Atividade – Aula 18/03

ALUNO(A):GUILHERME LUCAS PEREIRA BERNARDO_____

ATENÇÃO: Vale ressaltar que esta atividade será utilizada como critério para a contabilização de sua frequência de aula.

PRAZO DE ENTREGA: 22/03/2021

[Questão – 01] Implemente os componentes abaixo usando a linguagem VHDL, para cada componente apresenta os testes do componente, apresentando o resultado dos pinos de entrada e saída.

[COMPONENTE 01]. Multiplexador de quatro opções de entrada.

CÓDIGO VHDL DO MULTIPLEXADOR4TO1:

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

entity mult is
    port(
        in_port : in STD_LOGIC_VECTOR(1 downto 0);
        in_portA : in STD_LOGIC_VECTOR(1 downto 0);
        in_portB : in STD_LOGIC_VECTOR(1 downto 0);
        in_portC : in STD_LOGIC_VECTOR(1 downto 0);
        in_portD : in STD_LOGIC_VECTOR(1 downto 0);
        out_port : out STD_LOGIC_VECTOR(1 downto 0);
    );
end mult;

architecture behavior of mult is
begin
    process (in_port, in_portA, in_portB, in_portC, in_portD)
    begin
        case in_port is
            when "00" => out_port <= in_portA;
            when "01" => out_port <= in_portB;
            when "10" => out_port <= in_portC;
            when "11" => out_port <= in_portD;
        end case;

    end process;
end behavior;
```

REPRESENTAÇÃO RTL

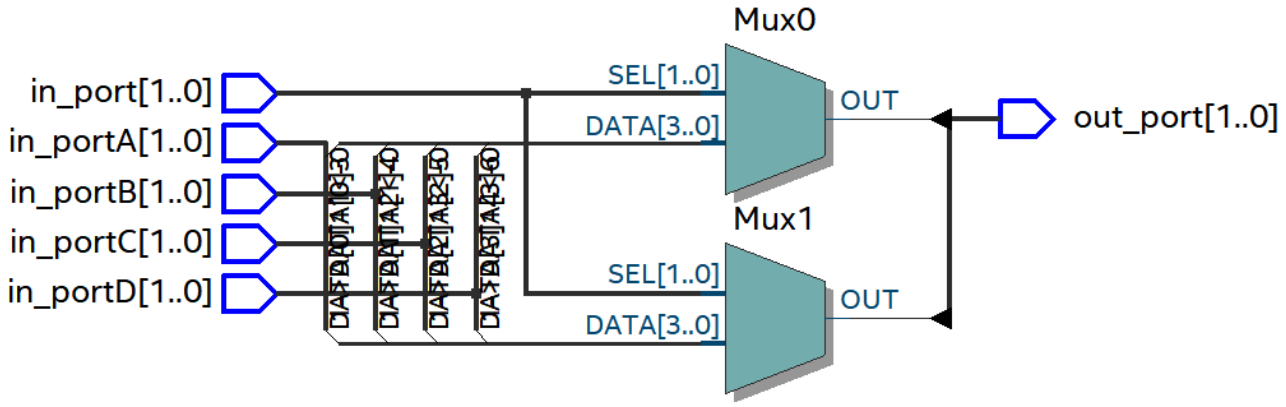
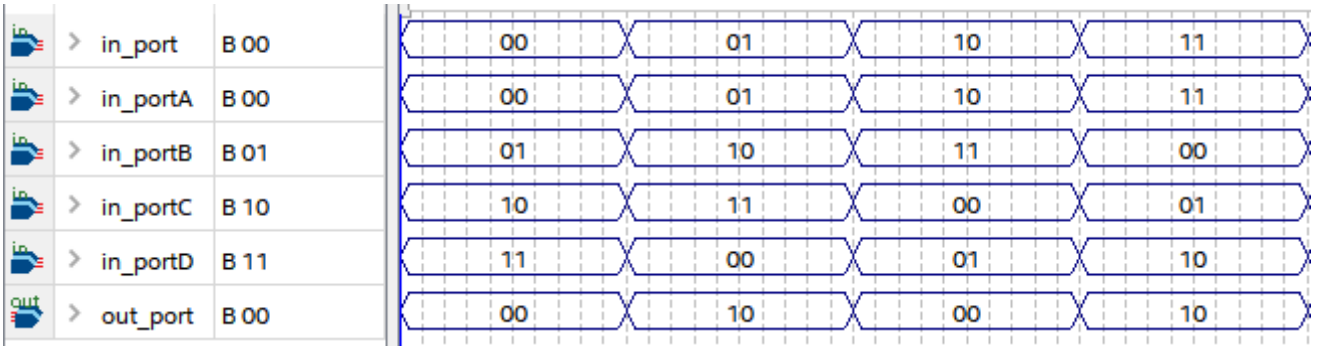


TABELA DE EXECUÇÃO DO MULTIPLEXADOR



LEGENDA:
quando in_port apresenta 00… out_port tem sinal igual a in_portA.
quando in_port apresenta 01… out_port tem sinal igual a in_portB.
quando in_port apresenta 10… out_port tem sinal igual a in_portC.
quando in_port apresenta 11… out_port tem sinal igual a in_portD.

[COMPONENTE 02]. Porta lógica XOR.

CÓDIGO VHDL DO P_XOR:

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

entity p_xor is
    port(
        in_portA : in STD_LOGIC;
        in_portB : in STD_LOGIC;
        out_port : out STD_LOGIC
    );
end p_xor;

architecture behavior of p_xor is
begin
    out_port <= in_portA xor in_portB;
end behavior;
library ieee;
use ieee.std_logic_1164.all;

entity p_xor is
    port(
        in_portA : in STD_LOGIC;
        in_portB : in STD_LOGIC;
        out_port : out STD_LOGIC
    );
end p_xor;

architecture behavior of p_xor is
begin
    out_port <= in_portA xor in_portB;
end behavior;
```

