UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA – UESB
DEPARTAMENTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS – DCET
COLEGIADO DO CURSO DE COMPUTAÇÃO – CCCOMP

DOCUMENTAÇÃO DE UMA MÁQUINA DE VENDER SALGADOS EM VHDL PARA UMA PLACA DE FPGA

Ademir de Jesus Reis Júnior
Cauê Rodrigues de Aguiar
João Henrique Silva Pinto

Vitória da Conquista – BA
Dezembro de 2023

ADEMIR DE JESUS REIS JÚNIOR
CAUÊ RODRIGUES DE AGUIAR
JOÃO HENRIQUE SILVA PINTO

DOCUMENTAÇÃO DE UMA MÁQUINA DE VENDER SALGADOS EM VHDL PARA UMA PLACA DE FPGA

Trabalho apresentado à disciplina de Circuitos Digitais (2023.2), do Curso de Ciência da Computação, da Universidade Estadual do Sudoeste da Bahia (UESB), como requisito parcial para a avaliação da referida disciplina.

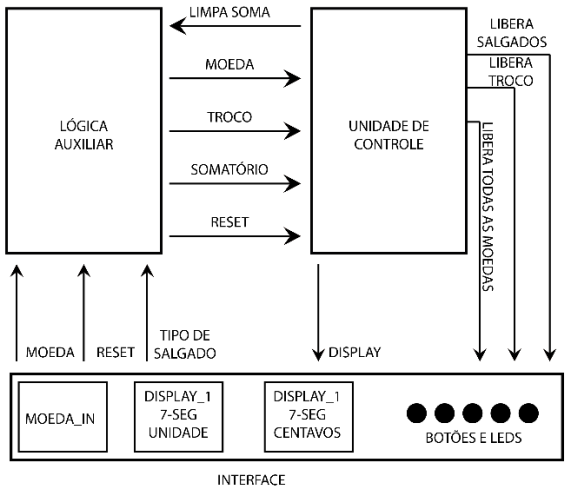
Prof. Orientador: **Prof. Me. Marco Antonio Dantas Ramos**

Vitória da Conquista – BA
Dezembro de 2023

I. RESUMO

O principal objetivo deste projeto é implementar uma máquina de vender salgados em VHDL. A máquina aceita somente moedas de 0,25 e 0,50 centavos e de R$ 1,00. Os produtos dispensados pela máquina são: batata frita grande (R$ 2,50), batata frita média (R$ 1,50), batata frita pequena (R$ 0,75), tortilha grande (R$ 3,50), tortilha pequena (R$ 2,00). Quando não há estoque disponível do salgado escolhido, a máquina emite um aviso. Caso contrário, o cliente pode inserir as moedas. Quando o cliente insere moedas que não são aceitas, a máquina as devolve, não realizando a soma. Uma vez inseridas as moedas aceitas, se o cliente desistir da compra, todas as moedas inseridas por ele são devolvidas. A máquina também exibe o somatório no display. Quando o valor das moedas válidas inseridas é igual ao valor do item escolhido, a máquina dispensa o produto, se o valor for acima, a máquina devolve o troco e dispensa o item escolhido.

II. ESQUEMÁTICO DA MÁQUINA



III. MÁQUINA DE ESTADOS

Considerando o diagrama acima, nossa máquina de vendas possui os seguintes estados:

- Estado Inicial/Seleção: Estado inicial da máquina, onde o cliente pode escolher o produto desejado.

- Inserção de moedas: Estado em que o cliente insere moedas na máquina e soma. A máquina aceita apenas moedas de 0,25, 0,50 e R$ 1,00.
- Dispensação: Estado em que a máquina dispensa o produto escolhido.

No estado inicial/seleção, o cliente escolhe o salgado informando o número da escolha (1 a 5), uma vez escolhido, aparece no display o número do salgado no display, seguido do valor, conforme a tabela abaixo:

Tabela 1: Especificações de Salgado e Valores

Salgados	Visualização no display
01-Batata frita grande: R$ 2,50	1250
02-Batata frita média: R$ 1,50	2150
03-Batata frita pequena: R$ 0,75	3075
04-Tortilha grande: R$ 3,50	4350
05-Tortilha pequena: R$ 2,00	5200

Quando o cliente escolher o salgado, a máquina muda para o segundo estado, que é o da inserção de moedas. O cliente pode então inserir moedas de diferentes tipos, mas só serão somadas as moedas aceitas: 0,25, 0,50 e R$ 1,00. Moedas diferentes destas serão devolvidas.

O próximo estado é o da dispensação do salgado, quando a máquina sinaliza que o salgado foi entregue, juntamente com o troco, se for o caso.

As transições entre os estados são as seguintes:

- Do estado Inicial/Seleção para o estado Inserção de Moedas, a transição ocorre quando o cliente pressiona o botão de confirmar o salgado desejado;
- Do estado Inserção de Moedas para o estado de Dispensação de Salgado, a transição ocorre quando o cliente insere moedas suficientes para comprar o salgado escolhido;
- Do estado Dispensação de Salgado para o estado Inicial/Seleção, a transição ocorre quando o a máquina indica que o salgado e o troco foram entregues ao cliente;

Obs.: a qualquer momento, o cliente pode desistir da comprando, pressionando o botão de cancelar, o qual retorna a máquina para o estado Inicial/Seleção, devolvendo as moedas, caso o cliente as tenha inserido.

A máquina de venda de salgados em VHDL pode ser representada por um diagrama de estados como o seguinte:

- q0: estado inicial/seleção
- q1: inserção de moedas
- q2: dispensa do produto

Figura 1: Estados da Máquina de Venda

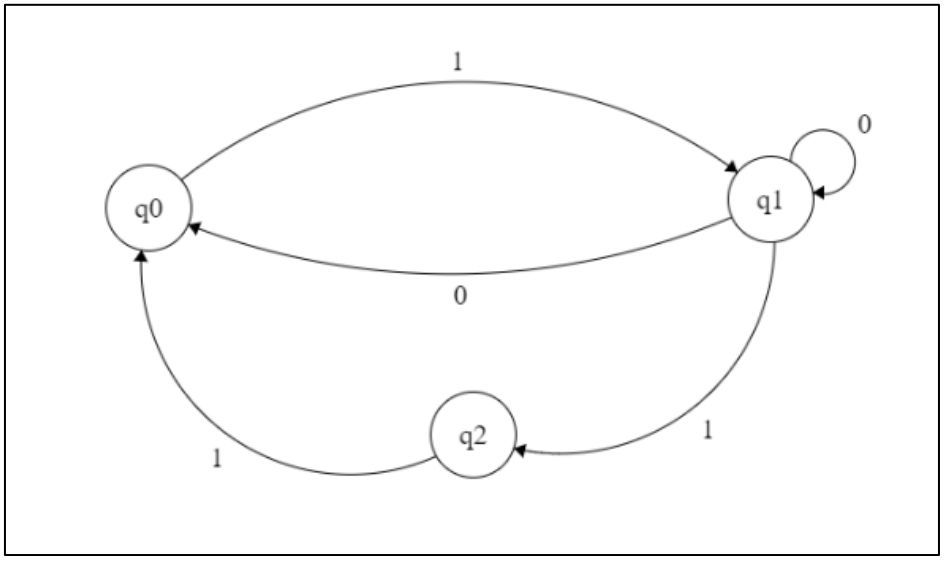


Figura 2: Esquema da Máquina de Estados no VHDL

State Machine - |vending_machine|CSTATE

Encoding Type: One-Hot

	Name	CSTATE.salgado_dispensation	CSTATE.Coin_Reception	CSTATE.INIT_STATE
1	CSTATE.INIT_STATE	0	0	0
2	CSTATE.Coin_Reception	0	1	1
3	CSTATE.salgado_dispensation	1	0	1

