UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA

BRENER GOMES DOS SANTOS

TIAGO SANTOS BELA

PROJETO
MÁQUINA DE VENDER SALGADOS

VITÓRIA DA CONQUISTA
2024

BRENER GOMES DOS SANTOS
TIAGO SANTOS BELA

PROJETO
MÁQUINA DE VENDER SALGADOS

Relatório apresentado para a disciplina de Circuitos Digitais do curso Ciência da Computação da Universidade Estadual do Sudoeste da Bahia.

Orientador: Marco Antonio Dantas Ramos

VITÓRIA DA CONQUISTA
2024

SUMÁRIO

1. INTRODUÇÃO

A disciplina de Circuitos Digitais é fundamental no estudo da Ciência da Computação, seu principal objetivo é estimular no aluno a capacidade de projetar e analisar sistemas que processam informações. Os Circuitos operam com sinais discretos, geralmente representados pelos estados binários 0 e 1, e são a base para diversas funcionalidades presentes no cotidiano como os computadores, IoT(Internet das Coisas), smartphones, inteligência artificial e equipamentos médicos.

Estes aparelhos digitais permitem a realização de operações complexas com alta eficiência, produtividade e precisão, o que é muito importante quando é necessário efetuar o processamento de dados em alta velocidade. Eles também são fundamentais no desenvolvimento de sistemas de controle automático, robótica, e automação industrial. O fato é que o estudo dos circuitos digitais é essencial para a formação de cientistas da computação, e proporciona ao discente uma base sólida para pesquisa e desenvolvimento em eletrônica, informática, operações lógicas(AND, OR, NOT), álgebra de boole, sistemas de numeração e circuitos combinacionais e circuitos Sequenciais que são importantes para a fundamentação dos conceitos computacionais teóricos.

2. OBJETIVO GERAL

Dentro deste contexto de aprendizado e ensino, o docente responsável por ministrar a disciplina, Marco Antonio Dantas Ramos, propôs aos seus alunos a implementação de um projeto de máquina de vender salgados com o intuito de reforçar o aprendizado e expandir os limites dos discentes em relação ao assunto. Foi definido que a máquina deveria possuir em seu funcionamento, lógica e tratamento de problemas relacionados a compra e venda dos salgados e o projeto deveria ser produzido na linguagem VHDL. O projeto contém em sua estrutura de código diversas funcionalidades que possibilitam uma eficiente máquina comercial, seguindo o padrão de projeto proposto pelo docente da disciplina.

3. VHDL

O **VHDL**(VHSIC Hardware Description Language) é uma linguagem de descrição de hardware, sua história começa na década de 1980, quando o Departamento de Defesa dos Estados Unidos teve a vontade de desenvolver e aprimorar circuitos integrados de alta velocidade. A necessidade era padronizar a documentação e facilitar a simulação de sistemas

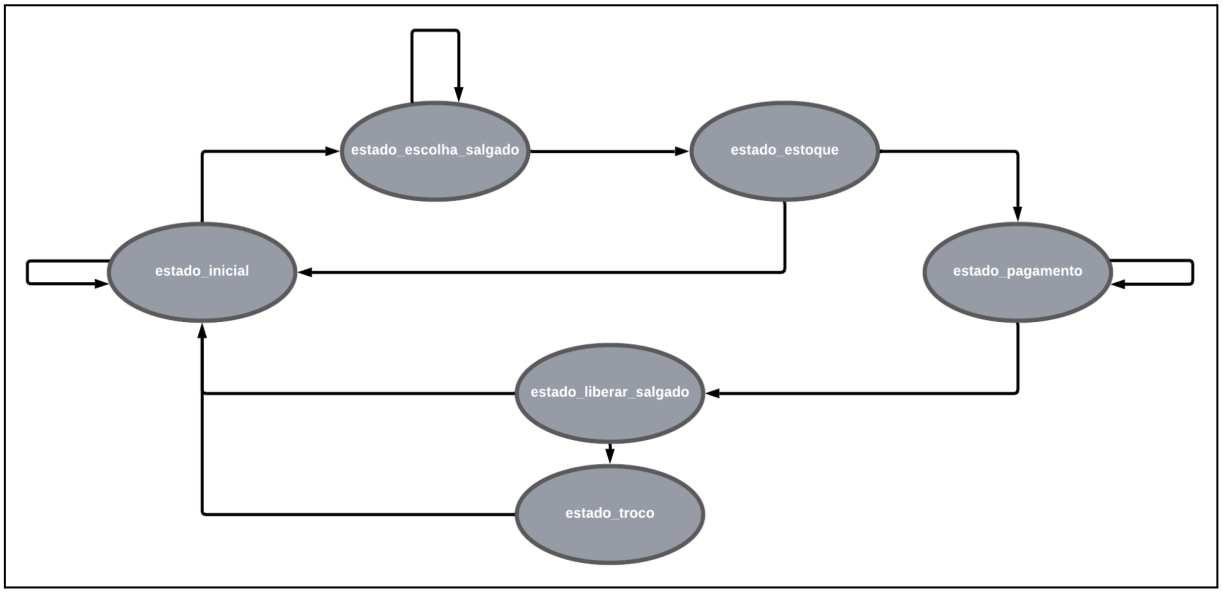
complexos. Hoje, após anos de experiência no mercado de tecnologia, o principal objetivo da linguagem é modelar e simular sistemas eletrônicos digitais permitindo a especificação precisa do comportamento e da estrutura dos sistemas digitais, desde simples circuitos lógicos até complexos sistemas integrados.

