UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE FÍSICA DE SÃO CARLOS

CAROLINA COTTAS MEDEIROS

Reestruturação do sistema de teste e diagnóstico de interfaces para o espectrômetro digital de ressonância magnética do CIERMag

São Carlos
2022

Carolina Cottas Medeiros

Reestruturação do sistema de teste e diagnóstico de interfaces para o espectrômetro digital de ressonância magnética do CIERMag

Trabalho de conclusão de curso apresentado ao Instituto de Física de São Carlos da Universidade de São Paulo para obtenção do título de Bacharel em Física Computacional.
Orientador: Prof. Dr. Alberto Tannús - Instituto de Física de São Carlos.

São Carlos
2022

Medeiros, Carolina Cottas
Reestruturação do sistema de teste e diagnóstico de interfaces para o espectrômetro digital de ressonância magnética do CIERMag / Carolina Cottas Medeiros; orientador Alberto Tannús -- São Carlos, 2022.
27 p.

Trabalho de Conclusão de Curso (Bacharela em Física Computacional) -- Instituto de Física de São Carlos, Universidade de São Paulo, 2022.

1. FPGA. 2. Embarcado. 3. Ressonância magnética. 4. Radiofrequência. I. Tannús, Alberto, orient. II. Título.

RESUMO

O espectrômetro digital de ressonância magnética desenvolvido no Centro de Imagens e Espectroscopia por Ressonância Magnética (CIERMag) surge como alternativa a sistemas comerciais existentes para aplicações em ambiente de pesquisa. Uma de suas características fundamentais é o uso de Field Programmable Gate Arrays (FPGAs), que permite a sintetização de *hardware* a partir de *software* e oferece versatilidade ao sistema, tornando-o multipropósito e imune à obsolescência. A fim de viabilizar tais características do projeto, seus módulos de *hardware* estão em constante atualização. Desenvolveu-se, nesse contexto, um sistema de testes de qualidade automatizado, baseado em arquitetura híbrida em um sistema embarcado de FPGA, para diagnóstico de qualidade de módulos conversores de radiofrequência e de gradientes, utilizando um kit de desenvolvimento Arrow Sockit. Este trabalho propõe o estudo e entendimento desse sistema e dos módulos conversores, a fim de não só fundamentar o conhecimento necessário para sua ampliação, tornando-o compatível com novas versões, como desenvolver familiaridade com um projeto abrangente que envolve *hardware*, *firmware*, *software* e suas integrações. Para tal, a abordagem seguirá as seguintes etapas: (i) compreensão de linguagens de descrição de *hardware* (VHDL e diagrama de blocos) usadas para o processamento de sinais, de programação de baixo nível (C) para desenvolvimento do *firmware*, de programação de alto nível (Python) para desenvolvimento da interface gráfica e do protocolo de comunicação serial SPI; (ii) revisão de todas as etapas de desenvolvimento, desde a compilação do projeto implementado na FPGA usando o programa Quartus, passando pela regravação do sistema operacional Linux LXDE Desktop versão Linaro no micro SD card compatível com o periférico I/O do processador ARM, até a atualização de bibliotecas utilizadas no *software* de validação; (iii) entendimento dos esquemas elétricos do embarcado Arrow Sockit e dos módulos conversores proprietários do CIERMag, a fim de compreender as particularidades de cada versão e analisar sua compatibilidade com o sistema; (iv) atualização do projeto da FPGA, uma vez que com a base necessária, para integração de novas versões, transicionando de um sistema que faz o diagnóstico de módulos de radiofrequência com 2 canais ADC e 2 canais DAC para o de módulos DAC com 4 canais, com devida alteração dos blocos de processamento de dados pela FPGA, além de modificações no SPI para todas as versões..

Palavras-chave: FPGA. Embarcado. Ressonância Magnética. Radiofrequência.

SUMÁRIO

1 INTRODUÇÃO

Imagem por Ressonância Magnética (IRM ou, em inglês, MRI) é um importante método de análise e diagnóstico (1, 2), uma vez que não configura invasão ou potencial de destruição aos núcleos em observação. Para aplicação, são desenvolvidos Sistemas de Ressonância Magnética, ou seja, equipamentos direcionados a gerar sinais eletromagnéticos e campos magnéticos que interajam com tais núcleos.

Estes equipamentos têm como componente de destaque o espectrômetro, que gera todos os sinais necessários aos métodos de Ressonância Magnética e é divido em subsistemas (para o escopo deste trabalho serão importantes apenas os subsistemas de radiofrequência – RF). Além disso, é importante ressaltar que ao longo do tempo, os equipamentos podem se tornar obsoletos ou impedir o desenvolvimento de novas técnicas não só devido à restrição de processadores já existentes no mercado, como também à descontinuação de componentes, além da dependência de assistências técnicas e manutenção de compiladores que podem ser perdidas.

A fim de resolver esse impasse, a tecnologia de "Field-Programmable Gate Array"(FPGA) foi utilizada pelo Centro de Imagens e Espectroscopia por Ressonância Magnética (CIERMag) no desenvolvimento de seu próprio Espectrômetro Digital (*Digital Magnetic Resonance Spectrometer*, DMRS) (3) cuja possibilidade de síntese de circuitos digitais usando linguagens de descrição de *hardware* (HDL) o torna passível de modificações sem necessidade de substituição de componentes. Ainda assim, módulos como conversores transmissores e receptores de radiofrequência são integrados ao sistema, fazendo a ponte entre circuitos digitais de processamento e circuitos analógicos de captação de sinais. Isso significa que, conforme o projeto digital é aperfeiçoado, podem se tornar necessárias modificações também nos projetos analógicos para acomodação das atualizações, resultando em diferentes versões para estas interfaces.

Como o laboratório possui alta demanda de produção de hardware, também há alta demanda de testes de qualidade dos componentes. Visando automatizá-los e evitar a necessidade de técnicos especializados, foi desenvolvido, para um projeto de mestrado (4), um *Sistema de Teste e Diagnóstico de Interfaces (STDI)* para os módulos de RF, aproveitando a disponibilidade de FPGAs pela facilidade de reprogramação conforme o desenvolvimento de novas interfaces.

1.1 Fundamentação teórica

1.1.1 Ressonância Magnética Nuclear

O fenômeno de ressonância a nível nuclear é manifestado com natureza magnética, uma vez que os núcleos de interesse possuem momentum angular intrínseco (spin) e momento magnético associado, significando que torques produzidos por um campo magnético externo forçarão precessão do núcleo (5) (Figura 1 A) como resposta do momentum angular a esse torque. A

quantidade de vezes por segundo que o eixo completa a figura do cone é chamada frequência de Larmor $\frac{\omega}{2\pi}$, que normalmente fica na faixa de radiofrequência. (6)

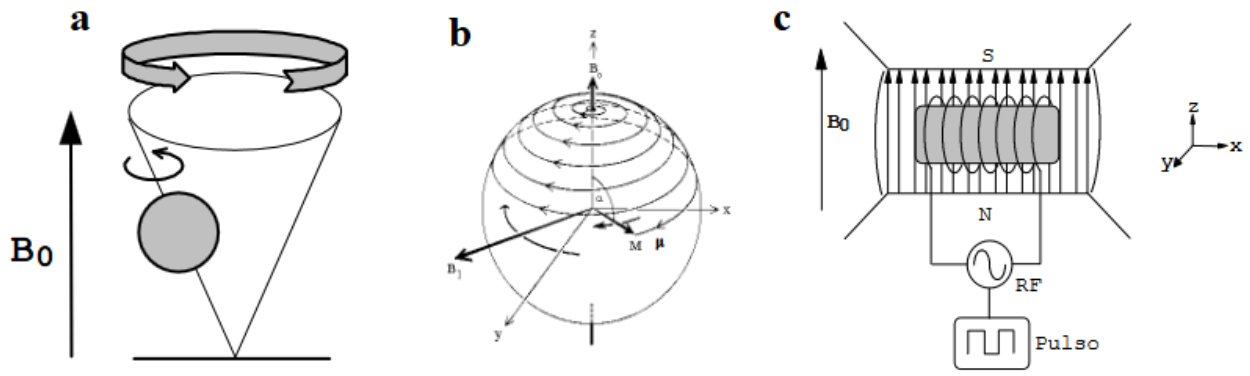

Figura 1 – (a) Movimento de precessão induzido por B_0. (b) Movimento de precessão em espiral induzido pela perturbação de um novo campo B_1. (c) Representação do circuito que induzirá B_1 na amostra.

