UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
DEPARTAMENTO DE ENGENHARIA DA COMPUTAÇÃO E AUTOMAÇÃO
DCA0212.1 - CIRCUITOS DIGITAIS - LABORATÓRIO
Prof. Emanoel Chaves e Prof. Kennedy Lopes
Aluno: Angelo Marcelino Cordeiro

ATIVIDADE AVALIATIVA I - CONTADOR *UP/DOWN*

INTRODUÇÃO

Este projeto foi realizado com o intuito de aperfeiçoar as habilidades aprendidas durante a primeira unidade do curso de Circuitos Digitais, especialmente sobre os assuntos de flip-flops e máquinas de estado (FSM).

Neste trabalho, foi dada a tarefa de projetar um circuito digital com a função de um contador no estilo *UP/DOWN* de três bits. Este contador realiza o incremento ou decremento contínuo de um número binário de três dígitos (000 a 111), ou seja, na base decimal, de 0 a 7.

Para o projeto do circuito bem como sua simulação, o *software* de *design* de circuitos digitais Quartus II da Altera foi utilizado. Já para a embarcação do projeto, foi utilizada a placa DE2, com o FPGA Cyclone II, também da Altera.

DESENVOLVIMENTO

O projeto se dividiu em três partes: projeto da FSM, implementação em VHDL e a simulação. Este relatório foi terminado antes da embarcação do projeto na placa FPGA.

A máquina de estados

Partindo do que foi pedido no comando deste trabalho, a implementação da FSM deste contador segue a seguinte lógica, com os *inputs* listados a seguir:

Inputs	Apelidos
$enable \cdot \overline{switch}$	C
$enable \cdot switch$	D
$\overline{enable} \cdot reset$	P
$\overline{enable} \cdot \overline{reset}$	Z

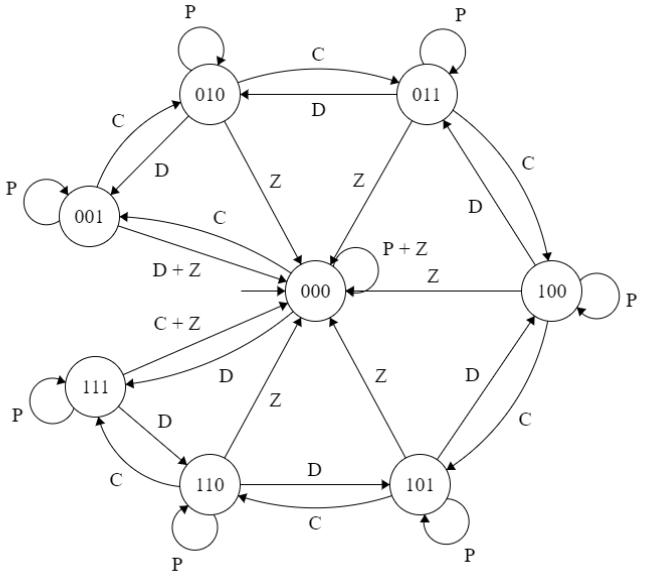

Figura 01: Máquina de estados

Dado a FSM acima é importante ressaltar que o *input* RESET é responsável por zerar o contador, porém ele realiza isto **somente** se o *input* ENABLE estiver barrado e após um pulso do CLOCK. A lógica desse RESET também implica que ele irá exercer sua função quando seu valor for 0. O *input* SWITCH também tem uma lógica especial, caso seu valor seja 0 e o ENABLE esteja em 1, ele fará com que o contador seja crescente, caso for 1, será decrescente.

Após a idealização da FSM, pôde-se descrever o projeto como um circuito de 4 *inputs*: CLOCK, SWITCH, ENABLE e RESET; e 3 *outputs*. Porém para exibir os números no display de 7 segmentos, será realizado mudanças no projeto quando formos descrevê-lo em VHDL.

