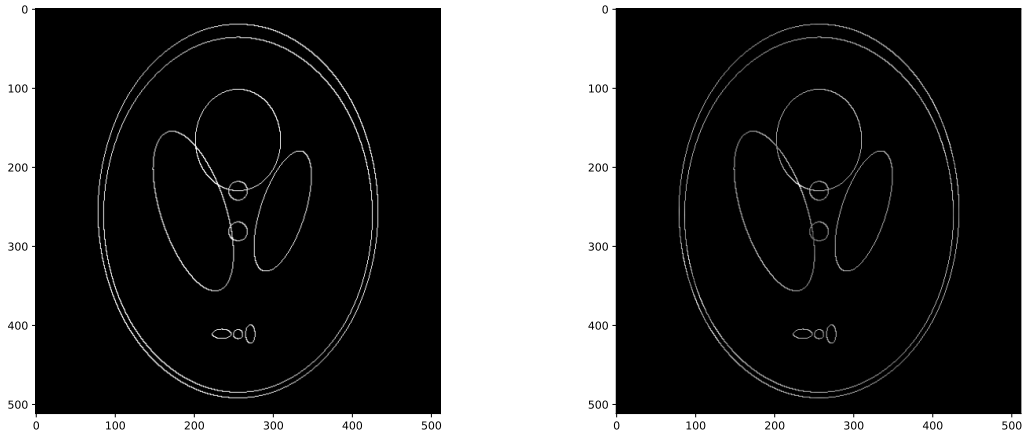


Figura 4 – Comparação do erro de 50 reconstruções de uma imagem 2D simples esparsa variando o número de medidas.

esse caso, foi possível obter imagens razoavelmente próximas à imagem original. Verificou-se também que mesmo com o aumento de aquisição de medidas o erro não variou muito da solução mais esparsa encontrada. Na figura também é apresentada a relação entre o número de trajetórias angulares com as porcentagens de medidas da imagem.

Para fins visuais, a imagem gerada está indicada na Figura 5a e a reconstrução por CS com 32 ângulos (equivalente a 6.15% das medidas necessárias para uma imagem utilizando o método clássico de aquisição) é exibida na Figura 5b.



(a) Exemplo clássico de uma imagem esparsa gerada computacionalmente. (512x512)

(b) Imagem reconstruída com 32 linhas radiais no domínio das frequências (6.15% das medidas).

Figura 5 – Comparação entre um imagem esparsa gerada computacionalmente e sua reconstrução com CS.

4.2.2 | Análise dos tempos de reconstrução em função do número de medidas

Dentro do mesmo experimento de reconstrução de 50 imagens já esparsas no domínio original, variando o número de medidas, foi possível captar a quantidade de tempo gasto com a resolução dos sistemas lineares de cada reconstrução. O resultado é apresentado na Figura 6.

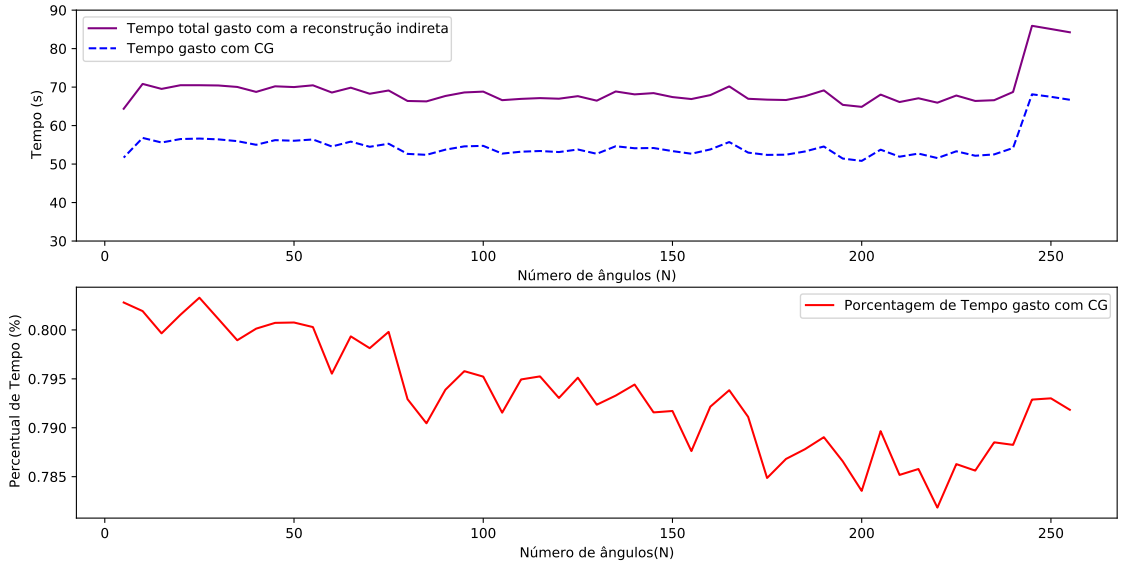


Figura 6 – Dados do tempo total gasto comparado ao tempo gasto com a resolução do sistema linear para 50 reconstruções de uma imagem 2D simples esparsa.