IV. ESTRUTURA DA MÁQUINA EM VHDL

Considerando a hierarquia de projetos no Quartus II 64bit, tem-se a máquina de vender salgados na seguinte estrutura:

1. Vending_machine:
    a. Accumulator8;
    b. Comparator8;
    c. Lpm_divide:Div0;
    d. Lpm_divide:Div1;
    e. Lpm_divide:Mod0;
    f. Lpm_divide:Mod1
    g. Mux21:mux;
    h. Subtractor8:subtractor.

No topo da estrutura hierárquica, está a codificação para a configuração da máquina de vender salgados. Tem-se então a importação das bibliotecas utilizadas, a estrutura da "entity" da vending_machine, conjuntamente com sua arquitetura. Abaixo, apresenta-se os códigos comentados:

```
1. library ieee;
2. use ieee.std_logic_1164.all;
3. use ieee.numeric_std.all;
```

Em VHDL, as linhas acima são utilizadas para importar as bibliotecas, que fornecem acesso a definições e pacotes predefinidos. A primeira linha declara a inclusão da biblioteca "ieee" (Institute of Electrical and Electronics Engineers), que é uma coleção padrão de pacotes e definições, os quais permitem que se utilizem vários recursos e tipos de dados padrão. A segunda linha importa todos os elementos do pacote "std_logic_1164", que permitem utilizar as definições relacionadas a sinais lógicos e funções relacionadas a operações lógicas. A terceira linha importa os elementos do "numeric_std", que permitem trabalhar com as definições com operações numéricas, tais como adição e subtração, permitindo utilização dos tipos de dados como "signed" e "unsigned".

```
1. entity vending_machine is
2.     port(
3.         nRST : in std_logic; -- Reset (para nova compra)
4.         clk : in std_logic; -- Clock padrao
```

```
 5.         C : in std_logic; -- Sensor de moedas ('1' para quando uma moeda eh inserida)
 6.         V_input : in std_logic_vector(2 downto 0); -- Entra de valores de moeda
 7.         choice : in std_logic_vector(2 downto 0); -- Escolha do salgado
 8.         cancel_purchase : in std_logic;  -- Opcao de cancelamento de compra
 9.
10.         dispense_signal : in std_logic; --Entrada para confirmar o salgado escolhido
11.         coin_confirm_signal : in std_logic; --Entrada para confirmar uma moeda adicionada
12.
13.         hexDisplay_choice : out std_logic_vector(6 downto 0); --Saida para o Display Opcao
Salgado
14.         hexDisplay_centena : out std_logic_vector(6 downto 0); --Saida para o Display
Dinheiro Centena
15.         hexDisplay_dezena : out std_logic_vector(6 downto 0);  --Saida para o Display
Dinehrio Dezena
16.         hexDisplay_unidade : out std_logic_vector(6 downto 0); --Saida para o Display
Dinheiro Unidade
17.
18.         P : out std_logic_vector(8 downto 0); -- Acumulador de moedas
19.         E : out std_logic_vector(8 downto 0); -- Troco/Retorno de moedas
20.         D1 : out std_logic_vector(2 downto 0) := "000"; -- Sinal para salgado liberado
21.         D2 : out std_logic_vector(2 downto 0) := "000"; -- Sinal para salgado liberado
22.         D3 : out std_logic_vector(2 downto 0) := "000"; -- Sinal para salgado liberado
23.         D4 : out std_logic_vector(2 downto 0) := "000"; -- Sinal para salgado liberado
24.         D5 : out std_logic_vector(2 downto 0) := "000"; -- Sinal para salgado liberado
25.         ESTQ1 : out std_logic_vector(8 downto 0) := "000000000"; -- Aviso de quantidade em
estoque
26.         ESTQ2 : out std_logic_vector(8 downto 0) := "000000000"; -- Aviso de quantidade em
estoque
27.         ESTQ3 : out std_logic_vector(8 downto 0) := "000000000"; -- Aviso de quantidade em
estoque
28.         ESTQ4   : out std_logic_vector(8 downto 0) := "000000000"; -- Aviso de quantidade em
estoque
29.         ESTQ5   : out std_logic_vector(8 downto 0) := "000000000" -- Aviso de quantidade em
estoque
30.     );
31. end vending_machine;
```

A entidade “vending_machine” representa uma máquina de venda automática, possuindo diversas portas de entradas (“in”) e saída (“out”), utilizadas para interconectar a máquina com outros componentes em um sistema digital. Sendo assim, a vending_machine modela uma máquina de venda automática com funcionalidade para aceitar moedas, realizar escolhas de produtos, gerenciar estoque e fornecer informações visuais em displays.

1. **Portas de entrada (“in”):**
    a. **“nRST”:** sinal de reset para iniciar uma nova compra;
    b. **“clk”:** sinal de clock padrão;
    c. **“C”:** sensor de moedas, onde “1” indica a inserção de uma moeda;
    d. **“V_input”:** entrada de valores de moedas representada como um vetor de lógica padrão de 3 bits;
    e. **“choice”:** vetor de lógica padrão de 3 bits representando a escolha do salgado;
    f. **“cancel_purchase”:** opção de cancelar a compra;
    g. **“dispense_signal”:** sinal para confirmar a dispensa do salgado escolhido;

h. **“coin_confirm_signal”:** sinal para confirmar a adição de uma moeda.

2. **Portas de saída (“out”):**
    a. **“hexDisplay_choice”:** saída para o display que mostra a opção do salgado escolhido;
    b. **“hexDisplay_centena”, “hexDisplay_dezena”, “hexDisplay_unidade”:** saídas para displays que representam o valor em dinheiro nas centenas, dezenas e unidades, respectivamente;
    c. **“P”:** acumulador de moedas;
    d. **“E”:** troco/retorno de moedas;
    e. **“D1”, “D2”, “D3”, “D4”, “D5”:** sinais indicando se os salgados 1 a 5 estão liberados ("000" indica não liberado);
    f. **“ESTQ1”, “ESTQ2”, “ESTQ3”, “ESTQ4”, “ESTQ5”:** avisos de quantidade em estoque para os salgados 1 a 5.

