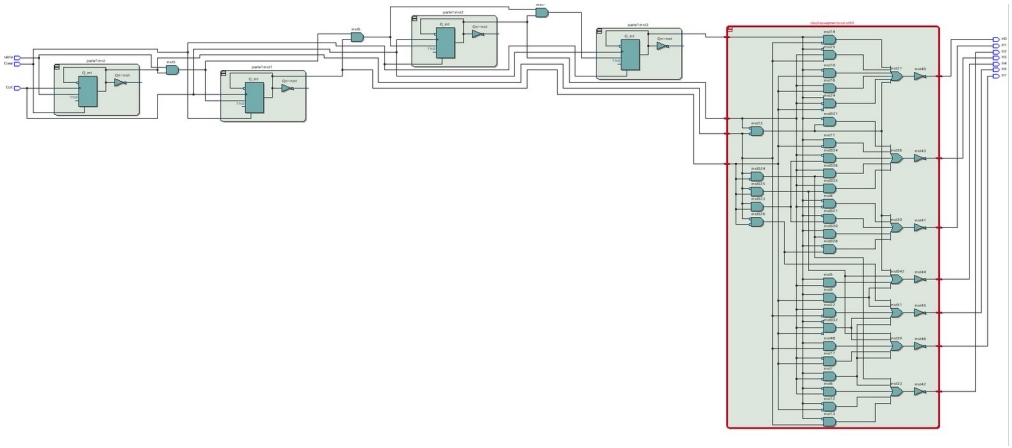
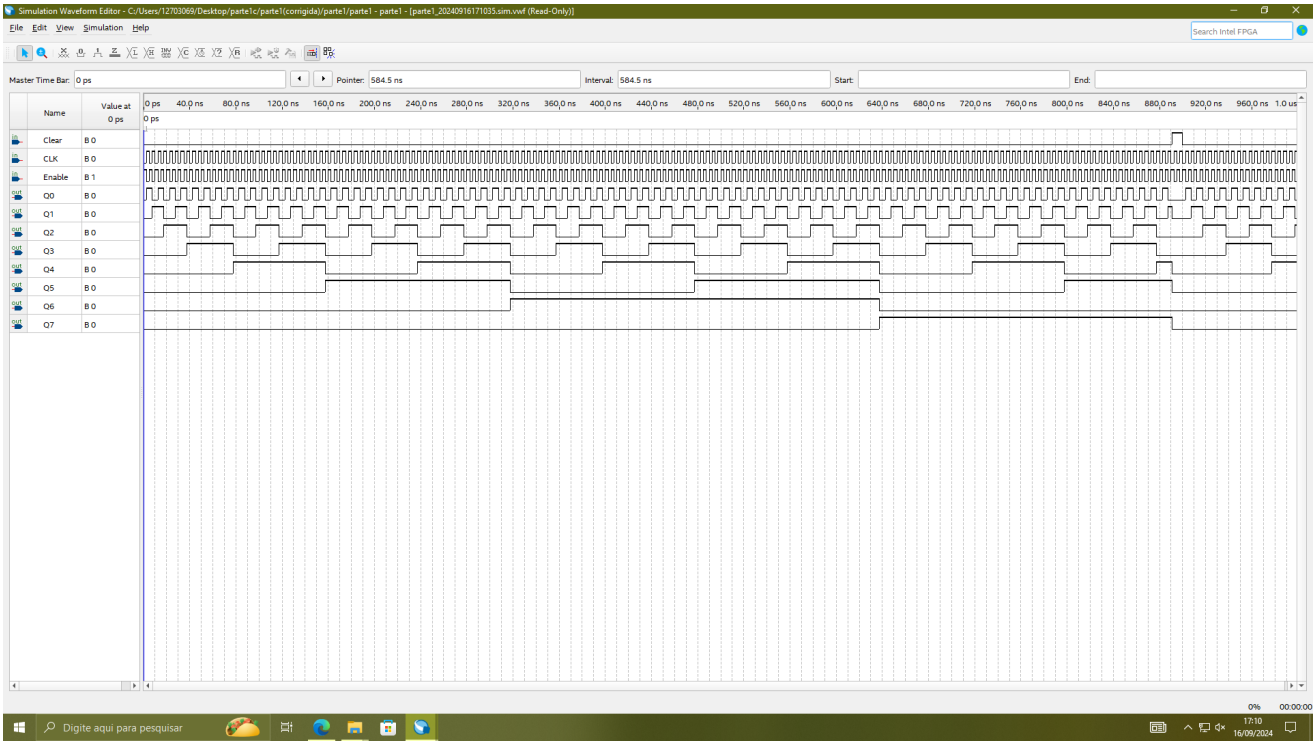
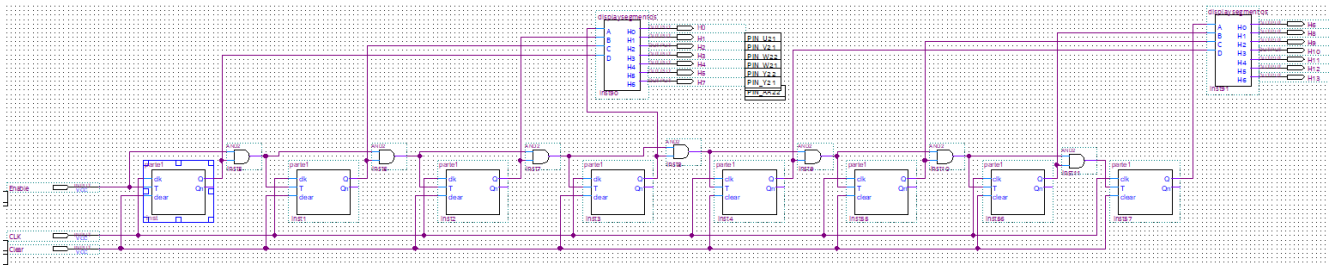
Aulas 4 e 5 – Contadores:

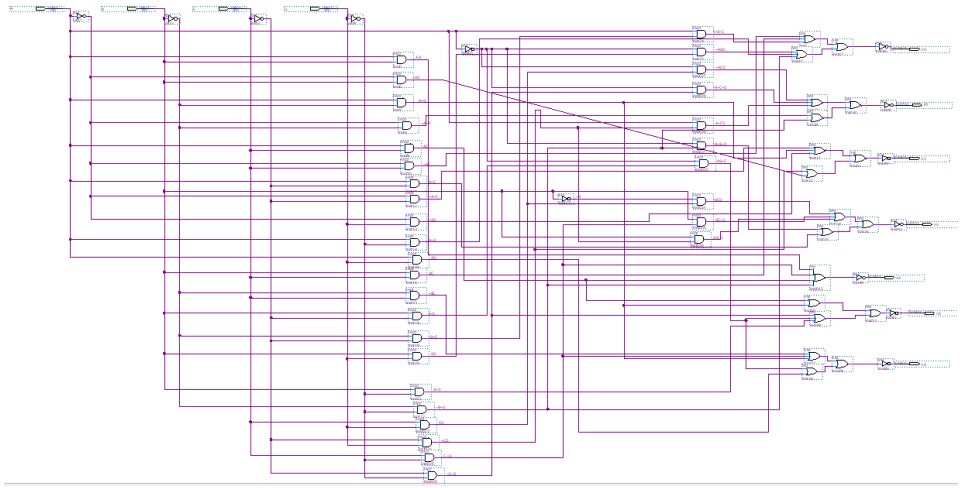
Parte 1: implementamos um contador síncrono de 8 bits utilizando flip-flops T pelo diagrama de blocos do Quartus, simulamos e verificamos o circuito e mapeamos as entradas para chaves, botões e displays de 7 segmentos de uma FPGA. Os blocos de flip-flops T foram gerados no Quartus e implementados com VHDL, enquanto os blocos “displaysegmentos” foram gerados em um circuito criado por diagrama de blocos.







