Centro de Tecnologia

Departamento de Engenharia Elétrica

Disciplina: Sistemas Digitais

Professor: Samaherni Morais Dias

RELATÓRIO

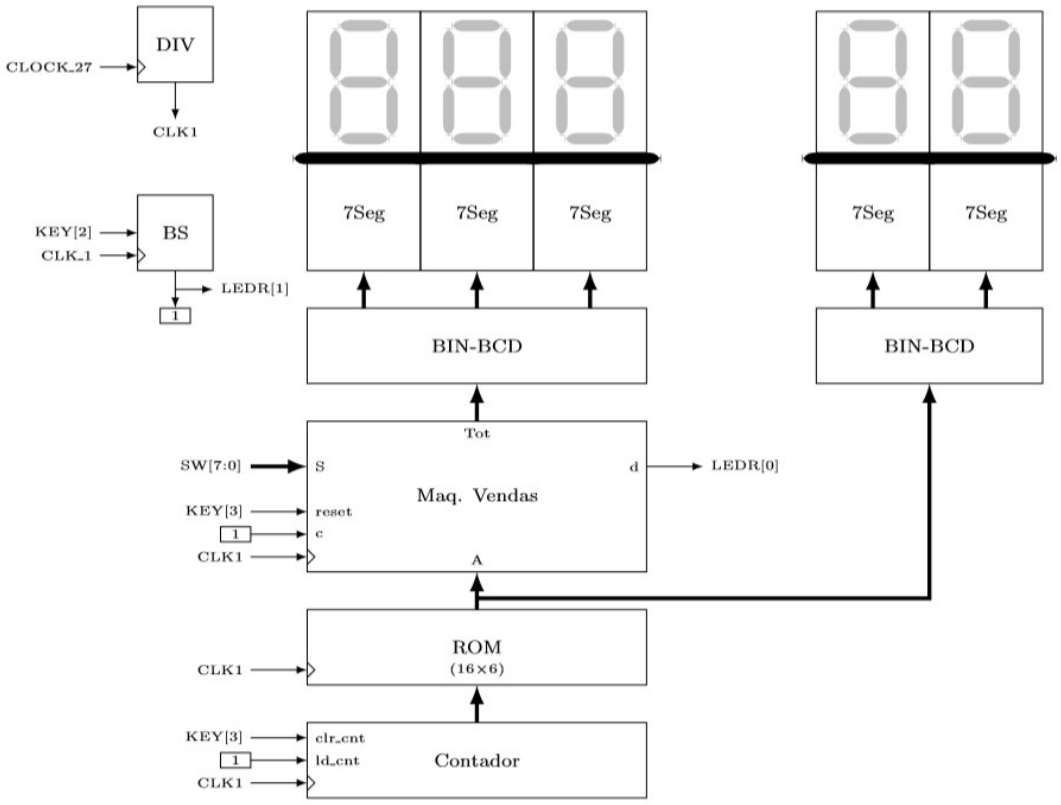
VINÍCIUS NASCIMENTO DE AZEVEDO

Natal , RN - Março de 2020

INTRODUÇÃO

A problemática apresentada para esta atividade consiste em desenvolver um sistema digital para uma máquina de vendas. O sistema consiste em fornecer à máquina um valor de 8 bits correspondente ao preço do produto, um botão onde informa que uma moeda foi inserida e uma memória ROM que mostra qual o valor da moeda inserida. Quando o valor do somatório das moedas for maior ou igual ao preço do produto, este é liberado (no nosso caso um LED é acesso). A máquina também deverá contar com um botão reset. O botão da moeda deverá advir de um bloco que sincroniza com o clock para evitar múltiplas moedas em um único pressionamento do botão. A máquina será implementada no kit DE2.

O sistema completo está detalhado na imagem abaixo:

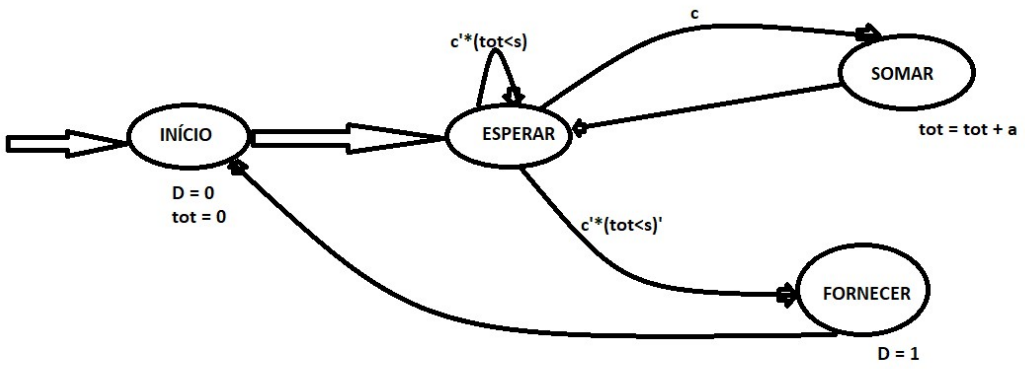


Os displays de sete segmentos HEX[2-0] mostrarão o valor armazenado no registrador da máquina de vendas. Os displays HEX[5-4] mostrarão o valor presenta da memória.

SOLUÇÃO

A solução passa pela máquina de estados da máquina de vendas. Os blocos anexos como o bin-bcd, o display, a ROM, contador, e o DIV são códigos anexos que serão incrementados no programa a partir de port map. Como esses códigos já foram elaborados anteriormente ou repassados pelo professor, não nos aprofundaremos neles. Todos os códigos estão ao final deste relatório.