3. **Observações adicionais:**
    a. Os displays são representados como vetores de lógica padrão de 7 bits;
    b. As quantidades em estoque são representadas como vetores de lógica padrão de 9bits;

A arquitetura “rtl” do código VHDL abaixo define a lógica de controle e operações de uma máquina de venda automática representada pela entidade “vending_machine”. O código da arquitetura é o que se segue:

```
1. architecture rtl of vending_machine is
2.     constant S0 : std_logic_vector(8 downto 0) := "011111010"; -- Preço da escolha 1 (250)
3.     constant S1 : std_logic_vector(8 downto 0) := "010010110"; -- Preço da escolha 2 (150)
4.     constant S2 : std_logic_vector(8 downto 0) := "001001011"; -- Preço da escolha 3 (75)
5.     constant S3 : std_logic_vector(8 downto 0) := "101011110"; -- Preço da escolha 4 (350)
6.     constant S4 : std_logic_vector(8 downto 0) := "011001000"; -- Preço da escolha 5 (200)
7.
8.     component accumulator8 is -- Acumulador de 8 bits
9.     port(
10.        clk     : in std_logic; -- Clock de entrada
11.        nRST_acc    : in std_logic; -- Reset
12.        C           : in std_logic; -- Sensor de moeda
13.        data_in     : in std_logic_vector(8 downto 0); -- Valor de entrada
14.        data_out : out std_logic_vector(8 downto 0); -- Valor de saida
```

```
15.         conf: in std_logic; --Entrada para confirmar uma moeda adicionada
16.         cancel : in std_logic
17.     );
18.     end component;
19.
20.     component adder8 is -- Somador de 8 bits
21.     port(
22.         a       : in std_logic_vector(8 downto 0); -- Operando 1
23.         b       : in std_logic_vector(8 downto 0); -- Operando 2
24.         c_in    : in std_logic; -- Bit pra moeda de entrada
25.         s       : out std_logic_vector(8 downto 0); -- Soma total
26.         c_out   : out std_logic -- Bit pra moeda de saida
27.     );
28.     end component;
29.
30.     component subtractor8 is -- Subtrator de 8 bits
31.     port(
32.         a       : in std_logic_vector(8 downto 0); -- Operando 1
33.         b       : in std_logic_vector(8 downto 0); -- Operando 2
34.         result: out std_logic_vector(8 downto 0) -- Resultado final
35.     );
36.     end component;
37.
38.     component comparator8 is -- Comparador de 8 bits
39.     port(
40.         a       : in std_logic_vector(8 downto 0); -- 1o input
41.         b       : in std_logic_vector(8 downto 0); -- 2o input
42.         g_out : out std_logic; -- Saida g
43.         e_out : out std_logic; -- Saida e
44.         l_out : out std_logic  -- Saida l
45.     );
46.     end component;
47.
48.     component mux21 is -- Componente para escolha do salgado
49.     port(
50.         A   : in std_logic_vector(8 downto 0); -- Escolha 01
51.         B   : in std_logic_vector(8 downto 0); -- Escolha 02
52.         C   : in std_logic_vector(8 downto 0); -- Escolha 03
53.         D   : in std_logic_vector(8 downto 0); -- Escolha 04
54.         E   : in std_logic_vector(8 downto 0); -- Escolha 05
55.         s   : in std_logic_vector(2 downto 0); -- Numero da escolha
56.         output : out std_logic_vector(8 downto 0) -- Saida
57.     );
58.     end component;
59.
60.     type FSMTYPE is (INIT_STATE, Coin_Reception, salgado_dispensation); -- Estados da
Maquina
61.
62.     signal CSTATE, NSTATE : FSMTYPE; -- Sinais para estado de moeda (CoinState) e proximo
estado (NextState)
63.
64.     signal balance, price, price_reg, coins_to_return : std_logic_vector(8 downto 0); --
Sinais para valores de preco, troco, retorno de moeda, etc.
65.
66.     signal price_choice_reg_EN, balance_greater, balance_equal, balance_lower: std_logic; --
Bits de controle para valores
67.
68.     signal nRST_acc : std_logic; -- Reset
69.     signal choice_reg : std_logic_vector(2 downto 0); -- Escolha do salgado
70.     signal unidade_reg : integer range 0 to 9999 := 0; -- Unidades de salgado
71.     signal stock_S0_reg, stock_S1_reg, stock_S2_reg, stock_S3_reg, stock_S4_reg,
stock_S5_reg : integer := 1; -- Estoque total de cada salgado na maquina (definido pelo
desenvolvedor)
72.
73.     signal V : STD_LOGIC_VECTOR (8 downto 0); -- Entrada de valor de moeda
74.     signal c_out : std_logic; -- Bit de controle para saida de moeda
75.     signal c_in : std_logic; -- Bit de controle para entrada de moeda
76.
77.     signal dispensation_EN : std_logic; -- Bit de controle para liberacao de salgado
78.
```

```
 79.     signal salgado_selecionado: integer range 1 to 6 := 1; --Salgado selecionado pelo
cliente
 80.
 81.     --Funcao para converter Numero Inteiro para Vetor de Saida do Display de 7 seg
 82.     function intToSevenSeg (numero: integer) return std_logic_vector is --Funcao recebe um
numero inteiro
 83.         variable bin_output: std_logic_vector(6 downto 0); --Vetor de 7 bits para saida do
Display
 84.
 85.     begin --Inicio da Funcao
 86.         case (numero) is
 87.             when 0 => bin_output := "1000000"; --Saida recebe o numero 0 na Cod. do Display
de 7 Seg
 88.             when 1 => bin_output := "1111001"; --Saida recebe o numero 1 na Cod. do Display
de 7 Seg
 89.             when 2 => bin_output := "0100100"; --Saida recebe o numero 2 na Cod. do Display
de 7 Seg
 90.             when 3 => bin_output := "0110000"; --Saida recebe o numero 3 na Cod. do Display
de 7 Seg
 91.             when 4 => bin_output := "0011001"; --Saida recebe o numero 4 na Cod. do Display
de 7 Seg
 92.             when 5 => bin_output := "0010010"; --Saida recebe o numero 5 na Cod. do Display
de 7 Seg
 93.             when 6 => bin_output := "0000010"; --Saida recebe o numero 6 na Cod. do Display
de 7 Seg
 94.             when 7 => bin_output := "1111000"; --Saida recebe o numero 7 na Cod. do Display
de 7 Seg
 95.             when 8 => bin_output := "0000000"; --Saida recebe o numero 8 na Cod. do Display
de 7 Seg
 96.             when 9 => bin_output := "0010000"; --Saida recebe o numero 9 na Cod. do Display
de 7 Seg
 97.             when others =>
 98.         end case;
 99.     return bin_output; --Retornando o vetor de Saida
100.     end intToSevenSeg; --Fim da funcao
101.
102. begin
103.
104.     price_registration : process( CLK ) -- Registro de Preco
105.     begin
106.         if (CLK'event and CLK = '1') then
107.             if (price_choice_reg_EN = '1') then
108.                 price_reg <= price;
109.                 choice_reg <= choice;
110.             end if ;
111.         end if ;
112.    end process;
113.
114.     state_registration : process( CLK ) -- Registro de Estado
115.     begin
116.         if (CLK'event and CLK = '1') then
117.             if (nRST = '0') then
118.                 CSTATE <= INIT_STATE;
119.             else
120.                 CSTATE <= NSTATE;
121.             end if;
122.         end if;
123.    end process;
124.
125.     salgado_dispensation_proc: process(CLK) -- Liberacao de salgado
126.     begin
127.         if (CLK'event and CLK = '1') then
128.             if (nRST = '0') then
129.             D1 <= (others => '0'); -- Sinal para saida de salgado (incialmente zerado)
130.             D2 <= (others => '0'); -- Sinal para saida de salgado (incialmente zerado)
131.             D3 <= (others => '0'); -- Sinal para saida de salgado (incialmente zerado)
132.             D4 <= (others => '0'); -- Sinal para saida de salgado (incialmente zerado)
133.             D5 <= (others => '0'); -- Sinal para saida de salgado (incialmente zerado)
134.
```

```
135.             ESTQ1 <= (others => '0'); -- Sinal de estoque (inicialmente 0 - tem salgado pra
vender)
136.             ESTQ2 <= (others => '0'); -- Sinal de estoque (inicialmente 0 - tem salgado pra
vender)
137.             ESTQ3 <= (others => '0'); -- Sinal de estoque (inicialmente 0 - tem salgado pra
vender)
138.             ESTQ4 <= (others => '0'); -- Sinal de estoque (inicialmente 0 - tem salgado pra
vender)
139.             ESTQ5 <= (others => '0'); -- Sinal de estoque (inicialmente 0 - tem salgado pra
vender)
140.             end if;
