Laboratório Digital I

Relatório da experiência 2

Professores:

Paulo Sergio Cugnasca

Edson Midorikawa

Integrantes da T3BB3:

Arthur Pires da Fonseca - 10773096

Lucas Lopes de Paula Junior - 9344880

Introdução

Neste experimento, faremos a simulação de um circuito contador de 8 bits usando uma placa FPGA (Altera DE0-CV) para implementar os circuitos lógicos descritos em VHDL fornecidos de antemão à dupla.

Atividade 1 - Estudo de descrições VHDL

Foram analisadas duas descrições VHDL, os arquivos contador_163.vhd e contador8bits.vhd. Cada linha do código deles foi comentada, a fim de descrever qual a respectiva função.

contador_163.vhd

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity contador_163 is -- declaração da entidade contador_163 (imagine como uma "caixa preta")
   port (
        clock    : in  std_logic; -- entrada clock do tipo std_logic
        clr, ld  : in  std_logic; -- entradas clr (clear) e ld (load) do tipo std_logic
        ent, enp : in  std_logic; -- entradas ent e enp (enable T e P) do tipo std_logic
        D        : in  std_logic_vector (3 downto 0); -- entrada D (data) de 4 bits do tipo std_logic_vector
        Q        : out std_logic_vector (3 downto 0); -- saída Q de 4 bits do tipo std_logic_vector
        rco      : out std_logic -- saída rco (Ripple Carry Out) do tipo std_logic
   );
end contador_163; -- fim da declaração da entidade contador_163

architecture comportamental of contador_163 is -- início da declaração da arquitetura da entidade contador_163
  signal IQ: integer range 0 to 15; -- sinal (fio) do tipo integer assumindo valores de  0 a 15
begin

  process (clock,clr,ld,ent,enp,IQ) -- processo (sequencial) e lista de sensibilidade
                                    --(se um desses sinais mudar o processo é ativado)
  begin

    if clock'event and clock='1' then -- se for detectada uma borda de subida, então
      if clr = '0' then   -- se o clear estiver ativo baixo, então
         IQ <= 0;  -- o sinal IQ assume valor 0
      elsif ld = '0' then --se não (clr != '0'), se load for ativo baixo, então
         IQ <= to_integer(unsigned(D)); -- IQ recebe o valor da entrada D (conversão de tipos é necessária)
      elsif ent = '1' and enp = '1' then -- se não (ld != '0'), se ent e enp forem ativos alto, então
         if IQ = 15 then -- se o sinal IQ for 15, então
            IQ <= 0; --sinal IQ assume 0
         else -- caso contrário (IQ != 15)
            IQ <= IQ + 1; -- sinal IQ é incrementado de 1
        end if; -- fim das condições no caso de ent e enp estarem ativos alto
      else -- caso contrário (nenhuma das condições para clr, ld, ent e enp)
         IQ <= IQ; --sinal IQ continua o mesmo pois assume o valor do último estado (sem mudança)
      end if; -- fim das condições para as entradas
    end if; -- fim das condições caso em que a borda de subida é detectada

    if IQ = 15 and ENT = '1' --se o sinal IQ = 15 e ENT = 1
      then rco <= '1'; -- sinal rco assume valor 1 (usado para cascateamento)
    else -- caso contrário
      rco <= '0'; -- rco = 0
    end if; --fim da detecção de "overflow"

    Q <= std_logic_vector(to_unsigned(IQ, Q'length)); --saída Q assume o valor do sinal IQ
                                                      --(conversão de tipo e adaptação de comprimento
                                                      --são necessários)

