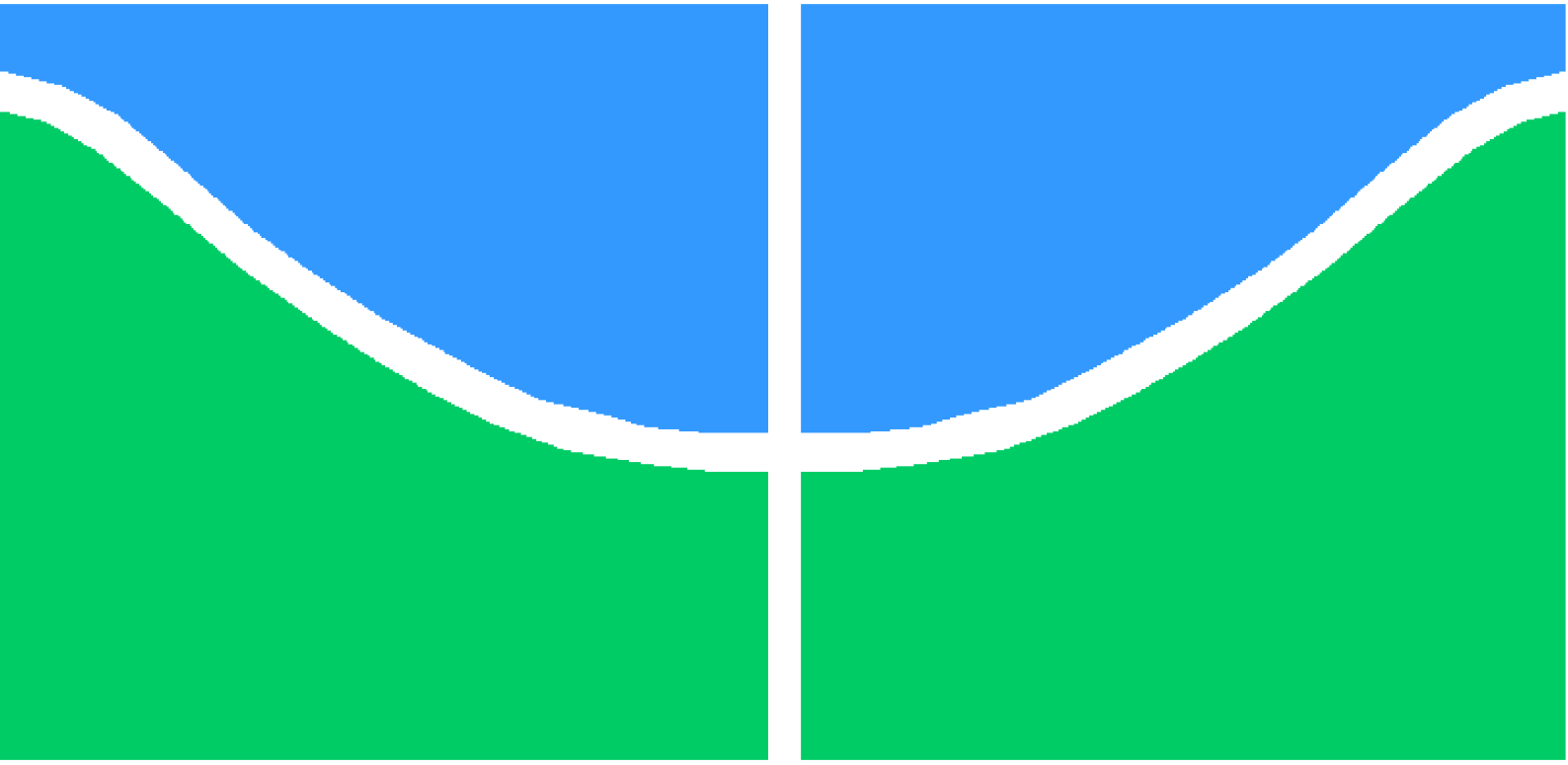

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Engenharia Eletrônica

Reconstrução de Imagem de Ressonância Magnética Nuclear Utilizando Sistema Reconfigurável Simulado.

Autor: Amauri da Costa Júnior

Orientador: Prof. Dr. Gerardo Antonio Idrobo Pizo

Brasília, DF

2020

Amauri da Costa Júnior

Reconstrução de Imagem de Ressonância Magnética Nuclear Utilizando Sistema Reconfigurável Simulado.

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: Prof. Dr. Gerardo Antonio Idrobo Pizo

Brasília, DF

2020

Amauri da Costa Júnior

Reconstrução de Imagem de Ressonância Magnética Nuclear Utilizando Sistema Reconfigurável Simulado./ Amauri da Costa Júnior. – Brasília, DF, 2020-

66 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Gerardo Antonio Idrobo Pizo

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2020.

1. Ressonância Magnética. 2. VHDL. I. Prof. Dr. Gerardo Antonio Idrobo Pizo. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Reconstrução de Imagem de Ressonância Magnética Nuclear Utilizando Sistema Reconfigurável Simulado.

CDU 02:141:005.6

Amauri da Costa Júnior

Reconstrução de Imagem de Ressonância Magnética Nuclear Utilizando Sistema Reconfigurável Simulado.

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, 30 de novembro de 2020 – Data da aprovação do trabalho:

Prof. Dr. Gerardo Antonio Idrobo Pizo
Orientador

Prof.a Dra. Gabriela Cunha Possa
Convidado 1

Prof. Dr. Marcus Vinícius Chaffim Costa
Convidado 2

Brasília, DF
2020

Agradecimentos

Primeiramente gostaria de agradecer a Deus por me dar a calma e paciência para chegar até aqui. Gostaria de agradecer meus pais pelo incentivo e por me dar condições de concluir esse trabalho, também agradecer meus irmãos pelo apoio e amizade durante momentos difíceis do curso .

Gostaria de agradecer minha namorada pelo apoio e companhia que me ajudou a enfrentar os momentos difíceis dessa reta final. Também sou grato pelas pessoas incríveis que conheci durante essa jornada na UnB e as amizades que levarei pelo resto da vida. Por fim gostaria de agradecer meu orientador, professor Gerardo, por ter apoiado minha ideia de TCC e pela sua empolgação com a engenharia que foi um incentivo durante todo o curso.

Resumo

A ressonância magnética tem sido uma das mais pesquisadas e utilizadas formas de obtenção de imagens médicas nos últimos anos devido a sua gama de possibilidades em imageamento, reconstruções mais eficientes surgem todos os dias e este trabalho abordará a reconstrução em hardware. A reconstrução de imagens de ressonância devem ser feitas de forma rápida com qualidade e precisão, esse processo geralmente é realizado por computadores porém, a utilização de algoritmos executados em hardware pode se mostrar eficiente por possuir alta velocidade e precisão no tempo. Foram executadas simulações do processo de reconstrução em hardware em VHDL utilizando imagens diferentes para testar o funcionamento do algoritmo, tanto imagens normais quanto imagens médicas de ressonância magnética, os resultados foram então comparados com as imagens originais e com os padrões esperados para exames de ressonância. O algoritmo produzido mostrou-se eficiente, produzindo imagens com alta qualidade, de forma rápida não sendo um fator limitante para as máquinas de ressonância e com alta precisão no tempo, executando a reconstrução de imagens diferentes sem flutuações no tempo de reconstrução. O trabalho consegue mostrar as vantagens da reconstrução em hardware, obtendo pelo menos 60% de semelhança entre a imagem reconstruída e a original e uma fórmula que permite calcular o tempo de processamento do algoritmo em função da frequência de clock e das dimensões da imagem, podendo assim reconstruir até mesmo 4000 imagens por segundo.

Palavras-chaves: MRI. FPGA. VHDL. Ressonância Magnética.

Abstract

Magnetic resonance imaging has been one of the most researched and used ways of obtaining medical images in recent years due to its range of possibilities in imaging, more efficient reconstructions appear every day and this work will address hardware reconstruction. The reconstruction of resonance images must be done quickly with quality and precision, this process is usually performed by computers, however, the use of algorithms performed on hardware can be efficient because it has high speed and accuracy over time. Simulations of the hardware reconstruction process in VHDL were performed using different images to test the operation of the algorithm, both normal images and medical magnetic resonance images, the results were then compared with the original images and with the expected patterns for resonance exams. The algorithm produced proved to be efficient, producing images with high quality, quickly and not being a limiting factor for resonance machines and with high precision in time, performing the reconstruction of different images without fluctuations in the reconstruction time. The work is able to show the advantages of hardware reconstruction, obtaining at least 60 % similarity between the reconstructed image and the original and a formula that allows to calculate the processing time of the algorithm according to the clock frequency and the dimensions of the image, thus being able to reconstruct even 4000 images per second.

Key-words: MRI. FPGA. VHDL. Magnetic Resonance Imaging.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

2DIFFT	2 Dimensions Inverse Fast Fourier Trasform, Transformada Inversa Rápida de Fourier de 2 Dimensões.
FFT	Fast Fourier Trasform, Transformada Rápida de Fourier.
FPGA	Field Programmable Gate Array.
GPU	Graphics Processing Unit.
I2C	Inter-Integrated Circuit.
IFFT	Inverse Fast Fourier Trasform, Transformada Inversa Rápida de Fourier.
MRI	Magnetic Resonance Imaging, Imageamento por Ressonância Magnética.
SoC	System on a Chip.
SPI	Serial Peripheral Interface.
UART	Universal Asynchronous Receiver/Transmitter.
VHDL	VHSIC Hardware Description Language.
VHSIC	Very High Speed Integrated Circuits.

Lista de símbolos

γ	Letra grega gama minúscula. Razão giromagnética. [MHz/T].
α	Letra grega alfa minúscula. Ângulo arbitrário. [rad].
Δ	Letra grega delta maiúscula. Variação.
ϕ	Letra grega phi maiúscula. Ângulo arbitrário. [rad].
φ	Letra grega phi minúscula. Codificação de Fase. [rad].
ω	Letra grega ômega minúscula. Codificação de Frequência. [rad].
τ	Letra grega tau minúscula. Tempo. [s].

Sumário

1 Introdução

Em 1973 o químico Paul C. Lauterbur publicou as primeiras imagens obtidas através da ressonância magnética de núcleos de átomos. Ele utilizou dois capilares de vidro contendo água em um campo magnético uniforme. Ao aplicar um campo magnético gradiente ele obteve como resposta um sinal unidimensional como mostrado na Figura 1, rotacionando os capilares em ângulos de 45º e combinando os sinais unidimensionais gerados, ele foi capaz de chegar em uma imagem bidimensional de um corte transversal dos capilares mostrado na Figura 2 (LAUTERBUR et al., 1973).

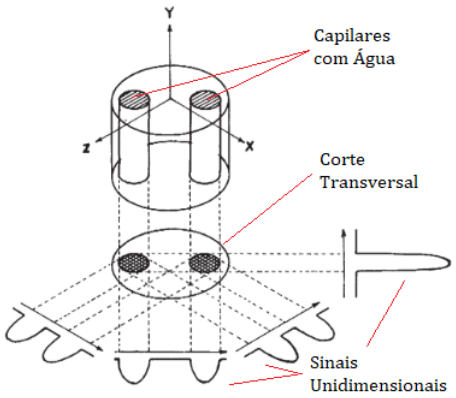

Figura 1 – Projeções unidimensionais realizadas.(LAUTERBUR et al., 1973) modificada.

Figura 2 – Imagem bidimensional obtida.(LAUTERBUR et al., 1973) modificada.

Formas de adquirir imagens do interior do corpo humano foram estudadas ostensivamente durante todo o século passado, e ainda hoje pesquisadores avançam ainda mais na área tentando encontrar formas cada vez mais eficientes e precisas de obter essas imagens. Entre as ferramentas mais estudadas de imageamento médico está a ressonância magnética, o número de artigos com MRI como palavra chave na plataforma de periódicos CAPES, publicados entre 1991 e 2000 foram de quase 100 mil, e entre 2001 e 2010 esse

número passou dos 300 mil artigos, esses números mostram o quanto a pesquisa na área vem crescendo e o quanto ainda pode ser explorado.

Imagens de ressonância magnética ganharam grande espaço no meio médico devido a sua precisão e pela quantidade de aplicações das quais pode se obter imagens. O número de máquinas ressonância magnética só nos Estados Unidos no ano de 2017 foi de mais de 12 mil equipamentos (OECD, 2019b) o número de exames realizados passou de 36 milhões (OECD, 2019a). O exame de ressonância magnética consegue obter imagens nítidas de tecidos, e é muito usado para criar imagens do cérebro, músculos, tendões e ligamentos. Diferentemente de outros métodos como raio-x e tomografia, a ressonância utiliza radiação não-ionizante, dessa forma, os risco envolvidos com o exame são menores, porém devido a intensidade do campo magnético nenhum metal ferromagnético deve entrar na mesma sala que o equipamento, por isso, pacientes com certos implantes não podem fazer ressonância magnética. A máquina também pode estimular alguns nervos do corpo, causando vibrações no tecido ou muito raramente queimaduras (HAIK et al., 2009).

O imageamento por ressonância magnética utiliza de poderosos campos magnéticos que são 30 mil vezes mais intensos que o campo magnético terrestre (MAZZOLA, 2009), para interagir com os tecidos do corpo, uma sequência de sinais eletromagnéticos então é aplicado fazendo com que os átomos de hidrogênio saiam do seu estado de equilíbrio, ao voltar para seu estado original emitem sinais, que são captados por bobinas, esses sinais são então utilizados para reconstruir a imagem.

Exames de ressonância magnética também podem ser utilizados para obter imagens em tempo real para que possam ser observados órgãos e tecidos em movimento, também é possível obter imagens em 3D, assim como imagens de fluxo sanguíneo e atividade cerebral (HEALTHINEERS, 2019).

Uma forma de aumentar a velocidade de reconstrução da imagem é através de algoritmos que utilizam da transformada rápida de Fourier (FFT) para reconstruir essa imagem, a implementação desses algoritmos em uma FPGA seria uma forma de acelerar esse processamento (RAO; KIM; HWANG, 2011).

FPGAs (Field Programmable Gate Arrays) consistem em uma matriz reprogramável de blocos lógicos, que pode ser programada de acordo com a necessidade do usuário, o que também permite que sejam feitas alterações de acordo com as evoluções do projeto. FPGAs possuem alta velocidade de processamento com clocks ultrapassando a barreira de 500 MHz, podendo realizar rapidamente o cálculo da FFT, são versáteis podendo ser integradas a diferentes sistemas usando diversos protocolos de comunicação (I2C, UART, SPI) e também possuem um baixo consumo energético (XILINX, 2019). A FPGA também permite a integração com ferramentas gráficas como MATLAB e Labview, o que facilita a programação, processamento e visualização das imagens. (MATHWORKS, 2019).

A escolha do uso da FPGA ao invés de uma GPU se dá devido alguns motivos. A relação desempenho por consumo energético da FPGA é superior a de GPUs, a FPGA também gera menos calor, dispensando espaço adicional para dissipadores de calor. Por ocupar pouco espaço a FPGA pode ser integrada a equipamentos existentes e a FPGA possui suporte a um maior número de interfaces que um GPU. GPUs são limitadas a plataformas com PCIe. FPGAs podem ser integradas a vários sistemas podendo ser programadas utilizando CPUs, e ser integradas a dispositivos SoC (BERTEN, 2016).

GPUs são processadores baseados em software e realizam operações procurando e decodificando instruções operacionais, todo esse processo toma muitos ciclos de clock. FPGAs são processadores baseados em hardware e devido ao fato de realizar as operações através de portas lógicas consegue realizar essas operações sem o processo de busca e decodificação.

1.1 Definição do Problema

Adquirir os dados, reconstruir de forma rápida e com qualidade são problemas encontrados no imageamento por ressonância magnética. Novas formas de se obter e reconstruir rapidamente as imagens são muito estudadas, de forma a aumentar o número de exames realizados, e aumentar a velocidade de emissão de diagnósticos. Equipamentos com capacidade de processamento cada vez maior, uso de compressed sensing, uso de processadores gráficos, e processamento em hardware são algumas das formas de agilizar o processo de aquisição e processamento desses dados. A instabilidade dos clocks de GPUs também é um problema na reconstrução de imagens em tempo real, GPUs possuem uma instabilidade de tempo dez vezes maior que a da FPGA (Jitter) a Figura 3 mostra como é a instabilidade de clock. (LI; WYRWICZ, 2018).

