Escola de Engenharia de São Carlos
Departamento de Engenharia Elétrica e de Computação

SEL0384 – Laboratório de Sistemas Digitais I

Profa. Luiza Maria Romeiro Codá

Laboratório de Eletrônica Digital SEL-EESC-USP

EESC · USP

Introdução a VHDL

Aula 4

Componente:

- Declaração de Componente
- Instanciação de Componente

Prática 10.a Somador completo de 2 bits usando comando COMPONENT

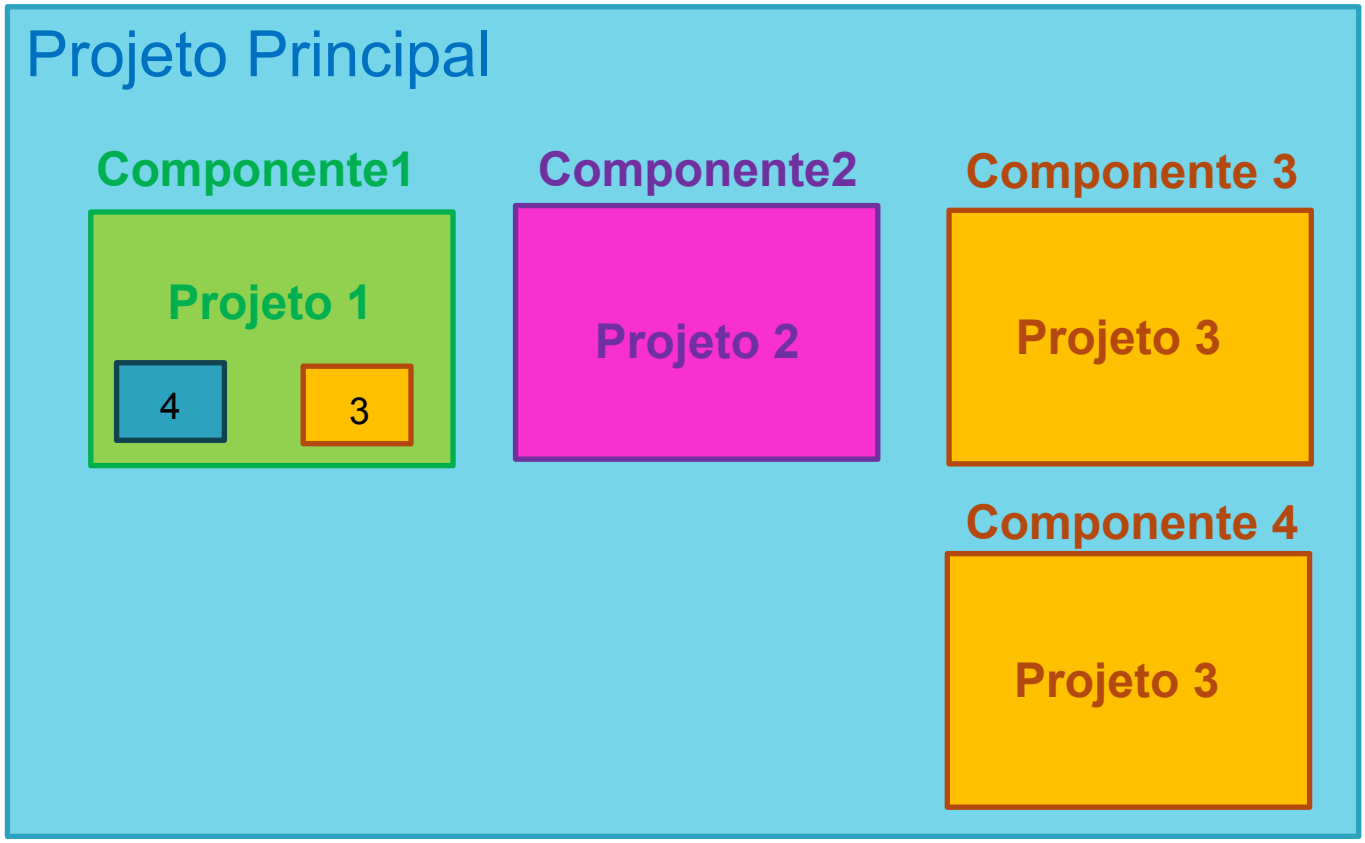
TIPO INTEGER

Cláusura GENERIC

Professora Luiza Maria Romeiro Codá

Componente-

Programação Top Down



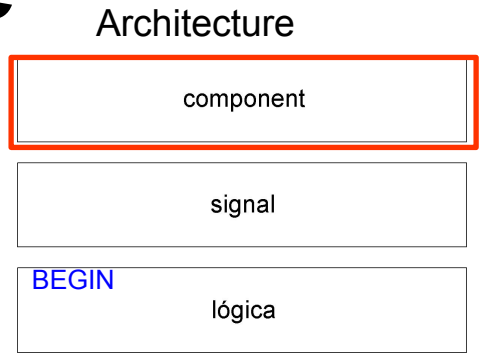
Interconexão de Componente-

A declaração do componente é feita depois da declaração da arquitetura e antes do BEGIN

O mapeamento de entradas e saídas dos componentes com o projeto principal é feito através do comando PORT MAP.

A declaração de um componente pode referenciar uma Entidade descrita em outro projeto VHDL, ou uma Entidade descrita no mesmo arquivo onde será utilizada (projeto atual).