Nota-se que a porcentagem de tempo gasto na resolução do sistema linear se manteve alta, mesmo para uma imagem 2D. E que, na média, quanto menos trajetórias angulares estão disponíveis para reconstrução, mais tempo do programa é dedicado ao CG, inclusive para os casos em que a solução convergiu para a imagem original.

Utilizando a ferramenta *Profiler* do programa *Spyder* do pacote Anaconda foi gerado também uma relação dos tempos gastos em cada função do programa. Esta relação é apresentada da Tabela 1. Na tabela é apresentado o tempo total gasto em uma reconstrução com 32 trajetórias angulares (6.15% das medidas). É indicado também o tempo gasto com a resolução do sistema linear por meio do Conjugate Gradient e a estratificação do tempo gasto em cada função utilizada pelo CG.

Tabela 1 – Dados do tempo total gasto comparado ao tempo gasto com a resolução do sistema linear para a reconstrução de uma imagem 2D simples esparsa com 6.15% das medidas.

	Tempo Gasto (s)	**Porcentagem relação ao total (%)**
Tempo total do Programa (Lp Minimization)	235.20	100%
Conjugate Gradient	213.60	91%
Matrix to Column	59.12	25%
Column to Matrix	38.28	16%
IFFT2	41.79	18%
FFT2	37.66	16%
Hermitiano	0.30	0%
Outras Operações	36.45	15%

4.3 | Reconstrução de uma imagem real com Pré-Filtragem

Foi proposto também um teste para verificação da técnica de pré-filtragem. Como descrito anteriormente, a pré-filtragem dispensa a necessidade de uma transformada esparsificante. Assim, uma imagem real de tamanho 256x256 de MRI foi retirada do banco de dados da Universidade de Stanford para ser reconstruída.

De forma semelhante ao experimento anterior, a imagem foi amostrada nas frequências das trajetórias angulares geradas, e foi realizada a variação do número de trajetórias, iniciando com 25 ângulos, cerca de 9.48% das medidas, e finalizando com 300 ângulos, equivalente a 81.31% das medidas para obtenção da imagem no método clássico. Entretanto, neste caso, foi realizada uma pré-filtragem das medidas realizando uma multiplicação no domínio da frequência com as funções de transferência dos Filtros de Haar. Assim, foi efetuada a reconstrução de três imagens esparsas filtradas (componentes de alta frequência) e, posteriormente, realizada a composição da imagem com os resultados obtidos das imagens filtradas com as componentes de baixa frequência medidas. Os dados para esse teste estão indicados na Figura 7.

Como era esperado, o Erro Quadrático Médio diminui com o acréscimo da quantidade de amostras medidas. E mesmo para os casos com uma baixa densidade de amostras o MSE se manteve relativamente baixo. Subjetivamente, também foi indicado no gráfico um limite visual do erro para a reconstrução da imagem em teste. Este limite foi verificado com 140 trajetórias angulares, cerca de 48% das medidas necessárias para a obtenção da imagem no método clássico. Tal reconstrução pode ser visualizada na Figura 8b ao lado