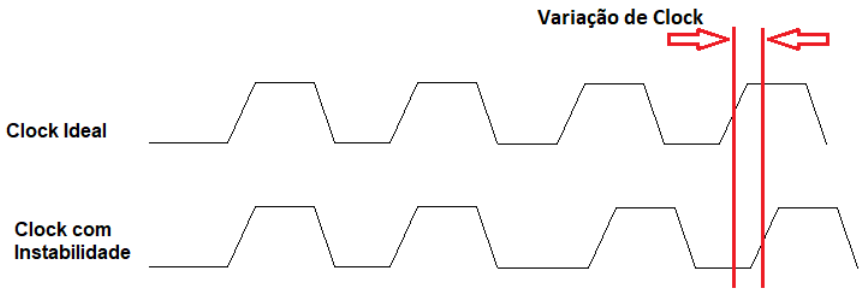

Figura 3 – Instabilidade de Clock.

Precisão no tempo é extremamente necessário na hora de processar imagens em tempo real, por isso para aplicações em que essa precisão é necessária a FPGA pode se mostrar mais adequada.

1.2 Objetivos

Objetivos Gerais

Este trabalho tem como objetivo apresentar a teoria relacionada ao processo de aquisição de imagens de ressonância magnética e sua reconstrução e desenvolver uma ferramenta que reconstrua essa imagem em VHDL para uma futura implementação em uma FPGA que entregue alta qualidade de imagem, velocidade e precisão no tempo e seja comparativamente tão eficiente quanto a reconstrução realizada em uma GPU.

Objetivos Específicos

- Apresentar uma introdução a base teórica da aquisição e reconstrução de imagens de ressonância magnética;
- Apresentar uma ferramenta simulada feita em VHDL capaz de fazer a reconstrução de imagens de ressonância magnética a partir de um espaço K simulado;
- Apresentar os resultados do algoritmo e comparar os resultados com as imagens originais reconstruídas em um computador.

1.3 Organização do Trabalho

O trabalho está organizado em cinco capítulos. O primeiro capítulo apresenta uma introdução ao assunto abordado. O segundo capítulo apresenta a base teórica e os conceitos relacionados a ressonância magnética, o processo de aquisição dos sinais, sequência de pulsos e cálculos envolvidos. No terceiro capítulo é introduzido a forma como o trabalho foi realizado, assim como o funcionamento do algoritmo. No quarto capítulo são apresentados os resultados obtidos e análises feitas, contendo comparações do desempenho da simulação realizada e exames de ressonância. No quinto capítulo é apresentada as considerações finais e sugestões para uma futura implementação prática em hardware.

2 Fundamentação Teórica

2.1 Considerações Iniciais

Nesta seção será abordada a base teórica física e matemática do imageamento por ressonância magnética apresentando cálculos e imagens que auxiliam no entendimento do tópico. Serão mostrados as diversas etapas que levam a criação de uma imagem de ressonância e também os diferentes métodos de obtenção da imagem.

2.2 Átomos de Hidrogênio

As máquinas modernas de ressonância magnética utilizam átomos de hidrogênio para a obtenção de sinais, principalmente pelo fato do hidrogênio ser o terceiro elemento mais abundante no corpo humano, totalizando 9.98% da massa do corpo. O átomo de hidrogênio possui alto momento magnético (BRUKER, 2011), tornando-o mais sensível a campos magnéticos externos. O hidrogênio também difere bastante de um tecido normal para um tecido com alguma patologia (HELLMICH; SZABO, 2015), fazendo com que seja uma boa escolha para observar essas diferenças.

2.3 Gradiente de Seleção de Corte

O processo de obtenção de imagens de ressonância segue um conjunto de procedimentos. Primeiramente é aplicado um campo magnético gradiente ao longo do corpo do paciente, eixo z, esse é chamado de campo magnético principal, ou de gradiente de seleção de corte.

$$B_z\hat{z} = B_0\hat{z} + zG_z \tag{2.1}$$

$$B(z) = B_z\hat{z} \tag{2.2}$$

G_z é a intensidade do gradiente aplicado e é medido ao longo do eixo z em mT/m, B_0 é o valor inicial do campo magnético e $B(z)$ é o valor do campo magnético para um dado valor de z. O campo magnético gradiente ao longo de 2 metros de acordo com a equação 2.1 pode ser visto na Figura 4.

Inconsistências no campo podem ocorrer devido inomogeneidades, susceptibilidade magnética e mudanças químicas, mas essas inconsistências são na ordem de partes por milhão e podem ser removidas no processamento posterior (PRINCE; LINKS, 2006).

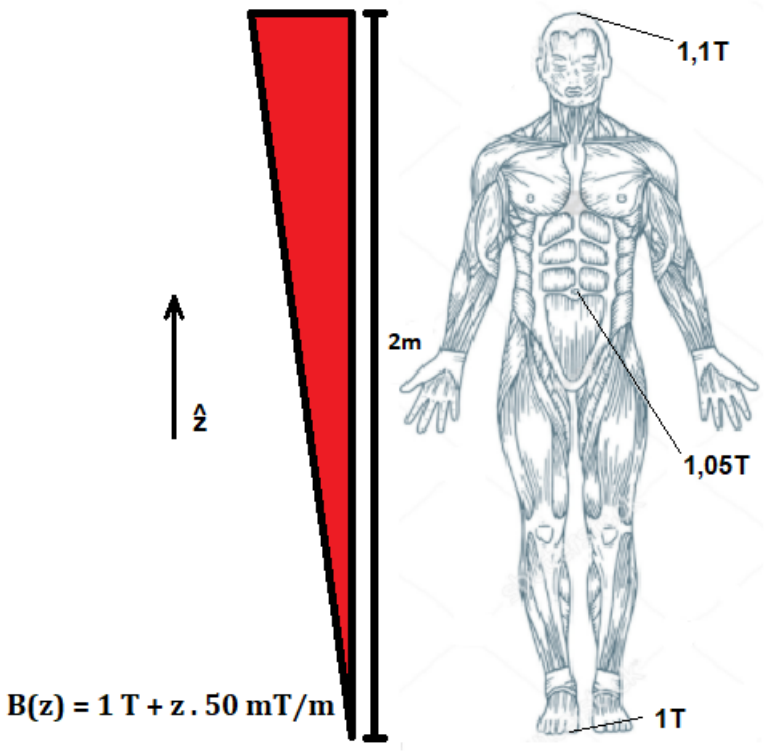

Figura 4 – Campo magnético gradiente.

Os átomos de hidrogênio assim como todos os átomos possuem um momento angular intrínseco chamado de spin que nos informa o sentido de giro átomo . O átomo de hidrogênio pode possuir os spin $\frac{+1}{2}$ que gira no sentido anti-horário ou $\frac{-1}{2}$ que gira no sentido horário (SHRIVER; ATKINS, 1999). Os núcleos que possuem um spin, também possuem um campo magnético. Quando não estão sobre efeito de campo magnético forte o suficiente, os spins dos átomos de hidrogênio no corpo humano tendem a se organizar de forma aleatória, porém quando aplicado um campo magnético forte como os usados em máquinas de ressonância (entre 0.5T e 3T)(PRINCE; LINKS, 2006) os átomos de hidrogênio tendem a realizar um movimento de precessão em torno do campo magnético imposto, chamada de Precessão de Larmor, tendendo a se alinhar ao campo magnético (TUBRIDY; MCKINSTRY, 2000).

O movimento de precessão em volta do campo magnético, devido a natureza do spin do átomo de hidrogênio, tende a ser paralelo, no mesmo sentido do campo magnético, ou antiparalelo, na mesma direção, porém em sentido oposto. Os átomos que se alinham paralelamente ao campo magnético imposto possuem menos energia do que os que se alinha antiparalelamente (LIBRETEXTS, 2019), e aproximadamente a mesma quantidade de átomos se alinham para cada sentido, porém a pequena quantidade de átomos que se alinham a mais no sentido paralelo, são o suficiente para produzir um sinal detectável nas próximas etapas.

> "Para um campo magnético de 1,5 T e na temperatura média do tecido humano, a diferença entre os spins que ocupam o estado de menor energia e o de maior energia é de aproximadamente 5 para 1 milhão. Do ponto de vista prático é somente com estes cinco spins resultantes que poderemos trabalhar para produzir sinal detectável na bobina."

(MAZZOLA, 2009)

2.4 Vetor Magnetização

Os átomos de hidrogênio que precessam em torno do eixo z que não se cancelaram, geram um vetor magnetização longitudinal M_0 resultante, composto pela combinação de vários momentos angulares de spins, que tendem a se alinhar na mesma direção e sentido com o campo magnético B(z). A magnitude de M_0 é dada pela "Equação 2.3".

$$M_0 = \frac{B(z)\gamma^2 h^2}{16\pi^2 kT} P_D \tag{2.3}$$

Na "Equação 2.3", k é a constante de Boltzmann, h a constante de Planck, T a temperatura em Kelvin, P_D a densidade de prótons, que é a quantidade de núcleos por unidade de volume. Na equação é possível observar que para um campo magnético $B(z)$ maior e para maior densidade de prótons, maior a magnitude de M_0. O valor de P_D irá alterar o contraste na imagem posteriormente (PRINCE; LINKS, 2006).

O campo magnético principal deve ser um gradiente, pois a intensidade do campo magnético é usado para selecionar o corte transversal do corpo. Quando um campo magnético B(t) é aplicado sobre o vetor magnetização M_0 este sofre um torque, a "Equação 2.4" descreve essa relação.

$$\frac{\mathrm{d}M}{\mathrm{d}t} = \gamma M(t) \times B(t) \tag{2.4}$$

Considerando agora o campo B(z) que é orientado na direção z, resolvendo a equação para cada uma das direções (x, y e z) e que o vetor magnetização M_0 esteja orientado com um ângulo α em relação ao eixo z e Φ seja um ângulo arbitrário, obtém-se as "Equações 2.5, 2.6 e 2.7".

$$M_x(t) = M_0 sin(\alpha) cos(-\gamma B(z)t + \Phi) \tag{2.5}$$

$$M_y(t) = M_0 sin(\alpha) sin(-\gamma B(z)t + \Phi) \tag{2.6}$$

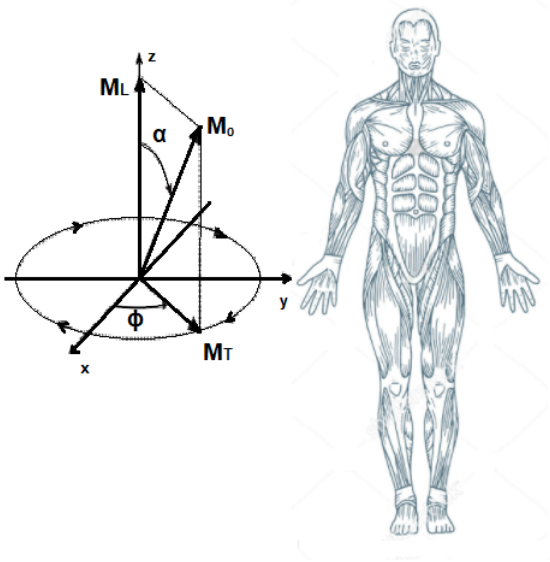

Figura 5 – Vetor magnetização M_0 precessando em torno do eixo z.(PRINCE; LINKS, 2006) modificada.

$$M_z(t) = M_0 cos(\alpha) \tag{2.7}$$

$$M_0 = |M(0)| \tag{2.8}$$

Essas equações descrevem o movimento de precessão do vetor magnetização M_0 em torno de B(z) como mostrado na Figura 5. A equação de Larmor mostra que para um determinado elemento, sobre o efeito de um determinado campo magnético B(z), irá realizar um movimento de precessão em determinada frequência angular ω (ou f em Hertz).

$$\omega = \gamma B(z) \tag{2.9}$$

$$f = \frac{\gamma}{2\pi} B(z) \tag{2.10}$$

O γ nas equações 2.9 e 2.10 representa a razão giromagnética do elemento observado, no caso do hidrogênio a razão giromagnética é [42.577 MHz/T] (MOHR; TAYLOR; NEWELL, 2008). Então para um campo magnético de 1T a frequência de precessão será [42.577 MHz]. Logo é possível observar que mudanças no campo magnético alteram diretamente a frequência de precessão. Assim para que os átomos de hidrogênio possuam frequências de precessão diferentes para diferentes partes do corpo é aplicado um campo magnético gradiente ao longo do corpo como mostrado na Figura 6.

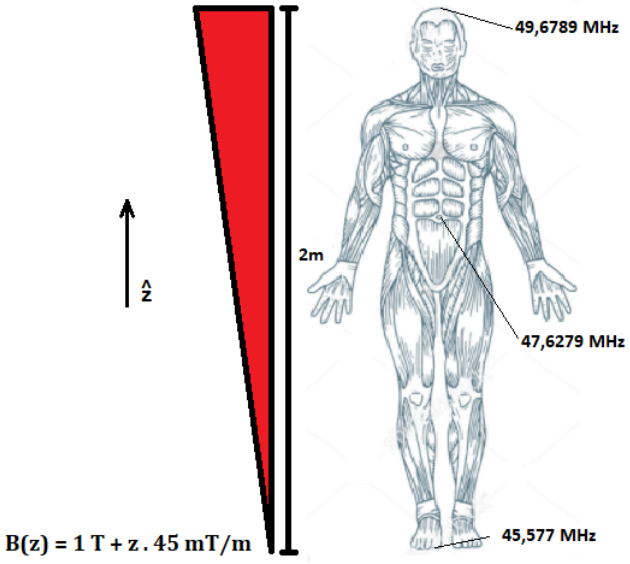

Figura 6 – Frequências de precessão ao longo do corpo para um campo magnético gradiente.

2.5 Pulso de RF

Um segundo campo magnético B_1 é aplicado perpendicular ao campo principal $B(z)$, o campo B_1 deve possuir a mesma frequência que a frequência de precessão desejada, pois a frequência desse sinal serve para selecionar o corte transversal do corpo desejado. Esse segundo pulso tem como objetivo desviar o vetor magnetização M_0, que antes estava orientado na direção z, em um ângulo α. Esse ângulo α costuma ser um ângulo de 90° pois assim transfere toda energia para o plano transversal fazendo com que a magnetização transversal chegue no seu valor máximo e o vetor magnetização M_0 passa induzir uma tensão em alguma das bobinas espalhadas localmente para captar esses sinais (MAZZOLA, 2009).

Desligando-se o campo B_1 o vetor magnetização M_0 tende a relaxar e voltar à posição original paralelo ao campo magnético $B(z)$ como mostrado na Figura 7.