A sintaxe da linguagem é organizada em duas partes principais: a entidade e a arquitetura. A entidade define a interface do componente, incluindo a definição das portas de entrada e saída, a passo que a arquitetura descreve a implementação interna desse componente, basicamente é onde a lógica do circuito é implementada de fato.

Tudo isso é tem uma utilidade vasta, sendo amplamente usado no design de ASICs (Application-Specific Integrated Circuits) e FPGAs (Field-Programmable Gate Arrays). O **VHDL** também permite a descrição precisa e a simulação de circuitos antes da fabricação, além de ser uma ferramenta educacional fundamental para ensinar conceitos de design digital e arquitetura de computadores em universidades e institutos técnicos. Em resumo, o **VHDL** é uma linguagem poderosa e versátil, essencial no design e desenvolvimento de sistemas eletrônicos digitais modernos.

4. DIAGRAMA DE ESTADOS

Figura 1: Diagrama de estados da máquina



Fonte: Autoria própria.

O modelo de diagrama de estados é uma excelente forma de expressar o funcionamento de máquinas, pois torna o entendimento da lógica aplicada para a produção do

projeto mais simples e objetiva, além de ser também uma ótima forma de documentação de trabalhos e documentos. Neste projeto, os discentes se propuseram a gerar o diagrama com o objetivo de torná-lo um fonte de documentação geral para o trabalho efetuado, o mesmo será explicado de forma eficiente nos textos que se seguem.

O primeiro estado da máquina é o ***estado_inicial***, que é o estado em que a máquina se encontra assim que é ligada, ou reiniciada. A lógica que se sucede se porta da seguinte forma, caso o pino para iniciar a máquina seja ativado, a máquina vai para o próximo estado que é o de ***estado_escolha_salgado***, aqui o usuário deve decidir qual salgado ele quer comprar, caso o salgado escolhido não seja válido, a máquina permanece no estado de escolha, caso seja um valor válido, ela avança para o estado ***estado_estoque***, onde ela verifica se o há estoque para o salgado que foi escolhido pelo utilizador da máquina, caso não tenha estoque, a máquina volta para o estado inicial, caso tenha estoque a máquina vai para o estado de ***estado_pagamento***.

O estado de pagamento é onde o usuário deve fornecer o pagamento necessário para efetuar a compra do salgado selecionado, caso o valor inserido seja invalido, a máquina permanece no estado atual, caso o pagamento seja efetuado de forma correta ela avança para ***estado_liberar_salgado***, onde ela vai verificar se há troco para devolver para o cliente, caso não haja troco, a máquina volta para o estado inicial da máquina, caso haja troco, ela vai para o ***estado_troco*** onde ela vai calcular o troco a ser devolvido para o usuário, terminando assim o ciclo da máquina.