  end process; --fim do processo
```

```
56        rco <= '0'; -- rco = 0
57      end if; --fim da detecção de "overflow"
58
59      Q <= std_logic_vector(to_unsigned(IQ, Q'length)); --saída Q assume o valor do sinal IQ
60                                                        --(conversão de tipo e adaptação de comprimento
61                                                        --são necessários)
62
63    end process; --fim do processo
64  end comportamental; -- fim da declaração da arquitetura comportamental
65
66
67  --1) A saída Q deve variar de 0 a 15. Quais linhas de código VHDL confirmam este intervalo de valores?
68  --       Linha 30 e também as 42 e 43 DESTE código.
69
70  --2) O sinal de CLEAR é síncrono e ativo em baixo. Quais linhas de código VHDL confirmam esta característica?
71  --       Linhas 36 e 37.
72
73  --3) Este componente é sensível a borda de subida do sinal de clock.
74  --Quais linhas de código VHDL confirmam esta característica?
75  --       Linha 36.
76
77  --4)Os sinais ENT e RCO devem ser usados para cascateamento de contadores.
78  --Quais linhas de código VHDL confirmam esta característica?
79  --       Linhas 52 até 56.
```

contador8bits.vhd

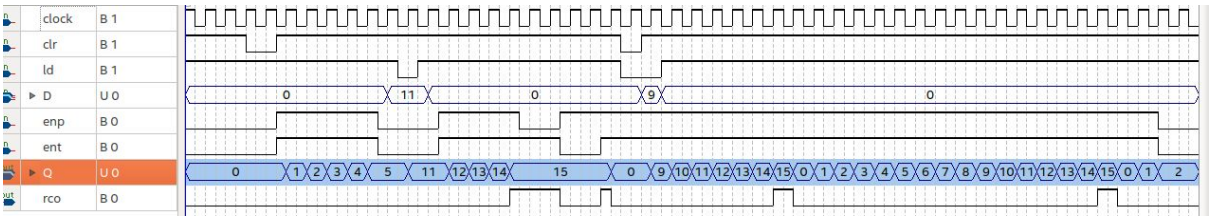
```
15  library IEEE;
16  use IEEE.std_logic_1164.all;
17  use IEEE.numeric_std.all;
18
19  entity contador8bits is --declaração da entidade contador8bits (imagine uma "caixa preta")
20     port (
21          clock : in  std_logic; --entrada clock do tipo std_logic
22          zera  : in  std_logic; --entrada zera do tipo std_logic
23          conta : in  std_logic; --entrada conta do tipo std_logic
24          Q     : out std_logic_vector (7 downto 0); --saída Q de 8 bits do tipo std_logic_vector
25          rco   : out std_logic -- saída rco do tipo std_logic
26     );
27  end contador8bits; --fim da declaração da entidade contador8bits
28
29  architecture estrutural of contador8bits is --declaração da arquitetura estrutural da entidade contador8bits
30
31    component contador_163 --declaração do componente contador_163 (ver arquivo contador_163.vhd para mais detalhes)
32      port (
33            clock    : in  std_logic;
34            clr, ld  : in  std_logic;
35            ent, enp : in  std_logic;
36            D        : in  std_logic_vector (3 downto 0);
37            Q        : out std_logic_vector (3 downto 0);
38            rco      : out std_logic
39      );
```

```
40  end component; --fim da declaração do componente contador_163
41
42    signal s_rco : std_logic; --sinal (fio) s_rco to tipo std_logic
43    signal s_Q   : std_logic_vector  (7 downto 0); --sinal (fio) s_Q de 8 bits do tipo std_logic_vector
44
45  begin
46    --CONT1 é uma instância do componente contador_163
47  CONT1: contador_163 port map ( clock=>clock, --conecta a entrada clock do componente CONT1 (contador_163) à entrada
48                                                --clock da entidade (contador8bits)
49           clr=>zera, --conecta a entrada clr do componente à entrada zera da entidade
50           ld=>'1', --liga o sinal lógico '1' à entrada ld do componente
51           ent=>'1', --liga o sinal lógico '1' à entrada ent do componente menos significativo
52           enp=>conta, --conecta a entrada enp do componente à entrada conta da entidade
53           D=>"1111", --liga o sinal "1111" (4 bits) à entrada D (4 bits) do componente CONT1
54           Q=>s_Q(3 downto 0), --conecta a saída Q (4 bits) do componente ao sinal s_Q (4 bits menos significativos)
55           rco=>s_rco --conecta a saída rco do componente menos significativo ao sinal s_rco
56         ); --fim da declaração do componente
57
58  --CONT2 é uma OUTRA instância do componente contador_163
59  CONT2: contador_163 port map ( clock=>clock, --conecta a entrada clock do componente CONT2 (contador_163) à entrada
60                                                --clock da entidade (contador8bits)
61           clr=>zera, --conecta a entrada clr do componente à entrada zera da entidade
62           ld=>'1', --liga o sinal lógico '1' à entrada ld do componente
63           ent=>s_rco, --liga o sinal s_rco à entrada ent do componente mais significativo
64           enp=>conta, --conecta a entrada enp do componente à entrada conta da entidade
```