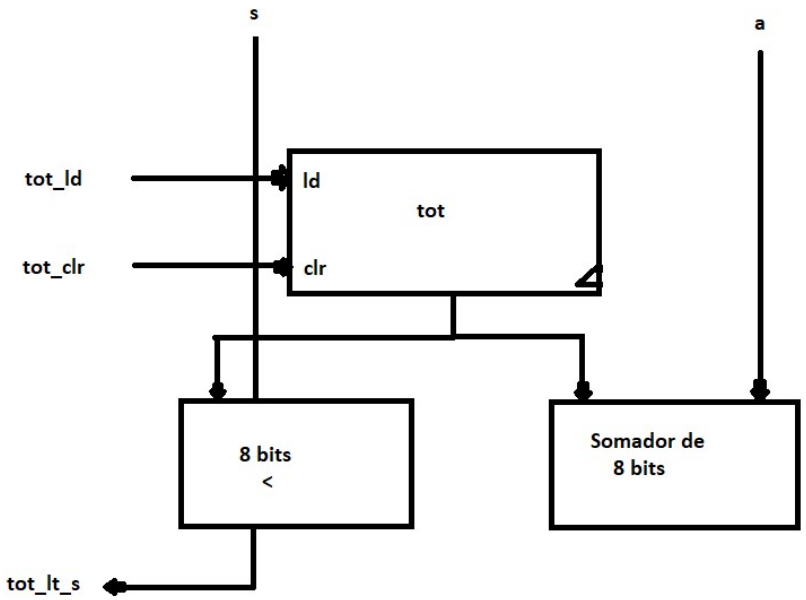
A máquina de estados de alto nível da máquina de vendas se estrutura como a imagem abaixo. Sendo 'D' a indicação de que o produto foi liberado, 'tot' o registrador da máquina, 'a' o valor da moeda inserida e 's' o preço do produto.



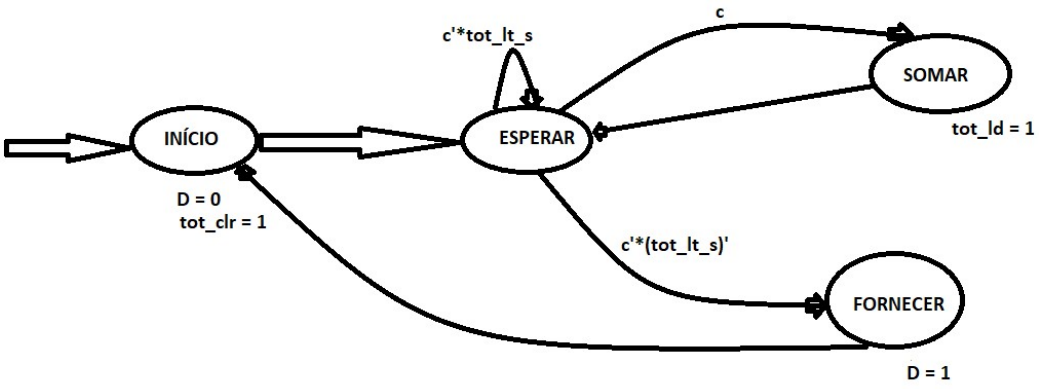
Há o primeiro estado "INÍCIO" onde zeramos o registrador, no próximo pulso de clock, o estado "ESPERAR" fica em modo de espera enquanto o botão não for pressionado e o acumulado no registrador for menor que o preço do produto. Ao apertar o botão avançamos ao estado "SOMAR" onde a soma com o valor inserido é realizado e armazenado no registrador. Voltando ao estado anterior, se o valor guardado no registrador for maior ou igual ao preço avança-se ao estado "FORNECER" onde o produto é liberado fazendo D=1 e voltando ao início para resetar a máquina.

Com a máquina de estados de alto nível podemos obter o datapath e a MDE do bloco de controle.

Datapath



MDE Bloco de Controle



Fazendo as operações sobre a máquina de estado para obter as equações de cada sinal e registrador do bloco de controle chegamos as seguintes equações, usando os registradores de estado s1 e s0. Chamamos no nosso código tot_ld de reg_c_ld e tot_clr de reg_c_clr:

n1 <= (not(s1) and s0 and not(btn) and tot_lt) or (not(s1) and s0 and btn);

n0 <= (not(s1) and s0 and not(btn)) or not(s0);

reg_c_ld <= s1 and not(s0);

reg_c_clr <= not(not(s1) and not(s0));

d <= s1 and s0;

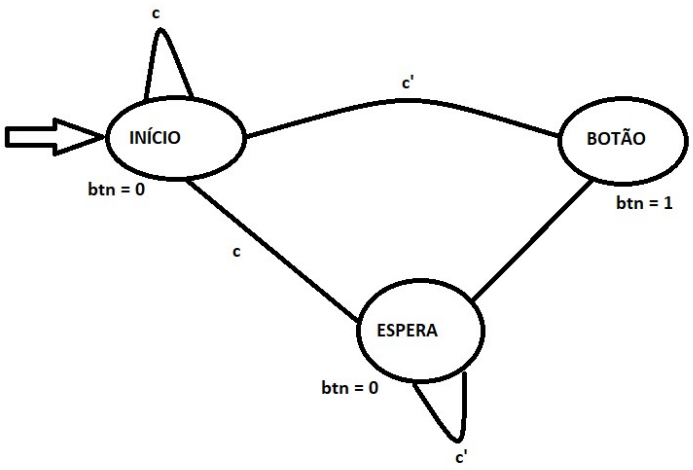
Obtivemos, então, a máquina de vendas.

Foi, também, pedido um botão ‘reset’ que deve reiniciar todo o sistema. A solução é simples, todos os flip flops D usados no programa receberão a variável reset no seu clear, realizando um clear assíncrono. Quando o botão reset for apertado todos os flip flops armazenarão 0 levando as máquinas de estado para o estado inicial e levando o registrador ‘tot’ a ‘0’.

O bloco DIV já nos foi fornecido pelo professor.

Vamos agora focar no bloco BS, o botão sincronizado, que tem função de impedir múltiplas entradas da moeda apertando somente uma vez o botão. Para criar a máquina de estados precisamos lembrar que o kit DE2 tem a lógica invertida, ou seja, quando um botão é pressionado ele vai a ‘0’ e não a ‘1’.

