Guilherme Monteiro Henriques Santos

Do Material ao Digital:

Implementação de Jogos e Utilitários em Hardware Reprogramável

PROJETO FINAL DE GRADUAÇÃO

DEPARTAMENTO DE INFORMÁTICA
E DEPARTAMENTO DE ENGENHARIA ELÉTRICA
Programa de Graduação em Engenharia de Computação

Rio de Janeiro
Julho de 2025

Pontifícia Universidade Católica
do Rio de Janeiro

Guilherme Monteiro
Henriques Santos

Do Material ao Digital:

Implementação de Jogos e Utilitários em Hardware Reprogramável

PROJETO FINAL DE GRADUAÇÃO

Relatório de Projeto Final, apresentado ao programa de Graduação do Departamento de Informática e Departamento de Engenharia Elétrica da PUC-Rio como requisito parcial para a obtenção do título de Engenheiro de Computação.

Orientador : Prof. Augusto Cesar Espíndola Baffa
Co-orientador: Prof. Felipe Calliari

Rio de Janeiro
Julho de 2025

Guilherme Monteiro Henriques Santos

Graduando em Engenharia de Computação pela PUC-Rio

Ficha Catalográfica

Santos, Guilherme Monteiro Henriques

Do Material ao Digital: / Guilherme Monteiro Henriques Santos; orientador: Augusto Cesar Espíndola Baffa; co-orientador: Felipe Calliari. – Rio de Janeiro: PUC-Rio, Departamento de Informática e Departamento de Engenharia Elétrica, 2025.

v., 84 f: il. color. ; 30 cm

Projeto de graduação - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática e Departamento de Engenharia Elétrica.

Inclui bibliografia

1. Arranjo de Porta Programável em Campo (FPGA);. 2. Hardware;. 3. Linguagem de Descrição de Hardware (HDL);. 4. Lógica Digital;. 5. Printed Circuit Board (PCB).. I. Baffa, Augusto Cesar Espíndola. II. Calliari, Felipe. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática
e Departamento de Engenharia Elétrica. Título.

CDD: 620.11

Para meus pais, irmã, familiares e amigos,
por todo apoio ao longo de minha vida.

Agradecimentos

Primeiramente, gostaria de agradecer aos meus pais, irmã, cunhado e demais familiares, por desde muito cedo, me apoiarem e me incentivarem ao longo de minha caminhada.

Em seguida, agradeço aos professores envolvidos nesse trabalho, cujo direcionamento foi essencial para que os objetivos propostos fossem alcançados.

Estendo meu agradecimento aos demais professores que compuseram toda minha trajetória acadêmica, desde o fundamental até o ensino superior, e que generosamente auxiliaram a construir meu conhecimento.

Por fim, devo agradecer a todos meus amigos, que me acompanharam ao longo da minha vida, tornando cada momento mais leve e significativo.

Resumo

Santos, Guilherme Monteiro Henriques; Baffa, Augusto Cesar Espíndola (Orientador); Calliari, Felipe (Coorientador). **Do Material ao Digital:**. Rio de Janeiro, 2025. 84p. Projeto de Graduação – Departamento de Informática e Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Esse projeto é um estudo com o intuito de compreender e explorar as possibilidades oferecidas pela plataforma FPGA, através do desenvolvimento de programas implementados em Hardware reprogramável. Durante a elaboração do projeto, a linguagem VHDL e um FPGA Cyclone IV foram utilizados para a implementação de jogos inspirados em clássicos como o Pong e Breakout, assim como uma aplicação gráfica focada para desenhar na tela. Paralelamente à relevância da implementação da lógica digital, foi necessário o estudo e construção de uma placa de circuito impresso(PCB) dedicada ao suporte de periféricos como teclado PS/2, saída VGA e saída de áudio. Sendo assim, o projeto abrange tanto aspectos físicos da computação com o desenho e construção do Hardware, quanto desenvolvimento digital por meio de linguagem de descrição de Hardware, resultando em uma plataforma versátil.

Palavras-chave

Arranjo de Porta Programável em Campo (FPGA); Hardware; Linguagem de Descrição de Hardware (HDL); Lógica Digital; Printed Circuit Board (PCB).

Abstract

Santos, Guilherme Monteiro Henriques; Baffa, Augusto Cesar Espíndola (Advisor); Calliari, Felipe (Co-Advisor). **From Material to Digital**. Rio de Janeiro, 2025. 84p. Undergraduate Thesis – Department of Informatics and Department of Electrical Engineering, Pontifícia Universidade Católica do Rio de Janeiro.

This project is a study focused on comprehending and exploring the possibilities offered by a FPGA platform, throughout the development of reprogrammable Hardware implementations. During its progression, the VHDL language and a Cyclone IV FPGA were used to implement classic inspired games such as Pong and Breakout, as well as a graphic tool aimed at on-screen drawing. In addition to the relevance of digital logic implementation, it was necessary to study and elaborate a dedicated printed circuit board(PCB) to support peripherals such as PS/2 keyboard, VGA output and sound output. Thus, the project covers both physical aspects of computing such as Hardware design and construction, and also digital development via Hardware description language, resulting in a versatile system.

Keywords

Digital Logic; Field Programmable Gate Array (FPGA); Hardware Description Language (HDL); Hardware; Printed Circuit Board (PCB).

Sumário

Lista de figuras

Lista de tabelas

Lista de Abreviaturas

CI – Circuito Integrado

CLB – *Configurable Logic Blocks*

CPU – *Central Processing Unit*

FPGA – *Field Programmable Gate Array*

HDL – *Hardware Description Language*

ISA – *Instruction Set Architecture*

IEEE – Instituto de Engenheiros Eletricistas e Eletrônicos

LUT – *Look Up Table*

PCB – *Printed Circuit Board*

PLD – *Programmable Logic Device*

PS/2 – *Personal System/2*

RAM – *Random Access Memory*

RGB – *Red, Green, Blue*

ROM – *Read Only Memory*

ROM – *Register-Transfer Level*

VGA – *Video Graphics Array*

VHDL – *Very High Speed Integrated Circuit Hardware Description Language*

"A vida me ensinou a nunca desistir
Nem ganhar, nem perder, mas procurar evoluir"

Charlie Brown Jr., *Dias de luta, dias de glória.*

1
Introdução

Desde os primeiros computadores analógicos, que ocupavam muito espaço e utilizavam válvulas termiônicas como o *UNIVAC*(1), até os mais recentes sistemas embarcados que cabem em um bolso, é notável a evolução vertiginosa e veloz que a computação levou. Os avanços no campo da eletrônica, como a criação e desenvolvimento de transistores, permitiram a elaboração de circuitos integrados. Com o aumento do número de transistores por CI, a computação foi ganhando proporções e capacidades cada vez maiores, até que em 1971 foi criado o primeiro processador pela Intel, o 4004(2), que contava com 2300 transistores.

Durante esse mesmo período, começam a surgir ferramentas como os *PLD's*, que possibilitavam a configuração de um circuito lógico através de uma linguagem de descrição de *Hardware*. No entanto, na época, tais dispositivos eram limitados por suas propriedades a serem programados apenas uma vez. Então em meio a 1985, é lançado o primeiro *FPGA*, o Xilinx XC2064(3), que permitia o usuário reprogramar os circuitos internos do chip, e assim marcando uma nova era da computação lógica digital.

Dessa forma, a flexibilidade e liberdade características dos *FPGA's* proporcionam ao usuário a capacidade de desenvolver sistemas sob medida de acordo com as necessidades, e que caso ocorra alguma alteração nos requisitos, fosse possível reprojetar a arquitetura a fim de atender às novas demandas.

No cenário atual, os *FPGAs* são muito aplicados, no mercado, em sistemas de controle em tempo real, sistemas embarcados, processamento de sinais e computação paralela devido aos pontos citados acima. Já no âmbito acadêmico, esses são cruciais para que o estudante tenha um contato prático com a face do *Hardware* da computação, permitindo aprendizados de lógica, arquitetura de computadores, e desenvolvimento de sistemas digitais.

Dessa forma, esse projeto teve como intuito, explorar o potencial permitido por uma plataforma reconfigurável, assim como construir conhecimentos práticos sobre o desenvolvimento de uma placa dedicada e também desenvolver o entendimento de operação de periféricos como um teclado *PS/2*, e uma saída *VGA*.

Assim, foram desenvolvidos jogos como Breakout, Pong e aplicações

interativas como uma interface de desenho e um sintetizador de som, de forma que esses foram implementados diretamente em *Hardware* através da linguagem *VHDL*. Ademais, para o controle de periféricos, desenvolveu-se módulos de controle específicos para a interpretação dos sinais e comunicação com tais. Por fim, pode-se pontuar a construção de uma *PCB* inteiramente dedicada para o projeto com conectores de áudio, vídeo e teclado, que permitiram a integração do *FPGA* com o sistema de periféricos.

2
Situação Atual

Levando em consideração a perspectiva educacional, tem-se projetos nos quais alunos de graduação usaram *FPGAs* para a implementação de arquiteturas clássicas. A exemplo disso, tem-se o projeto de Andreeti(4), que estudou profundamente a arquitetura do processador 6502, desde a forma que esses fazia o endereçamento até o conjunto de instruções, ou ISA, desse.

Após esse estudo e análise, o autor implementou o processador usando a linguagem *VHDL* para sintetizar os circuitos que compunham o componente. Para que fosse possível testar a implementação, esse utilizou um kit de desenvolvimento da Digilent, especificamente o Nexys 3. Após finalizada essa etapa, ele integrou sua *CPU* em *VHDL*, a um projeto de síntese de um *Nintendo Entertainment System*, a fim de demonstrar a funcionalidade do que ele construíra.

Apesar de tal projeto abordar de forma profunda a compreensão do funcionamento de uma *CPU*, e todas as etapas para implementa-lá em um sistema reconfigurável, tem-se pontos interessantes que esse projeto não explorou. A exemplo disso, pode-se mencionar a construção de um *Hardware* específico e o controle de periféricos usando o próprio *FPGA*. Sendo assim, o presente projeto visa trazer uma abordagem mais voltada ao baixo-nível incentivando o contato com a eletrônica, promovendo a construção de uma placa dedicada ao projeto para a integração de componentes periféricos e a implementação de jogos e aplicações.

2.1
Lógica Digital

Um ponto fundamental para construção desse projeto é o uso de uma área da eletrônica conhecida como lógica digital. Tendo como base o conceito vindo da matemática, a lógica digital consiste em implementar circuitos eletrônicos que operam a partir de dois níveis de tensão distintos. Tipicamente, os níveis são definidos como *Low* e *High*, que no caso do sistema implementado, esses são dispostos como 0 e 3,3 Volts.

No entanto, para que os circuitos operem, como um todo, não basta que apenas representem os níveis lógicos, é necessário que sejam capazes de realizar

operações. Para isso existem CI's responsáveis por realizar as operações de lógicas e suas derivações. Veja as tabelas abaixo, referentes a algumas delas:

E		
Entrada 1	Entrada 2	Saída
0	0	0
1	0	0
0	1	0
1	1	1

Ou		
Entrada 1	Entrada 2	Saída
0	0	0
1	0	1
0	1	1
1	1	1

Não	
Entrada 1	Saída
0	1
1	0

Tabela 2.1: Tabelas verdade de operações lógicas comuns

Além desses componentes conhecidos como portas lógicas, tem-se, por exemplo, operadores conhecidos como multiplexadores, que são responsáveis por receber múltiplas entradas e a definir uma delas como saída a partir de sinais de seleção recebidos. Tais elementos, são parte dos componentes, que podem ser utilizados para a construção de circuitos combinacionais, que é um dos pontos fundamentais da lógica digital.

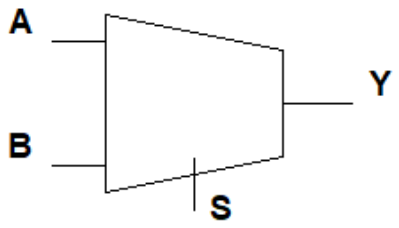

Figura 2.1: Multiplexador de duas entradas e uma saída.

Os circuitos combinacionais são aqueles que a saída não possui atraso em relação as entradas, ou seja, a resposta é direta em tempo real sem a

necessidade de um *clock*. Tal ponto permite o processamento paralelizado de vários desses circuitos, constituindo um dos pontos fundamentais dos sistemas digitais.

Em complemento a lógica combinacional, outra forma de estruturar os circuitos digitais se dá por meio da lógica sequencial. Conforme pontuado por Cruz(5), os circuitos sequenciais são aqueles que as entradas e saídas atuais afetam diretamente as respostas futuras visto que são circuito retroalimentados.

Para isso, existem componentes como *Flip-Flops*, ou registradores, que introduzem o conceito de memória ao sistema. Esses dispositivos conseguem armazenar suas saídas por um período de tempo contínuo, pois possuem entradas que funcionam como sinais de controle, como o sinal *set*, que configura a saída, e o sinal *reset* que redefine a saída para o 0 lógico. Além desses sinais de entrada, existe também o sinal de *clock*, que é crucial para a mudança de estados do *Flip-Flop*. Dessa forma, o *clock* dita o ritmo em que as transições ocorrem, sincronizando os componentes desse sistema.

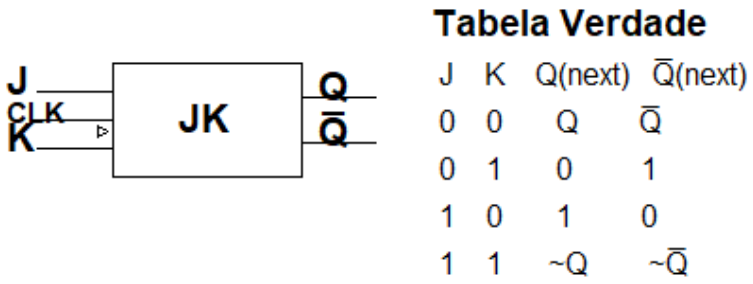

Figura 2.2: *Flip-Flop* J-K.

2.2 VHDL

O *VHSIC Hardware Description Language*, ou *VHDL*, é uma linguagem de descrição de *Hardware* elaborada Departamento de Defesa do Estados Unidos no início da década de 1980. Segundo Hüsemann(6), o *VHDL* foi desenvolvido para que fosse uma linguagem versátil e padronizada levando em consideração os avanços na produção de CIs e no campo da tecnologia.

Poucos anos depois, em 1987, essa linguagem tornou-se padronizada pelo IEEE, e assim ganhando proporções cada vez maiores no mercado. Atualmente o *VHDL*, é uma das principais linguagens de descrição de *Hardware*, sendo muito aplicadas em sistemas que usam *FPGAs*, ou seja, por exemplo em sistemas de controle de indústrias, aplicações de telecomunicações e de

computação paralela. Veja a pesquisa da Siemens(7) referente a presença de diferentes linguagens em projetos:

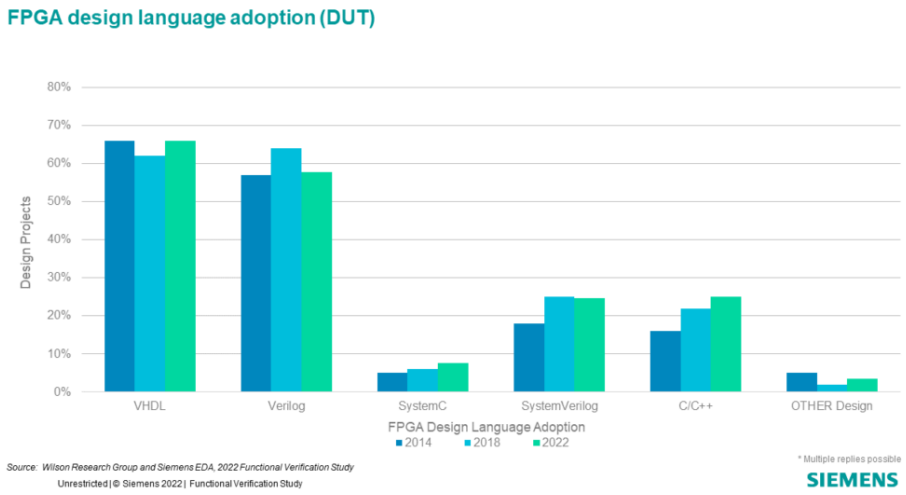

Figura 2.3: Presença de linguagens em projetos com *FPGA*[1]

2.2.1 Componentes da linguagem

O código *VHDL* possui como componentes principais a entidade, que define os dados que entram e/ou saem de um módulo, e a arquitetura, que abriga todo o funcionamento do módulo. Dessa forma, em uma arquitetura é possível declarar processos, que são blocos de códigos atualizados através dos componentes mencionados em sua lista de sensitividade, como uma mudança de borda de *clock*. Assim, através do uso de processos, é possível descrever circuitos sequenciais que serão compostos for estruturas como *flip-flops* ou registradores.

Outro ponto importante de VHDL são os sinais. que funcionam como elementos de conexão do circuito. Os sinais podem formar circuitos sequenciais quando são utilizados em blocos de processo, ou circuitos combinacionais quando associados diretamente à saídas do circuitos. Assim, quando utilizados em um processo, o valor desse é atualizado somente no próximo ciclo, diferente de uma variável em uma linguagem de programação comum.

[1]H. Foster, "Part 6: The 2022 Wilson Research Group Functional Verification Study," Siemens, 2022, Disponível em: `https://blogs.sw.siemens.com/verificationhorizons/2022/11/21/part-6-the-2022-wilson-research-group-functional-verification-study/`. Acesso em: 25 jun. 2025 (7)

2.2.2 Aplicação no projeto

O *VHDL* foi a linguagem selecionada para a execução desse projeto, devido ao contato prévio com a mesma. Através da mesma, foi possível estruturar de maneira modularizada diferentes circuitos do projeto, abrangendo desde os módulos de interface com periféricos, até mesmo os módulos principais de operação dos programas.

2.3 *Hardware* Reprogramável

Um dos pontos chaves na história da eletrônica digital, foi a invenção o *Hardware* reprogramável. Como mencionado na seção de Introdução, foi durante a década de 1980, que foi desenvolvido e lançado o primeiro *Field Programmable Gate Array (FPGA)*. Tal ferramenta, trouxe flexibilidade para os circuitos digitais ao permitir que um desenvolvedor projetasse e implementasse um circuito, e que caso necessário fosse possível reescreve esse mesmo sistema, sem a necessidade da troca ou fabricação de um novo *Hardware*.

Sobre os *FPGAs*, a presença de blocos lógicos configuráveis ou elementos lógicos, nomenclatura de acordo com o fabricante, é o que permite que sejam capazes de serem reconfigurados. Os elementos lógicos, de acordo com o manual da Altera(8) referente ao *FPGA Cyclone IV* que foi utilizado nesse projeto, são as menores unidades lógicas do *FPGA*. Essas possuem em sua composição um registrador programável, que pode ser definido como diferentes tipos de *flip-flops*. Além disso, existem estruturas como redes de conexão de *carry* e de registradores, assim como uma LUT de quatro entradas, que permite operações lógicas de até quatro variáveis. Ademais, esses elementos oferecem a capacidade de direcionar diferentes tipos de conexões internas e suporte a diversas funcionalidades do registrador.

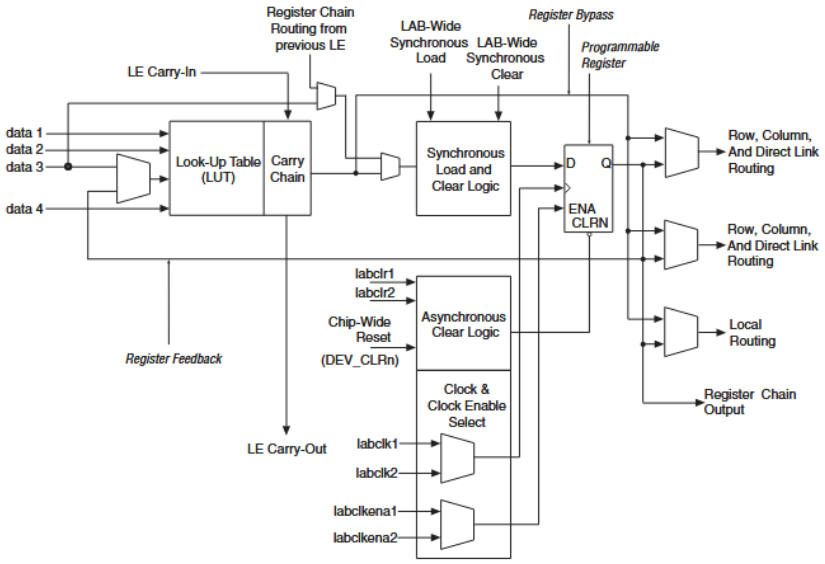

Figura 2.4: Componentes de um elemento lógico.[2]

2.3.1 Sobre o *FPGA* utilizado

No contexto desse projeto, foi-se empregado um placa de desenvolvimento da empresa QMTECH(9) equipada com um *FPGA Altera Cyclone IV GX* EP4CGX150DF27I7, que conta com cento e cinquenta mil elementos lógicas. Tal quantidade é muito mais que o necessário para o projeto atual, porém permite implementações futuras de maior grau de complexidade. Assim, essa placa serviu como *Hardware* principal do projeto.