```
65           D=>"1111", --liga o sinal "1111" (4 bits) à entrada D (4 bits) do componente CONT2
66           Q=>s_Q(7 downto 4), --contecta a saída Q (4 bits) do componente ao sinal s_Q (4 bits mais significativos)
67           rco=>rco --conecta a saída rco do componente mais significativo à saída rco da arquitetura
68         );
69
70    Q <= s_Q; --saída Q da entidade recebe o sinal s_Q de 8 bits
71
72  end estrutural; --fim da arquitetura estrutural
73
74
75  --1) A saída Q do contador de 8 bits deve variar de 0 a 255. Como isto pode ser confirmado pelo grupo?
76  --    Os 8 bits do contador são controlados pelos componentes CONT1 (4 bits menos significativos) e
77  --    CONT2 (4 bits mais significativos). Como são instâncias do componente contador_163, cada um deles
78  --    assume individualmente valores de 0 a F (base hexadecimal). Por tanto, o maior número representável
79  --    é FF, ou seja, 15*(16^1) + 15*(16^0) = 255.
80
81  --2) O contador de 8 bits é composto pelo cascateamento de dois contadores 74163.
82  --Qual componente interno da descrição se refere ao dígito hexadecimal mais significativo (CONT1 ou CONT2)?
83  --     De acordo com a descrição, e como já mencionado antes,
84  --     CONT2 é responsável pelo dígito hexadecimal mais significativo.
85
86  --3) Os sinais ENT e RCO são usados para cascateamento dos contadores. Quais linhas de código VHDL estrutural
87  --mostram esta ligação de sinais? Quais sinais internos VHDL são usados neste cascateamento?
88  --    As linhas 50, 54, 61 e 65 DESTE código mostram esta ligação.
89  --    O sinal interno usado no cascateamento é o s_rco do tipo std_logic.
```

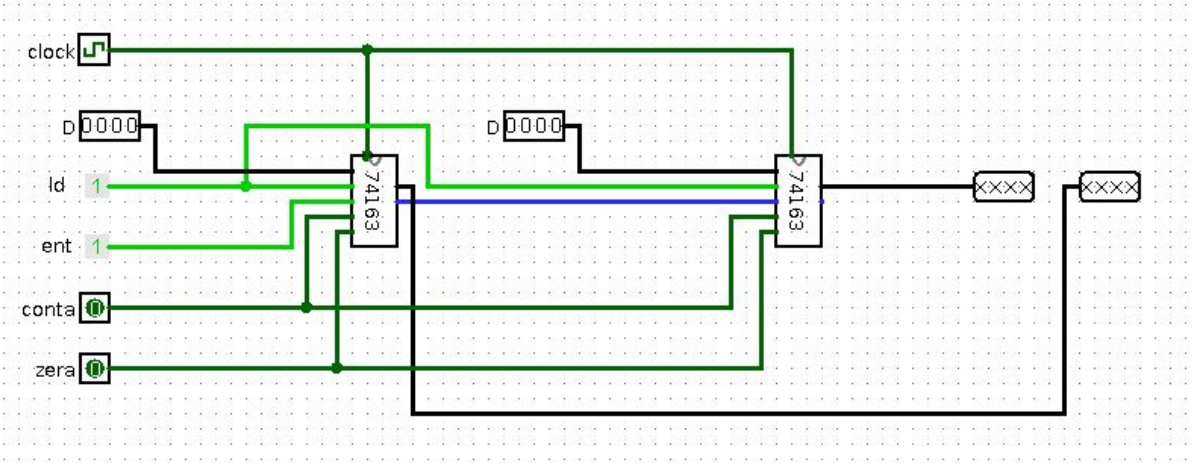
Atividade 2 - Simulação de circuitos em VHDL

Usando o *software* Intel Quartus Prime, simulamos o comportamento do circuito descrito pelo arquivo contador_163.vhd, os resultados são mostrados abaixo.



A simulação representou adequadamente a descrição VHDL, o sinal RCO apenas é ativado quando a saída Q vale 15 e o sinal ENT é 1.

Resumo de funcionamento do circuito: O contador de 8 bits é feito a partir de 2 contadores 74163 de 4 bits, que ficam cascateados. Cada contador por si só afere valores de 0 a 15 (decimal) e quando cascateados em dupla de 0 a 255. O efeito da cascata possibilita que um contador controle os 4 bits menos significativos e o outro os 4 mais significativos. O contador de 8 bits é síncrono e sensível à borda de subida do clock, assim como seus componentes de 4 bits do qual é formado.



Atividade 3 - Simulação e síntese do Circuito Contador de 8 bits

Esta é a relação entre os sinais da descrição VHDL e os componentes da placa FPGA que usaremos na experiência:

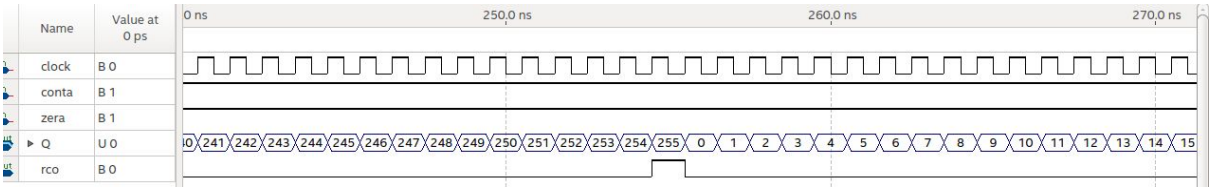
sinal	pino
clock	botão KEY0
zera	chave SW0
conta	chave SW1
Q	*leds* LEDR[0] a LEDR[7]
rco	*led* LEDR[9]

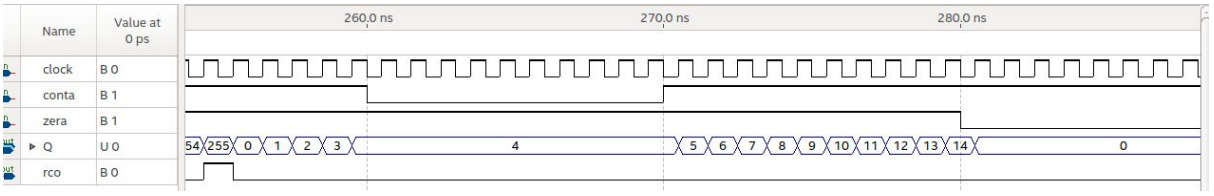
Plano de testes

Teste	Passos	Resultado observado
zerar saída Q	~CLR (L), CLK	todos os LEDs se apagaram
contar de 0 a 15	~CLR (H), ~LOAD (H), ENP (H), ENT (H),	configuração dos LEDs -> 00001111
	15 * CLK	
contar de 15 a 20	~CLR (H), ~LOAD (H), ENP (H), ENT (H),	configuração dos LEDs -> 00010100
	5 * CLK	
desativar ENP	~ENP(L), CLK	nenhuma mudança nos LEDs
desativar ENT	~ENT(L), CLK	todos os LEDs se apagaram
reativar ENP e ENT	~ENP(H), ENT(H), CLK	configuração dos LEDs -> 00000001
contar até 255	~CLR (H), ~LOAD (H), ENP (H), ENT (H),	todos os LEDs se ligaram
	5 * CLK	
contar mais uma vez	CLK	todos os LEDs se apagaram

Com os resultados observados, podemos concluir que o funcionamento do circuito emulado pelo FPGA está funcionando da forma esperada.

A execução do plano de testes resultou nas seguintes formas de onda:





Atividade 4 - Desafio

Foi solicitada uma modificação no circuito da experiência: criar um arquivo VHDL capaz de descrever um circuito contador capaz de ligar os *displays* da placa FPGA.

Criamos o arquivo exp2_desafio.vhd para descrever o novo circuito, adicionamos duas entradas nele para permitir a inicialização de valores através de um sinal externo.

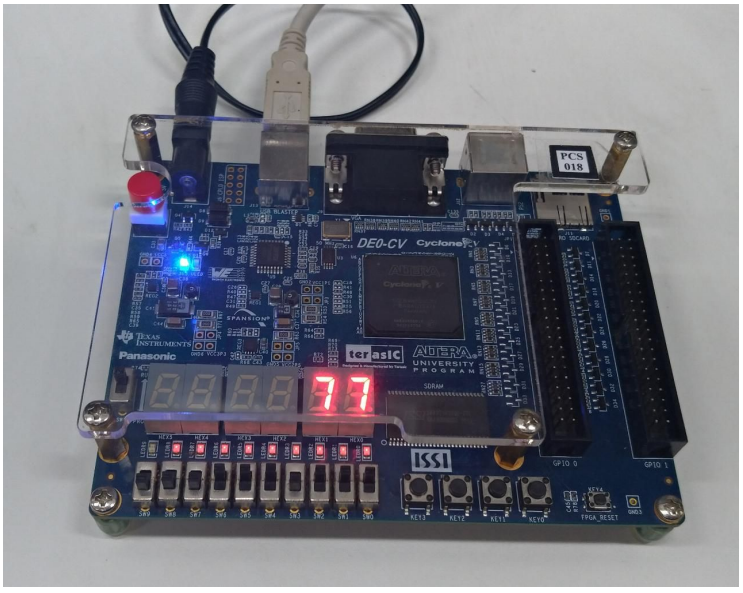
SImulamos tanto a contagem o como o carregamento dos valores no Intem Quartus Prime. Para isso foi necessária a modificação da entidade do VHDL contador8bits e do exp2_desafio. As mudanças e as simulaçẽos são amostradas abaixo:

contador8bits.vhd (modificado)

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity contador8bits is --declaração da entidade contador8bits (imagine uma "caixa preta")
   port (
        clock : in  std_logic; --entrada clock do tipo std_logic
        zera  : in  std_logic; --entrada zera do tipo std_logic
        conta : in  std_logic; --entrada conta do tipo std_logic
        Q     : out std_logic_vector (7 downto 0); --saída Q de 8 bits do tipo std_logic_vector
        rco   : out std_logic; -- saída rco do tipo std_logic
        load : in std_logic; -- entrada load do tipo std_logic
        init : in std_logic_vector(7 downto 0) --entrada init de 8 bits do tipo std_logic_vector
   );
end contador8bits; --fim da declaração da entidade contador8bits

architecture estrutural of contador8bits is --declaração da arquitetura estrutural da entidade contador8bits

  component contador_163 --declaração do componente contador_163 (ver arquivo contador_163.vhd para mais detalhes)