MDE Botão Sincronizado



Realizando os procedimentos para se obter as equações de n1, n0 e btn, usando s1 e s0 como registradores de estado, obtemos:

n1 <= s0 or (not(c) and s1);

n0 <= not(s1) and not(s0) and not(c);

btn <= n0;

Para o contador descrito no sistema, nada mais é do que um incrementador que vai de 0 a 63 e depois reseta. O contador enviará para a memória o endereço de onde está o valor da moeda. Para isso usaremos o mesmo esquema do datapath

da máquina de vendas, porém com ‘a’ sendo igual a 1, já que incrementaremos de um em um, e o ‘s’ igual a 63. Quando atingir 63, o registrador é resetado.

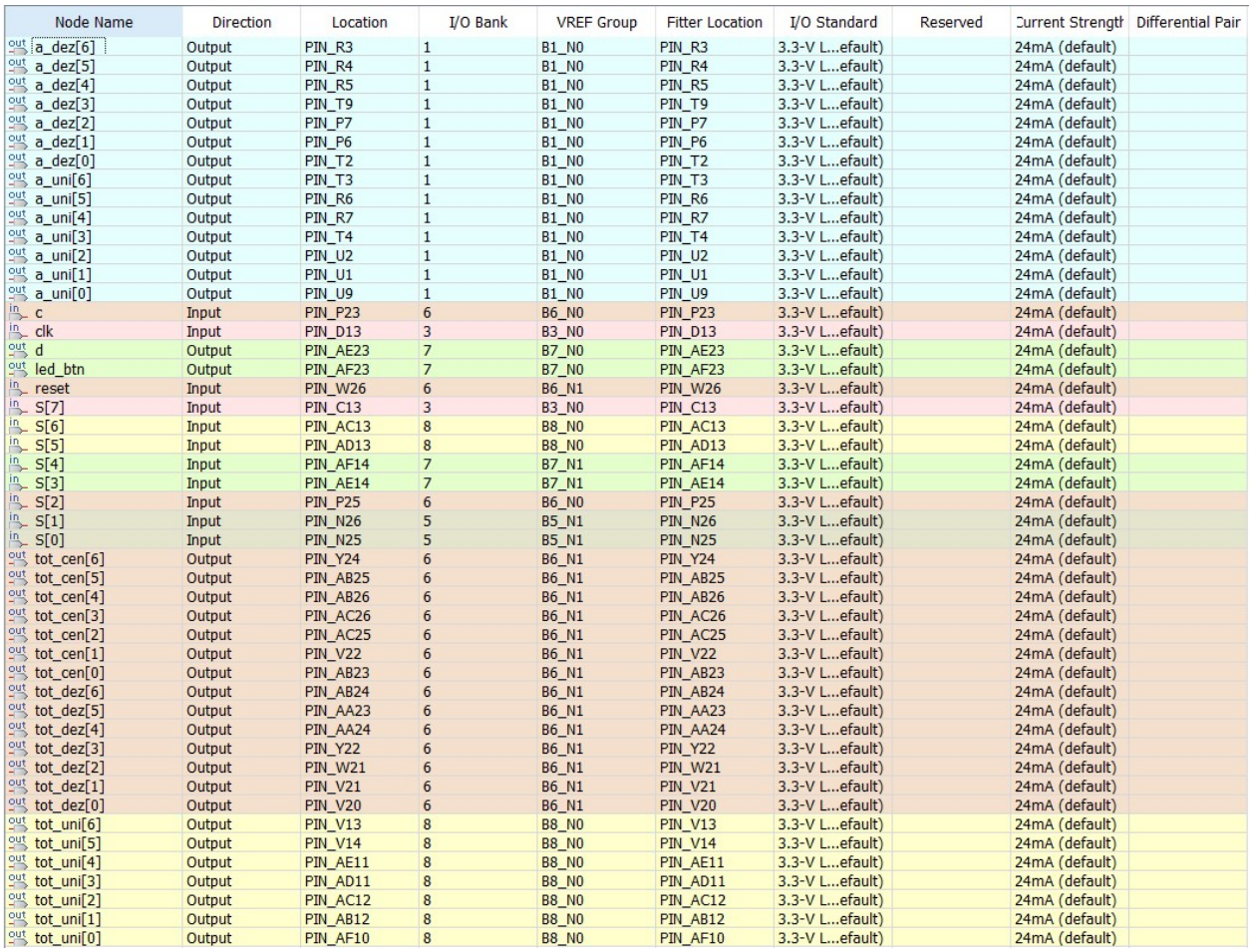
Um problema que foi notado se referia a um overflow. Como todos os componentes são de 8 bits, quando o somador do datapath atingia um valor maior que 8 bits não se fazia a comparação e, portanto, o produto não era liberado. Para resolver esse problema foi usado o carry out do somador, que foi armazenado em um flip flop, onde o carregamento se daria ao apertar o botão, e a saída desse flip flop faz uma OR com a saída do comparador do datapath. Com isso, quando carry out for ‘1’ significa que o somador já atingiu um valor de 9 bits e, portanto, ele dele liberar o produto e zerar o registrador.

Os códigos dos blocos restantes já foram feitos anteriormente, são básicos, por isso nós não comentaremos como chegamos a eles.

Com todos os códigos, juntaremos todos em um projeto específico utilizando a ferramenta port map, como se vê nos códigos ao final do relatório.

