UESB

Universidade Estadual do Sudoeste da Bahia
Bacharelado em Ciência da Computação

Andreina Melo, Christian Rocha e Thiago Sousa

Projeto Para a Matéria de Circuitos Digitais

Máquina de Venda de Salgados

Versão 1.0

Vitória da Conquista - Dezembro 2023

Guia de elaboração de propostas de projetos

Ficha Catalográfica

BIREME / OPAS / OMS (Brasil)

Guia de elaboração de propostas de projetos. BIREME / OPAS / OMS. São Paulo : BIREME / OPAS / OMS, Abril 2008.

39 p.

1. Manual do usuário. 2. Acesso à informação. 3. Sistemas de informação. 4. Gerenciamento de informação. 5. Saúde Pública. 6. Serviços de saúde. I. BIREME II. Título

Advertência - A menção a companhias e/ou instituições específicas ou a certos produtos não implica que estes sejam apoiados ou recomendados por BIREME / OPAS / OMS, e não significa que haja preferência em relação a outros de natureza similar, citados ou não.

BIREME / OPAS / OMS

Centro Latino-Americano e do Caribe de Informação em Ciências da Saúde

Rua Botucatu 862 V Clementino

Este documento foi produzido com a Metodologia para Normalização de Documentos (NorDoc) desenvolvida pela BIREM.

Sumário

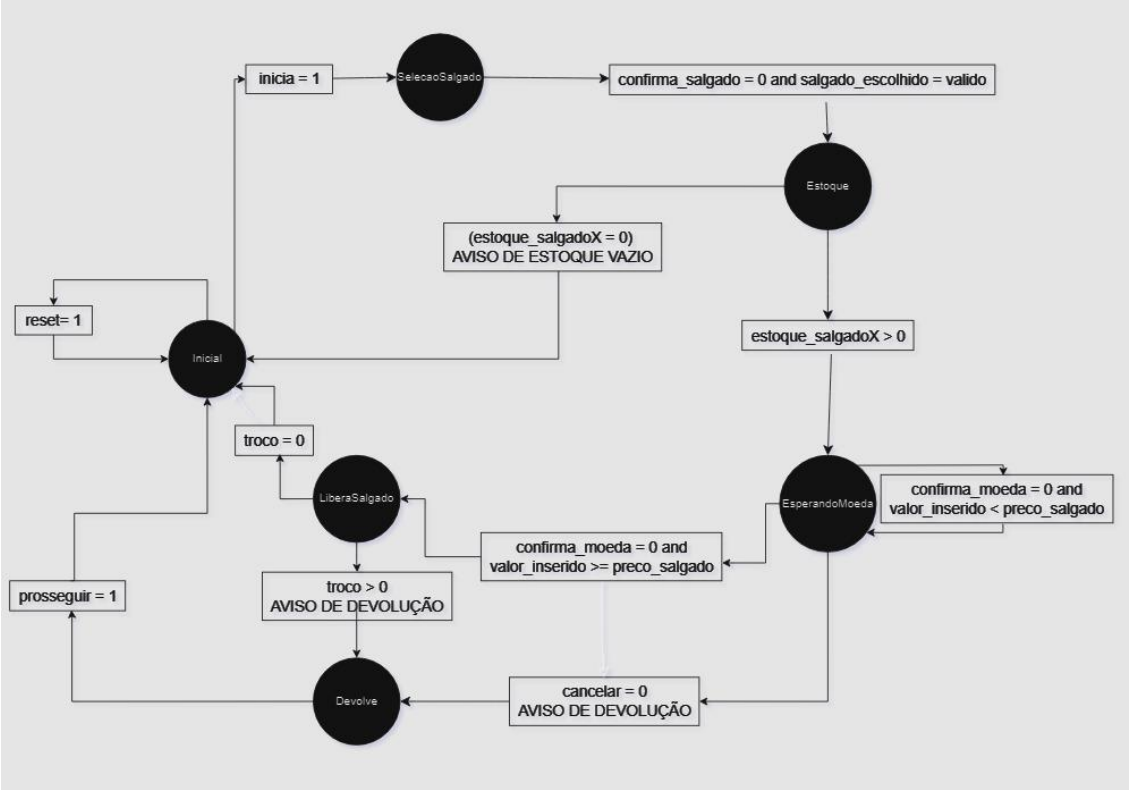
1 Introdução

Desenvolvido por:

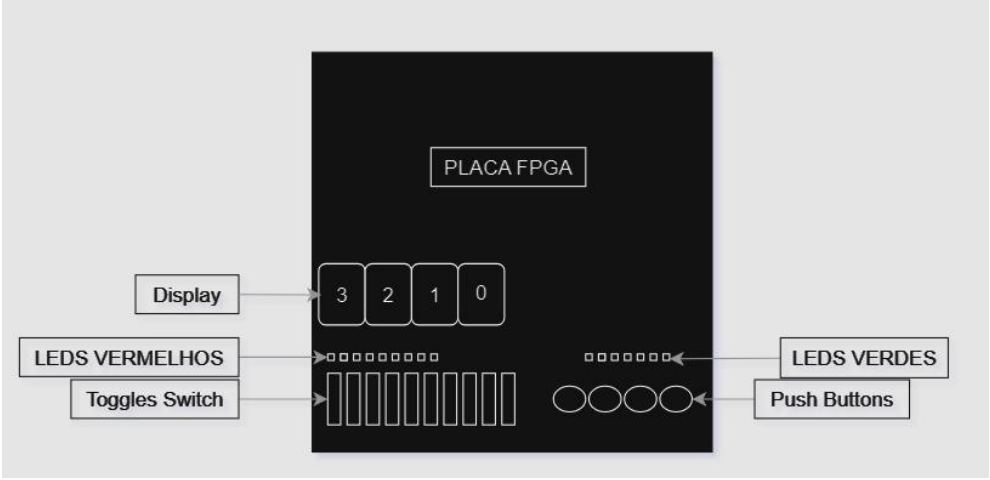
- Andreina Novaes Silva Melo
- Christian Schettine Paiva Rocha
- Thiago Fernandes Pereira de Sousa

O projeto desenvolvido pela equipe, de uma Máquina de Vender Salgados proposta pelo professor Marco Antônio para avaliação e culminância da matéria de Circuitos Digitais, é definido em todo o seu funcionamento, lógica e tratamento de problemas em um único arquivo VHDL. Nesse projeto foram definidos entidade e arquitetura, funções usadas pelo sistema para diversas operações e configuração de pins utilizados para o funcionamento numa placa FPGA. Seu funcionamento se baseia na transição dessa entidade definida entre os estados da máquina, onde cada uma faz cálculos e informa valores diferentes nos displays da placa, referentes à sua lógica interna e úteis ao usuário. Maior parte da lógica está construída no processo da máquina de estados e, além dela, há uma função e um processo para o display.

2 Diagrama de Estados



O Diagrama de Estados serve para ilustrar a lógica interna dos estados da máquina, mostrando as possíveis transições entre eles e as operações lógicas/aritméticas feitas nestas transições. Por meio dessa visualização também é possível observar o funcionamento que se espera da Máquina de Vender Salgados.