    port (
          clock    : in  std_logic;
          clr, ld  : in  std_logic;
          ent, enp : in  std_logic;
          D        : in  std_logic_vector (3 downto 0);
          Q        : out std_logic_vector (3 downto 0);
          rco      : out std_logic
    );
end component; --fim da declaração do componente contador_163
  signal s_rco : std_logic; --sinal (fio) s_rco to tipo std_logic
  signal s_Q   : std_logic_vector  (7 downto 0); --sinal (fio) s_Q de 8 bits do tipo std_logic_vector
  signal initialValue : std_logic_vector(7 downto 0);

  begin

  initialValue <= init;

    --CONT1 é uma instância do componente contador_163
  CONT1: contador_163 port map ( clock=>clock, --conecta a entrada clock do componente CONT1 (contador_163) à entrada
                                               --clock da entidade (contador8bits)
           clr=>zera, --conecta a entrada clr do componente à entrada zera da entidade
          -- ld=>'1', --liga o sinal lógico '1' à entrada ld do componente
          ld => load,
           ent=>'1', --liga o sinal lógico '1' à entrada ent do componente menos significativo
           enp=>conta, --conecta a entrada enp do componente à entrada conta da entidade
           D=>initialValue(3 downto 0), --liga o sinal "1111" (4 bits) à entrada D (4 bits) do componente CONT1
           Q=>s_Q(3 downto 0), --conecta a saída Q (4 bits) do componente ao sinal s_Q (4 bits menos significativos)
           rco=>s_rco --conecta a saída rco do componente menos significativo ao sinal s_rco
         ); --fim da declaração do componente

  --CONT2 é uma OUTRA instância do componente contador_163
  CONT2: contador_163 port map ( clock=>clock, --conecta a entrada clock do componente CONT2 (contador_163) à entrada
                                               --clock da entidade (contador8bits)
           clr=>zera, --conecta a entrada clr do componente à entrada zera da entidade
          -- ld=>'1', --liga o sinal lógico '1' à entrada ld do componente
           ld => load,
           ent=>s_rco, --liga o sinal s_rco à entrada ent do componente mais significativo
           enp=>conta, --conecta a entrada enp do componente à entrada conta da entidade
           D=>initialValue(7 downto 4), --liga o sinal "1111" (4 bits) à entrada D (4 bits) do componente CONT2
           Q=>s_Q(7 downto 4), --contecta a saída Q (4 bits) do componente ao sinal s_Q (4 bits mais significativos)
           rco=>rco --conecta a saída rco do componente mais significativo à saída rco da arquitetura
         );

    Q <= s_Q; --saída Q da entidade recebe o sinal s_Q de 8 bits

end estrutural; --fim da arquitetura estrutural