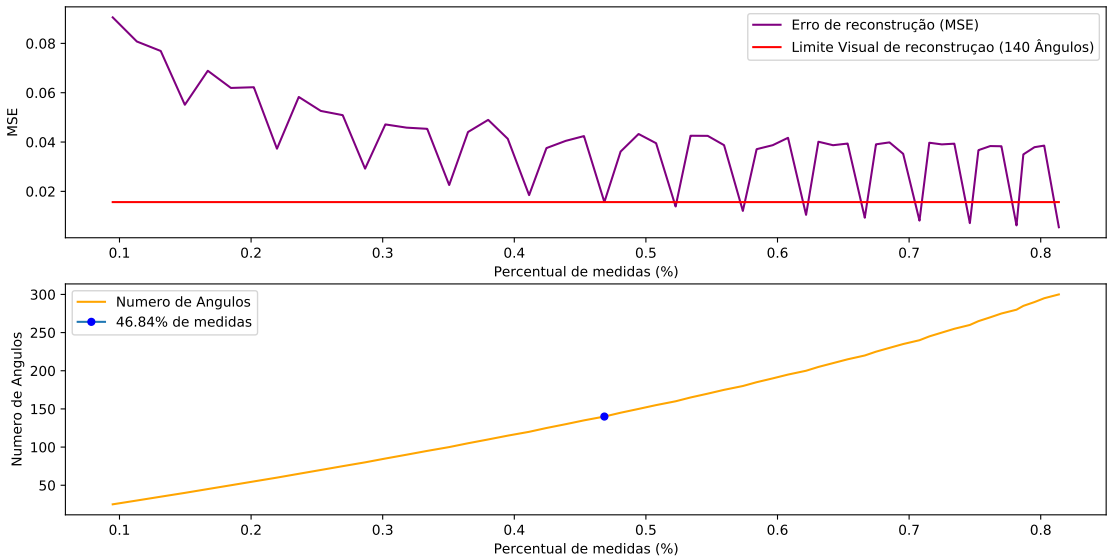
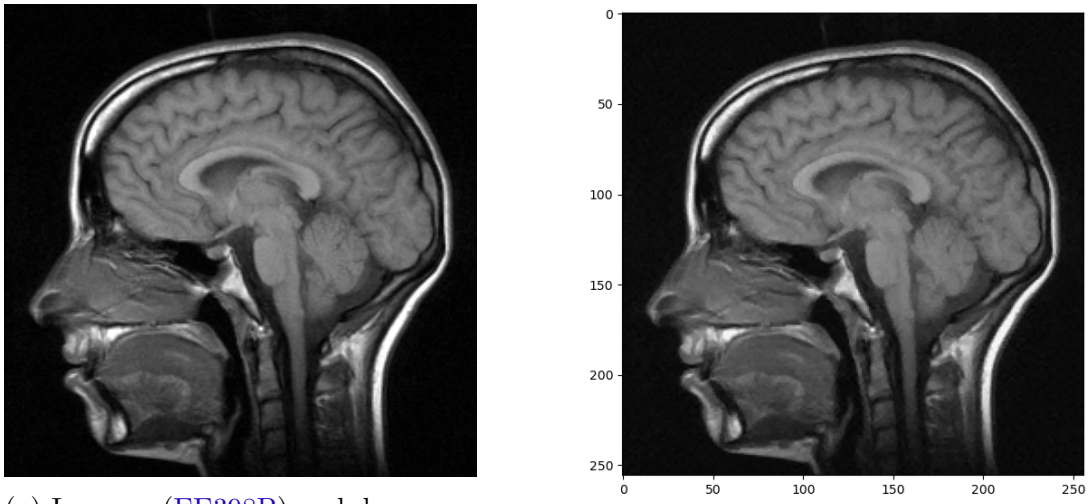


Figura 7 – Dados de reconstruções de uma imagem real com Pré-Filtragem variando o número de trajetórias angulares medidas.

na imagem real apresentada na Figura 8a.



(a) Imagem (EE398B) real de ressonância magnética. (256x256)

(b) Imagem reconstruída com 140 ângulos de medidas (48% das medidas)

Figura 8 – Comparação entre uma imagem real e sua reconstrução com CS.

Para o caso da pré-filtragem, é possível paralelizar a reconstrução da imagem, dado que a técnica de pré-filtragem consiste na realização de uma série de reconstruções simples de imagens esparsas filtradas, de maneira que cada reconstrução pode ser processada simultaneamente com as outras. O que reforça os benefícios da implementação em *hardware* dessa técnica. Apesar de não estar apresentado no gráfico, o tempo médio gasto com CG se manteve o mesmo que nas reconstruções anteriores.

4.4 | Análise comparativa da FFT e IFFT implementadas em FPGA e no processador ARM

Ainda para validação do uso de FPGAs para reconstrução por CS, foi criado um exemplo simples de implementação de uma FFT e IFFT dentro da PYNQ-Z2. Utilizando a ferramenta de IP do Vivado, foram geradas os blocos de comunicação com o processador, e duas memórias de acesso compartilhado entre o ARM e o módulo da Transformada de Fourier. Uma memória para configuração do bloco e outra para alocação dos dados transformados. Dessa forma, foi gerado o arquivo de descrição de *hardware* compilado (*.bitstrem*) e utilizando as bibliotecas em *Python* da PYNQ-Z2, foi realizado a associação do módulo com o processador. De maneira que o módulo para transformada de Fourier ficou disponível para utilização do processador com a sequência de alguns comandos simples.

