Centro de Tecnologia

Departamento de Engenharia Elétrica

Disciplina: Sistemas Digitais

Professor: Samaherni Morais Dias

RELATÓRIO

ATIVIDADE 2

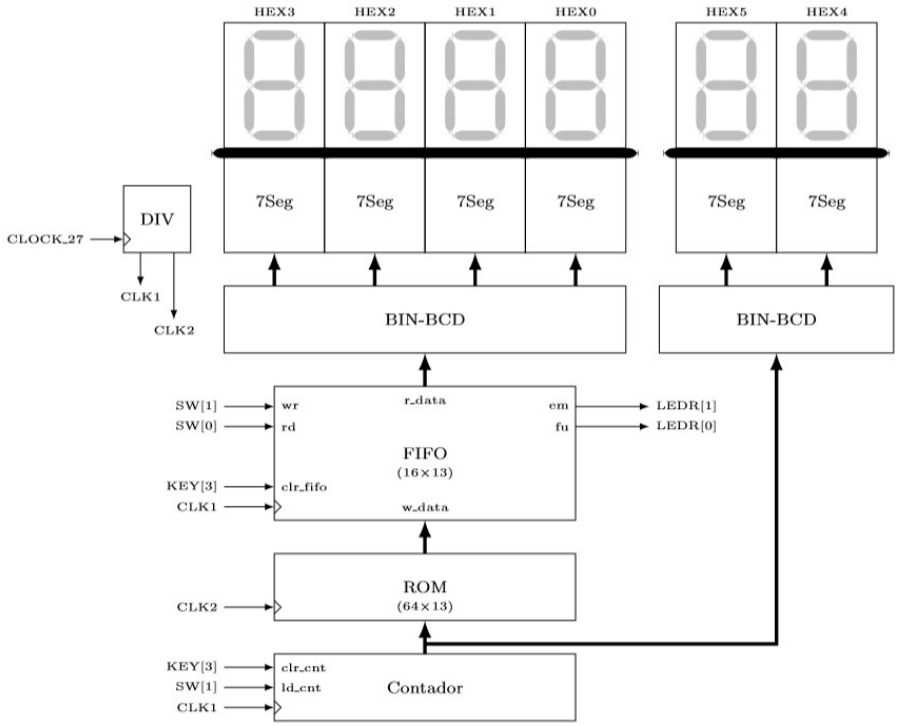
VINÍCIUS NASCIMENTO DE AZEVEDO

Natal, RN - Março de 2020

INTRODUÇÃO

A problemática apresentada para esta atividade consiste em desenvolver um sistema digital para uma FIFO. O sistema consiste em fornecer à FIFO um valor de 13 bits, proveniente de uma memória ROM, que será incluído na fila quando o switch ‘wr’ estiver em nível lógico alto. Quando o switch ‘rd’ estiver em nível lógico alto a fila deve ser lida. A FIFO também deverá contar com um botão reset. Dois LEDs deverão ser usados para indicar se a fila está cheia ou vazia. A FIFO será implementada no kit DE2.

O sistema completo está detalhado na imagem abaixo:

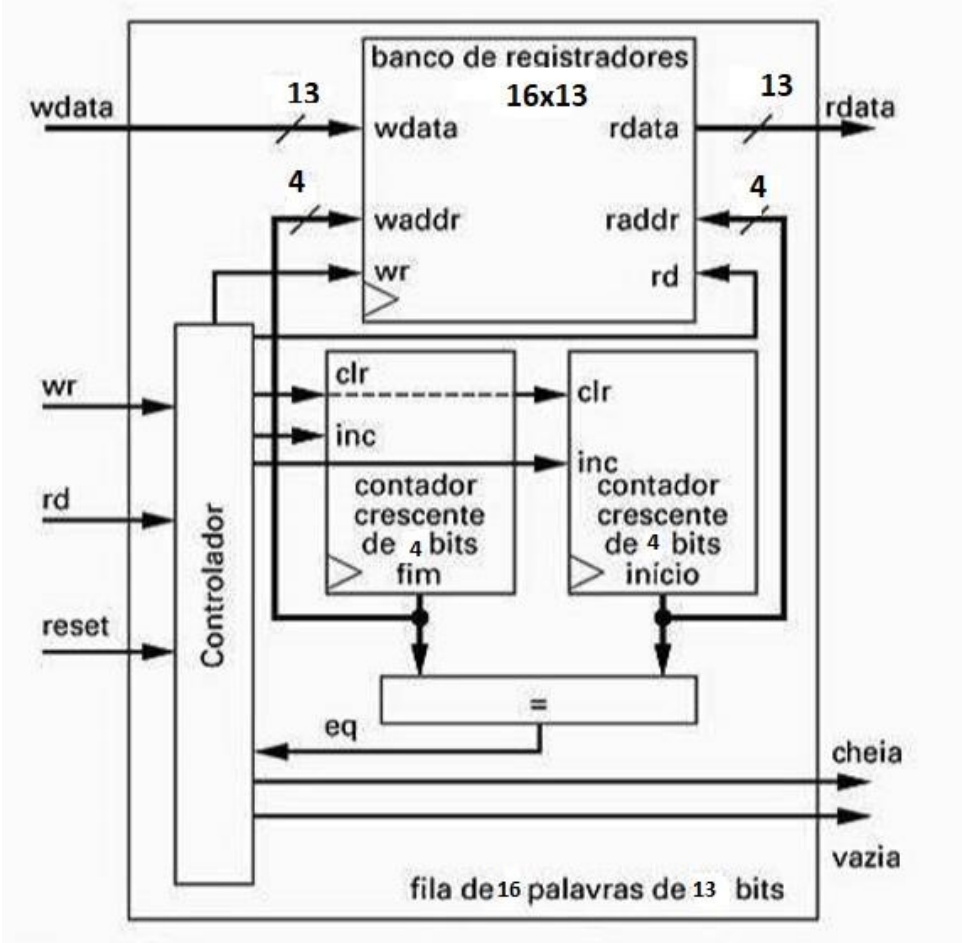


Os displays de sete segmentos HEX[3-0] mostrarão o valor lido na fila. Os displays HEX[5-4] mostrarão o valor presente do contador, ou seja, o endereço da memória onde estará o número a ser registrado na fila.

SOLUÇÃO

A solução passa pela elaboração do datapath da FIFO. 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.

O datapath da FIFO se estrutura como a imagem abaixo.



No início, todos os contadores e o banco de registradores estão zerados. Como os contadores estão com valores iguais, determinamos se a fila está cheia ou está vazia, e para determinarmos isso, basta sabermos se antes da comparação dar verdadeira houve uma escrita ou uma leitura. Como acabamos de iniciar o sistema, então, a fila está vazia.

Após 'wr' estar em nível lógico alto, o controlador envia a informação para o banco de registradores que vai permitir a entrada do valor wdata e vai registrar

no banco de registradores, no endereço definido pelo valor de saída do contador, e ao mesmo tempo ele incrementa ‘1’ no contador crescente de fim da fila. A partir de agora nossa fila não está mais vazia e pode ser lida.

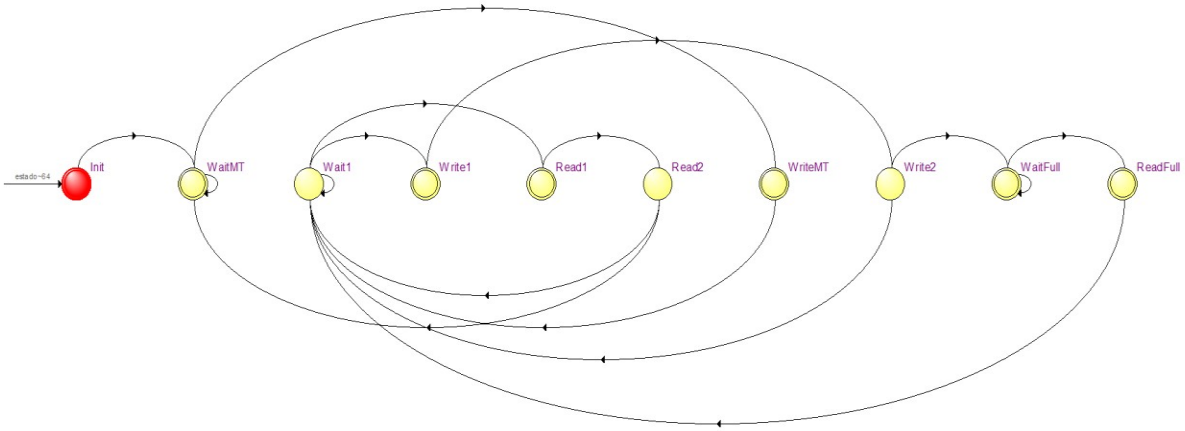
Após ‘rd’ estar em nível lógico alto, o controlador envia a informação para o banco de registradores que libera o valor rdata que está contido no endereço informado pelo contador. O controlador também autoriza o incremento de ‘1’ no contador crescente do início da fila.

A fila estará cheia quando um comando anterior ‘wr’ foi dado e os valores dos contadores forem iguais. O controlador não permite registrar quando a fila está cheia ou ler quando a fila está vazia.