```

exp2_desafio.vhd

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity exp2_desafio is
      port (
            clockxp    :  in std_logic;
            zeraxp     :  in std_logic;
            contador :  in std_logic;
            display0 : out std_logic_vector(6 downto 0);
            display1 : out std_logic_vector(6 downto 0);
            rcoxp      : out std_logic;
            loadxp : in std_logic;
            inicializador : in std_logic_vector(7 downto 0)
      );
end exp2_desafio;

architecture arch of exp2_desafio is
   component contador8bits is
   port (
        clock : in  std_logic;
        zera  : in  std_logic;
        conta : in  std_logic;
        Q     : out std_logic_vector (7 downto 0);
        rco   : out std_logic;
        load : in std_logic;
        init : in std_logic_vector(7 downto 0)
   );
   end component;
    component hexa7seg is
        port (
            hexa : in std_logic_vector(3 downto 0);
            sseg : out std_logic_vector(6 downto 0)
        );
    end component;

    signal menos, mais : std_logic_vector(3 downto 0);
    signal seg7_0, seg7_1 : std_logic_vector(6 downto 0);
    signal Qaux : std_logic_vector(7 downto 0);

begin

    mais <= Qaux(7 downto 4);
    menos <= Qaux(3 downto 0);
    --Qaux(7 downto 4) <= mais;
    --Qaux(3 downto 0) <= menos;

    cont8: contador8bits port map (  clock => clockxp,
                                     zera => zeraxp,
                                     conta => contador,
                                     Q => Qaux,
                                     rco => rcoxp,
                                     load => loadxp,
                                    init => inicializador
                                  );

    hex7_0: hexa7seg port map ( hexa => menos,
                                sseg => seg7_0
                               );

    hex7_1: hexa7seg port map ( hexa => mais,
                                sseg => seg7_1
                               );

    display0 <= seg7_0;
    display1 <= seg7_1;