Além disso, a placa conta com um conector para a configuração do *FPGA*, uma memória *RAM* que pode ser utilizada para armazenamento temporário de informações, dois barramentos de sessenta e quatro pinos. Dessa forma, mesmo com os pinos de alimentação e aterramento, tem-se bastantes pontos para configuração de comunicação. Além disso, estão presentes na placa *LEDs* e botões, que foram ferramentas relevantes para a depuração de códigos inicialmente.

[2]ALTERA, "Manual do dispositivo Cyclone IV®," ALTERA, San Jose, California, EUA, 2016, Disponível em: `https://www.intel.com.br/content/www/br/pt/content-details/653974/cyclone-iv-device-handbook.html`. Acesso em: 23 jun. 2025.(8)

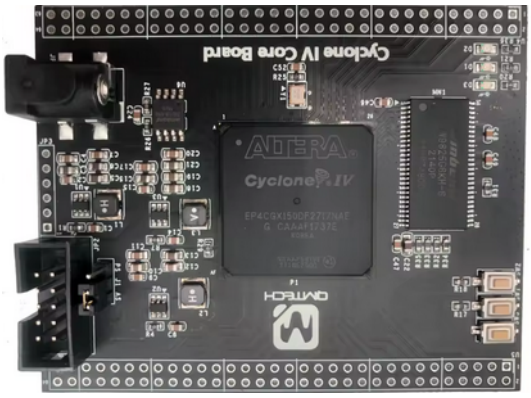

Figura 2.5: Placa QMTECH com FPGA usada no projeto[3].

Assim, esse kit concede ao usuário a flexibilidade de montar as conexões conforme as necessidades específicas do projeto, haja vista que a maioria dos pinos das laterais não possuem funções predefinidas. Dessa forma, para que os periféricos fossem integrados a esse kit de desenvolvimento, foi necessária a projeção e construção de uma placa responsável por disponibilizar uma interface de vídeo, uma saída de áudio e um sistema para conexão de um teclado.

2.4 Projeto de *PCB*

Conforme pontuado anteriormente, a placa de desenvolvimento empregada nesse projeto não possuía nenhum conector para comunicação de periféricos ou semelhantes, fazendo-se necessária a construção de uma placa de circuito impresso (PCB, do inglês *Printed Circuit Board*) dedicada. Sendo assim, a elaboração, construção e depuração do circuito dedicado foram pontos importantes do projeto.

Para que uma placa possa ser construída fisicamente, é interessante que essa seja, primeiro, desenvolvida em algum *software* especializado nesse tópico. Dessa forma, nesse trabalho, foi-se utilizado o programa KiCAD, levando em consideração que é uma ferramenta gratuita, e já conhecida pelo autor.

Assim, através dessa plataforma, foi possível criar o esquemático base do sistema que foi fundamental para a organização do projeto. Ademais, é importante pontuar que os componentes selecionados para o sistema, como resistores, capacitores de filtro e desacoplamento, foram selecionados com base em pesquisas, estudos e cálculos levando em consideração as necessidades do projeto.

[3]QMTECH, "Cyclone IV EP4CGX150GX Core Board UserManual," QM- TECH, China, 2022, Disponível em: `https://github.com/ChinaQMTECH/EP4CGX150DF27_CORE_BOARD/blob/main/QMTECH_CycloneIV_EP4CGX150GX_User_Manual(CoreBoard)-V01.pdf`. Acesso em: 23 jun. 2025(9)

Após o esquemático, é feita a montagem do *layout* da *PCB* no próprio KiCAD, sendo assim, foi necessário fazer o posicionamento dos componentes e o roteamento das trilhas. Durante essa fase, teve-se como objetivo elaborar a menor placa possível, levando em consideração a disposição e roteamento otimizados de cada componente. Então, após finalizada a montagem no programa, pode-se utilizar o *software* para gerar os arquivos necessários para a fabricação da placa.

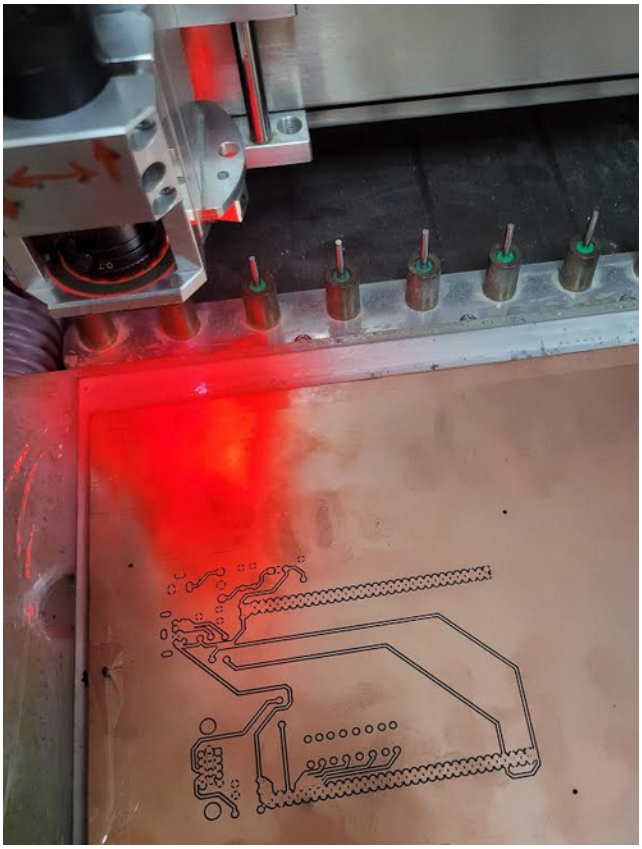

Figura 2.6: Fresagem da placa.

Durante o desenvolvimento desse trabalho, foram elaboradas duas versões da placa de suporte. A primeira versão serviu como um protótipo para avaliação do circuito, sendo utilizada para teste dos módulos, e identificação possíveis ajustes necessários no sistema. Assim, esse protótipo, apesar de possuir menos funcionalidades e ser menos otimizado, foi de suma importância para evitar falhas na versão final. Por fim, após projetado, a fabricação dessa foi feita de maneira simples usando uma placa de fenolite cobreada, e uma fresadora no laboratório CPTI.

Após os testes com o protótipo, foram feitas correções no circuito e implementações de funcionalidades novas, como um módulo de áudio. Além disso, os componentes foram reposicionados visando otimizar e minimizar a placa. Seguida a etapa de projeto da placa, foi encomendada a fabricação

dessa versão em uma empresa especializada para garantir a qualidade do projeto. Com essa já fabricada e entregue pelo fabricante, foi feita uma pequena correção do circuito de áudio, após testes e análises, no intuito de melhorar o desempenhos dos filtros implementados. Então, após essa alteração, conclui-se a etapa física do projeto.

2.5 Interface PS/2

Para a implementação de um teclado nesse projeto, foi-se utilizado um dispositivo de comunicação *PS/2*, devido ao contato prévio com esse tipo de comunicação. Sobre a interface de comunicação *Personal System/2*, essa foi lançada em 1987 no computador IBM(10), que carrega o mesmo nome. Sobre tal padrão, esse funciona de modo serial, usando um conector mini-DIN de 6 pinos, podendo ser utilizado para recebimento de dados de um teclado ou *mouse*.

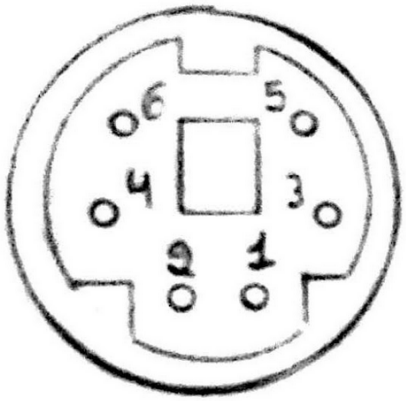

Figura 2.7: Conector *PS/2* fêmea.

2.5.1 Características de pinos

A estrutura física de conexão consiste em um conector de 6 pinos, embora na prática, apenas 4 sejam utilizados no sistema. Sobre os pinos ativos, esses possuem funções específicas e são numerados de maneira padronizada entre os fabricantes de eletrônicos.

No que tange a função de cada pino, pode-se dizer que os pinos 1 e 5 são utilizados para receber informações do teclado, enquanto os pinos 3 e 4 estão ligados circuito que energiza a placa. Tratando sobre a parte de comunicação do conector, o pino 1 tem como função receber os dados seriais referentes a tecla que foi pressionada, enquanto o pino 5 recebe os pulsos do *clock* gerado

pelo teclado para que os dados sejam sincronizados e interpretados de maneira correta. Sobre os pinos 3 e 4, o primeiro deles é atribuído ao aterramento do sistema, enquanto o outro é alimentado por uma tensão de 5 Volts que será transmitida para ligar o periférico conectado. Por fim, os pinos 2 e 6 não são conectados a nenhuma parte do circuito do sistema, permanecendo sem atribuição no padrão *PS/2*

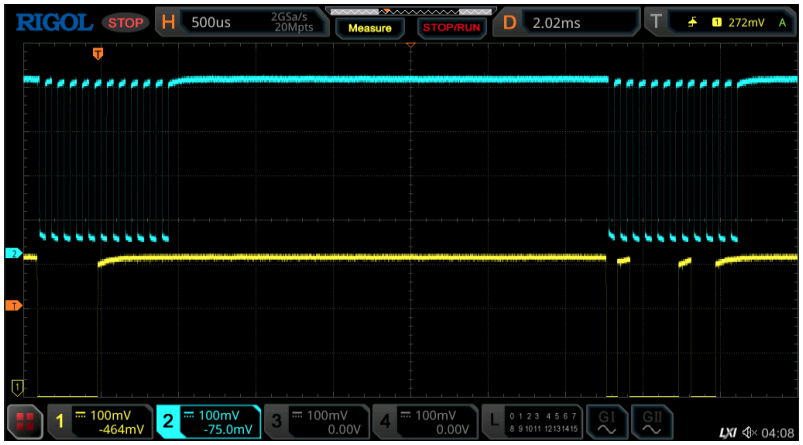

Figura 2.8: Leitura em osciloscópio de uma tecla sendo solta - Sinal F0 e Código 21 referente a tecla C.

2.5.2 Sinais de comunicação

Conforme mencionado pela Xilinx(11) no manual do kit de desenvolvimento Spartan-3E Starter Kit, o teclado *PS/2* envia um sinal de 11 bits através de pulsos para comunicar que tecla foi pressionada, e também 11 pulsos de *clock* em uma faixa de frequência entre 20 a 30 quilo-hertz para a sincronização da informação.

Detalhando a informação transmitida, o primeiro bit é um sinal *Low* conhecido como *starter* bit, indicando o começo de uma transmissão. Em seguida, são enviados 8 bits referentes ao código da tecla acionada. Por fim, é sinalizado um bit de paridade, em sequência de um sinal *High* representando o bit *stop*, sinalizando o final da mensagem. O envio do sinal de uma tecla é interrompido assim que essa é solta, e são enviados dois sinais de onze bits. O primeiro deles é o sinal de código "F0", indicando que a tecla está solta, seguido pelo sinal com o código referente a tecla recém liberada.

Além disso, vale observar que enquanto a tecla estiver pressionada, o teclado reenvia o código referente a essa a cada 100 milissegundos, visando a garantir a comunicação ao sistema de que essa está sendo pressionada.

2.6 Protocolo VGA

No intuito de gerar de vídeo nesse projeto, utilizou-se o padrão *VGA*, devido a sua simplicidade de implementação e à proposta de manter uma estética retrô. Então sobre o protocolo *Video Graphics Array*, esse foi lançado pela IBM integrado ao computador *Personal System/2* junto a interface mencionada na seção prévia. Posteriormente, esse consolidou-se como um dos padrões de transmissão de vídeo mais utilizados da indústria durante anos.

2.6.1 Características do protocolo

O *VGA* é um padrão de vídeo de transmissão analógica que, segundo Wilson(12), inicialmente contava com apenas 8 bits resultando em 256 cores, porém conforme se deram os avanços, o padrão alcançou uma paleta muito mais vasta em cores composta por 24 bits.

Nesse contexto, a cor de cada pixel podia ser codificada pela combinação de três canais analógicos de sinal, sendo eles vermelho, verde e azul, obedecendo assim o padrão *RGB*. Dessa forma, a representação da intensidade de cada componente da cor se dá através da tensão do pulso no canal podendo varia de 0 Volt quando a componente for ausente, até 0,7 Volt quando essa tiver intensidade máxima, conforme mencionado no manual da Xilinx(11).

Somado a esses três sinais de cores, ainda existem dois sinais fundamentais para gerar da imagem. A imagem do protocolo é formada a partir de uma varredura da tela pixel por pixel, na qual utilizam-se um sincronizador para cada eixo, garantindo a operação precisa do sistema. Dessa forma, no intuito de formar o vídeo, existe o sinal de sincronização vertical conhecido como *V-Sync*, e para o pareamento horizontal, existe o pulso *H-Sync*. No entanto, existem alguns monitores, que possuem suporte a uma técnica conhecida como *Sync-on-Green* para fazer a sincronização. Tal técnica, como seu nome propõe, utiliza-se diretamente do canal verde para a transmissão de pulsos de sincronização, diminuindo assim o número de canais necessários.

Então, para que esses sinais sejam transmitidos o protocolo utiliza comumente um conector conhecido como DE-15, que possui 3 fileiras de 5 pinos, tendo cada um deles uma função em específico. Assim os pinos 1, 2 e 3 transmitem, respectivamente, os canais vermelho, verde e azul. No intuito de fazer o aterramento do circuito, os pinos 5, 6, 7, 8 e 10 estão conectados. Na fileira inferior, estão os pinos de sincronização, sendo *H-Sync* a saída 13 e o *V-Sync* a saída 14. Por fim, os pinos 4, 9, 11, 12 e 15, de acordo com NextPCB(13), funcionam como pinos auxiliares para identificação do monitor,

ou alimentação a 5 Volts da placa gráfica. Dessa forma, por terem funções que não precisam ser controladas pelo *FPGA*, esses últimos pinos não serão abordados nesse projeto.

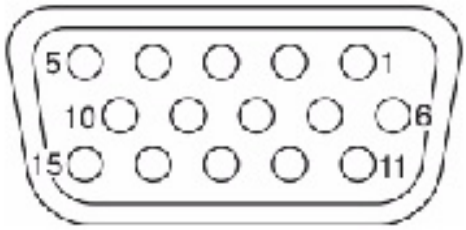

Figura 2.9: Conector DE-15 usado em *VGAs*[4].

2.6.2 Funcionamento dos pulsos de sincronização

A imagem gerada pelo protocolo é construída pixel por pixel, através dos dois pulsos de sincronização enviados. Para que isso ocorra, o sinal *H-Sync*, por exemplo, inicialmente é mantido em *Low* para fazer a um período de sincronização. Em seguida, o sinal muda para o estado *High*, porém a imagem ainda não está ativa, pois existe um *delay* inicial chamado *front porch*.

Após esse *delay*, o sinal continua em estado alto, porém está em sua área ativa, ou seja, a área que é disposta na tela. Após outro intervalo de tempo, o sinal ainda em estado alto, entra no *back porch*, que já não faz mais parte da imagem vísivel, funcionando como um tempo de espera antes do próximo pulso de sincronismo. Por fim, após o período do *back porch*, o sinal vai estado baixo e o ciclo é reiniciado.

Sendo assim, ao gerar a imagem existe uma área ativa, e outra área que não é exibida na imagem. Esse princípio também é válido para o sinal *V-sync*, que controla a transição vertical. Ademais, é importante pontuar que o tamanho de tais intervalos mencionados varia de acordo com a resolução utilizada, e frequência de atualização do sistema.

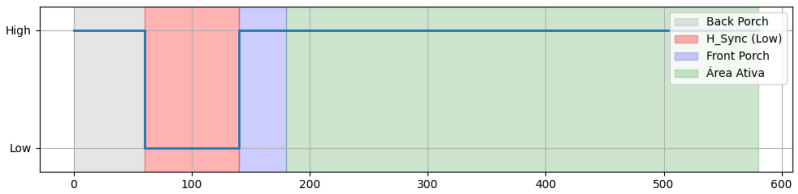

Figura 2.10: Representação de Pulso Horizontal.

[4]Sun, "C - Connector Pinouts," Oracle, Santa Clara, California, EUA, 2010, Disponível em: `https://docs.oracle.com/cd/E19201-01/821-0902-10/appC_Pinouts.html`. Acesso em: 25 jun. 2025.(14)

3
Objetivos do Trabalho

3.1
Objetivos específicos

A proposta por trás desse projeto é trabalhar conhecimentos de programação de *Hardware* reconfigurável, de forma que seja explorado o potencial de um *FPGA* para a construção de uma plataforma com a possibilidade de execução de jogos e aplicações interativas. Além disso, por meio desse trabalho devem ser executados de forma teórica e prática conceitos de eletrônica para circuitos, assim como aprendizados sobre a interpretação e aplicação de protocolos de periféricos, que são pontos valiosos para a formação de um engenheiro. Por fim, com o projeto concluído, tem-se uma plataforma capaz de oferecer suporte para a síntese de sistemas de retrocomputação em etapas posteriores

No âmbito da eletrônica, esse projeto tem como objetivo a construção de uma placa *PCB* para a integração do sistema. Primeiramente, a projeção de um circuito dedicado ao projeto trabalha conceitos como construção de circuitos eletrônicos, assim como a seleção e dimensionamento de componentes adequados. Tais conceitos, quando dominados acrescentam muito ao entendimento de sistemas eletrônicos. Somado a isso, a construção de módulos para utilizar periféricos, é uma forma efetiva de aprender a integrar um projeto autoral a sistemas já existentes. Além disso, a montagem física da placa proporciona a realização de diversos testes e análises, visando verificar o correto funcionamento do sistema. Para isso, empregam-se ferramentas fundamentais para o trabalho com *Hardware* como o multímetro para verificar tensões, correntes e possíveis curtos e o osciloscópio que é crucial para compreender a transmissão dos sinais e a análise dinâmica dos circuitos.

Na perspectiva da computação digital, a implementação de jogos e outras aplicações é uma forma interativa de trabalhar conceitos interessantes de computadores como memórias *RAM* e *ROM*, assim como estudar sobre circuitos lógicos e descrição de *Hardware*. O desenvolvimento dos códigos para comandar as operações do jogo, e controle de interfaces e periféricos, aperfeiçoa ideias de temporização, e modularização de projetos. Ademais, a implementação de tais módulos periféricos permite conhecer a operação de

padrões de comunicação já utilizados na indústria, aprimorando as habilidades integração em projetos de sistemas dedicados. Dessa forma, o objetivo contribui de maneira prática para o entendimento de circuitos digitais.

3.2 Escopo do sistema

Através do desenvolvimento de um sistema inteiro com suporte de áudio, vídeo e teclado, esse projeto tem como objetivo atingir dois públicos. O primeiro deles está atrelado a pessoas que desejam aprender sobre *Hardware* reconfigurável, e que precisam de um sistema já projetado para atender suas necessidades.

Além disso, esse sistema pode servir para pessoas com interesse pela área da retrocomputação, visto que esse oferece suporte de físico para periféricos básicos e pode ser usado para a síntese de arquiteturas legado. Essa liberdade de ser moldado de acordo com o interesse do usuário se deve ao fato de existirem códigos em linguagens de descrição de *Hardware* que implementam tais computadores. Para isso, bastaria fazer algumas adaptações em tais códigos para que consigam ser executados corretamente na plataforma desse projeto, assim permitindo o aprendizado e preservação de sistemas clássicos.

4
Atividades realizadas

Esse projeto desde seu princípio, trabalha diferentes áreas da Engenharia de computação, concentrando-se principalmente em dois aspectos, a eletrônica e a computação digital, campos que embora sejam usados em conjunto, possuem suas especificidades.

Assim, inicialmente, os conhecimentos do campo da computação digital ainda estavam sendo construídos. Apesar do contato prévio com circuitos lógicos em disciplinas como Técnicas Digitais, tal conhecimento ainda não era suficiente para a utilização de *FPGAs*. Além disso, o contato e domínio da linguagem *VHDL* ainda estava sendo moldado em disciplinas cursadas em paralelo.

Então, na área de eletrônica, também foi necessário aprofundar a compreensão do tema para que o projeto fosse materializado. Primeiramente, a base de conhecimento fornecida por matérias ligadas a circuitos, foi importante para que a elaboração das placas fosse possível. A partir da utilização de conceitos como filtros de áudio e seguidores de tensão, conseguiu-se implementar o módulo de áudio por exemplo. Porém, para implementar todas as necessidades do projeto, foi necessário aprender sobre módulos de conversão digital-analógica a partir de resistores, aprender sobre capacitores de desacoplamento e, também sobre capacitores de acoplamento em circuitos de áudio. Além disso, o contato em disciplinas de laboratório, e em trabalhos externos, utilizando ferramentas de eletrônica, como o osciloscópio e o multímetro, foi fundamental para garantir a plena transmissão de sinais e detecção de falhas nas placas.