Precisamos, também, entender o funcionamento do banco de registradores. O valor que sai do contador de fim de fila é um número de 4 bits e representa o endereço do registrador a ser carregado. Porém, há 16 registradores e, com isso, nós precisamos passar esse valor de 4 bits por um decodificador 4x16, ou seja, entra um número de 4 bits e sai um número de 16 bits. Cada elemento deste número de 16 bits será o load de um dos 16 registradores, ou seja, quando o elemento for ‘1’, aquele registrador será carregado. Para a leitura da fila acontece a mesma coisa, o vetor de 4 bits que sai do contador de início da fila passa por um decodificador, porém o vetor de 16 bits será o enable de um grande multiplexador que vai selecionar entre todas as saídas de todos os registradores aquela saída que deve ser lida. O diagrama de blocos do banco de registradores e do datapath está em anexo no fim do relatório.

Com o datapath, elaboramos a MDE do bloco de controle.

MDE Bloco de Controle



E as transições entre os estados:

	Source State	Destination State	Condition
1	Init	WaitMT	
2	Read1	Read2	
3	Read2	Wait1	(!eq)
4	Read2	WaitMT	(eq)
5	ReadFull	Wait1	
6	Wait1	Read1	(reset).(!wr).(rd)
7	Wait1	Wait1	(!reset) + (reset).(!wr).(!rd)
8	Wait1	Write1	(reset).(wr)
9	WaitFull	ReadFull	(reset).(rd)
10	WaitFull	WaitFull	(!reset) + (reset).(!rd)
11	WaitMT	WaitMT	(!reset) + (reset).(!wr)
12	WaitMT	WriteMT	(reset).(wr)
13	Write1	Write2	
14	Write2	Wait1	(!eq)
15	Write2	WaitFull	(eq)
16	WriteMT	Wait1	

O código do bloco de controle foi feito utilizando “process” com base no código fornecido pelo professor.

Ao analisarmos a máquina de estados, percebemos que ao ler o número, este somente ficará visível no display por um pulso de clock. Portanto, na saída do banco de registradores foi colocado um registrador de 13 bits que irá receber rdata e o seu ‘load’ será o comando ‘wr’ do controlador, a saída desse registrador irá para o conversor bin-bcd e depois para o display.

Para evitar que ocorra múltiplas escritas ou leituras quando os switches ficarem em nível lógico alto, as entradas são conectadas a blocos sincronizadores, onde ao ser colocado em ‘1’ o switch só realiza um comando e deverá ser desligado e ligado para realizar outro comando.

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.

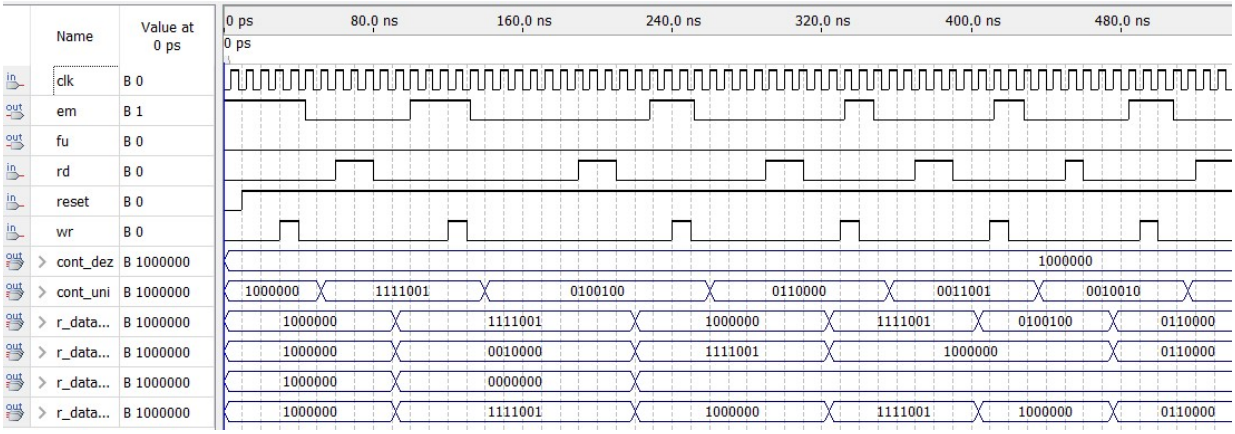
Natal, RN - Março de 2020

RESULTADOS

Para simularmos no Quartus 2, utilizamos os seguintes valores na ROM:

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	8191	10	101	200	333	500	742	899
8	900	1000	1111	1200	1333	1489	1500	1616
16	1700	7636	2400	2453	2600	4756	2800	3001
24	3107	4242	3300	3400	3500	4346	3700	3800
32	4000	4200	2068	4400	4500	4700	4850	5010
40	5100	5300	5556	5622	5700	5678	5252	6000
48	6200	6300	6400	6500	5555	6700	6800	6900
56	7000	7200	7300	7500	7600	7800	8147	8191

O resultado da simulação está mostrado na imagem abaixo.

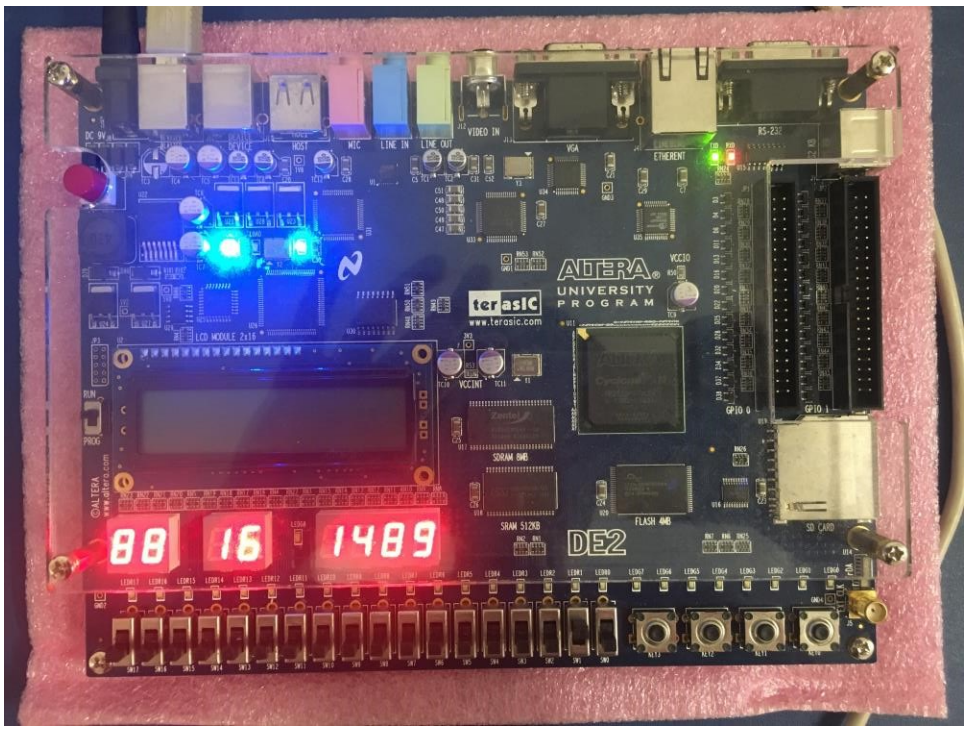


Os valores de saída mostrados estão de acordo com os LED's dos displays de sete segamentos.