Como os módulos da FFT e da IFFT estão limitados a utilização de potências de 2, foram realizados testes da transformada de Fourier com sinais aleatórios de tamanho $N = 2^6$ até $N = 2^{14}$. As Figuras 9 e 10 mostram a relação do tempo médio gasto de 10 mil transformadas para cada potência de 2. As Figuras ainda indicam o erro relativo médio (RMSE) das transformadas realizadas em FPGA e das realizadas no ARM.

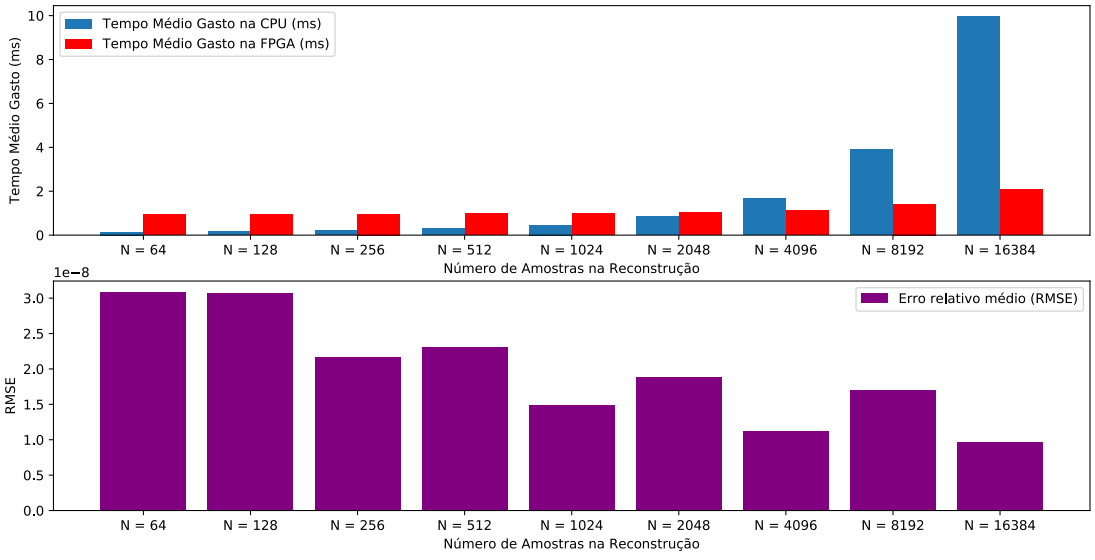


Figura 9 – Comparação do tempo de processamento da FFT em *hardware* e em *software* variando o tamanho do sinal com 10 mil interações.

Como mostra o gráfico, para sinais com tamanho $N = 2^{14}$, a FFT implementada em FPGA foi de 5 a 6 vezes mais rápida que a FFT da CPU e a IFFT foi de 2 a 3

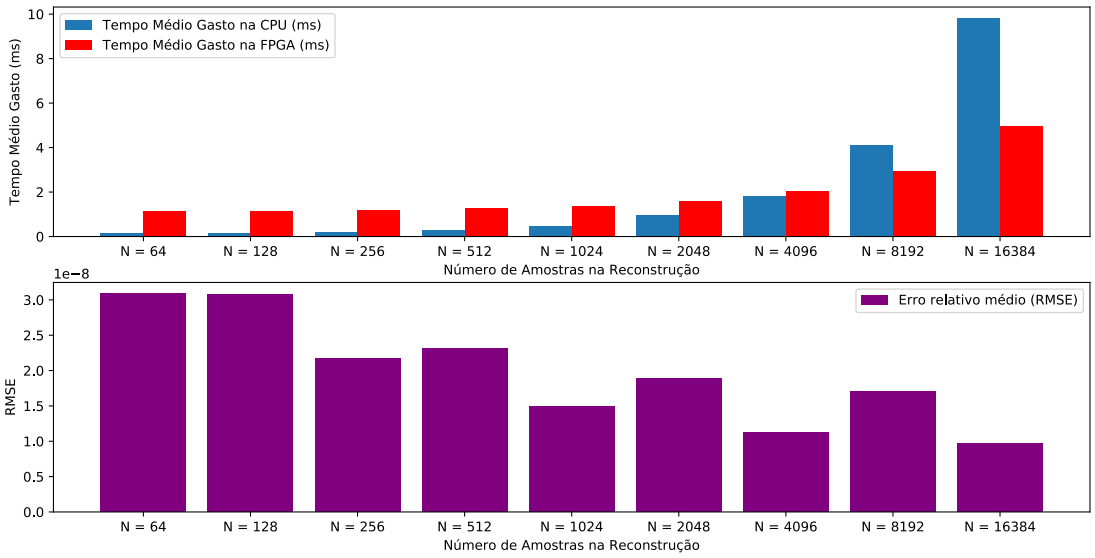


Figura 10 – Comparação do tempo de processamento da IFFT em *hardware* e em *software* variando o tamanho do sinal com 10 mil interações.