4.1
Estudos Conceituais e Tecnológicos

4.1.1
Linguagem VHDL e Desenvolvimento para FPGA

Conforme pontuado na seção 2.2 e na seção anterior, a linguagem VHDL foi selecionada para o desenvolvimento completo desse projeto. O contato em sala de aula com tal linguagem foi fundamental para a escolha da mesma, visto

que não seria necessário aprender algo completamente novo desde o princípio. No entanto, os aprendizados obtidos em sala de aula foram consolidados, após a conclusão da disciplina respectiva, através da leitura e estudos utilizando o livro de Sistemas Digitais por Cruz(5).

Tal livro trabalha desde os conceito mais simples de eletrônica digital, como portas lógicas, até estruturas relevantes para a construção de sistemas reprogramáveis como máquinas de estados. Além disso, esse utiliza-se de exemplos para demonstrar tais estruturas, tornando mais simples a compreensão.

Tratando sobre o *FPGA*, o manual da Altera(8) foi utilizado para entender as capacidades do CI utilizado, assim como também para consultar as informações referentes a disposição e função cada pino. Ademais, para aprofundar a compreensão acerca das capacidades e atribuições dos pinos da placa de desenvolvimento foi necessário conferir o manual disponibilizado pela QMTECH(9). Em tal manual, estavam contidas informações tanto sobre o *FPGA* quanto sobre o circuito da placa, possibilitando a compreensão do escopo de tal sistema.

Assim, com as informações disponibilizadas pelos *data sheets*, pôde-se fazer de forma precisa o roteamento da placa, selecionando os pinos necessários para cada módulo do sistema. Além disso, constatou-se que a placa QMTECH já possuía um sistema de desacoplamento próprio, o que simplificou a implementação do sistema de alimentação na placa de suporte para energizar o *FPGA*.

Por fim, para que fosse possível programar os módulos, assim como definir os pinos e a transferência de tais informações para o *Hardware* reprogramável, utilizou-se o *software* de desenvolvimento Quartus disponibilizado pela Intel. Tal escolha se justifica pelo fato de o *FPGA* utilizado ser da marca Altera, que atualmente pertence à Intel, que disponibiliza as ferramentas para desenvolvimento.

4.1.2 Protocolo e interface de comunicação com periféricos

Primeiramente, para que o usuário conseguisse enviar informações para o projeto, utilizou-se um teclado *PS/2*, devido a experiência prévia com a programação desse dispositivo em disciplinas já cursadas. Dessa forma, para compreender detalhadamente a forma que esse opera, e posteriormente implementá-lo de maneira eficiente utilizou-se como referência o manual técnico da Xilinx(11). Embora esse seja referente a *FPGAs* de uma fabricante diferente, o documento dispõe de descrições teóricas precisas que ilustram a operação de tal interface.

Em seguida, para que fosse gerada a imagem, foi-se utilizado o protocolo *VGA*, pois esse mantém o aspecto clássico do projeto, assim como possui uma implementação simples. Dessa forma, assim como o teclado *PS/2*, foi-se utilizado o manual disponibilizado pela Xilinx(11) para entender como a imagem é gerada, bem como as formas que o sistema sincroniza tal ciclo utilizando pulsos que proporcionam uma área ativa da imagem e uma área que não é mostrada.

Além disso, para a implementação em código do vídeo, o livro de Pong Chu(15) contém um exemplo de uma arquitetura para fazer o controle da sincronização. Esse exemplo foi muito útil para compreender como fazer uma implementação própria do protocolo *VGA* nesse projeto. Por fim, para fazer os cálculos referentes a etapa de implementação era importante entender, por exemplo, o tamanho dos intervalos de *front porch*,*back porch*, tendo em vista que tais parâmetros variam de acordo com as especificações do monitor devido a questões relacionadas a temporização. Dessa forma, como os monitores utilizados possuíam a resolução de 1024 por 768, foi-se utilizada as informações fornecidas por Secons(16), para a implementação de cálculos de forma correta e garantir a compatibilidade com os monitores utilizados.

4.1.3 Desenvolvimento da *PCB*

A etapa de projeto da placa de suporte também necessitou de aprimoramento de habilidades. Primeiramente, para construir os esquemáticos e os *layouts* das *PCBs* do projeto, foi selecionada a plataforma KiCAD por ser um *software* gratuito e também devido ao contato prévio com tal aplicação. Sendo assim, para elaborar os esquemáticos dos circuitos utilizando componentes específicos, foi necessário ler e estudar a ficha técnica de cada componente para fazer a correta conexão dos pinos de cada dispositivo empregado. Tais fichas técnicas estão disponibilizadas no repositório do projeto [1].

Além disso, para que os circuitos fossem montados, foi-se utilizado conceitos aprendidos em disciplinas da graduação como o uso de amplificadores operacionais para isolamento de partes do circuito, assim como a construção de filtros para a seleção da banda de frequência na reprodução do áudio. No entanto, para garantir o funcionamento pleno do sistema, precisou-se aprofundar o conhecimento sobre conceitos como capacitores de desacoplamento, capacitores de acoplamento.

[1]Disponível em: `https://github.com/MontSantos/Projeto_Final_2/tree/main/Datasheets`

4.1.3.1 Capacitores de desacoplamento

Na construção de circuitos, um ponto a ser levado em consideração é a oscilação na alimentação do sistema. Dessa forma, uma forma de contornar possíveis ruídos e oscilações na tensão geral da placa utilizou-se o documento de Kay(17). O documento propõe que em uma rede de desacoplamento existem dois tipos de capacitores, aqueles que fazem o desacoplamento local, que possuem menores valores, e os capacitores *bulk* que possuem valores maiores e são posicionados próximo a entrada de alimentação. Para compreender a aplicação de ambos deve-se conhecer o conceito de indutância parasita.

Esclarecendo sobre o termo anterior, capacitores reais não possuem comportamento capacitivo a todo momento. De fato, esses, dependendo da frequência do circuito, podem se comportar como indutores, trazendo assim ainda mais ruído ao sistema. Dessa forma, capacitores com baixas capacitância se comportam melhor em altas frequência, possuindo uma baixa indutância parasita. Enquanto, para frequências baixas, utilizam-se valores maiores de capacitância pois esses apresentam melhor desempenho com relação a presença de indutância parasita.

Sobre os capacitores de desacoplamento local, esses ficam associados a entradas de alimentação de CIs e outros componentes para servirem como um curto rápido de altas correntes em direção ao aterramento, e preferencialmente o mais próximo o possível de tais entradas a fim de garantir a menor indutância parasita causada pela trilha. Dessa forma, Kay(17) menciona que um valor comum para desacoplamento local é 100 nF, enquanto para capacitores do tipo *bulk* deve se utilizar um capacitor tendo pelo menos dez vezes a soma da capacitância dos capacitores de acoplamento que compõe o sistema, sendo 10 uF um valor comum para tais componentes.

Ainda sobre os capacitores conhecidos como *bulk*, esses tem como função garantir que a tensão que alimenta o circuito mantenha-se estável durante toda a operação do sistema. Além disso, como a rede elétrica trabalha com frequências menores, ao ser retificada essa propriedade continua, sendo assim são necessário capacitores com capacitâncias maiores. Dessa forma, a função desse componente também é guiar correntes residuais, *ripple*, vindas do processo de retificação da fonte de tomada e compensar as pequenas variações na tensão.

Assim, a combinação dos dois tipos de capacitores de desacoplamento proporciona uma filtragem eficiente em diferentes faixas de frequência, garantindo uma maior estabilidade ao sistema.

4.1.3.2 Capacitores de acoplamento

Os capacitores de acoplamento são estruturas comumente utilizadas em circuitos de áudio. Conforme pontuado por Workman(18), o fluxo de correntes contínuas em componentes de áudio como alto-falantes e fones de ouvido, é extremamente danoso para tais equipamentos podendo estragá-los indefinidamente. Sendo assim, comumente são associados capacitores em série com o circuito de áudio visando contornar tal problema.

Sobre os capacitores, esses possuem comportamentos diferentes de acordo com a corrente do circuito. Em um circuito de áudio, deseja-se garantir que as oscilações sejam reproduzidas com qualidade na saída. Contudo, é possível que existam componentes contínuas indesejadas em tal sinal. Sendo assim, para as componentes contínuas do sinal, os capacitores iniciam descarregados, operando como um curto, e conforme a corrente flui esses vão se carregando. Conforme o capacitor vai carregando, a corrente que atravessa ele vai diminuindo, até que a tensão armazenada no capacitor seja igual a tensão que o alimenta. Quando isso ocorre, o capacitor funciona como uma abertura no circuito bloqueando o fluxo até que se descarregue.

No entanto, para a componente alternada do sinal, o capacitor dificilmente irá se carregar por completo, tendo em vista que a corrente faz com que o capacitor carregue e descarregue constantemente. Ademais, como o equipamento de áudio apresenta resistência, a associação entre o componente eletrônico e o equipamento forma um filtro passa-altas.

Dessa forma, ao selecionar o capacitor de acoplamento deve-se levar em consideração as frequências que não são relevantes para o sinal final. Para isso, pode-se usar a equação da frequência de corte. Veja abaixo:

$$|fc| = \frac{1}{2\pi \cdot R \cdot C} \tag{4-1}$$

A equação 4-3 determina a frequência de corte de um filtro a partir da relação da resistência e capacitância dos componentes associados. Essa frequência demonstra o ponto no qual ocorre a atenuação de 3 dB no sinal de saída em relação ao inicial, e que a partir dessa frequência os sinais são atenuados consideravelmente visto que é uma escala logarítmica.

4.2 Testes e protótipos

As etapas de prototipação foram cruciais para a garantia de uma base sólida de *Hardware* capaz de oferecer suporte completo para os módulos que posteriormente fariam a integração entre o *FPGA* e os periféricos.

Dessa forma, após o desenvolvimento em *software* da primeira versão da placa, mencionado previamente, era necessário que a *PCB* fosse fabricada. Sendo assim,com o auxílio da equipe do Centro de Pesquisa em Tecnologia de Inspeção (CPTI), foi fresado o primeiro protótipo, em uma placa de fenolite cobreada de face dupla. Depois de fabricada, iniciou-se o processo de validação prática do sistema, no intuito de verificar se o projeto necessitaria de modificações para pleno funcionamento.

Assim, durante a fase de testes, foram detectados pontos que exigiram correções para que o circuito operasse de maneira correta. Tal etapa demonstrou-se desafiadora, pois foi necessário revisar as conexões entre os componentes e os manuais de especificação de múltiplos componentes, no intuito de compreender o que poderia estar falhando. A exemplo disso, tem-se o caso da conexão entre a fonte e a chave, que devido a um erro do projeto inicial, inverteu-se a conexão dos pinos da chave. Tal fato implicou na necessidade de cortar da trilha e refazer a conexão de alimentação do circuito usando o outro pino da chave, no intuito de corrigir esse erro. Assim, esses e outros ajustes foram fundamentais para o funcionamento do protótipo, e para que a versão seguinte da placa fosse projetada de maneira correta. Dessa forma, com a parte física resolvida, iniciou-se a implementação dos módulos.

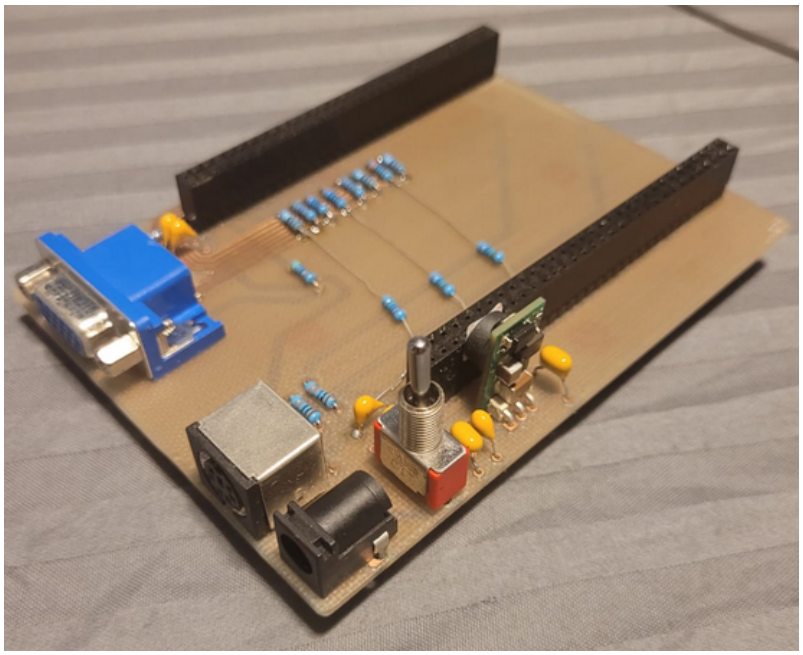

Figura 4.1: Protótipo após todas alterações realizadas.

4.2.1 Implementação e testes simples de VGA

O primeiro módulo a ser tratado foi o de vídeo, pois esse posteriormente tornaria mais simples a depuração de módulos seguintes ao fornecer uma interface visual. A primeira integração de tal módulo consistiu em um teste de exibição de barras de cores na tela do monitor, através do programa VGA_TV [2]. Além disso, é relevante pontuar que foi necessário usar um *Phase-Locked Loop* (PLL), para que o sistema estivesse na frequência necessária para a geração de imagem da resolução utilizada.

No intuito de implementar esse teste, o módulo de vídeo foi organizado em três arquivos principais. O primeiro deles é o `PLL_VGA.vhd`. Esse foi configurado a partir de ferramentas de criação de *PLLs* disponibilizadas no software de desenvolvimento da Altera, que após as definições do usuário, criam o módulo. Com isso, é ajustada a frequência do clock de 50 MHz para 65 MHz, garantindo a formação da imagem com a temporização correta.

O próximo arquivo é o `TIMING_CTRL.vhd`, nele estão presentes todas as informações referentes aos intervalos dos pulsos de sincronismo referente a resolução do monitor(16), para que possam ser feitas todas as operações do protocolo *video graphics array*. Assim, esse módulo é responsável por produzir os pulsos de sincronismo vertical e horizontal, assim como fornecer as coordenadas do pixel tratado naquele ciclo para outros módulos. Adicionalmente, o

[2]Disponível em: `https://github.com/MontSantos/Projeto_Final_2/tree/main/HDL_Testes`

`TIMING_CTRL.vhd` gera um sinal de controle, que indica se o pixel em questão esta incluso na área visível da tela.

Em seguida, existe o módulo `IMAGE_CTRL.vhd`, responsável pelo controle do que é exibido no monitor, através de sua lógica interna. Sendo assim, esse módulo recebe as coordenadas do pixel, e com base nessas indicações, ele determina qual cor deve ser exibida por esse pixel. A exibição das cores é feita através do padrão *RGB*, de forma que existem 3 bits para o vermelho, 3 bits para o verde e 2 para azul. Dessa maneira, a partir da quantidade de bits ativos é possível determinar a intensidade de tal cor na tela, e assim formar as barras de cores desejadas.

Figura 4.2: Barra de cores.

Como a estrutura e organização de módulos de geração de imagem demonstraram-se eficazes e flexíveis, esse padrão foi mantido durante o desenvolvimento do projeto como um todo, apenas realizando modificações no módulo `IMAGE_CTRL.vhd` conforme cada aplicação exigisse.

Em seguida, com a intenção de avaliar a intensidade e operação de cada bit de cor para a geração de imagens, foi feito um novo teste. Como o sistema apresentava apenas 8 bits de cores, esse era capaz de produzir ao máximo 256 cores distinta.

Assim, a partir da estrutura proporcionada pelo programa de teste prévio,

fez-se um novo programa de testes de nome VGA_CORES [3] para dividir a tela em 256 quadrados, de modo que cada um representava uma cor possível a ser apresentada pelo sistema. Para isso, o eixo horizontal foi segmentado em 8 colunas representando os oito valores que o vetor vermelho poderia transmitir. De forma semelhante, o eixo vertical foi divido em 8 linhas, representando os oito valores que verde poderia assumir. Assim, com 64 células, mais uma vez segmentou-se o eixo vertical, dividindo cada célula em quatro partes representando os 4 distintos valores de azul. Dessa maneira, é produzida uma imagem com todas as cores disponíveis no sistema.

Figura 4.3: Teste com 256 cores.

4.2.2 Implementação e testes simples de teclado

Após a validação do módulo de vídeo, era crucial certificar que o usuário fosse capaz de interagir com o sistema. Nesse sentido, era necessária a integração do teclado com o sistema, a fim de receber comandos do usuário.

Sendo assim, planejou-se um esquema de teste, consistindo em um sistema que exibisse no monitor uma cor padrão, e caso fossem pressionadas teclas específicas, a cor exibida mudaria de acordo. Assim, desenvolveu-se uma estrutura dividida em três blocos fundamentais que garantiam a integridade da leitura das teclas. Essa estrutura pertence ao programa *KEYBOARD*[4]

Primeiramente, tem-se o módulo `SYNC.vhd`, que recebe o *clock* de operação do sistema e o *clock* vindo do teclado. Dessa forma, esse módulo possui a função importante de sincronizar a operação de ambos pulsos, e em seguida gerar um sinal de saída. Caso esse sinal não fosse implementado, leituras de

[3]Disponível em: `https://github.com/MontSantos/Projeto_Final_2/tree/main/HDL_Testes`

[4]Disponível em: `https://github.com/MontSantos/Projeto_Final_2/tree/main/HDL_Testes`

teclado poderiam apresentar erros inesperados, como a não detecção de uma tecla ao ser pressionada, que foi um problema recorrente no projeto até a implementação de tal módulo.

Outro módulo importante é o `TECLADO.vhd`, que recebe o sinal do *clock* do teclado já sincronizado com o sistema e o dado serializado a ser interpretado. A partir do dado de onze bits vindo do teclado, o programa processa a informação e extrai o código referente a tecla que foi enviada. Sendo assim, com o código da tecla extraído, o módulo transmite-o por sua saída juntamente com um sinal que demonstra que foi terminada uma leitura.

Por fim, existe o módulo `TEC_FF.vhd`, que recebe da arquitetura anterior o código da tecla e o sinal de fim de leitura. A partir do sinal de leitura completa, é feita uma lógica de detecção de entradas, armazenando em 2 vetores os códigos referentes as últimas entradas. Alem disso, `TEC_FF.vhd`, possui *flip-flops* associados a teclas específicas utilizadas nesse programa. Dessa maneira, quando uma tecla nova é pressionada, é guardado em um *flip-flop* que essa está ativa, e caso sejam recebidos o sinal "F0" e o código da mesma tecla novamente esse *flip-flop* muda sua saída para "0". Assim, é possível que sejam pressionadas várias teclas simultaneamente, sem que uma interfira na outra. Ao final, esse módulo envia como saída o estado de cada *flip-flop*, para que outros módulos trabalhem com essa informação. Desta forma, com todos esses pontos integrados, foi possível enviar comandos que mudavam a cor da tela, demonstrando sucesso no teste.

Por fim, deve-se pontuar que a estrutura base desses módulos, foi reutilizada em outras aplicações do projeto que utilizam o teclado, apenas alterando `TEC_FF.vhd` para controlar as teclas específicas de cada aplicação.

4.2.3 Implementação e testes simples de áudio

Por fim, com ambos os módulos anteriores estando já implementados, havia espaço para que fosse possível implementar um módulo que não fora planejado antes, o sistema de áudio. Porém, a placa original não possuía suporte próprio para lidar com transmissão de áudio. Sendo assim, utilizou-se uma *protoboard* para montar um esboço simples de filtros de frequência, no intuito de atenuar frequências altas, com uma saída para conector P2. Essa estrutura será explicada no capítulo seguinte.

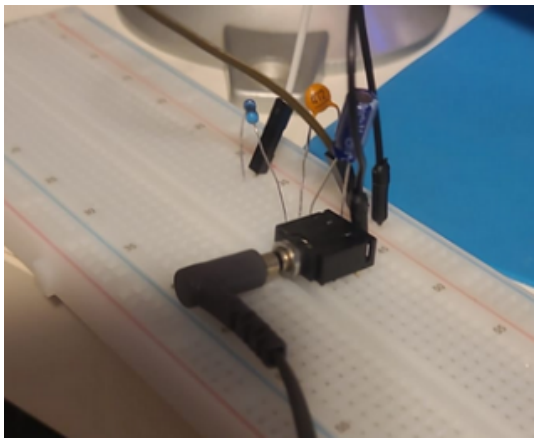

Figura 4.4: Protótipo de circuito de áudio.

Com o circuito montado, era necessário testá-lo e visualizar o seu funcionamento. Sendo assim, para isso, era necessário que o *FPGA* fosse capaz de gerar sons. Dessa forma, foi-se implementado o programa chamado SOM [5], com um módulo único responsável pela geração de sinais digitais em diferentes frequências. Tal módulo, chamado de `OSCILADOR.vhd` recebe apenas o *clock* do sistema e possui duas saídas lógicas que representam o áudio do lado esquerdo e do lado direito.

Internamente, o componente possui um vetor de 48 elementos, compostos por valores que representam a aproximação inteira da frequência de oscilação correspondente a cada nota musical(19), abrangendo assim quatro oitavas de notas. Assim, para gerar as oscilações de acordo com a frequência desejada, são utilizados contadores incrementados de acordo com o *clock*.