5. ENTIDADE

Neste módulo, a equipe se dispôs a definir os pinos de entradas e saídas do circuito. Essa entidade descreve todos os sinais de entrada, saída e buffers necessários para o funcionamento da máquina, incluindo controle de estados, LEDs indicativos, confirmação de escolha de salgado, pagamento liberação de troco, inserção de moeda e exibição em displays de 7 segmentos.

5.1. Pinos de entrada(IN):

- **clk**: Clock geral da aplicação.
- **rst**: Sinal de reset da máquina.
- **ligar_maquina**: sinal que inicia a máquina.
- **continuar**: Sinal para continuar o percurso da máquina.
- **salgado_escolhido**: Vetor que especifica o tipo de salgado escolhido.
- **liberar_salgado**: Sinal para liberar o salgado para o cliente.

- **confirmar_salgado**: Sinal para confirmar o salgado escolhido.
- **confirmar_moeda**: Sinal para confirmar a inserção de moeda.
- **moedas**: Vetor com os tipos de moedas aceitas.

5.2. Pinos de saída (OUT):

- **estados_atual_maquina**: Estado atual da máquina em formato binário.
- **salgado_invalido_led**: LED que indica salgado escolhido inválido.
- **salgado_terminado_led**: LED que indica que o salgado escolhido está sem estoque.
- **salgado_liberado_cliente_led**: LED que indica que o salgado foi liberado para o cliente.
- **moeda_invalida_led**: LED que indica moeda inserida inválida.
- **moeda_liberada_cliente_led**: LED que indica que a moeda foi liberada para o cliente.
- **estado_inicial_led**: LED do estado inicial da máquina.
- **estado_escolha_salgado_led**: LED do estado de escolha do salgado.
- **estado_estoque_led**: LED do estado de verificação de estoque.
- **estado_pagamento_led**: LED do estado de pagamento.
- **estado_liberar_salgado_led**: LED do estado de liberação do salgado.
- **estado_troco_led**: LED do estado de devolução de troco.

5.3. Buffers:

- **valor_salgado**: Valor do salgado escolhido (variável de saída que pode ser lida e escrita).

Displays de 7 Segmentos (OUT):

- **display7_salgado**: Saída para o display do tipo de salgado.
- **display7_quantia_centena**: Saída para o display da centena da quantia inserida.
- **display7_quantia_dezena**: Saída para o display da dezena da quantia inserida.
- **display7_quantia_unidade**: Saída para o display da unidade da quantia inserida.

6. ARCHITECTURE

Neste módulo, a equipe se dispôs a definir a lógica empregada no circuito.

FUNÇÃO '*mostrarDisplay7*':

É usada para converter um número inteiro (de 0 a 14) em um vetor de 7 bits que representa os segmentos de um display de 7 segmentos. Cada segmento do display pode ser aceso ou apagado para mostrar diferentes caracteres.

1. A função recebe um número inteiro (‘*numeroEscolhido*’) e retorna um vetor de 7 bits do tipo do tipo ‘*STD_LOGIC_VECTOR*’;

2. A variável ‘*saidaDisplay*’ é declarada para armazenar o valor que será retornado pela função;

3. O bloco ‘CASE’ é utilizado para verificar o valor de ‘*numeroEscolhido*’ e atribuir o valor correspondente a ‘*saidaDisplay*’;

4. Cada ‘*WHEN*’ define a correspondência entre o valor de ‘*numeroEscolhido*’ e o vetor de 7 bits que representa o estado dos segmentos do display;
    - Cada vetor de bits define quais segmentos do display serão acesos (‘*0*’) ou apagados (‘*1*’).

5. Por fim, a função retorna o vetor de bits ‘*saidaDisplay*’ que representa o número no display de 7 segmentos.

Ex. Se ‘*numeroEscolhido*’ for ‘*3*’, a função retornará ‘*0110000*’, que acende os segmentos necessários para formar o número '*3*' no display de 7 segmentos.

TROCA_DE_ESTADO (processo sensível ao clock):

É usado para descrever circuitos sequenciais, onde o estado interno do circuito é atualizado em sincronia com o clock.