3 Entity MaquinaSalgados

Na entidade vai ser onde definimos a interface interna e nossas portas de entrada ":in std_logic" e de saída ":out std_logic" da máquina (tudo isso dentro do "port();"). Nela também estão todos os buffers, controlando a parte financeira.

```
entity MaquinaSalgados is
port(
   -- entradas
   clock: in std_logic; -- clock essencial para a maquina
   reset: in std_logic; -- resetar a maquina
   inicia: in std_logic; -- para iniciar a maquina
   cancelar: in std_logic; -- cancelar compra
   salgado_escolhido: in std_logic_vector(2 downto 0); -- escolher um dos 5 salgados: 001, 010, 011, 100, 101
   confirma_salgado: in std_logic; -- confirmacao de salgado selecionado
   confirma_moeda: in std_logic; -- confirmacao de moeda
   moeda: in std_logic_vector(1 downto 0); -- as moedas podem ser de tres tipos: 01 p/ R$0,25, 10 p/ R$0,50 e 11 p/ R$1,00
   liberar: in std_logic; -- liberar salgado
   prosseguir: in std_logic; -- prosseguir depois da devolucao
   -- saidas
   sem_estoque: out std_logic; -- aviso de estoque vazio
   moeda_nao_permitida: out std_logic; -- aviso de moeda nao permitida
   salgado_invalido: out std_logic; -- aviso de salgadi invalido
   devolvido: out std_logic; -- aviso de dinheiro devolvido
   salgado_liberado: out std_logic; -- aviso de salgado liberado
   display_salgado: out std_logic_vector(6 downto 0); -- display para informar qual salgado foi escolhido
   display_moeda_centena: out std_logic_vector(6 downto 0); -- display para informar valor
   display_moeda_dezena: out std_logic_vector(6 downto 0); -- display para informar valor
   display_moeda_unidade: out std_logic_vector(6 downto 0); -- display para informar valor
   estado: out std_logic_vector(2 downto 0); -- estado atual
   estado_inicial: out std_logic; -- saida para o led
   estado_selecao_salgado: out std_logic; -- saida para o led
   estado_estoque: out std_logic; -- saida para o led
   estado_esperando_moeda: out std_logic; -- saida para o led
   estado_libera_salgado: out std_logic; -- saida para o led
   estado_devolve: out std_logic; -- saida para o led

   -- controle da parte financeira
   valor_inserido: buffer integer range 0 to 999 := 000; -- para somar as moedas inseridas
   preco_salgado: buffer integer range 0 to 999 := 000; -- buffer pode ser lido e escrito, contera o valor do salgado escolhido
   troco: buffer integer range 0 to 999 := 000; -- troco do cliente
   devolucao: buffer integer range 0 to 999 := 000 -- dinheiro devolvido
);
end MaquinaSalgados;
```

4 Architecture

Na arquitetura vão ser definidos o comportamento e funcionalidade do circuito, que são a lógica da nossa máquina.

Inicialmente é definido o "type", onde são definidos tipos enumerados de algo; neste caso, Estados.

Logo após, são definidos "signals" do tipo estado (atual e próximo) e inteiro (estoques dos salgados).

```
-- descreve o comportamento e a funcionalidade do circuito digital
architecture hardware of MaquinaSalgados is
   -- estados e sinais relacionados
   type estados is (Inicial, SelecaoSalgado, Estoque, EsperandoMoeda, LiberaSalgado, Devolve);
   signal estado_atual: estados;
   signal proximo_estado: estados;
   -- controle de estoque de salgados
   signal estoque_salgado1: integer range 0 to 10 := 0010; --Estoque do Salgado 1
   signal estoque_salgado2: integer range 0 to 10 := 0010; --Estoque do Salgado 2
   signal estoque_salgado3: integer range 0 to 10 := 0010; --Estoque do Salgado 3
   signal estoque_salgado4: integer range 0 to 10 := 0010; --Estoque do Salgado 4
   signal estoque_salgado5: integer range 0 to 10 := 0010;  --Estoque do Salgado 5
```

5 Function Display7

Função utilizada para converter valores inteiros passados como parâmetros para serem representados no display da placa.

```
-- funcao geral de display de 7 segmentos (altera valores inteiros para serem representados no display)
function display7 (inteiro: integer) return std_logic_vector is
   variable saida: std_logic_vector(6 downto 0);
   begin
      case(inteiro)is
         when 0 => saida := "1000000";
         when 1 => saida := "1111001";
         when 2 => saida := "0100100";
         when 3 => saida := "0110000";
         when 4 => saida := "0011001";
         when 5 => saida := "0010010";
         when 6 => saida := "0000010";
         when 7 => saida := "1111000";
         when 8 => saida := "0000000";
         when 9 => saida := "0010000";
         when others =>
      end case;
   return saida;
end display7;
```

6 Process Clock

É um processo padrão que observa quando ocorre um evento no clock e se evento é o clock = 1. Caso ocorra, atualiza o estado.

```
begin
    -- processo do clock | padrao
    process(clock, proximo_estado)
    begin
        if (clock'event and clock = '1') then
            estado_atual <= proximo_estado;
        end if;
    end process;
```