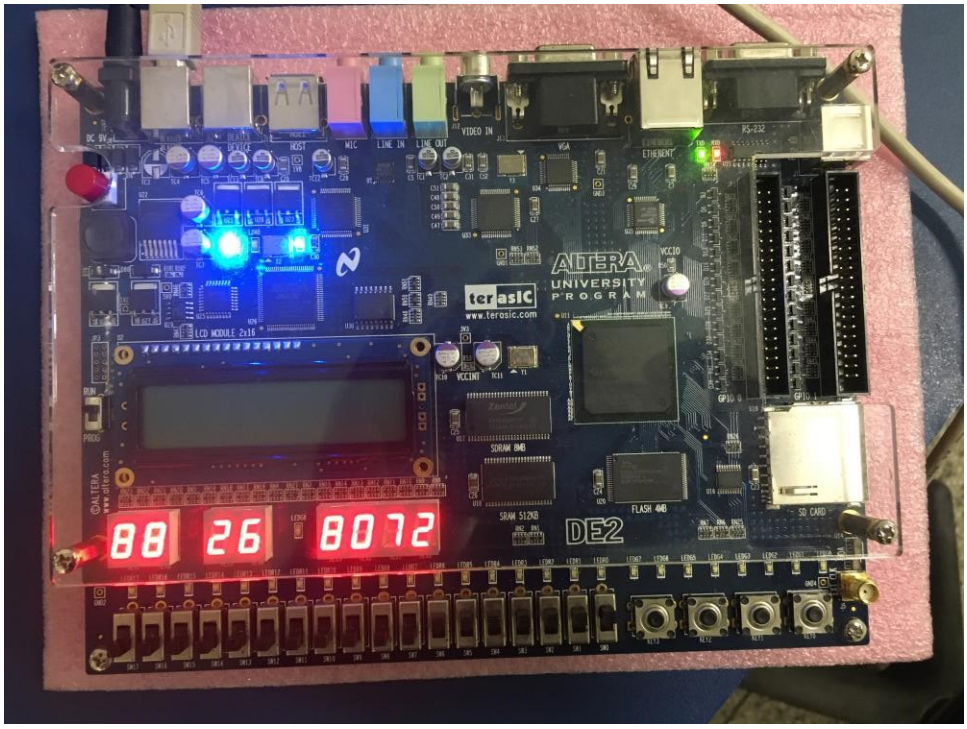
RESULTADOS

Para simular o projeto no kit DE2, usamos a pinagem abaixo, seguindo os pinos pedidos na descrição da atividade.



Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strengtl	Differential Pair
a_dez[6]	Output	PIN_R3	1	B1_N0	PIN_R3	3.3-V L...efault)		24mA (default)	
a_dez[5]	Output	PIN_R4	1	B1_N0	PIN_R4	3.3-V L...efault)		24mA (default)	
a_dez[4]	Output	PIN_R5	1	B1_N0	PIN_R5	3.3-V L...efault)		24mA (default)	
a_dez[3]	Output	PIN_T9	1	B1_N0	PIN_T9	3.3-V L...efault)		24mA (default)	
a_dez[2]	Output	PIN_P7	1	B1_N0	PIN_P7	3.3-V L...efault)		24mA (default)	
a_dez[1]	Output	PIN_P6	1	B1_N0	PIN_P6	3.3-V L...efault)		24mA (default)	
a_dez[0]	Output	PIN_T2	1	B1_N0	PIN_T2	3.3-V L...efault)		24mA (default)	
a_uni[6]	Output	PIN_T3	1	B1_N0	PIN_T3	3.3-V L...efault)		24mA (default)	
a_uni[5]	Output	PIN_R6	1	B1_N0	PIN_R6	3.3-V L...efault)		24mA (default)	
a_uni[4]	Output	PIN_R7	1	B1_N0	PIN_R7	3.3-V L...efault)		24mA (default)	
a_uni[3]	Output	PIN_T4	1	B1_N0	PIN_T4	3.3-V L...efault)		24mA (default)	
a_uni[2]	Output	PIN_U2	1	B1_N0	PIN_U2	3.3-V L...efault)		24mA (default)	
a_uni[1]	Output	PIN_U1	1	B1_N0	PIN_U1	3.3-V L...efault)		24mA (default)	
a_uni[0]	Output	PIN_U9	1	B1_N0	PIN_U9	3.3-V L...efault)		24mA (default)	
c	Input	PIN_P23	6	B6_N0	PIN_P23	3.3-V L...efault)		24mA (default)	
clk	Input	PIN_D13	3	B3_N0	PIN_D13	3.3-V L...efault)		24mA (default)	
d	Output	PIN_AE23	7	B7_N0	PIN_AE23	3.3-V L...efault)		24mA (default)	
led_btn	Output	PIN_AF23	7	B7_N0	PIN_AF23	3.3-V L...efault)		24mA (default)	
reset	Input	PIN_W26	6	B6_N1	PIN_W26	3.3-V L...efault)		24mA (default)	
S[7]	Input	PIN_C13	3	B3_N0	PIN_C13	3.3-V L...efault)		24mA (default)	
S[6]	Input	PIN_AC13	8	B8_N0	PIN_AC13	3.3-V L...efault)		24mA (default)	
S[5]	Input	PIN_AD13	8	B8_N0	PIN_AD13	3.3-V L...efault)		24mA (default)	
S[4]	Input	PIN_AF14	7	B7_N1	PIN_AF14	3.3-V L...efault)		24mA (default)	
S[3]	Input	PIN_AE14	7	B7_N1	PIN_AE14	3.3-V L...efault)		24mA (default)	
S[2]	Input	PIN_P25	6	B6_N0	PIN_P25	3.3-V L...efault)		24mA (default)	
S[1]	Input	PIN_N26	5	B5_N1	PIN_N26	3.3-V L...efault)		24mA (default)	
S[0]	Input	PIN_N25	5	B5_N1	PIN_N25	3.3-V L...efault)		24mA (default)	
tot_cen[6]	Output	PIN_Y24	6	B6_N1	PIN_Y24	3.3-V L...efault)		24mA (default)	
tot_cen[5]	Output	PIN_AB25	6	B6_N1	PIN_AB25	3.3-V L...efault)		24mA (default)	
tot_cen[4]	Output	PIN_AB26	6	B6_N1	PIN_AB26	3.3-V L...efault)		24mA (default)	
tot_cen[3]	Output	PIN_AC26	6	B6_N1	PIN_AC26	3.3-V L...efault)		24mA (default)	
tot_cen[2]	Output	PIN_AC25	6	B6_N1	PIN_AC25	3.3-V L...efault)		24mA (default)	
tot_cen[1]	Output	PIN_V22	6	B6_N1	PIN_V22	3.3-V L...efault)		24mA (default)	
tot_cen[0]	Output	PIN_AB23	6	B6_N1	PIN_AB23	3.3-V L...efault)		24mA (default)	
tot_dez[6]	Output	PIN_AB24	6	B6_N1	PIN_AB24	3.3-V L...efault)		24mA (default)	
tot_dez[5]	Output	PIN_AA23	6	B6_N1	PIN_AA23	3.3-V L...efault)		24mA (default)	
tot_dez[4]	Output	PIN_AA24	6	B6_N1	PIN_AA24	3.3-V L...efault)		24mA (default)	
tot_dez[3]	Output	PIN_Y22	6	B6_N1	PIN_Y22	3.3-V L...efault)		24mA (default)	
tot_dez[2]	Output	PIN_W21	6	B6_N1	PIN_W21	3.3-V L...efault)		24mA (default)	
tot_dez[1]	Output	PIN_V21	6	B6_N1	PIN_V21	3.3-V L...efault)		24mA (default)	
tot_dez[0]	Output	PIN_V20	6	B6_N1	PIN_V20	3.3-V L...efault)		24mA (default)	
tot_uni[6]	Output	PIN_V13	8	B8_N0	PIN_V13	3.3-V L...efault)		24mA (default)	
tot_uni[5]	Output	PIN_V14	8	B8_N0	PIN_V14	3.3-V L...efault)		24mA (default)	
tot_uni[4]	Output	PIN_AE11	8	B8_N0	PIN_AE11	3.3-V L...efault)		24mA (default)	
tot_uni[3]	Output	PIN_AD11	8	B8_N0	PIN_AD11	3.3-V L...efault)		24mA (default)	
tot_uni[2]	Output	PIN_AC12	8	B8_N0	PIN_AC12	3.3-V L...efault)		24mA (default)	
tot_uni[1]	Output	PIN_AB12	8	B8_N0	PIN_AB12	3.3-V L...efault)		24mA (default)	
tot_uni[0]	Output	PIN_AF10	8	B8_N0	PIN_AF10	3.3-V L...efault)		24mA (default)	