Fonte: MARTINS (7)

Por esse motivo, um gerador de pulsos de RF ligado a uma bobina (Figura 1 C) irá perturbar os spins de equilíbrio pelo campo B_1, fazendo com que a que a magnetização M seja rebatida ao plano transversal (Figura 1 B). Após o pulso, o sistema tende a voltar a seu estado natural de equilíbrio. Enquanto volta a magnetização, é induzida uma força eletromotriz de mesma frequência e de amplitude proporcional à componente transversal da magnetização no espaço em torno da bobina, que capta esses sinais. A partir de outros conceitos, os sinais são utilizados para gerar a imagem.

2 MATERIAIS E MÉTODOS

O *STDI* (4) foca nos módulos conversores de radiofrequência do equipamento de Ressonância Magnética. Foi, então, criado um software, que gera sinais digitais, e um *hardware*, que gerencia a interação desses sinais com o módulo conversor em teste. O projeto é representado pelo fluxograma da Figura 2, em que os identificadores (A), (B), (C) e (D) representam conexões de interface entre *software* e *hardware* sintetizado ((A) e (D)) e entre *hardware* sintetizado e módulos conversores ((B) e (C)) (notação que continuará sendo usada ao longo deste documento).

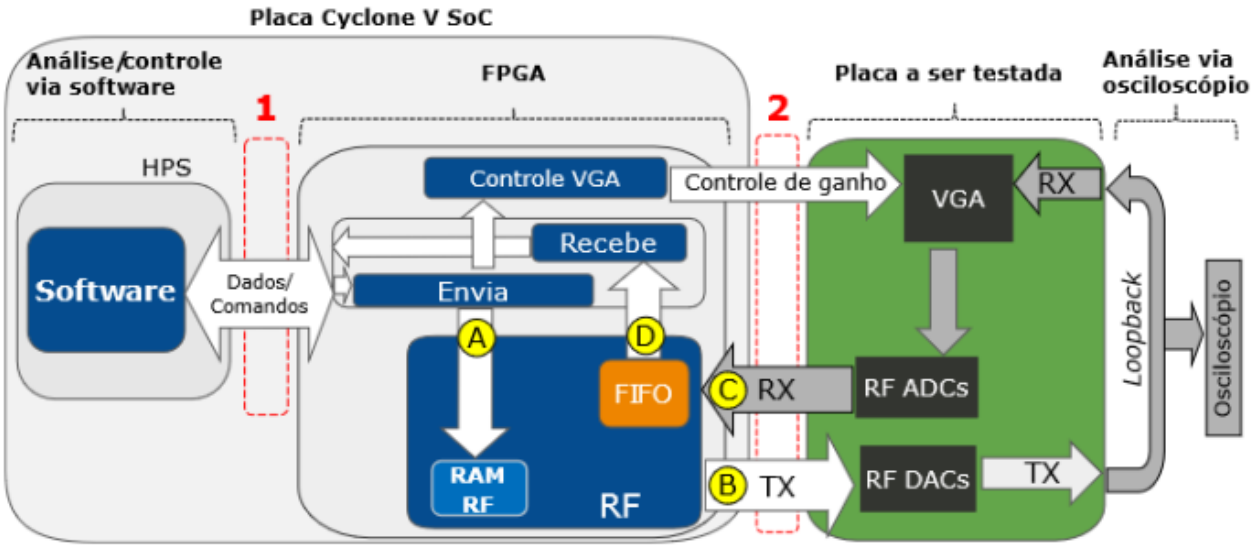

Figura 2 – Fluxograma do Sistema de Teste de Diagnóstico de Interfaces e conexão com o módulo conversor de RF do espectrômetro.

Fonte: BATISTA (4)

2.1 Módulos Conversores de Radiofrequência

São circuitos analógicos (desenvolvidos pelo CIERMag (3)) divididos em canais e compostos por:

- um conversor D/A de radiofrequência (**RF DAC**) de 14 bits, responsável pela *transmissão* da forma de onda (digital) que será amplificada e aplicada às bobinas de radiofrequência (analógicas) através de um conector "Subminiature version A" (SMA);
- um conversor A/D (**RF ADC**) de 14 ou 16 bits (a depender da versão da interface), responsável pela *recepção* dos sinais gerados pelos núcleos excitados também através de

um conector SMA (é importante observar que são sinais fracos, sendo necessária sua pré-amplificação;

- um amplificador de ganho variável (**VGA**), responsável por atenuar ou acentuar o sinal recebido pelo RF ADC, controlado por programação de protocolo Serial Peripheral Interface (SPI);

- um conector "High Speed Mezzanine Card" (HSMC), que é um barramento bidirecional de canais digitais.

- filtros passa-baixa / passa-banda para remoção de componentes indesejadas, uma vez que a amostragem ocorre em uma frequência f_1 e a forma de onda gerada digitalmente possui outra frequência f_0.

O fluxo de sinal nos módulos conversores envolve um gerador digital de pulsos que enviará formas de onda ao RF DAC (pelo conector HSMC), que as converterá para sinais analógicos, enviando-os pelo conector SMA à bobina envolvendo a amostra, o que perturbará o campo magnético induzido nos núcleos; nos momentos em que não houver pulsos, os núcleos tentam voltar a seu estado natural de equilíbrio, gerando outros sinais analógicos na bobina que serão captados e direcionados ao VGA por outro conector SMA para serem amplificados e enviados ao RF ADC, que os converterá em sinais digitais e os enviará pelo conector HSMC para outro componente do Sistema de Ressonância Magnética para processamento e geração de Imagens de Ressonância Magnética. Cada canal é desenvolvido de forma que seja possível realizar "loopback" entre RF DAC e RF ADC através de um cabo ligando os respectivos conectores SMA.

Para este trabalho foram testadas 3 diferentes versões:

- **Versão 2.0** – com 2 RF DACs de 14 bits e 2 RF ADCs também de 14 bits;

- **Versão 3.0** – com 2 RF DACs de 14 bits e 2 RF ADCs de 16 bits;

- **Versão 3.6** – com 4 RF DACs de 14 bits (interface mais atual desenvolvida pelo CIERMag que separa os módulos transmissores dos receptores).

2.2 Field-Programmable Gate Array (FPGA) e "System-on-a-chip" (SoC)

Em tradução livre, FPGA significa "arranjo de porta programável em campo". Ou seja, é um circuito integrado de células lógicas reconfiguráveis através de HDL, de forma que as funções lógicas e os roteamentos de comunicação podem ser rearranjados pelo usuário. O método utilizado pela Altera Corp (fabricante da "Arrow SoCKit Evaluation Board" – utilizado no

desenvolvimento deste projeto) para implementação de funções lógicas é o "Look-Up Table" (LUT), que é basicamente uma tabela que determina a saída para dada entrada.