Declaração do Componente



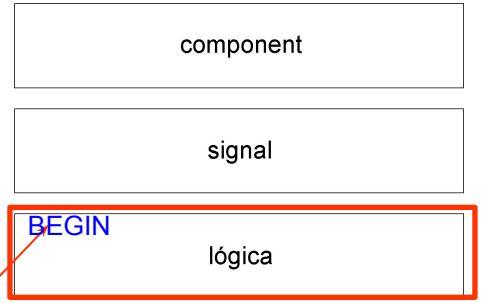
Formato da declaração:

```
COMPONENT <nome_do_componente> IS
PORT(entradas :  IN  BIT;
     saídas   :  OUT BIT);
END COMPONENT;
```

Observação: Cópia da entidade do projeto que será utilizado como Componente, trocando o nome ENTITY por COMPONENT. E fechando com END COMPONENT

Instanciação do Componente

Indica as ligações do componente com os PORTS ou Sinais no projeto principal através do Comando PORT MAP

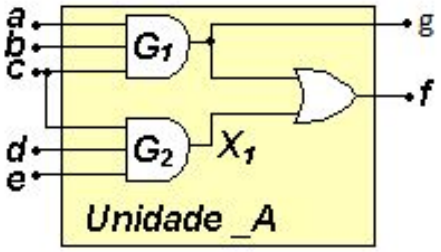


Formato da ligação do componente: Instanciação

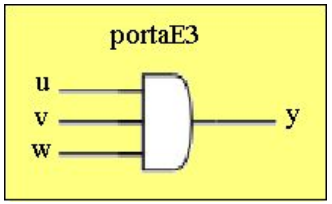
```
-- Instanciação (chamada) de Componentes
<rótulo_do_componente> : <nome_do_componente> PORT MAP(
      <nome_pinos_componente> => <sinal_projeto principal>);
END nome_da_architecture;
```

Instanciação do Componente

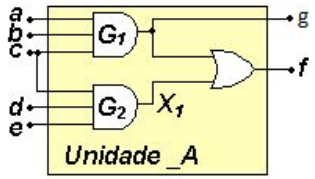
```
ENTITY unidade_A IS
   PORT(a, b, c, d, e     : IN  BIT;
            g             : BUFFER BIT;
            f             : OUT BIT);
END unidade_A;
```




```
ENTITY portaE3 IS
   PORT(u,v,w     : IN  BIT;
              f       : OUT BIT);
END portaE3;
```



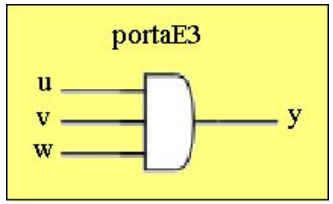
Instanciação do Componente



PORT MAP: especifica as conexões entre ports de uma entity (componente) e sinais na architecture onde o componente foi instanciado.

Existem duas formas de se fazer port map, e não podem ser misturadas:

- associação posicional
  nome do pino do componente => nome do pinodo projeto principal
  Ex: G1 : portaE3 PORT MAP (u=> a, v => b, w => c, y => g) ;