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

entity parte1 is
    Port (
        clk   : in STD_LOGIC;
        T     : in STD_LOGIC;
        clear : in STD_LOGIC;
        Q     : out STD_LOGIC;
        Qn    : out STD_LOGIC
    );
end parte1;

architecture Behavioral of parte1 is
    signal Q_int : STD_LOGIC := '0';
begin
    process (clk, clear)
    begin
        if clear = '1' 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;
    Qn <= not Q_int;
end Behavioral;
```



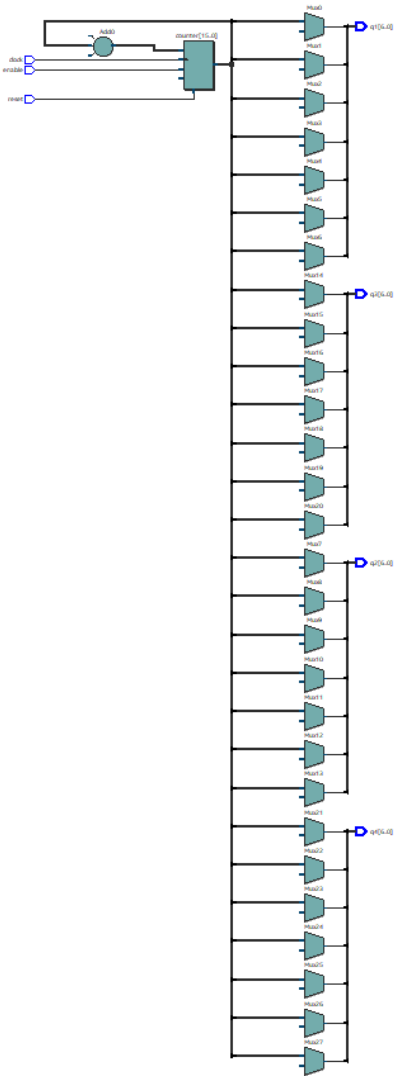
Bloco “displaysegmentos”

Parte 2: implementamos um contador de 16 bits utilizando a operação de incremento do VHDL, e simulamos no Quartus com a exibição das saídas do contador em um display de 7 segmentos.

Código para a parte 2.

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

entity Parte2a is
    port    (
        enable  : in    std_logic;
        clock   : in    std_logic;
        reset   : in    std_logic;
        q1      : out   std_logic_vector(6 downto 0);
        q2      : out   std_logic_vector(6 downto 0);
        q3      : out   std_logic_vector(6 downto 0);
        q4      : out   std_logic_vector(6 downto 0)
    );
end Parte2a;

architecture behavioral of Parte2a is
    signal counter    : unsigned(15 downto 0);

    -- Funções de conversão para display de 7 segmentos
    function int_to_7seg(d : integer) return std_logic_vector is
    begin
        case d is
            when 0   => return "1000000"; -- 0
            when 1   => return "1111001"; -- 1
            when 2   => return "0100100"; -- 2
            when 3   => return "0110000"; -- 3
            when 4   => return "0011001"; -- 4
            when 5   => return "0010010"; -- 5
            when 6   => return "0000010"; -- 6
            when 7   => return "1111000"; -- 7
            when 8   => return "0000000"; -- 8
            when 9   => return "0010000"; -- 9
            when 10  => return "0001000"; -- A
            when 11  => return "0000011"; -- B
            when 12  => return "1000110"; -- C
            when 13  => return "0100001"; -- D
            when 14  => return "0000110"; -- E
            when 15  => return "0001110"; -- F
            when others => return "0000000"; -- Desconhecido
        end case;
    end function;

begin

    process(clock, reset, enable)
    begin
        if reset = '1' then
            counter <= (others => '0');
        elsif rising_edge(clock) then
            if enable = '1' then
                counter <= counter + 1;
            end if;
        end if;
    end process;

    q1 <= int_to_7seg(to_integer(counter(3 downto 0))); -- Menos significativos
    q2 <= int_to_7seg(to_integer(counter(7 downto 4)));
    q3 <= int_to_7seg(to_integer(counter(11 downto 8)));
    q4 <= int_to_7seg(to_integer(counter(15 downto 12))); -- Mais significativos

end behavioral;
```

Visualização do código no RTL viewer



Embora tanto a parte 1 como a parte 2 implementem contadores, a parte 2 conta com uma lógica que realiza adição de suas entradas.