REPRESENTAÇÃO RTL

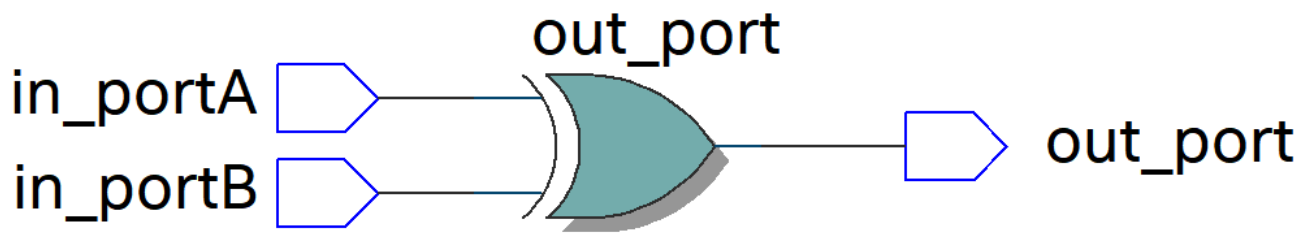
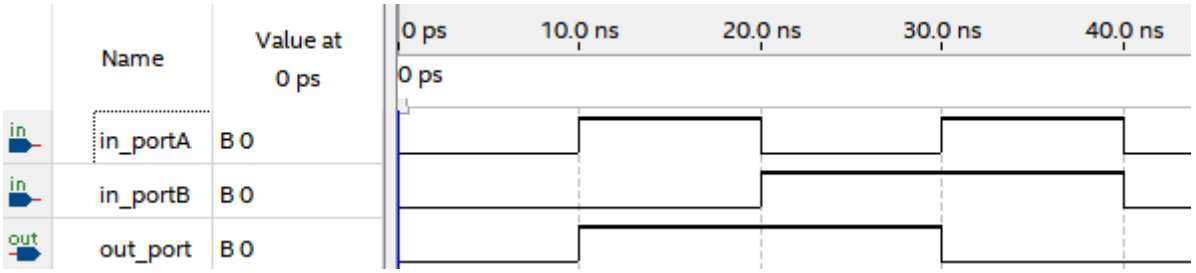


TABELA DE EXECUÇÃO DO P_XOR



LEGENDA:

out_port é igual a 1 se e somente se in_portA e in_portB forem igual a 1, fora isso a porta out_port é sempre 0.

[COMPONENTE 03]. Somador de 16 bits.