1. O processo será executado toda vez que houver uma mudança no sinal ‘*clk*’;
2. Dentro do processo, a função ‘*rising_edge(clk)*’ verifica se há uma borda de subida no sinal de clock. Esta função é essencial para implementar um circuito sequencial que responde apenas às transições específicas do clock.
3. Quando uma borda de subida é detectada, o valor de ‘*proximo_estado*’ é atribuído ao ‘*estado_atual*’. Isso representa a transição do estado atual para o próximo estado, em sincronia com o clock.

EXIBIR_FUNCOES (processo sensível ao clock):

Faz o controle de exibição utilizando o '*estado_atual*' para determinar quais informações exibir nos displays de 7 segmentos, dependendo do estágio do processo de venda. As informações são atualizadas em sincronia com o sinal de clock, garantindo que as transições de estado e atualizações de display ocorram de maneira ordenada e previsível.

1. O processo é declarado com '*clk*' na lista de sensibilidade. A variável '*numero_salgado*' é usada dentro do processo para armazenar o número do salgado selecionado;
2. Verifica se há uma borda de subida no sinal de clock usando a expressão (*clk'event AND clk = '1'*);
3. O bloco '*CASE*' seleciona ações diferentes com base no valor de '*estado_atual*'. Cada estado representa uma fase diferente da operação da máquina de salgados.

Descrição dos estados:

'*estado_inicial*' – todos os displays de 7 segmentos são desligados, indicando que nenhum item ou valor está sendo mostrado.

'*esta*do_escolha_salgado' – com base na escolha do salgado '*salgado*_escolhido', os displays são atualizados para mostrar o tipo de salgado e seu preço. A função '*mostrarDisplay7*' é usada para converter números em representações de 7 segmentos.

'*estado_estoque*' – o display mostra o número do salgado selecionado e a quantidade inserida pelo usuário. A quantidade inserida é dividida em centenas, dezenas e unidades para ser exibida nos displays de 7 segmentos.

'*estado_pagamento*' – similar ao estado de estoque, mas representando uma fase específica de pagamento, onde os displays mostram o número do salgado e a quantia inserida.

'*estado_liberar_salgado*' – os displays mostram o número do salgado e o troco que será devolvido ao cliente.

'*estado_troco*' – mostra o troco do cliente.

ADICAO_DE_MOEDAS (processo sensível ao clock e a outros pinos da máquina):

Controla a adição de moedas pelo cliente em uma máquina de vendas. Ele é sensível ao sinal de clock '*clk*', às entradas de moedas '*moedas*', e ao sinal de confirmação de moeda '*confirmar_moeda*'. Este processo gerencia a quantia inserida pelo cliente e aciona um LED de alerta para moedas inválidas.

1. O processo é declarado com '*clk*', '*moedas*' e '*confirmar_moeda*' na lista de sensibilidade. Isso significa que o processo será executado sempre que houver uma mudança em qualquer um desses sinais;

2. Se o sinal de clock estiver em nível alto (*'1'*) e o estado atual for o '*estado_inicial*', o LED de moeda inválida '*moeda_invalida_led*' é desligado (*'0'*) e a '*quantia_inserida*' é resetada para 0. Isso inicializa o sistema para um novo ciclo de inserção de moedas;

3. Caso não esteja no estado inicial, o código verifica se houve uma transição na entrada '*confirmar_moeda*' e se o estado atual é o estado de pagamento '*estado_pagamento*';

4. Dentro deste bloco, há um '*CASE*' que verifica o valor de cada '*moeda*':
    - **"001"** – representa uma moeda de R$0,25. O LED de moeda inválida é desligado e a quantia inserida é incrementada.
    - **"101"** – representa uma moeda de R$0,50. O LED de moeda inválida é desligado e a quantia inserida é incrementada.
    - **"001"** – representa uma moeda de R$1,00. O LED de moeda inválida é desligado e a quantia inserida é incrementada.
    - '*others*' – o LED de moeda inválido é ligado, sinalizando uma moeda inválida.

LOGICA_DE_ESTADOS (processo sensível ao clock e ao reset):

É responsável por controlar a lógica de estados da máquina de salgados. Ele é sensível ao sinal de '*clk*' e ao '*rst*'.

1. O processo é declarado com '*clk*' e '*rst*' na lista de sensibilidade, o que significa que o processo será executado sempre que houver uma mudança em qualquer um desses sinais.