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
clk	Input	PIN_D13	3	B3_N0	PIN_D13	3.3-V L...efault)		24mA (default)	
cont_dez[6]	Output	PIN_R3	1	B1_N0	PIN_R3	3.3-V L...efault)		24mA (default)	
cont_dez[5]	Output	PIN_R4	1	B1_N0	PIN_R4	3.3-V L...efault)		24mA (default)	
cont_dez[4]	Output	PIN_R5	1	B1_N0	PIN_R5	3.3-V L...efault)		24mA (default)	
cont_dez[3]	Output	PIN_T9	1	B1_N0	PIN_T9	3.3-V L...efault)		24mA (default)	
cont_dez[2]	Output	PIN_P7	1	B1_N0	PIN_P7	3.3-V L...efault)		24mA (default)	
cont_dez[1]	Output	PIN_P6	1	B1_N0	PIN_P6	3.3-V L...efault)		24mA (default)	
cont_dez[0]	Output	PIN_T2	1	B1_N0	PIN_T2	3.3-V L...efault)		24mA (default)	
cont_uni[6]	Output	PIN_T3	1	B1_N0	PIN_T3	3.3-V L...efault)		24mA (default)	
cont_uni[5]	Output	PIN_R6	1	B1_N0	PIN_R6	3.3-V L...efault)		24mA (default)	
cont_uni[4]	Output	PIN_R7	1	B1_N0	PIN_R7	3.3-V L...efault)		24mA (default)	
cont_uni[3]	Output	PIN_T4	1	B1_N0	PIN_T4	3.3-V L...efault)		24mA (default)	
cont_uni[2]	Output	PIN_U2	1	B1_N0	PIN_U2	3.3-V L...efault)		24mA (default)	
cont_uni[1]	Output	PIN_U1	1	B1_N0	PIN_U1	3.3-V L...efault)		24mA (default)	
cont_uni[0]	Output	PIN_U9	1	B1_N0	PIN_U9	3.3-V L...efault)		24mA (default)	
em	Output	PIN_AF23	7	B7_N0	PIN_AF23	3.3-V L...efault)		24mA (default)	
fu	Output	PIN_AE23	7	B7_N0	PIN_AE23	3.3-V L...efault)		24mA (default)	
r_data_cen[6]	Output	PIN_Y24	6	B6_N1	PIN_Y24	3.3-V L...efault)		24mA (default)	
r_data_cen[5]	Output	PIN_AB25	6	B6_N1	PIN_AB25	3.3-V L...efault)		24mA (default)	
r_data_cen[4]	Output	PIN_AB26	6	B6_N1	PIN_AB26	3.3-V L...efault)		24mA (default)	
r_data_cen[3]	Output	PIN_AC26	6	B6_N1	PIN_AC26	3.3-V L...efault)		24mA (default)	
r_data_cen[2]	Output	PIN_AC25	6	B6_N1	PIN_AC25	3.3-V L...efault)		24mA (default)	
r_data_cen[1]	Output	PIN_V22	6	B6_N1	PIN_V22	3.3-V L...efault)		24mA (default)	
r_data_cen[0]	Output	PIN_AB23	6	B6_N1	PIN_AB23	3.3-V L...efault)		24mA (default)	
r_data_dez[6]	Output	PIN_AB24	6	B6_N1	PIN_AB24	3.3-V L...efault)		24mA (default)	
r_data_dez[5]	Output	PIN_AA23	6	B6_N1	PIN_AA23	3.3-V L...efault)		24mA (default)	
r_data_dez[4]	Output	PIN_AA24	6	B6_N1	PIN_AA24	3.3-V L...efault)		24mA (default)	
r_data_dez[3]	Output	PIN_Y22	6	B6_N1	PIN_Y22	3.3-V L...efault)		24mA (default)	
r_data_dez[2]	Output	PIN_W21	6	B6_N1	PIN_W21	3.3-V L...efault)		24mA (default)	
r_data_dez[1]	Output	PIN_V21	6	B6_N1	PIN_V21	3.3-V L...efault)		24mA (default)	
r_data_dez[0]	Output	PIN_V20	6	B6_N1	PIN_V20	3.3-V L...efault)		24mA (default)	
r_data_mil[6]	Output	PIN_W24	6	B6_N1	PIN_W24	3.3-V L...efault)		24mA (default)	
r_data_mil[5]	Output	PIN_U22	6	B6_N1	PIN_U22	3.3-V L...efault)		24mA (default)	
r_data_mil[4]	Output	PIN_Y25	6	B6_N1	PIN_Y25	3.3-V L...efault)		24mA (default)	
r_data_mil[3]	Output	PIN_Y26	6	B6_N1	PIN_Y26	3.3-V L...efault)		24mA (default)	
r_data_mil[2]	Output	PIN_AA26	6	B6_N1	PIN_AA26	3.3-V L...efault)		24mA (default)	
r_data_mil[1]	Output	PIN_AA25	6	B6_N1	PIN_AA25	3.3-V L...efault)		24mA (default)	
r_data_mil[0]	Output	PIN_Y23	6	B6_N1	PIN_Y23	3.3-V L...efault)		24mA (default)	
r_data_uni[6]	Output	PIN_V13	8	B8_N0	PIN_V13	3.3-V L...efault)		24mA (default)	
r_data_uni[5]	Output	PIN_V14	8	B8_N0	PIN_V14	3.3-V L...efault)		24mA (default)	
r_data_uni[4]	Output	PIN_AE11	8	B8_N0	PIN_AE11	3.3-V L...efault)		24mA (default)	
r_data_uni[3]	Output	PIN_AD11	8	B8_N0	PIN_AD11	3.3-V L...efault)		24mA (default)	
r_data_uni[2]	Output	PIN_AC12	8	B8_N0	PIN_AC12	3.3-V L...efault)		24mA (default)	
r_data_uni[1]	Output	PIN_AB12	8	B8_N0	PIN_AB12	3.3-V L...efault)		24mA (default)	
r_data_uni[0]	Output	PIN_AF10	8	B8_N0	PIN_AF10	3.3-V L...efault)		24mA (default)	
rd	Input	PIN_N25	5	B5_N1	PIN_N25	3.3-V L...efault)		24mA (default)	
reset	Input	PIN_W26	6	B6_N1	PIN_W26	3.3-V L...efault)		24mA (default)	
wr	Input	PIN_N26	5	B5_N1	PIN_N26	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:



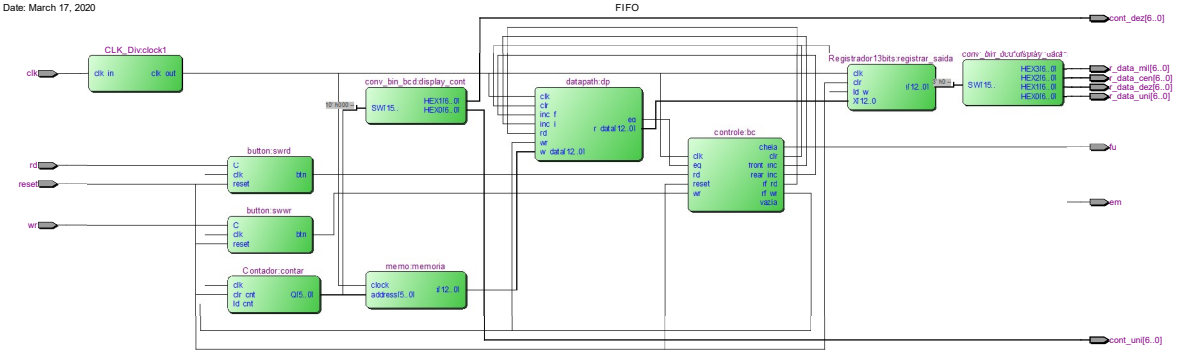
Natal, RN - Março de 2020

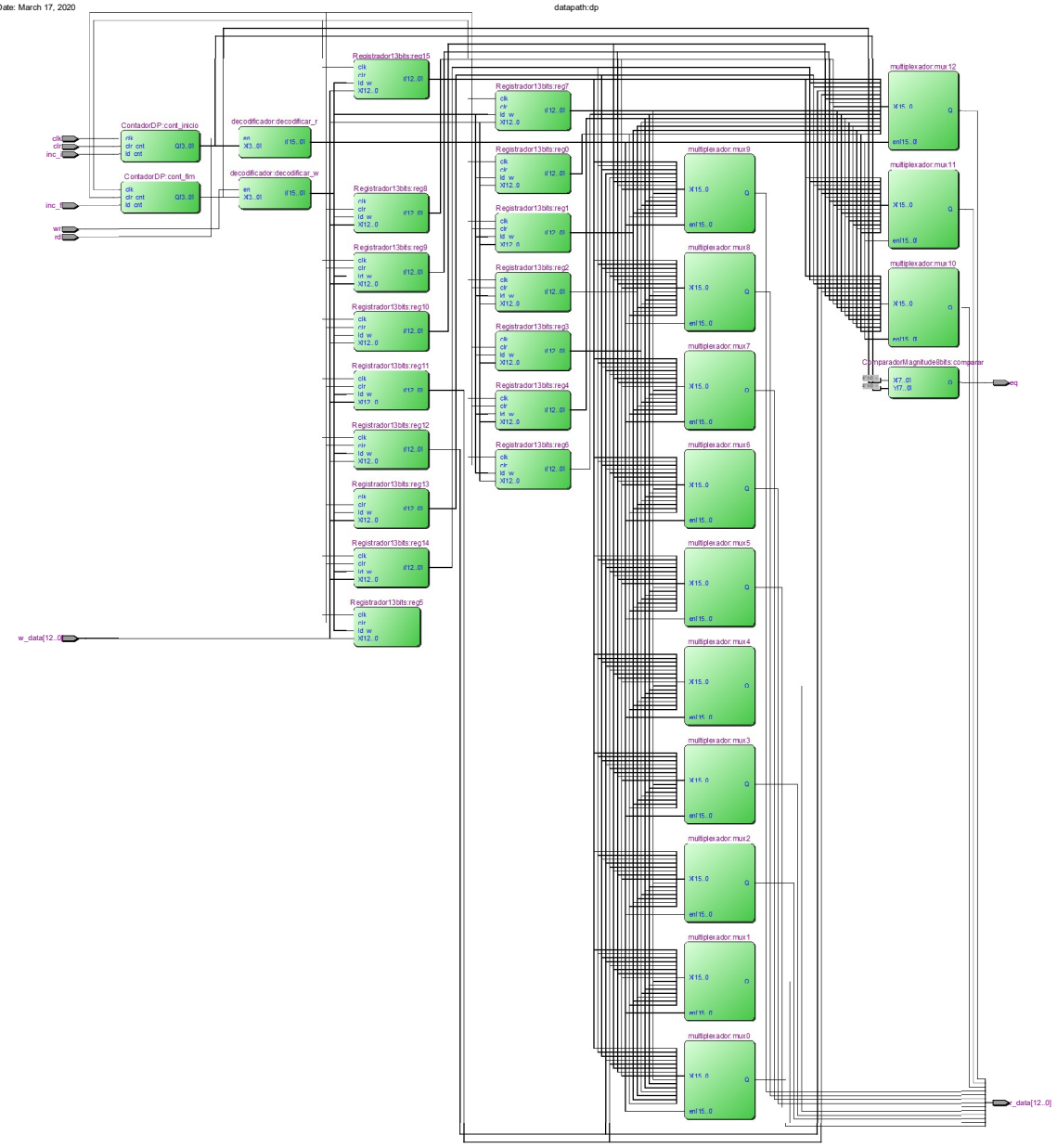
CONCLUSÕES

O código do sistema digital da FIFO 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