vezes. O que mostra o ganho de utilizar módulos *hardware* para aceleração de algoritmos, principalmente para sinais grandes. Outra coisa que pode ser observada, é que não houve um aumento significativo do tempo de processamento gasto para a FFT implementada em FPGA quando houve um acréscimo ao tamanho do sinal. Isso ocorre pelo fato do tempo de comunicação entre *chips* ser maior que a latência do próprio módulo. (XILINX, 2012) Vale notar que não foi possível realizar os testes com sinais de tamanho superior por conta da limitação do número de DSPs disponíveis na placa de desenvolvimento.

4.5 | Análise comparativa da FFT e IFFT com reconfiguração implementadas em FPGA e no processador ARM

Para aproveitar o máximo de DSPs disponíveis na PYNQ-Z2, ainda foram implementadas modos reconfiguráveis para utilização tando da FFT como da IFFT em um mesmo bloco implementado de silício. As Figuras 11 e 12 mostram a mesma comparação realizada do experimento anterior, porém, nota-se uma perda de processamento devido a necessidade de reconfiguração.

Note que, principalmente para o a FFT, houve uma perda em relação ao tempo de processamento observado anteriormente. Esse atraso ocorre devido a necessidade de aguardar que o módulo esteja pronto para receber o sinal a ser transformado. Mas principalmente,

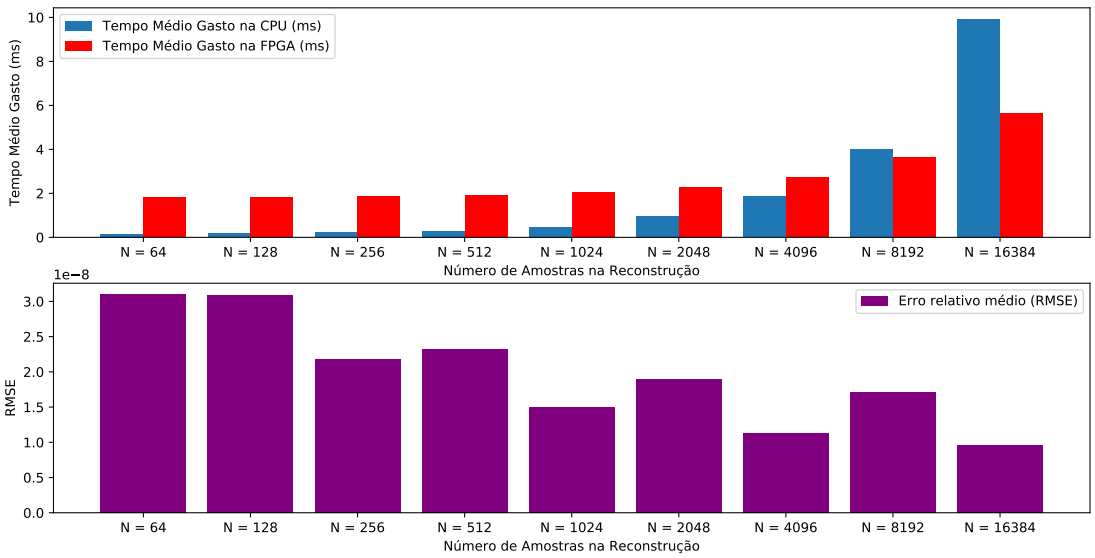


Figura 11 – Comparação do tempo de processamento da FFT em *hardware* com reconfiguração e em *software* variando o tamanho do sinal com 10 mil interações.