Também após feita a FSM acima, montou-se uma Tabela Verdade que descrevesse o circuito, com 3 Flip-flops D, e as entradas e saídas do parágrafo anterior, omitindo o CLOCK.

n2	n1	n0	E	S	R	s2	s1	s0
0	0	0	0	X	0	0	0	0
0	0	0	0	X	1	0	0	0
0	0	0	1	0	X	0	0	1
0	0	0	1	1	X	1	1	1
0	0	1	0	X	0	0	0	0
0	0	1	0	X	1	0	0	1
0	0	1	1	0	X	0	1	0
0	0	1	1	1	X	0	0	0
0	1	0	0	X	0	0	0	0
0	1	0	0	X	1	0	1	0
0	1	0	1	0	X	0	1	1
0	1	0	1	1	X	0	0	1
0	1	1	0	X	0	0	0	0
0	1	1	0	X	1	0	1	1
0	1	1	1	0	X	1	0	0
0	1	1	1	1	X	0	1	0

n2	n1	n0	E	S	R	s2	s1	s0
1	0	0	0	X	0	0	0	0
1	0	0	0	X	1	1	0	0
1	0	0	1	0	X	1	0	1
1	0	0	1	1	X	0	1	1
1	0	1	0	X	0	0	0	0
1	0	1	0	X	1	1	0	1
1	0	1	1	0	X	1	1	0
1	0	1	1	1	X	1	0	0
1	1	0	0	X	0	0	0	0
1	1	0	0	X	1	1	1	0
1	1	0	1	0	X	1	1	1
1	1	0	1	1	X	1	0	1
1	1	1	0	X	0	0	0	0
1	1	1	0	X	1	1	1	1
1	1	1	1	0	X	0	0	0
1	1	1	1	1	X	1	1	0

Figura 02: Tabela Verdade

Sendo as variáveis do estado atual (n_i) os bits que representam a saída do sistema, e as variáveis do estado seguinte (s_i) as entradas dos registradores de estados, foram obtidas as seguintes equações:

- $s_2 = n_2 E(n_1 \otimes n_0) + \overline{n_1\, n_0} E(n_2 \otimes S) + n_1 n_0 E(\overline{n_2 \otimes S})$;
- $s_1 = ES(\overline{n_1 \otimes n_0}) + \overline{S} E(n_1 \otimes n_0)$;
- $s_0 = \overline{n_0} E$;

A partir dessas equações, começa o processo de implementação do circuito na linguagem VHDL.

Implementação em VHDL

Para representar o circuito, foi utilizado dois tipos de estruturas *component* para facilitar o uso de múltiplos flip-flops e elegantemente transformar as saídas binárias para números decimais em um display de sete segmentos.

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

entity ffd is
   port (d, clock : in std_logic;
         q : out std_logic);
end ffd;

architecture ffd of ffd is
signal estado : std_logic;
begin
   process (clock, d)
   begin
      if (clock = '1' and clock'EVENT) then
         if (d = '0') then estado <= '0';
         else estado <= '1';
         end if;
      end if;
   end process;
   q <= estado;
end ffd;
```

Figura 03: Declaração do Flip-flop D

Para tornar mais prática a abordagem do problema, ao invés de preparar a lógica para transformar a saída de 3 bits para um display de 7 segmentos, construí a entidade genérica *display* que tem como função transformar um número de 4 bits num número decimal em um display de 7 segmentos.

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

entity display is
   port (i : in std_logic_vector (3 downto 0);
         q : out std_logic_vector (6 downto 0));
end display;

architecture display of display is
signal l : std_logic_vector (6 downto 0);
begin
   l(0) <= i(3) or i(1) or (i(2) and i(0)) or (not i(2) and not i(0));
   l(1) <= not i(2) or (not i(1) and not i(0)) or (i(1) and i(0));
   l(2) <= i(2) or not i(1) or i(0);
   l(3) <= (not i(2) and not i(0)) or (i(1) and not i(0)) or (i(2) and not i(1) and i(0)) or (not i(2) and i(1))
           or i(3);
   l(4) <= (not i(2) and not i(0)) or (i(1) and not i(0));
   l(5) <= i(3) or (not i(1) and not i(0)) or (i(2) and not i(1)) or (i(2) and not i(0));
   l(6) <= i(3) or (i(2) and not i(1)) or (not i(2) and i(1)) or (i(1) and not i(0));
   q <= l;
end display;
```

Figura 04: Declaração da conversão para Display de 7 segmentos

Depois de feito os componentes do projeto, faltava implementá-los junto da lógica criada pela utilização de máquinas de estado. Porém durante essa implementação outros *insights* surgiram, primeiramente, seguindo a dica que o professor forneceu, utilizei a entrada ENABLE como parâmetro dos flip-flops D, assim, os estados só seriam registrados e passados adiante mediante ao valor dessa variável. Além disso, também percebi que, se juntamente a isso, adicionarmos o sinal Z (ou zr, no código) à lógica do CLOCK, os eventos de RESET estariam atrelados tanto ao valor do ENABLE quando aos pulsos do CLOCK, como desejado.