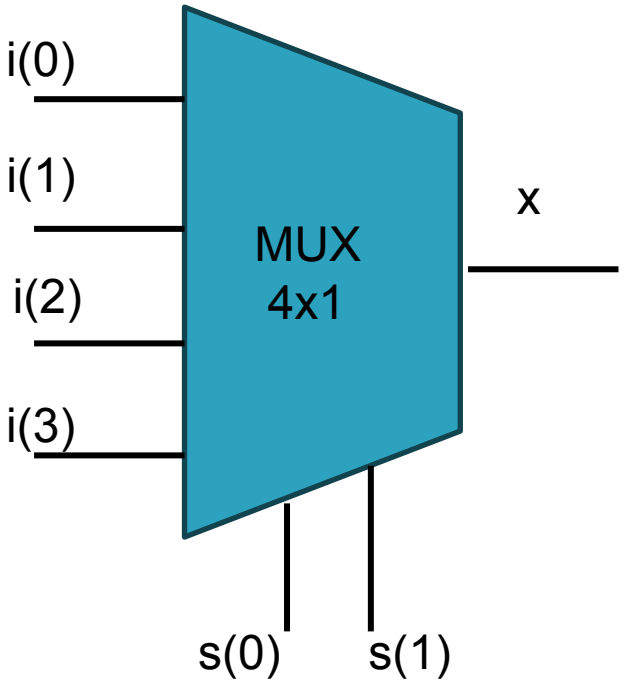


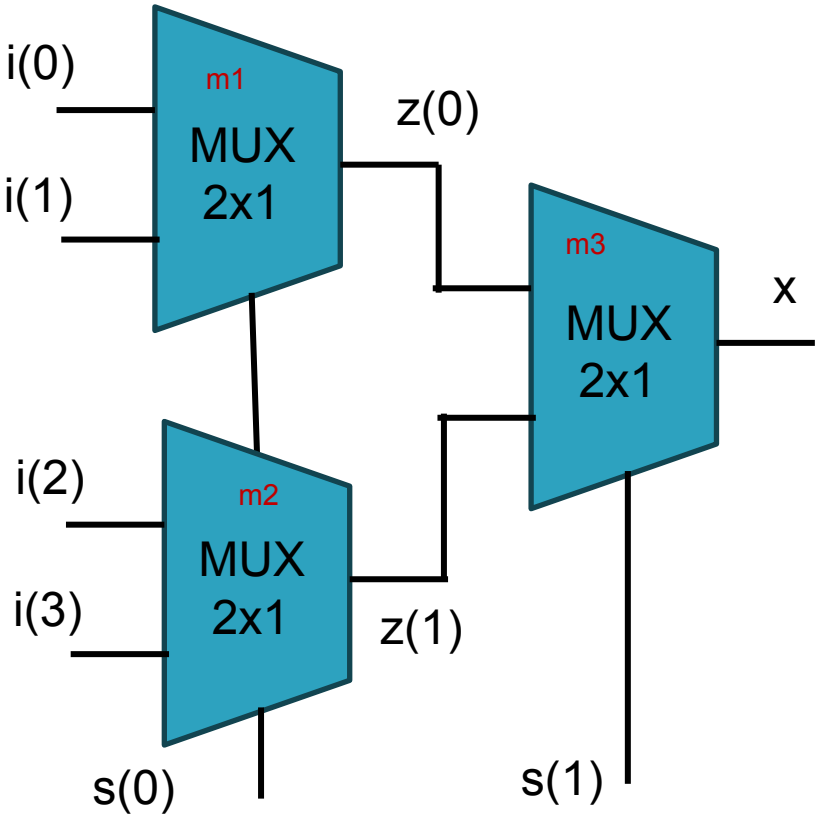
- associação por nome.
  Ex: G1 : portaE3 PORT MAP (a, b, c, g);**--segue a ordem da declaração dos PORTS**

✔ Qualquer combinação de ports e sinais é permitida, desde que haja compatibilidade entre ports e sinais. (devem ser do mesmo tipo)
✔ Todos os elementos do port do componente devem ser associados a algum sinal.
✔ Os ports não conectados podem ser especificados como OPEN
✔ no port map, (um port não usado pode ser deixado omitido no port map, porém não é recomendado).

- Ex: G3 : portaE3 PORT MAP (u => a, v => b, w => c, y => OPEN) ;

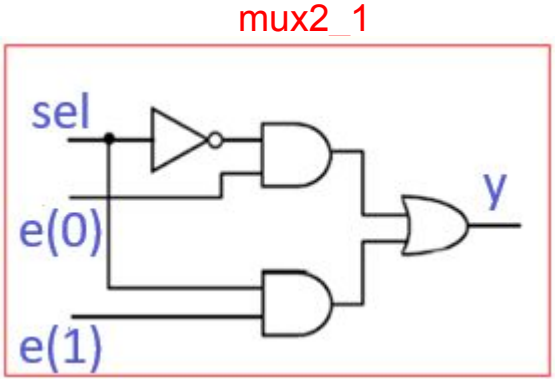
Multiplex 4x1 utilizando componente mux2_1

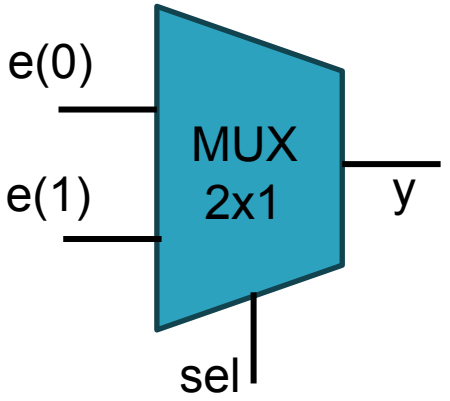




s(1)	s(0)	x
0	0	i(0)
0	1	i(1)
1	0	i(2)
1	1	i(3)

Descrição VHDL de um MUX 2x1

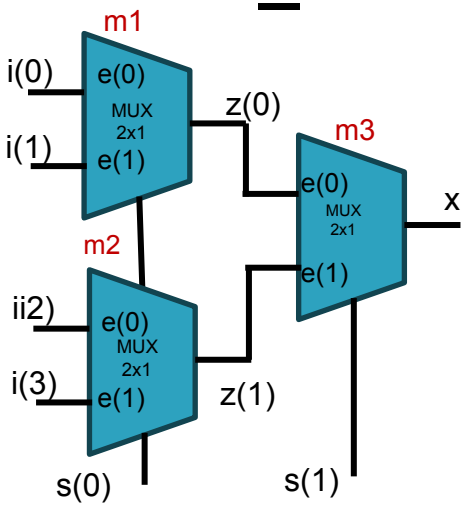




$$y = \overline{sel}.e[0] + sel.e[1]$$

```
ENTITY mux2_1 IS
   PORT(sel      : IN  BIT;
        e        : IN BIT_VECTOR ( 1 DOWNTO 0);
        y        : OUT BIT);
END mux2_1;

ARCHITECTURE logica OF mux2_1 IS
BEGIN
   y <= ( e(0) AND (NOT sel )) OR (e(1) AND sel);

END logica;
```