CLK_Div:clock1
clk_in
clk_out
clk
conv_bin_bcd:display_cont
SW[15..
HEX1[6..0]
HEX0[6..0]
datapath:dp
clk
clr
inc_f
inc_i
rd
wr
w_data[12..0]
ea
r_data[12..0]
Registrador13bits:registrar_saida
clk
clr
ld_w
xi[12..0]
xf[12..0]
conv_bin_bcd:display_uni
SW[15..
HEX3[6..0]
HEX2[6..0]
HEX1[6..0]
HEX0[6..0]
controle:bc
clk
eq
rd
reset
wr
cheia
clr
front_inc
rear_inc
rf_rd
rf_we
vazia
button:swrd
C
clk
reset
btn
button:swwr
C
clk
reset
btn
Contador:contar
clk
clr_cnt
ld_cnt
Q[5..0]
memo:memoria
clock
address[5..0]
if[12..0]
rd
reset
wr
cont_dez[6..0]
w_data_mil[6..0]
w_data_cen[6..0]
w_data_dez[6..0]
w_data_uni[6..0]
fu
em
cont_uni[6..0]


ContadorDP:cont_inicio
ContadorDP:cont_fim
decodificador:decodificar_r
decodificador:decodificar_w
Registrador13bits:reg15
Registrador13bits:reg8
Registrador13bits:reg9
Registrador13bits:reg10
Registrador13bits:reg11
Registrador13bits:reg12
Registrador13bits:reg13
Registrador13bits:reg14
Registrador13bits:reg5
Registrador13bits:reg7
Registrador13bits:reg0
Registrador13bits:reg1
Registrador13bits:reg2
Registrador13bits:reg3
Registrador13bits:reg4
Registrador13bits:reg6
multiplexador:mux12
multiplexador:mux11
multiplexador:mux10
multiplexador:mux9
multiplexador:mux8
multiplexador:mux7
multiplexador:mux6
multiplexador:mux5
multiplexador:mux4
multiplexador:mux3
multiplexador:mux2
multiplexador:mux1
multiplexador:mux0
ComparadorMagnitude8bits:comparar
clk
clr
inc_r
inc_f
wr
rd
w_data[12..0]
eq
r_data[12..0]

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity FIFO is
5      port (clk, reset, wr, rd : in std_logic;
6            em, fu : out std_logic;
7            r_data_mil, r_data_cen, r_data_dez, r_data_uni, cont_dez, cont_uni : out
    std_logic_vector(6 downto 0));
8   end FIFO;
9
10  architecture ckt of FIFO is
11
12    component datapath is
13      port (clk, clr, inc_f, inc_i, wr, rd : in std_logic;
14            w_data : in std_logic_vector(12 downto 0);
15            eq : out std_logic;
16            r_data : out std_logic_vector(12 downto 0));
17    end component;
18
19    component controle is
20      port (clk, reset, wr, rd, eq : in std_logic;
21           clr, rear_inc, front_inc, rf_wr, rf_rd, cheia, vazia: out std_logic);
22    end component;
23
24    component Contador is
25      port (ld_cnt, clk, clr_cnt : in std_logic;
26            Q : out std_logic_vector (5 downto 0));
27    end component;
28
29    component memo IS
30      port (address : IN STD_LOGIC_VECTOR (5 DOWNTO 0);
31            clock      : IN STD_LOGIC;
32            q    : OUT STD_LOGIC_VECTOR (12 DOWNTO 0));
33    end component;
34
35    component conv_bin_bcd is
36      port(SW : in std_logic_vector(15 downto 0);
37           HEX4 : out std_logic_vector(6 downto 0);
38           HEX3 : out std_logic_vector(6 downto 0);
39           HEX2 : out std_logic_vector(6 downto 0);
40           HEX1 : out std_logic_vector(6 downto 0);
41           HEX0 : out std_logic_vector(6 downto 0));
42    end component;
43
44    component Registrador13bits is
45      port(X : in  std_logic_vector (12 downto 0);
46           ld_w, clr, clk : in std_logic;
47           Q : out std_logic_vector (12 downto 0));
48    end component;
49
50    component button is
51      port(C, clk, reset : in std_logic;
52           btn : out std_logic);
53    end component;
54
55    component CLK_Div is
56      port (clk_in : in std_logic;
57            clk_out : out std_logic);
58    end component;
59
60    signal cnt_out : std_logic_vector(5 downto 0);
61    signal cont_mil, cont_cen, data_d_mil, cont_d_mil : std_logic_vector(6 downto 0);
62    signal w_data_aux, r_data_aux, r_data_aux2 : std_logic_vector(12 downto 0);
63    signal clr_cntrl, fim_inc, ini_inc, aux_wr, aux_rd, aux_clk, comp_out, btnrd, btnwr :
    std_logic;
64
```

```
65    begin
66
67    swrd : button port map(C=>rd,clk=>aux_clk,reset=>reset,btn=>btnrd);
68    swwr : button port map(C=>wr,clk=>aux_clk,reset=>reset,btn=>btnwr);
69
70    clock1 : CLK_Div port map(clk_in=>clk,clk_out=>aux_clk);
71    contar : Contador port map(ld_cnt=>aux_wr,clk=>aux_clk,clr_cnt=>reset,Q=>cnt_out);
72    memoria : memo port map(address=>cnt_out,clock=>aux_clk,q=>w_data_aux);
73    bc : controle port map(clk=>aux_clk,reset=>reset,wr=>btnwr,rd=>btnrd,eq=>comp_out,clr=>
      clr_cntrl,rear_inc=>fim_inc,front_inc=>ini_inc,
74                           rf_wr=>aux_wr,rf_rd=>aux_rd,cheia=>fu,vazia=>em);
75    dp : datapath port map(clk=>aux_clk,clr=>clr_cntrl,inc_f=>fim_inc,inc_i=>ini_inc,wr=>
      aux_wr,rd=>aux_rd,w_data=>w_data_aux,eq=>comp_out,r_data=>r_data_aux);
76    registrar_saida : Registrador13bits port map(X=>r_data_aux,ld_w=>aux_rd,clr=>reset,clk=>
      aux_clk,Q=>r_data_aux2);
77
78    display_data : conv_bin_bcd port map(SW(15)=>'0',SW(14)=>'0',SW(13)=>'0',SW(12)=>
      r_data_aux2(12),SW(11)=>r_data_aux2(11),SW(10)=>r_data_aux2(10),
79                                         SW(9)=>r_data_aux2(9),SW(8)=>r_data_aux2(8),SW(7)=>
      r_data_aux2(7),SW(6)=>r_data_aux2(6),SW(5)=>r_data_aux2(5),
80                                         SW(4)=>r_data_aux2(4),SW(3)=>r_data_aux2(3),SW(2)=>
      r_data_aux2(2),SW(1)=>r_data_aux2(1),SW(0)=>r_data_aux2(0),
81                                         HEX4=>data_d_mil,HEX3=>r_data_mil,HEX2=>r_data_cen,
      HEX1=>r_data_dez,HEX0=>r_data_uni);
82    display_cont : conv_bin_bcd port map(SW(15)=>'0',SW(14)=>'0',SW(13)=>'0',SW(12)=>'0',SW(
      11)=>'0',SW(10)=>'0',SW(9)=>'0',SW(8)=>'0',SW(7)=>'0',SW(6)=>'0',
83                                         SW(5)=>cnt_out(5),SW(4)=>cnt_out(4),SW(3)=>cnt_out(
      3),SW(2)=>cnt_out(2),SW(1)=>cnt_out(1),SW(0)=>cnt_out(0),
84                                         HEX4=>cont_d_mil,HEX3=>cont_mil,HEX2=>cont_cen,HEX1
      =>cont_dez,HEX0=>cont_uni);
85
86
87  end ckt;
88
89
90
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4
5   entity controle is
6     port (clk, reset, wr, rd, eq : in std_logic;
7           clr, rear_inc, front_inc, rf_wr, rf_rd, cheia, vazia: out std_logic);
8   end controle;
9
10  architecture ckt of controle is
11    type st is (Init, WaitMT, WriteMT, Wait1, Read1, Read2, Write1, Write2, WaitFull,
    ReadFull);
12    signal estado : st ;
13    begin
14  -----saidas --------
15
16    with estado select
17    clr <= '0' when Init,
18             '1' when others;
19
20    with estado select
21    rear_inc <= '1' when WriteMT,
22                '1' when Write1,
23             '0' when others;
24
25    with estado select
26    front_inc <= '1' when  Read1,
27                 '1' when ReadFull,
28           '0' when others;
29
30    with estado select
31    rf_wr <= '1' when  WriteMT,
32             '1' when Write1,
33           '0' when others;
34
35    with estado select
36    rf_rd <= '1' when  Read1,
37             '1'when ReadFull,
38           '0' when others;
39
40    with estado select
41    cheia <= '1' when  WaitFull,
42           '0' when others;
43
44    with estado select
45    vazia <= '1' when  Init,
46             '1' when WaitMT,
47           '0' when others;
48
49   ------------------
50
51  abc : process (clk , reset)
52  begin
53    if reset = '0' then
54      estado <= Init;
55    elsif ( clk'event and clk = '1') then
56     case estado is
57
58          when Init =>
59             estado <= WaitMT;
60
61
62       when WaitMT  =>
63
64           if (reset = '1' and wr = '1' ) then estado <= WriteMT;
65      elsif (reset = '1' and wr = '0' ) then estado <= WaitMT;
```