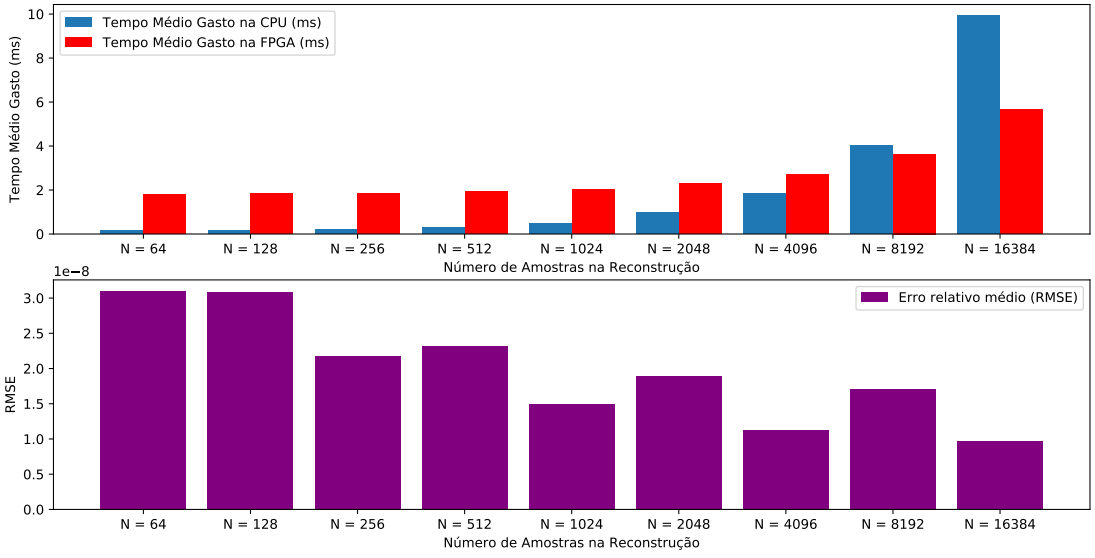


Figura 12 – Comparação do tempo de processamento da IFFT em *hardware* com reconfiguração e em *software* variando o tamanho do sinal com 10 mil interações.

o atraso é ocasionado pela demora de transferência de dados entre o ARM e a FPGA.

4.6 | Análise comparativa da velocidade de processamento de uma reconstrução de CS completa com FFT acelerada em FPGA

Para este último experimento, foi implementado uma reconstrução completa por *Compressive Sensing* na placa de desenvolvimento PYNQ-Z2, de forma que a FFT foi processado em *hardware*. Para a reconstrução em CS, a IFFT também é necessária. Portanto, é necessário escrever na memória alocada para configuração do módulo, de forma a aproveitar a mesma área de silício, tanto para a transformada de Fourier direta como para a inversa.

Foram realizadas 10 mil reconstruções para cada tamanho de sinal, foi calculada e média do tempo gasto e o erro em comparação com o sinal real. Os dados estão apresentados na Figura 13.

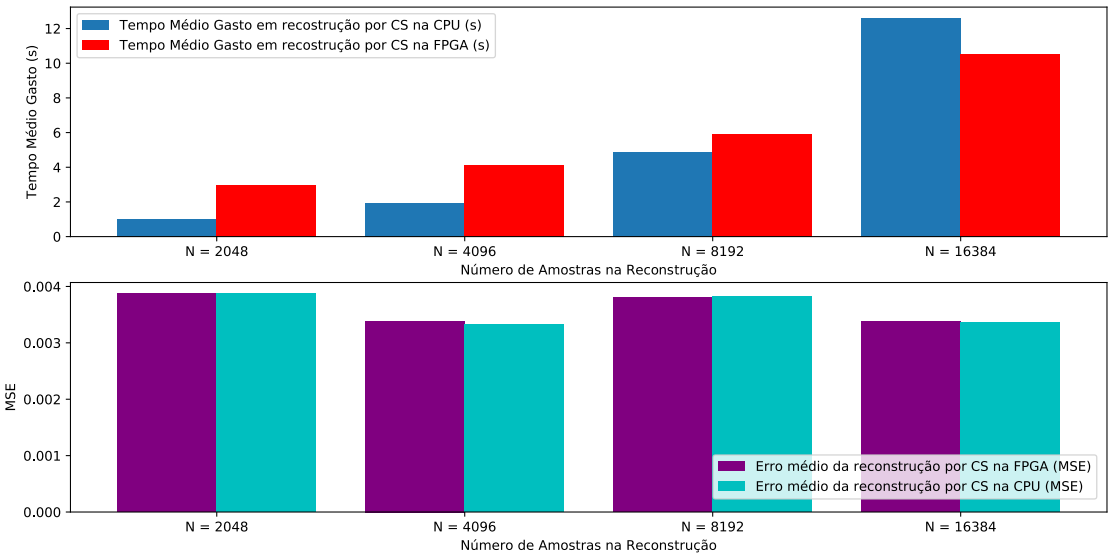


Figura 13 – Comparação do tempo de processamento de 10 mil reconstruções de CS com a FFT acelerada em *hardware* variando o tamanho do sinal.