Para o clock, utilizamos o divisor de clock fornecido com um valor de 10Hz. Na placa, o sistema ficou da seguinte forma:



Natal , RN - Março de 2020

CONCLUSÕES

O código do sistema digital da máquina de vendas funcionou da forma desejada. Produzindo os resultados esperados e sem nenhum erro verificado.

A construção do código foi facilitada visto que alguns componentes utilizados já tinham sido elaborados na matéria de Circuitos Digitais.

Natal , RN - Março de 2020

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity MaquinaDeVendas is
5      port (c, clk, reset : in std_logic;
6             S : in std_logic_vector(7 downto 0);
7             d, led_btn : out std_logic;
8             tot_uni, tot_dez, tot_cen, a_uni, a_dez : out std_logic_vector(6 downto 0));
9   end MaquinaDeVendas;
10
11  architecture ckt of MaquinaDeVendas is
12
13   component datapath is
14     port (S,A : in std_logic_vector(7 downto 0);
15            clk2, reg_ld, reg_clr: in std_logic;
16              reg_mq : out std_logic;
17              tot2 : out std_logic_vector(7 downto 0));
18   end component;
19
20   component ffd is
21      port (clk,D,P,C : in  std_logic;
22                    q : out std_logic);
23   end component;
24
25   component conv_bin_bcd is
26      port(SW : in std_logic_vector(7 downto 0);
27           HEX2 : out std_logic_vector(6 downto 0);
28           HEX1 : out std_logic_vector(6 downto 0);
29           HEX0 : out std_logic_vector(6 downto 0));
30   end component;
31
32   component ROM is
33      port(address : in std_logic_vector(5 downto 0);
34           clock : in std_logic;
35           q : out std_logic_vector(7 downto 0));
36   end component;
37
38   component Contador is
39    port (ld_cnt, clk, clr_cnt : in std_logic;
40          Q : out std_logic_vector (5 downto 0));
41   end component;
42
43   component button is
44    port (C, clk, reset : in std_logic;
45          btn : out std_logic);
46   end component;
47
48   component CLK_Div is
49    port (clk_in : in std_logic;
50           clk_out : out std_logic);
51   end component;
52
53
54   signal reg_c_ld, reg_c_clr, n1, n0, s1, s0, tot_lt, not_reset, btn, clk_aux : std_logic;
55   signal tot_aux, A : std_logic_vector(7 downto 0);
56   signal cnt_out : std_logic_vector(5 downto 0);
57   signal a_cen: std_logic_vector(6 downto 0);
58
59
60   begin
61
62    not_reset <= not(reset);
63
64    botao : button port map(C=>c,clk=>clk_aux,reset=>reset,btn=>btn);
65    comando : datapath port map(S=>S,A=>A,clk2=>clk_aux,reg_ld=>reg_c_ld,reg_clr=>reg_c_clr,
    reg_mq=>tot_lt,tot2=>tot_aux);
```