```
66              end if;
67
68      when WriteMT =>
69              estado <= Wait1;
70
71      when Wait1  =>
72
73           if (reset = '1' and wr = '0' and rd = '1' ) then estado <= Read1;
74      elsif (reset = '1' and wr = '0' and rd = '0' ) then estado <= Wait1;
75      elsif (reset = '1' and wr = '1' ) then estado <= Write1;
76              end if;
77
78      when Read1 =>
79              estado <= Read2;
80
81      when Read2  =>
82
83           if (eq = '1' ) then estado <= WaitMT;
84      elsif (eq = '0') then estado <= Wait1;
85              end if;
86
87      when Write1 =>
88              estado <= Write2;
89
90      when Write2  =>
91
92           if (eq = '1' ) then estado <= WaitFull;
93      elsif (eq = '0') then estado <= Wait1;
94              end if;
95
96
97       when WaitFull  =>
98
99           if (reset = '1' and rd = '1' ) then estado <= ReadFull;
100     elsif (reset = '1' and wr = '0' ) then estado <= WaitFull;
101             end if;
102
103      when ReadFull =>
104             estado <= Wait1;
105
106
107  end case ;
108  end if;
109  end process abc ;
110  end ckt ;
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity datapath is
5     port (clk, clr, inc_f, inc_i, wr, rd : in std_logic;
6           w_data : in std_logic_vector(12 downto 0);
7           eq : out std_logic;
8           r_data : out std_logic_vector(12 downto 0));
9   end datapath;
10
11  architecture ckt of datapath is
12
13    component ContadorDP is
14      port (ld_cnt, clk, clr_cnt : in std_logic;
15            Q : out std_logic_vector (3 downto 0));
16    end component;
17
18    component ComparadorMagnitude8bits is
19      port(X, Y : in std_logic_vector(7 downto 0);
20           Q : out std_logic);
21    end component;
22
23    component decodificador is
24      port(X : in std_logic_vector(3 downto 0);
25         en : in std_logic;
26         Q : out std_logic_vector(15 downto 0));
27    end component;
28
29    component Registrador13bits is
30      port (X : in  std_logic_vector (12 downto 0);
31            ld_w, clr, clk : in std_logic;
32            Q : out std_logic_vector (12 downto 0));
33    end component;
34
35    component multiplexador is
36      port(X, en : in std_logic_vector(15 downto 0);
37         Q : out std_logic);
38    end component;
39
40
41    signal cont_fim_out, cont_inicio_out : std_logic_vector(3 downto 0);
42    signal add_out_w, add_out_r : std_logic_vector(15 downto 0);
43    signal q0, q1, q2, q3, q4, q5, q6, q7, q8, q9, q10, q11, q12, q13, q14, q15 :
    std_logic_vector(12 downto 0);
44
45
46    begin
47
48      cont_fim : ContadorDP port map(ld_cnt=>inc_f,clk=>clk,clr_cnt=>clr,Q=>cont_fim_out);
49      cont_inicio : ContadorDP port map(ld_cnt=>inc_i,clk=>clk,clr_cnt=>clr,Q=>
    cont_inicio_out);
50      comparar : ComparadorMagnitude8bits port map(X(7)=>'0',X(6)=>'0',X(5)=>'0',X(4)=>'0',X
    (3)=>cont_fim_out(3),X(2)=>cont_fim_out(2),
51                                                   X(1)=>cont_fim_out(1),X(0)=>cont_fim_out(0),
52                                                   Y(7)=>'0',Y(6)=>'0',Y(5)=>'0',Y(4)=>'0',Y(3)=>
    cont_inicio_out(3),Y(2)=>cont_inicio_out(2),
53                                                   Y(1)=>cont_inicio_out(1),Y(0)=>cont_inicio_out(0),
    Q=>eq);
54      decodificar_w : decodificador port map(X=>cont_fim_out,en=>wr,Q=>add_out_w);
55      decodificar_r : decodificador port map(X=>cont_inicio_out,en=>rd,Q=>add_out_r);
56
57
58      reg0 : Registrador13bits port map(X=>w_data,ld_w=>add_out_w(0),clr=>clr,clk=>clk,Q=>q0
    );
59      reg1 : Registrador13bits port map(X=>w_data,ld_w=>add_out_w(1),clr=>clr,clk=>clk,Q=>q1
    );
```

```
60      reg2 : Registrador13bits port map(X=>w_data,ld_w=>add_out_w(2),clr=>clr,clk=>clk,Q=>q2
    );
61      reg3 : Registrador13bits port map(X=>w_data,ld_w=>add_out_w(3),clr=>clr,clk=>clk,Q=>q3
    );
62      reg4 : Registrador13bits port map(X=>w_data,ld_w=>add_out_w(4),clr=>clr,clk=>clk,Q=>q4
    );
63      reg5 : Registrador13bits port map(X=>w_data,ld_w=>add_out_w(5),clr=>clr,clk=>clk,Q=>q5
    );
64      reg6 : Registrador13bits port map(X=>w_data,ld_w=>add_out_w(6),clr=>clr,clk=>clk,Q=>q6
    );
65      reg7 : Registrador13bits port map(X=>w_data,ld_w=>add_out_w(7),clr=>clr,clk=>clk,Q=>q7
    );
66      reg8 : Registrador13bits port map(X=>w_data,ld_w=>add_out_w(8),clr=>clr,clk=>clk,Q=>q8
    );
67      reg9 : Registrador13bits port map(X=>w_data,ld_w=>add_out_w(9),clr=>clr,clk=>clk,Q=>q9
    );
68      reg10 : Registrador13bits port map(X=>w_data,ld_w=>add_out_w(10),clr=>clr,clk=>clk,Q=>
    q10);
69      reg11 : Registrador13bits port map(X=>w_data,ld_w=>add_out_w(11),clr=>clr,clk=>clk,Q=>
    q11);
70      reg12 : Registrador13bits port map(X=>w_data,ld_w=>add_out_w(12),clr=>clr,clk=>clk,Q=>
    q12);
71      reg13 : Registrador13bits port map(X=>w_data,ld_w=>add_out_w(13),clr=>clr,clk=>clk,Q=>
    q13);
72      reg14 : Registrador13bits port map(X=>w_data,ld_w=>add_out_w(14),clr=>clr,clk=>clk,Q=>
    q14);
73      reg15 : Registrador13bits port map(X=>w_data,ld_w=>add_out_w(15),clr=>clr,clk=>clk,Q=>
    q15);
74
75      mux0 : multiplexador port map(x(0)=>q0(0),x(1)=>q1(0),x(2)=>q2(0),x(3)=>q3(0),x(4)=>q4
    (0),x(5)=>q4(0),x(6)=>q6(0),x(7)=>q7(0),
76                                    x(8)=>q8(0),x(9)=>q9(0),x(10)=>q10(0),x(11)=>q11(0),x(12
    )=>q12(0),x(13)=>q13(0),x(14)=>q14(0),x(15)=>q15(0),
77                                    en=>add_out_r,Q=>r_data(0));
78      mux1 : multiplexador port map(x(0)=>q0(1),x(1)=>q1(1),x(2)=>q2(1),x(3)=>q3(1),x(4)=>q4
    (1),x(5)=>q4(1),x(6)=>q6(1),x(7)=>q7(1),
79                                    x(8)=>q8(1),x(9)=>q9(1),x(10)=>q10(1),x(11)=>q11(1),x(12
    )=>q12(1),x(13)=>q13(1),x(14)=>q14(1),x(15)=>q15(1),
80                                    en=>add_out_r,Q=>r_data(1));
81      mux2 : multiplexador port map(x(0)=>q0(2),x(1)=>q1(2),x(2)=>q2(2),x(3)=>q3(2),x(4)=>q4
    (2),x(5)=>q4(2),x(6)=>q6(2),x(7)=>q7(2),
82                                    x(8)=>q8(2),x(9)=>q9(2),x(10)=>q10(2),x(11)=>q11(2),x(12
    )=>q12(2),x(13)=>q13(2),x(14)=>q14(2),x(15)=>q15(2),
83                                    en=>add_out_r,Q=>r_data(2));
84      mux3 : multiplexador port map(x(0)=>q0(3),x(1)=>q1(3),x(2)=>q2(3),x(3)=>q3(3),x(4)=>q4
    (3),x(5)=>q4(3),x(6)=>q6(3),x(7)=>q7(3),
85                                    x(8)=>q8(3),x(9)=>q9(3),x(10)=>q10(3),x(11)=>q11(3),x(12
    )=>q12(3),x(13)=>q13(3),x(14)=>q14(3),x(15)=>q15(3),
86                                    en=>add_out_r,Q=>r_data(3));
87      mux4 : multiplexador port map(x(0)=>q0(4),x(1)=>q1(4),x(2)=>q2(4),x(3)=>q3(4),x(4)=>q4
    (4),x(5)=>q4(4),x(6)=>q6(4),x(7)=>q7(4),
88                                    x(8)=>q8(4),x(9)=>q9(4),x(10)=>q10(4),x(11)=>q11(4),x(12
    )=>q12(4),x(13)=>q13(4),x(14)=>q14(4),x(15)=>q15(4),
89                                    en=>add_out_r,Q=>r_data(4));
90      mux5 : multiplexador port map(x(0)=>q0(5),x(1)=>q1(5),x(2)=>q2(5),x(3)=>q3(5),x(4)=>q4
    (5),x(5)=>q4(5),x(6)=>q6(5),x(7)=>q7(5),
91                                    x(8)=>q8(5),x(9)=>q9(5),x(10)=>q10(5),x(11)=>q11(5),x(12
    )=>q12(5),x(13)=>q13(5),x(14)=>q14(5),x(15)=>q15(5),
92                                    en=>add_out_r,Q=>r_data(5));
93      mux6 : multiplexador port map(x(0)=>q0(6),x(1)=>q1(6),x(2)=>q2(6),x(3)=>q3(6),x(4)=>q4
    (6),x(5)=>q4(6),x(6)=>q6(6),x(7)=>q7(6),
94                                    x(8)=>q8(6),x(9)=>q9(6),x(10)=>q10(6),x(11)=>q11(6),x(12
    )=>q12(6),x(13)=>q13(6),x(14)=>q14(6),x(15)=>q15(6),
95                                    en=>add_out_r,Q=>r_data(6));
96      mux7 : multiplexador port map(x(0)=>q0(7),x(1)=>q1(7),x(2)=>q2(7),x(3)=>q3(7),x(4)=>q4
    (7),x(5)=>q4(7),x(6)=>q6(7),x(7)=>q7(7),
```