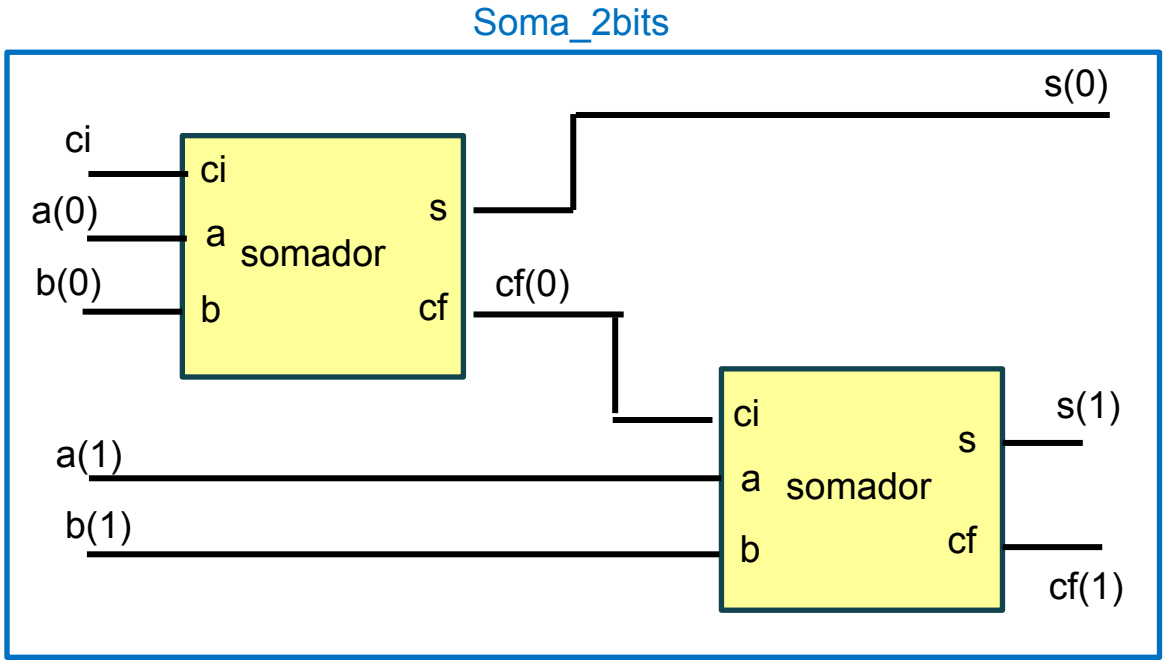
Multiplex 4x1 utilizando componente mux2_1




```
ENTITY mux4_1 IS
   PORT( s        : IN BIT_VECTOR ( 1 DOWNTO 0);
         i        : IN BIT_VECTOR ( 3 DOWNTO 0);
         x        : OUT BIT);
END mux4_1;


 ARCHITECTURE  a OF mux4_1 IS
 SIGNAL z: BIT_VECTOR ( 1 DOWNTO 0);-- Declaração de Sinais
-- Declaração de Componentes
 COMPONENT mux2_1 IS
   PORT(sel      : IN  BIT;
         e        : IN BIT_VECTOR ( 1 DOWNTO 0);
         y        : OUT BIT);
END COMPONENT;
 BEGIN
 -- Instanciação (chamada) de Componentes
 m1: mux2_1 PORT MAP ( sel=>s(0) , e(0)=>i(0) ,e(1) => i(1), y=>z(0) );
 -- Ordem da declaração não importa
 m2: mux2_1 PORT MAP ( sel=>s(0), e(1)=>i(3) , y =>z(1), e(0) =>i(2) );
 m3: mux2_1 PORT MAP (s(1), z ,  x); --  sinais e pinos na ordem do componente mux2_1
 END a;
```

Prática Nº10 VHDL

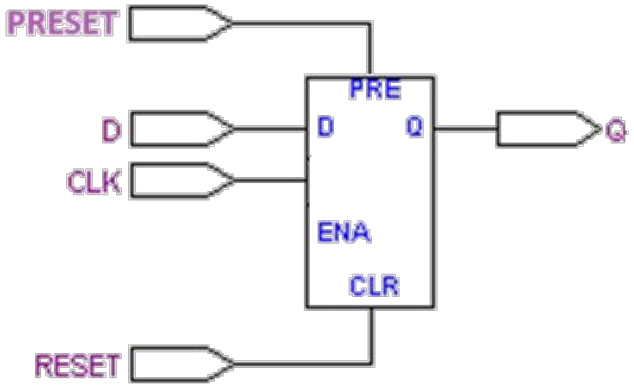
Faça um projeto em VHDL de um somador completo de 2 bits (c), o projeto somador completo de 1 bit (prática nº7) como componente . Escolha o FPGA da placa mercúrio IV, família Cyclone IV-E dispositivo EP4CE30F23C-7 e simule para verificar o funcionamento.