```
66      contar : Contador port map(ld_cnt=>btn,clk=>clk_aux,clr_cnt=>reset,Q=>cnt_out);
67      memoria : ROM port map(address=>cnt_out,clock=>clk_aux,q=>A);
68      clkk : CLK_Div port map(clk_in=>clk,clk_out=>clk_aux);
69
70
71
72      s11 : ffd port map(clk=>clk_aux,D=>n1,P=>'1',C=>reset,q=>s1);
73      s00 : ffd port map(clk=>clk_aux,D=>n0,P=>'1',C=>reset,q=>s0);
74
75      n1 <= (not(s1) and s0 and not(btn) and tot_lt) or (not(s1) and s0 and btn);
76      n0 <= (not(s1) and s0 and not(btn)) or not(s0);
77      reg_c_ld <= s1 and not(s0);
78      reg_c_clr <= not(not(s1) and not(s0));
79      d <= s1 and s0;
80      led_btn <= btn;
81
82      display_tot : conv_bin_bcd port map(SW=>tot_aux,HEX2=>tot_cen,HEX1=>tot_dez,HEX0=>
    tot_uni);
83      display_A : conv_bin_bcd port map(SW=>A,HEX2=>a_cen,HEX1=>a_dez,HEX0=>a_uni);
84
85      end ckt;
86
87
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity datapath is
5      port (S,A : in std_logic_vector(7 downto 0);
6             clk2, reg_ld, reg_clr: in std_logic;
7               reg_mq : out std_logic;
8               tot2 : out std_logic_vector(7 downto 0));
9   end datapath;
10
11  architecture ckt of datapath is
12
13   component Registrador8bits is
14     port (X : in  std_logic_vector (7 downto 0);
15           ld, clr, clk, cout2 : in std_logic;
16           coutc : out std_logic;
17           Q : out std_logic_vector (7 downto 0));
18   end component;
19
20   component ComparadorMagnitude8bits is
21     port( X, Y : in std_logic_vector(7 downto 0);
22           Q : out std_logic);
23   end component;
24
25   component Somador8bits is
26     port ( X : in  std_logic_vector(7 downto 0);
27              Y : in  std_logic_vector(7 downto 0);
28              ci : in std_logic;
29               Q : out std_logic_vector(7 downto 0);
30               cout : out std_logic);
31   end component;
32
33   signal reg_out, soma_out : std_logic_vector(7 downto 0);
34   signal gb, comp_out, aux_clr, aux_gb : std_logic;
35
36   begin
37
38   registrar : Registrador8bits port map(X=>soma_out,ld=>reg_ld,clr=>reg_clr,clk=>clk2,cout2
    =>gb,coutc=>aux_gb,Q=>reg_out);
39
40   somar : Somador8bits port map(X=>A,Y=>reg_out,ci=>'0',Q=>soma_out,cout=>gb);
41
42   comp : ComparadorMagnitude8bits port map(X=>S,Y=>reg_out,Q=>comp_out);
43
44
45  reg_mq <= comp_out or aux_gb;
46  tot2(0) <= reg_out(0);
47  tot2(1) <= reg_out(1);
48  tot2(2) <= reg_out(2);
49  tot2(3) <= reg_out(3);
50  tot2(4) <= reg_out(4);
51  tot2(5) <= reg_out(5);
52  tot2(6) <= reg_out(6);
53  tot2(7) <= reg_out(7);
54
55  end ckt;
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity button is
5     port (C, clk, reset : in std_logic;
6           btn : out std_logic);
7   end button;
8
9   architecture ckt of button is
10
11   component ffd is
12      port (clk,D,P,C : in  std_logic;
13                    q : out std_logic);
14   end component;
15
16   signal s1, s0, n1, n0 : std_logic;
17
18   begin
19
20    s11 : ffd port map(clk=>clk,D=>n1,P=>'1',C=>reset,q=>s1);
21    s00 : ffd port map(clk=>clk,D=>n0,P=>'1',C=>reset,q=>s0);
22
23    n1 <= s0 or (not(c) and s1);
24    n0 <= not(s1) and not(s0) and not(c);
25    btn <= n0;
26
27  end ckt;
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity CLK_Div is
  port (clk_in : in std_logic;
         clk_out : out std_logic);
end CLK_Div;

architecture logica of CLK_Div is

  signal ax : std_logic;