7 Somador

Serve para fazer as adições de moedas. Se o botão de confirmar moeda for pressionado, e o estado atual é Esperando Moeda, é adicionado o valor informado pelo switch; se não houver valor de moeda informado, a invalidade é informada e a máquina continua neste estado.

```
-- processo para adicionar moeda
somador: process (clock, moeda, confirma_moeda)
begin
   if(clock = '1' and estado_atual = Inicial)then
      moeda_nao_permitida <= '0';
      valor_inserido <= 000;
   elsif((confirma_moeda'event and confirma_moeda = '0') and estado_atual = EsperandoMoeda) then
      case(moeda) is
         when "01" =>
            moeda_nao_permitida <= '0';
            valor_inserido <= valor_inserido + 025;
         when "10" =>
            moeda_nao_permitida <= '0';
            valor_inserido <= valor_inserido + 050;
         when "11" =>
            moeda_nao_permitida <= '0';
            valor_inserido <= valor_inserido + 100;
         when others =>
            moeda_nao_permitida <= '1';
      end case;
   end if;
end process;
```

8 Máquina de Estados

Onde ocorre a lógica dentro dos estados, (aguarda sinal do usuário, dá algum sinal para o usuário e altera valores) transitando entre os estados dependendo da lógica.

Recebe todas as entradas como parâmetro.

Começa analisando se o valor do reset é igual a 1. Se for, ele reinicia a máquina, resetando valores e estoques, além de definir o estado Inicial como o próximo até que o reset seja igual a 0; quando o valor do reset for 0, ele entra num else e depois num case que, dependendo do estado atual, entra em outro estado.

```
-- toda a logica dos estados
maquina_estados: process(clock, reset, inicia,
estado_atual, confirma_salgado, salgado_escolhido,
estoque_salgado1, estoque_salgado2, estoque_salgado3,
estoque_salgado4, estoque_salgado5, cancelar,
valor_inserido, confirma_moeda, moeda, preco_salgado, troco, prosseguir)

begin
   if (reset = '1') then -- reiniciar a maquina
      estado <= "000";
      preco_salgado <= 000;
      troco <= 000;
      devolucao <= 000;
      salgado_invalido <= '0';
      sem_estoque <= '0';
      salgado_liberado <= '0';
      devolvido <= '0';
      estoque_salgado1 <= 0010;
      estoque_salgado2 <= 0010;
      estoque_salgado3 <= 0010;
      estoque_salgado4 <= 0010;
      estoque_salgado5 <= 0000;
      proximo_estado <= Inicial;
```

8.1 Estado Inicial

Seta todos os valores necessários, como a parte financeira, todos os sinais inicializados como 0, e aguarda o sinal do usuário atráves da alteração do valor de *inicia*; quando essa sinalização for feita, atualiza o estado e vai para Seleção de Salgados.

```
else
   case (estado_atual) is
      when Inicial => --comecar a maquina
         estado <= "001";
         -- definindo led do estado
         estado_inicial <= '1';
         estado_selecao_salgado <= '0';
         estado_estoque <= '0';
         estado_esperando_moeda <= '0';
         estado_libera_salgado <= '0';
         estado_devolve <= '0';
         -- definindo valores de saida e sinais para o inicial
         preco_salgado <= 000;
         troco <= 000;
         devolucao <= 000;
         salgado_liberado <= '0';
         devolvido <= '0';

         if(inicia = '1') then
            salgado_invalido <= '0';
            sem_estoque <= '0';
            proximo_estado <= SelecaoSalgado;
         else
            proximo_estado <= Inicial;
         end if;
```