Dessa forma, para gerar a oscilação desejada basta que o sinal de saída fique alternando entre estados, configurando, assim, uma onda quadrada. Sendo assim, para determinar quando os estados devem ser alterados, basta dividir a frequência máxima do sistema por duas vezes a frequência desejada para encontrar a quantidade de *clocks* a serem contados até que possa ser alternado o estado do sinal de saída. Isso se justifica, pois para gerar uma onda de determinada frequência, o sinal deve alternar entre estados a cada meio-período, no qual o estado *High* é quando está sendo emitido som.

Assim, o código percorre o vetor sequencialmente, gerando cada nota em sua saída por um curto período de tempo. Dessa maneira, foi possível testar o desempenho do circuito de áudio para frequências entre 63 a 988 Hertz, que apresentou bons resultados.

[5]Disponível em: `https://github.com/MontSantos/Projeto_Final_2/tree/main/HDL_Testes`

5
Projeto e especificação do sistema

Esse capítulo tem como objetivo explicar de forma detalhada o desenvolvimento do produto final, desde o projeto dos circuitos e montagem do *Hardware* até a construção dos jogos e aplicações interativas.

Inicialmente, serão demonstradas as razões pela forma que a placa foi construída demonstradas através de cálculos e explicações sobre o posicionamento de alguns componentes. Ademais, serão tratadas as alterações realizadas após a fabricação da placa no intuito de melhorar o desempenho de módulos do circuito.

Em seguida, tem-se a parte que irá explicitar as aplicações implementadas. Sendo assim, o Pong, *Breakout*, Interface de desenho e sintetizador de áudio serão abordados de maneira minuciosa, através da explicação de seus módulos e a integração entre eles.

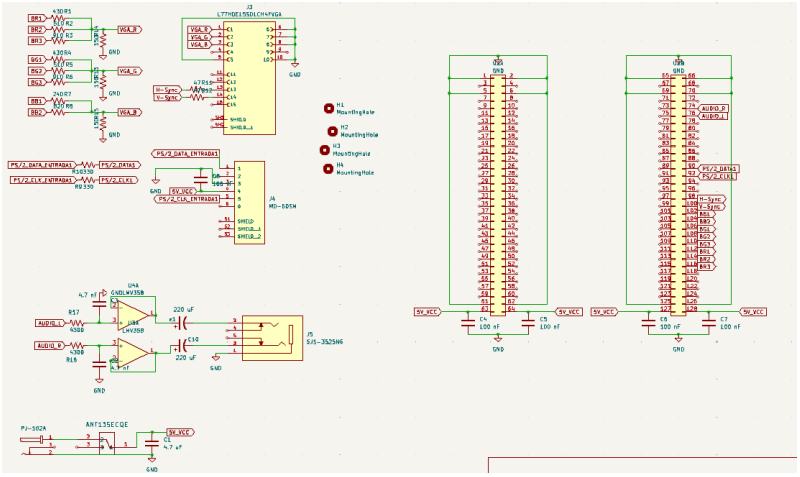

Figura 5.1: Esquemático da Placa Final corrigida.

O esquemático [1] demonstrado na Figura 5.1 ilustra o panorama geral do sistema que foi montado, exibindo assim as estruturas e conexões referentes aos módulos de alimentação, áudio, comunicação *PS/2* e geração de imagem. Além disso, também pode ser evidenciada a presença de componentes de filtragem e estabilização da alimentação, que garantem o bom funcionamento do sistema

[1] Disponível em: `https://github.com/MontSantos/Projeto_Final_2/tree/main/PCB_Projetos/Vers%C3%A3o_Final`

através da mitigação de ruídos. As fichas técnicas, utilizadas no projeto final da placa, estão disponíveis no repositório [2]

Além disso, pode-se ressaltar algumas mudanças significativas como a concentração dos pinos utilizados do *FPGA* em apenas um lado, bem como o aprimoramento no espaçamento entre os módulos. Tais pontos permitiram que o roteamento fosse simplificado e que a placa reduzisse consideravelmente suas dimensões.

5.0.1 Módulo de alimentação

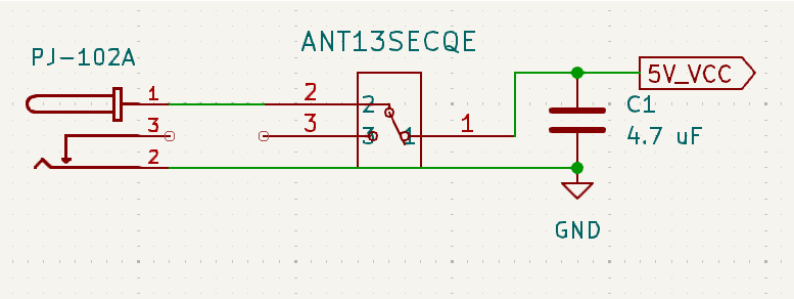

Figura 5.2: Entrada da alimentação.

O módulo de alimentação tem como única função fornecer energia a todos componentes que estejam conectados a eles. Sendo assim, no intuito de oferecer suporte a fonte de alimentação vinda com a placa de desenvolvimento QMTECH(9), foi selecionado um conector barril padrão compatível com a mesma fonte.

Em seguida, tem-se a chave e um capacitor de desacoplamento do tipo *bulk* para lidar com os ruídos e oscilações vindos do processo de retificação por parte da fonte de tomada, garantindo assim a estabilidade da alimentação. Além disso, próximo a entrada de alimentação de cada componente do circuito, estão posicionados capacitores de desacoplamento local visando minimizar as variações na tensão de alimentação e mitigar o ruído advindo de operações em frequências altas. Ademais, é relevante pontuar que os capacitores de desacoplamento foram posicionados próximos aos componentes que esses estão associados, visando reduzir a indutância parasítica da trilha.

No intuito de determinar quais valores a serem usados para esses componentes, foram feitas diversas pesquisas. Em um documento da *Texas Instruments*, companhia de eletrônicos, o autor Art Kay(17) menciona que para capacitores de desacoplamento local é muito comum se utilizar componentes com

[2]Disponível em: `https://github.com/MontSantos/Projeto_Final_2/tree/main/Datasheets/Vers%C3%A3o_Final`

a capacitância de 100 nano-Farad. Conforme comentado na seção 4.1.3.1, isso se justifica pois esses possuem baixa indutância parasítica em altas frequências demonstrando o seu bom desempenho para tal aplicação.

Continuando, nesse mesmo documento é pontuado que para desacoplar oscilações vindas da fonte, são habitualmente dimensionados capacitores cuja capacitância é pelo menos dez vezes maior do que a soma da capacitância dos desacopladores locais. Sendo assim, no sistema como um todo, estão posicionados seis capacitores de desacoplamento local, na qual a soma desses resultaria em menos de 1 uF. Dessa maneira, é razoável a escolha de um componente de 10 uF, visto que esse está de acordo com o que foi proposto para o funcionamento correto.

5.0.2
Módulo de vídeo

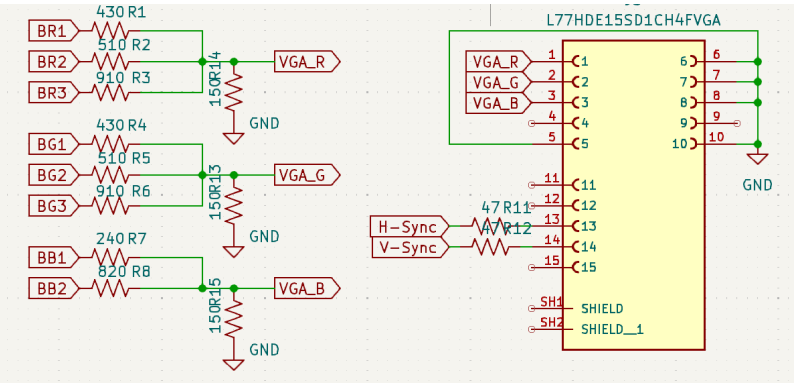

Figura 5.3: Módulo de vídeo.

A transmissão de vídeo foi planejada para estar de acordo com o protocolo *VGA*. Dessa forma, era necessário que fosse transmitidas informações analógicas para o terminal de conexão. No entanto, o *FPGA* possui apenas terminais de saída digital, necessitando assim que fosse montada uma estrutura para gerar um sinal analógico a partir da informação enviada pelo *FPGA*.

5.0.2.1
Sistema de cores

Conforme pontuado no manual do kit de desenvolvimento Xilinx(11), o padrão *VGA* utiliza valores entre 0 a 0,7 Volts para os canais de cores. Sendo assim, a intensidade da cor é determinada pela tensão dos terminais, de forma que quanto maior a tensão, mais intensa é tal cor.

Uma forma de implementar uma informação analógica usando sistema digitais é segmentando o canal em múltiplas entradas, criando assim um

sistema de bits de informação. Dessa forma, foi-se projetado que o sistema utilizasse oito bits ao todo, sendo 3 para o vermelho, 3 para o verde e 2 para o azul.

No entanto, ainda era necessário que a informação que chegasse no canal do monitor estivesse unifica da e dentro da faixa de operação dos aparelhos, caso contrário o equipamento seria danificado visto que o sinal *High* do *FPGA* é de 3,3 Volts. Sendo assim, para unir os pinos de informação e alcançar uma saída compatível com a entrada analógica do monitor, foi montada uma estrutura utilizando divisores de tensão resistivos conforme na Figura 5.4.

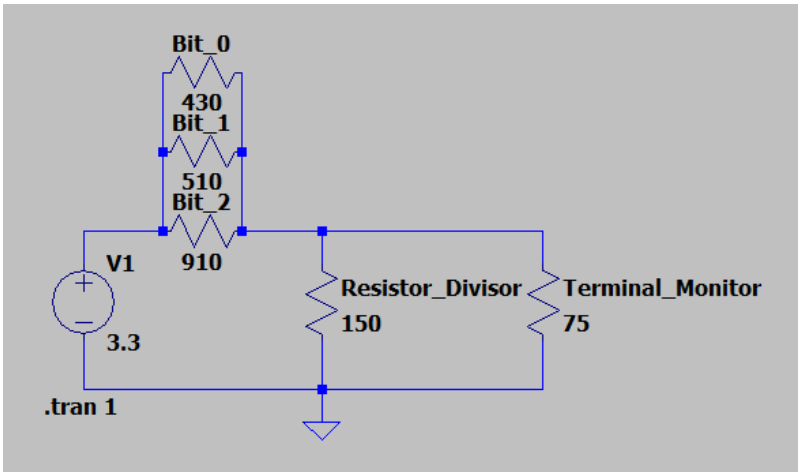

Figura 5.4: Divisor de tensão usado no projeto com todos bits ligados.

Dessa forma, para montagem do sistema de cores, cada pino foi conectado a um resistor. Em sequência a esse resistor, é feita a união dos bits do mesmo canal em um nó comum seguido por outro resistor, que está aterrado e, simultaneamente, em paralelo com a resistência do terminal do monitor. Sendo assim, a tensão desse nó comum é aquela que atua efetivamente na entrada do canal correspondente do monitor. Assim, a tensão de saída é obtida através da equação seguinte:

$$V_{saída} = V_{entrada} \cdot \frac{R_{eq_{saída}}}{R_{bits} + R_{eq_{saída}}} \tag{5-1}$$

Montada a estrutura, era necessário que fossem escolhidos os valores para os resistores. Sendo assim, sabia-se que a resistência do terminal *VGA* de cores de um monitor é de 75 Ω(11), dessa forma arbitrou-se que o resistor paralelo a esse seria de 150 Ω, fazendo uma resistência equivalente de saída de 50 Ω. Para decidir o valor dos resistores de cada bit, precisava-se que quando todos estivessem ligados a tensão de saída fosse 0,7 Volt. Sendo assim tem-se a equação:

$$0,7 = 3,3 \cdot \frac{50}{R_{eq_{bits}} + 50} \tag{5-2}$$

Então, para calcular o valor dos três resistores, foi montado o programa *CalcBest.py* [3] contendo valores de resistores disponíveis no acervo de componentes do autor. A partir desses valores, o programa utilizou todas permutações possíveis de 3 resistores diferentes para calcular a divisão de tensão e retornar aquela com o resultado mais próximo a 0,7 V. Sendo assim, foi determinado que para as cores de 3 bits deveria se usar resistores de 430, 510 e 910 Ω, enquanto o canal azul devia usar resistores de 240 e 820 Ω.

Dessa maneira, para averiguar se tais valores eram coerentes, e a tensão de saída estava no total de interesse, simulou-se o circuito [4] no *software* LTspice. Como pode-se visualizar na Figura 5.5, as tensões de saída resultaram em aproximadamente 700 mV, confirmando que aqueles valores poderiam ser utilizados no sistema.

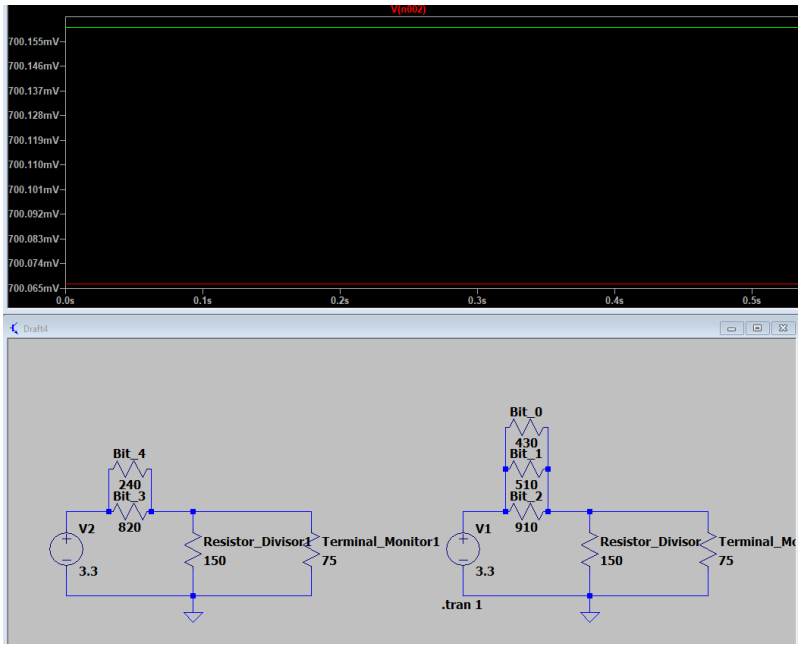

Figura 5.5: Divisores de tensão e gráfico da tensão de saída.

Por fim, através de todos esses processos, foi-se construído o sistema de conversão digital-analógico para geração de cores através de bits. Sobre o sistema de bits, quanto menor a resistência associada a esse, maior será a corrente que atravessa o nó comum, resultando em uma participação mais

[3] Disponível em: `https://github.com/MontSantos/Projeto_Final_2/tree/main/Resistor`

[4] Disponível em: `https://github.com/MontSantos/Projeto_Final_2/tree/main/LTspice_Sims`

significativa para a tensão final de saída. Dessa forma, bits associados a resistência menores, produzem cores mais intensas.

5.0.2.2
Sistema de sincronização

Além do sistema de cores, era necessário a elaboração do circuito conectados aos pinos de sincronização do protocolo. Sobre esses sinais, esses são pulsos de nível lógico, e que portanto não necessitaram de um tratamento específico para a tensão haja vista que as saídas do *FPGA* são digitais. Sendo assim, a tensão nas quais esses operam é a mesma, ou seja, ela varia entre 0 Volts para o estado *Low* e 3,3 V para o estado *High*. No entanto, ainda era necessária a avaliação da impedância de ambos os pinos, pois impactaria diretamente a qualidade da transmissão de sinal.

Assim, o teorema da máxima transferência de potência dita que para alcançar a maior eficácia na transmissão do sinal é necessário que a impedância entre fonte e carga estejam casadas(20). Sendo assim, para montar um circuito eficiente, precisava-se saber qual era a impedância dos cabos *VGA*. Dessa forma, encontrou-se que a impedância da linha de transmissão do padrão *VGA* costuma ser 50 ou 75 Ω(21).

Sendo assim, optou-se por utilizar um resistor de 47 Ω em série para ambos os pinos, como uma medida conservadora, no intuito de evitar o *overshoot*. Tal evento seria causado, pela utilização de um resistor, associado em série aos pinos, de valor maior do que a impedância do cabo, que poderia ocorrer caso fosse selecionado um resistor de 75 Ω e a impedância do cabo fosse 50 Ω por exemplo.

5.0.3
Módulo de teclado

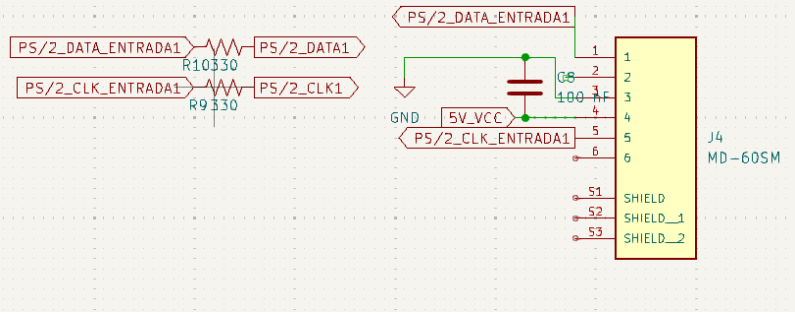

Figura 5.6: Circuito do teclado.

A implementação do módulo de teclado se deu de forma mais simples e menos elaborada. Sabia-se que a *interface PS/2* usa apenas quatro dos

seis pinos disponíveis no conector. Desses pinos, dois deles são referentes a alimentação do teclado e dois deles são referentes a transmissão de informações vindas do teclado.

Dessa forma, o pino de alimentação foi conectado a rede de alimentação principal da placa, e o pino de aterramento conectado também ao terra do sistema. Para garantir que a alimentação ocorresse de maneira estável foi-se atribuído um capacitor de 100 nF para fazer o desacoplamento.

Então, o teclado *PS/2*, por estar com os pinos de comunicação conectados ao *FPGA*, envia seus sinais de acordo com a tensão desses que nesse caso é 3,3 V. Esse valor está de acordo com a faixa de operação do componente e, sendo assim, não foi necessário elaborar um circuito muito complexo.

Portanto, para os pinos de *clock* e dados foi implementado um limitador de corrente para proteger o *FPGA* visto que suas portas de comunicação podem ser atravessadas somente por correntes entre -25 mA até 40 mA, conforme mencionado no manual da Altera(8). Sendo assim, decidiu-se adotar o valor de 10 mA como alvo, o que estabelecia uma margem segura em relação aos limites propostos no manual. Para isso, considerando a tensão com 3,3 V e a corrente desejada como 10 mA, pode-se utilizar a lei de Ohm para obter o valor do resistor:

$$\begin{aligned} R &= \frac{V}{I} \\ &= \frac{3,3\ \text{V}}{10\ \text{mA}} \\ &= 330\ \Omega \end{aligned} \tag{5-3}$$

Desse modo, montou-se o circuito de comunicação PS/2, utilizando resistores de 330 Ω para garantir que a corrente não alcance os limites dos pinos. Assim, foi conferida robustez e segurança ao circuito do teclado.

5.0.4
Módulo de áudio

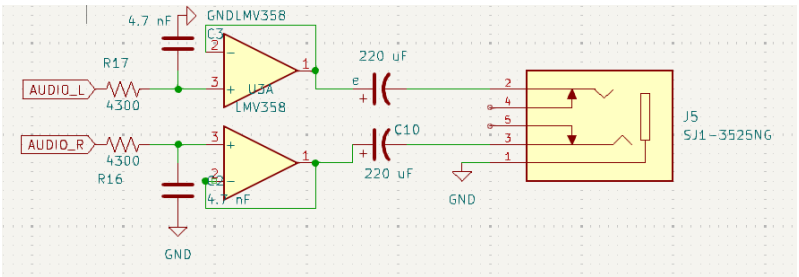

Figura 5.7: Módulo de áudio corrigido.

Por fim, o último módulo a ser implementado e descrito foi o responsável pela reprodução de áudio. Deve-se pontuar que, embora o circuito tenha sido fabricado com uma falha de projeto, esse passou por uma depuração e correção durante a montagem física da placa final. Dessa forma, nessa seção será tratado apenas o circuito atualizado e funcional.

Sendo assim, o módulo de áudio foi projetado com a intenção de garantir um som mais limpo, tentando reduzir a presença do som característico de uma quadrada visto que em sua composição estão presentes infinitos harmônicos ímpares(22). Para isso, foi arbitrado que a frequência máxima do sinal deveria ser por volta de 8000 Hertz. Então, para que isso seja possível, basta que seja utilizado um filtro passa-baixa na configuração resistor-capacitor.

