**UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA – UESB**

**DEPARTAMENTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS – DCET**

**COLEGIADO DO CURSO DE COMPUTAÇÃO – CCCOMP**

**CIRCUITOS DIGITAIS**

**PROF. ME. MARCO ANTONIO DANTAS RAMOS**

Ademir de Jesus Reis Júnior

Cauê Rodrigues de Aguiar

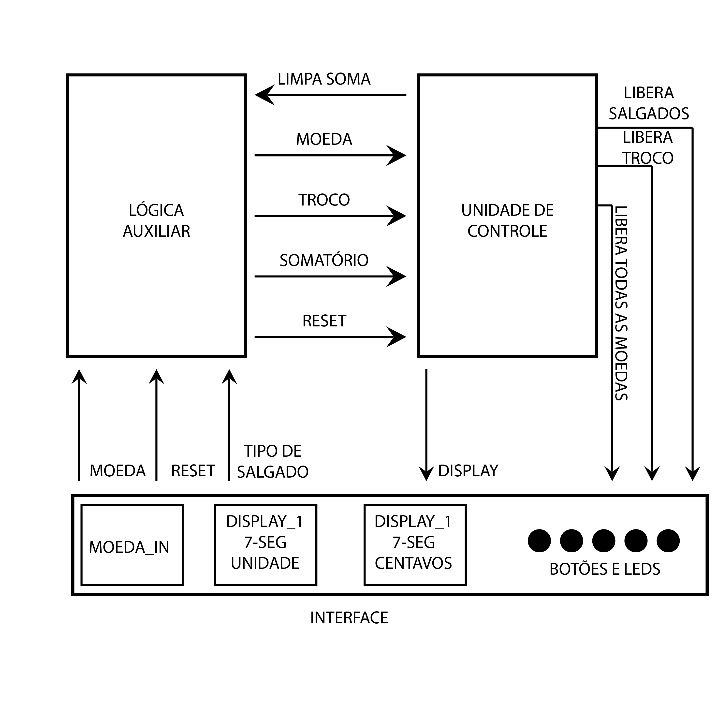
João Henrique Silva Pinto

**Documentação da Máquina de Vender Salgados**

1. **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.

1. **ESQUEMÁTICO DA MÁQUINA**



1. **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:

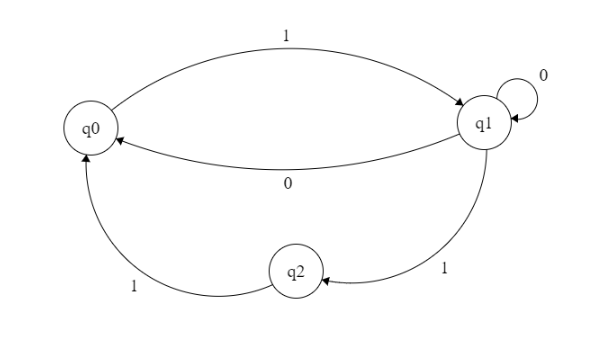
* Do estado Inicial/Seleção para o estado Inserção de Moedas, a transição ocorre quando o cliente pressiona o botão de confirmar o salgado desejado;
* Do estado Inserção de Moedas para o estado de Dispensação de Salgado, a transição ocorre quando o cliente insere moedas suficientes para comprar o salgado 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;

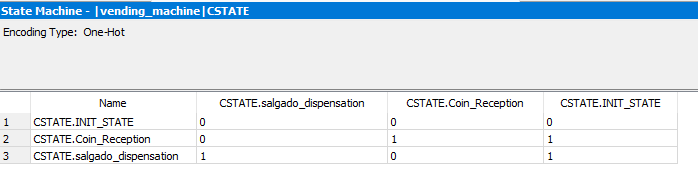
Obs.: a qualquer momento, o cliente pode desistir da comprando, pressionando o botão de cancelar, o qual retorna a máquina para o estado Inicial/Seleção, devolvendo as moedas, caso o cliente as tenha inserido.