Isso significa que a plataforma de desenvolvimento Quartus gerencia a síntese de um circuito digital escrito em um tipo de HDL (podendo ser VDHL, Verilog ou diagrama de blocos), convertendo-o para portas lógicas (AND, OR, XOR), registradores, memórias, flip-flops, funções aritméticas *etc.* Nesse processo, as portas lógicas são separadas em blocos lógicos e os sinais são roteados através de estruturas de chaveamento, levando em conta os endereços dos periféricos de entradas e saídas a fim de mapear o circuito de maneira estratégica entre pinos de entrada (por exemplo de um botão) e pinos de saída (por exemplo de um LED). (8)

O kit de desenvolvimento utilizado possui a FPGA **Cyclone V SoC 5CSXFC6D6F31** (9) e um processador com arquitetura "Advanced RISC Machine" (ARM) **Dual-core Cortex-A9**, permitindo a instalação e execução de sistemas operacionais como **Linux LXDE Desktop versão Linaro**. Com essa tecnologia, é possível então desenvolver um *software* gerador de ondas para testar os módulos conversores por intermédio da FPGA.

2.3 Hardware Sintetizado

Desenvolveu-se (4) pelo Quartus os circuitos digitais necessários para o processamento dos sinais digitais, considerando o fluxograma da Figura 2, subdividindo-os em blocos de **transmissão**, que *envia* a forma de onda para o conversor RF DAC, **recepção** (Figura 3 A), que *recebe* a forma de onda processada pelo RF ADC, e de **controle do VGA**, que determina o ganho com que o amplificador do módulo multiplicará a forma de onda do RF ADC. Todo o projeto foi desenvolvido utilizando como base a comunicação "Serial Peripheral Interface" (SPI), sincronizada por um gerador de *clock* que controla o roteamento dos sinais. O protocolo SPI trabalha com direção fixa definida (p. e. pinos exclusivos de saída), o que demanda maior cuidado no roteamento, todavia sempre trabalha em *Full-Duplex* (ou seja, a transmissão de dados ocorre em dois sentidos simultaneamente; p. e. o transmissor envia 1 bit para o receptor ao mesmo tempo que esse receptor envia 1 bit para aquele transmissor), atingindo altas velocidades comunicação.

O super-bloco **Transmissão** é responsável por receber os dados (pontos da forma de onda e outras variáveis como valor de ganho e comandos de controle do pulso) do *software* (via processador HPS) e transmiti-los para o módulo através do circuito digital (sintetizado na FPGA). Inicialmente, o sinal é tratado pelo sub-bloco **"Envia"**, conectado ao sub-bloco "RAM RF", que armazena todos os pontos do pulso gerado pelo *software* (p.e. um período da função sinc, além de outras variáveis como frequência do pulso e comandos de controle) e ao sub-bloco "Controle VGA" que armazena e envia o valor de ganho ao VGA do módulo; é o sub-bloco "Controla RAM" que em toda borda de subida do clock da FPGA recebe tais comandos e determina se os pontos serão gravados nos endereços de memória (geração de novos pulsos), apenas lidos a

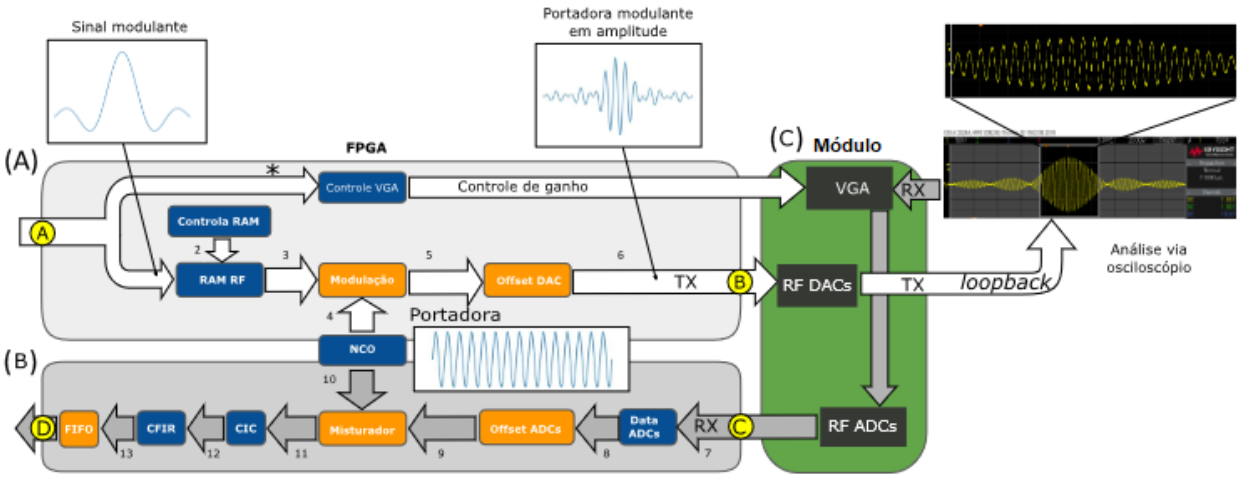

Figura 3 – Projeto sintetizado pela FPGA com subdivisão em blocos de (A) transmissão e (B) recepção. (C) Módulo conversor de RF desenvolvido pelo CIERMag de forma genérica.

Fonte: BATISTA (4)

partir do já está gravado (*loop* de pulsos) ou apagados (nenhum pulso é transmitido). Quando o "RAM RF" identificar que deve transmitir, o pulso é direcionado para o sub-bloco "Modulação", que multiplica seus pontos por uma função moduladora, criando as envoltórias de onda (gerada pelo sub-bloco "NCO" a partir de comandos também presentes nos dados recebidos do *software* – p. e. função seno) e direcionando o sinal ao sub-bloco "Offset DAC", que converte os pontos do formato *complemento de 2* (apropriado para o processador) para *binário com offset deslocado* (apropriado para os chips conversores) e envia para o módulo RF DAC.

O super-bloco **Recepção**, por sua vez, é responsável por receber os dados do módulo e transmiti-los para o *software*. Consequentemente, o sinal é tratado pelo sub-bloco **"Data ADCs"**, recebe os dados do módulo RF ADC, direcionando-os para o "Offset ADCs", que os converte de volta de binário com offset deslocado para complemento de 2; o sinal então é passado para o sub-bloco "Misturador", que multiplica cada ponto da forma de onda pela mesma função do NCO que o modulou (seno), gerando um sinal misturado (com 1/2 da amplitude do sinal original), e os direciona para filtros passa-baixa que operam em conjunto, o primeiro no sub-bloco "Filtro CIC" (de baixa latência, que remove sinais com frequências superiores a 1 MHz, porém diminui a amplitude para 1/4 do sinal original) e o segundo no sub-bloco "Filtro CFIR" (de alta latência, porém remove frequências superiores a 1 MHz que o CIC deixa passar) (10); por fim, os pontos da forma de onda reconstituída são enviados ao módulo "FIFO" que armazena os dados do teste em uma fila e os envia para o *software* (via processador) através do bloco **"Recebe"**.

As etapas realizadas pelo "Modulador" e pelo "Misturador" aumentam a resistência do sistema a ruídos e interferências e possibilitam a transmissão de mais informações simultaneamente.