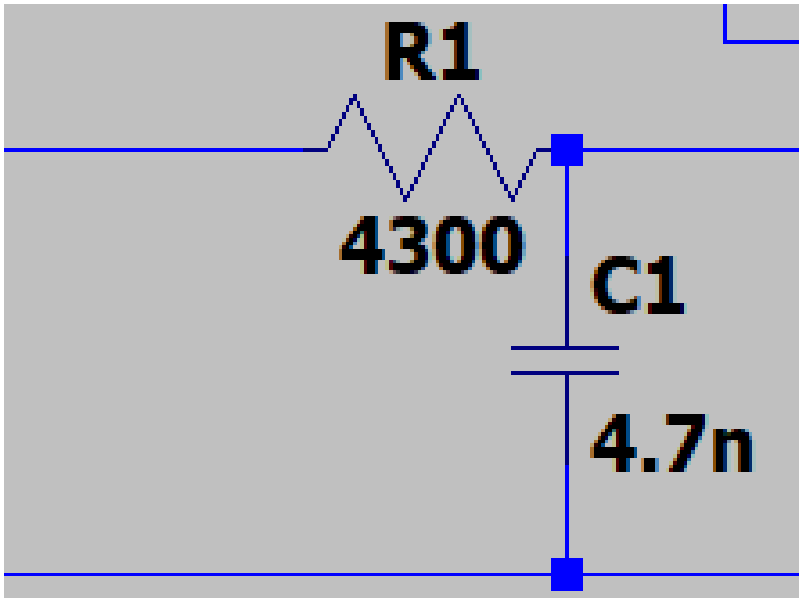

Figura 5.8: Filtro Passa-baixas usado no projeto.

Portanto, para selecionar os componentes foi arbitrado o uso de um capacitor de 4,7 nF, tendo em vista que tem-se menos opções de valores para capacitores no mercado, e então calculou-se a resistência a partir da equação da frequência de corte:

$$\begin{aligned} |fc| &= \frac{1}{2\pi \cdot R \cdot C} \\ 8000 &= \frac{1}{2\pi \cdot R \cdot 4,7 \cdot 10^{-9}} \\ R &\approx 4233\Omega \end{aligned} \tag{5-4}$$

Então, como não se tinha resistores disponíveis com tal valor foi feita uma aproximação utilizando um resistor de 4300Ω. Desse modo, apesar de não ser o valor exato para filtrar até 8000 Hz, esse teria um valor de atenuação próximo

o suficiente do ideal para ser satisfatório, de forma a eliminar os harmônicos indesejados e conferindo uma qualidade de áudio relevante.

Em seguida, foi-se colocado um amplificador operacional com retroalimentação negativa no intuito de implementar um seguidor de tensão. Essa estrutura foi utilizada a fim de isolar o circuito, de forma que a impedância da carga conectada na saída do circuito não alterasse a impedância do filtro passa-baixas, e consequentemente, prejudicasse seu desempenho. Dessa forma, é garantida a integridade do sinal, conforme planejado.

Segundo Workman(18), é necessário colocar um capacitor de acoplamento em um circuito de áudio para bloquear a corrente contínua, e assim evitar danos a equipamentos que sejam conectados. No entanto, para as componentes alternadas da corrente, o capacitor permite a passagem do sinal já que esse não ficará carregado devido ao fluxo alternado da corrente. Assim, quando conectada uma carga na saída do circuito, é formado um filtro passa-alta, devido associação do capacitor com a resistência do dispositivo conectado.

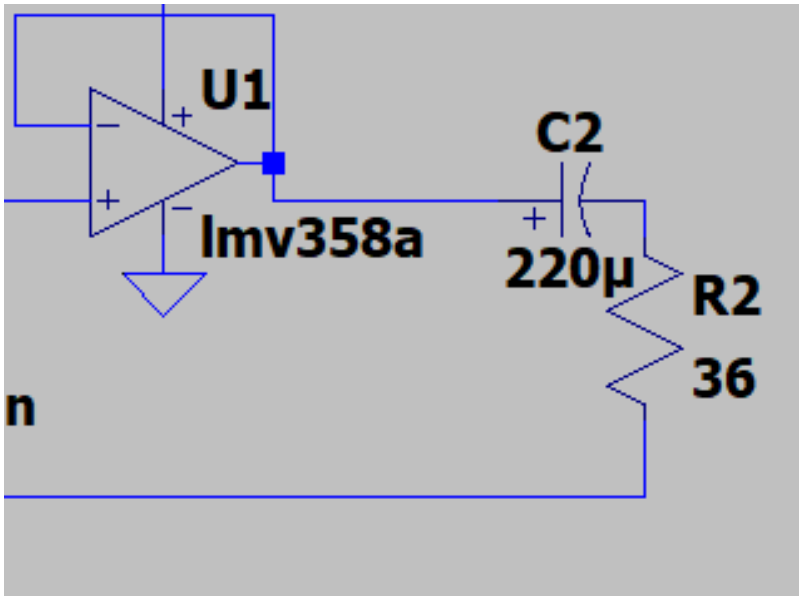

Figura 5.9: Filtro passa-alta e buffer presentes no projeto.

Sendo assim, para determinar o valor do capacitor a ser utilizado, arbitrou-se que a frequência mínima seria 20 Hz, considerando que essa seja o limiar da audição humana(23). Em seguida, mediu-se a resistência de um par de fones de ouvido para representar o equipamento conectado ao módulo de áudio. Então, após a medição, contatou-se o valor de 36Ω. Dessa forma, pôde-se determinar a capacitância do capacitor a ser adicionado através da equação da frequência de corte;

$$
\begin{aligned}
|fc| &= \frac{1}{2\pi \cdot R \cdot C} \\
20 &= \frac{1}{2\pi \cdot 36 \cdot C} \\
C &\approx 221uF
\end{aligned}
\tag{5-5}
$$

Assim, como o valor obtido não está disponível comercialmente, foi-se aproximado para 220 uF. Dessa forma, foi possível selecionar o valor para o capacitor de acoplamento de forma que o filtro passa-altas formado por ele e o elemento de carga atuassem na região de frequência desejada. Então, para testar a eficiência e funcionalidade do circuito, implementou-se essa estrutura[5] no software LTspice e feita uma análise de frequência.

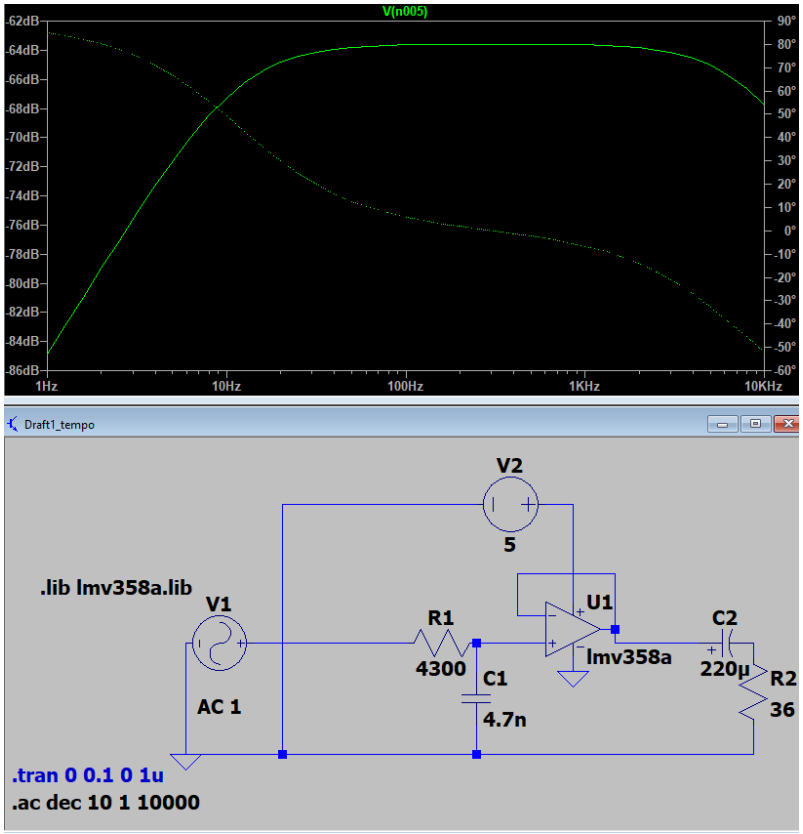

Figura 5.10: Estrutura e simulação do módulo de áudio.

Como pode-se observar na Figura 5.10, ao verificar o nó do componente de carga, é demonstrado que o circuito comporta-se como um filtro passa-faixa evidenciado pela existência de duas frequências de corte distintas e uma região central de passagem plena. Tal comportamento é compatível com o esperado, considerando que o módulo é composto por um filtro passa-baixas seguido

[5]Disponível em: `https://github.com/MontSantos/Projeto_Final_2/tree/main/LTspice_Sims`

por um passa-altas. Além disso, deve-se pontuar que as frequências nas quais ocorrem as quedas de 3 dB no gráfico, estão próximas aos valores estabelecidos para as frequências de corte, o que confirma o bom funcionamento do sistema.

5.0.4.1 Construção da PCB

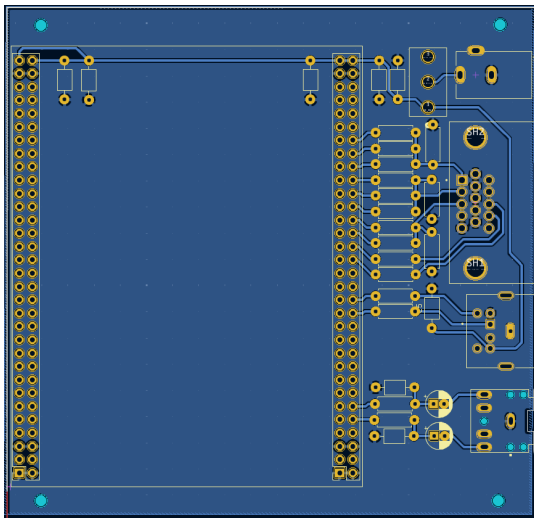

Figura 5.11: *Layout* da placa enviada para fabricação.

O desenvolvimento da *PCB* final foi uma das principais partes de todo o projeto, tendo em vista que o circuito foi todo remodelado e reorganizado. Os principais objetivos dessa etapa foram a minimização da placa, a simplificação do roteamento e a implementação o novo módulo. Para que a placa fosse da menor dimensão possível, os conectores e pinos foram agrupados do mesmo lado, o que permitiu o roteamento de trilhas curtas, melhorando o desempenho do sistema. Além disso, a forma que o roteamento foi feito possibilitou a construção de uma placa face simples, diferente do original, demonstrando a organização que foi realizada.

Assim após terminada, a placa[6] foi enviada para fabricação em uma empresa especializada utilizando a configuração demonstrada na Figura 5.11. Com a placa já fabricada e recebida, conforme na Figura 5.12, iniciou-se as etapas de montagem e testes finais. Durante a etapa de testes, foram identificados pontos à serem ajustados, especificamente no módulo de áudio.

[6]Disponível em: `https://github.com/MontSantos/Projeto_Final_2/tree/main/PCB_Projetos/Vers%C3%A3o_Final`

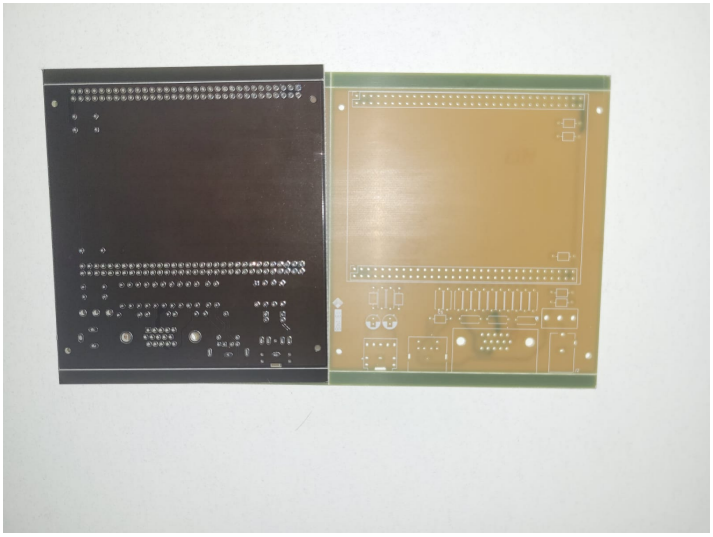

Figura 5.12: Placa após fabricada.

Para contornar os pontos identificados, foi adicionado um amplificador operacional no sistema, que não estava presente no layout fabricado, a fim de garantir o isolamento entre os dois filtros conforme explicado na seção prévia. Assim, para adaptá-lo ao sistema, utilizou-se uma placa de suporte com entradas para solda de fios. Em seguida, escolheu-se uma região próxima aos filtros, na parte superior da placa principal, em uma área sem trilhas na camada inferior. Nessa região, foi realizado cuidadosamente um furo para permitir a passagem dos fios de conexão.

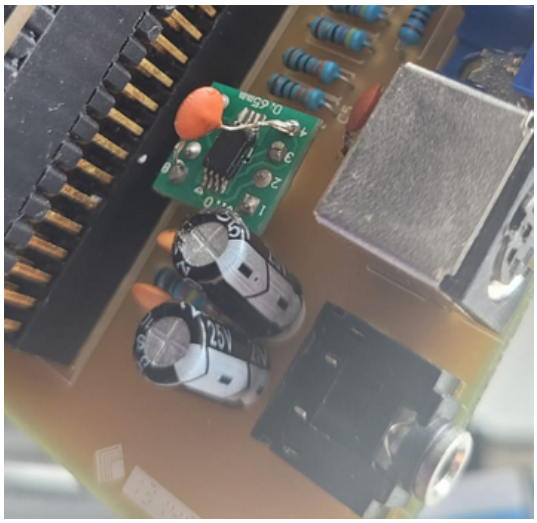

Figura 5.13: Amplificador operacional adicionado ao circuito.

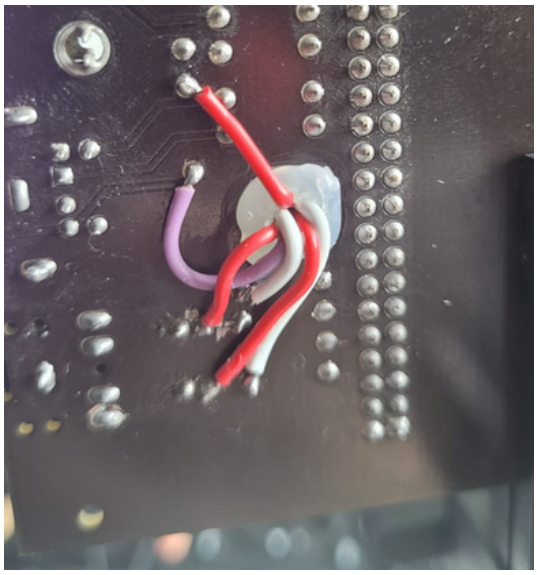

Figura 5.14: Modificação feita na placa.

Por fim, foram cortadas as trilhas necessárias para viabilizar a nova montagem, e então foram soldados os fios nos pontos correspondentes do circuito principal, permitindo a integração das entradas e saídas do amplificador operacional aos demais componentes do módulo de áudio. Dessa forma, foi garantido o isolamento entre os dois filtros e, consequentemente, o correto funcionamento deles, conforme planejado.

5.1 Aplicações e Jogos Desenvolvidos

Com a plataforma de *Hardware* finalizada, e todos os módulos básicos de interação com periféricos testados no protótipo, foi possível começar a implementação de jogos e aplicações no sistema. A intenção de tal desenvolvimento é demonstrar a capacidade do sistema, além de trabalhar conceitos de computação digital através da implementação de tais programas.

Dessa forma, durante essa fase foram implementados 4 aplicações totalmente em linguagem de descrição de *Hardware*, utilizando os módulos a disposição na placa. Sobre as aplicações, foi-se implementado:

- Pong - Um jogo clássico para dois jogadores com duas raquetes e uma bola.
- *Breakoout* - Um jogo antigo para um jogador no qual o objetivo é quebrar blocos rebatendo uma bola.
- Aplicação de Desenho - Um programa que permite ao usuário que desenhe livremente e faça arte em pixels no monitor, utilizando o teclado como movimentação e ferramenta para seleção de cores.

– Sintetizador - Uma aplicação que torna o sistema em um instrumento digital, permitindo o usuário tocar melodias utilizando o teclado para selecionar as notas.

Assim, cada aplicação demonstra pontos interessantes do sistema, e a estruturação de cada uma delas será explicada nessa seção.

5.1.1 Implementação do Pong

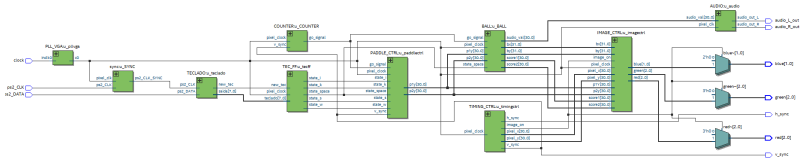

Figura 5.15: Abstração *RTL* da implementação do Pong.

O jogo implementado consiste em uma partida de tênis virtual para dois jogadores utilizando uma bola, que aumenta sua velocidade a cada vez que é rebatida. Para ganhar, basta que o jogador faça nove pontos. Os códigos referentes a esse jogo estão na pasta PONG [7].

A Figura 5.15 é um diagrama *RTL*, que consiste na representação gráfica da organização estrutural do projeto de acordo com o que foi instanciado no módulo principal, o `main.vhd`. A partir dessa abstração, pode-se visualizar as conexões entre os módulos implementados, bem como identificar os elementos utilizados na descrição do *Hardware* e compreender as estruturas construídas. Dessa maneira, esse é uma ferramenta importante para a análise do fluxo de dados.

Sendo assim, a partir de tal Figura 5.15, pode-se inferir que o projeto possui dez módulos instanciados em `main.vhd` que executam operações, incluindo o *PLL* que é essencial para gerar a frequência de clock adequada à resolução do monitor. Porém, existe um outro módulo que é uma *ROM*, instanciada pelo módulo de vídeo. Sobre os módulos presentes na imagem em questão, o sistema composto por eles possui três entradas, de forma que duas são provenientes do teclado e a outra referente ao *clock*. Além disso, o sistema também possui sete saídas que correspondem aos dois sinais de áudio, os três vetores de cor, e os dois sinais de sincronização. Assim, serão apresentados os módulos de acordo com blocos de funcionamento a que pertencem:

[7] Disponível em: `https://github.com/MontSantos/Projeto_Final_2/tree/main/HDL_Projetos`

5.1.1.1 Bloco auxiliar

O bloco auxiliar é composto apenas pelo `PLL_VGA.vhd`, que tem como função ajustar a frequência do clock do sistema para 65 MHz, que é a frequência necessária para gerar a imagem na resolução utilizada no projeto. Esse módulo foi criado a partir de ferramentas disponíveis no software de desenvolvimento.

5.1.1.2 Bloco de comunicação *PS/2*

Nesse bloco, estão todos os módulos responsáveis por captar, sincronizar e processar as entradas vindas do usuário através do teclado *PS/2*. O seu funcionamento se dá da mesma forma que o teclado implementado na seção 4.2.2, tendo os módulos `SYNC.vhd` e `TECLADO.vhd` idênticos ao que foi proposto.

Sendo assim, o único módulo que sofreu pequenas alterações, foi o código `TEC_FF.vhd`, que nesta versão foi ajustado para verificar e armazenar informações de teclas específicas de acordo com a necessidade do jogo Pong. Portanto, o módulo tem como função receber o código da tecla e o sinal de fim de leitura, e a partir disso, ele gerencia uma rede de *flip-flops* guardando e transmitindo a informação de que tecla está sendo pressionada.

Dessa forma, a lógica de funcionamento do módulo consiste em receber códigos referentes a teclas, e então verificar se corresponde a alguma das seguintes teclas: “W” e “S”, referentes a movimentação do jogador 1; “I” e “K”, referentes a movimentação do jogador 2; “espaço”, caso deseje reiniciar a partida. Assim, ao receber uma tecla de interesse um *flip-flop* guarda essa informação, até que se receba o sinal de tecla solta “F0”, seguido do código da tecla.

Essa estrutura permite que dois jogadores consigam pressionar teclas ao mesmo tempo, sem que sejam gerados conflitos nos controles, garantindo um controle responsivo e rápido, que é crucial para um jogo.

5.1.1.3 Bloco principal do jogo

O bloco principal do jogo é responsável pela lógica de controle do comportamento dos componentes do jogo, como a movimentação e posição das raquetes de ambos jogadores, o deslocamento e colisão da bola e o gerenciamento do placar.

Primeiramente, tem-se o módulo `COUNTER.vhd` que recebe o sinal de *clock* vindo do *PLL* e o sinal de sincronização vertical vindo do bloco de geração de imagem. A partir de tais sinais, o módulo tem como tarefa gerir o ritmo

de atualização do jogo através da sincronização de ambos, pois a taxa de atualização vertical é mais lenta do que a frequência do sistema. Dessa forma, através do sinal de sincronismo *go_signal* enviado na saída, são comandadas as operações dos outros dois módulos do bloco de controle, garantido que essas ocorram de forma ordenada, estabelecendo como tempo base de atualização o momento de sincronia entre ambos sinais de entrada.

