Relatório

Tópico: Aula 4 & 5 - Contadores
Grupo: Frederico Scheffel Oliveira 15452718
Leonardo Massuhiro Sato 15469108
Pedro Henrique Perez Dias 15484075
Github: https://github.com/johncleyton/SistemasDigitais/tree/main/Aula_4-5

Parte 1

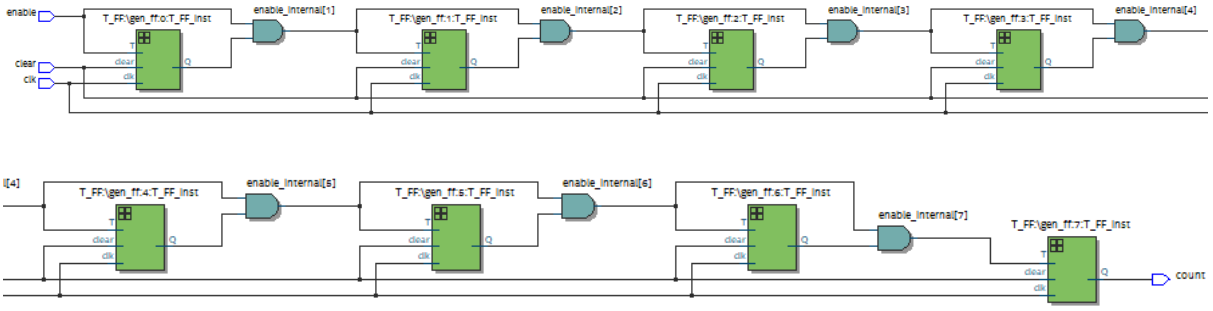
1.1)

Código VHDL:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity T_FF is
    Port (
    clk     : in  std_logic;
    clear   : in  std_logic;  -- Synchronous clear
    T       : in  std_logic;  -- Enable to toggle
    Q       : out std_logic;
    Q_n     : out std_logic
    );
end T_FF;

architecture Behavioral of T_FF is
    signal Q_int : std_logic := '0';
begin
    process(clk, clear)
    begin
    if clear = '0' then
    Q_int <= '0';
    elsif rising_edge(clk) then
    if T = '1' then
    Q_int <= not Q_int;
    end if;
    end if;
    end process;

    Q     <= Q_int;
    Q_n   <= not Q_int;
end Behavioral;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Counter8bit is
    Port (
    clk   : in  std_logic;
    clear : in  std_logic;
    enable : in  std_logic;
    count : out std_logic
    );
end Counter8bit;

architecture Behavioral of Counter8bit is
    signal Q   : std_logic_vector(7 downto 0);
```

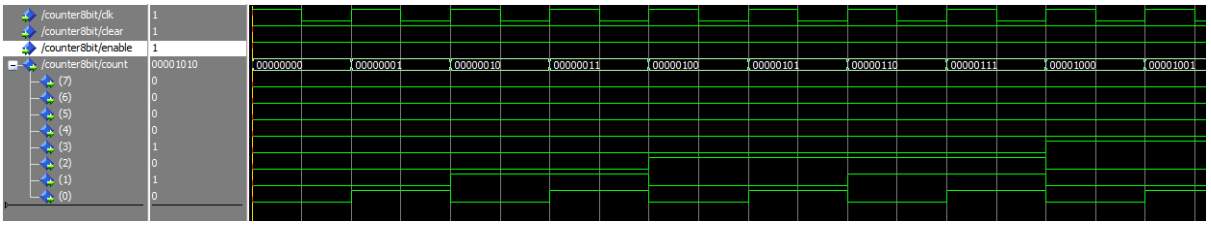
```
    signal Q_n   : std_logic_vector(7 downto 0);
    signal enable_internal : std_logic_vector(7 downto 0);

begin
    -- Generate enable signals for each flip-flop
    enable_internal(0) <= enable;  -- First flip-flop
    enable_internal(1) <= Q(0) and enable_internal(0);
    enable_internal(2) <= Q(1) and enable_internal(1);
    enable_internal(3) <= Q(2) and enable_internal(2);
    enable_internal(4) <= Q(3) and enable_internal(3);
    enable_internal(5) <= Q(4) and enable_internal(4);
    enable_internal(6) <= Q(5) and enable_internal(5);
    enable_internal(7) <= Q(6) and enable_internal(6);

    gen_ff: for i in 0 to 7 generate
    T_FF_inst : entity work.T_FF
    Port map (
    clk         => clk,
    clear  => clear,
    T => enable_internal(i),
    Q           => Q(i),
    Q_n         => Q_n(i)
    );
    end generate;

    count <= Q(7);