141.             if (cancel_purchase = '0') then
142.                 case choice_reg is
143.                     when "001" =>
144.                         if stock_S0_reg <= 0  then
145.                              -- Sem estoque pro produto 1
146.                              -- Permanece no estado inicial
147.                              ESTQ1 <= "000000001";
148.                         end if;
149.                   when "010" =>
150.                         if stock_S1_reg <= 0  then
151.                              -- Sem estoque pro produto 2
152.                              -- Permanece no estado inicial
153.                              ESTQ2 <= "000000001";
154.                         end if;
155.                   when "011" =>
156.                         if stock_S2_reg <= 0  then
157.                              -- Sem estoque pro produto 3
158.                              -- Permanece no estado inicial
159.                              ESTQ3 <= "000000001";
160.                         end if;
161.                   when "100" =>
162.                         if stock_S3_reg <= 0  then
163.                              -- Sem estoque pro produto 4
164.                              -- Permanece no estado inicial
165.                              ESTQ4 <= "000000001";
166.                         end if;
167.                   when "101" =>
168.                         if stock_S4_reg <= 0  then
169.                              -- Sem estoque pro produto 5
170.                              -- Permanece no estado inicial
171.                              ESTQ5 <= "000000001";
172.                         end if;
173.                   when others =>
174.                         null;
175.                end case;
176.
177.                 if (dispensation_EN = '1') then
178.                     if(choice_reg = "001") then -- Para o produto 1
179.                         if stock_S0_reg > 0 then
180.                             salgado_selecionado <= 1;
181.                             D1 <= "001";
182.                             stock_S0_reg <= stock_S0_reg - 1;
183.                   end if;
184.                     elsif(choice_reg = "010") then -- Para o produto 2
185.                         if stock_S1_reg > 0 then
186.                             salgado_selecionado <= 2;
187.                             D2 <= "001";
188.                          stock_S1_reg <= stock_S1_reg - 1;
189.                         end if;
190.                     elsif(choice_reg = "011") then -- Para o produto 3
191.                         if stock_S2_reg > 0 then
192.                             salgado_selecionado <= 3;
193.                             D3 <= "001";
194.                             stock_S2_reg <= stock_S2_reg - 1;
195.                         end if;
196.                     elsif(choice_reg = "100") then -- Para o produto 4
197.                         if stock_S3_reg > 0 then
198.                             salgado_selecionado <= 4;
199.                             D4 <= "001";
```

```
200.                            stock_S3_reg <= stock_S3_reg - 1;
201.                        end if;
202.                    elsif(choice_reg = "101") then -- Para o produto 5
203.                        if stock_S4_reg > 0 then
204.                            salgado_selecionado <= 5;
205.                            D5 <= "001";
206.                            stock_S4_reg <= stock_S4_reg - 1;
207.                        end if;
208.                    end if;
209.                end if;
210.            end if;
211.        end if;
212.    end process; --salgado_dispensation_Proc;
213.
214.    hex_display_updt : process(clk)
215.    variable hundred, ten, unit : integer range 0 to 999; -- Variaveis de calculo para valor
no display
216.    begin
217.        if (clk'event and clk = '1') then
218.            case (CSTATE) is
219.                when INIT_STATE =>
220.                    case choice is
221.                        when "001" =>
222.                            hexDisplay_choice       <= intToSevenSeg(1);
223.                            hexDisplay_centena  <= intToSevenSeg(2); --Display que mostra a
Centena
224.                            hexDisplay_dezena   <= intToSevenSeg(5);  --Display que mostra a
Dezena
225.                            hexDisplay_unidade  <= intToSevenSeg(0); --Display que mostra a
Unidade
226.
227.                      when "010" =>
228.                            hexDisplay_choice       <= intToSevenSeg(2);
229.                            hexDisplay_centena  <= intToSevenSeg(1); --Display que mostra a
Centena
230.                            hexDisplay_dezena   <= intToSevenSeg(5);  --Display que mostra a
Dezena
231.                            hexDisplay_unidade  <= intToSevenSeg(0); --Display que mostra a
Unidade
232.
233.                      when "011" =>
234.                            hexDisplay_choice       <= intToSevenSeg(3);
235.                            hexDisplay_centena  <= intToSevenSeg(0); --Display que mostra a
Centena
236.                            hexDisplay_dezena   <= intToSevenSeg(7);  --Display que mostra a
Dezena
237.                            hexDisplay_unidade  <= intToSevenSeg(5); --Display que mostra a
Unidade
238.
239.                      when "100" =>
240.                            hexDisplay_choice       <= intToSevenSeg(4);
241.                            hexDisplay_centena  <= intToSevenSeg(3); --Display que mostra a
Centena
242.                            hexDisplay_dezena   <= intToSevenSeg(5);  --Display que mostra a
Dezena
243.                            hexDisplay_unidade  <= intToSevenSeg(0); --Display que mostra a
Unidade
244.
245.                      when "101" =>
246.                            hexDisplay_choice       <= intToSevenSeg(5);
247.                            hexDisplay_centena  <= intToSevenSeg(2); --Display que mostra a
Centena
248.                            hexDisplay_dezena   <= intToSevenSeg(0);  --Display que mostra a
Dezena
249.                            hexDisplay_unidade  <= intToSevenSeg(0); --Display que mostra a
Unidade
250.
251.                      when others =>
252.                            hexDisplay_choice   <= "1111111"; --Desligando Display
253.                            hexDisplay_centena <= "1111111"; --Desligando Display
```

```
254.                             hexDisplay_dezena  <= "1111111"; --Desligando Display
255.                             hexDisplay_unidade <= "1111111"; --Desligando Display
256.                     end case;
257.
258.                 when Coin_Reception =>
259.                     hexDisplay_choice       <= intToSevenSeg(0);
260.
261.                     -- Obtendo a Centena
262.                     hexDisplay_centena  <= intToSevenSeg(to_integer(unsigned(balance)) /
100);
263.                     -- Obtendo a Dezena
264.                     hexDisplay_dezena   <= intToSevenSeg((to_integer(unsigned(balance)) /
10) mod 10);
265.                     -- Obtendo a Unidade
266.                     hexDisplay_unidade  <= intToSevenSeg(to_integer(unsigned(balance)) mod
10);
267.                 when others =>
268.             end case;
269.         end if;
270.     end process;
271.
272.     next_state : process( CSTATE, balance, C, balance_equal, balance_greater,
coins_to_return)
273.     begin
274.         if (nRST = '0') then
275.             P <= (others => '0');
276.             E <= (others => '0');
277.         end if;
278.
279.         NSTATE      <= CSTATE;  -- Configura a alteracao do NSTATE (NextSTATE) para o estado
atual CSTATE (CurrentSTATE)
280.         nRST_acc    <= '1';         -- Reset em 1 (off)
281.       price_choice_reg_EN   <= '0'; -- Bit de controle de preco
282.         dispensation_EN         <= '0'; -- Bit de controle para salgado liberado ou nao
283.
284.         if (dispense_signal = '0') then
285.             P <= (others => '0'); -- Acumulo de moedas
286.         end if;
287.