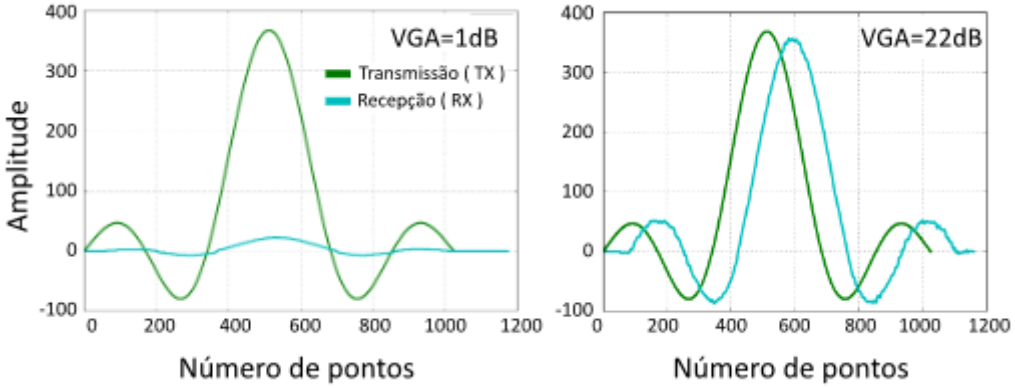

Figura 4 – Diferença entre 2 experimentos com ganhos diferentes aplicados no conversor RF ADC exemplificando a diferença em amplitude entre o sinal enviado para o módulo RF DAC e recebido pelo RF ADC.

Fonte: BATISTA (4)

A necessidade dos filtros para reconstrução dos sinais é um dos motivos por que é necessário o ganho controlado pelo VGA e aplicado sobre o sinal do RF ADC, como demonstrado pela Figura 4. Além disso, previamente à transmissão, é comum a modulação **em amplitude** (AM, que mantém a frequência constante enquanto há variação em amplitude junto ao sinal modulante) em processamento de sinais.

2.4 Software

Seu código-fonte é composto por uma mistura de programações em linguagens Python e C, a primeira por sua facilidade de aprendizado (11) e por possuir uma robusta biblioteca e ferramentas auxiliares de desenvolvimento para interfaces gráficas (12) e a segunda por possuir técnicas de mapeamento de memória (13), necessárias para a conexão do processador HPS com a FPGA.

A primeira etapa para utilização do *software* do STDI envolve gerar o arquivo "soc_system.rbf" (que é um "Quartus II RAW Binary Data"), a partir do projeto sintetizado na FPGA, para garantir a integridade de comunicação entre o *hardware* sintetizado e o processador HPS. O projeto é único, mas cada versão a ser testada demanda algumas especificidades que são alteradas manualmente antes da compilação pelo Quartus. Uma vez concluído o processo de síntese e geração do arquivo ".rbf", ele deve ser colocado no cartão Micro SD que contém o sistema operacional a ser executado pelo embarcado.

Concluída essa configurado, é possível conectar o módulo a ser analisado ao kit embarcado, inicializar o kit de desenvolvimento, que disparará o sistema operacional, e realizar os testes com apoio de periféricos como monitor, mouse e teclado. A execução do *software* deve ser feita pelo terminal, dentro da pasta em que está gravado e rodar o arquivo principal, escrito em Python e

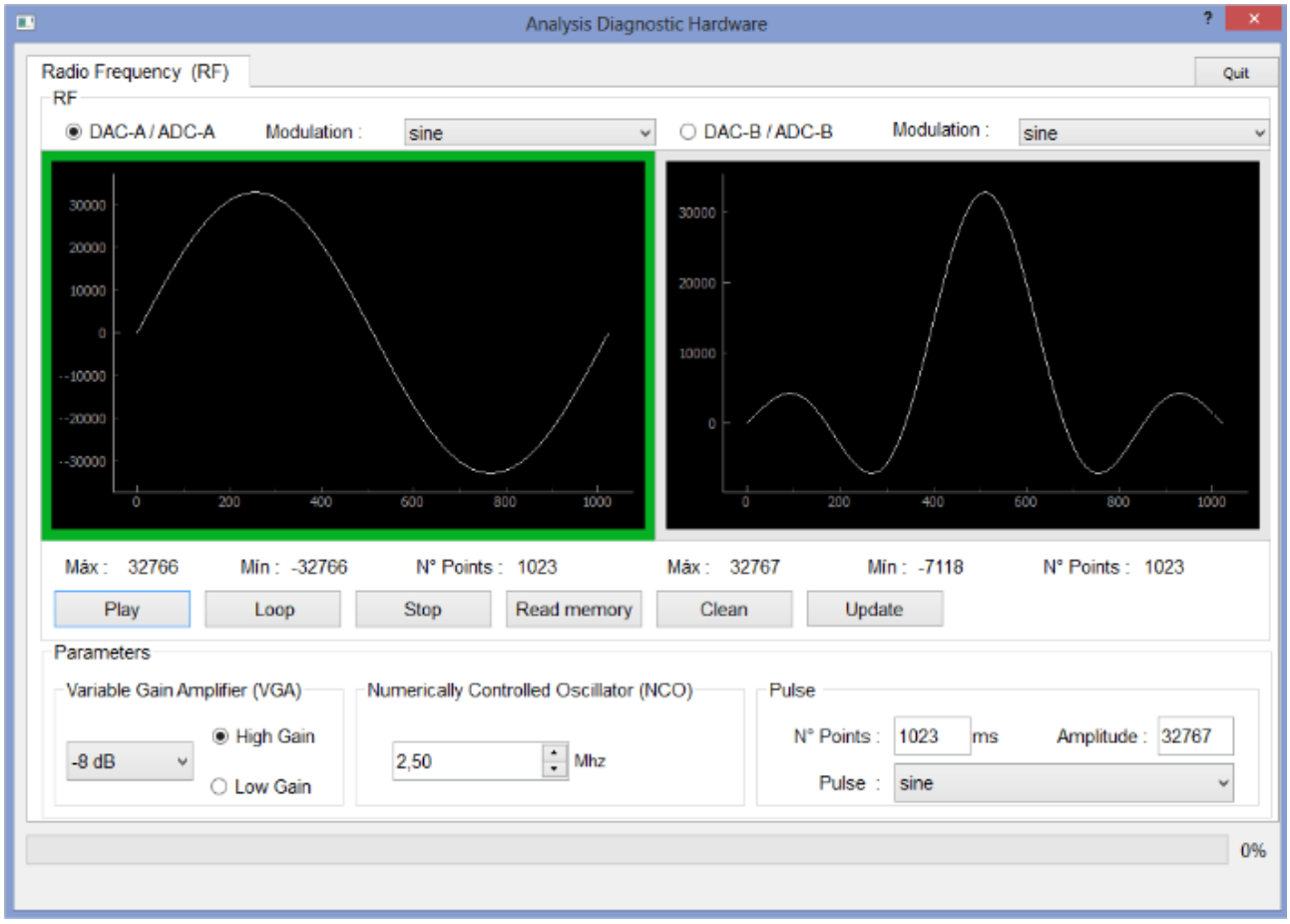

Figura 5 – Interface gráfica do *software* parte do STDI.

Fonte: Elaborada pela autora.

nomeado “principal.py”, acionando a interface gráfica representada na Figura 5.