A intensidade do vetor magnetização pode ser observada em função do tempo tanto no sentido longitudinal, na direção z, como também transversalmente ao desligar o campo magnético B_1. O tempo $T1$ é o tempo que a magnetização longitudinal demora para recuperar 63% da sua intensidade inicial. A função de relaxamento longitudinal em função de $T1$ é dada pela "Equação 2.11".

$$M_L(t) = M_0.(1 - e^{\frac{-t}{T1}}) + M_0.cos(\alpha).e^{\frac{-t}{T1}} \tag{2.11}$$

Na "Equação 2.11", $M_L(t)$ é a intensidade do vetor magnetização longitudinal, M_0 é a intensidade inicial do vetor magnetização, α é o ângulo do desvio, t é o tempo e $T1$

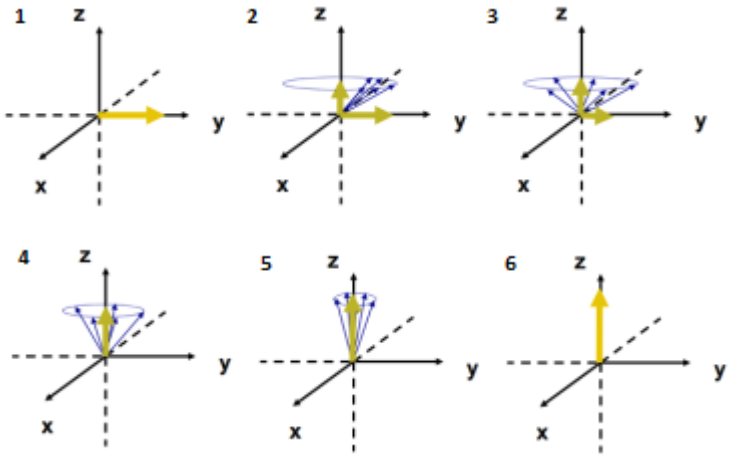

Figura 7 – Vetor Magnetização M_0 retornando à posição inicial. (MAZZOLA, 2009)

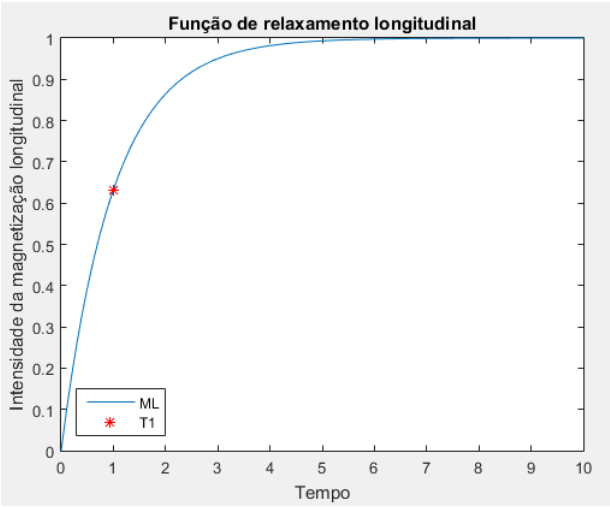

Figura 8 – Retorno da magnetização longitudinal.

é a constante de relaxamento longitudinal que muda de acordo com o tecido do corpo. A Figura 8 mostra a curva dessa equação, a intensidade da magnetização longitudinal tende a intensidade inicial de M_0. M_0 vale 1 neste gráfico. A magnetização transversal é composta pelas projeções em x e em y do vetor magnetização M_0, e pode ser expressa de acordo com a "Equação 2.12".

$$M_T(t) = M_x(t) + jM_y(t) \tag{2.12}$$

Combinando as "Equações 2.5, 2.6 e 2.12", é possível chegar em outra forma de calcular a magnetização transversal (Equação 2.13).

$$M_T(t) = M_0.sin(\alpha).e^{-j(\gamma B(z)t+\Phi)} \tag{2.13}$$

Também é possível calcular a função de relaxamento transversal em função de $T2$

através da "Equação 2.14", o tempo $T2$ é o tempo necessário para a intensidade do campo magnético transversal atingir 37% do valor de intensidade longitudinal inicial (M_0).

$$M_T(t) = M_0.sin(\alpha).e^{-j(\gamma B(z)t+\Phi)}.e^{\frac{-t}{T2}} \tag{2.14}$$

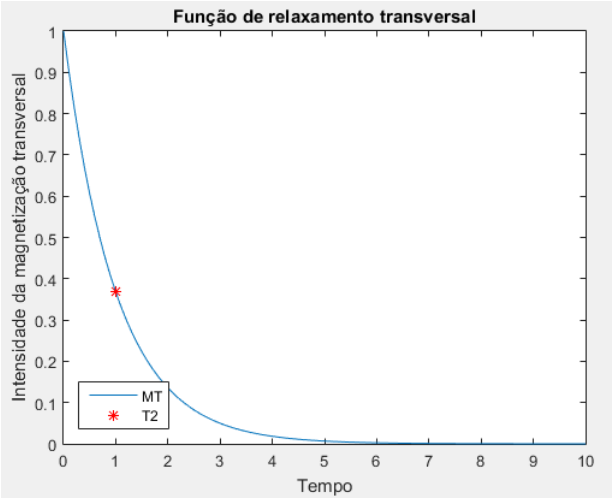

Figura 9 – Decaimento da magnetização transversal.

Na "Equação 2.14", $M_T(t)$ é a intensidade do vetor magnetização transversal, M_0 é a intensidade do vetor magnetização, α é o ângulo do desvio, t é o tempo e $T2$ é a constante de relaxamento transversal, que muda de acordo com o tecido do corpo. A Figura 9 mostra a curva da função de relaxamento transversal, a intensidade transversal se afasta da intensidade longitudinal inicial M_0. M_0 vale 1 no gráfico.

A “Equação 2.14” está expressa em radianos, e pode ser expressa em ciclos por segundo, aplicando a “Equação 2.10”, obtém-se a “Equação 2.15”.

$$M_T(t) = M_0.sin(\alpha).e^{-j(2\pi ft+\Phi)}.e^{\frac{-t}{T2}} \tag{2.15}$$

A magnetização transversal ainda pode ser escrita levando em consideração o seu aspecto antes da excitação pelo pulso de RF ($M(0^-)$) e logo após o pulso de RF ($M(0^+)$), como mostram as “Equações 2.16 e 2.17”.

$$M_T(t) = M_z(0^-).sin(\alpha).e^{-j(2\pi ft+\Phi)}.e^{\frac{-t}{T2}} \tag{2.16}$$

$$M_T(t) = M_T(0^+).e^{-j(2\pi ft+\Phi)}.e^{\frac{-t}{T2}} \tag{2.17}$$

Variações no campo magnético B(z) podem ocorrer devido a diferenças de composição dos tecidos do corpo humano, problemas na fabricação ou calibragem do magneto ou

até mesmo o gradiente usado para seleção de corte. Essas perturbações fazem com que o sinal recebido decaia mais rápido do que o T2 esperado. Esse sinal recebido é chamado de $T2^*$, a relação observada entre $T2$ e $T2^*$ é demonstrado pela "Equação 2.18", seu cálculo é feito através da combinação do relaxamento natural do núcleo de hidrogênio ao desligar o sinal de magnetização transversal, com o relaxamento causado por essas inconsistências no campo magnético (Equação 2.19).

$$\frac{1}{T2^*} = \frac{1}{T2} + \frac{1}{T2'} \tag{2.18}$$

$$T2' = \gamma B_{variação} \tag{2.19}$$

Onde $T2'$ é o relaxamento causado pelas variações no campo magnético e $T2$ é o tempo de relaxamento transversal natural. Devido ao fato de $T2^*$ ser a combinação de dois relaxamentos diferentes, ele sempre irá decair mais rápido, dessa forma $T2^* < T2$.

Os tempos $T1$ e $T2$ variam para diferentes tecidos do corpo humano e isso serve para identificar esses tecidos (Tabela 1).

Tabela 1 – Tempos $T1$ e $T2$ para um campo magnético de $1.5T$. (STANISZ et al., 2005)

Tecido	**T1** $[ms]$	**T2** $[ms]$
Sangue	1441 ± 120	290 ± 30
Nervo Óptico	815 ± 30	77 ± 9
Coração	1030 ± 34	40 ± 6
Rim	690 ± 30	55 ± 3
Matéria Cinzenta	1124 ± 50	95 ± 8
Matéria Branca	884± 50	72 ± 4

Essas diferenças entre os tempos $T2$ criam tons de cinza diferentes na imagem gerada após todo o processo, como será mostrado.

2.6 Sequência de Pulsos

O sinal obtido vindos dos spins dos átomos de hidrogênio se dá devido ao efeito de eco dos spins (HAHN, 1950), onde após aplicar o pulso que defasa os spins em 90° é aplicado um segundo pulso mais rápido após t segundos do primeiro, que defasa os os átomos de hidrogênios em 180°, é possível observar que surgem dois sinais, o primeiro devido o relaxamento dos spins voltando a precessar em volta do campo $B(z)$, mas também é possível observar um segundo sinal que é um eco desse relaxamento a exatos $2t$ segundos após a emissão do primeiro pulso do campo magnético B_1 como mostrado na Figura 10 e também na animação do Anexo A, Figura 27.

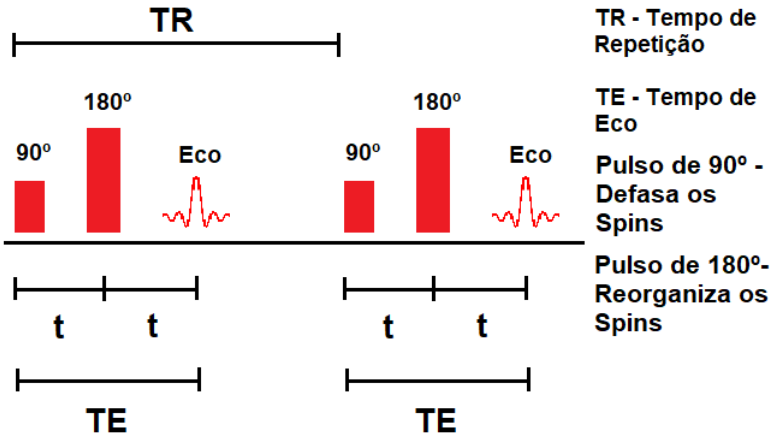

Figura 10 – Sequência de pulsos e eco observado.

Durante o relaxamento transversal, após o pulso de 90°, os spins tendem a perder a fase entre si se espalhando devido a pequenas diferenças locais no campo magnético, o pulso de 180° reorganiza esses spins fazendo com que eles entrem em fase novamente durante um curto período de tempo gerando o eco que é observado. Os spins ficam defasados até que o pulso de 180° seja feito t segundos depois do pulso de 90°, e levam o mesmo tempo t para que entrem em fase novamente e gerem o eco (BREWER; HAHN, 1984). Após gerarem o eco os spins voltam a defasar, é possível gerar um segundo eco com um segundo pulso de 180°, a cada eco gerado após o primeiro perde-se amplitude, devido a perda de intensidade do campo magnético transversal, devido o relaxamento dos spins. O tempo entre o primeiro pulso e a aparição do eco é chamado de TE (Tempo de Eco) e o tempo entre um pulso de 90° e outro é chamado de TR (Tempo de repetição).

O segundo pulso não necessariamente precisa ser de 180°, o eco é gerado após dois pulsos de frequência específica serem gerados (BREWER; HAHN, 1984).

Além do gradiente de seleção de corte para selecionar o corte transversal do qual se deseja a imagem, é necessário mais dois sinais que informem de qual parte do corte está vindo o sinal captado. Considerando o corte transversal como uma matriz 2D, é necessário duas coordenadas para localizar qualquer valor dessa matriz, dessa forma, as coordenadas utilizam valores de frequência e fase dos spins. São usados gradientes de frequência e de fase, dessa forma, cada spin nessa matriz possuirá valores diferentes que podem ser utilizados para identificar a origem do sinal.

Para a codificação em frequência o funcionamento é semelhante com o usado para selecionar o corte, é aplicado um campo magnético gradiente paralelo às linhas da matriz, dessa forma todos os átomos em linhas diferentes vão possuir uma banda de frequências estreita e diferente dos outros átomos na mesma linha como mostrado na Figura 11. Substituindo o campo magnético gradiente calculado pela Equação 2.1 no cálculo da

frequência na Equação 2.2, porém considerando que o campo está sendo aplicado na direção x agora, podemos observar a diferença de frequência de acordo com o gradiente de campo magnético.

$$f(x) = \frac{\gamma}{2\pi}(B(z) + G_x x) \tag{2.20}$$

$$f(x) = f_0 + x f_x \tag{2.21}$$

Em que f_0 é uma frequência inicial específica, e f_x é a frequência do gradiente aplicado, e por fim $f(x)$, é a frequência em função do eixo x.

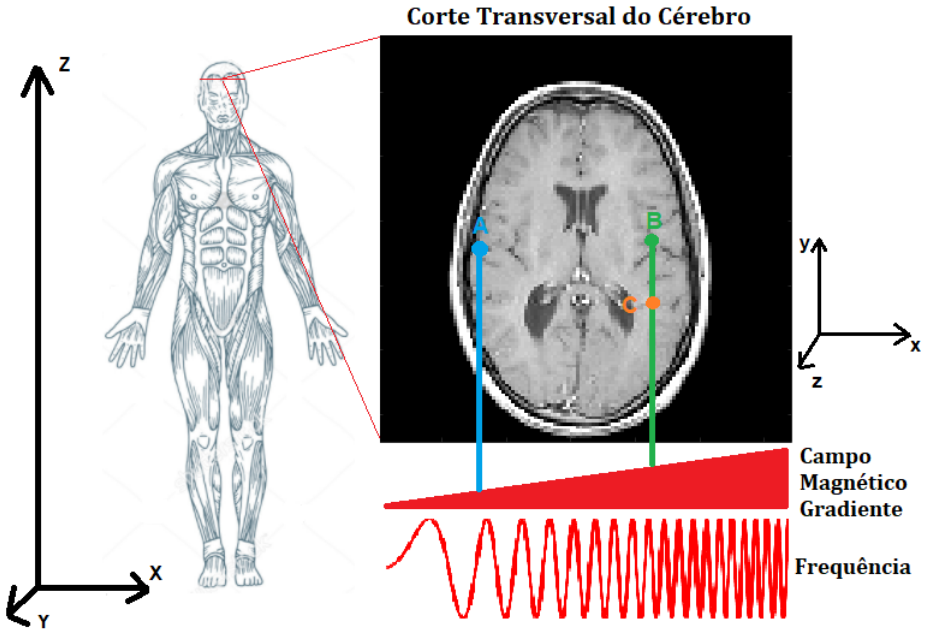

Figura 11 – Gradiente de Codificação em Frequência.(MATHWORKS, 2020a) modificado.

Codificando as linhas da matriz em frequência faz com que seja possível diferenciar pontos diferentes posicionados no eixo x, como por exemplo na Figura 11, os pontos A e B possuem frequências distintas, porém os ponto B e C por estarem na mesma posição em relação ao eixo horizontal, não possuem diferença em frequência, por isso é usado a codificação em fase para diferenciar pontos na vertical.

Antes da coleta do sinal de ressonância é aplicado um gradiente na direção y (Gy), durante uma duração de tempo τ, quando esse campo é desligado o sinal gerados pelos spins é coletado pela bobina local. O gradiente aplicado não causa efeito nenhum na frequência na hora da coleta do sinal, pois ele é desligado antes dessa coleta, porém durante a aplicação do gradiente, esses spins precessam em frequências diferentes, o que causa um atraso de fase que depende do tempo que a aplicação desse gradiente durou.

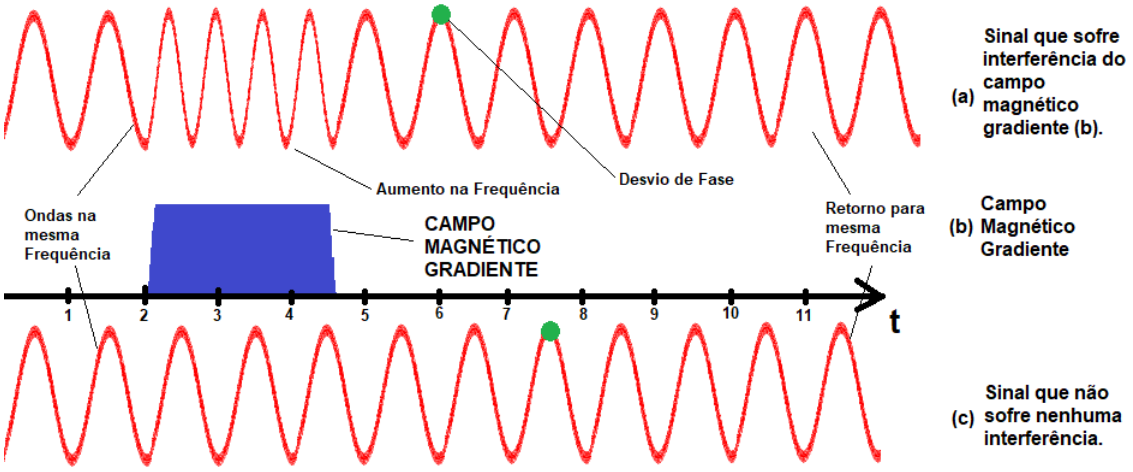

Figura 12 – Desvio de Fase.