```
97                                      x(8)=>q8(7),x(9)=>q9(7),x(10)=>q10(7),x(11)=>q11(7),x(12
    )=>q12(7),x(13)=>q13(7),x(14)=>q14(7),x(15)=>q15(7),
98                                      en=>add_out_r,Q=>r_data(7));
99          mux8 : multiplexador port map(x(0)=>q0(8),x(1)=>q1(8),x(2)=>q2(8),x(3)=>q3(8),x(4)=>q4
    (8),x(5)=>q4(8),x(6)=>q6(8),x(7)=>q7(8),
100                                     x(8)=>q8(8),x(9)=>q9(8),x(10)=>q10(8),x(11)=>q11(8),x(12
    )=>q12(8),x(13)=>q13(8),x(14)=>q14(8),x(15)=>q15(8),
101                                     en=>add_out_r,Q=>r_data(8));
102         mux9 : multiplexador port map(x(0)=>q0(9),x(1)=>q1(9),x(2)=>q2(9),x(3)=>q3(9),x(4)=>q4
    (9),x(5)=>q4(9),x(6)=>q6(9),x(7)=>q7(9),
103                                     x(8)=>q8(9),x(9)=>q9(9),x(10)=>q10(9),x(11)=>q11(9),x(12
    )=>q12(9),x(13)=>q13(9),x(14)=>q14(9),x(15)=>q15(9),
104                                     en=>add_out_r,Q=>r_data(9));
105         mux10 : multiplexador port map(x(0)=>q0(10),x(1)=>q1(10),x(2)=>q2(10),x(3)=>q3(10),x(4
    )=>q4(10),x(5)=>q4(10),x(6)=>q6(10),x(7)=>q7(10),
106                                     x(8)=>q8(10),x(9)=>q9(10),x(10)=>q10(10),x(11)=>q11(10),
    x(12)=>q12(10),x(13)=>q13(10),x(14)=>q14(10),x(15)=>q15(10),
107                                     en=>add_out_r,Q=>r_data(10));
108         mux11 : multiplexador port map(x(0)=>q0(11),x(1)=>q1(11),x(2)=>q2(11),x(3)=>q3(11),x(4
    )=>q4(11),x(5)=>q4(11),x(6)=>q6(11),x(7)=>q7(11),
109                                     x(8)=>q8(11),x(9)=>q9(11),x(10)=>q10(11),x(11)=>q11(11),
    x(12)=>q12(11),x(13)=>q13(11),x(14)=>q14(11),x(15)=>q15(11),
110                                     en=>add_out_r,Q=>r_data(11));
111         mux12 : multiplexador port map(x(0)=>q0(12),x(1)=>q1(12),x(2)=>q2(12),x(3)=>q3(12),x(4
    )=>q4(12),x(5)=>q4(12),x(6)=>q6(12),x(7)=>q7(12),
112                                     x(8)=>q8(12),x(9)=>q9(12),x(10)=>q10(12),x(11)=>q11(12),
    x(12)=>q12(12),x(13)=>q13(12),x(14)=>q14(12),x(15)=>q15(12),
113                                     en=>add_out_r,Q=>r_data(12));
114
115
116 end ckt;
117
118
119
120
121
122
123
124
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity multiplexador is
5     port(X, en : in std_logic_vector(15 downto 0);
6          Q : out std_logic);
7   end multiplexador;
8
9   architecture ckt of multiplexador is
10
11    begin
12
13    Q <= (X(0) and en(0)) or (X(1) and en(1)) or (X(2) and en(2)) or (X(3) and en(3)) or (X(
      4) and en(4)) or (X(5) and en(5))
14       or (X(6) and en(6)) or (X(7) and en(7)) or (X(8) and en(8)) or (X(9) and en(9)) or (X(
      10) and en(10)) or (X(11) and en(11))
15       or (X(12) and en(12)) or (X(13) and en(13)) or (X(14) and en(14)) or (X(15) and en(15
      ));
16
17  end ckt;
```

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

entity decodificador is
  port(X : in std_logic_vector(3 downto 0);
       en : in std_logic;
       Q : out std_logic_vector(15 downto 0));
end decodificador;

architecture ckt of decodificador is

   signal aux_2 : std_logic_vector(15 downto 0);
   signal aux : std_logic_vector(7 downto 0);

   begin

   aux(0) <= not(X(0)) and not(X(1));
   aux(1) <= X(1) and not(X(0));
   aux(2) <= not(X(1)) and X(0);
   aux(3) <= X(0) and X(1);
   aux(4) <= not(X(2)) and not(X(3));
   aux(5) <= X(3) and not(X(2));
   aux(6) <= not(X(3)) and X(2);
   aux(7) <= X(2) and X(3);

   aux_2(0) <= en and aux(0) and aux(4);
   aux_2(1) <= en and aux(0) and aux(5);
   aux_2(2) <= en and aux(0) and aux(6);
   aux_2(3) <= en and aux(0) and aux(7);
   aux_2(4) <= en and aux(1) and aux(4);
   aux_2(5) <= en and aux(1) and aux(5);
   aux_2(6) <= en and aux(1) and aux(6);
   aux_2(7) <= en and aux(1) and aux(7);
   aux_2(8) <= en and aux(2) and aux(4);
   aux_2(9) <= en and aux(2) and aux(5);
   aux_2(10) <= en and aux(2) and aux(6);
   aux_2(11) <= en and aux(2) and aux(7);
   aux_2(12) <= en and aux(3) and aux(4);
   aux_2(13) <= en and aux(3) and aux(5);
   aux_2(14) <= en and aux(3) and aux(6);
   aux_2(15) <= en and aux(3) and aux(7);

   Q <= aux_2;

end ckt;
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4
5   entity Registrador13bits is
6
7     port (X : in  std_logic_vector (12 downto 0);
8
9           ld_w, clr, clk : in std_logic;
10
11          Q : out std_logic_vector (12 downto 0));
12
13  end Registrador13bits;
14
15
16
17  architecture ckt of Registrador13bits is
18
19
20
21    component ffd is
22
23      port (clk,D,P,C : in  std_logic;
24
25                    q : out std_logic);
26
27    end component;
28
29
30
31    signal mux, aux : std_logic_vector(12 downto 0);
32
33
34
35    begin
36
37     mux(12) <= (X(12) and ld_w) or (aux(12) and not(ld_w));
38
39     mux(11) <= (X(11) and ld_w) or (aux(11) and not(ld_w));
40
41     mux(10) <= (X(10) and ld_w) or (aux(10) and not(ld_w));
42
43     mux(9) <= (X(9) and ld_w) or (aux(9) and not(ld_w));
44
45     mux(8) <= (X(8) and ld_w) or (aux(8) and not(ld_w));
46
47     mux(7) <= (X(7) and ld_w) or (aux(7) and not(ld_w));
48
49     mux(6) <= (X(6) and ld_w) or (aux(6) and not(ld_w));
50
51     mux(5) <= (X(5) and ld_w) or (aux(5) and not(ld_w));
52
53     mux(4) <= (X(4) and ld_w) or (aux(4) and not(ld_w));
54
55     mux(3) <= (X(3) and ld_w) or (aux(3) and not(ld_w));
56
57     mux(2) <= (X(2) and ld_w) or (aux(2) and not(ld_w));
58
59     mux(1) <= (X(1) and ld_w) or (aux(1) and not(ld_w));
60
61     mux(0) <= (X(0) and ld_w) or (aux(0) and not(ld_w));
62
63
64     ffd12 : ffd port map (clk => clk, D => mux(12), P => '1', C => clr, q => aux(12));
65
66     ffd11 : ffd port map (clk => clk, D => mux(11), P => '1', C => clr, q => aux(11));
```

```
67
68      ffd10 : ffd port map (clk => clk, D => mux(10), P => '1', C => clr, q => aux(10));
69
70      ffd9 : ffd port map (clk => clk, D => mux(9), P => '1', C => clr, q => aux(9));
71
72      ffd8 : ffd port map (clk => clk, D => mux(8), P => '1', C => clr, q => aux(8));
73
74      ffd7 : ffd port map (clk => clk, D => mux(7), P => '1', C => clr, q => aux(7));
75
76      ffd6 : ffd port map (clk => clk, D => mux(6), P => '1', C => clr, q => aux(6));
77
78      ffd5 : ffd port map (clk => clk, D => mux(5), P => '1', C => clr, q => aux(5));
79
80      ffd4 : ffd port map (clk => clk, D => mux(4), P => '1', C => clr, q => aux(4));
81
82      ffd3 : ffd port map (clk => clk, D => mux(3), P => '1', C => clr, q => aux(3));
83
84      ffd2 : ffd port map (clk => clk, D => mux(2), P => '1', C => clr, q => aux(2));
85
86      ffd1 : ffd port map (clk => clk, D => mux(1), P => '1', C => clr, q => aux(1));
87
88      ffd0 : ffd port map (clk => clk, D => mux(0), P => '1', C => clr, q => aux(0));
89
90
91      Q(12) <= aux(12);
92
93      Q(11) <= aux(11);
94
95      Q(10) <= aux(10);
96
97      Q(9) <= aux(9);
98
99      Q(8) <= aux(8);
100
101     Q(7) <= aux(7);
102
103     Q(6) <= aux(6);
104
105     Q(5) <= aux(5);
106
107     Q(4) <= aux(4);
108
109     Q(3) <= aux(3);
110
111     Q(2) <= aux(2);
112
113     Q(1) <= aux(1);
114
115     Q(0) <= aux(0);
116
117
118
119 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(15 downto 0);
8
9           HEX4 : out std_logic_vector(6 downto 0);
10
11          HEX3 : out std_logic_vector(6 downto 0);
12
13          HEX2 : out std_logic_vector(6 downto 0);
14
15          HEX1 : out std_logic_vector(6 downto 0);
16
17          HEX0 : out std_logic_vector(6 downto 0));
18
19
20  end conv_bin_bcd;
21
22
23
24  architecture bin_bcd of conv_bin_bcd is
25
26
27
28    component conversor_somador
29
30       port ( A : in std_logic_vector(3 downto 0);
31
32              S : out std_logic_vector(3 downto 0));
33
34    end component;
35
36
37
38    component bcd_display
39
40      port( X : in std_logic_vector(3 downto 0);
41
42            D : out std_logic_vector(0 to 6));
43
44
45
46    end component;
47
48
49
50    signal m1, m2, m3, m4, m5, m6, m7, m8, m9, m10, m11, m12, m13, m14, m15, m16, m17, m18,
    m19, m20, m21,
51          m22, m23, m24, m25, m26, m27, m28, m29, m30 : std_logic_vector(3 downto 0);
52
53    signal bcd : std_logic_vector(19 downto 0);
54
55
56
57    begin
58
59
60
61       bloco1 : conversor_somador port map(A(3) => '0', A(2) => SW(15), A(1) => SW(14), A(0)
     => SW(13),
62
63                                S(3) => m1(3), S(2) => m1(2), S(1) => m1(1), S(0) => m1(0));
64
```

```
65        bloco2 : conversor_somador port map(A(3) => m1(2), A(2) => m1(1), A(1) => m1(0), A(0)
      => SW(12),
66
67                                 S(3) => m2(3), S(2) => m2(2), S(1) => m2(1), S(0) => m2(0));
68
69        bloco3 : conversor_somador port map(A(3) => m2(2), A(2) => m2(1), A(1) => m2(0), A(0)
      => SW(11),
70
71                                 S(3) => m3(3), S(2) => m3(2), S(1) => m3(1), S(0) => m3(0));
72
73        bloco4 : conversor_somador port map(A(3) => '0', A(2) => m1(3), A(1) => m2(3), A(0)
     => m3(3),
74
75                                 S(3) => m4(3), S(2) => m4(2), S(1) => m4(1), S(0) => m4(0));
76
77        bloco5 : conversor_somador port map(A(3) => m3(2), A(2) => m3(1), A(1) => m3(0), A(0)
      => SW(10),
78
79                                 S(3) => m5(3), S(2) => m5(2), S(1) => m5(1), S(0) => m5(0));
80
81        bloco6 : conversor_somador port map(A(3) => m4(2), A(2) => m4(1), A(1) => m4(0), A(0)
      => m5(3),
82
83                                 S(3) => m6(3), S(2) => m6(2), S(1) => m6(1), S(0) => m6(0));
84
85        bloco7 : conversor_somador port map(A(3) => m5(2), A(2) => m5(1), A(1) => m5(0), A(0)
      => SW(9),
86
87                                 S(3) => m7(3), S(2) => m7(2), S(1) => m7(1), S(0) => m7(0));
88
89        bloco8 : conversor_somador port map(A(3) => m6(2), A(2) => m6(1), A(1) => m6(0), A(0)
      => m7(3),