Em sequência, destaca-se `PADDLE_CTRL.vhd` que é responsável pelo controle da posição vertical das raquetes dos jogadores. Esse, recebe como entradas o *clock* e *go_signal* para controlar o processo das operações a serem feitas, e também recebe os estados das teclas. A partir dos estados das teclas é possível calcular a movimentação vertical da raquete, tendo como limites as bordas da tela. Além disso, caso seja detectado que o "espaço" esteja pressionado, as posições são reiniciadas para as configurações iniciais. Sendo assim, esse entrega como saída o valor da coordenada Y de cada raquete.

Ademais, tem-se o módulo `BALL.vhd` que gerencia a lógica de movimentação e colisão da bola, além de também calcular a pontuação dos jogadores. Esse módulo tem como entradas o *clock* e *go_signal* para ditar o ritmo das operações, as coordenadas Y de cada raquete e o estado do botão de reinício, espaço.

Então, para movimentação da bola utiliza-se em seu cálculo um multiplicador referente a velocidade em cada eixo, e variáveis para definir o seu sentido. Porém, antes disso são feitas verificações de condições. Sendo assim, a cada pulso *go_signal*, o módulo verifica a pontuação da partida, caso tenha sido alcançado 9 pontos por qualquer jogador, é interrompido o fluxo da partida, esperando um sinal de reinício. Caso a pontuação limite não tenha sido alcançada, são feitas verificações de acordo com a posição atual e futura da bola no intuito de detectar alguma colisão. Em caso de colisão com a parede superior ou inferior, o sentido do eixo y da bola é invertido, e é enviado um sinal contendo um valor 2 para o módulo de som. Outra possível colisão é com uma raquete, que a sua ocorrência causa a inversão do sentido do eixo X da bola, bem como o aumento da velocidade horizontal dessa e o envio de um sinal com o valor 1 para o módulo de som. Além disso, o sistema verifica se a bola encontrou a parede esquerda ou direita da tela, e em caso positivo, incrementa a pontuação do jogador que marcou, reposiciona a bola no centro da tela, envia o valor 3 para o módulo de som e reinicia o multiplicador de velocidade horizontal.

Por fim, se nenhuma das condições citadas acima ocorreu, é finalmente feito o cálculo padrão da movimentação da bola, conforme explicado previamente. Além disso, é enviado um sinal de valor 0 para o módulo de som,

indicando que não ocorreu colisão.

Portanto, na saída do módulo são enviados os sinais referentes a posição da bola em ambos eixos, o placar de ambos jogadores e um valor que determina se algum som será emitido e, em caso afirmativo, qual som será reproduzido.

Dessa forma, o bloco de controle do jogo é responsável por verificar todas as condições da partida, desde as movimentações dos jogadores, operações com a bola, até delimitar a pontuação máxima em nove pontos.

5.1.1.4 Bloco de geração de imagem

O bloco de geração de imagem utiliza-se de dois módulos citados e explicados na seção 4.2.1, tendo um deles sofrido alterações para atender as necessidades do jogo. Dessa forma, o módulo que faz os cálculos do protocolo *VGA*, o `TIMING_CTRL.vhd`, manteve-se sem qualquer alteração.

Já o módulo `IMAGE_CTRL.vhd` precisou de modificações para conseguir gerar no monitor as figuras que compõem o jogo. Em sua entrada são recebidos sinais de diversos módulos, entre eles o sinal de *clock* do *PLL*, as coordenadas do pixel atual e o sinal de indicação de área de vídeo vindos do `TIMING_CTRL.vhd`, as coordenadas verticais de cada raquete vindas do `PADDLE_CTRL.vhd`, as coordenadas da bola e a pontuação de cada jogador vindas do `BALL.vhd`.

O funcionamento interno do módulo consiste essencialmente em verificações de acordo com as coordenadas do pixel atual, e as coordenadas dos sinais recebidos de outros módulos. Dessa forma, a cada pulso do *clock* é verificado se o pixel atual se encontra na mesma coordenada que uma figura, como a raquete ou a bola. Em caso verdadeiro é atribuída a cor apropriada para aquele pixel na saída através dos vetores das cores vermelho, verde e azul que são enviados na saída. Assim, em todo módulo não é feita nem uma operação aritmética, apenas comparações lógicas para decidir como será carregada a imagem. Dessa forma, são construídas as figuras que compõem os gráficos do jogo, através de cores selecionadas pixel por pixel.

Adicionalmente, tem-se o módulo `ROM.vhd`, que é instanciado por `IMAGE_CTRL.vhd`. A implementação desse módulo é feita com base no código `"font_rom.vhd"` do livro de Pong Chu(15). Esse módulo é uma memória de leitura, e portanto, recebe como entrada um endereço e retorna a informação armazenada em tal localidade. Dessa forma, no interior do circuito desse, estão armazenados dez caracteres em formato de *bitmap*. Esses caracteres, são números de zero a nove, que foram obtidos no projeto *VGA-Text-Generator* por Derek Wang(24). Portanto, esse módulo recebe como entrada o endereço referente a uma linha do numero do placar de um dos jogadores e retorna

um vetor de uma das linhas da matriz que compõem o bitmap do caractere. Dessa forma, o módulo fornece linha por linha as informações necessárias para o desenho de cada dígito, sendo necessários então uma instancia para cada jogador.

5.1.1.5 Bloco de produção de som

Por fim, tem-se o bloco de produção de som, composto pelo módulo `AUDIO.vhd` que teve como base o código de geração de onda explicado na seção 4.2.3. Esse recebe como entradas o *clock* e um valor vindo do módulo `BALL.vhd`. Esse sinal indica os diferentes eventos do jogo, como colisão da bola com as paredes, com as raquetes ou a marcação de pontos

Então o `AUDIO.vhd`, a cada ciclo verifica qual valor foi enviado para determinar o valor da frequência que será usado no código. Assim, com o valor da frequência selecionado, o código gera a onda quadrada conforme a lógica implementada em `OSCILADOR.vhd`. O tempo de duração da emissão do som é o tempo de atualização de *go_signal* em `BALL.vhd`, pois o sinal que define a frequência é atualizado a cada ciclo desse.

5.1.2 Implementação do *Breakout*

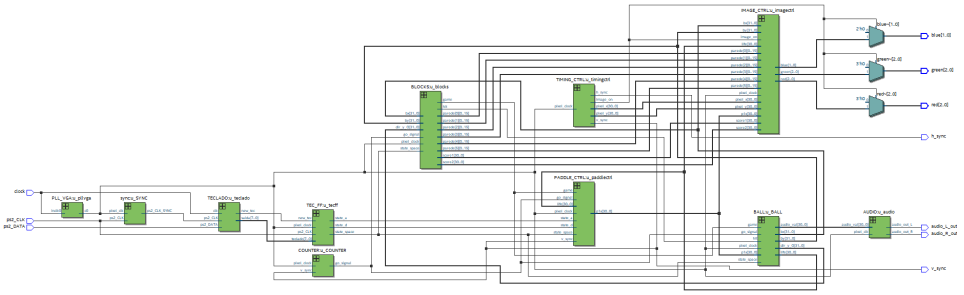

Figura 5.16: Abstração *RTL* da implementação de *Breakout*.

O *Breakout* é um jogo clássico de apenas um jogador, no qual o objetivo é quebrar os blocos disposta na tela, antes que suas vidas se esgotem. Nessa implementação, o jogo apresenta 96 blocos ,e o jogador conta com apenas 3 vidas. A seguinte implementação está disponível na pasta *BREAKOUT* [8].

O projeto do *Breakout* possui onze módulos instanciados em seu arquivo principal, contabilizando o *Phase-Locked Loop*. A partir da Figura 5.16 pode-se inferir que o sistema, assim como o Pong, possui apenas 3 entradas, sendo essas

[8]Disponível em: `https://github.com/MontSantos/Projeto_Final_2/tree/main/HDL_Projetos`

o *clock* e os dados vindos do teclado, assim como o *clock* do *FPGA*. Com relação as saídas, o sistema possui sete saídas ao todo contendo os dois sinais de áudio, os vetores das cores vermelho, azul e verde e os dois sinais de sincronização de imagem. Dessa forma, pode-se perceber que o *Breakout* reutilizou a estrutura proposta na implementação do Pong, precisando fazer algumas modificações e acréscimos para adaptar o jogo. Portanto, nessa sessão serão tratados forma mais aprofundada apenas os módulos novos ou modificados.

5.1.2.1 Bloco auxiliar

O bloco auxiliar é, novamente, composto somente pelo `PLL_VGA`. Sua implementação também se deu através de ferramentas do software de desenvolvimento. Esse existe para redefinir a frequência do sistema para que os requisitos do padrão de resolução da imagem sejam atendidos.

5.1.2.2 Bloco de comunicação *PS/2*

A estrutura do bloco de comunicação segue o mesmo padrão descrito anteriormente na implementação do Pong. Dessa forma, os módulos responsáveis pela sincronização, leitura e decodificação da informação foram reutilizados integralmente. No entanto, para implementar esse novo jogo foram necessárias pequenas alterações em `TEC_FF.vhd`.

Sobre esse último módulo, a lógica de sua operação foi mantida, ou seja, continuou-se a utilizar *flip-flops* para manter a informação de qual tecla está ativa. Nesse contexto, o *Breakout* é um jogo onde só há um jogador com movimentação no eixo X. Dessa forma, para que fosse possível controlar a raquete de maneira intuitiva, foram implementados verificadores apenas para as teclas "A" e "D", responsáveis respectivamente pela movimentação da raquete para a esquerda e direita, e um verificar para o "espaço" que reinicia o jogo.

Assim, os controles foram implementados de forma coerente e fluida para o jogo.

5.1.2.3 Bloco principal do jogo

No bloco de lógica geral do jogo, tem-se quatro módulos, incluindo os três utilizados na implementação de *Pong* e um novo módulo para fazer o gerenciamento da matriz de blocos. Nos módulos reaproveitados, precisou-se

fazer modificações no controle da raquete e no módulo responsável pela bola. O módulo `COUNTER.vhd` manteve-se sem mudanças.

O módulo `BLOCKS.vhd`, é responsável por armazenar informações referentes à matriz de blocos e ao placar do jogo. Dessa forma, esse recebe como entradas o *clock* e o sinal de sincronização *go_signal*, assim como as coordenadas e direção do eixo Y da bola, e o estado do botão de reinicio. Internamente, é criado uma matriz de 16 x 6, inicialmente preenchida por "1" em cada posição. Dessa forma, essa matriz guarda o estado de cada bloco, sendo "1" utilizado para representar os blocos ativos e "0" os blocos inativos. Além disso, também existem registradores para indicar que um bloco foi atingido e a pontuação máxima foi alcançada. Há também um contador para o placar que é composto por dois sinais que representam os dois dígitos visando facilitar a geração da imagem posteriormente.

Sobre o funcionamento do módulo de blocos, primeiramente, é verificado se o botão de reinicio foi pressionado, e em caso positivo são reconfigurados para o estado inicial a matriz, os placares e o registradores de eventos. Em seguida, é verificada se a pontuação máxima foi alcançada, sendo essa 96 pontos haja vista que esse é o número de blocos. Caso isso tenha ocorrido, o registrador de pontuação máxima assume o valor "1". Caso a pontuação máxima não tenha sido alcançada e o botão não esteja pressionado, é feita a última verificação. Portanto, o código faz a análise das coordenadas da bola, a fim de detectar se a bola está na região da matriz. Se isso for verdadeiro, é calculado o endereço do bloco na matriz, a partir das coordenadas da bola. A partir desse, pode-se verificar o estado do bloco. Se o bloco estiver ativo, então irá ocorrer uma colisão. Dessa forma, o bloco torna-se inativ, o registrador de colisão é ativado e é incrementado o placar. Se o bloco não estiver ativo, não é feita nenhuma ação. Sendo assim, na saída do módulo, são enviados os dois dígitos do placar, a matriz de blocos, e os indicadores de colisão e pontuação máxima.

O `PADDLE_CTRL.vhd`, responsável pelas lógicas de movimentação da raquete, foi modificado para oferecer suporte a apenas um jogador, que se movimenta horizontalmente. Em sua entrada, foram acrescentados o sinal de pontuação máxima e o contador de vidas do jogador vindo do `BALL.vhd`. Primeiramente é verificado se o botão de reinicio foi acionado. Em seguida, é checado se o sinal de pontuação máxima é "1" ou se o total de vidas do jogador é 0, caso isso seja verdadeiro a movimentação da raquete é bloqueada, necessitando que seja reiniciado o jogo. Após esses testes, tem-se a lógica da movimentação do jogador, sendo guiada pelos *flip-flops* das teclas "S" e "D". Assim, a partir das entradas vindas das teclas, esse calcula a movimentação da raquete no eixo X tendo como limites as bordas da esquerda e direita da

tela.

Ademais, o módulo `BALL.vhd` manteve uma estrutura semelhante, porém necessitou de adaptações para ficar de acordo com o funcionamento geral do projeto. Suas entradas, são os sinais de *clock* e *go_signal*, o sinal de colisão com blocos e pontuação máxima vindo de `BLOCK.vhd`, a posição da raquete no eixo X, e o estado do botão de reinicio. Internamente, existem sinais para armazenar a direção e multiplicador de velocidade da bola em cada eixo, um sinal para armazenar as vidas do jogador e um sinal para comunicar com o módulo de áudio. Dessa forma, a lógica de operação manteve-se parecida com a do módulo vindo do Pong, ou seja, são feitas uma série de verificações antes que seja possível realizar o cálculo da nova posição da bola.

A princípio, assim como em outros módulos, checa-se o reinicio para uma possível reconfiguração ao padrão base. Em seguida, é avaliado se foi alcançada a pontuação máxima, de modo a interromper o jogo quando detectados os 96 pontos. Após essa verificação, é analisado se a bola encostou na borda inferior na tela, caso isso seja verdadeiro, é decrementado em 1 ponto a vida do jogador, além de ser enviado o sinal de valor 4 para o módulo de áudio, indicando a perda de vida. Caso o jogador não tenha mais vidas é interrompida a execução do jogo. Caso contrário a bola é reposicionada no centro da tela e a partida é continuada. Em seguida, são feitas as verificações de colisão com a borda esquerda, direita e superior, que em caso positivo é emite-se o som de colisão, assim como inverte-se o sentido de movimentação da bola. Após isso, é verificado se o sinal de colisão com blocos é "1", de maneira que em caso positivo, é enviado o sinal de valor 3 para o módulo de áudio, alterado o sentido no eixo Y da bola, e aumenta-se a velocidade vertical da bola. A última checagem realizada é referente à colisão com a raquete, que em caso positivo: inverte-se o sentido da bola no eixo Y; incrementa-se o multiplicador de velocidade da bola no eixo horizontal; envia-se um sinal de valor 1 para o módulo de áudio; determina-se o novo sentido horizontal da bola a partir da região que a raquete foi acertada. Por fim, caso a resposta de nenhuma das checagens anteriores tenha sido verdadeira, então é feita a movimentação comum da bola.

Assim, esse bloco administra a lógica principal do jogo através de seus cálculos, com a adição de um novo módulo para armazenar a informação dos blocos e placares. Dessa forma, o jogo pode ir até 96 pontos e o jogador possui 3 vidas.

5.1.2.4
Bloco de geração de imagem

Para gerar a imagem, manteve-se a mesma estrutura do Pong, que se deu através de um módulo para implementar o protocolo *VGA* e o módulo que define as cores de cada pixel da tela. Desses, apenas o último foi alterado, pois foram acrescentados novos elementos a imagem.

Sendo assim, o módulo `IMAGE_CTRL.vhd` recebe como entradas o *clock*, o sinal que indica se o vídeo está na região ativa, as coordenadas do pixel atual, a coordenada X da raquete, as coordenadas da bola, os dois dígitos do placar da partida, a matriz de blocos, e o total de vidas do jogador. Sendo assim, esse módulo a cada pulso, verifica a coordenada do pixel para determinar o que desenhar. Portante, esse é responsável por desenhar as bordas, bola, raquete, placar, vidas do jogador e o conjunto de blocos ativos. Sobre os blocos desenhados, a cor desses varia de acordo com o índice da linha que esses se encontram, semelhante ao jogo clássico. Para gerar, o placar e o desenho das vidas, foi-se utilizada a `ROM.vhd` que foi alterada para receber dois endereços e retornar duas linhas distintas. Além disso, foi acrescentado um *bitmap*, feito pelo autor, em forma de coração para ser posto ao lado do número de vidas. Dessa forma esse arquivo é instanciado duas vezes, a primeira para retornar os dois dígitos do placar e a segunda para a representação de vidas do jogador.

5.1.2.5
Bloco de produção de som

O último bloco a ser tratado é referente a geração de som. A sua operação ocorre de mesma forma que o bloco feito no Pong. No entanto, para essa implementação, foi-se acrescentado mais um valor válido para o sinal de áudio vindo de `BALL.vhd`, assim aumentando a quantidade de sons válidos para o jogo.

5.1.3
Implementação da interface de desenho

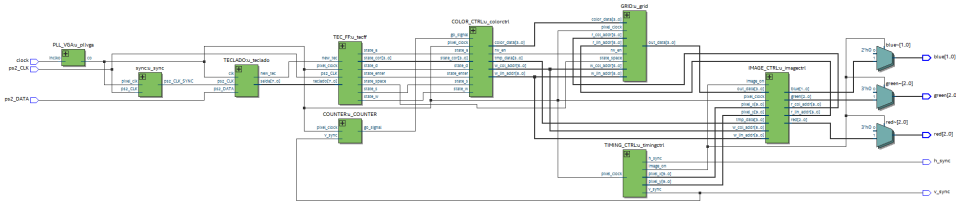

Figura 5.17: Abstração *RTL* da interface de desenho.

A implementação da interface de desenho se deu através uma tela composta por uma matriz 63 X 47 quadrados que podem ter as suas cores definidas pelo usuário. Para movimentar-se pela tela, podem ser utilizadas as teclas "W", "A", "S" e "D". Sobre a matriz essa está implementada através uma *RAM* que para cada célula, são aceitos valores de 0 a 15 que representam as cores. Então, para a escolha de cores, as teclas de números de "0" a "9" e as letras "T","Y","U","I","O" e "P" foram configuradas com cores que o usuário pode selecionar. Então, para confirmar que um quadrado seja colorido, basta apertar "*Enter*", que será salvo o valor na *RAM*. Caso queira apagar todas as informações da tela, pode-se apertar a tecla "espaço", e a *RAM* será preenchida com o valor correspondente a cor branco. A implementação em questão está na pasta *PAINT* [9].

A estrutura de projeto é composta por 9 módulos instanciados no arquivo `main.vhd`. A partir da Figura 5.17 pode-se perceber que a sua estrutura é semelhante aos outros dois jogos implementados, mas com algumas diferentes como a ausência do módulo de áudio e a presença do módulo `GRID.vhd`, que implementa a *RAM* do sistema. Ao todo o sistema possui três entradas, assim como os outros jogos, sendo elas os sinais de *clock* e dados vindos do teclado, e o clock do sistema. Por não possuir módulo de áudio, esse apresenta apenas cinco saídas, que são referentes ao bloco de vídeo, sendo elas o vetores das três cores e os dois sinais de sincronismo da tela.

5.1.3.1 Bloco auxiliar

Esse bloco, assim como nos jogos explicados previamente, possui somente o `PLL_VGA.vhd` em seu interior e foi implementado de mesma forma. Portanto, sua função é ajustar o *clock* do sistema para que esteja de acordo com a frequência necessária para atender a resolução utilizada.

5.1.3.2 Bloco de comunicação *PS/2*

Assim, como o *Breakout*, foi-se utilizada a estrutura de três módulos utilizada no Pong. Dessa forma, `TECLADO.vhd` e `SYNC.vhd` mantiveram-se sem alterações. O único módulo a ser modificado foi `TEC_FF.vhd`, que foram alteradas as teclas monitoradas. Para movimentação do jogador tem-se *flip-flops* para as teclas "W", "A", "S" e "D", o que permite a movimentação em diagonal por exemplo. Para a escolha de cor, o usuário tem acesso a uma paleta

[9]Disponível em: `https://github.com/MontSantos/Projeto_Final_2/tree/main/HDL_Projetos`

com 16 valores, podendo selecionar alguma das teclas de “0” a “9” ou alguma das letras “T”,“Y”,“U”,“I”,“O” e “P”, para que então um *flip-flop* receba o valor da cor associada a essa tecla. Deve-se pontuar que ao soltar a tecla de cor, o *flip-flop* não muda o valor interno, e para isso é necessário escolher outra cor. No intuito de confirmar que o usuário deseja pintar algum quadrado, tem-se um *flip-flop* que armazena o estado da tecla *Enter*. Por fim, é monitorado o *espaço* caso deseja-se apagar o que foi colorido.

5.1.3.3 Bloco principal da aplicação

O bloco principal do sistema tem como função realizar as operações de movimentação do cursor pela tela, colorir e guardar as informações dos quadrados em uma *RAM*. Sendo assim, esse bloco é composto por três módulos, o `COUNTER.vhd`, o `COLOR_CTRL.vhd` e o `GRID.vhd`. O primeiro deles, foi implementado de mesma forma que os módulos homônimos presentes no *Breakout* e Pong, ou seja, esse apenas sincroniza o *clock* do sistema com o sinal de sincronização vertical vindo do bloco que gera a imagem.