A Figura 12 mostra um exemplo de como é realizado o atraso de fase, temos dois sinais eletromagnéticos (a) e (c) captados pelas bobinas de corpo, que são inicialmente iguais em fase e em frequência. Em t = 2 é ligado um campo magnético gradiente (b) que afeta apenas o sinal (a), em t = 4,5 o gradiente é desligado. Enquanto o gradiente está ligado, o sinal (a) tem um aumento da sua frequência, após o desligamento a sua frequência volta ao estado anterior, igual ao sinal (c). Apesar de possuírem frequências iguais, os sinais agora estão defasados, para melhor observar essa defasagem foi marcado com um ponto verde o oitavo pico de cada sinal.

$$\Delta\varphi = \tau\Delta\omega_0(y) \tag{2.22}$$

A "Equação 2.22" mostra que o atraso de fase causado depende do tempo τ no qual o gradiente ficou acionado, e da diferença de de frequência ($\Delta\omega_0$) causado pelo gradiente ao longo do eixo y. Resolvendo essa equação utilizando a "Equação 2.1" para calcular o gradiente aplicado, considerando que o campo magnético inicial é 0, e a "Equação 2.9" para o cálculo da frequência de precessão, obtemos a "Equação 2.23".

$$\varphi(t) = -\tau\gamma G_y y \tag{2.23}$$

Com a "Equação 2.23" é possível observar que o desvio de fase depende da posição do spin em relação ao eixo y, da intensidade do gradiente G_y e da duração da aplicação do gradiente τ. O sinal é então adquirido mantendo a fase constante, e preenchendo todos os valores de uma mesma frequência na matriz, assim é preenchida uma linha, para mudar para preencher a próxima linha, é causado o atraso de fase, dessa forma uma outra linha é preenchida, alterando a frequência e a fase sucessivamente.

2.7 Reconstrução da Imagem

A matriz preenchida com a codificação de fase e frequência é chamada de Espaço K e contém os sinais obtidos pelas bobinas locais que ficam em volta de todo o corpo analisado, e por isso recebe sinais de todo o corte selecionado. Para um Espaço K de 128x256, é necessário fazer 128 mudanças de fase, a duração do gradiente de frequência é em torno de 10 ms, porém entre uma mudança e outra no gradiente de fase é esperado em torno de 1s, então para preencher todo o espaço, é levado aproximadamente 128 segundos. Cada coluna do Espaço K gera uma projeção no eixo x, essa projeção é um sinal que possui a contribuição de cada spin na coluna, porém possuem contribuições diferentes devido a uma mudança de fase entre eles, as colunas diferem entre si devido a diferença de frequência que codifica o eixo x, para conseguir diferenciar os sinais de N spins individualmente, considerando que cada spin é uma variável de uma equação, são necessárias N equações. Dessa forma quanto mais codificações de fase são realizadas maior é a qualidade final da imagem.

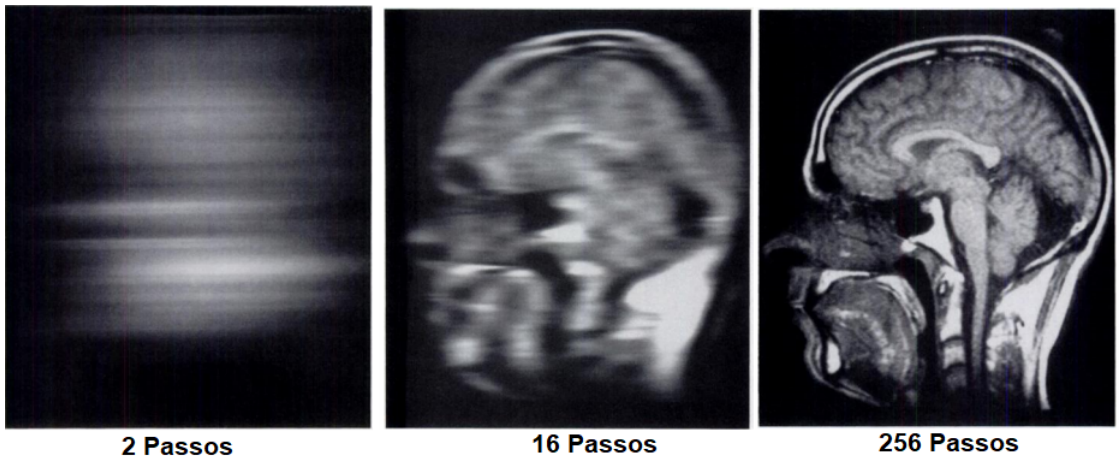

Figura 13 – Diferença na qualidade da imagem devido ao número de passos de codificação de fase realizados. (FELMLEE et al., 1989) modificado.

A Figura 13 mostra a diferença na qualidade da imagem de acordo com o número de codificações de fase (FELMLEE et al., 1989).

O Espaço K pode ser preenchido de várias formas, não necessariamente linha por linha, algumas formas de se preencher tem sua aplicação específica, como por exemplo a radial, que permite a movimentação do paciente, enquanto outras são devido a maior velocidade de preenchimento, como por exemplo a eco planar.

A Figura 14 mostra os diferentes tipos de preenchimento do espaço K, indicado pela letra (a), os tipos indicados são: (b) Cartesiano; (c) Radial; (d) Eco Planar. O preenchimento eco planar economiza tempo pois é necessário apenas um pulso, ou um pequeno número de pulsos para preencher todo o Espaço K, como mostrado na Figura 15, essa

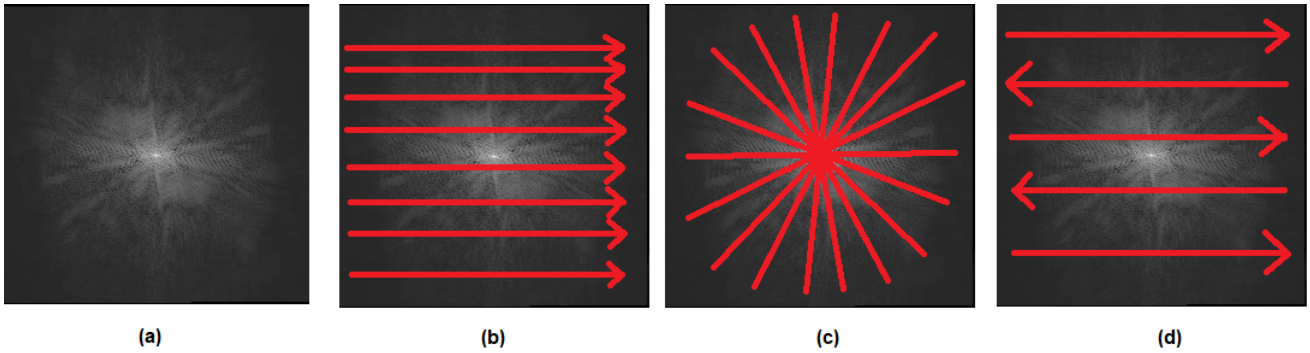

Figura 14 – Diferentes tipos de preenchimento do Espaço K.

aquisição rápida permite obter imagens de órgãos em constante movimento, também faz com que seja menos suscetível a vibrações mecânicas, e movimentações do paciente (THURSTON, 2019) tornando-se uma das técnicas mais utilizadas para o preenchimento do Espaço K.

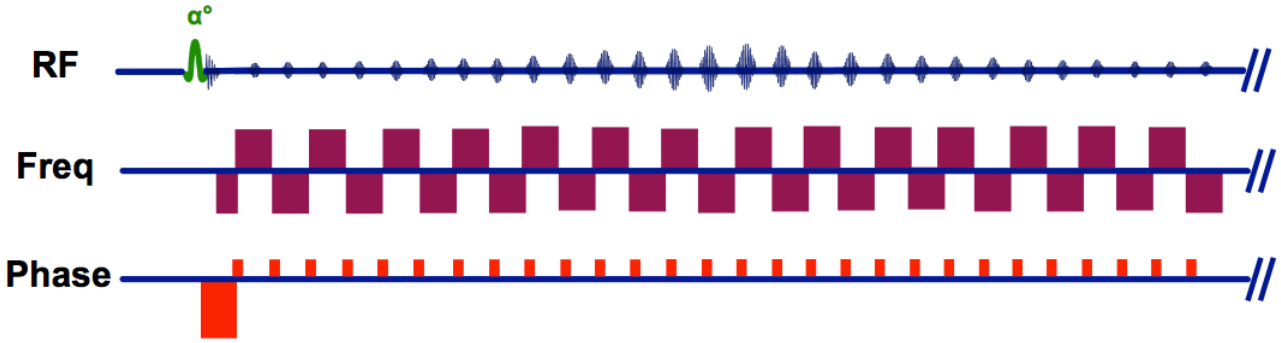

Figura 15 – Sequência de pulsos para o preenchimento Eco Planar. (ELSTER, 2019)

Considerando que o sinal emitido por uma determinada parte (x,y) do tecido possui o sinal de relaxamento transversal de todos os spins naquele espaço o sinal detectado é proporcional ao número de spins nessa localização. As bobinas recebem a contribuição de todos as localizações e o sinal obtido por essas bobinas é descrito pela "Equação 2.24" onde A é a área do corte transversal.

$$s(t) = \int\int A.M(x,y;0^+)e^{-j2\pi ft}.e^{\frac{-t}{T2(x,y)}}dxdy \tag{2.24}$$

Considerando que a densidade de spins efetiva é dada pela "Equação 2.25".

$$f(x,y) = A.M(x,y;0^+).e^{\frac{-t}{T2(x,y)}} \tag{2.25}$$

A "Equação 2.24" pode ser reescrita da seguinte forma:

$$s(t) = e^{-j2\pi ft}\int\int f(x,y)dxdy \tag{2.26}$$

Substituindo a frequência codificada espacialmente da “Equação 2.20” na “Equação 2.26”, obtém-se a “Equação 2.27” que descreve o sinal s(t) recebido nas bobinas em função da codificação de frequência e da densidade de spins.

$$s(t) = \int\int f(x,y)e^{-j\gamma G_x xt}dxdy \tag{2.27}$$

Resolvendo as equações anteriores para também levar em consideração a codificação de fase da “Equação 2.23”, obtém-se a “Equação 2.28”.

$$s(t) = \int\int f(x,y)e^{-j\gamma G_x xt}e^{-j\gamma G_y y\tau}dxdy \tag{2.28}$$

Substituindo os termos $\gamma G_x t$ e $\gamma G_y \tau$ por k_x e k_y respectivamente, obtemos a "Equação 2.29"(ROSEN; WALD,).

$$s(t) = \int\int f(x,y)e^{-jk_x x}e^{-jk_y y}dxdy \tag{2.29}$$

Isolando f(x,y), que representa a imagem do corte transversal, é possível observar que este é dado pela transformada inversa de Fourier de duas dimensões do sinal s(t), coletado pelas bobinas:

$$f(x,y) = \int\int s(k_x,k_y)e^{jk_x x}e^{jk_y y}dk_x dk_y \tag{2.30}$$

Por fim é obtida a matriz f(x,y) com a imagem do corte transversal selecionado, a velocidade da reconstrução da imagem de ressonância depende de dois fatores, da velocidade de aquisição dos dados e da velocidade de processamento de imagem. Métodos que aceleram esse processo são muito estudados para aumentar o número de exames realizados e dessa forma otimizar o uso do equipamento.

2.8 Considerações Finais

A reconstrução de uma imagem de ressonância é um processo complexo feito por computadores de forma extremamente precisa e eficiente. Algoritmos que realizam esse processo utilizam de transformadas inversas rápidas (IFFT) que são mais fáceis de ser executadas por um computador por se tratar de uma sequência de somas e subtrações. A transformada inversa rápida é uma operação que também pode ser realizada por uma FPGA e o processo de reconstrução utilizando a IFFT será explicado nas próximas seções.

3 Metodologia

3.1 Considerações Iniciais

Nesta seção será abordado o procedimento realizado para a criação do algoritmo de reconstrução de imagens de ressonância magnética. Serão apresentados as ferramentas utilizadas e as etapas realizadas. Também serão apresentados alguns fluxogramas e imagens que auxiliam no entendimento de como tudo foi feito, para então apresentar os resultados na próxima seção.

3.2 Implementação

Para verificar se a reconstrução de imagens de ressonância seria viável se realizada em hardware foi feito uma simulação do processamento realizado por uma FPGA, utilizando a ferramenta de design ISE 14.7 e a escolha do modelo foi a placa XC6SLX45 da família Spartan 6. A reconstrução da imagem foi realizada utilizando os LogiCORE da Xilinx, que permitem a criação de blocos de memória e de transformadas de Fourier com maior facilidade.

As imagens utilizadas para os testes foram imagens de exames de ressonância e imagens normais que passaram por transformadas duplas de Fourier utilizando o MATLAB e então são gravadas em dois documentos .coe, parte real e parte imaginária, que são lidos pelo algoritmo em VHDL. Os detalhes do código em MATLAB que salvam os arquivos (imagetocoe.m) e todos os outros códigos usados no projeto podem ser vistos no repositório do Github no Anexo B, Figura 28, e também nos Anexos E, F e G. O processo completo de reconstrução da imagem pode ser observado no Anexo C, Figura 29. O funcionamento da Test Bench e os detalhes das configurações dos LogiCORE serão vistos nas seções seguintes.

Os espaços K foram criados dessa forma pois tentou-se obter espaços K saídos de máquinas de ressonância, porém não foi possível salvar os dados obtidos de bancos de dados online em arquivos .coe. Decidiu-se então utilizar a transformada dupla de imagens durante as simulações que por conceito seriam iguais aos espaços K. Uma das vantagens de poder utilizar qualquer imagem, foi poder testar a precisão do algoritmo com os mais diversos tipos de imagem.

3.3 Processamento

As etapas da reconstrução da imagem podem ser vistos abaixo e na Figura 16. Essas etapas serão referenciadas ao longo deste documento.

1. A imagem armazenada em duas memórias ROM (parte real e parte imaginária) são carregadas para o núcleo que realiza a primeira transformada;
2. A primeira transformada inversa (IFFT) é realizada;
3. O resultado da primeira IFFT é enviado para a memória RAM e é realizada a transposição desta matriz invertendo os valores dos endereços de linha por coluna, o resultado é carregado no núcleo que realizará a segunda transformada;
4. A segunda transformada inversa (IFFT) é realizada;
5. O resultado da segunda IFFT é enviado para a memória RAM e é realizada a transposição desta matriz invertendo os valores dos endereços de linha por coluna;
6. É realizada a soma do módulo da parte real com o módulo da parte imaginária;
7. É finalizado o processo de reconstrução.

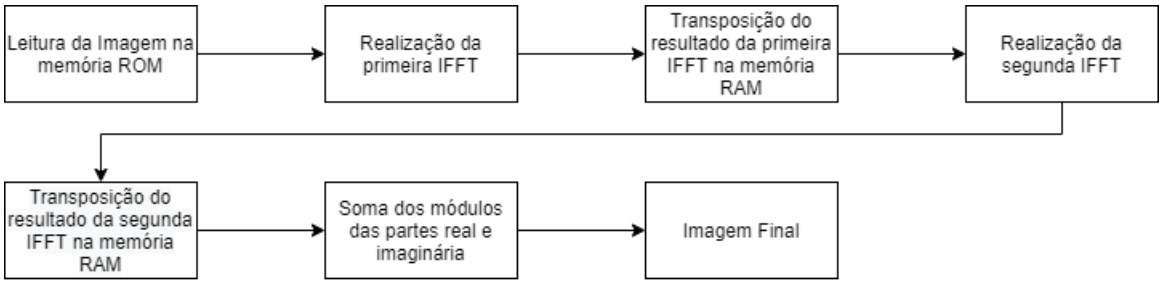

Figura 16 – Etapas da reconstrução.