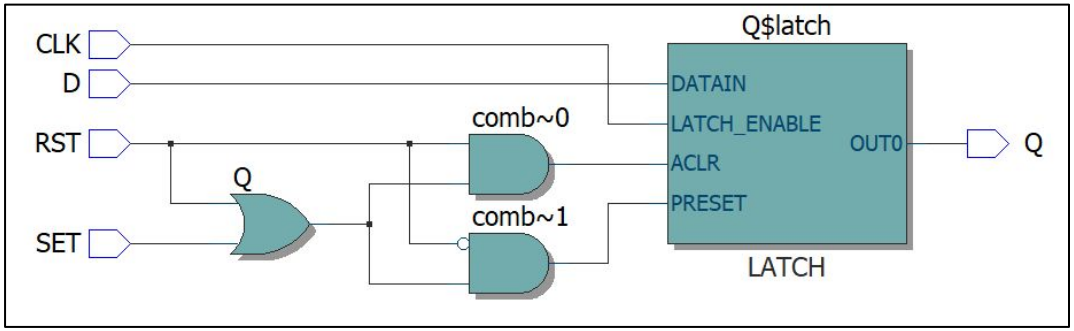
TIPO INTEGER

Descrição de 1 FFs tipo D sensível ao nível alto do clock com RESET e SET Assíncronos
Usando : Tipo BIT e IF-ELSIF-ELSE

```
ENTITY FF_D_nivel IS
    PORT(CLK, RST, SET, D : IN  BIT;
         Q                : OUT BIT);
END FF_D_nivel;

ARCHITECTURE a OF FF_D_nivel IS
BEGIN
    PROCESS (CLK, D, RST, SET)
    BEGIN
        IF (RST = '1') THEN
            Q <= ‘0'; -- Q = 000 independe de CLK e de D
        ELSIF (SET = '1') THEN
            Q <= '1'; -- Q = 111 independe de CLK e de D
        ELSIF (CLK = '1') THEN
            Q <= D;
        END IF;
    END PROCESS;
END a;
```



Circuito Gerado:

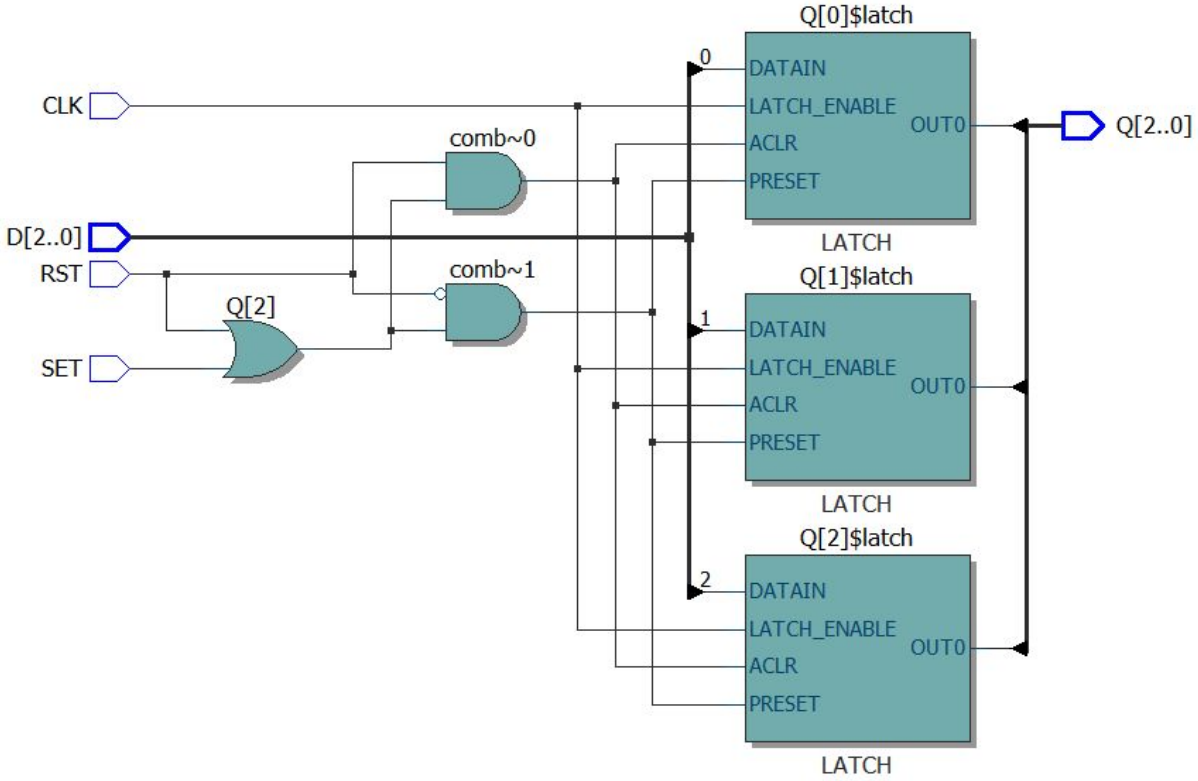


Descrição de 3 FFs tipo D em paralelo sensíveis a nível alto do clock com RESET e SET Assíncronos
Usando : Tipo BIT e IF-ELSIF-ELSE

```
ENTITY FF3_D_nivel IS
    PORT(CLK, RST, SET : IN  BIT;
         D             : IN  BIT_VECTOR(2 DOWNTO 0);
         Q             : OUT BIT_VECTOR(2 DOWNTO 0));
END FF3_D_nivel;

ARCHITECTURE a OF FF3_D_nivel IS
BEGIN
    PROCESS (CLK, D, RST, SET)
    BEGIN
        IF (RST = '1') THEN
            Q <= “000” ; -- Q = 000 independe de CLK e de D
        ELSIF (SET = '1') THEN
            Q <= “111”; -- Q = 111 independe de CLK e de D
        ELSIF (CLK = '1') THEN
            Q <= D;
        END IF;
    END PROCESS;
END a;
```

Circuito Gerado de 3 FFs tipo D em paralelo sensíveis a nível alto do clock com RESET e SET Assíncronos

Usando IF-ELSIF-ELSE



Tipo INTEGER:

É um valor inteiro positivo ou negativo ou nulo (dentro de uma faixa). Apesar de um INTEGER ser, na prática, um vetor de BITs, ele é tratado como um valor indivisível, isto é, não é possível referenciar seus bits separadamente.

São números que variam de $(-2^{31} - 1) \leq x \leq (2^{31} - 1)$.