Notou-se uma melhora considerável no tempo de processamento de sinais com tamanho $N = 2^{14}$ porém, não foram tão expressivos como os ganhos da FFT isolada. Isso se dá pelo fato de, durante a reconstrução, a FFT e a IFFT são chamadas por várias vezes. De maneira que o módulo precisa sempre ser reconfigurado. Além do tempo gasto para transmissão de todos esses dados. Porém, mesmo assim, nota-se que, com o aumento do tamanho do sinal a melhora fica cada vez mais significativa.

5 Conclusão

Os experimentos realizados sugerem que a utilização do método direto para reconstruções de imagens reais em CS (figuras de tamanho superior a 512×512) é ineficaz e limitada à memória disponível para processamento. Principalmente se comparadas ao método indireto de resolução dos sistemas lineares. Verificou-se também que, no caso da reconstrução por CS usando o método indireto, a maior parte do tempo de processamento é gasta com a resolução dos sistemas lineares.

Não só isso, mas também ficou constatado que a utilização do método de pré-filtragem para reconstrução em CS viabiliza a reconstrução de sinais com o auxilio de filtros que esparsificam o sinal de interesse para o domínio da frequência. Possibilitando a utilização da FFT como função da matriz de medidas para casos conhecidos como imagens de MRI, por exemplo. Além do enorme potencial para paralelização do processamento.

Por fim, ficou evidenciado também os enormes ganhos que a utilização de *hardware* reprogramável e FPGAs podem trazer para dentro da área de CS. Tanto para aceleração de algoritmos custosos para o processador, como para paralelização de etapas normalmente sequenciais dentro do processador.

Como próximas etapas, fica a possibilidade de implementar a FFT de domínio bidimensional em FPGA. Ainda é possível implementar N FFTs 1D de tamanho N aproveitando a característica de paralelização da FFT 2D. Outra possibilidade é a implementação das interações do *Conjugate Gradient* em FPGA, para evitar o alto fluxo de dados entre *chips* e focar no processamento. Assim, reduzindo o tempo de acesso à memória e paralelizando o processamento da FFT bidimensional. E também, utilizar métodos como o de pré-filtragem com módulos de hardware implementados em FPGA para paralelizar a reconstrução de cada parte do sinal filtrado. Espera-se um ganho expressivo na velocidade de processamento do *Compressive Sensing* para implementações semelhantes a essas citadas.

Referências

BOYD, S.; BOYD, S. P.; VANDENBERGHE, L. *Convex optimization*. [S.l.]: Cambridge university press, 2004. Citado na página 22.

CANDÈS, E. J.; WAKIN, M. B. An introduction to compressive sampling. *IEEE signal processing magazine*, IEEE, v. 25, n. 2, p. 21–30, 2008. Citado na página 15.

DONG, X.; LI, H.; WANG, Y. High-speed fpga implementation of an improved lms algorithm. In: *2013 International Conference on Computational Problem-Solving (ICCP)*. [S.l.: s.n.], 2013. p. 342–345. Citado na página 31.

DONOHO, D. L. For most large underdetermined systems of linear equations the minimal ℓ_1-norm solution is also the sparsest solution. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, Wiley Online Library, v. 59, n. 6, p. 797–829, 2006. Citado na página 24.

FILHO, J. A.; SILVA, N.; MIOSSO, C. Detection of schizophrenia based on brain structural analysis, using machine learning over different combinations of multi-slice magnetic resonance images. In: SPRINGER. *Brazilian Congress on Biomedical Engineering*. [S.l.], 2022. p. 2039–2044. Citado na página 13.

GARCIA, Y.; FRANCO, C.; MIOSSO, C. Method for improved image reconstruction in computed tomography and positron emission tomography, based on compressive sensing with prefiltering in the frequency domain. In: SPRINGER. *Brazilian Congress on Biomedical Engineering*. [S.l.], 2022. p. 2019–2025. Citado na página 14.

KATO, M. et al. Non-destructive drug inspection in covering materials using a terahertz spectral imaging system with injection-seeded terahertz parametric generation and detection. *Optics express*, Optical Society of America, v. 24, n. 6, p. 6425–6432, 2016. Citado na página 13.