A imagem reconstruída é salva em um arquivo de texto contendo 16384 valores, um para cada ponto de uma imagem de 128x128 e a visualização dessa imagem foi feita através do MATLAB, o código do MATLAB usado para fazer a visualização pode ser visto no repositório do Github (VisualizarImagem.m), os códigos feitos em MATLAB não realizam nenhuma reconstrução ou operação na imagem, são apenas utilizados para preparar os dados para a reconstrução e para visualizar esses dados após a conclusão do processo.

O resultado foi comparado com a imagem original usada para criar a imagem de teste para chegar a conclusão se o processo de reconstrução foi bem sucedido ou não. Os testes inicialmente produziram imagens que se assemelhavam porém não era um resultado

satisfatório por se tratar de imagens muito escuras. O algoritmo foi corrigido e o resultado final foi satisfatório como será visto na próxima seção.

3.4 Algorítmo

Durante o desenvolvimento da TestBench foram utilizados alguns LogiCORES, suas aplicações e explicações podem ser observadas a seguir.

3.4.1 FFTs (Transformadas Rápidas de Fourier)

Para a realização das transformadas rápidas foi utilizado o LogiCORE IP Fast Fourier Transform v7.1 da Xilinx, que implementa o método Cooley-Tukey de transformada rápida de Fourier. O tipo de implementação escolhida foi a Pipelined por processar os dados de forma constante executando Radix-2 Butterfly conforme os dados são recebidos pelo núcleo como mostrado na Figura 17 (XILINX, 2011).

O LogiCORE de FFT foi utilizado pois foi uma ferramenta criada pela Xilinx para facilitar o processo de implementação de uma transformada rápida de Fourier, ela permite implementar a transformada com uma interface simples e gera resultados precisos.

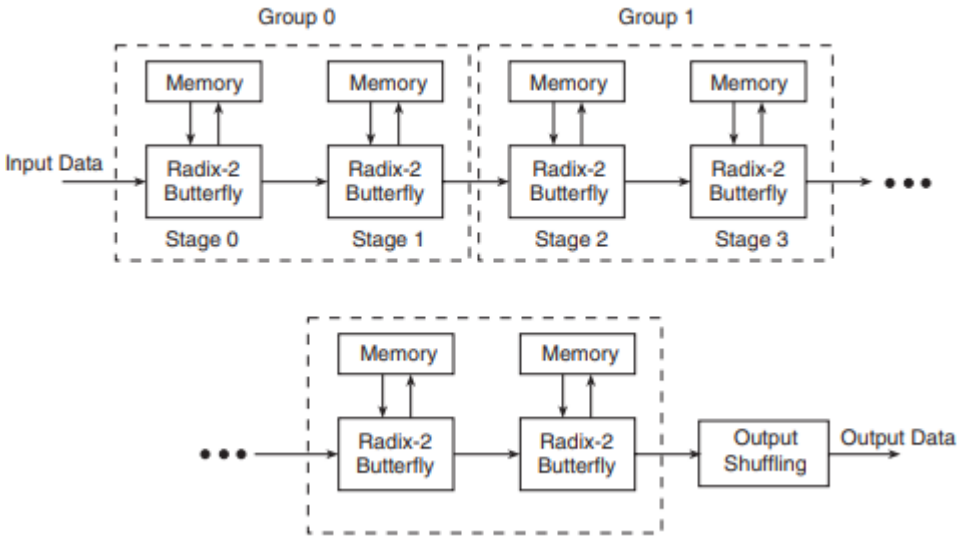

Figura 17 – Funcionamento do modelo Pipelined. (XILINX, 2011).

A transformada é calculada conforme os dados são lidos, isso permite tanto um processamento mais rápido quanto processamento constante em caso de imagens sendo obtidas em tempo real. O núcleo aceita entradas de 32 bits em ponto fixo e disponibiliza a saída em ordem natural. Uma das vantagens do núcleo é controlar as constantes que são multiplicadas dentro de cada soma realizada pela FFT em cada um de seus estágios (scale_sch) podendo assim evitar erros ao se realizar duas transformadas em sequência.

A transformada realizada é de tamanho 128 assim cada uma das linhas das entradas é transformada por vez e o resultado é disponibilizado em ordem natural, custando um pouco mais do tempo de processamento porém sendo necessário para visualizar os resultados. O núcleo realiza a transformada em uma frequência máxima de 395 MHz e para as simulações foi utilizada uma frequência de 100 MHz. O núcleo foi configurado para utilizar o arredondamento convergent rounding para evitar algum erro que o arredondamento truncado pudesse gerar. Foi incluída uma variável que reinicia o núcleo ao final da transformada (sclr), para assim deixá-lo pronto para uma nova sequência de dados. (XILINX, 2011).

Por só fazer transformadas em uma direção, devido os dados entrarem de forma serial, a transformada é realizada verticalmente. Para conseguir realizar a transformada nas duas direções, as imagens após a primeira transformada passam por uma transposição e então passam pela segunda transformada, dessa forma é realizada a operação em todas as linhas e colunas como mostrado na Figura 18.

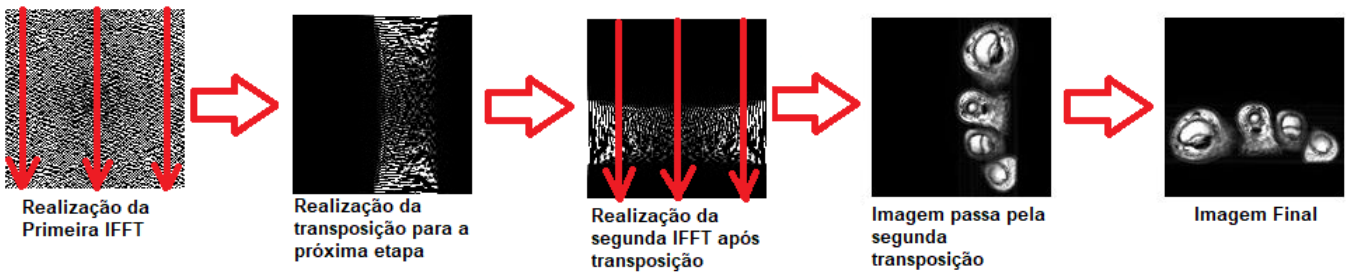

Figura 18 – Etapas da reconstrução.

Foram utilizados dois núcleos diferentes pois para imagens em que o comprimento é diferente da largura o mesmo núcleo não poderia ser reaproveitado para realizar as duas transformadas, pois o comprimento da transformada é fixo, não podendo ser alterado durante a execução do algoritmo. Na figura 19 é possível observar a tela de configuração do núcleo FFT.

3.4.2 Memória ROM

As memórias ROM foram geradas utilizando o LogiCORE IP Block Memory Generator v7.3 da Xilinx, utilizou-se o modo single port pois seria utilizado apenas uma saída de leitura dos dados. A Read Width deve ser configurada para ser do tamanho do número de bits de cada pixel da imagem armazenada, no caso 32 bits, e a Read Depth ajustada para a quantidade de pixels na imagem, para uma imagem de 128x128 são 16384 (XILINX, 2012).

São utilizadas duas memórias ROM para armazenar a imagem, uma para a parte real e outra para a parte imaginária, as imagens são armazenadas em formato .coe (Figura

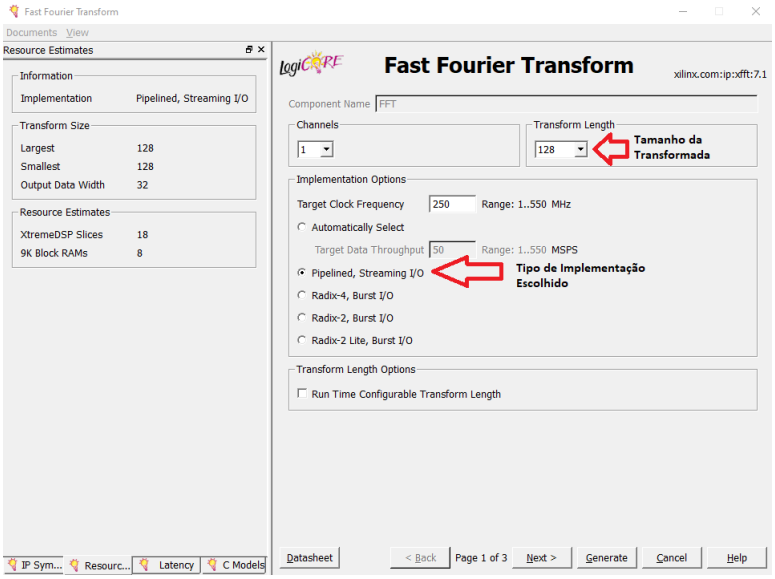

Figura 19 – Tela de configuração do LogiCore de FFT.

20) para serem lidas pelos núcleos e dentro de cada um dos arquivos cada pixel da imagem é representado por um valor decimal, sendo assim, cada arquivo possui 16384 valores representando cada uma das partes da imagem. Cada um desses valores é transformado em uma variável binária de 32 bits sendo o primeiro bit o bit de sinal, dessa forma a amplitude máxima dos bits é de 2.147.483.647.

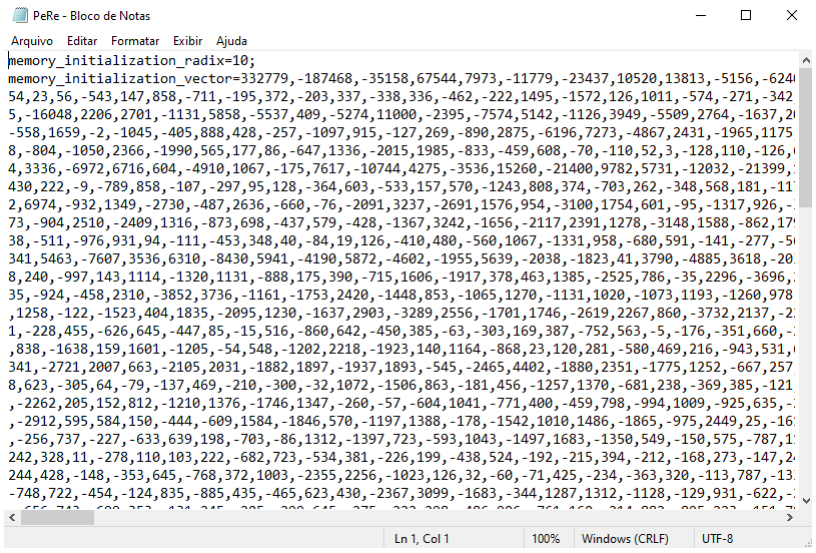

Figura 20 – Arquivo que armazena a parte real da imagem.

Os valores mostrados na Figura 20 são a parte real da transformada dupla de Fourier da imagem original, simulando o espaço-K, quanto maior o valor, mais clara a representação do pixel na parte real do espaço-K como pode ser visto na Figura 21.

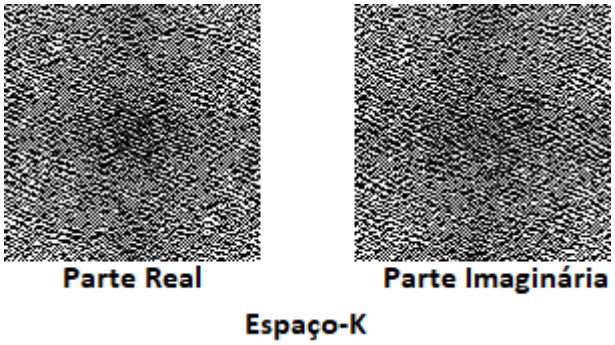

Figura 21 – Espaço-K Simulado.

3.4.3 Memória RAM

Assim como as memórias ROM, as RAM foram geradas utilizando o LogiCORE IP Block Memory Generator v7.3 da Xilinx, as configurações são as mesmas da memória ROM, single port com Read Width de 32 bits e Read Depth de 16384 no modo No Change, que não permite que a saída mude enquanto a memória está sendo escrita, para impedir que ocorra algum erro na transposição (XILINX, 2012).

As memórias RAM são utilizadas para armazenar temporariamente os valores da matriz durante as transposições, ela aceita até 16384 valores de 32 bits, após cada transformada o resultado é armazenado nas memórias RAM e cada pixel da imagem possui um endereço, dessa forma para realizar a transposição basta inverter os valores de endereço da matriz, na última etapa da reconstrução também é realizada a soma do módulo das partes reais e imaginárias dos valores vindos da última transposição, realizada nas memórias RAM. O uso das memórias ao longo do processo podem ser vistos na Figura 22.

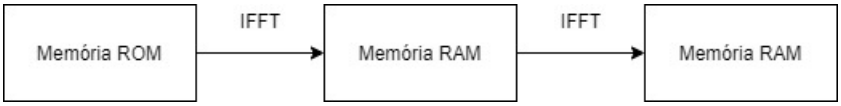

Figura 22 – Armazenamento dos dados ao longo da reconstrução.

3.5 Considerações Finais

Nesta seção foi abordada a forma que o algoritmo trata os dados e como eles são processados. Na próxima seção serão mostradas as etapas até se obter os resultados finais e comparação desses resultados com os esperados para exames de ressonância em questão de qualidade, velocidade e precisão no tempo.

4 Resultados

4.1 Considerações Iniciais

Nesta seção serão apresentados os resultados finais, procedimentos para refinar estes resultados, comparações e cálculos realizados que validam a eficiência do algoritmo. Ao final da sessão será apresentado uma fórmula que permite dimensionar o sistema de acordo com a taxa de imagens por segundo, a frequência e as dimensões da imagem.

O algoritmo foi testado com várias imagens diferentes não apenas imagens de ressonância, para poder observar a qualidade das imagens reconstruídas e também se o tempo levado seria diferente de acordo com a complexidade da imagem. Foram realizados testes com matrizes 8x8 até que o código estivesse realizando corretamente as transformadas, o código para as matrizes 8x8 pode ser encontrado no repositório do projeto no Github (FFTIPCore71_8.vhd), após acertar a transformada, foram iniciados testes com imagens de 128x128, o tamanho foi escolhido por permitir visualizar uma imagem nítida e não tomando muito tempo de simulação, pois cada simulação de reconstrução de matriz 128x128 já levava em torno de 40 minutos, não retratando o tempo real que tomaria para reconstruir a imagem. O código final usado para obter os resultados contidos nesta seção pode ser encontrado no repositório do Github (FFTIPCore71_128.vhd).

4.2 Qualidade da Imagem

Os primeiros testes com imagens de 128x128 foram realizados utilizando vetores de 16 bits, porém não resultaram em imagens satisfatórias pois essas eram muito escuras, as imagens quando transformadas no MATLAB tinham que ser alteradas para diminuir os valores maiores do que 32767 para este valor, pois era o máximo que um vetor de 16 bits com sinal suporta. Os resultados obtidos com esses testes, e sem a soma dos módulos das partes reais e imaginárias podem ser vistos na Figura 23. A imagem original possui dimensões de 128x128 e é a imagem das pontas dos dedos do pé, foi escolhida por ser mais simples e sem muitos detalhes o que facilita a visualização de erros nos testes iniciais.

Foi aumentado o valor de cada pixel da matriz para 32 bits sinalizados e, apesar da qualidade da imagem ter aumentando nas extremidades, o centro continuava escuro. Analisando cada estágio do algoritmo, observou-se que parte da imagem reconstruída estava na parte imaginária da segunda transformada, a imagem obtida anteriormente é uma visualização apenas da parte real que deveria possuir toda a imagem, decidiu-se então fazer a soma dos módulos das partes real e imaginária, e o resultado dessa operação pode

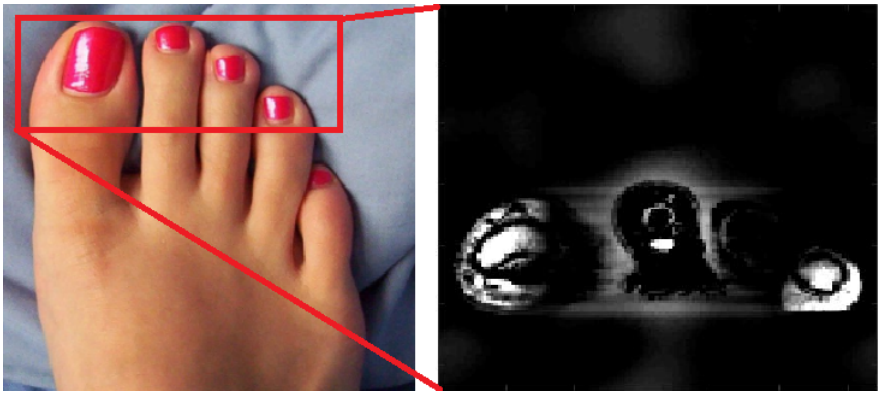

Figura 23 – Resultado obtido com imagem contendo apenas a parte real da transformada.

ser visto na Figura 24.