288.         case to_integer(unsigned(V_input)) is -- Valores-teste de entrada de V
289.              when 1 =>
290.                   V <= "000000101";
291.              when 2 =>
292.                   V <= "000001010";
293.              when 3 =>
294.                   V <= "000011001";
295.              when 4 =>
296.                   V <= "000110010";
297.              when 5 =>
298.                   V <= "001100100";
299.              when others =>
300.                   V <= "000000000";
301.         end case;
302.
303.         case( CSTATE ) is
304.             when INIT_STATE => -- Quando no estado inicial
305.                 if cancel_purchase = '1' then -- Cancela a compra e reseta valores
306.                     E <= balance;
307.                     P <= (others => '0');
308.
309.                     if V /= "000011001" and V /= "000110010" and V /= "001100100" and C =
'1' then
310.                         E <= V; -- Retorna qualquer moeda invalida que fora inserida
311.                     end if;
312.
313.                     NSTATE <= INIT_STATE; -- Retorna para o estado inicial
314.                 else -- Caso a compra siga adiante
315.
316.                     nRST_acc <= '0';
317.                     price_choice_reg_EN <= '1';
```

```
318.
319.                 case choice_reg is
320.                     when "001" =>
321.                         if stock_S0_reg <= 0  then
322.                              -- Sem estoque pro produto 1
323.                              -- Permanece no estado inicial
324.                              NSTATE <= INIT_STATE;
325.                         else
326.                             nRST_acc <= '1';
327.                             price_choice_reg_EN <= '1';
328.                             E <= balance;
329.                             NSTATE <= Coin_Reception;
330.                         end if;
331.                   when "010" =>
332.                         if stock_S1_reg <= 0  then
333.                              -- Sem estoque pro produto 2
334.                              -- Permanece no estado inicial
335.                              NSTATE <= INIT_STATE;
336.                         else
337.                             nRST_acc <= '1';
338.                             price_choice_reg_EN <= '1';
339.                             E <= balance;
340.                             NSTATE <= Coin_Reception;
341.                         end if;
342.                   when "011" =>
343.                         if stock_S2_reg <= 0  then
344.                              -- Sem estoque pro produto 3
345.                              -- Permanece no estado inicial
346.                              NSTATE <= INIT_STATE;
347.                         else
348.                             nRST_acc <= '1';
349.                             price_choice_reg_EN <= '1';
350.                             E <= balance;
351.                             NSTATE <= Coin_Reception;
352.                         end if;
353.                   when "100" =>
354.                         if stock_S3_reg <= 0  then
355.                              -- Sem estoque pro produto 4
356.                              -- Permanece no estado inicial
357.                              NSTATE <= INIT_STATE;
358.                         else
359.                             nRST_acc <= '1';
360.                             price_choice_reg_EN <= '1';
361.                             E <= balance;
362.                             NSTATE <= Coin_Reception;
363.                         end if;
364.                   when "101" =>
365.                         if stock_S4_reg <= 0  then
366.                              -- Sem estoque pro produto 5
367.                              -- Permanece no estado inicial
368.                              NSTATE <= INIT_STATE;
369.                         else
370.                             nRST_acc <= '1';
371.                             price_choice_reg_EN <= '1';
372.                             E <= balance;
373.                             NSTATE <= Coin_Reception;
374.                         end if;
375.                   when others =>
376.                         null;
377.                 end case;
378.
379.                 if C = '1' and (choice = "001" and stock_S0_reg > 0) then
380.                     nRST_acc <= '1';
381.                     NSTATE <= Coin_Reception;
382.
383.                 elsif C = '1' and (choice = "010" and stock_S1_reg > 0) then
384.                     nRST_acc <= '1';
385.                     NSTATE <= Coin_Reception;
386.
387.                 elsif C = '1' and (choice = "011" and stock_S2_reg > 0) then
```

```
388.                     nRST_acc <= '1';
389.                     NSTATE <= Coin_Reception;
390.
391.                 elsif C = '1' and (choice = "100" and stock_S3_reg > 0) then
392.                     nRST_acc <= '1';
393.                     NSTATE <= Coin_Reception;
394.
395.                 elsif C = '1' and (choice = "101" and stock_S4_reg > 0) then
396.                     nRST_acc <= '1';
397.                     NSTATE <= Coin_Reception;
398.
399.                 end if;
400.
401.             end if; -- Fim do else (if cancel_purchase = '0')
402.
403.
404.             when Coin_Reception => -- Quando no estado de recepcao de moedas
405.                 if cancel_purchase = '1' then -- Caso o cliente cancele
406.                     -- Cancela a compra, reseta os valores e retorna ao estado inicial
407.                     E <= balance;
408.                     P <= (others => '0');
409.                     NSTATE <= INIT_STATE;
410.                 else -- Caso a compra siga adiante
411.                     P <= balance; -- Acumulamos os valores de moeda em P
412.
413.                     if V /= "000011001" and V /= "000110010" and V /= "001100100" then
414.                     -- Retornamos ao usuario qualquer moeda invalida
415.                         E <= V;
416.                     end if;
417.
418.                     if ((balance_equal = '1' or balance_greater = '1') and dispense_signal =
'0') then
419.                     -- Se o acumulo de moedas for igual ou maior que o preco do salgado e
houver o sinal para liberacao:
420.                         if cancel_purchase = '0' then
421.                             dispensation_EN <= '1';
422.                         end if;
423.                         NSTATE <= salgado_dispensation;
424.                 end if;
425.             end if;
426.
427.             when salgado_dispensation => -- Quando no estado de liberacao do salgado
428.                 if cancel_purchase = '1' then-- Caso o cliente cancele
429.                 -- Cancela a compra, limpa o acumulado em P e retorna ao estado inicial
430.                     E <= balance;
431.                     P <= (others => '0');
432.                     NSTATE <= INIT_STATE;
433.                 elsif (dispense_signal = '0') then -- Caso a compra siga adiante
434.                     E <= coins_to_return; -- Troco
435.
436.                     nRST_acc <= '0'; -- Reset da compra
437.                     NSTATE <= INIT_STATE; -- Volta ao estado inicial
438.                 end if;
439.             when others => -- Qualquer outro estado nao estara configurado para a maquina
440.         end case ;
441.     end process ; -- Fim do processo NSTATE (NextState)
442.
443.
444.     -- Instancias para constroladores de display e manipulacao de valoes inseridos (acumulo,
soma, subtracao, etc.)
445.     mux : mux21 port map(S0, S1, s2, s3, s4, choice, price);
446.
447.     accumulator : accumulator8 port map (clk, nRST_acc, C, V, balance, coin_confirm_signal,
cancel_purchase);
448.
449.     comparator : comparator8 port map (balance, price_reg, balance_greater, balance_equal,
balance_lower);
450.
451.     subtractor : subtractor8 port map (balance, price_reg, coins_to_return);
452. end rtl; -- Fim da Arquitetura da Maquina de Salgados
```