CÓDIGO VHDL DO SOMADOR16:

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

entity somador16 is
    port(
        in_portA : in STD_LOGIC_VECTOR(15 downto 0);
        in_portB : in STD_LOGIC_VECTOR(15 downto 0);
        out_port : out STD_LOGIC_VECTOR(15 downto 0)
    );
end somador16;

architecture behavior of somador16 is
begin
    out_port <= in_portA + in_portB;
end behavior;
```

REPRESENTAÇÃO RTL

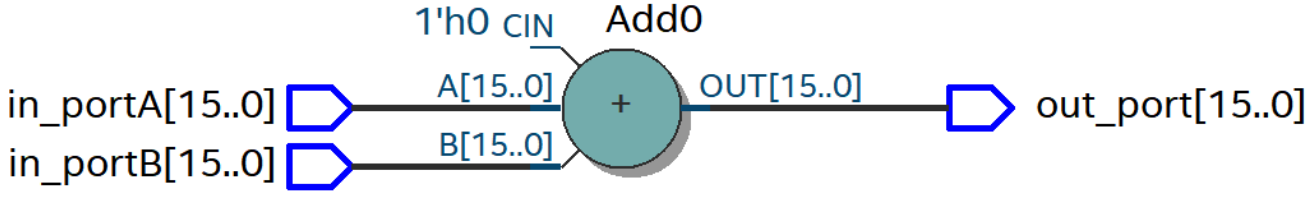
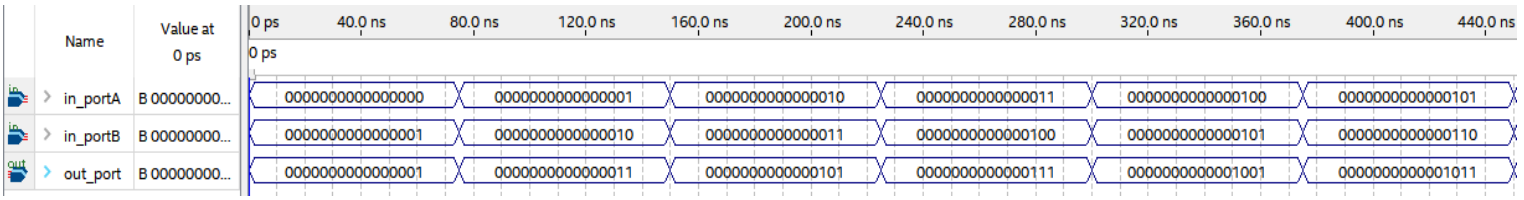


TABELA DE EXECUÇÃO DO SOMADOR16:



LEGENDA:

out_port é o resultado da soma dos dois in_ports, no primeiro ciclo soma-se 0000000000000000 + 0000000000000000, no próximo ciclo realiza a soma 0000000000000001 + 0000000000000010 e por aí vai

[COMPONENTE 04]. Extensor de sinal de 8 bits para 16 bits.

CÓDIGO VHDL DO EXTENDER8TO16:

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

entity extender8to16 is
    port (
        in_port : in std_logic_vector(7 downto 0);
        out_port : out std_logic_vector(15 downto 0)
    );
end extender8to16;

architecture behavior of extender8to16 is
begin
    process (in_port)
    begin
        out_port <= ("00000000") & in_port;
    end process;
end behavior;
```

REPRESENTAÇÃO RTL:

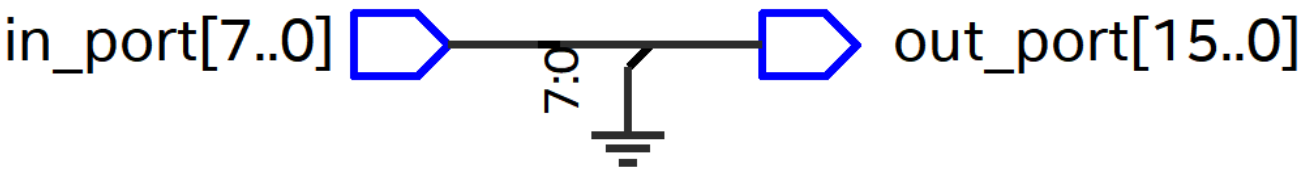
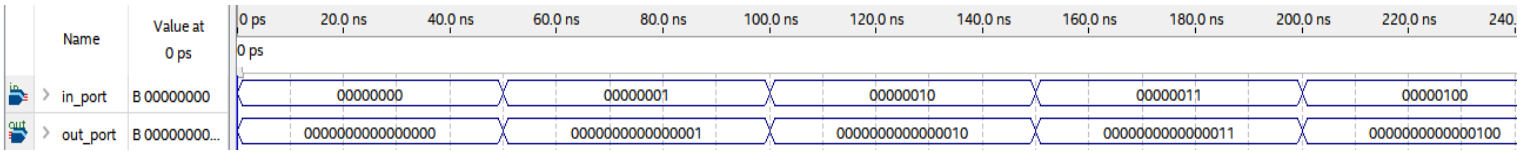


TABELA DE EXECUÇÃO DO EXTENDER8TO16:



LEGENDA:
conversor simples de binário de 8 bits para 16 onde os números em 8bit de “in_port” são transferidos para “out_port” que é STD_LOGIC_VECTOR que comporta 16 bits

[COMPONENTE 05]. Contador Síncrono.