O *software* desenvolvido (4) permite a escolha do canal do módulo conversor que será testado (“DAC-A/ADC-A” para o canal A, “DAC-B/ADC-B” para o canal B), da função modulante (“Modulation”), do pulso de RF a ser gerado (“Pulse”) – com possibilidade de determinação do número de pontos (“No Points”) e amplitude (“Amplitude”) –, do clock do NCO em MHz (“Numerically Controlled Oscillator”), do ganho em dB aplicado ao chip VGA do módulo (“Variable Gain Amplifier”), e dos comandos ligados ao bloco “Controla RAM” do *hardware*: “Play” para gerar o pulso de RF, gravando seus pontos na memória “RAM RF” e disparando-os uma vez para o módulo, “Loop” para dispará-los em *loop* na taxa de *clock* de 50 MHz (compatível com os chips conversores de RF e permitindo a observação do sinal por um osciloscópio, como exemplificado na Figura 3), “Stop” para cessar o *loop*, “Read Memory” para ler os pontos gravados e poder dispará-los em *loop* novamente (caso “Stop” tenha sido acionado anteriormente), “Clean” para reiniciar todos os parâmetros e apagar os pontos gravados no bloco “RAM RF” e “Update” para atualizar os parâmetros mostrando os valores que foram

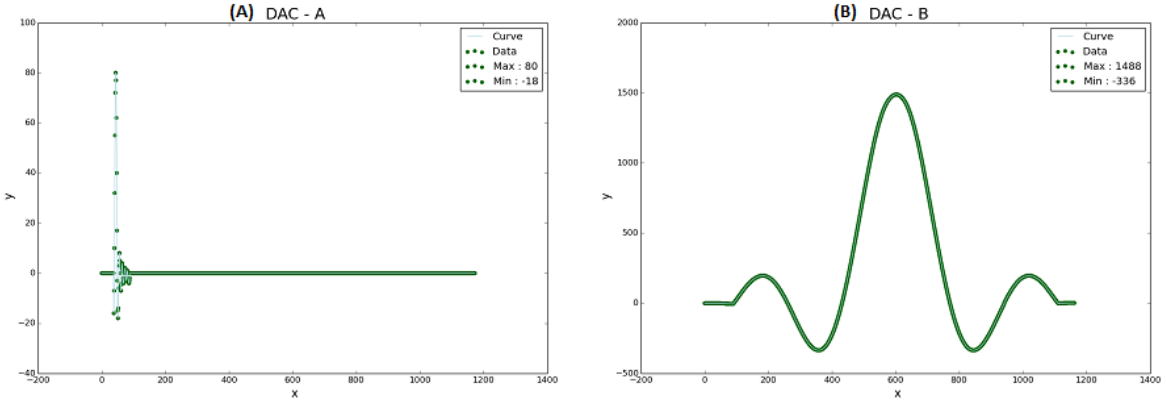

Figura 6 – Gráficos gerados a partir dos sinais recebidos do conversor RF ADC, sendo x o número de pontos retornados e y a amplitude. (a) Gráfico não compatível com pulso gerado para o canal A, podendo indicar falha no funcionamento do conversor RF ADC ou de ambos conversores, e (b) gráfico compatível com pulso gerado para o canal B, indicando funcionamento adequado de ambos conversores de RF.

Fonte: Elaborada pela autora.

carregados no *hardware*, caso haja alguma mudança não concretizada.

Uma vez que o pulso é enviado para o módulo e recebido em "loopback", o *software* lê os pontos do sinal recebido e plota um gráfico, como exemplificado na Figura 6, permitindo um diagnóstico visual, de forma que se deve observar nesse gráfico a mesma forma de onda definida em "Pulse".

3 RESULTADOS

3.1 Revisão do STDI

Após a escolha de qual versão será testada (para fins de revisão e controle escolheu-se a versão 2.0) e configuração dos equipamentos, é feita a montagem experimental representada pela Figura 7.

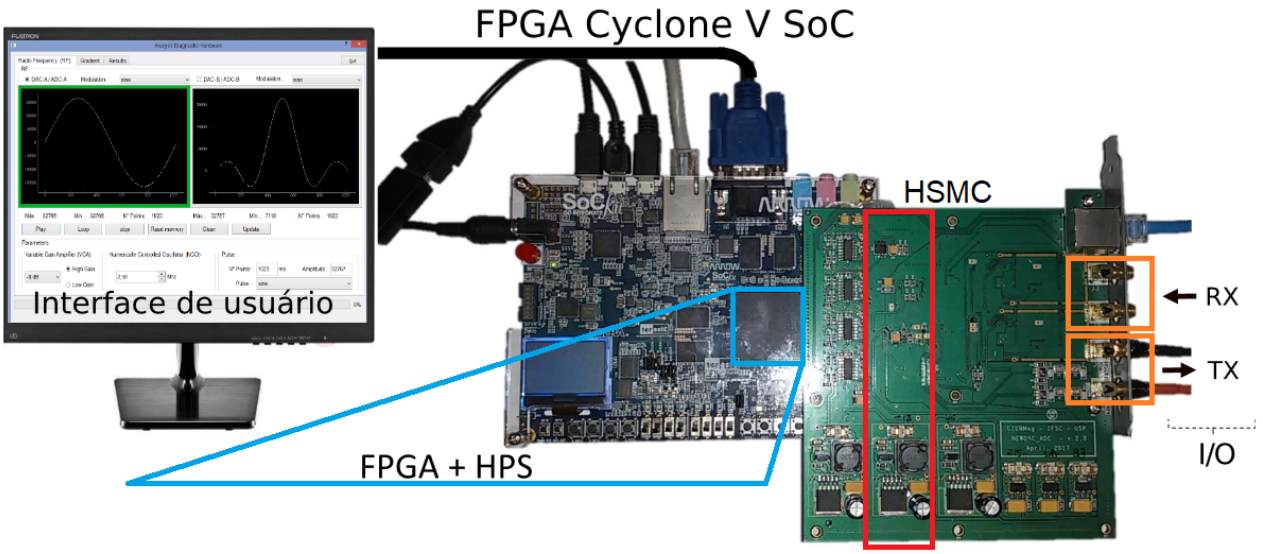

Figura 7 – Montagem experimental para teste da versão 2.0 do módulo de conversão de RF.

Fonte: BATISTA (4)

Na primeira tentativa, utilizando o STDI em *backup*, a interface gráfica carregava, porém ao tentar enviar um pulso de RF ao módulo 2.0, a barra de progresso localizada no canto inferior da interface representada na Figura 5 não carregava completamente. Ao analisar o terminal, as mensagens mais importantes encontradas, apresentadas no Listing 1, indicavam falhas de comunicação entre *software* e *hardware*, possivelmente no controle dos periféricos I/O do HPS e da FPGA.

Listing 1: Mensagens retornadas pelo terminal ao tentar carregar a interface gráfica do *software* do STDI.

```
sockit@sockit-ubuntu-core:~/Desktop/Software_22_04_2020 python
   principal.py
libEGL warning: DRI1: failed to authenticate
ERROR: could not open "/dev/mem"...
ERROR: mmap() failed...
Segmentation fault
```

Outras mensagens também retornadas no terminal durante as tentativas de uso do *software*, indicavam que nenhum dado estava sendo passado para a FPGA ao longo da execução, o que indicaria possíveis falhas de conexão na administração do fluxo de dados, no gerenciamento e distribuição de tarefas (funções) ou na manipulação dos arquivos que armazenam os pontos da forma de onda a ser pulsada.

Por esse motivo, foi feita uma revisão completa de todos os arquivos do *software*, com foco nas bibliotecas e funções utilizadas tanto pela parte escrita em Python (14), quanto pela escrita em C (13), a fim de encontrar conexões quebradas, funções desatualizadas, erros de acesso de variáveis *etc* e corrigi-los, garantindo a comunicação entre o *software* (via HPS) e a FPGA, podendo validá-las com os gráficos retornados digitalmente pelo STDI e com os gráficos gerados de forma síncrona pelo osciloscópio, como exemplificado nas Figuras 6 e 7, respectivamente.

3.2 Reestruturação para abrangência de novas versões de interface

3.2.1 Automatização da alternância entre versões do projeto

Como as especificidades de cada interface precisavam ser alteradas manualmente, a primeira inovação foi gerar *scripts* na linguagem *Tool Command Language* (TCL) para automatizá-las quando necessário testar diferentes interfaces (p. e. testar diversos módulos na versão 2.0 e na versão 3.0).