8.2 Seleção de Salgado

Baseado em estruturas condicionais (if/elsif), analisa se o botão do salgado foi apertado e se o salgado escolhido é válido. Se o botão tiver sido apertado, mas o salgado informado não for válido, é dado sinal de inválido quando o botão for apertado e a máquina segue neste estado; mas, se o botão tiver sido apertado e o salgado for válido, por meio de uma estrutura if o preço do salgado é verificado e se há estoque desse salgado.

```
when SelecaoSalgado => -- logica de escolha entre as 5 opcoes de salgado e definicao do preco
   estado <= "010";
   -- definindo led do estado
   estado_inicial <= '0';
   estado_selecao_salgado <= '1';
   estado_estoque <= '0';
   estado_esperando_moeda <= '0';
   estado_libera_salgado <= '0';
   estado_devolve <= '0';

   preco_salgado <= 000;
   if (confirma_salgado = '0') then
      case (salgado_escolhido) is
         when "001" =>
            preco_salgado <= 250;
            proximo_estado <= Estoque;

         when "010" =>
            preco_salgado <= 150;
            proximo_estado <= Estoque;

         when "011" =>
            preco_salgado <= 075;
            proximo_estado <= Estoque;

         when "100" =>
            preco_salgado <= 350;
            proximo_estado <= Estoque;

         when "101" =>
            preco_salgado <= 200;
            proximo_estado <= Estoque;

         when others =>
            salgado_invalido <= '1'; -- led vermelho de salgado_invalido
            proximo_estado <= SelecaoSalgado;
      end case;
   else
      salgado_invalido <= '0';
      sem_estoque <= '0';
      proximo_estado <= SelecaoSalgado;
   end if;
```

8.3 Estoque

Onde é feita a verificação se há estoque do salgado selecionado. Se houver, desliga o sinal “Sem Estoque” e troca o estado da máquina para Esperando Moeda; se não, dá o sinal “Sem Estoque” e retorna para o Estado Inicial.

```
when Estoque => -- analisa estoque
   estado <= "011";
   -- definindo led do estado
   estado_inicial <= '0';
   estado_selecao_salgado <= '0';
   estado_estoque <= '1';
   estado_esperando_moeda <= '0';
   estado_libera_salgado <= '0';
   estado_devolve <= '0';

   case (salgado_escolhido) is
      when "001" =>
         if(estoque_salgado1 > 0) then -- se tiver estoque do salgado 1
            sem_estoque <= '0';
            proximo_estado <= EsperandoMoeda;
         else -- se nao tiver
            sem_estoque <= '1'; -- avisa que nao tem
            proximo_estado <= Inicial; -- volta para comeco
         end if;
      when "010" =>
         if(estoque_salgado2 > 0) then -- se tiver estoque do salgado 2
            sem_estoque <= '0';
            proximo_estado <= EsperandoMoeda;
         else -- se nao tiver
            sem_estoque <= '1'; -- avisa que nao tem
            proximo_estado <= Inicial; -- volta para comeco
         end if;
      when "011" =>
         if(estoque_salgado3 > 0) then -- se tiver estoque do salgado 3
            sem_estoque <= '0';
            proximo_estado <= EsperandoMoeda;
         else -- se nao tiver
            sem_estoque <= '1'; -- avisa que nao tem
            proximo_estado <= Inicial; -- volta para comeco
         end if;
      when "100" =>
         if(estoque_salgado4 > 0) then -- se tiver estoque do salgado 4
            sem_estoque <= '0';
            proximo_estado <= EsperandoMoeda;
         else -- se nao tiver
            sem_estoque <= '1'; -- avisa que nao tem
            proximo_estado <= Inicial; -- volta para comeco
         end if;
      when "101" =>
         if(estoque_salgado5 > 0) then -- se tiver estoque do salgado 5
            sem_estoque <= '0';
            proximo_estado <= EsperandoMoeda;


         else -- se nao tiver
            sem_estoque <= '1'; -- avisa que nao tem
            proximo_estado <= Inicial; -- volta para comeco
         end if;
      when others =>
   end case;
```

8.4 Esperando Moeda

Onde é feito ou o cancelamento da compra ou o avanço da máquina para o estado Libera Salgado. Feito o cancelamento, é feita a verificação do valor a ser devolvido e se ele existe; se existir, vai para o estado Devolve, e se não houver, vai para o Estado Inicial. Se não for feito o cancelamento, verifica-se

o valor inserido e se ele é maior ou igual ao preço do salgado; se for, vai para o estado Libera Salgado, se não for, permanece no estado Esperando Moeda.

```
when EsperandoMoeda => -- espera moedas ate que o valor atinja o preco do salgado ou cancela pedido
   estado <= "100";
   -- definindo led do estado
   estado_inicial <= '0';
   estado_selecao_salgado <= '0';
   estado_estoque <= '0';
   estado_esperando_moeda <= '1';
   estado_libera_salgado <= '0';
   estado_devolve <= '0';

   if (cancelar ='0') then
      devolucao <= valor_inserido; -- utilizado no estado devolve
      if (valor_inserido > 0) then
         proximo_estado <= Devolve;
      else
         proximo_estado <= Inicial;
      end if;
   else
      if (valor_inserido >= preco_salgado) then -- se atingiu o valor, libera salgado
         proximo_estado <= LiberaSalgado;
      else -- espera se nao atingiu
         proximo_estado <= EsperandoMoeda;
      end if;
   end if;
```