A lógica do código é bem simples. O vetor counter armazena um número na base binária e 4 vetores binários, q1, q2, q3 e q4, armazenam, cada um, a sequência de bits necessária para representar um dígito, na base hexadecimal, de counter pelos displays de 7 segmentos. Essa sequência é determinada pela função "int_to_7seg", que recebe como parâmetro um inteiro, sequência de 4 bits de counter convertida para a base 10, e retorna a sequência. Finalmente, cada um dos 4 vetores binários é mapeado por meio do pin assignment para um display de 7 segmentos, para representar um dígito.

Parte 3: desenvolvemos um circuito que exibe dígitos de 0 a 9 no display de 7 segmentos, alternando a cada 1 segundo. Um contador é responsável por controlar os intervalos de tempos.

Código para a parte 3

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

entity parte3 is
    Port (
        clk     : in  STD_LOGIC;   -- Clock de 50 MHz
        reset   : in  STD_LOGIC;   -- Reset para inicializar o contador
        seg     : out STD_LOGIC_VECTOR (6 downto 0)  -- Saída para o display de 7 segmentos
    );
end parte3;

architecture Behavioral of parte3 is
    -- Constantes
    constant COUNT_MAX : integer := 50000000; -- Contador para 1 segundo (50 MHz * 1 s)

    -- Sinais
    signal counter_1s : integer := 0;
    signal digit_counter : integer := 0;
    signal clk_1s : STD_LOGIC := '0';  -- Sinal de clock para 1 segundo

    -- Funções de conversão para display de 7 segmentos
    function int_to_7seg (d : integer) return STD_LOGIC_VECTOR is
    begin
        case d is
             when 0 => return "1000000"; -- A para 0
              when 1 => return "1111001"; -- B para 1
              when 2 => return "0100100"; -- C para 2
              when 3 => return "0110000"; -- D para 3
              when 4 => return "0011001"; -- E para 4
              when 5 => return "0010010"; -- F para 5
              when 6 => return "0000010"; -- G para 6
              when 7 => return "1111000"; -- H para 7
              when 8 => return "0000000"; -- I para 8
              when 9 => return "0011000"; -- J para 9
            when others => return "1111111"; -- Desconhecido
        end case;
    end function;

begin
    process(clk, reset)
    begin
        if reset = '1' then
            counter_1s <= 0;
            clk_1s <= '0';
            digit_counter <= 0;
            seg <= int_to_7seg(digit_counter);
        elsif rising_edge(clk) then
            -- Contador de 1 segundo
            if counter_1s = COUNT_MAX - 1 then
                counter_1s <= 0;
                clk_1s <= '1';
            else
                counter_1s <= counter_1s + 1;
            end if;

            -- Atualiza o contador de dígitos a cada pulso de 1 segundo
            if clk_1s = '1' then
                clk_1s <= '0';
                if digit_counter = 9 then
                    digit_counter <= 0;
                else
                    digit_counter <= digit_counter + 1;
                end if;
                seg <= int_to_7seg(digit_counter);
            end if;
        end if;
    end process;