A mais importante diferença entre as versões é a atribuição de pinos dos conectores HSMC, responsáveis pela comunicação entre FPGA e módulo conversor, como exemplificado na Figura 8. É imprescindível o roteamento correto (p. e. o pino "DAB_D15" (Figura 8 a) é conectado ao "HSMC_TX_p9" (Figura 8 b), identificado por "D2" no *Pin Planner* do Quartus), uma vez que sinais trocados podem danificar a interface ou a FPGA.

3.2.2 Criação de novos blocos para abrangência de novas versões

Uma vez que as alterações necessárias para a mudança de versões são automatizadas, é possível manter um circuito digital genérico (p. e. considerando simultaneamente as arquiteturas das versões 2.0, 3.0 e 3.6 DAC, seriam necessários 4 blocos de transmissão e 2 blocos de recepção), em contrapartida à criação de circuitos digitais diferentes para versões diferentes (p. e. para a versão 2.0 seriam necessários 2 blocos de transmissão e 2 de recepção, enquanto para a versão 3.6 DAC, apenas 4 blocos de transmissão), visto que os princípios de funcionamento dos módulos conversores se mantém constantes, tornando tal abordagem desnecessária.

Com o objetivo de que o mesmo projeto do Quartus funcione perfeitamente para todas as versões de interfaces de teste, além da criação de novos blocos, são necessárias alterações para que determinados blocos sejam ou não utilizados na síntese do circuito digital a depender da

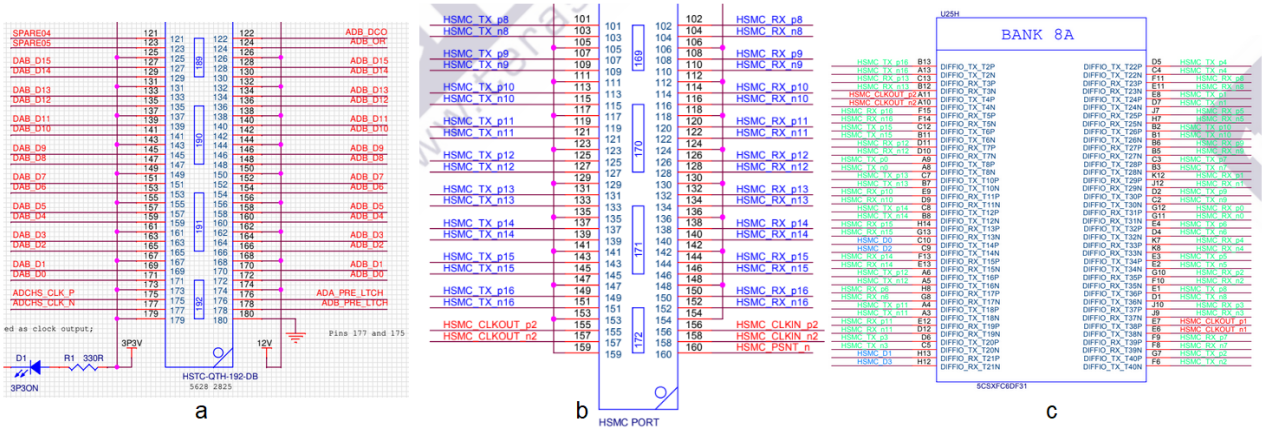

Figura 8 – Esquemas elétricos de alguns pinos (a) dos chips conversores ligados ao conector HSMC do módulo; (b) dos pinos da FPGA ligados ao conector HSMC do embarcado; e (c) correspondência de nomenclaturas dos pinos da PFGA no esquema elétrico e na interface do Quartus.

Fonte: Elaborada pela autora com esquemáticos do CIERMag e da Altera.

versão que se deseja analisar (p. e. quando compilando o projeto para a versão 2.0, apenas 2 blocos TX e 2 blocos RX serão utilizados; já para a versão 3.6 DAC, apenas 4 blocos TX).

Isso significa que, independente das alterações feitas para adequação à versão 3.6 DAC, quando carregado o script para a 2.0, o STDI deve se manter funcional, de forma que, caso não ocorra, isso significa que as implementações não foram feitas da forma correta.

Após as alterações, o projeto foi compilado com sucesso, porém quando se realizaram testes pelo *software*, foi possível observar que, apesar de os dados serem passados corretamente para o conversor RF DAC (confirmado com o osciloscópio), o mesmo não acontecia com o RF ADC, gerando resultados análogos ao da Figura 6 A. Sabendo que esse resultado indica falha de conexão entre os conversores, e que a interface usada para o teste estava perfeitamente funcional, levantou-se a hipótese de que o roteamento de sinais na FPGA poderia não ter sido implementado corretamente devido às alterações.

Dessa hipótese, surgiu a necessidade de analisar o diagrama "Register Transfer Logic" (RTL), que é sempre gerado após a compilação de um projeto no Quartus e demonstra todos os pinos I/O, os blocos programados e roteamentos que foram implementados na FPGA (15). Na Figura 9 A, os pinos são representados como pentágonos azuis, os blocos como retângulos verdes e os roteamentos como linhas pretas.

Os blocos de programação divididos em sub-blocos podem ser expandidos para visualização mais detalhada, como demonstrado na Figura 9 B, que evidencia que os sub-blocos de transmissão adicionados (identificados nesse nível do projeto como "RF_Channel") não estavam sendo implementados (deviam ser geradas 4 instâncias distintas e observa-se apenas 2), o que poderia ser a causa do mau funcionamento.