8.5 Libera Salgado

Chegando a máquina a esse estado, as demais condições para a liberação do salgado foram todas atendidas. Nesse momento, é dado o sinal de que o salgado foi liberado, define-se o troco e prepara-o

para a devolução e aguarda o sinal de liberação de salgado; dado esse sinal, acompanhado da definição de qual salgado será liberado, decrementa 1 unidade do seu estoque. Se depois disso, houver troco a ser retornado, vai para o estado Devolve; se não houver, vai para o Estado Inicial.

```
when LiberaSalgado => -- libera salgado escolhido caso pagamento tenha sido efetuado
   estado <= "101";
   -- definindo led do estado
   estado_inicial <= '0';
   estado_selecao_salgado <= '0';
   estado_estoque <= '0';
   estado_esperando_moeda <= '0';
   estado_libera_salgado <= '1';
   estado_devolve <= '0';

   salgado_liberado <= '1'; --led verde de salgado liberado
   troco <= valor_inserido - preco_salgado; -- define troco
   devolucao <= troco; -- utilizado no estado devolve
   if (liberar = '1')then
      case (salgado_escolhido) is
         when "001" =>
            estoque_salgado1 <= estoque_salgado1 - 0001; -- decrementa o estoque
         when "010" =>
            estoque_salgado2 <= estoque_salgado2 - 0001; -- decrementa o estoque
         when "011" =>
            estoque_salgado3 <= estoque_salgado3 - 0001; -- decrementa o estoque
         when "100" =>
            estoque_salgado4 <= estoque_salgado4 - 0001; -- decrementa o estoque
         when "101" =>
            estoque_salgado5 <= estoque_salgado5 - 0001; -- decrementa o estoque
         when others =>
      end case;
      --logica para devolucao de troco
      if (troco > 0) then
         proximo_estado <= Devolve;
      else
         proximo_estado <= Inicial;
      end if;
   else
      proximo_estado <= LiberaSalgado;
   end if;
```

8.6 Devolve

Criado para manter o sinal de “Devolvido” visível, pois se encapsulado em outros estados, tem sua visualização dificultada. Fica ativado até receber do usuário o sinal “Prosseguir”, por meio do switch,

quando vai para o Estado Inicial, onde esse sinal de “Devolvido” vai ser apagado.

```
            when Devolve =>
               estado <= "110";
               -- definindo led do estado
               estado_inicial <= '0';
               estado_selecao_salgado <= '0';
               estado_estoque <= '0';
               estado_esperando_moeda <= '0';
               estado_libera_salgado <= '0';
               estado_devolve <= '1';

               devolvido <= '1';-- led verde de dinheiro devolvido
               if(prosseguir = '1') then
                  proximo_estado <= Inicial;
               else
                  proximo_estado <= Devolve;
               end if;

            when others =>
         end case;
      end if;
   end process maquina_estados;
```

9 Process Display

Serve para mostrar no display informações referentes ao estado atual da máquina. Define variáveis do tipo inteiro como unidade, dezena, centena e o número do salgado para que essas informações sejam atualizadas conforme em que momento de seu funcionamento a máquina está.