O módulo `GRID.vhd` é onde foi feita a implementação da *RAM* com endereços em forma de matriz utilizando *flip-flops* para guardar a informação. Uma *RAM* consiste em um tipo de memória que pode ser acessada e reescrita. No intuito de permitir a leitura e a escrita nessa estrutura de forma simultânea, foi-se implementada uma *RAM Dual Port*, que consiste em uma estrutura com endereço de leitura e escrita distintos permitindo assim que ambas operações ocorram paralelamente. Sendo assim, como entradas, esse módulo recebe o *clock*, o estado do botão de reinicio, os endereços das linhas e colunas desejados para leitura e escrita, além do sinal de controle que indica uma escrita, e o valor a ser escrito na *RAM*. Internamente, a matriz é inicializada através de um sinal de 63 x 47 unidades, que é preenchida por zeros, ou seja, é feita uma tela branca. A cada pulso do *clock* é verificado se o usuário deseja reiniciar, em caso positivo a matriz é preenchida com zeros novamente. Caso contrário, é verificado se o sinal de indicação da escrita é “1”. Se for verdadeiro, então é acessado o endereço da *RAM*, a partir dos índices transmitidos por `COLOR_CTRL.vhd`, e esse é preenchido pela cor selecionada pelo usuário. Com relação a saída, essa é a informação contida na RAM no endereço de leitura enviados pelo bloco de imagem.

Por fim, tem-se o módulo `COLOR_CTRL.vhd`, responsável pelo controle dos comandos enviados pelo usuário seja a movimentação do cursor, a escolha de uma cor, ou a opção de pintar um quadrado. Esse recebe em sua entrada os sinais de *clock* e o pulso de sincronização *go_signal*, os *flip-flops* das teclas de

movimentação, seleção de cor e *Enter*. Internamente, são declarados sinais para armazenar a linha e coluna que se encontra o cursor, um registrador de controle, um sinal para armazenar a cor selecionada pelo usuário e outro para armazenar a cor a ser enviada para a *RAM*. No intuito de garantir uma movimentação mais responsiva, foi implementado um contador que suaviza o deslocamento do cursor na tela, de modo que o cursor e sua movimentação só podem ser atualizados após um intervalo de aproximadamente 100 milissegundos. Sendo assim, após esse período, o sinal referente a cor do cursor é preenchido pela cor transmitida pelo *Flip-flop* correspondente e o sinal de habilitação de escrita torna-se "0".

Em seguida, é verificado se as teclas de movimentação estão sendo pressionadas. Se houver detecção, o endereço do cursor da matriz é atualizado conforme a direção indicada pela tecla, tendo como limites os cantos da tela, ou seja, o valor mínimo e máximo para as linhas e colunas. Assim é garantido que os limites não são ultrapassados. Ademais, é verificado se o usuário apertou *Enter*. Caso tenha sido detectada tal tecla, o registrador de controle torna-se "1" e o sinal para envio de cor recebe a cor do cursor. Sendo assim, na saída do módulo são enviados, os endereços da linha e coluna do cursor, o valor referente a cor do cursor, o valor da cor a ser escrita na *RAM*, além do sinal de habilitação de escrita, que é resultado da operação lógica entre o registrador de controle e o *go_signal*. Assim, é assegurada a escrita na memória de forma sincronizada, constituindo assim um sistema robusto.

5.1.3.4 Bloco de geração de imagem

A estrutura que é responsável formar a imagem na tela, segue o mesmo padrão dos módulos dos jogos, através de dois módulos. O primeiro deles é o `TIMING_CTRL.vhd`, que implementa a lógica do protocolo *PS/2*, e portanto não precisou-se fazer alterações. Já o módulo `IMAGE_CTRL.vhd`, também segue o padrão de lógica dos módulos homônimos implementados em seções prévias, porém sofrendo adaptações.

Então, esse recebe o *clock*, o sinal de região ativa da imagem, as coordenadas do pixel, o endereço da linha e coluna do cursor, a cor do cursor e a cor do quadrado lido na matriz. Em seguida, na arquitetura, são declarados os sinais para as três cores, assim como dois contadores e dois índices, responsáveis por calcular o endereço na matriz referente ao quadrado que se encontra o pixel. Na declaração do processo é definida uma variável para receber o valor da cor, tendo em vista que é necessário que essa seja atualizada no mesmo instante que recebe o valor. Assim, esse módulo tem a função de gerar no monitor as bordas

do gráfico e os quadrados que compõem a imagem. Caso o pixel esteja dentro da área válida para desenho, então, para determinar a cor de cada pixel, é feita um cálculo através de contadores para determinar quais os índices na memória referentes ao quadrado que o pixel se encontra. Em seguida, é verificado se os índices do pixel são os mesmos do cursor, caso sejam, a variável cor assume o valor da cor do cursor, caso contrário assume o valor da cor do endereço lido na *RAM*. Então a partir de tal variável, é definida a cor que será transmitida através dos três sinais de cores. Sendo assim, como saídas, esse módulo tem os três vetores referentes às cores do pixel gerado e o endereço na *RAM* do bloco referente a localização do pixel.

5.1.4 Implementação do sintetizador de áudio

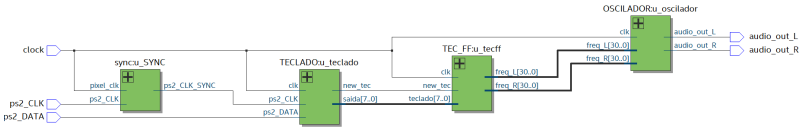

Figura 5.18: Abstração *RTL* do sintetizador.

A última aplicação implementada é um sintetizador de áudio, que funciona basicamente como um instrumento musical. Assim, foram associadas 48 notas, ou 4 oitavas, a 48 teclas disponíveis no teclado, começando cada oitava pela nota dó e terminando na nota si. Dessa forma, ao pressionar a tecla é gerado o som na saída da nota correspondente. O projeto no qual estão os códigos do sintetizador chama-se *SYNTH* [10].

Como pode-se observar na Figura 5.18, essa última aplicação possui a estrutura muito simples em comparação às outras, possuindo apenas quatro módulos instanciados no arquivo principal. Em sua operação, só existem dois blocos, o bloco de comunicação *PS/2* utilizando a estrutura usada nas outras implementações, e o bloco de áudio que também funciona de forma semelhante aos que foram implementados nas outras aplicações. Assim, o sistema possui as mesmas três entradas dos outros projetos, e apenas duas saídas referentes ao módulo de áudio.

[10]Disponível em: `https://github.com/MontSantos/Projeto_Final_2/tree/main/HDL_Projetos`

5.1.5
Bloco de comunicação PS/2

Então, assim como nas outras aplicações, como no Pong, se utilizam os três módulos de mesma origem, mantendo `SYNC.vhd` e `TECLADO.vhd` sem alterações. Já `TEC_FF.vhd` recebe como entradas o *clock*, o sinal de tecla lida e o código da tecla. A partir do código da tecla, é verificado se essa é uma das teclas especificadas para o programa. Em caso positivo, é atribuído a apenas um *flip-flop*, o valor da frequência correspondente a nota atribuída aquela tecla. Caso a tecla seja solta, a frequência torna-se zero. Optou-se por apenas um *flip-flop*, para a reprodução de notas, pois criar um *flip-flop* para cada tecla tornaria a implementação consideravelmente mais complexa já que seriam necessárias lógicas para lidar com múltiplas teclas sendo pressionadas simultaneamente.

Em seguida, na saída, é enviado o *flip-flop* com o valor da frequência através de dois sinais, sendo um para cada lado do áudio.

5.1.6
Bloco de produção de som

A operação do último bloco ocorre de forma semelhante aos blocos homônimos prévios. Assim, esse recebe o valor da frequência a ser emitida, e através do mesmo processo, gera uma onda quadrada através da alternância entre estados do sinal que é enviado para a saída. Assim, é gerado o som do instrumento digital comandado pelo usuário.

6
Implementação e avaliação

Nesse capítulo serão tratados os resultados da compilação e as formas que foram testados cada jogo e aplicação construídos ao longo do projeto, assim como a placa. Tendo que o sistema implementado consistem em uma plataforma interativa com os usuários, foram realizados testes práticos com o sistema executando as aplicações. Tais testes foram conduzidos para validar o funcionamento do conjunto como um todo.

No intuito de testar os jogos Pong e *Breakout*, foram realizadas partidas completas para que fossem verificados diferentes formas de situações de colisão, detecção de pontuação e fim de jogo, além dos processos de reinicio e reconfiguração. Ademais, foi-se avaliado o comportamento dos controles e fluidez, buscando garantir que esses não sofram com interferência, proporcionando uma melhor experiência para os usuários.

Na aplicação da interface de desenho, foram feitos desenhos na tela, de forma que fosse possível testar as capacidades de movimentação e seleção de cores, além de avaliar a escrita na memória do projeto em diferentes regiões da tela.

Já para o sintetizador, foram executadas melodias de temas de jogos, músicas conhecidas e escalas musicais, a fim de averiguar a fluidez dos comandos do instrumento, bem como confirmar se as frequências produzidas estavam coerentes com as notas desejadas.

6.1
Execução de testes funcionais

6.1.1
Pong

A Figura 6.1 ilustra as especificações da compilação. Como pode-se ver, o sistema demonstrou-se leve e simples, visto que foram necessários apenas 324 registradores, e menos de 5% da capacidade máxima de elementos lógicos do *FPGA*. Seguindo a compilação, configurou-se o *FPGA* com o projeto e então realizados os testes.

Flow Status	Successful - Sat Jul 5 11:30:37 2025
Quartus Prime Version	23.1std.1 Build 993 05/14/2024 SC Lite Edition
Revision Name	PONG
Top-level Entity Name	MAIN
Family	Cyclone IV GX
Device	EP4CGX150DF27I7
Timing Models	Final
Total logic elements	2,438 / 149,760 (2 %)
Total registers	324
Total pins	15 / 426 (4 %)
Total virtual pins	0
Total memory bits	0 / 6,635,520 (0 %)
Embedded Multiplier 9-bit elements	0 / 720 (0 %)
Total GXB Receiver Channel PCS	0 / 8 (0 %)
Total GXB Receiver Channel PMA	0 / 8 (0 %)
Total GXB Transmitter Channel PCS	0 / 8 (0 %)
Total GXB Transmitter Channel PMA	0 / 8 (0 %)
Total PLLs	1 / 8 (13 %)

Figura 6.1: Resultados da compilação do Pong.

O procedimento de testes se deu de forma simples, foram realizadas partidas completas entre dois usuários. Essa abordagem permitiu avaliar, de maneira integrada, a responsividade das teclas, a movimentação das raquetes e da bola, o funcionamento do sistema de colisões, a precisão do sistema de contagem, bem como a emissão de som de acordo com as condições específicas.

Ao longo dos testes, foi constado que os dois jogadores conseguiam acionar teclas de forma simultânea, sem interferências entre si, mesmo que a interface *PS/2* utilize apenas um canal para transmissão de dados. Esse bom desempenho foi permitido pela implementação dos *flip-flops* para as teclas, que guardam o estado da tecla pressionada e só alteram se receberem os sinais correspondentes. Assim, são evitadas leituras erradas devido ao uso de múltiplas teclas simultâneas, o que é fundamental para que o jogo seja agradável.

Ademais, sobre o sistema de movimentação das raquetes e bola, esses demonstraram-se efetivos e corretos, visto que os movimentos ocorrem de maneira fluida, estando a todo momento dentro das limitações impostas para o jogo. Sobre o sistema de colisões, esse demonstrou-se satisfatório, haja vista que a bola reflete quando colide com a raquete ou parede. No entanto, foi-se observado que em situações específicas, como por exemplo quando a bola está em alta velocidade e avançando em diagonal em direção ao canto da raquete, é possível que a bola atravesse o canto da raquete devido a forma que é calculada a detecção. Entretanto, como mencionado, esses ocorridos foram altamente incomuns, e sendo assim não comprometendo o jogo.

A partir da Figura 6.2, pode-se observar uma situação de jogo onde os os dois jogadores já pontuaram, e naquele instante a bola colide com a raquete de um deles, demonstrando o funcionamento de múltiplos sistemas do jogo.

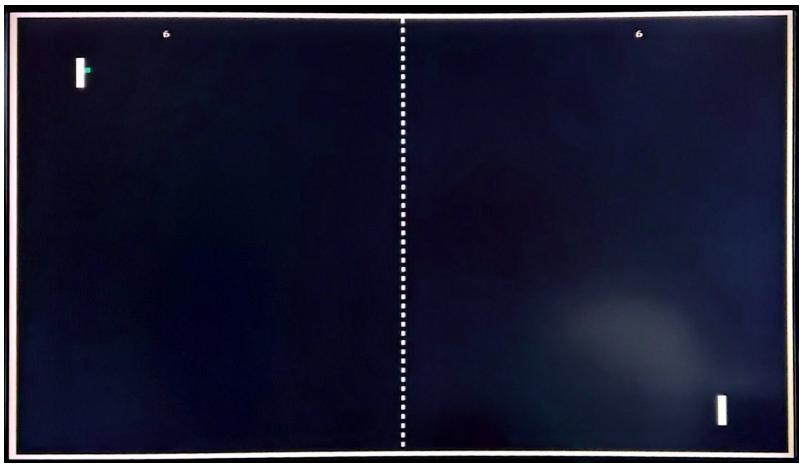

Figura 6.2: Colisão da bola com raquete.

Em relação ao sistema de som, esse demonstrou um bom funcionamento. Sua execução está atrelada a diferentes eventos da partida, como colisões ou pontuações, que possuem notas distintas para representá-los. Além disso, o tempo de execução de cada nota foi avaliado como bom, pois é suficiente para informar o evento ocorrido e não ser incômodo.

Por fim, o sistema de pontuação e gerência geral do jogo funcionou de forma correta. Através dos testes, foi possível demonstrar a efetividade da contagem de pontos, visto que essa só é incrementada quando ocorre de fato um ponto. Além disso, constatou-se, que a cada ponto marcado, a bola é reposicionada corretamente ao longo do eixo vertical referente a metade do eixo X da tela. Por fim, o mecanismo de fim de jogo também mostrou-se funcional, interrompendo o jogo quando um jogador alcança nove pontos, e assim aguardando o sinal de reinicio. Sobre esse comando, pode-se afirmar que esse também operou corretamente, visto que para restaurar as configurações iniciais do jogo, basta pressionar a tecla “espaço”.

Dessa forma, a implementação do Pong funcionou conforme planejado, mantendo uma jogabilidade efetiva e responsiva, apresentando respostas fluidas aos comandos. As funcionalidades implementadas operaram de maneira adequada, possibilitando a boa execução das partidas.

6.1.2 *Breakout*

Flow Status	Successful - Sun Jul 6 11:05:44 2025
Quartus Prime Version	23.1std.1 Build 993 05/14/2024 SC Lite Edition
Revision Name	Breakout
Top-level Entity Name	MAIN
Family	Cyclone IV GX
Device	EP4CGX150DF27I7
Timing Models	Final
Total logic elements	5,817 / 149,760 (4 %)
Total registers	438
Total pins	15 / 426 (4 %)
Total virtual pins	0
Total memory bits	0 / 6,635,520 (0 %)
Embedded Multiplier 9-bit elements	0 / 720 (0 %)
Total GXB Receiver Channel PCS	0 / 8 (0 %)
Total GXB Receiver Channel PMA	0 / 8 (0 %)
Total GXB Transmitter Channel PCS	0 / 8 (0 %)
Total GXB Transmitter Channel PMA	0 / 8 (0 %)
Total PLLs	1 / 8 (13 %)

Figura 6.3: Resultados da compilação do *Breakout*.

A Figura 6.3 ilustra o resultado da implementação do jogo *Breakout*. Pode-se perceber que a quantidade de elementos lógicos aumentou em pouco mais que o dobro, quando comparado com a implementação do Pong, enquanto os registradores utilizados aumentaram em menos de 120 unidades. No entanto, é possível afirmar que ainda foi mantido o baixo custo para o desempenho do sistema, haja vista que foi exigida menos de 5% da capacidade total do *FPGA*. Sobre os testes, esses foram conduzidos através de partidas de completas, nas quais foram avaliados componentes de movimento, colisão, detecção e atualização da matriz de blocos, incrementação de pontos, gerenciamento de vidas e emissão de som.

A Figura 6.4 apresenta o jogo em sua situação inicial, com a bola e a raquete em suas posições padrão, apenas aguardando o comando de inicio, além de ter todos os blocos dispostos na parte superior da tela. Esse último fato, evidencia o bom funcionamento da matriz. Ademais, pode-se notar na parte superior, a pontuação estando em zero e o jogador com o máximo de vidas, o que é coerente com o contexto.

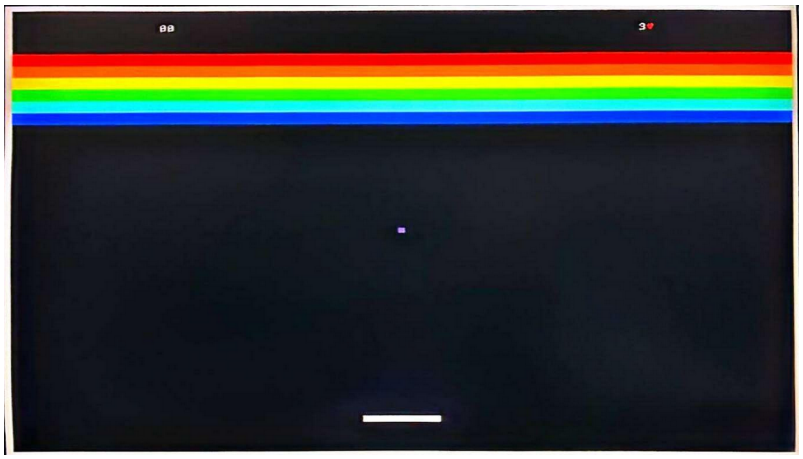

Figura 6.4: Início de partida.

Em sequência, a Figura 6.5 demonstra um jogo em andamento, de forma que alguns blocos já foram atingidos. Outro ponto a se destacar está relacionado ao placar, que está devidamente atualizado com a quantidade de blocos quebrados, explicitando que a detecção de colisão de blocos e o sistema de armazenamento de informações dos blocos estão em bom funcionamento.

Sobre o sistema de colisão com a raquete, esse foi implementado baseando-se no jogo previamente estruturado. Sendo assim, pode-se afirmar que em casos raros, quando a bola se movimenta de forma veloz em diagonal, é possível que a bola atravesse a parte mais extrema da raquete devido a forma que é calculada a detecção. No entanto, novamente, esses casos não foram suficientes para prejudicar o jogo.

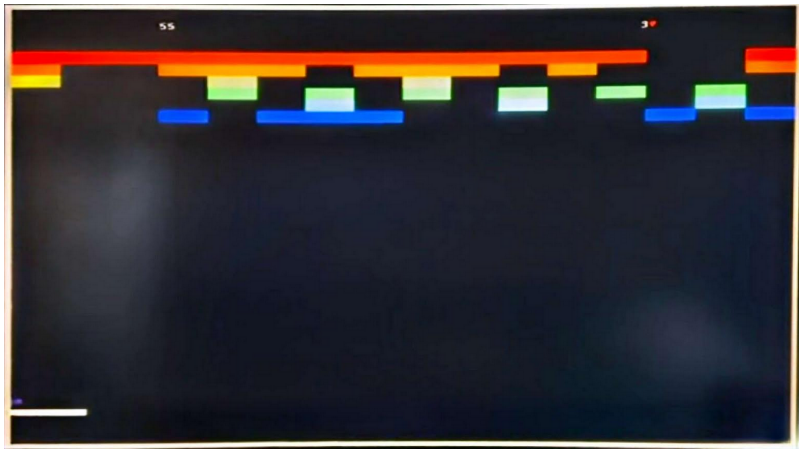

Figura 6.5: Partida em execução.

Por fim, na Figura 6.6 está representada a situação de fim de jogo, na qual a movimentação dos elementos foi interrompida apenas aguardando o comando de reinicio. Assim, todos os blocos da matriz foram quebrados, e portanto não estão mais presentes na imagem. Além disso, o placar está devidamente

atualizado, evidenciando que o gerenciamento da matriz e contagem estão funcionando corretamente. Por fim é possível ver que a vida foi decrementada em um ponto, o que demonstra que ao longo da partida esse valor é atualizado de forma dinâmica.

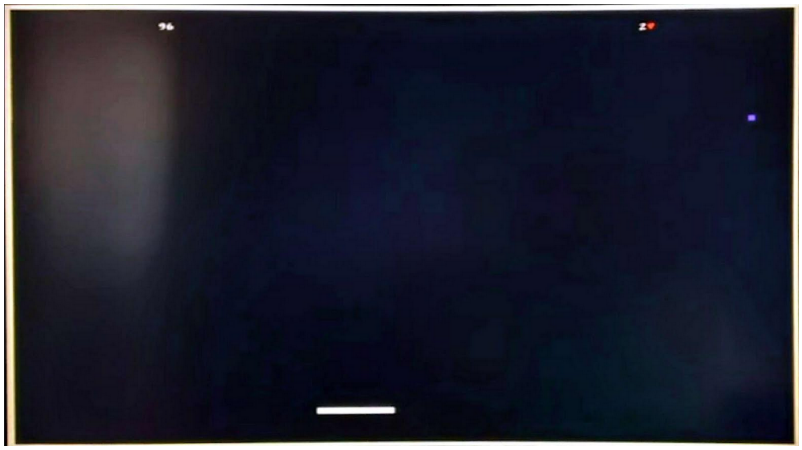

Figura 6.6: Fim de jogo.