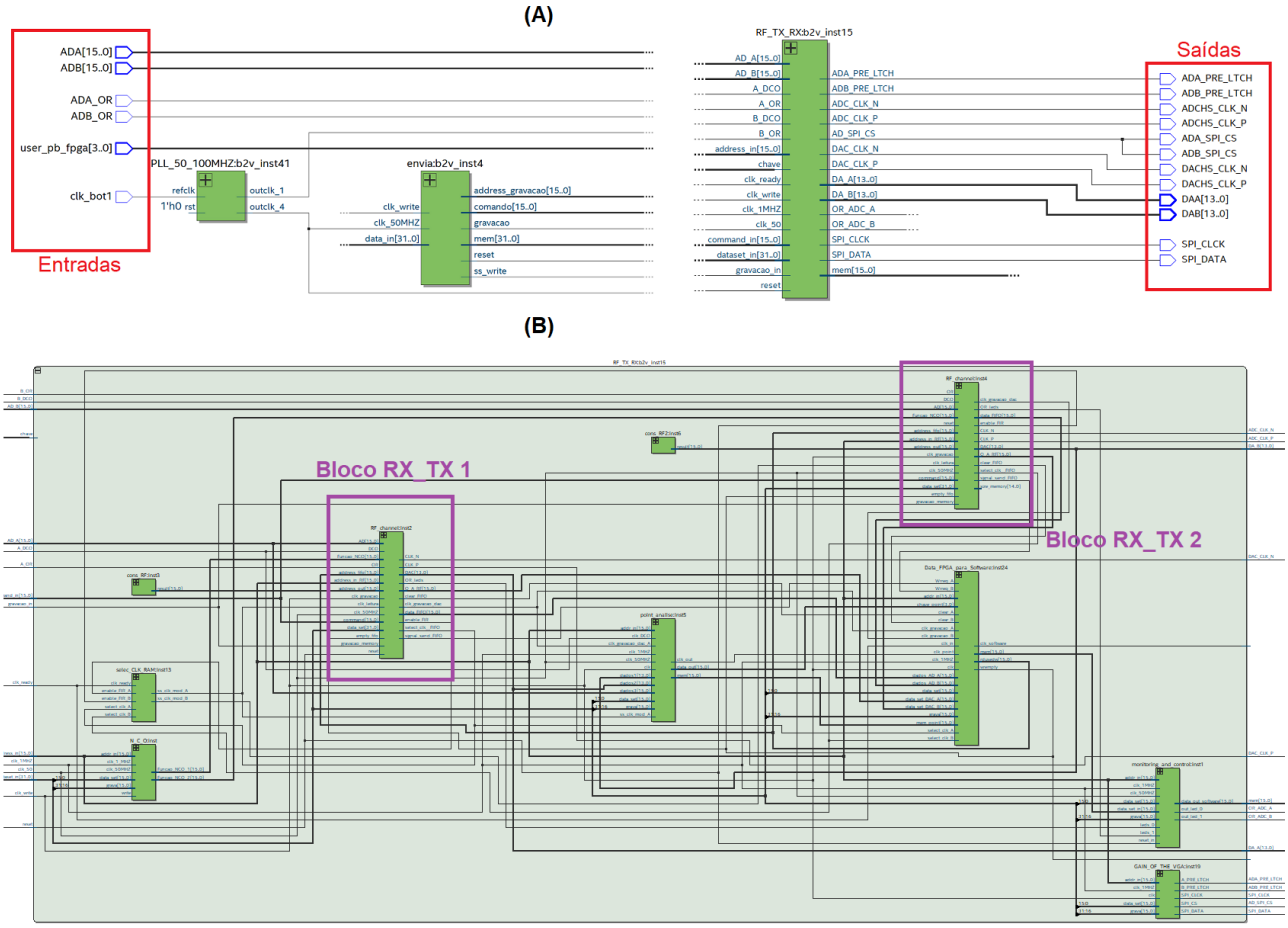

Figura 9 – Diagrama RTL gerado após compilação do projeto no Quartus, em que se pode observar: (A) os pinos de entrada e saída e suas conexões com blocos de exemplo "PLL" (gerador de *clock*) e "envia" (gerenciador de conexões entre *software* e FPGA) e destaque para o bloco de interesse "RF_TX_RX" (super-bloco de transmissão e recepção); e (B) "RF_TX_RX" expandido apresentando apenas 2 sub-blocos "RF_Channel".

Fonte: Elaborada pela autora.

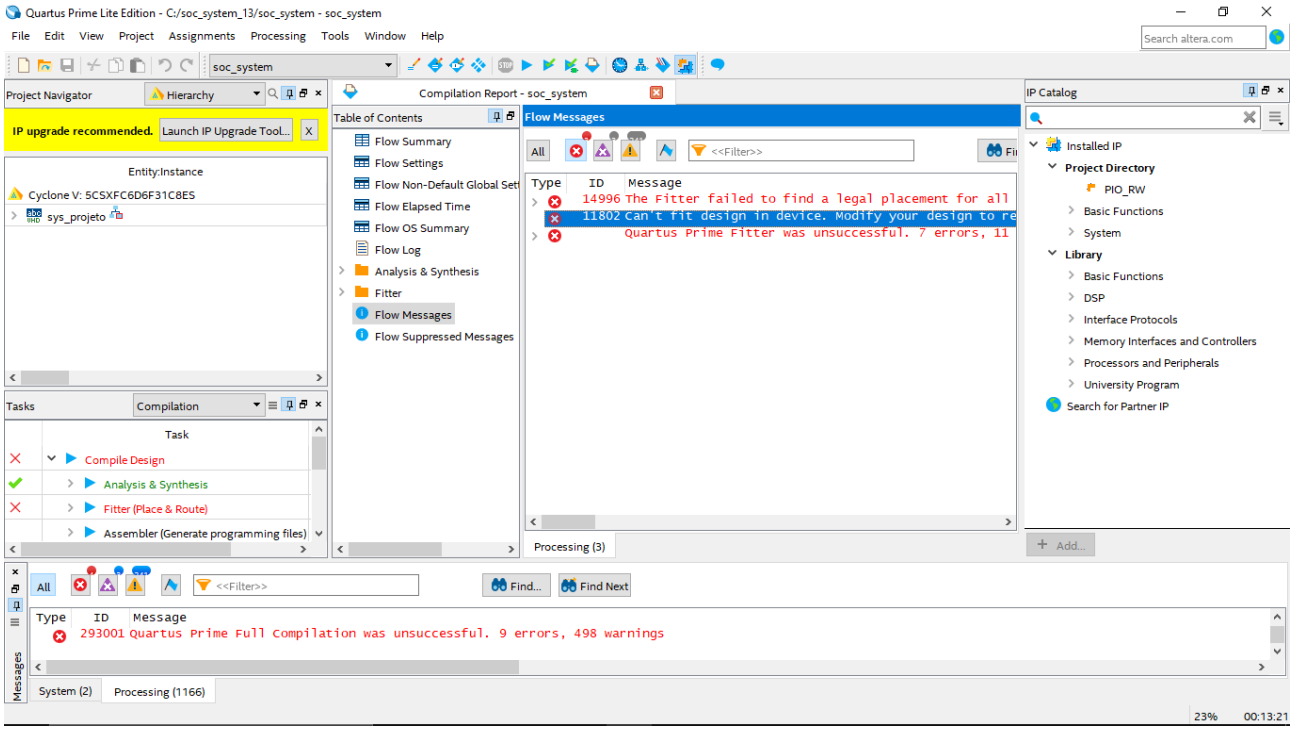

Figura 10 – Mensagem de erro "Error (11802): Can't fit design in device. Modify your design to reduce resources, or choose a larger device. The Intel FPGA Knowledge Database contains many articles with specific details on how to resolve this error." ao tentar compilar o projeto para a FPGA.

Fonte: Elaborada pela autora.

Todas as conexões entre os blocos foram revisadas e identificou-se a necessidade de implementação mais HDL garantindo as conexões corretas. O resultado então obtido foi a impossibilidade de síntese do circuito digital devido ao novo número de blocos exceder a quantidade de portas lógicas disponíveis na *Cyclone V*, como demonstrado na Figura 10. Isso acontece porque, ainda que os blocos adicionais não sejam utilizados em todas as versões, eles são implementados no projeto de maneira isolada e sem conexão (roteamento) com o circuito funcional. Para que isso não aconteça, projetos específicos a cada versão deveriam ser desenvolvidos, o que iria contra o objetivo inicialmente traçado.

3.2.3 Manutenção da quantidade de blocos

Essa abordagem limita o STDI a testar apenas 2 canais por vez, ainda que a interface de interesse possua mais (p. e. versão 3.6 com 4 canais DAC), e gera certo desperdício de aproveitamento das portas lógicas (p. e. serão implementados 2 blocos de recepção mesmo quando o objetivo for testar a versão 3.6 DAC), porém é uma abordagem genérica o suficiente para abranger o objetivo e, principalmente, abrir novas possibilidades.

No escopo proposto por este trabalho, todas as alterações necessárias já foram introduzidas

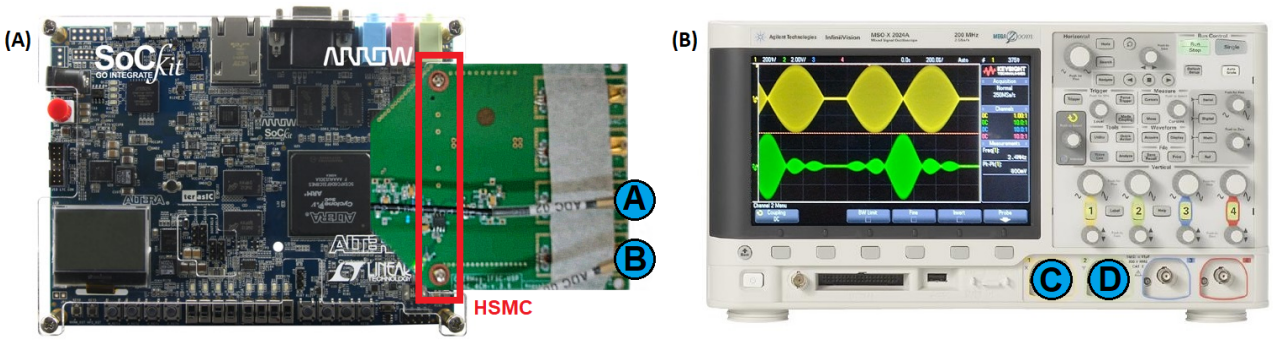

Figura 11 – Montagem experimental para testes da versão 3.6 DAC do módulo conversor de radiofrequência em que: (a) mostra o módulo ligado à FPGA e (b) mostra os resultados observados via osciloscópio.