INTEGER é um número binário com sinal (*signed*).

Exemplo de declaração:

```
X : IN INTEGER RANGE 0 TO 9;  -- X é um vetor de 4 bits
Y : IN INTEGER RANGE 0 TO 10; -- Y é um vetor de 4 bits
SIGNAL Z:  INTEGER; -- Z é um vetor de 32 bits
```

Portanto , o tipo INTEGER possibilita que X, Y e Z, vetores de 4 e 32 *bits*, possam ser tratados como números inteiros.

Obs.: O tipo NATURAL é um subconjunto do tipo INTEGER, e exclui os números negativos ($n \geq 0$). Valores NATURAL: 0 até $2^{31} - 1$

OPERADORES usados no Tipo INTEGER:

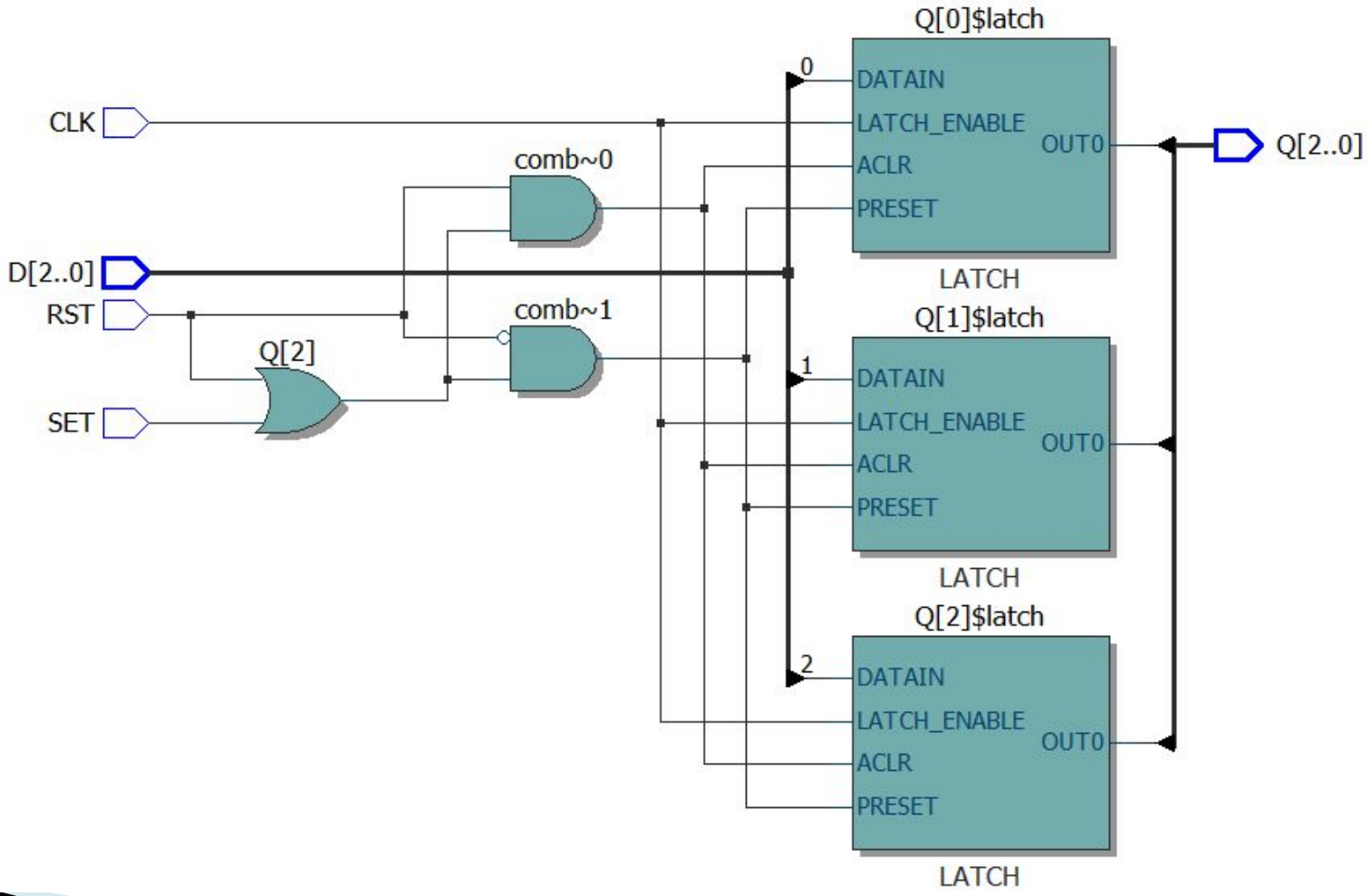
Operador aritméticos	Operação	Exemplo de atribuiçao
+	adição	a <= a + 2
-	subtração	a <= a -2
*	multiplicação	a <= 2*n
/	Divisão	a <= n/2
**	potenciação	a <= a**3
mod	módulo	a := mod (t)
rem	resto da divisão	a <= rem(d)

Operadores relacionais	Operação
=	igual
/=	diferente
>	maior que
<	menor que
>=	maior ou igual
<=	menor ou igual