2. Quando '*rst*' é igual a '1', todos os estados e LEDs são inicializados. Os LEDs são

desligados (exceto o '*estado_inicial_led*'), o troco é zerado, e os estoques são definidos com valores iniciais, e o próximo estado é definido como '*estado_inicial*'.

3. Se o '*rst*' não estiver ativo, o processo verifica o estado atual e realiza ações específicas baseadas no estado.

Descrição dos estados:

'*estado_inicial*' – define o estado inicial da máquina e liga o LED correspondente. Se '*ligar_maquina*' é verdadeiro, avança para o estado de escolha do salgado; caso contrário, permanece no estado inicial.

'*estado_escolha_salgado*' – permite ao usuário escolher o tipo de salgado e verifica se o salgado escolhido é válido. Se confirmado, avança para o '*estado_estoque*'; se não, permanece no '*estado_escolha_salgado*'.

'*estado_estoque*' – verifica se há estoque disponível para o salgado escolhido. Se houver estoque, avança para o '*estado_pagamento*'; caso contrário, retorna para '*estado_escolha_salgado*'.

'*estado_pagamento*' – realiza o processo de pagamento. Se a quantia inserida é suficiente, avança para '*estado_liberar_salgado*'; se não, verifica se precisa devolver o troco.

'estado_liberar_salgado' – libera o salgado escolhido para o cliente após o pagamento, deduz o salgado do estoque e calcula o troco, se necessário. Retorna ao '*estado_inicial*' se não houver troco; caso contrário, avança para o estado de devolução de troco.

'*estado_troco*' – devolve o troco para o cliente, se necessário. Retorna ao '*estado_inicial*' após a operação de troco, a menos que o cliente deseje continuar.

7. PINAGEM

A seção da pinagem é de extrema importância, pois é através dela que é feita a ligação direta com a placa Fpga, esta inclusive é a última seção do projeto, uma vez que terminada a sessão da pinagem basta apenas realizar os testes com o circuito integrado. Segue a relação da seleção dos pinos e LEDS dos componentes na placa FPGA Cyclone II modelo EP2C20F484C7, abaixo:

7.1. LEDS verdes

Figura 2.1: Tabela de LEDS verdes

Nome do Sinal	Pino FPGA	Descrição
LEDG[0]	PIN_U22	estado_inicial_led
LEDG[1]	PIN_U21	estado_escolha_salgado_led
LEDG[2]	PIN_V22	estado_estoque_led
LEDG[3]	PIN_V21	estado_pagamento_led
LEDG[4]	PIN_W22	estado_liberar_salgado_led
LEDG[5]	PIN_W21	estado_troco_led

7.2. LEDS vermelhos

Figura 2.2: Tabela de LEDS vermelhos

Nome do Sinal	Pino FPGA	Descrição
LEDR[0]	PIN_R20	salgado_invalido_led
LEDR[1]	PIN_R19	salgado_terminado_led
LEDR[2]	PIN_U19	moeda_invalida_led

7.3. Switches

Figura 2.3: Tabela de Switches

Nome do Sinal	Pino FPGA	Descrição
SW[0]	PIN_L22	ligar_maquina
SW[1]	PIN_L21	salgado escolhido[0]
SW[2]	PIN_M22	salgado escolhido[1]
SW[3]	PIN_V12	salgado escolhido[2]
SW[4]	PIN_W12	moedas[0]
SW[5]	PIN_U12	moedas[1]
SW[6]	PIN_U11	moedas[2]
SW[7]	PIN_M2	confirmar_moeda
SW[8]	PIN_M1	liberar_salgado
SW[9]	PIN_L2	continuar

7.4. Botões

Figura 2.4: Tabela de Botões

Nome do Sinal	Pino FPGA	Descrição
KEY[0]	PIN_R22	confirmar_salgado
KEY[1]	PIN_R21	confirmar_moeda
CLOCK_27	PIN_D12	clk