end Behavioral;
```

RTL Viewer - Contador de 8 bits



1.2)

Simulação do contador de 8 bits



1.3)

Código VHDL do display de 7 segmentos

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

Library work;

ENTITY display_7seg IS
   PORT
   (
            input1 :  IN  STD_LOGIC;
            input2 :  IN  STD_LOGIC;
            input3 :  IN  STD_LOGIC;
            input4 :  IN  STD_LOGIC;
            output0 :  OUT  STD_LOGIC;
            output1 :  OUT  STD_LOGIC;
            output2 :  OUT  STD_LOGIC;
            output3 :  OUT  STD_LOGIC;
            output4 :  OUT  STD_LOGIC;
            output5 :  OUT  STD_LOGIC;
            output6 :  OUT  STD_LOGIC
   );
END display_7seg;

ARCHITECTURE bdf_type OF display_7seg IS

SIGNAL    FIO1 :  STD_LOGIC;
SIGNAL    FIO2 :  STD_LOGIC;
SIGNAL    FIO3 :  STD_LOGIC;
SIGNAL    FIO4 :  STD_LOGIC;
SIGNAL    fio_final0 :  STD_LOGIC;
SIGNAL    fio_final1 :  STD_LOGIC;
SIGNAL    fio_final2 :  STD_LOGIC;
SIGNAL    fio_final3 :  STD_LOGIC;
SIGNAL    fio_final4 :  STD_LOGIC;
SIGNAL    fio_final5 :  STD_LOGIC;
SIGNAL    fio_final6 :  STD_LOGIC;
SIGNAL    fio_numero0 :  STD_LOGIC;
SIGNAL    fio_numero1 :  STD_LOGIC;
SIGNAL    fio_numero2 :  STD_LOGIC;
SIGNAL    fio_numero3 :  STD_LOGIC;
SIGNAL    fio_numero4 :  STD_LOGIC;
SIGNAL    fio_numero5 :  STD_LOGIC;
SIGNAL    fio_numero6 :  STD_LOGIC;
SIGNAL    fio_numero7 :  STD_LOGIC;
SIGNAL    fio_numero8 :  STD_LOGIC;
SIGNAL    fio_numero9 :  STD_LOGIC;
SIGNAL    fio_numeroA :  STD_LOGIC;
SIGNAL    fio_numeroB :  STD_LOGIC;
SIGNAL    fio_numeroC :  STD_LOGIC;
SIGNAL    fio_numeroD :  STD_LOGIC;
SIGNAL    fio_numeroE :  STD_LOGIC;
SIGNAL    fio_numeroF :  STD_LOGIC;
SIGNAL    NOT1 :  STD_LOGIC;
SIGNAL    NOT2 :  STD_LOGIC;
SIGNAL    NOT3 :  STD_LOGIC;
SIGNAL    NOT4 :  STD_LOGIC;
SIGNAL    SYNTHESIZED_WIRE_0 :  STD_LOGIC;
```

```
SIGNAL    SYNTHESIZED_WIRE_1 :  STD_LOGIC;


BEGIN

NOT1 <= NOT(FIO1);
NOT2 <= NOT(FIO2);
NOT3 <= NOT(FIO3);
NOT4 <= NOT(FIO4);


fio_final0 <= fio_numero1 OR fio_numeroB OR fio_numeroD OR fio_numero4;
fio_final1 <= fio_numero5 OR fio_numeroB OR fio_numero6 OR fio_numeroC OR fio_numeroE OR fio_numeroF;
fio_final2 <= fio_numero2 OR fio_numeroE OR fio_numeroF OR fio_numeroC;
SYNTHESIZED_WIRE_0 <= fio_numero4 OR fio_numero1;
fio_final3 <= SYNTHESIZED_WIRE_0 OR fio_numeroA OR fio_numeroF OR fio_numero7;
fio_final4 <= fio_numero1 OR fio_numero4 OR fio_numero3 OR fio_numero5 OR fio_numero7 OR fio_numero9;
SYNTHESIZED_WIRE_1 <= fio_numero2 OR fio_numero1;
fio_final5 <= SYNTHESIZED_WIRE_1 OR fio_numero7 OR fio_numeroD OR fio_numero3;
fio_final6 <= fio_numero0 OR fio_numero7 OR fio_numeroC OR fio_numero1;


fio_numero0 <= NOT1 AND NOT2 AND NOT3 AND NOT4;
fio_numero1 <= FIO1 AND NOT2 AND NOT3 AND NOT4;
fio_numero2 <= NOT1 AND FIO2 AND NOT3 AND NOT4;
fio_numero3 <= FIO1 AND FIO2 AND NOT3 AND NOT4;
fio_numero4 <= NOT1 AND NOT2 AND FIO3 AND NOT4;
fio_numero5 <= FIO1 AND NOT2 AND FIO3 AND NOT4;
fio_numero6 <= NOT1 AND FIO2 AND FIO3 AND NOT4;
fio_numero7 <= FIO1 AND FIO2 AND FIO3 AND NOT4;
fio_numero9 <= FIO1 AND NOT2 AND NOT3 AND FIO4;
fio_numeroA <= NOT1 AND FIO2 AND NOT3 AND FIO4;
fio_numeroB <= FIO1 AND FIO2 AND NOT3 AND FIO4;
fio_numeroC <= NOT1 AND NOT2 AND FIO3 AND FIO4;
fio_numeroD <= FIO1 AND NOT2 AND FIO3 AND FIO4;
fio_numeroE <= NOT1 AND FIO2 AND FIO3 AND FIO4;
fio_numeroF <= FIO1 AND FIO2 AND FIO3 AND FIO4;


output0 <= fio_final0;
FIO1 <= input1;
FIO2 <= input2;
FIO3 <= input3;
FIO4 <= input4;
output1 <= fio_final1;
output2 <= fio_final2;
output3 <= fio_final3;
output4 <= fio_final4;
output5 <= fio_final5;
output6 <= fio_final6;

END bdf_type;
```