90
91                                 S(3) => m8(3), S(2) => m8(2), S(1) => m8(1), S(0) => m8(0));
92
93        bloco9 : conversor_somador port map(A(3) => m7(2), A(2) => m7(1), A(1) => m7(0), A(0)
      => SW(8),
94
95                                 S(3) => m9(3), S(2) => m9(2), S(1) => m9(1), S(0) => m9(0));
96
97        bloco10 : conversor_somador port map(A(3) => '0', A(2) => m4(3), A(1) => m6(3), A(0)
     => m8(3),
98
99                                 S(3) => m10(3), S(2) => m10(2), S(1) => m10(1), S(0) => m10(0
     ));
100
101       bloco11 : conversor_somador port map(A(3) => m8(2), A(2) => m8(1), A(1) => m8(0), A(0
     ) => m9(3),
102
103                                S(3) => m11(3), S(2) => m11(2), S(1) => m11(1), S(0) => m11(0
     ));
104
105       bloco12 : conversor_somador port map(A(3) => m9(2), A(2) => m9(1), A(1) => m9(0), A(0
     ) => SW(7),
106
107                                S(3) => m12(3), S(2) => m12(2), S(1) => m12(1), S(0) => m12(0
     ));
108
109       bloco13 : conversor_somador port map(A(3) => m10(2), A(2) => m10(1), A(1) => m10(0),
     A(0) => m11(3),
110
111                                S(3) => m13(3), S(2) => m13(2), S(1) => m13(1), S(0) => m13(0
     ));
112
113       bloco14 : conversor_somador port map(A(3) => m11(2), A(2) => m11(1), A(1) => m11(0),
     A(0) => m12(3),
```

```
114
115                             S(3) => m14(3), S(2) => m14(2), S(1) => m14(1), S(0) => m14(0
    ));
116
117         bloco15 : conversor_somador port map(A(3) => m12(2), A(2) => m12(1), A(1) => m12(0),
    A(0) => SW(6),
118
119                             S(3) => m15(3), S(2) => m15(2), S(1) => m15(1), S(0) => m15(0
    ));
120
121         bloco16 : conversor_somador port map(A(3) => m13(2), A(2) => m13(1), A(1) => m13(0),
    A(0) => m14(3),
122
123                             S(3) => m16(3), S(2) => m16(2), S(1) => m16(1), S(0) => m16(0
    ));
124
125         bloco17 : conversor_somador port map(A(3) => m14(2), A(2) => m14(1), A(1) => m14(0),
    A(0) => m15(3),
126
127                             S(3) => m17(3), S(2) => m17(2), S(1) => m17(1), S(0) => m17(0
    ));
128
129         bloco18 : conversor_somador port map(A(3) => m15(2), A(2) => m15(1), A(1) => m15(0),
    A(0) => SW(5),
130
131                             S(3) => m18(3), S(2) => m18(2), S(1) => m18(1), S(0) => m18(0
    ));
132
133         bloco19 : conversor_somador port map(A(3) => '0', A(2) => m10(3), A(1) => m13(3), A(0
    ) => m16(3),
134
135                             S(3) => m19(3), S(2) => m19(2), S(1) => m19(1), S(0) => m19(0
    ));
136
137         bloco20 : conversor_somador port map(A(3) => m16(2), A(2) => m16(1), A(1) => m16(0),
    A(0) => m17(3),
138
139                             S(3) => m20(3), S(2) => m20(2), S(1) => m20(1), S(0) => m20(0
    ));
140
141         bloco21 : conversor_somador port map(A(3) => m17(2), A(2) => m17(1), A(1) => m17(0),
    A(0) => m18(3),
142
143                             S(3) => m21(3), S(2) => m21(2), S(1) => m21(1), S(0) => m21(0
    ));
144
145         bloco22 : conversor_somador port map(A(3) => m18(2), A(2) => m18(1), A(1) => m18(0),
    A(0) => SW(4),
146
147                             S(3) => m22(3), S(2) => m22(2), S(1) => m22(1), S(0) => m22(0
    ));
148
149         bloco23 : conversor_somador port map(A(3) => m19(2), A(2) => m19(1), A(1) => m19(0),
    A(0) => m20(3),
150
151                             S(3) => m23(3), S(2) => m23(2), S(1) => m23(1), S(0) => m23(0
    ));
152
153         bloco24 : conversor_somador port map(A(3) => m20(2), A(2) => m20(1), A(1) => m20(0),
    A(0) => m21(3),
154
155                             S(3) => m24(3), S(2) => m24(2), S(1) => m24(1), S(0) => m24(0
    ));
156
157         bloco25 : conversor_somador port map(A(3) => m21(2), A(2) => m21(1), A(1) => m21(0),
    A(0) => m22(3),
```

```
158
159                             S(3) => m25(3), S(2) => m25(2), S(1) => m25(1), S(0) => m25(0
    ));
160
161         bloco26 : conversor_somador port map(A(3) => m22(2), A(2) => m22(1), A(1) => m22(0),
    A(0) => SW(3),
162
163                             S(3) => m26(3), S(2) => m26(2), S(1) => m26(1), S(0) => m26(0
    ));
164
165         bloco27 : conversor_somador port map(A(3) => m23(2), A(2) => m23(1), A(1) => m23(0),
    A(0) => m24(3),
166
167                             S(3) => m27(3), S(2) => m27(2), S(1) => m27(1), S(0) => m27(0
    ));
168
169         bloco28 : conversor_somador port map(A(3) => m24(2), A(2) => m24(1), A(1) => m24(0),
    A(0) => m25(3),
170
171                             S(3) => m28(3), S(2) => m28(2), S(1) => m28(1), S(0) => m28(0
    ));
172
173         bloco29 : conversor_somador port map(A(3) => m25(2), A(2) => m25(1), A(1) => m25(0),
    A(0) => m26(3),
174
175                             S(3) => m29(3), S(2) => m29(2), S(1) => m29(1), S(0) => m29(0
    ));
176
177         bloco30 : conversor_somador port map(A(3) => m26(2), A(2) => m26(1), A(1) => m26(0),
    A(0) => SW(2),
178
179                             S(3) => m30(3), S(2) => m30(2), S(1) => m30(1), S(0) => m30(0
    ));
180
181         bloco31 : conversor_somador port map(A(3) => m27(2), A(2) => m27(1), A(1) => m27(0),
    A(0) => m28(3),
182
183                             S(3) => bcd(16), S(2) => bcd(15), S(1) => bcd(14), S(0) => bcd
    (13));
184
185         bloco32 : conversor_somador port map(A(3) => m28(2), A(2) => m28(1), A(1) => m28(0),
    A(0) => m29(3),
186
187                             S(3) => bcd(12), S(2) => bcd(11), S(1) => bcd(10), S(0) => bcd
    (9));
188
189         bloco33 : conversor_somador port map(A(3) => m29(2), A(2) => m29(1), A(1) => m29(0),
    A(0) => m30(3),
190
191                             S(3) => bcd(8), S(2) => bcd(7), S(1) => bcd(6), S(0) => bcd(5
    ));
192
193         bloco34 : conversor_somador port map(A(3) => m30(2), A(2) => m30(1), A(1) => m30(0),
    A(0) => SW(1),
194
195                             S(3) => bcd(4), S(2) => bcd(3), S(1) => bcd(2), S(0) => bcd(1
    ));
196
197         bcd(19) <= m19(3);
198
199         bcd(18) <= m23(3);
200
201         bcd(17) <= m27(3);
202
203         bcd(0) <= SW(0);
204
```

```
205
206         DEZENA_MILHAR : bcd_display port map(X => bcd(19 downto 16), D => HEX4);
207
208         MILHAR : bcd_display port map(X => bcd(15 downto 12), D =>  HEX3);
209
210         CENTENA : bcd_display port map(X => bcd(11 downto 8), D =>  HEX2);
211
212         DEZENA : bcd_display port map(X => bcd(7 downto 4), D =>  HEX1);
213
214         UNIDADE : bcd_display port map(X => bcd(3 downto 0), D =>  HEX0);
215
216
217
218   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   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;
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity ContadorDP is
5     port (ld_cnt, clk, clr_cnt : in std_logic;
6           Q : out std_logic_vector (3 downto 0));
7   end ContadorDP;
8
9   architecture ckt of ContadorDP 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 : in std_logic;
21           Q : out std_logic_vector (7 downto 0));
22   end component;
23
24   component ComparadorMagnitude8bits is
25     port( X, Y : in std_logic_vector(7 downto 0);
26           Q : out std_logic);
27   end component;
28
29   signal soma_um, soma_out, reg_out, comp_64 : std_logic_vector(7 downto 0);
30   signal aux_not_clr_cnt, comp_out, gb : std_logic;
31
32   begin
33
34
35     aux_not_clr_cnt <= clr_cnt and not(comp_out);
36
37     soma_um <= "00000001";
38     comp_64 <= "00011111";
39
40     registrar : Registrador8bits port map(X=>soma_out,ld=>ld_cnt,clr=>aux_not_clr_cnt,clk=>
   clk,Q=>reg_out);
41
42     somar : Somador8bits port map(X=>soma_um,Y=>reg_out,ci=>'0',Q=>soma_out,cout=>gb);
43
44     comp : ComparadorMagnitude8bits port map(X=>comp_64,Y=>reg_out,Q=>comp_out);
45
46     Q(3) <= reg_out(3);
47     Q(2) <= reg_out(2);
48     Q(1) <= reg_out(1);
49     Q(0) <= reg_out(0);
50
51  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 : in std_logic;
21           Q : out std_logic_vector (7 downto 0));
22   end component;
23
24   component ComparadorMagnitude8bits is
25     port( X, Y : in std_logic_vector(7 downto 0);
26           Q : out std_logic);
27   end component;
28
29   signal soma_um, soma_out, reg_out, comp_64 : std_logic_vector(7 downto 0);
30   signal aux_not_clr_cnt, comp_out, gb, aux_gb : std_logic;
31
32   begin
33
34
35     aux_not_clr_cnt <= clr_cnt and not(comp_out);
36
37     soma_um <= "00000001";
38     comp_64 <= "01111111";
39
40     registrar : Registrador8bits port map(X=>soma_out,ld=>ld_cnt,clr=>aux_not_clr_cnt,clk=>
   clk,Q=>reg_out);
41
42     somar : Somador8bits port map(X=>soma_um,Y=>reg_out,ci=>'0',Q=>soma_out,cout=>gb);
43
44     comp : ComparadorMagnitude8bits port map(X=>comp_64,Y=>reg_out,Q=>comp_out);
45
46     Q(5) <= reg_out(5);
47     Q(4) <= reg_out(4);
48     Q(3) <= reg_out(3);
49     Q(2) <= reg_out(2);
50     Q(1) <= reg_out(1);
51     Q(0) <= reg_out(0);
52
53  end ckt;
```

```
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 <= igual;
79
80
81
82
83
84    end ckt;
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity CLK_Div is
5     port (clk_in : in std_logic;
6           clk_out : out std_logic);
7   end CLK_Div;
8
9   architecture logica of CLK_Div is
10
11    signal ax : std_logic;
12
13    begin
14      process(clk_in)
15         variable cnt: integer range 0 to 1350000 := 0;
16        begin
17          if (rising_edge(clk_in)) then
18             if (cnt = 1350000) then
19                cnt:=0;
20                ax <= not ax;
21             else
22                cnt:=cnt+1;
23             end if;
24          end if;
25      end process;
26      clk_out<=ax;
27  end logica;
```

```
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;
```