    -- Saída para o display de 7 segmentos
    -- seg <= int_to_7seg(digit_counter);
end Behavioral;
```

Este código reutiliza a função “int_to_7seg” e seu contador é semelhante ao programa anterior. Definimos a constante COUNT_MAX, que corresponde à frequência do clock, ou seja, quando counter alcançar o antecessor desse número (pois counter começa do número zero), 1 segundo terá se passado, o que implicará no incremento do sinal “clk_1s”. O sinal “digit_counter” é um contador cíclico que é incrementado a cada ciclo de “clk_1s”, iterando entre os números 0 e 9. Finalmente, o vetor seg armazena a sequência de bits, gerada pela função “int_to_7seg”, que representa o dígito atual de “digit_counter” e é mapeado para um display de 7 segmentos da FPGA.

Parte 4: implementamos um código VHDL que rotaciona a palavra "dE0 " nos displays HEX3-0 no sentido da direita para a esquerda com intervalo de 1 segundo entre cada deslocamento da palavra.

Código para a parte 4

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

entity parte4 is
    Port (
        clk     : in  STD_LOGIC;   -- Clock de 50 MHz
        reset   : in  STD_LOGIC;   -- Reset para inicializar o contador
        seg0    : out STD_LOGIC_VECTOR (6 downto 0);  -- Saída para o d
        seg1    : out STD_LOGIC_VECTOR (6 downto 0);
        seg2    : out STD_LOGIC_VECTOR (6 downto 0);
        seg3    : out STD_LOGIC_VECTOR (6 downto 0)
    );
end parte4;

architecture Behavioral of parte4 is
    -- Constantes
    constant COUNT_MAX : integer := 50000000; -- Contador para 1 segundo

    -- Sinais
    signal counter_1s : integer := 0;
    signal digit_counter : integer := 0;
    signal clk_1s : STD_LOGIC := '0';  -- Sinal de clock para 1 segundo

    -- Funções de conversão para display de 7 segmentos
    function int_to_7seg (d : integer) return STD_LOGIC_VECTOR is
    begin
        case d is
              when 0 => return "0100001"; -- letra d
              when 1 => return "0000110"; -- letra E
              when 2 => return "1000000"; -- digito 0
            when others => return "1111111"; -- Desligado
        end case;
    end function;

begin
    process(clk, reset)
    begin
        if reset = '1' then
            counter_1s <= 0;
            clk_1s <= '0';
            digit_counter <= 0;
            seg0 <= int_to_7seg(digit_counter mod 4);
            seg1 <= int_to_7seg((digit_counter + 1) mod 4);
            seg2 <= int_to_7seg((digit_counter + 2) mod 4);
            seg3 <= int_to_7seg((digit_counter + 3) mod 4);

        elsif rising_edge(clk) then
            -- Contador de 1 segundo
            if counter_1s = COUNT_MAX - 1 then
                counter_1s <= 0;
                clk_1s <= '1';
            else
                counter_1s <= counter_1s + 1;
            end if;

            if clk_1s = '1' then
                clk_1s <= '0';
                digit_counter <= digit_counter + 1;
                seg0 <= int_to_7seg(digit_counter mod 4);
                seg1 <= int_to_7seg((digit_counter + 1) mod 4);
                seg2 <= int_to_7seg((digit_counter + 2) mod 4);
                seg3 <= int_to_7seg((digit_counter + 3) mod 4);
            end if;

        end if;
    end process;

    -- Saída para o display de 7 segmentos
    -- seg <= int_to_7seg(digit_counter);
end Behavioral;
```

Nesse programa, a função “int_to_7seg” foi modificada, de modo que aceita como parâmetros apenas os números 0, 1 e 2, cada 1 retornando a sequência para representação em um display de 7 segmentos dos caracteres “d”, “e”, e “0” e “ “ respectivamente, mas preserva a lógica do contador de 1 segundo. A cada segundo, o sinal “digit_counter” é incrementado e 4 vetores binários que armazenam a sequência necessária para acender os LEDs de um display, seg0, seg1, seg2 e seg3, recebem o valor de retorno da função “int_to_7seg”, cujo parâmetro é um número de 0 a 3, graças a operação de módulo aplicada em “digit_counter”. Cada vetor recebe um valor diferente da função, já que cada parâmetro é incrementado por um número diferente, fazendo com que cada um se diferencie do outro. Essa variação permite que, a cada segundo, um dos 4 displays de 7 segmentos receba um caractere diferente de forma cíclica, sendo que o vetor cuja função receber como argumento o número 3

receberá uma sequência de bits que torna seu display apagado. Caso a chave corresponde ao reset seja acionada, os caracteres param de oscilar, transmitindo, apenas, a sequência “d” “e” “0” e “ ”.