Fonte: Elaborada pela autora.

na Seção 3.2.1, e os resultados obtidos são apresentados na Figura 11, em que os conectores do módulo (A) e (B) são ligados com os conectores do osciloscópio (C) e (D), respectivamente, demonstrando sucesso de implementação ao ser possível observar as formas de onda configuradas (analogamente à configuração da Figura 5) para os canais 3 e 4 em teste.

4 CONCLUSÕES E CONSIDERAÇÕES FINAIS

Para este trabalho foi necessário profundo entendimento do funcionamento de um projeto abrangente de instrumentação em física envolvendo *hardware*, *hardware* sintetizado (digital), *firmware*, *software* e interfaces gráficas para que alterações fossem efetivamente realizadas.

Primeiro, para compreender o funcionamento dos módulos conversores de radiofrequência (*hardware*), foi necessário estudar a teoria por trás de Imagens por Ressonância Magnética (MRI) e suas implicações no escopo do projeto, de esquemas elétricos com os roteamentos de sinais e dos princípios de funcionamento dos chips e componentes utilizados na arquitetura, com destaque para particularidades envolvendo voltagens limite nas trilhas que impactam diretamente nas conexões e integrações pertinentes ao STDI.

Segundo, para compreender e modificar o projeto de *hardware* sintetizável desenvolvido com o auxílio da plataforma Quartus, foi necessário estudar o funcionamento e programação de FPGAs, as linguagens de descrição de *hardware* (HDL), as formas de *debugging* pertinentes à análise dos blocos e roteamentos sintetizados, além do projeto propriamente dito: as lógicas por trás dos blocos sintetizados e impactos no processamento dos sinais.

Terceiro, para compreender e modificar o *software*, foi necessário estudar as linguagens de programação Python (14) e C (13) e algumas de suas bibliotecas, o desenvolvimento de programas de computador e as integrações entre arquivos e bibliotecas, a comunicação entre *software* e processador (*firmware*) com as particularidades e restrições de acesso e escrita em memória e as interações de variáveis do programa com a interface gráfica para execução de comandos (para que as funções corretas sejam acionadas após o clique em um botão).

Dessa forma, inovações importantes foram implementadas no STDI, atingindo o objetivo de generalizar um sistema, antes específico a antigas versões de módulos conversores de RF, possibilitando a adaptação para novas versões.

4.1 Possibilidades futuras

Ainda que haja, no final deste trabalho, algumas restrições (como explicado na Seção 3.2.3, a nova estruturação possibilita modificações que as resolveria, tais como:

1. Implementar multiplexadores, no *hardware* sintetizado, entre os sinais dos módulos de conversão e da FPGA (via conector HSMC), a fim de selecionar dinamicamente os pinos

referentes ao canal que se deseja testar (p. e. canal A da versão 2.0), evitando a restrição a apenas 2 canais de teste (visto que a FPGA utilizada comporta apenas dois super-blocos de transmissão e dois de recepção);

2. Adicionar pinos digitais de recepção, no *hardware* sintetizado, para que versões sem receptores consigam ser validadas sem a necessidade de um osciloscópio (uma vez que, no atual projeto de *hardware* sintetizável, nenhum sinal seria passado para o receptor pelo transmissor na versão 3.6 RF DAC, pela impossibilidade de *loopback* via conector analógico, que é necessário para o completo funcionamento do diagnóstico através do *software*, pela falta de *chips* conversores ADC de radiofrequência);

3. Adicionar, também, pinos digitais de transmissão, para que versões sem transmissores sejam validadas sem a necessidade de um gerador analógico de forma de onda (analogamente ao explicado no item anterior, nenhum sinal seria passado pelo transmissor para o receptor em versões com apenas conversores RF ADCs).

REFERÊNCIAS

1 KATTI, Girish; ARA, Syeda Arshiya; SHIREEN, Ayesha. Magnetic resonance imaging (mri)–a review. *International Journal of Dental Clinics*, v. 3, n. 1, p. 65–70, 2011.

2 TIROTTA, Ilaria *et al.* 19f magnetic resonance imaging (MRI): from design of materials to clinical applications. *Chemical Reviews*, v. 115, n. 2, p. 1106–1129, 2015.

3 UNIVERSIDADE DE SÃO PAULO. Unidade de Apoio, Pesquisa e Desenvolvimento de Instrumentação Agropecuária (São Carlos). Alberto Tannús; Edson Luiz Géa Vidoto; Mateus José Martins. *Espectrômetro para uso em sistemas de ressonância magnética e sistema de ressonância magnética*. BR 10 2015 000624-1, 12 jan. 2015, 19 jul. 2016.

4 BATISTA, João Carlos. *Teste e diagnóstico de interfaces utilizando FPGA com reprogramação dinâmica e software embarcado para o Espectrômetro Digital de Ressonância Magnética da CIERMag. 2020*. 131 f. Tese (Mestrado Profissional em Matemática, Estatística e Computação Aplicadas à Indústria) — Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2020.

5 WEISHAUPT, Dominik *et al. How does MRI work?: an introduction to the physics and function of magnetic resonance imaging*. 2nd ed. Berlin: Springer, 2006.

6 LISANTI, Christopher J; HASHEMI, Ray H; BRADLEY, William G. *MRI: The basics*. Philadelphia: Lippincott, Williams & Wilkins, 2018.

7 MARTINS, Mateus Jose. *Desenvolvimento de um tomógrafo de ressonância magnética*: integração e otimização. 1995. 89 f. Tese (Doutorado em Física-Aplicada) — Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos, 1995.

8 PEDRONI, Volnei A. *Digital electronics and design with VHDL*. Burlington: Morgan Kaufmann, 2008.

9 TERASIC TECHNOLOGIES. *Cyclone V SoC development kit and SoC embedded design suite.* Disponível em: https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=Englishamp;CategoryNo=205amp;No=819amp;PartNo=3. Acesso em: 30 abr. 2022.

10 MEDDINS, Robert. *Introduction to digital signal processing*. Oxford: Newnes, 2000.

11 SRINATH, KR. Python–the fastest growing programming language. *International Research Journal of Engineering and Technology*, v. 4, n. 12, p. 354–357, 2017.

12 QT. The Qt Company Ltd. Disponível em: https://doc.qt.io/qtforpython/index.html. Acesso em: 16 fev. 2022.

13 ALMEIDA, Rodrigo Maximiano Antunes de; MORAES, Carlos Henrique Valério de; SERAPHIM, Thatyana de Faria Piola. *Programação de Sistemas Embarcados: Desenvolvendo Software para Microcontroladores em Linguagem C*. Rio de Janeiro: Elsevier Brasil, 2017.

14 PYTHON 3.11.0. Disponível em: https://docs.python.org/3/. Acesso em: 16 fev. 2022.

15 PEDRONI, Volnei A. *Circuit design with VHDL*. Cambridge: MIT press, 2020.