CÓDIGO VHDL DO UPCOUNTER:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.all;
use IEEE.STD_LOGIC_UNSIGNED.all;

entity upcounter is
    Port ( rst,clk : in std_logic;
           o: out std_logic_vector(0 to 3));
end upcounter;

architecture count_arch of upcounter is
   signal count : std_logic_vector(0 to 3);
    begin
      process(rst,clk)
        begin
          if (rst = '1') then count <= "0000";
          elsif (clk'event and clk = '1') then count <= count + 1;
          end if;
         end process;
         o <= count;
   end count_arch;
```

REPRESENTAÇÃO RTL:

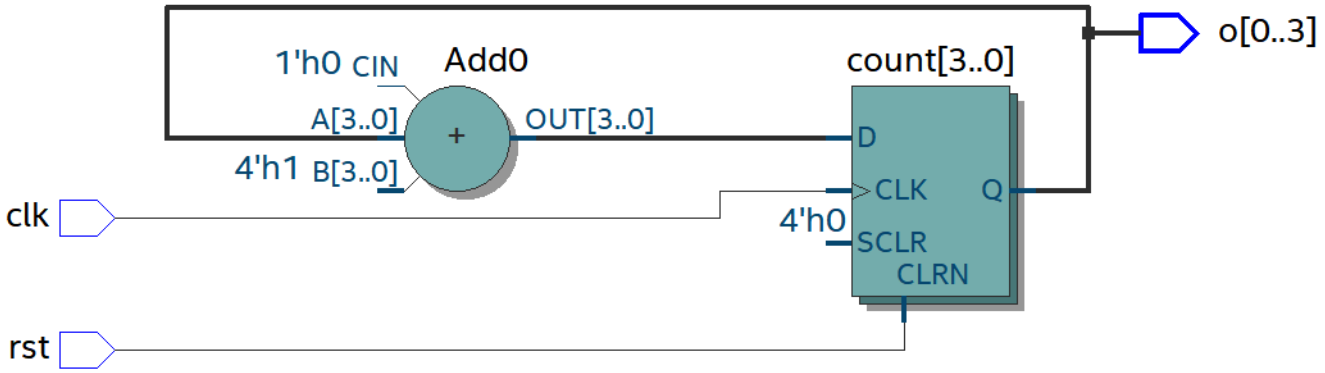
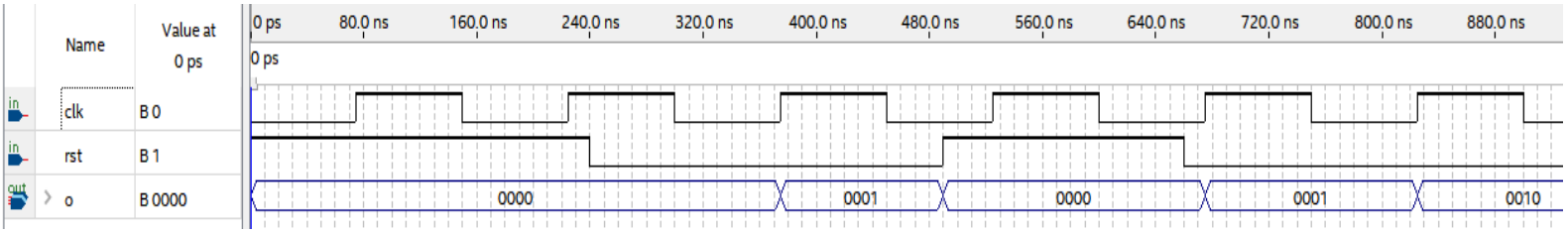


TABELA DE EXECUÇÃO DO UPCOUNTER:



LEGENDA:

Sendo “clk” o clock e “rst” o reset, a cada ciclo de clock de “clk”, “o” que é o contador é incrementado em 1, até um total de “1111”.

Toda vez que “rst” recebe 1, o contador “o” é resetado de volta para “0000” e enquanto “rst” for 1, “o” continuará nesse estado.