Código completo da parte 1 - Counter8bit + display_7seg:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity T_FF is
	Port (
	clk		: in  std_logic;
	clear  : in  std_logic;  -- Synchronous clear
	T		 : in  std_logic;  -- Enable to toggle
	Q		: out std_logic;
	Q_n		: out std_logic
	);
end T_FF;

architecture Behavioral of T_FF is
	signal Q_int : std_logic := '0';
begin
	process(clk, clear)
	begin
	if clear = '0' then
	Q_int <= '0';
	elsif rising_edge(clk) then
	if T = '1' then
	Q_int <= not Q_int;
	end if;
	end if;
	end process;

	Q		<= Q_int;
	Q_n   <= not Q_int;
end Behavioral;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Counter8bit is
	Port (
	clk   : in  std_logic;
	clear : in  std_logic;
	enable : in  std_logic;
	count : out std_logic_vector(7 downto 0);
	  hex00 : out std_logic;
	  hex01 : out std_logic;
	  hex02 : out std_logic;
	  hex03 : out std_logic;
	  hex04 : out std_logic;
	  hex05 : out std_logic;
	  hex06 : out std_logic;
	  hex10 : out std_logic;
	  hex11 : out std_logic;
	  hex12 : out std_logic;
	  hex13 : out std_logic;
	  hex14 : out std_logic;
	  hex15 : out std_logic;
	  hex16 : out std_logic
	);
end Counter8bit;

architecture Behavioral of Counter8bit is
```

```
        signal Q   : std_logic_vector(7 downto 0);
        signal Q_n   : std_logic_vector(7 downto 0);
        signal enable_internal : std_logic_vector(7 downto 0);

    component display_7seg is
    Port
    (
            input1 :  IN  STD_LOGIC;
            input2 :  IN  STD_LOGIC;
            input3 :  IN  STD_LOGIC;
            input4 :  IN  STD_LOGIC;
            output0 :  OUT  STD_LOGIC;
            output1 :  OUT  STD_LOGIC;
            output2 :  OUT  STD_LOGIC;
            output3 :  OUT  STD_LOGIC;
            output4 :  OUT  STD_LOGIC;
            output5 :  OUT  STD_LOGIC;
            output6 :  OUT  STD_LOGIC
    );
    end component;

begin
        -- Generate enable signals for each flip-flop
        enable_internal(0) <= enable;  -- First flip-flop
        enable_internal(1) <= Q(0) and enable_internal(0);
        enable_internal(2) <= Q(1) and enable_internal(1);
        enable_internal(3) <= Q(2) and enable_internal(2);
        enable_internal(4) <= Q(3) and enable_internal(3);
        enable_internal(5) <= Q(4) and enable_internal(4);
        enable_internal(6) <= Q(5) and enable_internal(5);
        enable_internal(7) <= Q(6) and enable_internal(6);

        -- Instantiate 8 T flip-flops
        gen_ff: for i in 0 to 7 generate
        T_FF_inst : entity work.T_FF
        Port map (
        clk         => clk,
        clear  => clear,
        T => enable_internal(i),
        Q           => Q(i),
        Q_n         => Q_n(i)
        );
        end generate;

    display_7_min : entity work.display_7seg
                                Port map (
                                        input1 => Q(0),
                                        input2 => Q(1),
                                        input3 => Q(2),
                                        input4 => Q(3),
                                        output0 => hex00,
                                        output1 => hex01,
                                        output2 => hex02,
                                        output3 => hex03,
                                        output4 => hex04,
                                        output5 => hex05,
                                        output6 => hex06
                                );
```

```
        display_7_max : entity work.display_7seg
                Port map (
                        input1 => Q(4),
                        input2 => Q(5),
                        input3 => Q(6),
                        input4 => Q(7),
                        output0 => hex10,
                        output1 => hex11,
                        output2 => hex12,
                        output3 => hex13,
                        output4 => hex14,
                        output5 => hex15,
                        output6 => hex16
                );

        count <= Q;