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


entity Registrador8bits is

  port (X : in  std_logic_vector (7 downto 0);

        ld, clr, clk : in std_logic;

        Q : out std_logic_vector (7 downto 0));

end Registrador8bits;


architecture ckt of Registrador8bits is


  component ffd is

    port (clk,D,P,C : in  std_logic;

                  q : out std_logic);

  end component;


  signal mux, aux : std_logic_vector(7 downto 0);


  begin

   mux(7) <= (X(7) and ld) or (aux(7) and not(ld));

   mux(6) <= (X(6) and ld) or (aux(6) and not(ld));

   mux(5) <= (X(5) and ld) or (aux(5) and not(ld));

   mux(4) <= (X(4) and ld) or (aux(4) and not(ld));

   mux(3) <= (X(3) and ld) or (aux(3) and not(ld));

   mux(2) <= (X(2) and ld) or (aux(2) and not(ld));

   mux(1) <= (X(1) and ld) or (aux(1) and not(ld));

   mux(0) <= (X(0) and ld) or (aux(0) and not(ld));


   ffd7 : ffd port map (clk => clk, D => mux(7), P => '1', C => clr, q => aux(7));

   ffd6 : ffd port map (clk => clk, D => mux(6), P => '1', C => clr, q => aux(6));

   ffd5 : ffd port map (clk => clk, D => mux(5), P => '1', C => clr, q => aux(5));

   ffd4 : ffd port map (clk => clk, D => mux(4), P => '1', C => clr, q => aux(4));

   ffd3 : ffd port map (clk => clk, D => mux(3), P => '1', C => clr, q => aux(3));

   ffd2 : ffd port map (clk => clk, D => mux(2), P => '1', C => clr, q => aux(2));
```

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

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



entity Somador8bits is

  port ( X, Y : in  std_logic_vector(7 downto 0);

           ci : in std_logic;

            Q : out std_logic_vector(7 downto 0);

            cout : out std_logic);

end Somador8bits;


architecture ckt of Somador8bits is


  signal co : std_logic_vector (7 downto 0);


  begin


    co(0) <= ci;

    Q(0)  <=  Y(0) xor X(0) xor co(0);

    co(1) <= (Y(0) and co(0)) or (X(0) and co(0)) or (X(0) and Y(0));

    Q(1)  <=  Y(1) xor X(1) xor co(1);

    co(2) <= (Y(1) and co(1)) or (X(1) and co(1)) or (X(1) and Y(1));

    Q(2)  <=  Y(2) xor X(2) xor co(2);

    co(3) <= (Y(2) and co(2)) or (X(2) and co(2)) or (X(2) and Y(2));

    Q(3)  <=  Y(3) xor X(3) xor co(3);

    co(4) <= (Y(3) and co(3)) or (X(3) and co(3)) or (X(3) and Y(3));

    Q(4)  <=  Y(4) xor X(4) xor co(4);

    co(5) <= (Y(4) and co(4)) or (X(4) and co(4)) or (X(4) and Y(4));

    Q(5)  <=  Y(5) xor X(5) xor co(5);

    co(6) <= (Y(5) and co(5)) or (X(5) and co(5)) or (X(5) and Y(5));

    Q(6)  <=  Y(6) xor X(6) xor co(6);

    co(7) <= (Y(6) and co(6)) or (X(6) and co(6)) or (X(6) and Y(6));

    Q(7)  <=  Y(7) xor X(7) xor co(7);

    cout <= (Y(7) and co(7)) or (X(7) and co(7)) or (X(7) and Y(7));


```

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