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

entity updown is
   port(enable, reset, switch, clk : in std_logic;
        coiso : out std_logic_vector(3 downto 0);
        q : out std_logic_vector(6 downto 0));
end updown;

architecture contador of updown is
component ffd is
   port (d, clock : in std_logic;
         q : out std_logic);
end component;
component display is
   port (i : in std_logic_vector (3 downto 0);
         q : out std_logic_vector (6 downto 0));
end component;
signal n, s : std_logic_vector (3 downto 0);
signal disp : std_logic_vector (6 downto 0);
signal zr : std_logic;
begin
   n(3) <= '0';
   s(3) <= '0';

   zr <= not enable and not reset;

   s(2) <= (n(2) and enable and (n(1) xor n(0))) or (not n(1) and not n(0) and enable and (n(2) xor switch)) or
           (n(1) and n(0) and enable and (n(2) xnor switch));
   s(1) <= (enable and switch and (n(1) xnor n(0))) or (enable and not switch and (n(1) xor n(0)));
   s(0) <= not n(0) and enable;

   ff2 : ffd port map (s(2), clk and (enable or zr), n(2));
   ff1 : ffd port map (s(1), clk and (enable or zr), n(1));
   ff0 : ffd port map (s(0), clk and (enable or zr), n(0));

   dsp : display port map (n, disp);
   coiso <= n;
   q <= not disp;
end contador;
```

Figura 05: Projeto em VHDL completo

Simulação

Por fim, foram feitos vários testes para verificar se o código acima estava de acordo com a teoria sobre o que ele poderia fazer.

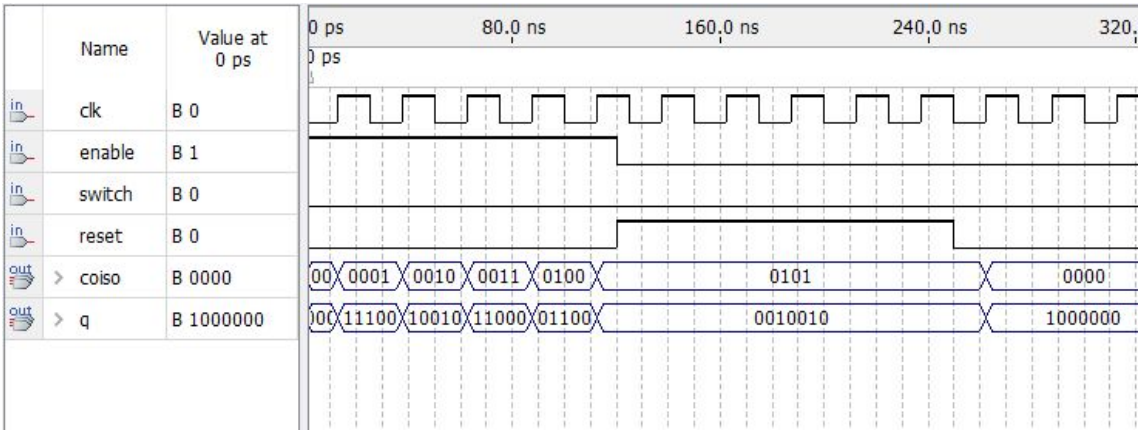

Figura 06: Exemplo de teste

Algumas curiosidades sobre o projeto: a saída “coiso” foi um método temporário de conseguir visualizar mais facilmente qual número estava saindo do sistema. Para a implementação na placa FPGA, não o utilizarei, pois a saída real é a que vai para o display de sete segmentos da placa DE2; a lógica de binário para decimal foi feita levando em conta um display que acende com entrada 1, porém para a placa, seus displays acendem com 0, portanto foi necessário negar a atribuição do componente display antes de mandá-lo para a saída.

CONCLUSÃO

Como visto no exemplo de simulação, até o presente momento, não foi encontrado nenhuma situação que o circuito desenvolvido falhasse. Portanto concluo que o projeto foi um sucesso. A atividade cumpriu o seu objetivo e testou meu conhecimento e perseverança para conseguir completá-lo dentro dos conformes. Este projeto também serviu para demonstrar o poder e a importância de se aprender a lidar com Projeto e Design de circuitos.