- Inicial: São alocados os displays que exibirão o nº do salgado e valor.
- Seleção Salgado: Informa o nº do salgado através de um vetor de 3 bits, consultando o valor na posição informada no pedido, e informa também centena, dezena e unidade do preço referente ao salgado selecionado.
- Estoque: Informa o salgado escolhido e o valor já inserido.
- Esperando Moeda: Informa o salgado escolhido e o valor já inserido, atualizando-o a cada nova inserção de moeda.
- Libera salgado: Informa o salgado escolhido e o troco dele.
- Devolve: Informa o salgado escolhido e a devolução.

```
--  responsavel por atualizar as informacoes exibidas na maquina de salgados de acordo o estado atual
display: process(clock)
   variable centena, dezena, unidade, numero_salgado: integer range 0 to 999;
   begin
      if (clock'event and clock = '1') then
         case(estado_atual) is
            when Inicial => -- mostrar desligado
               display_salgado <= "1111111";
               display_moeda_centena <= "1111111";
               display_moeda_dezena <= "1111111";
               display_moeda_unidade <= "1111111";
```

```
when SelecaoSalgado =>
   case (salgado_escolhido) is
      when "001" =>
         numero_salgado := 001;
      when "010" =>
         numero_salgado := 002;
      when "011" =>
         numero_salgado := 003;
      when "100" =>
         numero_salgado := 004;
      when "101" =>
         numero_salgado := 005;
      when others =>
         numero_salgado := 000;
   end case;

   display_salgado <= display7(numero_salgado);
      case(numero_salgado) is
         when 001 =>
            display_moeda_centena <= display7(2);
            display_moeda_dezena <= display7(5);
            display_moeda_unidade <= display7(0);

         when 002 =>
            display_moeda_centena <= display7(1);
            display_moeda_dezena <= display7(5);
            display_moeda_unidade <= display7(0);

         when 003 =>
            display_moeda_centena <= display7(0);
            display_moeda_dezena <= display7(7);
            display_moeda_unidade <= display7(5);

         when 004 =>
            display_moeda_centena <= display7(3);
            display_moeda_dezena <= display7(5);
            display_moeda_unidade <= display7(0);

         when 005 =>
            display_moeda_centena <= display7(2);
            display_moeda_dezena <= display7(0);
            display_moeda_unidade <= display7(0);
         when others =>
            display_moeda_centena <= display7(0);
            display_moeda_dezena <= display7(0);
            display_moeda_unidade <= display7(0);
      end case;
```

```
when Estoque =>
   display_salgado <= display7(numero_salgado);
   unidade := (valor_inserido mod 10);
   dezena := (valor_inserido - unidade)/10;
   dezena := (dezena mod 10);
   centena := (valor_inserido - unidade)/10;
   centena := (centena - dezena)/10;

   display_moeda_centena <= display7(centena);
   display_moeda_dezena <= display7(dezena);
   display_moeda_unidade <= display7(unidade);


when EsperandoMoeda =>
   display_salgado <= display7(numero_salgado);
   unidade := (valor_inserido mod 10);
   dezena := (valor_inserido - unidade)/10;
   dezena := (dezena mod 10);
   centena := (valor_inserido - unidade)/10;
   centena := (centena - dezena)/10;

   display_moeda_centena <= display7(centena);
   display_moeda_dezena <= display7(dezena);
   display_moeda_unidade <= display7(unidade);


when LiberaSalgado =>
   display_salgado <= display7(numero_salgado);
   unidade := (troco mod 10);
   dezena := (troco - unidade)/10;
   dezena := (dezena mod 10);
   centena := (troco - unidade)/10;
   centena := (centena - dezena)/10;

   display_moeda_centena <= display7(centena);
   display_moeda_dezena <= display7(dezena);
   display_moeda_unidade <= display7(unidade);
```

```
            when Devolve =>
               display_salgado <= display7(numero_salgado);
               unidade := (devolucao mod 10);
               dezena := (devolucao - unidade)/10;
               dezena := (dezena mod 10);
               centena := (devolucao - unidade)/10;
               centena := (centena - dezena)/10;

               display_moeda_centena <= display7(centena);
               display_moeda_dezena <= display7(dezena);
               display_moeda_unidade <= display7(unidade);

            when others =>
         end case;
      end if;
   end process;
end hardware;
```

10 Pinagem

9.1 LEDs Verdes

COMPONENTES	PIN	FUNÇÃO
LEDG[0]	PIN_U22	saida = devolvido
LEDG[1]	PIN_U21	saida = salgado_liberado
LEDG[2]	PIN_V22	saida = estado_devolve
LEDG[3]	PIN_V21	saida = estado_libera_salgado
LEDG[4]	PIN_W22	saida = estado_esperando_moeda
LEDG[5]	PIN_W21	saida = estado_estoque
LEDG[6]	PIN_Y22	saida = estado_selecao_salgado
LEDG[7]	PIN_Y21	saida = estado_inicial