6.1.3 Interface de desenho

Flow Status	Successful - Sun Jul 6 14:40:53 2025
Quartus Prime Version	23.1std.1 Build 993 05/14/2024 SC Lite Edition
Revision Name	PAINT
Top-level Entity Name	MAIN
Family	Cyclone IV GX
Device	EP4CGX150DF27I7
Timing Models	Final
Total logic elements	23,311 / 149,760 (16 %)
Total registers	12006
Total pins	13 / 426 (3 %)
Total virtual pins	0
Total memory bits	0 / 6,635,520 (0 %)
Embedded Multiplier 9-bit elements	0 / 720 (0 %)
Total GXB Receiver Channel PCS	0 / 8 (0 %)
Total GXB Receiver Channel PMA	0 / 8 (0 %)
Total GXB Transmitter Channel PCS	0 / 8 (0 %)
Total GXB Transmitter Channel PMA	0 / 8 (0 %)
Total PLLs	1 / 8 (13 %)

Figura 6.7: Resultados da compilação da interface de desenho.

O resultado de compilação presente na Figura 6.7 evidencia que o circuito implementado nessa aplicação necessitou de uma estrutura significativamente mais robusta que os outros jogos. Para que fosse estruturado por completo, a quantidade de elementos lógicos saltou em aproximadamente dez vezes, quando comparado à quantidade necessária para implementação do Pong, enquanto a quantidade de registradores ultrapassou as 10000 unidades. Todos essas necessidades de implementação são refletidas no aumento do tempo de compilação notado ao longo dos testes.

Esse aumento na complexidade está relacionado a forma que foi implementada a *RAM*. No projeto, optou-se por utilizar uma matriz de *flip-flops*, ao invés do blocos de memória interna disponibilizados pelo próprio *FPGA*. Sobre a matriz, essa conta com 2961 endereços, haja vista que são 63 colunas e 47 linhas, responsáveis por armazenar um entre 16 possíveis valores, ou seja, informações de quatro bits. Dessa forma, cada endereço utiliza quatro *flip-flops*, um para cada bit de informação, totalizando 11844 *flip-flops* somente para a *RAM*, o que justifica a quantidade total de registradores utilizados.

No intuito de realizar testes para verificar o funcionamento do programa, foram feitos desenhos e artes, para assim averiguar a movimentação do cursor, as cores disponibilizadas, e também o sistema de armazenamento de informações passadas pelo usuário. Além disso, tinha-se o interesse de visualizar o que poderia ser criado a partir das ferramentas disponibilizadas pelo sistema

Nesse contexto, Figura 6.8 é um exemplo prático do potencial da interface de desenho construída ao longo do projeto. No centro da imagem, encontra-se um barco, desenhado por um usuário utilizando diferentes tons da paleta de cores do projeto.

A partir da imagem, pode-se perceber que é possível aplicar efeitos como sombreamento, a partir das cores disponíveis. Dessa forma demonstra-se a profundidade e versatilidade que podem ser alcançados com as ferramentas disponibilizadas pelo projeto. Além disso, no canto esquerdo inferior estão demonstradas as cores disponíveis, que abrangem diferentes tons. Então, foi demonstrado que mesmo com recursos limitados, como apenas 16 cores e usando menos do que a capacidade máxima de células disponíveis para desenho, é possível criar figuras bem detalhadas.

Figura 6.8: Desenho de barco utilizando o projeto.

Por fim, é importante pontuar que ao longo dos testes, inicialmente percebeu-se que o cursor movimentava-se várias células de uma vez, o que difi-

cultava muito o usuário para utilizar a ferramenta. Sendo assim, implementou-se um sistema de contagem de 100 milissegundos, conforme mencionado na Subsubseção 5.1.3.3, para que fosse atualizada a posição do cursor, tornando assim mais responsiva a movimentação.

6.1.4 Sintetizador de áudio

Flow Status	Successful - Sat Jul 5 11:50:47 2025
Quartus Prime Version	23.1std.1 Build 993 05/14/2024 SC Lite Edition
Revision Name	SYNTH
Top-level Entity Name	MAIN
Family	Cyclone IV GX
Device	EP4CGX150DF27I7
Timing Models	Final
Total logic elements	1,440 / 149,760 (< 1 %)
Total registers	113
Total pins	5 / 426 (1 %)
Total virtual pins	0
Total memory bits	0 / 6,635,520 (0 %)
Embedded Multiplier 9-bit elements	0 / 720 (0 %)
Total GXB Receiver Channel PCS	0 / 8 (0 %)
Total GXB Receiver Channel PMA	0 / 8 (0 %)
Total GXB Transmitter Channel PCS	0 / 8 (0 %)
Total GXB Transmitter Channel PMA	0 / 8 (0 %)
Total PLLs	0 / 8 (0 %)

Figura 6.9: Resultado de compilação do sintetizador.

A implementação do sintetizador, conforme esperado, foi a que teve a estrutura mais simples, visto que utilizou-se apenas teclado e áudio. Dessa forma, Figura 6.9 ilustra as baixas exigências para a síntese do sistema, destacando o uso de menos de 1% da capacidade de elementos lógicos disponíveis, e de pouco mais de 100 registradores.

Para a realização de testes, primeiramente, verificou-se as notas produzidas estavam sendo produzidas com fidelidade. Dessa forma, para averiguar tal desempenho, utilizou-se um afinador de instrumentos que ao ser posicionado sob a superfície que emite o som, determina qual nota está sendo reproduzida e o quanto precisa se ajustar para que essa esteja na tonalidade correta. Assim, ao longo dos testes, grande parte das 48 notas foi reconhecida pelo aparelho como corretamente afinadas. Já as demais, o dispositivo conseguiu identificar a nota, porém apontava a necessidade de pequenos ajustes para atingir a frequência exata. Essas pequenas diferenças se devem ao fato do projeto estar limitado a utilizar aproximações de valores inteiros para as frequências, sendo que algumas notas possuem valores fracionados.

A Figura 6.10 retrata um teste prático do sistema utilizando um par de fones de ouvido como saída de áudio e um afinador junto a um dos fones. Assim, como pode-se observar a tecla “V”, correspondente a nota Mi, está sendo pressionada por um usuário, enquanto o afinador reconhece corretamente a nota e mostra no visor que a afinação está certa.

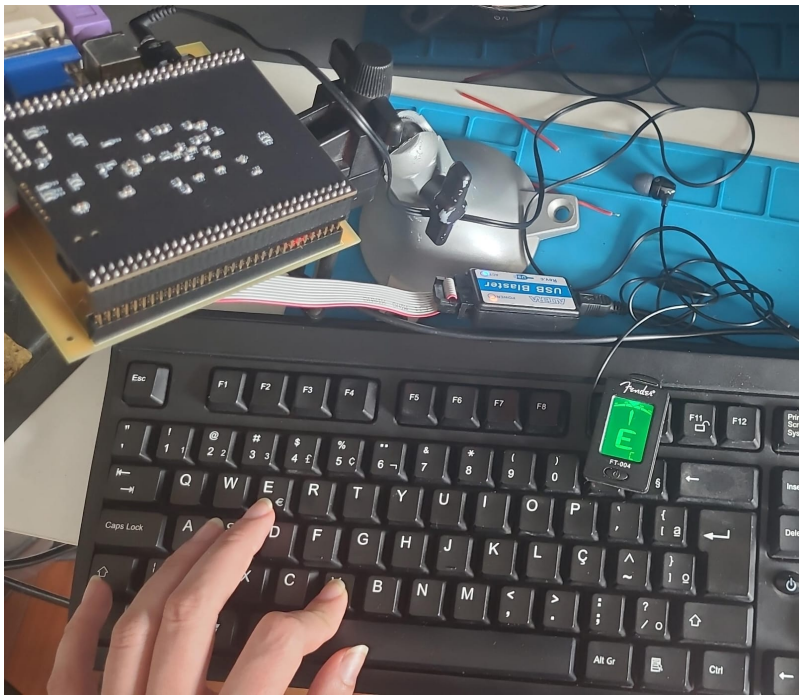

Figura 6.10: Afinador reconhecendo a nota Mi.

Por fim, após verificar que o sistema era capaz de produzir notas com precisão satisfatória, testou-se a capacidade de reproduzir melodias utilizando o projeto. Para isso, utilizou-se o sistema para a execução de músicas a fim de verificar como o sistema desempenha com teclas sendo pressionadas por períodos diferentes, tal qual um instrumento musical. Assim, após esses últimos testes, observou-se que ao pressionar uma tecla X, e em seguida pressionar e soltar outra tecla Y, o som de X não voltava a ser reproduzido, característica decorrente da forma que foi implementado o registro de qual tecla está sendo pressionada. No entanto, constatou-se que, com a prática, o usuário pode contornar essa limitação, tal qual um instrumento de cordas exige conhecimento técnicos, como a pressão correta a ser aplicada sob a cordas, obtidos através do contato com o instrumento. Sendo assim, pode-se afirmar que essa implementação consegue cumprir sua função como instrumento musical digital.

6.1.5 Placa

Após a implementação de todos os programas citados previamente com sucesso, pode-se afirmar que a *PCB* montada especificamente para esse projeto, atendeu a todos os pontos planejados para essa. Sendo assim, pode-se dizer que a placa integrou vídeo, áudio e teclado de forma eficiente, sem apresentar falhas na comunicação ou instabilidades na rede de alimentação. As alterações realizadas melhoraram o desempenho do circuito de áudio, garantindo um

melhor desempenho para o som produzido pelo sistema como um todo.

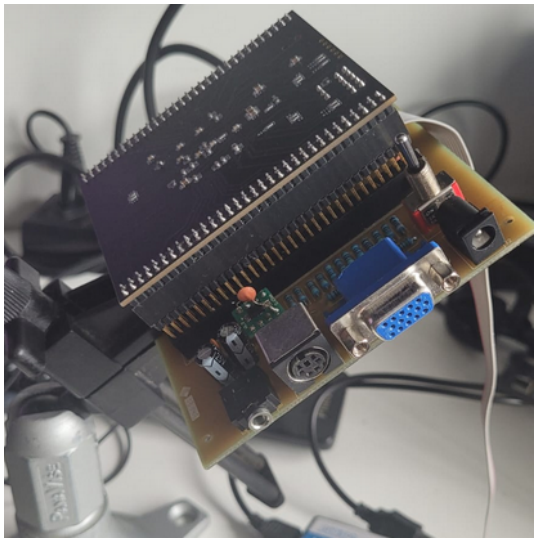

Figura 6.11: Placa final montada com a placa de desenvolvimento *FPGA* posicionada.

Assim, é demonstrada a capacidade de tal *PCB* como uma plataforma de suporte para implementações e estudos de programas interativos utilizando uma placa de desenvolvimento com um *FPGA*.

7
Considerações finais

Esse projeto trouxe à tona a construção de uma plataforma interativa, demonstrando o potencial de implementação de diferentes jogos e aplicações utilizando uma linguagem de descrição de *Hardware*. Além disso, com a placa desenvolvida, tem-se um sistema físico que pode ser usado na integração de periféricos. Para o campo da retrocomputação, essa sistema físico, composto pelo *FPGA* e placa, é robusto o suficiente para que possa ser usado na implementação de arquiteturas clássicas utilizando códigos disponíveis na rede, apenas necessitando fazer alterações para que esses estejam aptos a serem executados nesse *Hardware*. Além disso, as explicações registradas nesse documento detalham a construção de cada parte do projeto, desde o *Hardware* até o digital, de forma a esclarecer dúvidas de interessados e entusiastas do assunto.

Durante o desenvolvimento desse projeto, foi necessário aprender sobre protocolos e interfaces conhecidos da indústria para que fosse possível construir a comunicação com os periféricos referentes a esses. Além disso, para que os jogos fossem desenvolvidos com um bom desempenho, precisou-se aprofundar consideravelmente os conhecimento práticos de computação digital. Por fim, para a confecção da PCB de suporte ao projeto teve-se que estudar detalhadamente a elaboração de diferentes estruturas de circuito, além do contato prático com o ciclo de desenvolvimento de um *Hardware*, desde a prototipação e depuração, até a montagem final.

7.1
Implementações Futuras

No aspecto físico do projeto, pode-se pontuar que a quantidade de cores disponíveis pelo sistema utilizado no projeto é bem limitada. Sendo assim, seria interessante a construção do módulo gráfico utilizando mais bits para cada canal de cor, dessa forma aumentado a possibilidade de detalhes disponíveis para a formação da imagem. Além disso, é relevante a reformulação do circuito de áudio, de forma que esse módulo consiga reproduzir mais de uma nota simultaneamente, e assim, possibilitando a execução de acordes por parte do usuário.

No aspecto digital, pode-se pontuar que os jogos apesar de apresentarem um bom funcionamento possuem aspectos que podem ser reavaliados. Sobre esses, seria relevante a reformulação do sistema de colisão da bola de forma a garantir que a bola em nenhuma circunstância atravesse a raquete. Sobre a interface de desenho, a implementação de um sistema com ainda mais opções de cores seria valiosa para a experiência do usuário que teria mais graus de liberdade para a elaboração dos desenhos.

Ademais, sobre o sintetizador, a implementação de diferentes formas de onda, como ondas triangulares ou dente de serra, seria interessante no aspecto musical, haja vista que cada forma de onda possui um timbre característico devido aos harmônicos que as compõem.

Por fim, com o sistema físico, já implementado, tem-se a possibilidade de executar códigos de sistemas de computação clássicos amplamente disponibilizados na Internet. Dessa forma, essa capacidade pode tornar o projeto como um ferramenta de resgate a tecnologias que já não mais comercializadas. Assim, a tarefa de adaptar tais códigos, para utilizá-los nessa plataforma, é plenamente possível e uma oportunidade de se ter contato com a retrocomputação através de *Hardware* reprogramável.

Referências

[1] M. R. Swaine and P. A. Freiberger, "Univac," Britannica, Disponível em: https://www.britannica.com/technology/UNIVAC. Acesso em: 25 jun. 2025.

[2] S. Cass, "Chip hall of fame: Intel 4004 microprocessor," IEEE, Disponível em: https://spectrum.ieee.org/chip-hall-of-fame-intel-4004-microprocessor. Acesso em: 25 jun. 2025.

[3] I. SPECTRUM, "Chip hall of fame: Xilinx xc2064 fpga," IEEE SPECTRUM, Disponível em: https://spectrum.ieee.org/chip-hall-of-fame-xilinx-xc2064-fpga. Acesso em: 25 jun. 2025.

[4] B. F. Andreeti, "Descrição VHDL e Prototipação em FPGA do Processador MOS 6502," Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) – Universidade Federal de Santa Maria, Santa Maria, Rio Grande do Sul, Brasil, 2015, Disponível em: https://repositorio.ufsm.br/bitstream/handle/1/25228/TCC%20Bernardo%20Favero%20Andreeti.pdf. Acesso em: 23 jun. 2025.

[5] E. Cruz, D. Adriano, E. Gaudino, and S. Júnior, *Sistemas Digitais Reconfiguráveis: FPGA e VHDL*, 1st ed. Rio de Janeiro, Rio de Janeiro, Brasil: Alta Books, 2022.

[6] R. HÜSEMANN, "APOSTILA DE VHDL," 2001, Disponível em: https://www.cin.ufpe.br/~sjbj/Arquivos/vhdl_basico.pdf. Acesso em: 25 jun. 2025.

[7] H. Foster, "Part 6: The 2022 Wilson Research Group Functional Verification Study," Siemens, 2022, Disponível em: https://blogs.sw.siemens.com/verificationhorizons/2022/11/21/part-6-the-2022-wilson-research-group-functional-verification-study/. Acesso em: 25 jun. 2025.

[8] ALTERA, "Manual do dispositivo Cyclone IV®," ALTERA, San Jose, California, EUA, 2016, Disponível em: https://www.intel.com.br/content/www/br/pt/content-details/653974/cyclone-iv-device-handbook.html. Acesso em: 23 jun. 2025.

[9] QMTECH, "Cyclone IV EP4CGX150GX Core Board User Manual," QMTECH, China, 2022, Disponível em: https://github.com/ChinaQMTECH/EP4CGX150DF27_CORE_BOARD/blob/main/QMTECH_CycloneIV_EP4CGX150GX_User_Manual(CoreBoard)-V01.pdf. Acesso em: 23 jun. 2025.

[10] IBM, "The PS/2," IBM, disponível em: https://www.ibm.com/history/ps-2. Acesso em: 24 jun. 2025.

[11] XILINX, "Spartan-3E FPGA Starter Kit Board User Guide, v1.2," XILINX, San Jose, California, EUA, 2011, Disponível em: https://docs.amd.com/v/u/en-US/ug230. Acesso em: 23 jun. 2025.

[12] M. WILSON, "What is VGA? A Comprehensive Guide to Video Graphics Array," HP, 2024, Disponível em: https://www.hp.com/us-en/shop/tech-takes/what-is-vga-comprehensive-guide-video-graphics-array. Acesso em: 23 jun. 2025.

[13] NextPCB, "VGA Connector Pinout - Basic Introduction is Here," NextPCB, Shenzhen, China, Disponível em: https://www.nextpcb.com/blog/vga-connector-pinout. Acesso em: 25 jun. 2025.

[14] Sun, "C - Connector Pinouts," Oracle, Santa Clara, California, EUA, 2010, Disponível em: https://docs.oracle.com/cd/E19201-01/821-0902-10/appC_Pinouts.html. Acesso em: 25 jun. 2025.

[15] P. P. Chu, *FPGA PROTOTYPING BY VHDL EXAMPLES*. New Jersey, EUA: John Wiley & Sons, 2008.

[16] S. LTD, "XGA Signal 1024 x 768 @ 60 Hz timing," SECONS LTD, Disponível em: http://www.tinyvga.com/vga-timing/1024x768@60Hz. Acesso em: 29 jun. 2025.

[17] A. KAY, "Decoupling capacitors," Texas Instruments, 2022, Disponível em: https://www.ti.com/content/dam/videos/external-videos/de-de/9/3816841626001/6313253251112.mp4/subassets/notes-decoupling_capacitors.pdf. Acesso em: 29 jun. 2025.

[18] S. Workman, "A Low-Cost, Single Coupling Capacitor Configuration for Stereo Headphone Amplifiers," Texas Instruments, 1999, Disponível em: https://www.ti.com/lit/an/sloa043/sloa043.pdf?ts=1751255506510. Acesso em: 29 jun. 2025.

[19] F. Iazzeta, "Tabela de Frequências, Períodos e Comprimentos de Onda," ECA-USP, Disponível em: https://iazzetta.eca.usp.br/tutor/acustica/introducao/tabela1.html. Acesso em: 1 jul. 2025.

[20] R. Carreira and P. Fonseca, " Teorema da Máxima Transferência de Potência," 1999, Disponível em: https://www.ufrgs.br/eng04030/Aulas/teoria/cap_06/maxtrpot.htm. Acesso em: 8 jul. 2025.

[21] S. Circuits, "Characteristic Impedance," Disponível em: https://www.protoexpress.com/kb/pcb-design-controlled-impedance/. Acesso em: 8 jul. 2025.

[22] Mathworks, "Square Wave from Sine Waves," The Mathworks, inc, Disponível em: https://www.mathworks.com/help/matlab/math/square-wave-from-sine-waves.html. Acesso em: 2 jul. 2025.

[23] A. Surfaces, "Exploring the Limits: What Frequency Can Humans Hear?" Acoustical Surfaces, Inc, 2024, Disponível em: https://www.acousticalsurfaces.com/blog/acoustics-education/what-frequency-can-humans-hear/. Acesso em: 3 jul. 2025.

[24] D. Wang, "VGA-Text-Generator," 2015, Disponível em: https://github.com/Derek-X-Wang/VGA-Text-Generator/. Acesso em: 3 jul. 2025.

[25] J. Schneider, "Field programmable gate arrays (FPGAs) vs. microcontrollers: What's the difference?" IBM, 2024, Disponível em: https://www.ibm.com/think/topics/fpga-vs-microcontroller. Acesso em: 23 jun. 2025.

[26] A. Tihon, "Experiments in VGA colors," 2024, disponível em: https://www.alrj.org/experiments-in-vga-colors.html. Acesso em: 25 jun. 2025.

[27] G. Santos, "Projeto_Final_2," 2025, Disponível em: https://github.com/MontSantos/Projeto-Final-2/tree/main. Acesso em: 5 jul. 2025.

A
Repositório Oficial do Projeto

Todos os datasheets utilizados, códigos desenvolvidos, simulações realizadas e placas construídas nesse projeto estão disponíveis em:
`https://github.com/MontSantos/Projeto_Final_2/tree/main`