1. **Constantes:**
    a. **“S0”, “S1”, “S2”, “S3”, “S4”:** Constantes que representam os preços das escolhas 1 a 5, respectivamente.

2. **Componentes:**
    a. **“accumulator8”:** componente para um acumulador de 8bits, usado para somar os valores das moedas inseridas;
    b. **“adder8”:** componente para um somador de 8 bits, utilizado para operações de soma;
    c. **“subtractor8”:** componente para um subtrador de 8bits, usado para comparar o saldo acumulado com o preço do produto escolhido;
    d. **“mux21”:** componente para um multiplexador de 2 para 1, usado para escolher entre os preços das opções de salgado;

3. **Tipos e Sinais:**
    a. **“FSMTYPE”:** enumeração dos estados da máquina (INIT_STATE, Coin_Reception, salgado_dispensation);
    b. **“CSTATE”, “NSTATE”:** sinais que representam o estado atual e próximo da máquina;
    c. **“balance”, “price”, “price_reg”, “coins_to_return”:** sinais para valores de saldo, preço, registro do preço, e troco/retorno de moedas;
    d. **“price_choice_reg_EN”, “balance_greater”, “balance_equal”, “balance_lower”:** sinais de controle para valores;
    e. **“nRST_acc”:** sinal de reset para o acumulador;
    f. **“choice_reg”:** registro da escolha do salgado;
    g. **“unidade_reg”:** registro da quantidade de salgado selecionado;
    h. **“stock_S0_reg”, “stock_S1_reg”, “stock_S2_reg”, “stock_S3_reg”, “stock_S4_reg”, “stock_S5_reg”:** registros do estoque de cada salgado;
    i. **“V”, “c_out”, “c_in”:** sinais relacionados à entrada e saída de moedas;
    j. **“dispensation_EN”:** sinal de controle para liberar o salgado;

k. **“salgado_selecionado”:** registro do salgado selecionado pelo cliente;

4. **Funções:**
    a. **“intToSevenSeg”:** função que converte um número inteiro em um vetor de 7 bits para exibição em um display de 7 segmentos;

5. **Processos:**
    a. **“price_registration”:** processo de registro de preço;
    b. **“state_registration”:** processo de registro de estado;
    c. **“salgado_dispensation_proc”:** processo para controlar a dispensa do salgado, verificando o estoque e atualizando os sinais correspondentes;
    d. **“hex_display_updt”:** processo para atualizar os displays de sete segmentos com base no estado atual da máquina;

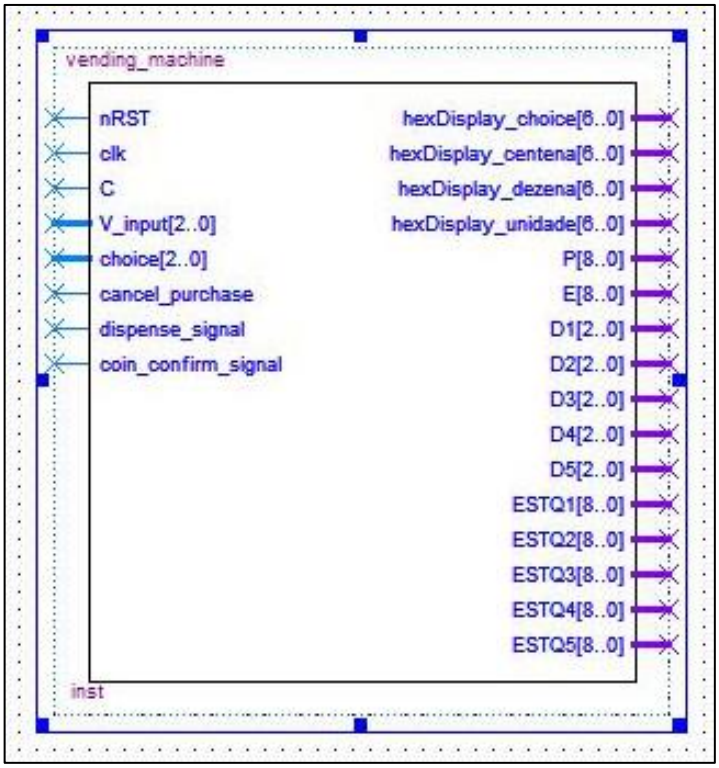
6. **Instâncias:**
    a. **“mux”:** instância do multiplexador para escolher entre os preços das opções de salgado;
    b. **“acumulator”:** instância do acumulador de 8 bits;
    c. **“comparator”:** instância do comparador de 8 bits;
    d. **“subtractor”:** instância do subtrador de 8 bits.