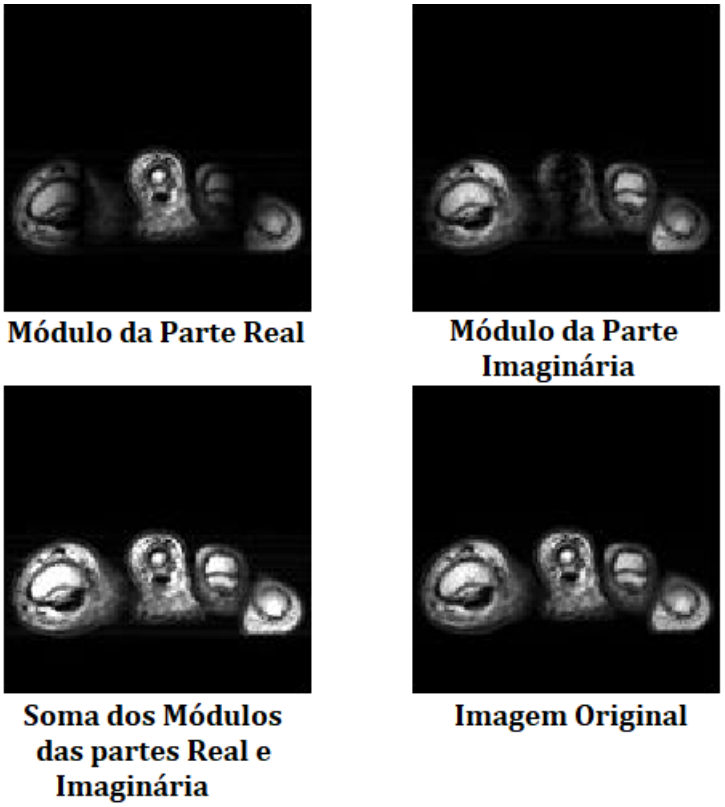

Figura 24 – Etapas para a obtenção da imagem final.

Outras imagens reconstruídas com o algoritmo, de exames de ressonância e de outros tipos podem ser observadas no Anexo D, Figura 30 sendo comparadas com suas versões originais, e com seu coeficiente de semelhança.

Para checar a diferença da imagem original e a reconstruída com o código em VHDL, foi utilizado a função do MATLAB ssim(A,ref), que mostra o coeficiente de similaridade estrutural entre uma imagem de teste (A) e uma imagem de referência (ref). Com essa função também é possível obter uma imagem que destaca a diferença entre as

imagens, que pode ser observada na Figura 25, as partes claras da imagem consistem em partes que a imagem reconstruída é mais semelhante a imagem original, e as partes mais escuras mostram os locais onde são diferentes. A figura foi colocada dentro de um fundo azul para melhor visualização da parte branca (MATHWORKS, 2020b).

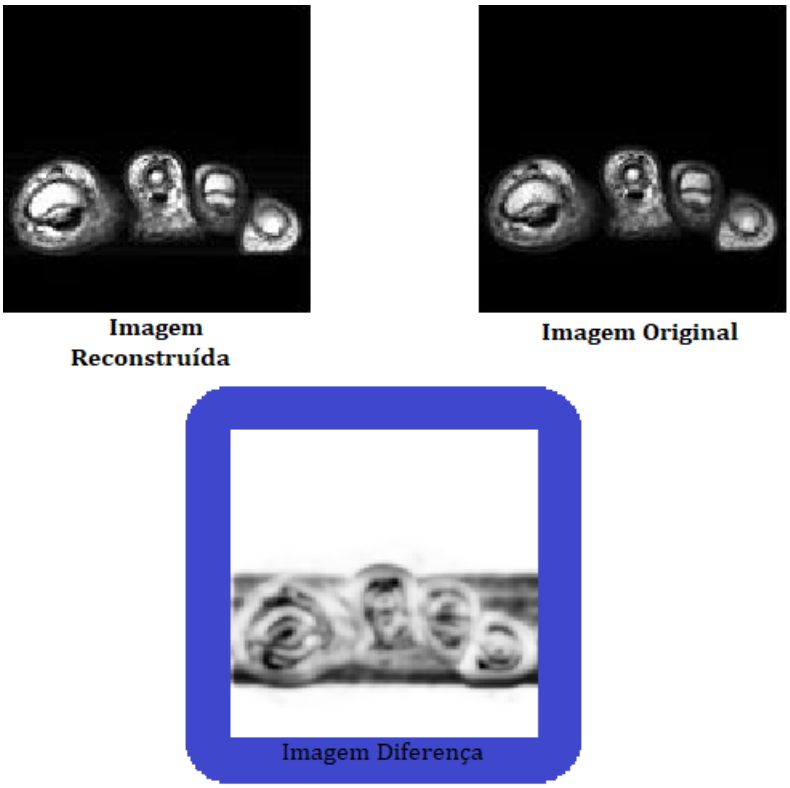

Figura 25 – Comparação entre imagem obtida e imagem original.

$$K = 0,85658 \tag{4.1}$$

O coeficiente de similaridade estrutural obtido através da função é indicado na Equação 4.1, isso mostra uma similaridade de 85,658%. Para outras imagens testadas esse valor ficou maior do que 60% mostrando uma alta semelhança entre as imagens originais e as reconstruídas. Outras imagens testadas podem ser observadas na pasta "Imagens" no Github e também no Anexo D, Figura 30.

4.3 Velocidade e Precisão no Tempo

Em todos os testes foi observado o tempo levado para concluir a reconstrução, com o objetivo de obter uma medida mais precisa foi inserida uma variável que contou o número de ciclos de clock, para uma imagem de 128x128 foram necessários 49.900 ciclos de clock, esse valor permaneceu o mesmo independente da imagem que estava sendo reconstruída.

Para uma frequência de trabalho de 200 MHz, em torno da metade da frequência máxima que o bloco de FFT suporta, um ciclo de clock demoraria 5 ns para ser realizado, logo 49.900 ciclos de clock tomaria 0,2495 ms, uma frequência de 4008,016 imagens por segundo.

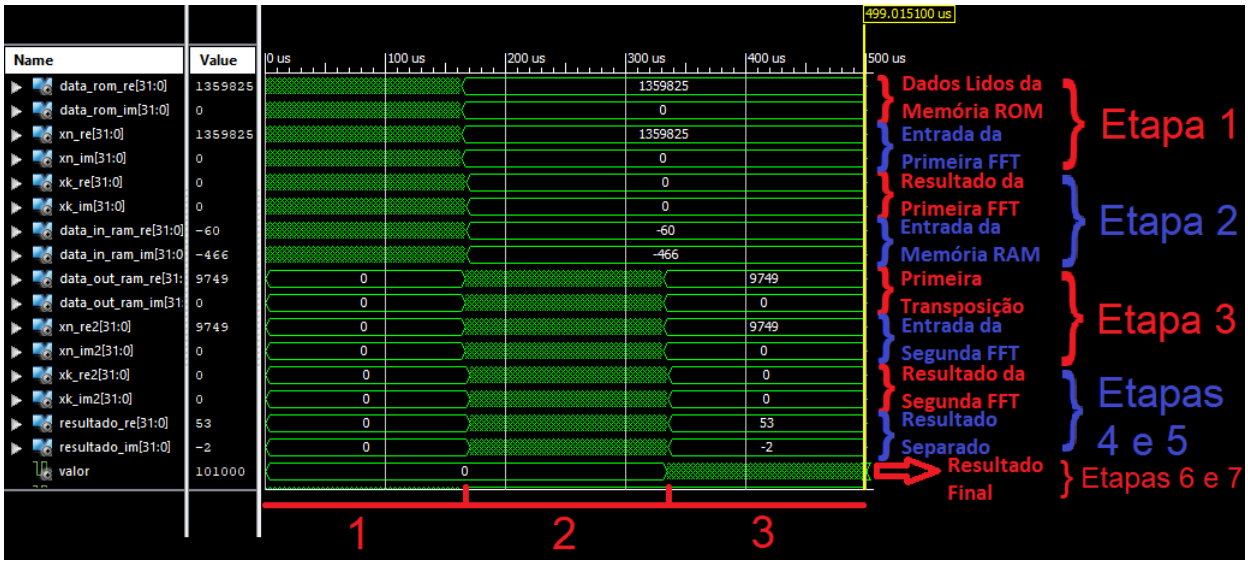

Figura 26 – Simulação visualizada no ISE 14.7 com legendas de dados.

Como pode ser observado na Figura 26, o processo de reconstrução pode ser dividido em três partes aproximadamente iguais, cada um desses estágios possui 16.384 ciclos de clock, um para cada valor da imagem de 128x128, totalizando 49.152 ciclos de clock. São 374 ciclos de clock do início da leitura dos dados no núcleo FFT para o início da entrega dos resultados, como são dois núcleos FFT, totalizam 49.900 ciclos de clock para todo o processo. Dessa forma é possível estimar um tempo para cálculos de reconstruções de imagens maiores, o número de ciclos de clock para o início dos resultados varia de acordo com o tamanho da transformada, porém para transformadas de tamanho 128 ou maior eles correspondem a menos de 1% do tempo de reconstrução. Portanto é possível obter uma equação que calcule aproximadamente o número de clocks.

$$Nclk \simeq 3.(C.L) \tag{4.2}$$

A equação 4.2 mostra como calcular o número aproximado de ciclos de clock, onde C é o comprimento da imagem em pixels e L a largura da imagem em pixels, com essa fórmula é possível calcular o número de imagens por segundo que o algoritmo consegue entregar de acordo com o tamanho da imagem.

$$Tx \simeq \frac{F}{Nclk} \tag{4.3}$$

$$Tx \simeq \frac{F}{3.(C.L)} \tag{4.4}$$

Nas equações 4.3 e 4.4 é possível calcular essa taxa de imagens por segundo de acordo com a frequência F do sistema, em Hertz, e o comprimento e largura da imagem, em pixels. Podendo assim dimensionar o sistema de acordo com as suas entradas permitindo projetar o sistema para que cumpra certos requisitos de velocidade de execução e qualidade da imagem, por exemplo aumentando o tamanho da imagem e sacrificando parte da velocidade de reconstrução.

4.4 Considerações Finais

Nesta seção foram apresentados os resultados e feitas análises e comparações, chegando em uma fórmula que permite o dimensionamento do sistema de acordo com seus requisitos. Na próxima seção serão feitas as conclusões e considerações finais.

5 Conclusão

Exames de ressonância magnética estão sendo constantemente aprimorados e refinados. modos diferentes de aquisição de imagens, reconstruções mais eficientes e formas diferentes de observar os dados são desenvolvidos todos os anos. A reconstrução da imagem em hardware é um dos possíveis passos que serão realizados nos próximos anos pois o uso de hardwares reprogramáveis está crescendo em diversas áreas da indústria.

Os resultados obtidos no projeto, tanto em qualidade das imagens, quanto em velocidade e precisão de processamento conseguiram se comparar com resultados obtidos em processamentos realizados em computadores por GPUs. As imagens reconstruídas pelo código em VHDL possuem nitidez o suficiente para observar detalhes e são pelo menos 60% idênticas as imagens reconstruídas por computadores como mostrado nas Figuras 25 e 30. A velocidade de reconstrução da imagem pelo algoritmo não seria um gargalo para as máquinas, exames em tempo real são executados com até mesmo 18 imagens por segundo (UECKER et al., 2010), o algoritmo conseguiu taxas superiores a essa, podendo ser aumentada o tamanho da imagem para ganho de qualidade sacrificando tempo de processamento com uma folga relativamente grande. Por último variações em número de clocks não foram observadas em nenhuma das simulações, imagens de mesmo tamanho foram reconstruídas na mesma quantidade de ciclos de clock independente da complexidade da imagem.

5.1 Trabalhos Futuros

A implementação do algoritmo em hardware seria o próximo passo para mostrar as vantagens de se realizar a reconstrução em hardware, e assim fazer um estudo mais aprofundado da influência do jitter de clock da FPGA na reconstrução dessas imagens. Também poderia refinar-se o algoritmo e entender o porquê da imagem reconstruída não estar apenas na parte real do resultado. Entretanto, os dados obtidos com essa simulação conseguiram provar que uma possível implementação feita em uma FPGA do algoritmo não decepcionaria em termos de qualidade, velocidade e precisão no tempo, conseguindo suceder nos três parâmetros com qualidade superior à esperada.

5.2 Considerações Finais

O projeto sucedeu em todos os pontos esperados e todos os objetivos citados no início do projeto foram cumpridos, provando o valor de hardwares reprogramáveis. Posteriormente uma implementação prática do algoritmo criado faz-se necessária para

aprimoramento da ferramenta criada e verificação dos resultados na prática, realizando tanto reconstruções de imagens únicas quanto imagens em sequência criando vídeos em tempo real.

Referências

BERTEN. *GPU vs FPGA Performance Comparison*. 2016. Disponível em: <http://www.bertendsp.com/pdf/whitepaper/BWP001_GPU_vs_FPGA_Performance_Comparison_v1.0.pdf>. Citado na página 16.

BREWER, R. G.; HAHN, E. L. Atomic memory. *Scientific American*, JSTOR, v. 251, n. 6, p. 50–57, 1984. Citado na página 26.

BRUKER. *Almanac 2011*. 2011. Disponível em: <https://lsa.umich.edu/content/dam/chem-assets/chem-docs/BrukerAlmanac2011.pdf>. Citado na página 18.

ELSTER, A. D. *Echo-Planar Imaging (EPI)*. 2019. Disponível em: <http://mriquestions.com/echo-planar-imaging.html>. Citado 2 vezes nas páginas 7 e 30.

FAVREAU, J.; FILONI, D. *The Mandalorian*. [S.l.]: Disney, 2019. Citado 2 vezes nas páginas 8 e 52.

FELMLEE, J. et al. Magnetic resonance imaging phase encoding: a pictorial essay. *radiographics*, v. 9, n. 4, p. 717–722, 1989. Citado 2 vezes nas páginas 7 e 29.

GONZALES, R. C.; WOODS, R. E. *Digital image processing*. [S.l.]: Prentice hall New Jersey, 2002. Citado 2 vezes nas páginas 8 e 52.

HAHN, E. L. Spin echoes. *Physical review*, APS, v. 80, n. 4, p. 580, 1950. Citado na página 25.

HAIK, J. et al. Mri induced fourth-degree burn in an extremity, leading to amputation. *burns*, Elsevier, v. 35, n. 2, p. 294–296, 2009. Citado na página 15.

HEALTHINEERS, S. *Clinical Imaging Solutions*. 2019. Disponível em: <https://www.siemens-healthineers.com/magnetic-resonance-imaging/clinical-specialities>. Citado na página 15.

HELLMICH, M. R.; SZABO, C. Hydrogen sulfide and cancer. In: *Chemistry, Biochemistry and Pharmacology of Hydrogen Sulfide*. [S.l.]: Springer, 2015. p. 233–241. Citado na página 18.

LAUTERBUR, P. C. et al. Image formation by induced local interactions: examples employing nuclear magnetic resonance. 1973. Citado 2 vezes nas páginas 7 e 14.

LI, L.; WYRWICZ, A. M. Parallel 2d fft implementation on fpga suitable for real-time mr image processing. *Review of Scientific Instruments*, AIP Publishing, v. 89, n. 9, p. 093706, 2018. Citado na página 16.