7.5. Display de 7 segmentos

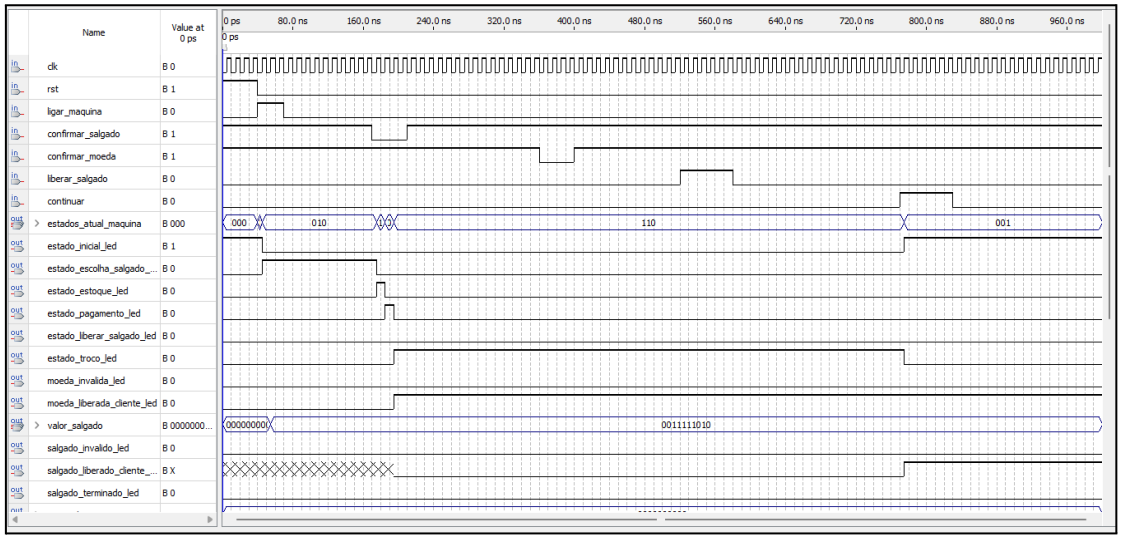
Figura 2.4: Tabela de Displays

Nome do Sinal	Pino FPGA	Descrição
HEX0[0]	PIN_J2	display7_quantia_unidade[0]
HEX0[1]	PIN_J1	display7_quantia_unidade[1]
HEX0[2]	PIN_H2	display7_quantia_unidade[2]
HEX0[3]	PIN_H1	display7_quantia_unidade[3]
HEX0[4]	PIN_F2	display7_quantia_unidade[4]
HEX0[5]	PIN_F1	display7_quantia_unidade[5]
HEX0[6]	PIN_E2	display7_quantia_unidade[6]
HEX1[0]	PIN_E1	display7_quantia_dezena[1]
HEX1[1]	PIN_H6	display7_quantia_dezena[2]
HEX1[2]	PIN_H5	display7_quantia_dezena[2]
HEX1[3]	PIN_H4	display7_quantia_dezena[3]
HEX1[4]	PIN_G3	display7_quantia_dezena[4]
HEX1[5]	PIN_D2	display7_quantia_dezena[5]
HEX1[6]	PIN_D1	display7_quantia_dezena[6]
HEX2[0]	PIN_G5	display7_quantia_centena[0]
HEX2[1]	PIN_G6	display7_quantia_centena[1]
HEX2[2]	PIN_C2	display7_quantia_centena[2]
HEX2[3]	PIN_C1	display7_quantia_centena[3]
HEX2[4]	PIN_E3	display7_quantia_centena[4]
HEX2[5]	PIN_E4	display7_quantia_centena[5]
HEX2[6]	PIN_D3	display7_quantia_centena[6]
HEX3[0]	PIN_F4	display7_salgado[0]
HEX3[1]	PIN_D5	display7_salgado[1]
HEX3[2]	PIN_D6	display7_salgado[2]
HEX3[3]	PIN_J4	display7_salgado[3]
HEX3[4]	PIN_L8	display7_salgado[4]
HEX3[5]	PIN_F3	display7_salgado[5]
HEX3[6]	PIN_D4	display7_salgado[6]