Para dar suporte à lógica de processamento acima, recorreu-se a componentes de terceiros, devidamente referenciados[1].

7. **Esquemático da vending_machine**

[1] Os componentes: “accumulator8”, “adder8”, “comparator_1bit”, “comparator”, “comparator8”, “mux21” e “subtractor8” foram retirados do projeto de Mohamma Niknam, conforme extensa documentação disponível neste pdf: <<https://github.com/MohammadNiknam17/vending_machine_processor/blob/master/vending_machine_processor.pdf>>.

Figura 3: Esquemático da Máquina de Venda



V. TESTES DE ONDA

Figura 4: *Waveform* configurado para testes

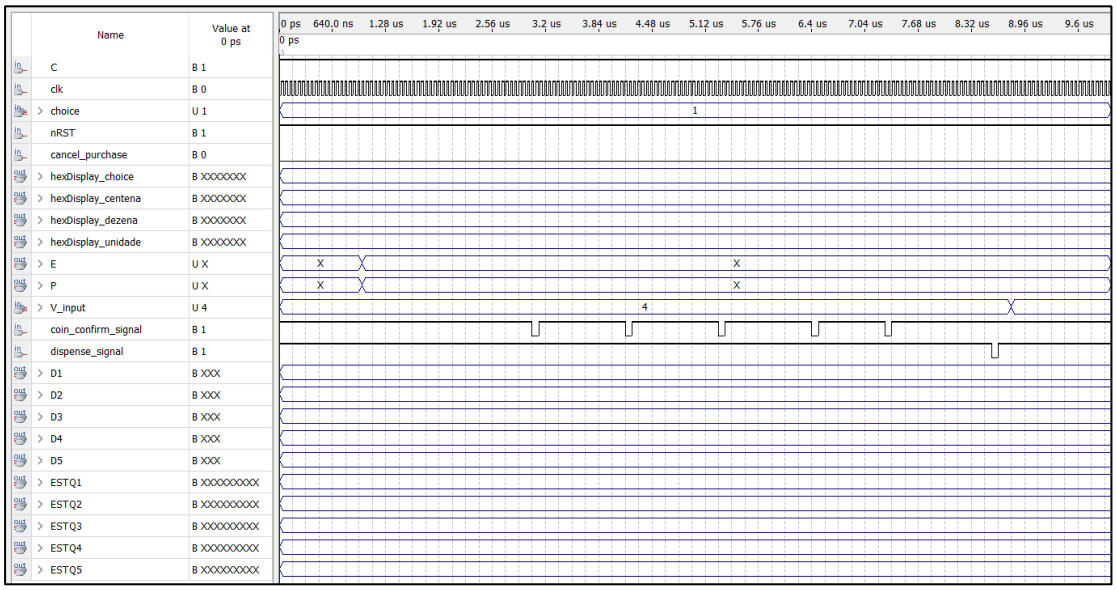


Figura 5: *Waveform* com o teste geral

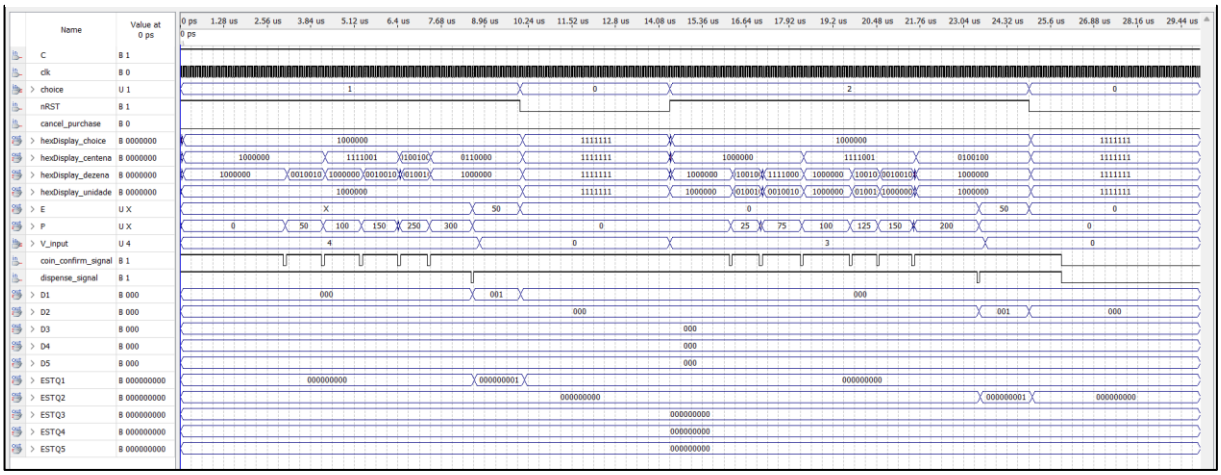


Figura 6: *Waveform* com o teste para o salgado 01

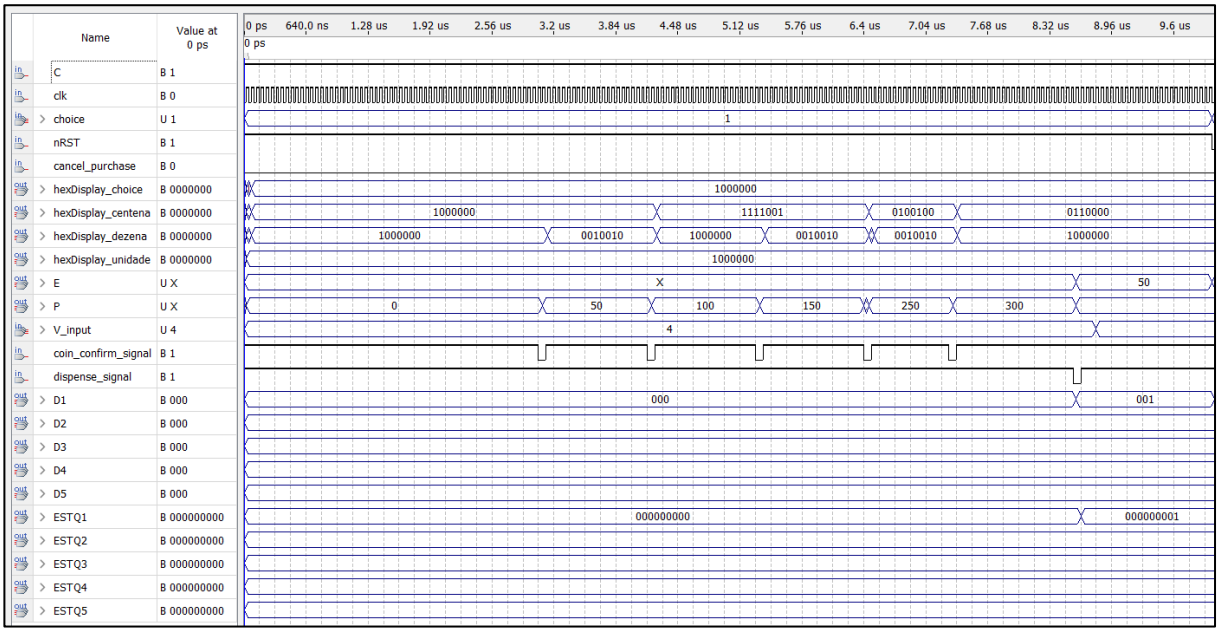
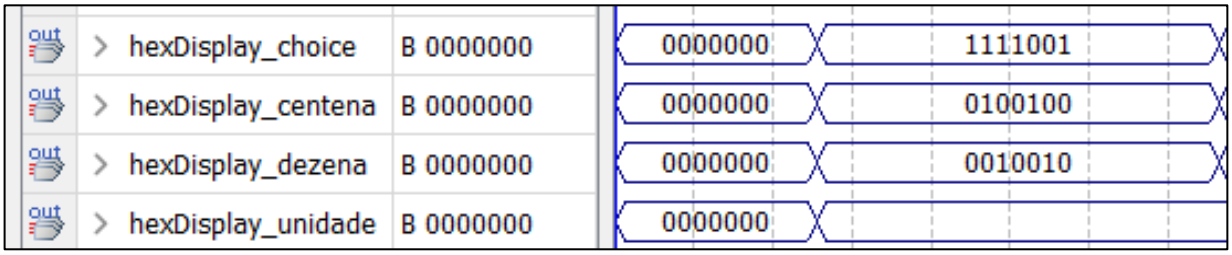