end Behavioral;
```

1.5)

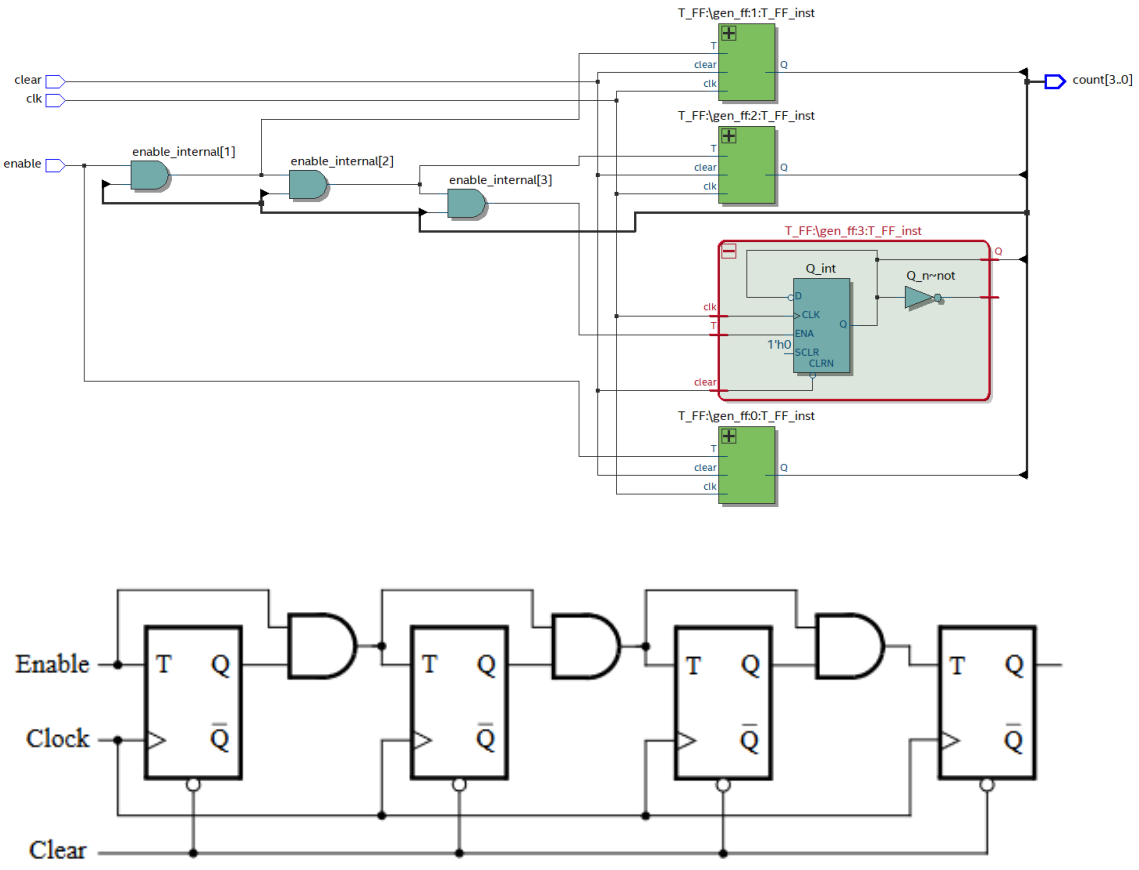


Figure 1: A 4-bit counter.

Ao analisarmos o circuito de 4 bits pelo RTL viewer, é perceptível que não houve uma grande mudança no próprio sistema de portas lógicas, com exceção do nosso output ser mais compacto, utilizando um vetor de outputs, e também pelo modo de como os fios foram

conectados, que parece mais caótico no RTL viewer, apesar de manter a mesma função do sistema.

Parte 2

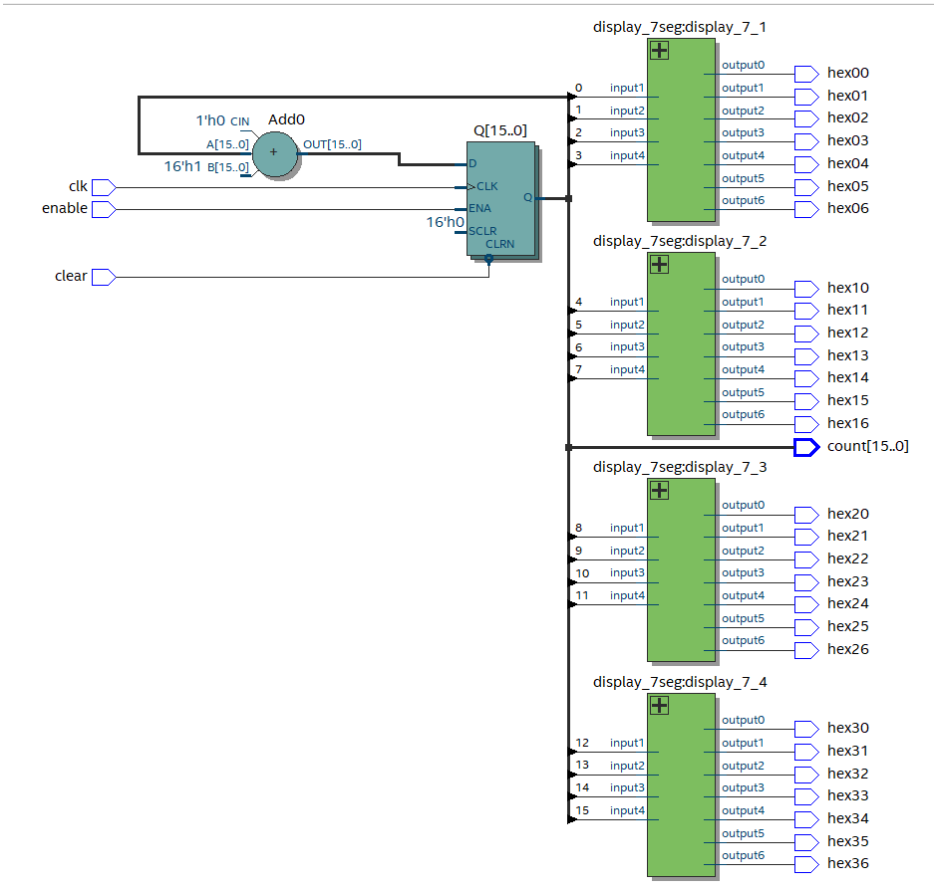
Código VHDL - 4 Displays de 7 segmentos, usando contador de 16 bits

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Counter16bit is
	Port (
	clk   : in  std_logic;
	clear : in  std_logic;
	enable : in  std_logic;
	count : out std_logic_vector(15 downto 0);
	  hex00 : out std_logic;
	  hex01 : out std_logic;
	  hex02 : out std_logic;
	  hex03 : out std_logic;
	  hex04 : out std_logic;
	  hex05 : out std_logic;
	  hex06 : out std_logic;
	  hex10 : out std_logic;
	  hex11 : out std_logic;
	  hex12 : out std_logic;
	  hex13 : out std_logic;
	  hex14 : out std_logic;
	  hex15 : out std_logic;
	  hex16 : out std_logic;
	  hex20 : out std_logic;
	  hex21 : out std_logic;
	  hex22 : out std_logic;
	  hex23 : out std_logic;
	  hex24 : out std_logic;
	  hex25 : out std_logic;
	  hex26 : out std_logic;
	  hex30 : out std_logic;
	  hex31 : out std_logic;
	  hex32 : out std_logic;
	  hex33 : out std_logic;
	  hex34 : out std_logic;
	  hex35 : out std_logic;
	  hex36 : out std_logic
	);
end Counter16bit;

architecture Behavioral of Counter16bit is
	signal Q   : std_logic_vector(15 downto 0);

    component display_7seg is
    Port
    (
	    input1 :  IN  STD_LOGIC;
	    input2 :  IN  STD_LOGIC;
	    input3 :  IN  STD_LOGIC;
```

```
        input4 :  IN  STD_LOGIC;
        output0 :  OUT  STD_LOGIC;
        output1 :  OUT  STD_LOGIC;
        output2 :  OUT  STD_LOGIC;
        output3 :  OUT  STD_LOGIC;
        output4 :  OUT  STD_LOGIC;
        output5 :  OUT  STD_LOGIC;
        output6 :  OUT  STD_LOGIC
    );
    end component;

begin
        process(clk)
        begin
                if clear = '0' then
                        Q <= "0000000000000000";
                elsif rising_edge(clk) then
                        if enable = '1' then
                                        Q <= std_logic_vector(unsigned(Q) + 1);
                        end if;
                end if;
        end process;


        display_7_1 : entity work.display_7seg
                        Port map (
                                input1 => Q(0),
                                input2 => Q(1),
                                input3 => Q(2),
                                input4 => Q(3),
                                output0 => hex00,
                                output1 => hex01,
                                output2 => hex02,
                                output3 => hex03,
                                output4 => hex04,
                                output5 => hex05,
                                output6 => hex06
                        );
        display_7_2 : entity work.display_7seg
                Port map (
                        input1 => Q(4),
                        input2 => Q(5),
                        input3 => Q(6),
                        input4 => Q(7),
                        output0 => hex10,
                        output1 => hex11,
                        output2 => hex12,
                        output3 => hex13,
                        output4 => hex14,
                        output5 => hex15,
                        output6 => hex16
                );
        display_7_3 : entity work.display_7seg
                        Port map (
                                input1 => Q(8),
                                input2 => Q(9),
                                input3 => Q(10),
                                input4 => Q(11),
```

```
                                output0 => hex20,
                                output1 => hex21,
                                output2 => hex22,
                                output3 => hex23,
                                output4 => hex24,
                                output5 => hex25,
                                output6 => hex26
                        );
        display_7_4 : entity work.display_7seg
                Port map (
                        input1 => Q(12),
                        input2 => Q(13),
                        input3 => Q(14),
                        input4 => Q(15),
                        output0 => hex30,
                        output1 => hex31,
                        output2 => hex32,
                        output3 => hex33,
                        output4 => hex34,
                        output5 => hex35,
                        output6 => hex36
                );

        count <= Q;

end Behavioral;
```