Parte 5: expandimos o circuito da parte 4 para que a palavra percorra todos os displays de 7 segmentos da FPGA.

Código parte 5

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

entity parte5 is
    Port (
        clk     : in  STD_LOGIC;   -- Clock de 50 MHz
        reset   : in  STD_LOGIC;   -- Reset para inicializar o contador
        seg0    : out STD_LOGIC_VECTOR (6 downto 0);  -- Saida para o display de 7 segmentos primeiro da esq pra dir
        seg1    : out STD_LOGIC_VECTOR (6 downto 0);
        seg2    : out STD_LOGIC_VECTOR (6 downto 0);
        seg3    : out STD_LOGIC_VECTOR (6 downto 0);
        seg4    : out STD_LOGIC_VECTOR (6 downto 0);
        seg5    : out STD_LOGIC_VECTOR (6 downto 0)
    );
end parte5;

architecture Behavioral of parte5 is
    -- Constantes
    constant COUNT_MAX : integer := 50000000; -- Contador para 1 segundo (50 MHz * 1 s)

    -- Sinais
    signal counter_1s : integer := 0;
    signal digit_counter : integer := 0;
    signal clk_1s : STD_LOGIC := '0';  -- Sinal de clock para 1 segundo

    -- Funções de conversão para display de 7 segmentos
    function int_to_7seg (d : integer) return STD_LOGIC_VECTOR is
    begin
        case d is
              when 0 => return "0100001"; -- letra d
              when 1 => return "0000110"; -- letra E
              when 2 => return "1000000"; -- digito 0
            when others => return "1111111"; -- Desligado
        end case;
    end function;

begin
    process(clk, reset)
    begin
        if reset = '1' then
            counter_1s <= 0;
            clk_1s <= '0';
            digit_counter <= 0;
            seg0 <= int_to_7seg(digit_counter mod 6);
            seg1 <= int_to_7seg((digit_counter + 1) mod 6);
            seg2 <= int_to_7seg((digit_counter + 2) mod 6);
            seg3 <= int_to_7seg((digit_counter + 3) mod 6);
            seg4 <= int_to_7seg((digit_counter + 4) mod 6);
            seg5 <= int_to_7seg((digit_counter + 5) mod 6);

        elsif rising_edge(clk) then
            -- Contador de 1 segundo
            if counter_1s = COUNT_MAX - 1 then
                counter_1s <= 0;
                clk_1s <= '1';
            else
                counter_1s <= counter_1s + 1;
            end if;

            if clk_1s = '1' then
                clk_1s <= '0';
                digit_counter <= digit_counter + 1;
                seg0 <= int_to_7seg(digit_counter mod 6);
                seg1 <= int_to_7seg((digit_counter + 1) mod 6);
                seg2 <= int_to_7seg((digit_counter + 2) mod 6);
                seg3 <= int_to_7seg((digit_counter + 3) mod 6);
                seg4 <= int_to_7seg((digit_counter + 4) mod 6);
                seg5 <= int_to_7seg((digit_counter + 5) mod 6);
            end if;

        end if;
    end process;

    -- Saida para o display de 7 segmentos
    -- seg <= int_to_7seg(digit_counter);
end Behavioral;
```

Nesse programa, os caracteres “d”, “e”, e “0” e “ “ precisam se deslocar para a direita a cada segundo passando por todos os 6 displays de 7 segmentos, o que exige a criação de 6 vetores pra armazenar as sequências de bits retornadas por “int_to_7seg”. Novamente, por causa do caráter cíclico da aritmética modular, o parâmetro “digit_counter”, responsável pela iteração entre os caracteres, é incrementado por um número diferente e é convertido para um inteiro no módulo 6. Assim, a cada segundo, um vetor assume a sequência de um caractere diferente, e aqueles cujas funções receberem como argumentos números diferentes de 0 a 2 receberão uma sequência que torna seu display apagado.