9.2 LEDs Vermelhos

COMPONENTES	PIN	FUNÇÃO
LEDR[0]	PIN_R20	saida =
LEDR[1]	PIN_R19	saida =
LEDR[2]	PIN_U19	saida =
LEDR[3]	PIN_Y19	saida =
LEDR[4]	PIN_T18	saida =
LEDR[5]	PIN_V19	saida =
LEDR[6]	PIN_Y18	saida =
LEDR[7]	PIN_U18	saida = moeda_nao_permitida
LEDR[8]	PIN_R18	saida = sem_estoque
LEDR[9]	PIN_R17	saida = salgado_invalido

9.3 Displays

COMPONENTES	PIN	FUNÇÃO
HEX0[0]	PIN_J2	saida = unidade[0]
HEX0[1]	PIN_J1	saida = unidade[1]
HEX0[2]	PIN_H2	saida = unidade[2]
HEX0[3]	PIN_H1	saida = unidade[3]
HEX0[4]	PIN_F2	saida = unidade[4]
HEX0[5]	PIN_F1	saida = unidade[5]
HEX0[6]	PIN_E2	saida = unidade[6]
HEX1[0]	PIN_E1	saida = dezena[0]
HEX1[1]	PIN_H6	saida = dezena[1]
HEX1[2]	PIN_H5	saida = dezena[2]
HEX1[3]	PIN_H4	saida = dezena[3]
HEX1[4]	PIN_G3	saida = dezena[4]
HEX1[5]	PIN_D2	saida = dezena[5]
HEX1[6]	PIN_D1	saida = dezena[6]
HEX2[0]	PIN_G5	saida = centena[0]
HEX2[1]	PIN_G6	saida = centena[1]
HEX2[2]	PIN_C2	saida = centena[2]
HEX2[3]	PIN_C1	saida = centena[3]
HEX2[4]	PIN_E3	saida = centena[4]
HEX2[5]	PIN_E4	saida = centena[5]
HEX2[6]	PIN_D3	saida = centena[6]
HEX3[0]	PIN_F4	saida = salgado[0]
HEX3[1]	PIN_D5	saida = salgado[1]
HEX3[2]	PIN_D6	saida = salgado[2]
HEX3[3]	PIN_J4	saida = salgado[3]
HEX3[4]	PIN_L8	saida = salgado[4]
HEX3[5]	PIN_F3	saida = salgado[5]
HEX3[6]	PIN_D4	saida = salgado[6]

9.4 Switches

COMPONENTES	PIN	FUNÇÃO
Toggle Switch[0]	PIN_L22	entrada = prosseguir
Toggle Switch[1]	PIN_L21	entrada = liberar
Toggle Switch[2]	PIN_M22	entrada = moeda[0]
Toggle Switch[3]	PIN_V12	entrada = moeda[1]
Toggle Switch[4]	PIN_W12	
Toggle Switch[5]	PIN_U12	entrada = salgado_escolhido[0]
Toggle Switch[6]	PIN_U11	entrada = salgado_escolhido[1]
Toggle Switch[7]	PIN_M2	entrada = salgado_escolhido[2]
Toggle Switch[8]	PIN_M1	entrada = inicia
Toggle Switch[9]	PIN_L2	entrada = reset

9.5 Botões

COMPONENTES	PIN	FUNÇÃO
Push Button[0]	PIN_R22	entrada = cancelar
Push Button[1]	PIN_R21	
Push Button[2]	PIN_T22	entrada = confirma_moeda
Push Button[3]	PIN_T21	entrada = canfirma_salgado
CLOCK_27	PIN_D12	entrada = clock

11 Referências bibliográficas

- YouTube, 2005. Disponível em: < https://www.youtube.com/channel/UC6lwL_1tcr7ubV1clS-D-KQ>. Acesso em: 11/12/2023.

- FPGA Para Todos, 2011. Disponível em: < http://fpgaparatodos.com.br/>. Acesso em: 09/12/2023.

12 Formas de Onda