RTL Viewer - Contador de 16 bits



Se compararmos as estruturas desse componente com a do anteriormente desenvolvido, pode-se observar que cada bit possui seu próprio contador. Além disso, percebe-se que o

circuito com registradores é mais simples do que o que utiliza flip-flops T, possuindo menos estruturas e portas lógicas para realizar o mesmo tipo de problema.

Parte 3

Por estar na frequência de 50MHz, foi necessário comparar os valores do “clock” da placa com "10111110101111000010000000" (50 milhões em binário), para atualizar o contador que seria exibido no display

Código VDHL - Contador de 0 a 9, usando o clock da FPGA

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Counter10 is
        Port (
        clk   : in  std_logic;
        clear : in  std_logic;
        enable : in  std_logic;
        count : out std_logic_vector(3 downto 0);
          hex00 : out std_logic;
          hex01 : out std_logic;
          hex02 : out std_logic;
          hex03 : out std_logic;
          hex04 : out std_logic;
          hex05 : out std_logic;
          hex06 : out std_logic
        );
end Counter10;

architecture Behavioral of Counter10 is
        -- Representa o clock da FPGA
    signal fpgaCounter          : std_logic_vector(25 downto 0);
    -- Representa os números do display
    signal displayCounter  : std_logic_vector(3 downto 0);