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

1. **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:
   1. Accumulator8;
   2. Comparator8;
   3. Lpm\_divide:Div0;
   4. Lpm\_divide:Div1;
   5. Lpm\_divide:Mod0;
   6. Lpm\_divide:Mod1
   7. Mux21:mux;
   8. 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”):
   1. “nRST”: sinal de reset para iniciar uma nova compra;
   2. “clk”: sinal de clock padrão;
   3. “C”: sensor de moedas, onde “1” indica a inserção de uma moeda;
   4. “V\_input”: entrada de valores de moedas representada como um vetor de lógica padrão de 3 bits;
   5. “choice”: vetor de lógica padrão de 3 bits representando a escolha do salgado;
   6. “cancel\_purchase”: opção de cancelar a compra;
   7. “dispense\_signal”: sinal para confirmar a dispensa do salgado escolhido;
   8. “coin\_confirm\_signal”: sinal para confirmar a adição de uma moeda.
2. Portas de saída (“out”):
   1. “hexDisplay\_choice”: saída para o display que mostra a opção do salgado escolhido;
   2. “hexDisplay\_centena”, “hexDisplay\_dezena”, “hexDisplay\_unidade”: saídas para displays que representam o valor em dinheiro nas centenas, dezenas e unidades, respectivamente;
   3. “P”: acumulador de moedas;
   4. “E”: troco/retorno de moedas;
   5. “D1”, “D2”, “D3”, “D4”, “D5”: sinais indicando se os salgados 1 a 5 estão liberados ("000" indica não liberado);
   6. **“ESTQ1”**, “**ESTQ2”**, “**ESTQ3”**, “**ESTQ4”**, “**ESTQ5”**: avisos de quantidade em estoque para os salgados 1 a 5.
3. Observações adicionais:
   1. Os displays são representados como vetores de lógica padrão de 7 bits;
   2. 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:
   1. “S0”, “S1”, “S2”, “S3”, “S4”: Constantes que representam os preços das escolhas 1 a 5, respectivamente.
2. Componentes:
   1. “accumulator8”: componente para um acumulador de 8bits, usado para somar os valores das moedas inseridas;
   2. “adder8”: componente para um somador de 8 bits, utilizado para operações de soma;
   3. “subtractor8”: componente para um subtrador de 8bits, usado para comparar o saldo acumulado com o preço do produto escolhido;
   4. “mux21”: componente para um multiplexador de 2 para 1, usado para escolher entre os preços das opções de salgado;
3. Tipos e Sinais:
   1. “FSMTYPE”: enumeração dos estados da máquina (INIT\_STATE, Coin\_Reception, salgado\_dispensation);
   2. “CSTATE”, “NSTATE”: sinais que representam o estado atual e próximo da máquina;
   3. “balance”, “price”, “price\_reg”, “coins\_to\_return”: sinais para valores de saldo, preço, registro do preço, e troco/retorno de moedas;
   4. “price\_choice\_reg\_EN”, “balance\_greater”, “balance\_equal”, “balance\_lower”: sinais de controle para valores;
   5. “nRST\_acc”: sinal de reset para o acumulador;
   6. “choice\_reg”: registro da escolha do salgado;
   7. “unidade\_reg”: registro da quantidade de salgado selecionado;
   8. “stock\_S0\_reg”, “stock\_S1\_reg”, “stock\_S2\_reg”, “stock\_S3\_reg”, “stock\_S4\_reg”, “stock\_S5\_reg”: registros do estoque de cada salgado;
   9. “V”, “c\_out”, “c\_in”: sinais relacionados à entrada e saída de moedas;
   10. “dispensation\_EN”: sinal de controle para liberar o salgado;
   11. “salgado\_selecionado”: registro do salgado selecionado pelo cliente;
4. Funções:
   1. “intToSevenSeg”: função que converte um número inteiro em um vetor de 7 bits para exibição em um display de 7 segmentos;
5. Processos:
   1. “price\_registration”: processo de registro de preço;
   2. “state\_registration”: processo de registro de estado;
   3. “salgado\_dispensation\_proc”: processo para controlar a dispensa do salgado, verificando o estoque e atualizando os sinais correspondentes;
   4. “hex\_display\_updt”: processo para atualizar os displays de sete segmentos com base no estado atual da máquina;
6. Instâncias:
   1. “mux”: instância do multiplexador para escolher entre os preços das opções de salgado;
   2. “acumulator”: instância do acumulador de 8 bits;
   3. “comparator”: instância do comparador de 8 bits;
   4. “subtractor”: instância do subtrador de 8 bits.
7. **TESTES FEITOS**