8. FORMAS DE ONDA

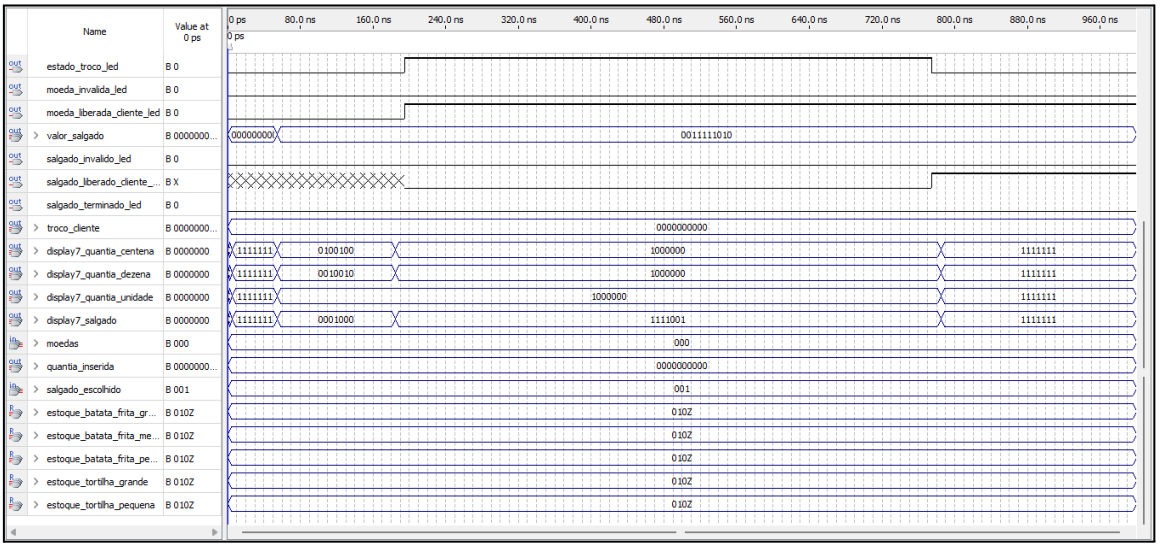
Simular a execução do projeto verificando o formato de onda relativo aos pinos antes de implementá-lo de fato na placa do Fpga é uma etapa muito importante para o processo de construção de trabalhos na área de circuitos digitais. É através deste determinado tipo de teste que o projetista vai analisar os possíveis erros lógicos presentes na construção do circuito, prevenindo dessa forma acontecimentos inoportunos. A simulação de onda do projeto pode ser visualizada abaixo:

Figura 3.1: Forma de onda do projeto



Fonte: Autoria própria.

Figura 3.2: Formas de onda do projeto



Fonte: Autoria própria.

9. CONCLUSÃO

Para concluir o relatório do projeto, é essencial recapitular os principais pontos vivenciados pelos discentes nessa jornada em circuitos digitais. Em primeira instância, os alunos fizeram uma análise detalhada dos requisitos do projeto e de todas as instruções que foram requisitados pelo docente. Após isso, elaboraram uma especificação clara e precisa de como seria feita a implementação do trabalho. Tendo todos esses dados, os alunos, enfim, partiram para a implementação do exercício solicitado.

Durante o processo de desenvolvimento, os alunos enfrentaram desafios que foram superados com soluções que aumentaram seu nível técnico de computação, e sobretudo de sistemas digitais. Este projeto não apenas consolidou os conhecimentos teóricos, mas também capacitou os discentes a enfrentar desafios mais complexos que podem vir a surgir posteriormente.

Assim, este relatório é encerrado com a satisfação do desafio ser cumprido com os objetivos e a expectativa de que este trabalho contribua positivamente para estudos futuros e aplicações práticas na área de **VHDL** e circuitos digitais.

REFERÊNCIAS

USP. Apostila de Introdução a VHDL. Universidade de São Paulo, 2014. Disponível em:<https://edisciplinas.usp.br/pluginfile.php/530833/mod_resource/content/1/Apostila%20de%20Introdu%C3%A7%C3%A3o%20a%20VHDL_2014.pd>. Acesso em: 27 jun. 2024.

YOUTUBE. Eletrônica Digital II, 2021. Disponível em: <https://www.youtube.com/playlist?list=PLI-A7AA2TwhXPTrClHIa4EG4zz0kkAYrD>. Acesso em: 27 jun. 2024.

YOUTUBE. Curso de FPGAs (VHDL). Disponível em: <https://www.youtube.com/playlist?list=PLZ8dBTV2_5HS79fVexGTtCMDUp7kjnumS>. Acesso em: 28 jun. 2024.