    component display_7seg is
    Port
    (
            input1 :  IN  STD_LOGIC;
            input2 :  IN  STD_LOGIC;
            input3 :  IN  STD_LOGIC;
            input4 :  IN  STD_LOGIC;
            output0 :  OUT  STD_LOGIC;
            output1 :  OUT  STD_LOGIC;
            output2 :  OUT  STD_LOGIC;
            output3 :  OUT  STD_LOGIC;
            output4 :  OUT  STD_LOGIC;
            output5 :  OUT  STD_LOGIC;
            output6 :  OUT  STD_LOGIC
    );
    end component;

begin
```

```
process(clk)
        begin
        if (rising_edge(clk)) then
        if (Clear = '0') then
        fpgaCounter <= "00000000000000000000000000";
                                displayCounter <= "0000";
        elsif (enable = '1') then
        fpgaCounter <= std_logic_vector(unsigned(fpgaCounter) + 1);
                        -- 50MHz -> 50 milhões de 'ticks' da placa, ou seja, 1 segundo alterado no display
                        if (fpgaCounter = "10111110101111000010000000") then
                                -- printar no máximo até 9
                                if (displayCounter = "1001") then
                                        displayCounter <= "0000";
        else
                                        displayCounter <= displayCounter + 1;
                                end if;
                                fpgaCounter <= "00000000000000000000000000";
                        end if;
        else
        fpgaCounter <= fpgaCounter;
        end if;
        end if;
end process;



        display_7_1 : entity work.display_7seg
                        Port map (
                                input1 => displayCounter(0),
                                input2 => displayCounter(1),
                                input3 => displayCounter(2),
                                input4 => displayCounter(3),
                                output0 => hex00,
                                output1 => hex01,
                                output2 => hex02,
                                output3 => hex03,
                                output4 => hex04,
                                output5 => hex05,
                                output6 => hex06
                        );

        count <= displayCounter;

end Behavioral;
```

Parte 4

Nesse exercício foi utilizado o mecanismo de clock desenvolvido na parte 3, utilizando o clock de 50MHz da placa.

Código VHDL - Rotacionar palavra entre 3 displays

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity RotateOnFourDisplays is
```

```
    Port (
        clk   : in  std_logic;
        clear : in  std_logic;
        enable : in  std_logic;
        count : out std_logic_vector(1 downto 0);
                        hex00 : out std_logic;
                        hex01 : out std_logic;
                        hex02 : out std_logic;
                        hex03 : out std_logic;
                        hex04 : out std_logic;
                        hex05 : out std_logic;
                        hex06 : out std_logic;
                        hex10 : out std_logic;
                        hex11 : out std_logic;
                        hex12 : out std_logic;
                        hex13 : out std_logic;
                        hex14 : out std_logic;
                        hex15 : out std_logic;
                        hex16 : out std_logic;
                        hex20 : out std_logic;
                        hex21 : out std_logic;
                        hex22 : out std_logic;
                        hex23 : out std_logic;
                        hex24 : out std_logic;
                        hex25 : out std_logic;
                        hex26 : out std_logic
    );
end RotateOnFourDisplays;

architecture Behavioral of RotateOnFourDisplays is
    -- Representa o clock da FPGA
            signal fpgaCounter     : std_logic_vector(25 downto 0);
            -- Representa os números do display
            signal displayCounter  : std_logic_vector(1 downto 0);

            signal display1                         : std_logic_vector(3 downto 0);
            signal display2                         : std_logic_vector(3 downto 0);
            signal display3                         : std_logic_vector(3 downto 0);


            component display_7seg is
            Port
            (
                    input1 :  IN  STD_LOGIC;
                    input2 :  IN  STD_LOGIC;
                    input3 :  IN  STD_LOGIC;
                    input4 :  IN  STD_LOGIC;
                    output0 :  OUT  STD_LOGIC;
                    output1 :  OUT  STD_LOGIC;
                    output2 :  OUT  STD_LOGIC;
                    output3 :  OUT  STD_LOGIC;
                    output4 :  OUT  STD_LOGIC;
                    output5 :  OUT  STD_LOGIC;
                    output6 :  OUT  STD_LOGIC
            );
            end component;

begin
```

```
        process(clk)
                begin
    if (rising_edge(clk)) then
      if (Clear = '0') then
        fpgaCounter <= "00000000000000000000000000";
                                        displayCounter <= "00";
      elsif (enable = '1') then
        fpgaCounter <= std_logic_vector(unsigned(fpgaCounter) + 1);
                                -- 50MHz -> 50 milhões de 'ticks' da placa, ou seja, 1 segundo alterado no display
                                if (fpgaCounter = "10111110101111000010000000") then
                                        if(displayCounter = "11") then
                                                displayCounter <= "00";
                                        end if;
                                        case displayCounter is
                                                when "00" => display1 <= "0000"; display2 <= "1110"; display3
<= "1101";
                                                when "01" => display1 <= "1101"; display2 <= "0000"; display3
<= "1110";
                                                when "10" => display1 <= "1110"; display2 <= "1101"; display3
<= "0000";
                                                when others => display1 <= "0000"; display2 <= "1110";
display3 <= "1101";
                                        end case;
                                        displayCounter <= displayCounter + 1;
                                        fpgaCounter <= "00000000000000000000000000";
                                end if;
      else
        fpgaCounter <= fpgaCounter;
    end if;
                end if;
        end process;

                display_7_1 : entity work.display_7seg
                                Port map (
                                        input1 => display1(0),
                                        input2 => display1(1),
                                        input3 => display1(2),
                                        input4 => display1(3),
                                        output0 => hex00,
                                        output1 => hex01,
                                        output2 => hex02,
                                        output3 => hex03,
                                        output4 => hex04,
                                        output5 => hex05,
                                        output6 => hex06
                                );

                display_7_2 : entity work.display_7seg
                                Port map (
                                        input1 => display2(0),
                                        input2 => display2(1),
                                        input3 => display2(2),
                                        input4 => display2(3),
                                        output0 => hex10,
                                        output1 => hex11,
                                        output2 => hex12,
                                        output3 => hex13,
                                        output4 => hex14,
```

```
                                        output5 => hex15,
                                        output6 => hex16
                                );

                display_7_3 : entity work.display_7seg
                                Port map (
                                        input1 => display3(0),
                                        input2 => display3(1),
                                        input3 => display3(2),
                                        input4 => display3(3),
                                        output0 => hex20,
                                        output1 => hex21,
                                        output2 => hex22,
                                        output3 => hex23,
                                        output4 => hex24,
                                        output5 => hex25,
                                        output6 => hex26
                                );

   count <= displayCounter;

end Behavioral;
```