Descrição de 3 FFs tipo D em paralelo sensíveis a nível alto do clock com RESET e SET Assíncronos
Usando : Tipo INTEGER e IF-ELSIF-ELSE

```
ENTITY FF3_D_nivel_inteiro IS
    PORT(CLK, RST, SET : IN  BIT;
         D             : IN  INTEGER RANGE 0 TO 7;
         Q             : OUT INTEGER RANGE 0 TO 7);
END FF3_D_nivel_inteiro;

ARCHITECTURE a OF FF3_D_nivel_inteiro IS
BEGIN
    PROCESS (CLK, D, RST, SET)
    BEGIN
        IF (RST = '1') THEN
            Q <= 0 ; -- Equivale Q = 000 e independe de CLK e de D
        ELSIF (SET = '1') THEN
            Q <= 7; -- Equivale Q = 111 e independe de CLK e de D
        ELSIF (CLK = '1') THEN
            Q <= D;
        END IF;
    END PROCESS;
END a;
```

Circuito Gerado de 3 FFs tipo D em paralelo sensíveis a nível alto do clock com RESET e SET Assíncronos
Usando : Tipo INTEGER e IF-ELSIF-ELSE



Cláusula GENERIC

- Declarado na ENTITY para definir uma constante;
- similar a CONSTANT, porém é definido na entidade e não na arquitetura;
- Seu âmbito é global.
- pode ser mapeado para outro valor, quando importado como componente;
- Formato:

```
GENERIC(<nome> : <TIPO> := <Valor_Inicial>);
```

Obs: Não é estritamente necessário atribuir um valor inicial a Genéricos, no entanto, se em nenhum momento for atribuído um valor a um Genérico, será gerado uma mensagem de Erro no *software*.

GENERIC - Exemplo

A declaração de Genéricos é feita antes da expressão PORT, na declaração da Entidade:

```
ENTITY Divisor_2n IS

  GENERIC(num_reg : INTEGER := 9);

  PORT(clk_in  : IN  BIT;
       Q       : OUT BIT_VECTOR(num_reg DOWNTO 0);

END Divisor_2n;
```

GENERIC – Atribuição de valores

O valor de um Genérico pode ser atribuído em diversos pontos da descrição. Os principais são:

- Declaração da Entidade
- Declaração do Componente
- Solicitação do Componente

Uma vez que o valor do Genérico pode ser especificado em mais de um local, existe uma regra de prioridade para decidir qual valor será usado:

O valor usado será o mais específico.

Isto significa que a prioridade para atribuir o valor ao Genérico é:

1. Solicitação do Componente
2. Declaração do Componente
3. Declaração da Entidade

GENERIC – Atribuição de valores

Prioridade de atribuição de valor

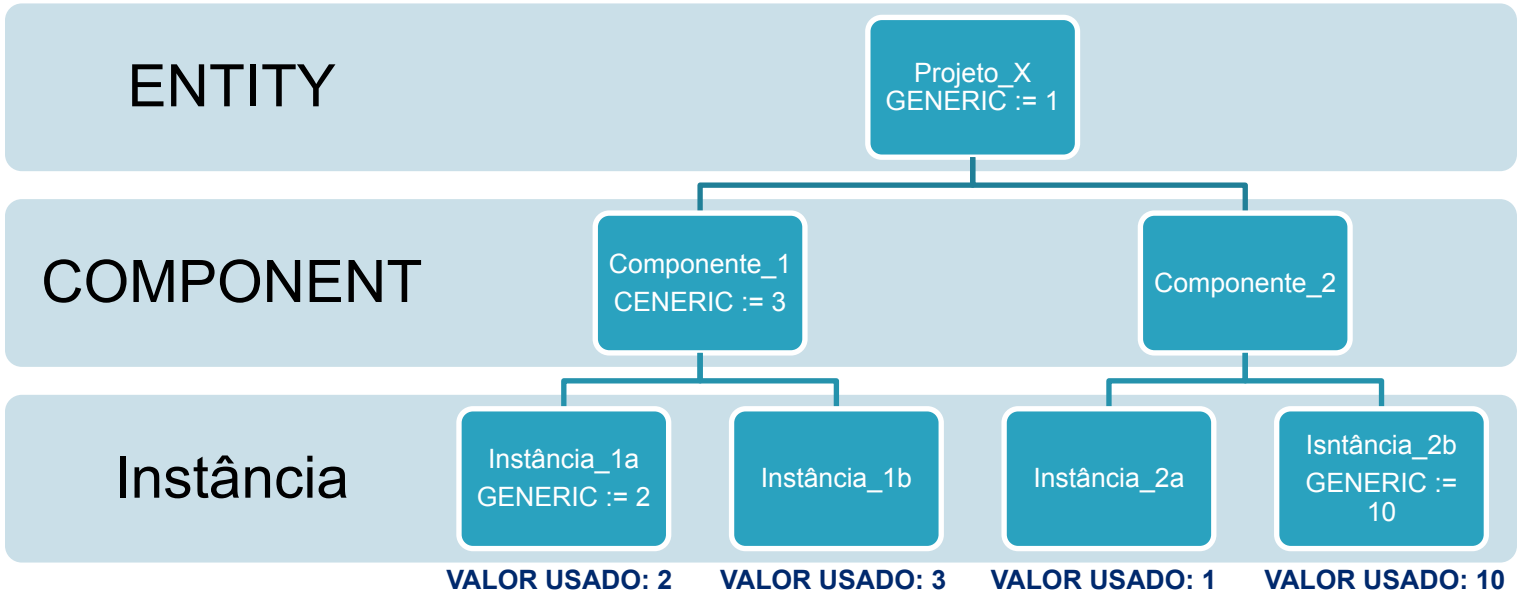
1. Solicitação do Componente
2. Declaração do Componente
3. Declaração da Entidade

Deste modo, por exemplo, se foi atribuído o valor "2" ao genérico na declaração da Entidade, e durante a declaração de um Componente que usa esta Entidade for atribuído o valor "3", todas as instâncias deste Componente utilizarão o valor "3", e não o valor "2".