LIBRETEXTS. *Atomic Orbitals and Quantum Numbers*. 2019. Disponível em: <https://chem.libretexts.org/Courses/Oregon_Institute_of_Technology/OIT%3A_CHE_202_-_General_Chemistry_II/Unit_2%3A_Electrons_in_Atoms/2.2%3A_Atomic_Orbitals_and_Quantum_Numbers>. Citado na página 19.

MATHWORKS. *FPGA Image Processing - Target image processing algorithms to FPGA or ASIC hardware*. 2019. Disponível em: <https://www.mathworks.com/discovery/fpga-image-processing.html>. Citado na página 15.

MATHWORKS. *Exploring Slices from a 3-Dimensional MRI Data Set*. 2020. Disponível em: <https://www.mathworks.com/help/images/exploring-slices-from-a-3-dimensional-mri-data-set.html>. Citado 2 vezes nas páginas 7 e 27.

MATHWORKS. *ssim*. 2020. Disponível em: <https://www.mathworks.com/help/images/ref/ssim.html>. Citado na página 40.

MAZZOLA, A. A. Ressonância magnética: princípios de formação da imagem e aplicações em imagem funcional. *Revista Brasileira de Física Médica*, v. 3, n. 1, p. 117–129, 2009. Citado 5 vezes nas páginas 7, 15, 20, 22 e 23.

MOHR, P. J.; TAYLOR, B. N.; NEWELL, D. B. Codata recommended values of the fundamental physical constants: 2006. *Journal of Physical and Chemical Reference Data*, NIST, v. 80, n. 3, p. 633–1284, 2008. Citado na página 21.

OECD. *Magnetic resonance imaging (MRI) exams (indicator)*. 2019. Disponível em: <https://data.oecd.org/healthcare/magnetic-resonance-imaging-mri-exams.htm>. Citado na página 15.

OECD. *Magnetic resonance imaging (MRI) units (indicator)*. 2019. Disponível em: <https://data.oecd.org/healtheqt/magnetic-resonance-imaging-mri-units.htm>. Citado na página 15.

PRINCE, J. L.; LINKS, J. M. *Medical imaging signals and systems*. [S.l.]: Pearson Prentice Hall Upper Saddle River, NJ, 2006. Citado 5 vezes nas páginas 7, 18, 19, 20 e 21.

RAO, K. R.; KIM, D. N.; HWANG, J. J. *Fast Fourier transform-algorithms and applications*. [S.l.]: Springer Science & Business Media, 2011. Citado na página 15.

ROSEN, B.; WALD, L. Mr image encoding. Citado na página 31.

SHRIVER, D. F.; ATKINS, P. W. *Química Inorgânica*. [S.l.]: Bookman, 1999. v. 3. Citado na página 19.

STANISZ, G. J. et al. T1, t2 relaxation and magnetization transfer in tissue at 3t. *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, Wiley Online Library, v. 54, n. 3, p. 507–512, 2005. Citado 2 vezes nas páginas 9 e 25.

THURSTON, U. B. M. *Echo planar imaging*. 2019. Disponível em: <https://radiopaedia.org/articles/echo-planar-imaging-1>. Citado na página 30.

TUBRIDY, N.; MCKINSTRY, C. Neuroradiological history: Sir joseph larmor and the basis of mri physics. *Neuroradiology*, Springer, v. 42, n. 11, p. 852–855, 2000. Citado na página 19.

UECKER, M. et al. Real-time mri at a resolution of 20 ms. *NMR in Biomedicine*, Wiley Online Library, v. 23, n. 8, p. 986–994, 2010. Citado na página 43.

XILINX. *LogiCORE IP Fast Fourier Transform v7.1*. 2011. Citado 3 vezes nas páginas 7, 34 e 35.

XILINX. *LogiCORE IP Block Memory Generator v7.3*. 2012. Citado 2 vezes nas páginas 35 e 37.

XILINX. *What is an FPGA?* 2019. Disponível em: <https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html>. Citado na página 15.

Anexos

ANEXO A – QR-Code para animação para os pulsos de $90°$ e $180°$.

Figura 27 – QR-Code para animação para os pulsos de 90° e 180°.

http://mriphysics.github.io/images/M_spin_echo.gif.

ANEXO B – QR-Code do repositório do projeto no Github

Figura 28 – QR-Code do repositório do projeto no Github.

https://github.com/AmauriCJr/TCC.

ANEXO C – Etapas de reconstrução da imagem.

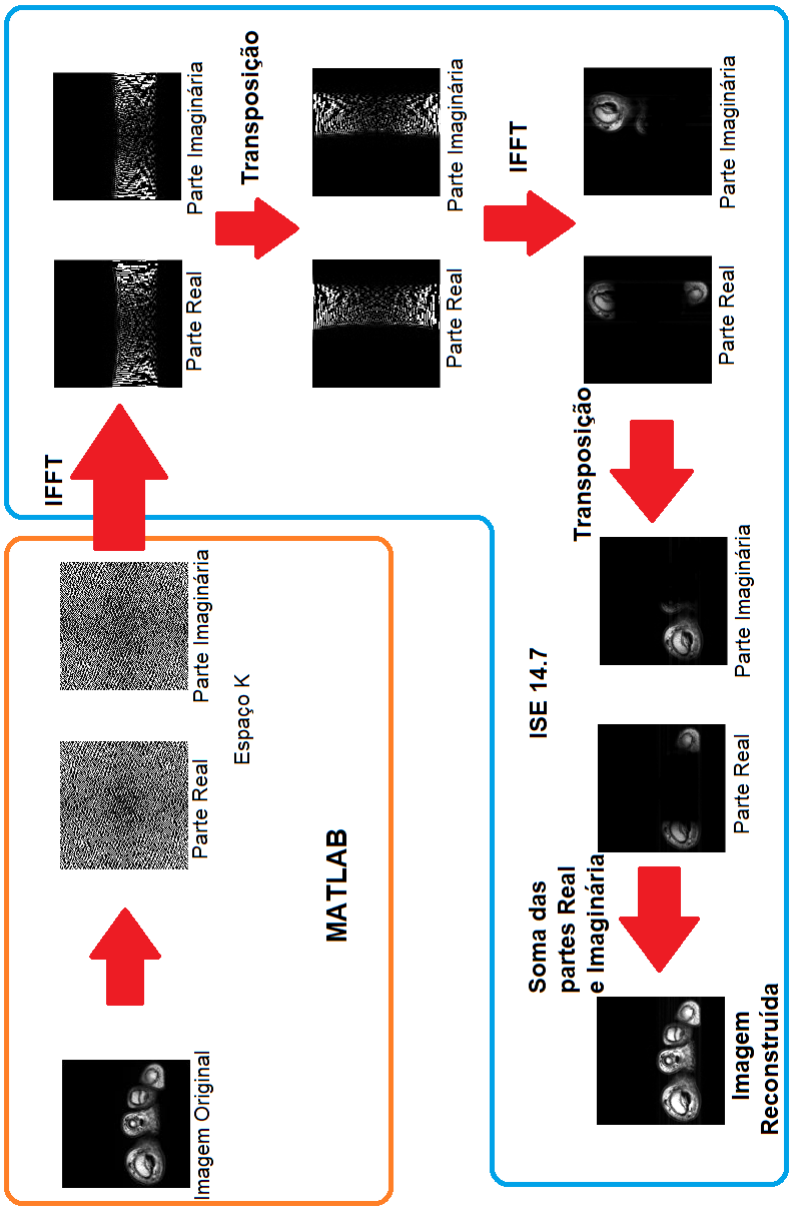

Figura 29 – Etapas de reconstrução da imagem.

ANEXO D – Comparação de diversos resultados.

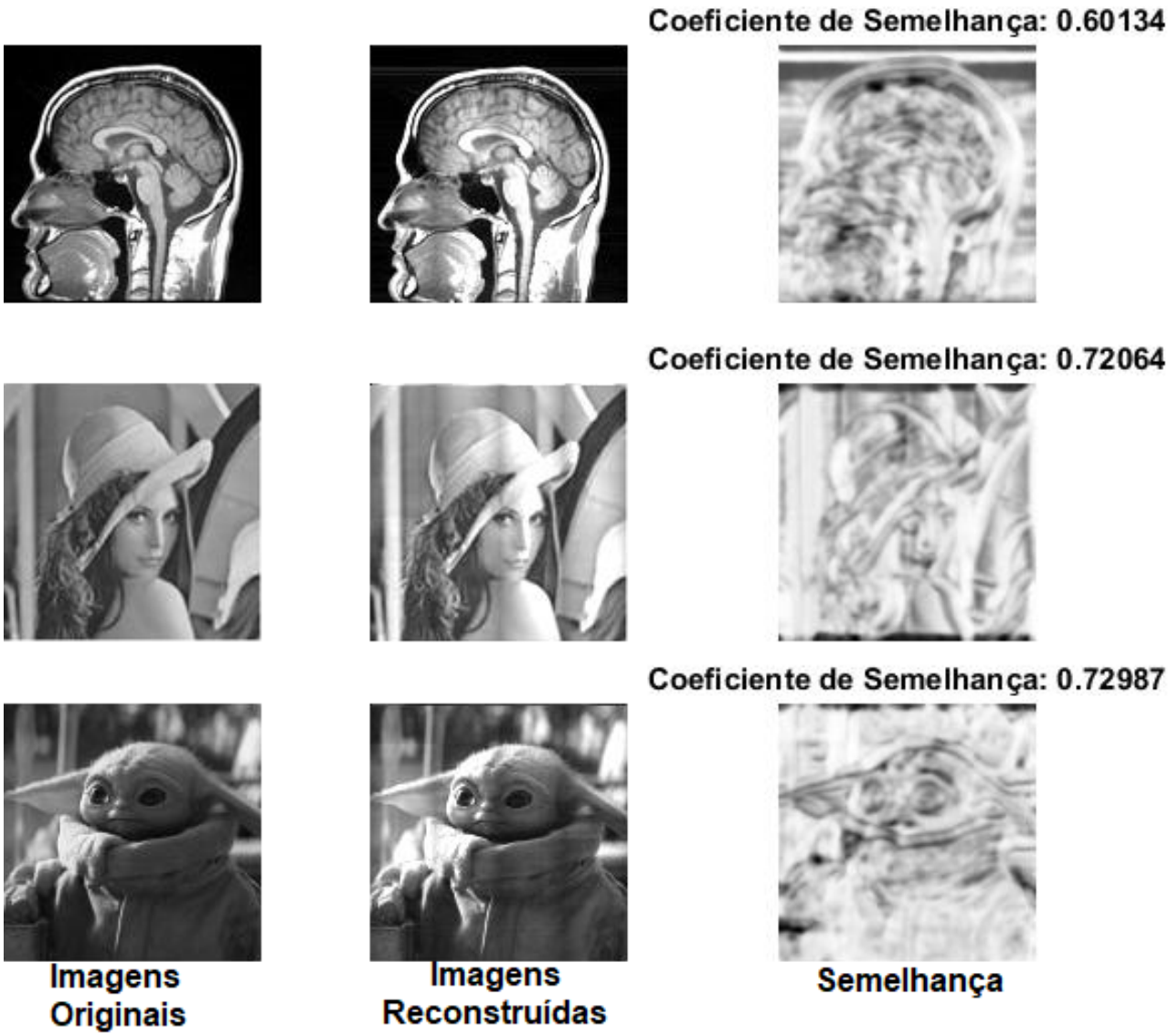

Figura 30 – Comparação de diversos resultados (GONZALES; WOODS, 2002), (FAVREAU; FILONI, 2019).

ANEXO E – imagetocoe.m

Código em MATLAB que salva a imagem em formato .coe.

```
clc, clear all, format long


img = imread('C:\Users\amaur\OneDrive\Área de Trabalho\Eletrônica\TCC1\Yoda.jpg');
X = img;

img = imresize((img),[128 128]);

t = img(:,:,1)
figure
y = imshow(t);

img;

FFT = fft2(t);



% F = abs(fftshift(FFT));
%
% %
% % X = ifft2(F);
%
% % figure
% % y = imshow(X);

Sat = 65535;

Re = real(FFT);
% Re(Re>Sat) = Sat;
% Re(Re<-Sat) = -Sat;

Im= imag(FFT);
% Im(Im>Sat) = Sat;
% Im(Im<-Sat) = -Sat;



X = Re + Im*i;

% X = uint8(X);

X = ifft2(X);

% T = ifft2 (FFT)
% K = ifft2 (X);

% img2 = uint8(K);

imag2 = imag(X);
```

```
real2 = real(X);

figure
Y = imshow(uint8(X));


fid = fopen('C:\Users\amaur\OneDrive\Documentos\Vivado\PeRe.coe', 'wt');
fprintf(fid,'memory_initialization_radix=10;\n');
fprintf(fid,'memory_initialization_vector=');
fprintf(fid, '%0.f,', Re);
fclose(fid)


fid = fopen('C:\Users\amaur\OneDrive\Documentos\Vivado\PeIm.coe', 'wt');
fprintf(fid,'memory_initialization_radix=10;\n');
fprintf(fid,'memory_initialization_vector=');
fprintf(fid, '%.0f,', Im);
fclose(fid)
```

ANEXO F – VisualizarImagem.m

Código em MATLAB para visualizar a imagem após reconstrução.

```
clc, clear all, format long


fid = fopen('C:\Users\amaur\OneDrive\Documentos\Vivado\Resultado.txt');
data = textscan(fid,'%f');
data = data{:};
fid = fclose(fid);



for j = 0:127
    x(1:128,j+1) = data((1+(128*j)):(128*(j+1)),1);

end


k = uint8(x);


figure
y = imshow(k);


img = imread('C:\Users\amaur\OneDrive\Área de Trabalho\Eletrônica\TCC1\MRIBrain.jpg');
X = img;

img = imresize((img),[128 128]);

t = img(:,:,1);
figure
y = imshow(t);


[K,ssimmap] = ssim(k,t);

figure
imshow(ssimmap,[])
title(['Coeficiente de Semelhança: ',num2str(K)])
```