Parte 5

Nesse exercício foi utilizado o mecanismo de clock desenvolvido na parte 3, utilizando o clock de 50MHz da placa.

Código VHDL - Rotacionar palavra entre 6 displays (sem mostrar nada nos outros)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity RotateOnSixDisplays is
   Port (
      clk   : in  std_logic;
      clear : in  std_logic;
      enable : in  std_logic;
      count : out std_logic_vector(2 downto 0);
                        hex00 : out std_logic;
                        hex01 : out std_logic;
                        hex02 : out std_logic;
                        hex03 : out std_logic;
                        hex04 : out std_logic;
                        hex05 : out std_logic;
                        hex06 : out std_logic;
                        hex10 : out std_logic;
                        hex11 : out std_logic;
                        hex12 : out std_logic;
                        hex13 : out std_logic;
                        hex14 : out std_logic;
                        hex15 : out std_logic;
                        hex16 : out std_logic;
                        hex20 : out std_logic;
                        hex21 : out std_logic;
                        hex22 : out std_logic;
```

```
                hex23 : out std_logic;
                hex24 : out std_logic;
                hex25 : out std_logic;
                hex26 : out std_logic;
                hex30 : out std_logic;
                hex31 : out std_logic;
                hex32 : out std_logic;
                hex33 : out std_logic;
                hex34 : out std_logic;
                hex35 : out std_logic;
                hex36 : out std_logic;
                hex40 : out std_logic;
                hex41 : out std_logic;
                hex42 : out std_logic;
                hex43 : out std_logic;
                hex44 : out std_logic;
                hex45 : out std_logic;
                hex46 : out std_logic;
                hex50 : out std_logic;
                hex51 : out std_logic;
                hex52 : out std_logic;
                hex53 : out std_logic;
                hex54 : out std_logic;
                hex55 : out std_logic;
                hex56 : out std_logic
    );
end RotateOnSixDisplays;

architecture Behavioral of RotateOnSixDisplays is
    -- Representa o clock da FPGA
        signal fpgaCounter     : std_logic_vector(25 downto 0);
        -- Representa os números do display
        signal displayCounter  : std_logic_vector(2 downto 0);

        signal display1                 : std_logic_vector(3 downto 0);
        signal display2                 : std_logic_vector(3 downto 0);
        signal display3                 : std_logic_vector(3 downto 0);
        signal display4                 : std_logic_vector(3 downto 0);
        signal display5                 : std_logic_vector(3 downto 0);
        signal display6                 : std_logic_vector(3 downto 0);


        component display_7seg is
        Port
        (
            input1 :  IN  STD_LOGIC;
            input2 :  IN  STD_LOGIC;
            input3 :  IN  STD_LOGIC;
            input4 :  IN  STD_LOGIC;
            output0 :  OUT  STD_LOGIC;
            output1 :  OUT  STD_LOGIC;
            output2 :  OUT  STD_LOGIC;
            output3 :  OUT  STD_LOGIC;
            output4 :  OUT  STD_LOGIC;
            output5 :  OUT  STD_LOGIC;
            output6 :  OUT  STD_LOGIC
        );
        end component;
```