  begin
    process(clk_in)
       variable cnt: integer range 0 to 1350000 := 0;
      begin
        if (rising_edge(clk_in)) then
           if (cnt = 1350000) then
              cnt:=0;
              ax <= not ax;
           else
              cnt:=cnt+1;
           end if;
        end if;
    end process;
    clk_out<=ax;
end logica;

```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4
5   entity Registrador8bits is
6
7     port (X : in  std_logic_vector (7 downto 0);
8
9           ld, clr, clk, cout2 : in std_logic;
10
11          coutc : out std_logic;
12          Q : out std_logic_vector (7 downto 0));
13
14  end Registrador8bits;
15
16
17
18  architecture ckt of Registrador8bits is
19
20
21
22    component ffd is
23
24      port (clk,D,P,C : in  std_logic;
25
26                    q : out std_logic);
27
28    end component;
29
30
31
32    signal mux, aux : std_logic_vector(8 downto 0);
33    signal co : std_logic;
34
35
36
37    begin
38
39
40     mux(8) <= (cout2 and ld) or (co and not(ld));
41
42     mux(7) <= (X(7) and ld) or (aux(7) and not(ld));
43
44     mux(6) <= (X(6) and ld) or (aux(6) and not(ld));
45
46     mux(5) <= (X(5) and ld) or (aux(5) and not(ld));
47
48     mux(4) <= (X(4) and ld) or (aux(4) and not(ld));
49
50     mux(3) <= (X(3) and ld) or (aux(3) and not(ld));
51
52     mux(2) <= (X(2) and ld) or (aux(2) and not(ld));
53
54     mux(1) <= (X(1) and ld) or (aux(1) and not(ld));
55
56     mux(0) <= (X(0) and ld) or (aux(0) and not(ld));
57
58
59     ffd7 : ffd port map (clk => clk, D => mux(7), P => '1', C => clr, q => aux(7));
60
61     ffd6 : ffd port map (clk => clk, D => mux(6), P => '1', C => clr, q => aux(6));
62
63     ffd5 : ffd port map (clk => clk, D => mux(5), P => '1', C => clr, q => aux(5));
64
65     ffd4 : ffd port map (clk => clk, D => mux(4), P => '1', C => clr, q => aux(4));
66
```

```
67      ffd3 : ffd port map (clk => clk, D => mux(3), P => '1', C => clr, q => aux(3));
68
69      ffd2 : ffd port map (clk => clk, D => mux(2), P => '1', C => clr, q => aux(2));
70
71      ffd1 : ffd port map (clk => clk, D => mux(1), P => '1', C => clr, q => aux(1));
72
73      ffd0 : ffd port map (clk => clk, D => mux(0), P => '1', C => clr, q => aux(0));
74
75
76      ffdgb : ffd port map (clk => clk, D => mux(8), P => '1', C => clr, q => co);
77
78      Q(7) <= aux(7);
79
80      Q(6) <= aux(6);
81
82      Q(5) <= aux(5);
83
84      Q(4) <= aux(4);
85
86      Q(3) <= aux(3);
87
88      Q(2) <= aux(2);
89
90      Q(1) <= aux(1);
91
92      Q(0) <= aux(0);
93
94      coutc <= co;
95
96
97
98  end ckt;
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4
5
6   entity Somador8bits is
7
8     port ( X, Y : in  std_logic_vector(7 downto 0);
9
10             ci : in std_logic;
11
12              Q : out std_logic_vector(7 downto 0);
13
14              cout : out std_logic);
15
16  end Somador8bits;
17
18
19
20  architecture ckt of Somador8bits is
21
22
23
24    signal co : std_logic_vector (7 downto 0);
25
26
27
28    begin
29
30
31
32      co(0) <= ci;
33
34      Q(0)  <=  Y(0) xor X(0) xor co(0);
35
36      co(1) <= (Y(0) and co(0)) or (X(0) and co(0)) or (X(0) and Y(0));
37
38      Q(1)  <=  Y(1) xor X(1) xor co(1);
39
40      co(2) <= (Y(1) and co(1)) or (X(1) and co(1)) or (X(1) and Y(1));
41
42      Q(2)  <=  Y(2) xor X(2) xor co(2);
43
44      co(3) <= (Y(2) and co(2)) or (X(2) and co(2)) or (X(2) and Y(2));
45
46      Q(3)  <=  Y(3) xor X(3) xor co(3);
47
48      co(4) <= (Y(3) and co(3)) or (X(3) and co(3)) or (X(3) and Y(3));
49
50      Q(4)  <=  Y(4) xor X(4) xor co(4);
51
52      co(5) <= (Y(4) and co(4)) or (X(4) and co(4)) or (X(4) and Y(4));
53
54      Q(5)  <=  Y(5) xor X(5) xor co(5);
55
56      co(6) <= (Y(5) and co(5)) or (X(5) and co(5)) or (X(5) and Y(5));
57
58      Q(6)  <=  Y(6) xor X(6) xor co(6);
59
60      co(7) <= (Y(6) and co(6)) or (X(6) and co(6)) or (X(6) and Y(6));
61
62      Q(7)  <=  Y(7) xor X(7) xor co(7);
63
64      cout <= (Y(7) and co(7)) or (X(7) and co(7)) or (X(7) and Y(7));
65
66
```

```
67
68    end ckt;
69
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4
5
6
7   entity ComparadorMagnitude8bits is
8
9     port( X, Y : in std_logic_vector(7 downto 0);
10
11          Q : out std_logic);
12
13  end ComparadorMagnitude8bits;
14
15
16
17  architecture ckt of ComparadorMagnitude8bits is
18
19
20
21  signal v,f : std_logic_vector(7 downto 0);
22
23  signal igual,maior,menor : std_logic;
24
25
26
27  begin
28
29
30  v(7) <= X(7) xnor Y(7);
31
32  v(6) <= X(6) xnor Y(6);
33
34  v(5) <= X(5) xnor Y(5);
35
36  v(4) <= X(4) xnor Y(4);
37
38  v(3) <= X(3) xnor Y(3);
39
40  v(2) <= X(2) xnor Y(2);
41
42  v(1) <= X(1) xnor Y(1);
43
44  v(0) <= X(0) xnor Y(0);
45
46
47
48  f(7) <= X(7) and not(Y(7));
49
50  f(6) <= X(6) and not(Y(6)) and v(7);
51
52  f(5) <= X(5) and not(Y(5)) and v(6) and v(7);
53
54  f(4) <= X(4) and not(Y(4)) and v(5) and v(6) and v(7);
55
56  f(3) <= X(3) and not(Y(3)) and v(4) and v(5) and v(6) and v(7);
57
58  f(2) <= X(2) and not(Y(2)) and v(3) and v(4) and v(5) and v(6) and v(7);
59
60  f(1) <= X(1) and not(Y(1)) and v(2) and v(3) and v(4) and v(5) and v(6) and v(7);
61
62  f(0) <= X(0) and not(Y(0)) and v(1) and v(2) and v(3) and v(4) and v(5) and v(6) and v(7);
63
64
65
66  igual <= (v(7) and v(6)) and (v(5) and v(4)) and (v(3) and v(2)) and (v(1) and v(0));
```

```
67
68
69
70    maior <= f(7) or f(6) or f(5) or f(4) or f(3) or f(2) or f(1) or f(0);
71
72
73
74    menor <= not(maior);
75
76
77
78    q <= menor;
79
80
81
82
83
84    end ckt;
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity Contador is
5     port (ld_cnt, clk, clr_cnt : in std_logic;
6           Q : out std_logic_vector (5 downto 0));
7   end Contador;
8
9   architecture ckt of Contador is
10
11   component Somador8bits is
12     port ( X, Y : in  std_logic_vector(7 downto 0);
13             ci : in std_logic;
14              Q : out std_logic_vector(7 downto 0);
15              cout : out std_logic);
16   end component;
17
18   component Registrador8bits is
19     port (X : in  std_logic_vector (7 downto 0);
20           ld, clr, clk, cout2 : in std_logic;