MIOSSO, C. J. *Compressive sensing with prior information applied to magnetic resonance imaging*. [S.l.]: The University of Texas at El Paso, 2010. Citado na página 24.

MIOSSO, C. J. et al. Compressive sensing reconstruction with prior information by iteratively reweighted least-squares. *IEEE Transactions on Signal Processing*, v. 57, n. 6, p. 2424–2431, 2009. Citado 3 vezes nas páginas 16, 18 e 29.

MIOSSO, C. J.; BORRIES, R. von; PIERLUISSI, J. H. Compressive sensing method for improved reconstruction of gradient-sparse magnetic resonance images. In: *2009 Conference Record of the Forty-Third Asilomar Conference on Signals, Systems and Computers*. [S.l.: s.n.], 2009. p. 799–806. Citado na página 26.

NASCIMENTO, J. M. P.; VéSTIAS, M. Fpga-based architecture for hyperspectral compressive sensing. In: *2021 Telecoms Conference (ConfTELE)*. [S.l.: s.n.], 2021. p. 1–6. Citado na página 31.

NOCEDAL, J.; WRIGHT, S. J. *Numerical optimization*. [S.l.]: Springer, 1999. Citado 4 vezes nas páginas 22, 23, 26 e 29.

SHANNON, C. E. A mathematical theory of communication. *The Bell system technical journal*, Nokia Bell Labs, v. 27, n. 3, p. 379–423, 1948. Citado na página 20.

SHEVCHUK, R. M. Technique for satellite monitoring of illegal amber mining territories based on integrated landsat and sentinel data processing. *Journal of the Georgian Geophysical Society*, v. 21, n. 1, 2018. Citado 2 vezes nas páginas 13 e 14.

SHILOV, G. E. *Linear algebra*. [S.l.]: Courier Corporation, 2012. Citado na página 21.

VETTERLI, M.; KOVAČEVIĆ, J.; GOYAL, V. K. *Foundations of signal processing*. [S.l.]: Cambridge University Press, 2014. Citado na página 29.

WATTJES, M.; STEENWIJK, M.; STANGEL, M. Mri in the diagnosis and monitoring of multiple sclerosis: an update. *Clinical neuroradiology*, Springer, v. 25, n. 2, p. 157–165, 2015. Citado na página 13.

XILINX. Zynq-7000 soc data sheet: Overview. 2018. Citado na página 31.

XILINX, I. Logicore ip fast fourier transform v8. 0, product specifications ds808. 2012. Citado na página 41.

A Proposta de Implementação do *Conjugate Gradient* em FPGA

Fica como estudo futuro a implementação da resolução do sistema linear completo em FPGA, para o caso do método indireto. O diagrama da Figura 14 é uma proposta de implementação do *Conjugate Gradient* para FPGA.

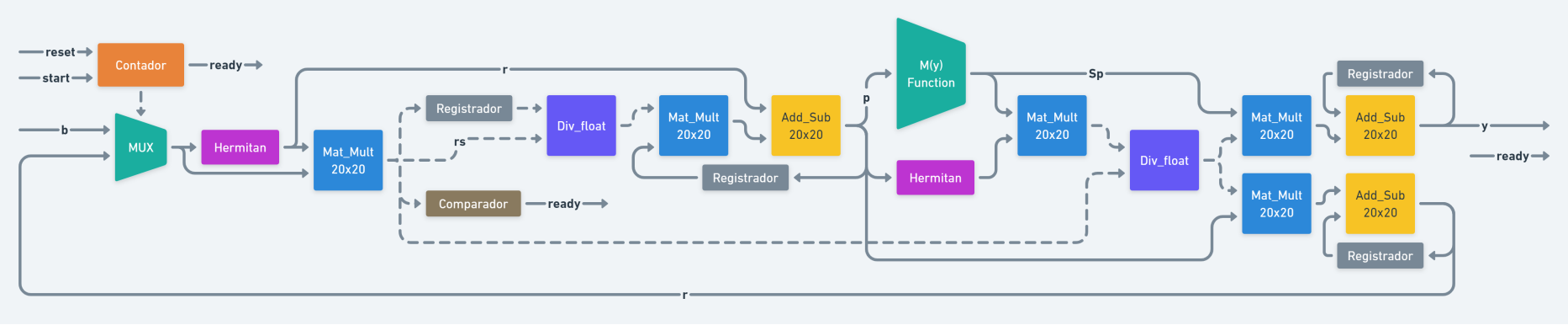


Figura 14 – Diagrama para implementação em FPGA do Conjugate Gradient