```
begin
        process(clk)
                begin
    if (rising_edge(clk)) then
      if (Clear = '0') then
        fpgaCounter <= "00000000000000000000000000";
                                        displayCounter <= "000";
      elsif (enable = '1') then
        fpgaCounter <= std_logic_vector(unsigned(fpgaCounter) + 1);
                                -- 50MHz -> 50 milhões de 'ticks' da placa, ou seja, 1 segundo alterado no display
                                if (fpgaCounter = "10111110101111000010000000") then
                                        if(displayCounter = "111") then
                                                displayCounter <= "000";
                                        end if;
                                        case displayCounter is
                                                when "000"  => display1 <= "0000"; display2 <= "1110";
display3 <= "1101"; display4 <= "1111"; display5 <= "1111"; display6 <= "1111";
                                                when "001"  => display1 <= "1111"; display2 <= "0000";
display3 <= "1110"; display4 <= "1101"; display5 <= "1111"; display6 <= "1111";
                                                when "010"  => display1 <= "1111"; display2 <= "1111";
display3 <= "0000"; display4 <= "1110"; display5 <= "1101"; display6 <= "1111";
                                                when "011"  => display1 <= "1111"; display2 <= "1111";
display3 <= "1111"; display4 <= "0000"; display5 <= "1110"; display6 <= "1101";
                                                when "100"  => display1 <= "1101"; display2 <= "1111";
display3 <= "1111"; display4 <= "1111"; display5 <= "0000"; display6 <= "1110";
                                                when "101"  => display1 <= "1110"; display2 <= "1101";
display3 <= "1111"; display4 <= "1111"; display5 <= "1111"; display6 <= "0000";
                                                when others => display1 <= "0000"; display2 <= "1110";
display3 <= "1101"; display4 <= "1111"; display5 <= "1111"; display6 <= "1111";
                                        end case;

                                        displayCounter <= displayCounter + 1;
                                        fpgaCounter <= "00000000000000000000000000";
                                end if;
      else
        fpgaCounter <= fpgaCounter;
    end if;
                end if;
        end process;

                display_7_1 : entity work.display_7seg
                                Port map (
                                        input1 => display1(0),
                                        input2 => display1(1),
                                        input3 => display1(2),
                                        input4 => display1(3),
                                        output0 => hex00,
                                        output1 => hex01,
                                        output2 => hex02,
                                        output3 => hex03,
                                        output4 => hex04,
                                        output5 => hex05,
                                        output6 => hex06
                                );

                display_7_2 : entity work.display_7seg
                                Port map (
```

```
                input1 => display2(0),
                input2 => display2(1),
                input3 => display2(2),
                input4 => display2(3),
                output0 => hex10,
                output1 => hex11,
                output2 => hex12,
                output3 => hex13,
                output4 => hex14,
                output5 => hex15,
                output6 => hex16
        );

display_7_3 : entity work.display_7seg
        Port map (
                input1 => display3(0),
                input2 => display3(1),
                input3 => display3(2),
                input4 => display3(3),
                output0 => hex20,
                output1 => hex21,
                output2 => hex22,
                output3 => hex23,
                output4 => hex24,
                output5 => hex25,
                output6 => hex26
        );

display_7_4 : entity work.display_7seg
        Port map (
                input1 => display4(0),
                input2 => display4(1),
                input3 => display4(2),
                input4 => display4(3),
                output0 => hex30,
                output1 => hex31,
                output2 => hex32,
                output3 => hex33,
                output4 => hex34,
                output5 => hex35,
                output6 => hex36
        );

display_7_5 : entity work.display_7seg
        Port map (
                input1 => display5(0),
                input2 => display5(1),
                input3 => display5(2),
                input4 => display5(3),
                output0 => hex40,
                output1 => hex41,
                output2 => hex42,
                output3 => hex43,
                output4 => hex44,
                output5 => hex45,
                output6 => hex46
        );
```

```
		display_7_6 : entity work.display_7seg
			Port map (
				input1 => display6(0),
				input2 => display6(1),
				input3 => display6(2),
				input4 => display6(3),
				output0 => hex50,
				output1 => hex51,
				output2 => hex52,
				output3 => hex53,
				output4 => hex54,
				output5 => hex55,
				output6 => hex56
			);

   count <= displayCounter;

end Behavioral;
```

Para esse exercício, foi necessário realizar uma adaptação no display usado anteriormente: adicionando uma verificação para apagar o display quando fosse enviado “1111”, pois no caso não era necessário escrever no display a letra ‘F’.
Foi adicionado esse processo no VHDL do display:

```
PROCESS(input1, input2, input3, input4)
BEGIN
	fio_final0 <= fio_numero1 OR fio_numeroB OR fio_numeroD OR fio_numero4;
	fio_final1 <= fio_numero5 OR fio_numeroB OR fio_numero6 OR fio_numeroC OR fio_numeroE OR
fio_numeroF;
	fio_final2 <= fio_numero2 OR fio_numeroE OR fio_numeroF OR fio_numeroC;
	SYNTHESIZED_WIRE_0 <= fio_numero4 OR fio_numero1;
	fio_final3 <= SYNTHESIZED_WIRE_0 OR fio_numeroA OR fio_numeroF OR fio_numero7;
	fio_final4 <= fio_numero1 OR fio_numero4 OR fio_numero3 OR fio_numero5 OR fio_numero7 OR
fio_numero9;
	SYNTHESIZED_WIRE_1 <= fio_numero2 OR fio_numero1;
	fio_final5 <= SYNTHESIZED_WIRE_1 OR fio_numero7 OR fio_numeroD OR fio_numero3;
	fio_final6 <= fio_numero0 OR fio_numero7 OR fio_numeroC OR fio_numero1;

	if(fio_numeroF = '1') then
		fio_final0 <= '1';
		fio_final1 <= '1';
		fio_final2 <= '1';
		fio_final3 <= '1';
		fio_final4 <= '1';
		fio_final5 <= '1';
		fio_final6 <= '1';
	end if;
end process;
```