end architecture;
```



Simulações

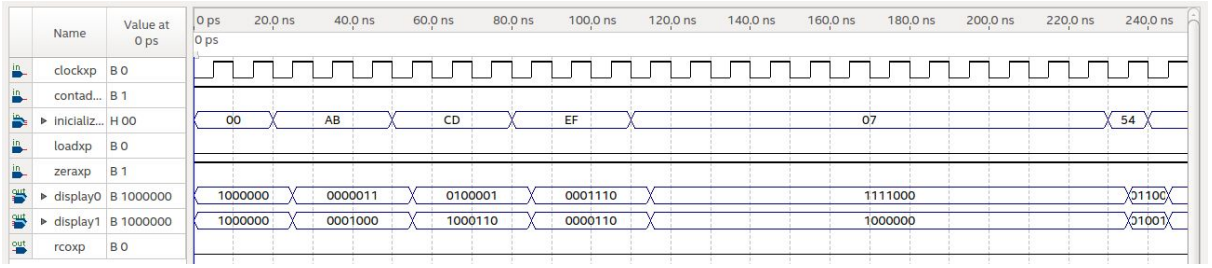


Imagem: carregamento de valores

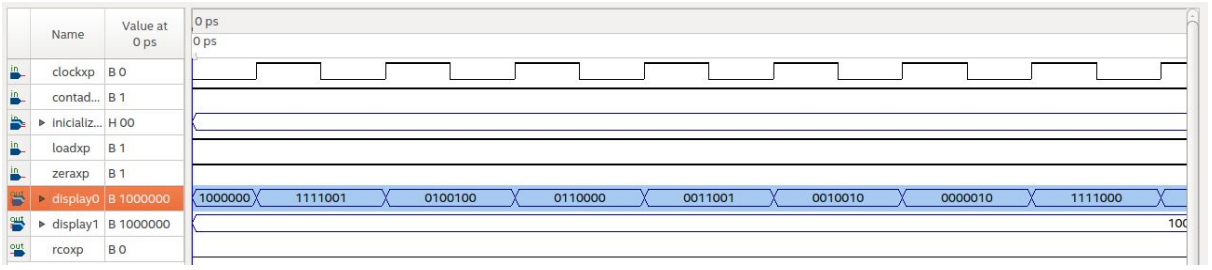


Imagem: contagem de valores (display menos significativo em evidência)

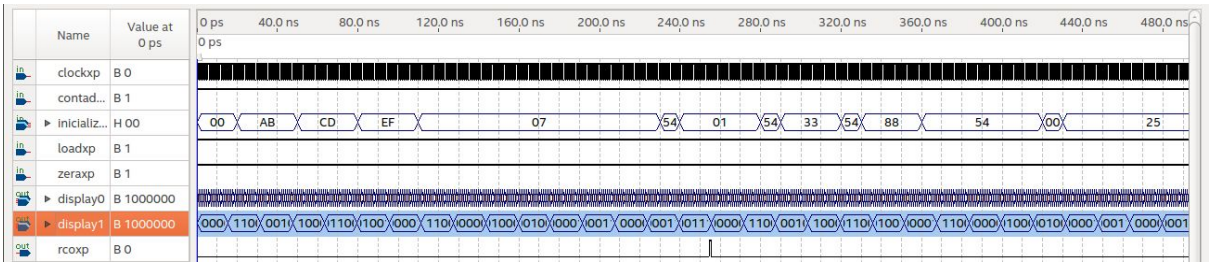


Imagem: contagem de valores (display mais significativo em evidência)

Mudamos um pouco a seleção de pinos da atividade anterior e adicionamos uma comunicação com os sinais de carregamento:

sinal	nome	número do pino
rcoxp	LEDR[9]	PIN_L1
display0[0]	HEX00	PIN_U21
display0[1]	HEX01	PIN_V21
display0[2]	HEX02	PIN_W22
display0[3]	HEX03	PIN_W21
display0[4]	HEX04	PIN_Y22
display0[5]	HEX05	PIN_Y21
display0[6]	HEX06	PIN_AA22
display1[0]	HEX10	PIN_AA20
display1[1]	HEX11	PIN_AB20
display1[2]	HEX12	PIN_AA19
display1[3]	HEX13	PIN_AA18
display1[4]	HEX14	PIN_AB18
display1[5]	HEX15	PIN_AA17
display1[6]	HEX16	PIN_U22
clockxp	KEY0	PIN_U7
zera	KEY1	PIN_W9
inicializador[0]	SW0	PIN_U13
inicializador[1]	SW1	PIN_V13
inicializador[2]	SW2	PIN_T13
inicializador[3]	SW3	PIN_T12
inicializador[4]	SW4	PIN_AA15
inicializador[5]	SW5	PIN_AB15
inicializador[6]	SW6	PIN_AA14
inicializador[7]	SW7	PIN_AA13
loadxp	SW8	PIN_AB13
contador	SW9	PIN_AB12

Teste	Passos	Resultado observado
zerar saída	~zera (L), clockxp	Display com valor 0x00
contar de 0 a 15	~zera (H), ~loadxp (H), contador (H), 15 * clockxp	Display com valor 0x0F
contar de 15 a 20	~zera (H), ~loadxp (H), contador (H), 5 * clockxp	Display com valor 0x14
inicializar valor 0xF7	~loadxp (L) ~inicializador = (LLLL_LHHH), clockxp	Display com valor 0x07
contar até 255	~zera (H), ~load (H), contador (H), 255 * clockxp	Display com valor 0xFF
contar mais uma vez	clockxp	Display com valor 0x00

Após o experimento

Resultados alcançados

Conseguimos entender como funciona o 74163 e como configurá-lo afim de formar um contador com maior capacidade à partir do cascateamento de 2 ou mais deles. O desafio nos proporcionou a oportunidade de implementar o hardware na placa FPGA de forma ímpar, modificando algumas coisas como fora mostrado neste relatório. Tudo funcionou como o esperado e em concordância com as simulações.

Pontos positivos

Do nosso ponto de vista estamos entendendo, na prática, como funciona o projeto e implementação de hardware, bem como as etapas de simulação, teste e depuração, além de é claro, estudar o funcionamento e operação de CIs e placas FPGAs.

Lições aprendidas

A prática, organização e planejamento para se projetar um circuito digital (ou hardware) são coisas que andam juntas e vão além da técnica/teoria sobre como um componente funciona ou como combiná-los a fim de projetar outras coisas à partir destes.

“Se o seu esforço é pequeno você provavelmente não está focando na OPORTUNIDADE. Você provavelmente está focando na OBRIGAÇÃO.”