21           coutc : out std_logic;
22           Q : out std_logic_vector (7 downto 0));
23   end component;
24
25   component ComparadorMagnitude8bits is
26     port( X, Y : in std_logic_vector(7 downto 0);
27           Q : out std_logic);
28   end component;
29
30   signal soma_um, soma_out, reg_out, comp_64 : std_logic_vector(7 downto 0);
31   signal aux_not_clr_cnt, comp_out, gb, aux_gb : std_logic;
32
33   begin
34
35
36     aux_not_clr_cnt <= clr_cnt and not(comp_out);
37
38     soma_um <= "00000001";
39     comp_64 <= "01111111";
40
41     registrar : Registrador8bits port map(X=>soma_out,ld=>ld_cnt,clr=>aux_not_clr_cnt,clk=>
   clk,cout2=>gb,coutc=>aux_gb, Q=>reg_out);
42
43     somar : Somador8bits port map(X=>soma_um,Y=>reg_out,ci=>'0',Q=>soma_out,cout=>gb);
44
45     comp : ComparadorMagnitude8bits port map(X=>comp_64,Y=>reg_out,Q=>comp_out);
46
47     Q(5) <= reg_out(5);
48     Q(4) <= reg_out(4);
49     Q(3) <= reg_out(3);
50     Q(2) <= reg_out(2);
51     Q(1) <= reg_out(1);
52     Q(0) <= reg_out(0);
53
54  end ckt;
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4
5   entity conv_bin_bcd is
6
7     port( SW : in std_logic_vector(7 downto 0);
8
9           HEX2 : out std_logic_vector(6 downto 0);
10
11          HEX1 : out std_logic_vector(6 downto 0);
12
13          HEX0 : out std_logic_vector(6 downto 0));
14
15
16
17  end conv_bin_bcd;
18
19
20
21  architecture bin_bcd of conv_bin_bcd is
22
23
24
25    component conversor_somador
26
27       port ( A : in std_logic_vector(3 downto 0);
28
29              S : out std_logic_vector(3 downto 0));
30
31    end component;
32
33
34
35    component bcd_display
36
37      port( X : in std_logic_vector(3 downto 0);
38
39            D : out std_logic_vector(0 to 6));
40
41
42
43    end component;
44
45
46
47    signal m1,m2,m3,m4,m5 : std_logic_vector(3 downto 0);
48
49    signal bcd : std_logic_vector(11 downto 0);
50
51
52
53    begin
54
55
56
57        bloco1 : conversor_somador port map(A(3) => '0', A(2) => SW(7), A(1) => SW(6), A(0)
    => SW(5),
58
59                                S(3) => m1(3), S(2) => m1(2), S(1) => m1(1), S(0) => m1(0));
60
61        bloco2 : conversor_somador port map(A(3) => m1(2), A(2) => m1(1), A(1) => m1(0), A(0)
     => SW(4),
62
63                                S(3) => m2(3), S(2) => m2(2), S(1) => m2(1), S(0) => m2(0));
64
```

```
65        bloco3 : conversor_somador port map(A(3) => m2(2), A(2) => m2(1), A(1) => m2(0), A(0)
      => SW(3),
66
67                                 S(3) => m3(3), S(2) => m3(2), S(1) => m3(1), S(0) => m3(0));
68
69        bloco4 : conversor_somador port map(A(3) => '0', A(2) => m1(3), A(1) => m2(3), A(0)
     => m3(3),
70
71                                 S(3) => m4(3), S(2) => m4(2), S(1) => m4(1), S(0) => m4(0));
72
73        bloco5 : conversor_somador port map(A(3) => m3(2), A(2) => m3(1), A(1) => m3(0), A(0)
      => SW(2),
74
75                                 S(3) => m5(3), S(2) => m5(2), S(1) => m5(1), S(0) => m5(0));
76
77        bloco6 : conversor_somador port map(A(3) => m4(2), A(2) => m4(1), A(1) => m4(0), A(0)
      => m5(3),
78
79                                 S(3) => bcd(8), S(2) => bcd(7), S(1) => bcd(6), S(0) => bcd(5
     ));
80
81        bloco7 : conversor_somador port map(A(3) => m5(2), A(2) => m5(1), A(1) => m5(0), A(0)
      => SW(1),
82
83                                 S(3) => bcd(4), S(2) => bcd(3), S(1) => bcd(2), S(0) => bcd(1
     ));
84
85
86
87        bcd(9) <= m4(3);
88
89        bcd(11) <= '0';
90
91        bcd(10) <= '0';
92
93        bcd(0) <= SW(0);
94
95
96
97         CENTENA : bcd_display port map(X => bcd(11 downto 8), D =>  HEX2);
98
99         DEZENA : bcd_display port map(X => bcd(7 downto 4), D =>  HEX1);
100
101        UNIDADE : bcd_display port map(X => bcd(3 downto 0), D =>  HEX0);
102
103
104
105   end bin_bcd;
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4
5   entity conversor_somador is
6
7     port ( A : in std_logic_vector(3 downto 0);
8
9            S : out std_logic_vector(3 downto 0));
10
11
12
13  end conversor_somador;
14
15
16
17  architecture cs of conversor_somador is
18
19      begin
20
21        S(3) <= A(3) or (A(2) and A(0)) or (A(2) and A(1));
22
23        S(2) <= (A(2) and (not(A(1))) and (not(A(0)))) or (A(3) and A(0));
24
25        S(1) <= (A(1) and A(0)) or (A(3) and (not(A(0)))) or ((not(A(2))) and A(1));
26
27        S(0) <= ((not(A(3))) and (not(A(2))) and A(0)) or (A(3) and (not(A(0)))) or (A(2)
    and A(1) and (not(A(0))));
28
29
30
31   end cs;
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4
5
6   entity bcd_display is
7
8     port( X : in std_logic_vector(3 downto 0);
9
10          D : out std_logic_vector(0 to 6));
11
12
13
14  end bcd_display;
15
16
17
18  architecture display of bcd_display is
19
20
21
22    begin
23
24      D(6) <= not(X(1) or X(3) or ((not(X(2))) and (not(X(0)))) or (X(2) and X(0)));
25
26      D(5) <= not((not(X(2))) or ((not(X(1))) and (not(X(0)))) or (X(1) and X(0)));
27
28      D(4) <= not((not(X(1))) or X(0) or X(2));
29
30      D(3) <= not(X(3) or ((not(X(2))) and (not(X(0)))) or ((not(X(2))) and X(1)) or (X(1)
    and (not(X(0)))) or (X(2) and (not(X(1))) and X(0)));
31
32      D(2) <= not(((not(X(2))) and (not(X(0)))) or (X(1) and (not(X(0)))));
33
34      D(1) <= not(X(3) or ((not(X(1))) and (not(X(0)))) or (X(2) and (not(X(1)))) or (X(2)
    and (not(X(0)))));
35
36      D(0) <= not(X(3) or ((not(X(2))) and X(1)) or (X(1) and (not(X(0)))) or (X(2) and (not
    (X(1)))));
37
38
39
40  end display;
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4
5   entity ffd is
6
7     port (clk,D,P,C : in  std_logic;
8
9                   q : out std_logic);
10
11  end ffd;
12
13
14
15  architecture ckt of ffd is
16
17
18
19    signal qS : std_logic;
20
21
22
23    begin
24
25      process(clk,P,C)
26
27        begin
28
29          if P = '0' then qS <= '1';
30
31          elsif C = '0' then qS <= '0';
32
33          elsif clk = '1' and clk'EVENT then qS <= D;
34
35          end if;
36
37      end process;
38
39      q <= qS;
40
41
42
43  end ckt;
```