Figura 7: *Waveform* ilustrandoo display



VI. INPUTS E OUTPUTS

Tabela 2: Tabela de pins de entrada e saída

Status	From	To	Assignment Name	Value	Enabled
Ok		C	Location	PIN_L2	Yes
Ok		cancel_purchase	Location	PIN_M2	Yes
Ok		nRST	Location	PIN_M1	Yes
Ok		clk	Location	PIN_D12	Yes
Ok		choice[0]	Location	PIN_W12	Yes
Ok		choice[1]	Location	PIN_U12	Yes
Ok		choice[2]	Location	PIN_U11	Yes
Ok		V_input[0]	Location	PIN_L22	Yes
Ok		V_input[1]	Location	PIN_L21	Yes
Ok		V_input[2]	Location	PIN_M22	Yes
Ok		dispense_signal	Location	PIN_R22	Yes
Ok		coin_confirm_signal	Location	PIN_T21	Yes
Ok		hexDisplay_centena[1]	Location	PIN_G6	Yes
Ok		hexDisplay_centena[2]	Location	PIN_C2	Yes
Ok		hexDisplay_centena[3]	Location	PIN_C1	Yes
Ok		hexDisplay_centena[4]	Location	PIN_E3	Yes
Ok		hexDisplay_centena[5]	Location	PIN_E4	Yes
Ok		hexDisplay_centena[6]	Location	PIN_D3	Yes
Ok		hexDisplay_dezena[0]	Location	PIN_E1	Yes
Ok		hexDisplay_dezena[1]	Location	PIN_H6	Yes
Ok		hexDisplay_dezena[2]	Location	PIN_H5	Yes
Ok		hexDisplay_dezena[3]	Location	PIN_H4	Yes
Ok		hexDisplay_dezena[4]	Location	PIN_G3	Yes
Ok		hexDisplay_dezena[5]	Location	PIN_D2	Yes
Ok		hexDisplay_dezena[6]	Location	PIN_D1	Yes
Ok		hexDisplay_unidade[0]	Location	PIN_J2	Yes
Ok		hexDisplay_unidade[1]	Location	PIN_J1	Yes
Ok		hexDisplay_unidade[2]	Location	PIN_H2	Yes
Ok		hexDisplay_unidade[3]	Location	PIN_H1	Yes
Ok		hexDisplay_unidade[4]	Location	PIN_F2	Yes
Ok		hexDisplay_unidade[5]	Location	PIN_F1	Yes

Ok		hexDisplay_unidade[6]	Location	PIN_E2	Yes
Ok		hexDisplay_choice[0]	Location	PIN_F4	Yes
Ok		hexDisplay_choice[1]	Location	PIN_D5	Yes
Ok		hexDisplay_choice[2]	Location	PIN_D6	Yes
Ok		hexDisplay_choice[3]	Location	PIN_J4	Yes
Ok		hexDisplay_choice[4]	Location	PIN_L8	Yes
Ok		hexDisplay_choice[5]	Location	PIN_F3	Yes
Ok		hexDisplay_choice[6]	Location	PIN_D4	Yes
Ok		hexDisplay_centena[0]	Location	PIN_G5	Yes
Ok		D1[0]	Location	PIN_Y21	Yes
Ok		D2[0]	Location	PIN_Y22	Yes
Ok		D3[0]	Location	PIN_W21	Yes
Ok		D4[0]	Location	PIN_W22	Yes
Ok		D5[0]	Location	PIN_V21	Yes
Ok		ESTQ1[0]	Location	PIN_T18	Yes
Ok		ESTQ2[0]	Location	PIN_Y19	Yes
Ok		ESTQ3[0]	Location	PIN_U19	Yes
Ok		ESTQ4[0]	Location	PIN_R19	Yes
Ok		ESTQ5[0]	Location	PIN_R20	Yes

A placa FPGA Cyclone II está sendo programada para realizar operações relacionadas a uma Máquina de Vender Salgados. Os sinais de entrada incluem botões (cancel_purchase, nRST, clk, choice[0-2], V_input[0-2], dispense_signal, coin_confirm_signal), enquanto os sinais de saída abrangem displays hexadecimais, LEDs e sinais relacionados ao controle de estoque (ESTQ1-5) e escolha (hexDisplay_choice[0-6]).

Cada botão e sinal de entrada tem uma localização específica na placa FPGA, atribuída a pinos específicos (PIN_xxx). Os displays hexadecimais e LEDs também têm locais específicos atribuídos.

Os LEDs, displays e sinais de controle estão organizados para representar visualmente o estado da máquina (por meio dos displays) e para indicar ações como confirmar a compra, cancelar a compra, e outros estados específicos da máquina.

A seção "Output" indica as atribuições para as saídas da FPGA, incluindo displays hexadecimais, LEDs e sinais relacionados ao controle de estoque. A seção "Input" mostra as atribuições para os botões e sinais de entrada.

O status "Ok" indica que as atribuições foram feitas com sucesso. Essas atribuições configuram a lógica da FPGA para operar a Máquina de Vender Salgados de acordo com as especificações fornecidas.