ANEXO G – Código que realiza a reconstrução

Código em VHDL que realiza a reconstrução da imagem de ressonância magnética.

```
----------------------------------------------------------------------------------
-- Company: Universidade de Brasília
-- Engineer: Amauri da Costa Júnior
--
-- Create Date:    17:28:58 11/02/2020
-- Design Name: Recontrução da imagem do pé da minha mãe
-- Module Name:    TopLevel - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use ieee.numeric_std.all;
use ieee.math_real.all;
use STD.textio.all;
use ieee.std_logic_textio.all;




entity TopLevel is
end TopLevel;

architecture Behavioral of TopLevel is



COMPONENT image1
  PORT (
    clka : IN STD_LOGIC;
    addra : IN STD_LOGIC_VECTOR(13 DOWNTO 0);
    douta : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
  );
END COMPONENT;

COMPONENT image4
  PORT (
```

```
    clka : IN STD_LOGIC;
    addra : IN STD_LOGIC_VECTOR(13 DOWNTO 0);
    douta : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
  );
END COMPONENT;

COMPONENT image2
  PORT (
    clka : IN STD_LOGIC;
    wea : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
    addra : IN STD_LOGIC_VECTOR(13 DOWNTO 0);
    dina : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
    douta : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
  );
END COMPONENT;

COMPONENT image3
  PORT (
    clka : IN STD_LOGIC;
    wea : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
    addra : IN STD_LOGIC_VECTOR(13 DOWNTO 0);
    dina : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
    douta : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
  );
END COMPONENT;


COMPONENT Resultado
  PORT (
    clka : IN STD_LOGIC;
    wea : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
    addra : IN STD_LOGIC_VECTOR(13 DOWNTO 0);
    dina : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
    douta : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
  );
END COMPONENT;


COMPONENT ResultadoIm
  PORT (
    clka : IN STD_LOGIC;
    wea : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
    addra : IN STD_LOGIC_VECTOR(13 DOWNTO 0);
    dina : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
    douta : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
  );
END COMPONENT;



signal done1 : std_logic := '0';
signal wr_enable, wr_res_enable: STD_LOGIC_VECTOR(0 DOWNTO 0) := "0";
signal addr_rom_re,addr_rom_im,addr_ram : STD_LOGIC_VECTOR(13 DOWNTO 0) := (others => '0');
signal data_rom_re,data_in_ram_re,data_out_ram_re : STD_LOGIC_VECTOR(31 DOWNTO 0) := (others => '0');
signal data_rom_im,data_in_ram_im,data_out_ram_im : STD_LOGIC_VECTOR(31 DOWNTO 0) := (others => '0');


signal row_index,col_index,finalizaT, VALOR, VALOR2, VALOR3, VALOR4, VALOR5, VALOR6, Resposta, CONTAGEMCLK : integer := 0

signal SAIDA, SAIDA2, SAIDA3, SAIDA4, SAIDA5, SAIDA6 : signed(31 downto 0);
```

```
signal xk_index_row,xk_index_col : integer := -1;

signal xk_index_row2,xk_index_col2, contador3 : integer := -1;



signal contador : STD_LOGIC_VECTOR(7 DOWNTO 0) := (others => '0');
signal contador2 : STD_LOGIC_VECTOR(7 DOWNTO 0) := (others => '0');
signal contador4 : STD_LOGIC_VECTOR(31 DOWNTO 0) := (others => '0');

signal TransformadaPronta : STD_LOGIC := '0';


signal resultado_re, resultadoT_re, resultado_im, resultadoT_im: STD_LOGIC_VECTOR(31 DOWNTO 0) := (others => '0');
signal addr_res : STD_LOGIC_VECTOR(13 DOWNTO 0) := (others => '0');


constant CLOCK_PERIOD : time := 10 ns;

------------------------------------------------------------------------------------------------
---------------------------------SINAIS DO PRIMEIRO NÚCLEO--------------------------------------
------------------------------------------------------------------------------------------------

    signal clk  : STD_LOGIC:= '0'; --Clock
    signal start  : STD_LOGIC:= '1'; --Manda o processo de começar o carregamento de dados e fft começar
 signal sclr  : STD_LOGIC:= '0'; --reseta o núcleo
    signal xn_re  : STD_LOGIC_VECTOR(31 DOWNTO 0); --Dados reais
    signal xn_im  : STD_LOGIC_VECTOR(31 DOWNTO 0); --Dados imaginários
    signal fwd_inv  : STD_LOGIC:= '0'; --1 pra transformada direta, 0 pra inversa
    signal fwd_inv_we  : STD_LOGIC:= '1'; --permite a variavel de cima modificar
 signal scale_sch  : STD_LOGIC_VECTOR(7 DOWNTO 0):= "01101010";
    signal  scale_sch_we  : STD_LOGIC:= '1';
    signal rfd  : STD_LOGIC;  --Daqui pra baixo é saida
    signal xn_index  : STD_LOGIC_VECTOR(6 DOWNTO 0);
    signal busy  : STD_LOGIC;
    signal edone  : STD_LOGIC;
    signal done  : STD_LOGIC;
    signal dv  : STD_LOGIC;
    signal xk_index  : STD_LOGIC_VECTOR(6 DOWNTO 0);
 signal  cpv  : STD_LOGIC;
    signal  rfs  : STD_LOGIC;
    signal xk_re  : STD_LOGIC_VECTOR(31 DOWNTO 0);
    signal xk_im  : STD_LOGIC_VECTOR(31 DOWNTO 0);

------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------



------------------------------------------------------------------------------------------------
---------------------------------SINAIS DO SEGUNDO NÚCLEO---------------------------------------
------------------------------------------------------------------------------------------------


 signal clk2  : STD_LOGIC:= '0'; --Clock
    signal start2  : STD_LOGIC:= '0'; --Manda o processo de começar o carregamento de dados e fft começar
 signal sclr2  : STD_LOGIC:= '0'; --reseta o núcleo
 signal  addr_in_fft2 : STD_LOGIC_VECTOR(6 DOWNTO 0):= (others => '0');
    signal xn_re2  : STD_LOGIC_VECTOR(31 DOWNTO 0); --Dados reais
```

```
    signal xn_im2  : STD_LOGIC_VECTOR(31 DOWNTO 0); --Dados imaginários
    signal fwd_inv2 : STD_LOGIC:= '0'; --1 pra transformada direta, 0 pra inversa
    signal fwd_inv_we2  : STD_LOGIC:= '1'; --permite a variavel de cima modificar
 signal scale_sch2  : STD_LOGIC_VECTOR(7 DOWNTO 0):= "11111111";
    signal  scale_sch_we2  : STD_LOGIC:= '1';
    signal rfd2  : STD_LOGIC;  --Daqui pra baixo é saida
    signal xn_index2  : STD_LOGIC_VECTOR(6 DOWNTO 0);
    signal busy2  : STD_LOGIC;
    signal edone2  : STD_LOGIC;
    signal done2 : STD_LOGIC;
    signal dv2  : STD_LOGIC;
    signal xk_index2  : STD_LOGIC_VECTOR(6 DOWNTO 0);
 signal  cpv2  : STD_LOGIC;
 signal  rfs2  : STD_LOGIC;
    signal xk_re2  : STD_LOGIC_VECTOR(31 DOWNTO 0);
    signal xk_im2  : STD_LOGIC_VECTOR(31 DOWNTO 0);


------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
signal Transformada2Pronta : STD_LOGIC := '0';



COMPONENT FFT
  PORT (
    clk : IN STD_LOGIC;
    sclr : IN STD_LOGIC;
    start : IN STD_LOGIC;
    xn_re : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
    xn_im : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
    fwd_inv : IN STD_LOGIC;
    fwd_inv_we : IN STD_LOGIC;
    scale_sch : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    scale_sch_we : IN STD_LOGIC;
    rfd : OUT STD_LOGIC;
    xn_index : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
    busy : OUT STD_LOGIC;
    edone : OUT STD_LOGIC;
    done : OUT STD_LOGIC;
    dv : OUT STD_LOGIC;
    xk_index : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
    xk_re : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
    xk_im : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
  );
END COMPONENT;

COMPONENT FFT2
  PORT (
    clk : IN STD_LOGIC;
    sclr : IN STD_LOGIC;
    start : IN STD_LOGIC;
    xn_re : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
    xn_im : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
    fwd_inv : IN STD_LOGIC;
    fwd_inv_we : IN STD_LOGIC;
    scale_sch : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    scale_sch_we : IN STD_LOGIC;
    rfd : OUT STD_LOGIC;
    xn_index : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
```

```
    busy : OUT STD_LOGIC;
    edone : OUT STD_LOGIC;
    done : OUT STD_LOGIC;
    dv : OUT STD_LOGIC;
    xk_index : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
    xk_re : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
    xk_im : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
  );
END COMPONENT;


begin

FFT_IPCore : FFT
  PORT MAP (
    clk => clk,
    sclr => sclr,
    start => start,
    xn_re => xn_re,
    xn_im => xn_im,
    fwd_inv => fwd_inv,
    fwd_inv_we => fwd_inv_we,
    scale_sch => scale_sch,
    scale_sch_we => scale_sch_we,
    rfd => rfd,
    xn_index => xn_index,
    busy => busy,
    edone => edone,
    done => done,
    dv => dv,
    xk_index => xk_index,
    xk_re => xk_re,
    xk_im => xk_im
  );

FFT2_IPCore : FFT2
  PORT MAP (
    clk => clk2,
    sclr => sclr2,
    start => start2,
    xn_re => xn_re2,
    xn_im => xn_im2,
    fwd_inv => fwd_inv2,
    fwd_inv_we => fwd_inv_we2,
 scale_sch => scale_sch2,
    scale_sch_we => scale_sch_we2,
    rfd => rfd2,
    xn_index => xn_index2,
    busy => busy2,
    edone => edone2,
    done => done2,
    dv => dv2,
    xk_index => xk_index2,
    xk_re => xk_re2,
    xk_im => xk_im2
  );
```

```
  ---------------------------------------------------------------------
  -- Gera o clock
  ---------------------------------------------------------------------

clock_gen : process
begin
clk <= '0';
loop
clk <= '0';
      wait for CLOCK_PERIOD/2;
      clk <= '1';
      wait for CLOCK_PERIOD/2;
end loop;
end process clock_gen;



image_rom_re : image1 port map(clk,addr_rom_re,data_rom_re);

image_rom_im : image4 port map(clk,addr_rom_im,data_rom_im);

image_ram_re : image2 port map(clk,wr_enable,addr_ram,data_in_ram_re,data_out_ram_re);

image_ram_im : image3 port map(clk,wr_enable,addr_ram,data_in_ram_im,data_out_ram_im);

ImagemFinal : Resultado port map(clk,wr_res_enable,addr_res,resultado_re,resultadoT_re);

Parte_Im : ResultadoIm port map(clk,wr_res_enable,addr_res,resultado_im,resultadoT_im);


-- Para a escrita das variáveis de configuração
process(clk)
begin
    if(rising_edge(clk)) then
fwd_inv_we <= '0';
scale_sch_we <= '0';
 end if;
end process;


process(clk)
begin
    if(rising_edge(clk) and (start2 = '1')) then
fwd_inv_we2 <= '0';
scale_sch_we2 <= '0';
 end if;
end process;




--LE A IMAGEM SEPARADA EM PARTE REAL E IMAGINARIA
process(clk)
```

```
begin
    if(rising_edge(clk)) then
        if(done1 = '0') then
addr_rom_re <= addr_rom_re + "00000000000001";
         addr_rom_im <= addr_rom_im + "00000000000001";
            if(col_index = 127) then
col_index <= 0;
                if(row_index = 127) then
row_index <= 0;
done1 <= '1';
                else
row_index <= row_index + 1;
                end if;
            else
col_index <= col_index + 1;
            end if;
end if;
     end if;
end process;


xn_re <= data_rom_re;
xn_im <= data_rom_im;

xn_re2 <= data_out_ram_re;
xn_im2 <= data_out_ram_im;

---------------------------SINAL DE CONCLUSÃO DA PRIMEIRA FFT----------------------
process(clk)
begin
if(rising_edge(clk)) and(edone = '1') and (sclr = '0') then
contador <= contador + "00000001";
if (contador = "10000000") then
TransformadaPronta <= '1';
contador <= "00000000";
else
TransformadaPronta <= '0';
end if;
end if;
end process;

---------------------------SINAL DE CONCLUSÃO DA SEGUNDA FFT----------------------
process(clk2)
begin
if(rising_edge(clk2)) and xk_index2 = "1111111" then
contador2 <= contador2 + "00000001";
if (contador2 = "01111111") then
Transformada2Pronta <= '1';
sclr2 <= '1';
else
Transformada2Pronta <= '0';
end if;
end if;
end process;




process(clk)
begin
```

```
if(rising_edge(clk) and edone = '1') then
xk_index_row <= xk_index_row + 1;
end if;
end process;

xk_index_col <= conv_integer(xk_index);

process(clk)
begin
if(rising_edge(clk)) then
if(TransformadaPronta = '0' and dv = '1') then
wr_enable <= "1"; --liga o modo escrever
data_in_ram_re <= xk_re;
data_in_ram_im <= xk_im;
addr_ram <= std_logic_vector(to_unsigned((xk_index_col*128 + xk_index_row),14)); --o número 6 é o tamanho do vetor de end
else
if (finalizaT < 2) then
wr_enable <= "0";  --desliga o modo escrever
--addr_rom_re <= "000000";
if(addr_ram = "11111111111111") then
addr_ram <= "00000000000000";
finalizaT <= finalizaT +1;
            else
addr_ram <= addr_ram + 1;
            end if;
        end if;
    end if;
end if;
end process;


process(clk2)
begin
if(rising_edge(clk2) and edone2 = '1') then
xk_index_row2 <= xk_index_row2 + 1;
end if;
end process;

xk_index_col2 <= conv_integer(xk_index2);

process(clk2)
begin
if(rising_edge(clk2)) then
if(Transformada2Pronta = '0' and dv2 = '1') then
wr_res_enable <= "1"; --liga o modo escrever
resultado_re <= xk_re2;
resultado_im <= xk_im2;
addr_res <= std_logic_vector(to_unsigned((xk_index_col2*128 + xk_index_row2),14)); --o número 14 é o tamanho do vetor de
else
if (Resposta < 2) then
wr_res_enable <= "0";  --desliga o modo escrever
if(addr_res = "11111111111111") then
addr_res <= "00000000000000";
Resposta <= Resposta +1;
            else
addr_res <= addr_res + 1;
            end if;
        end if;
    end if;
end if;
```

```
end process;




process(clk)
begin
if(TransformadaPronta = '1') then
sclr <= '1';
end if;
end process;

process(clk)
begin
if(TransformadaPronta = '1') then
clk2 <= clk;
start2 <= '1';
end if;
end process;

process(clk2)
begin
if(rising_edge(clk2) and (finalizaT = 1)) then
addr_in_fft2 <= addr_in_fft2 + "0000001";

end if;
end process;


process(clk)
begin
if(rising_edge(clk) and (Transformada2Pronta = '1') and contador3 < 6) then
contador3 <= contador3 + 1;
end if;
end process;



CONTAR_CLKs: process(clk)
begin
if(rising_edge(clk)) then
CONTAGEMCLK <= CONTAGEMCLK + 1;
end if;
end process CONTAR_CLKs;

-----------------------------------------------------------------------------------------------


VALOR <= to_integer(abs(signed(resultadot_re))) + to_integer(abs(signed(resultadot_im)));

WRITE_FILE_RE: process(clk)
variable VEC_LINE : line;
file VEC_FILE : text is out "C:\Users\amaur\OneDrive\Documentos\Vivado\Resultado.txt";
begin
if (rising_edge(clk) and contador3 >= 0 and contador4 < 16384) then
write (VEC_LINE, VALOR);
writeline (VEC_FILE, VEC_LINE);
contador4 <= contador4 + 1;
end if;
```

```
end process WRITE_FILE_RE;



SAIDA2 <= signed(xk_re2);
VALOR2 <= to_integer(SAIDA2);

WRITE_FILE_RE2: process(clk)
variable VEC_LINE : line;
file VEC_FILE : text is out "C:\Users\amaur\OneDrive\Documentos\Vivado\Resultado2.txt";
begin
if (rising_edge(clk) and dv2 = '1') then
write (VEC_LINE, VALOR2);
writeline (VEC_FILE, VEC_LINE);
--contador4 <= contador4 + 1;
end if;
end process WRITE_FILE_RE2;



SAIDA4 <= signed(xk_im2);
VALOR4 <= to_integer(SAIDA4);

WRITE_FILE_RE5: process(clk)
variable VEC_LINE : line;
file VEC_FILE : text is out "C:\Users\amaur\OneDrive\Documentos\Vivado\Resultado3.txt";
begin
if (rising_edge(clk) and dv2 = '1') then
write (VEC_LINE, VALOR4);
writeline (VEC_FILE, VEC_LINE);
--contador4 <= contador4 + 1;
end if;
end process WRITE_FILE_RE5;


SAIDA6 <= signed(xk_re);
VALOR6 <= to_integer(SAIDA6);

WRITE_FILE_RE6: process(clk)
variable VEC_LINE : line;
file VEC_FILE : text is out "C:\Users\amaur\OneDrive\Documentos\Vivado\Resultado4.txt";
begin
if (rising_edge(clk) and dv = '1') then
write (VEC_LINE, VALOR6);
writeline (VEC_FILE, VEC_LINE);
--contador4 <= contador4 + 1;
end if;
end process WRITE_FILE_RE6;

SAIDA3 <= signed(xk_im);
VALOR3 <= to_integer(SAIDA3);

WRITE_FILE_RE3: process(clk)
variable VEC_LINE : line;
file VEC_FILE : text is out "C:\Users\amaur\OneDrive\Documentos\Vivado\Resultado5.txt";
begin
if (rising_edge(clk) and dv = '1') then
write (VEC_LINE, VALOR3);
writeline (VEC_FILE, VEC_LINE);
--contador4 <= contador4 + 1;
```

```
end if;
end process WRITE_FILE_RE3;




end Behavioral;
```