Adicionalmente, se durante uma solicitação (instanciação) do Componente citado for novamente atribuído um valor ao Genérico ("4", por exemplo), então este valor será utilizado para esta instância específica do Componente.

GENERIC – Atribuição de valores

Ilustração do esquema de prioridades:



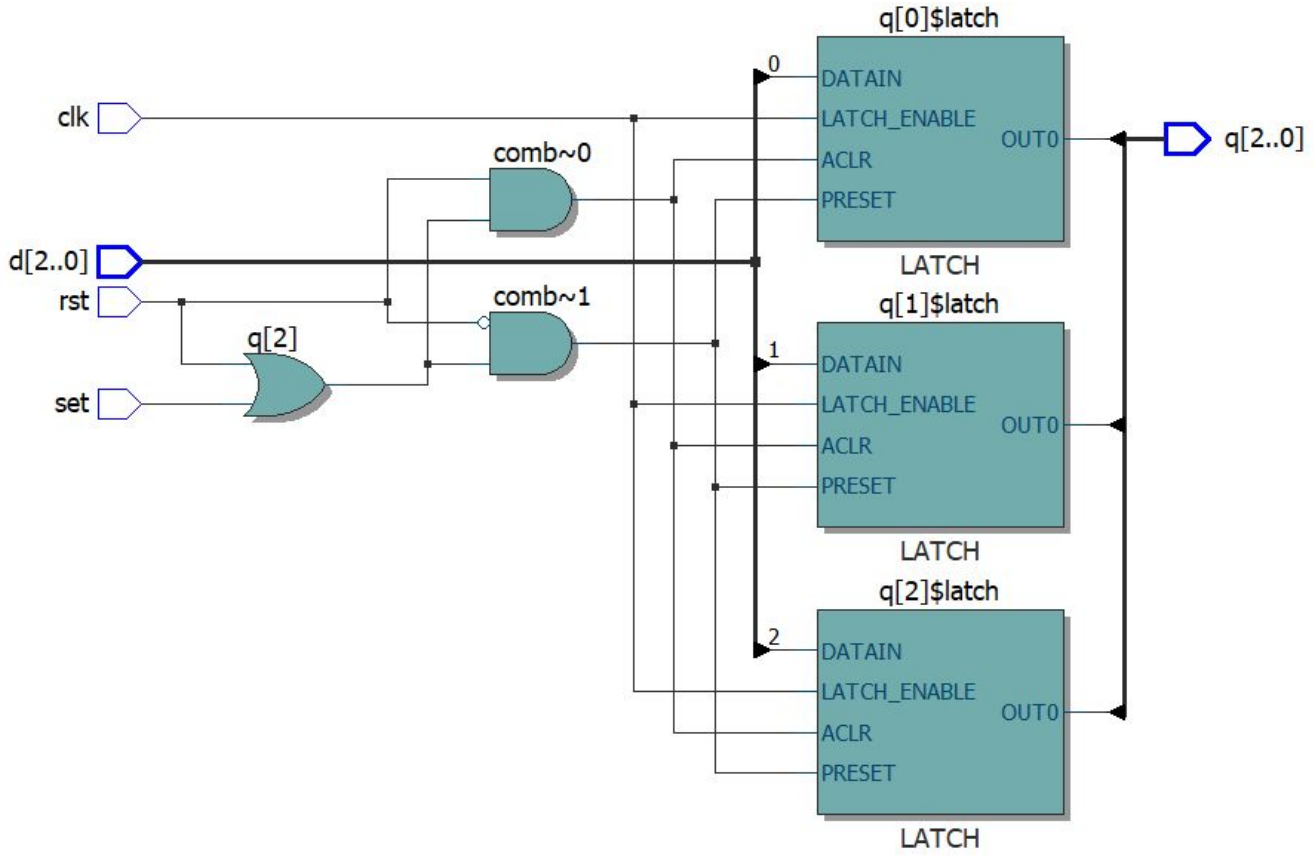
FFs tipo D em paralelo sensíveis a nível alto do clock RESET e SET Assíncronos Usando IF-ELSIF-ELSE, INTEGER e GENERIC

N = 3

```
ENTITY FF_D_generico IS
    GENERIC(n : NATURAL := 3);
    PORT(clk, rst, set : IN  BIT;
         d             : IN  INTEGER RANGE 0 TO (2**n) - 1;
         q             : OUT INTEGER RANGE 0 TO (2**n) - 1);
END FF_D_generico;

ARCHITECTURE a OF FF_D_generico IS
BEGIN
    PROCESS(clk, d, rst, set)
    BEGIN
        IF (rst = '1') THEN
            q <= 0;
        ELSIF (set = '1') THEN
            q <= (2**n) - 1; -- (2^3 - 1) = 111
        ELSIF (clk = '1') THEN
            q <= d;
        END IF;
    END PROCESS;
END a;
```

FFs tipo D em paralelo sensíveis a nível alto do clock RESET e SET Assíncronos Usando IF-ELSIF-ELSE, INTEGER e GENERIC

Circuito Gerado com N=3



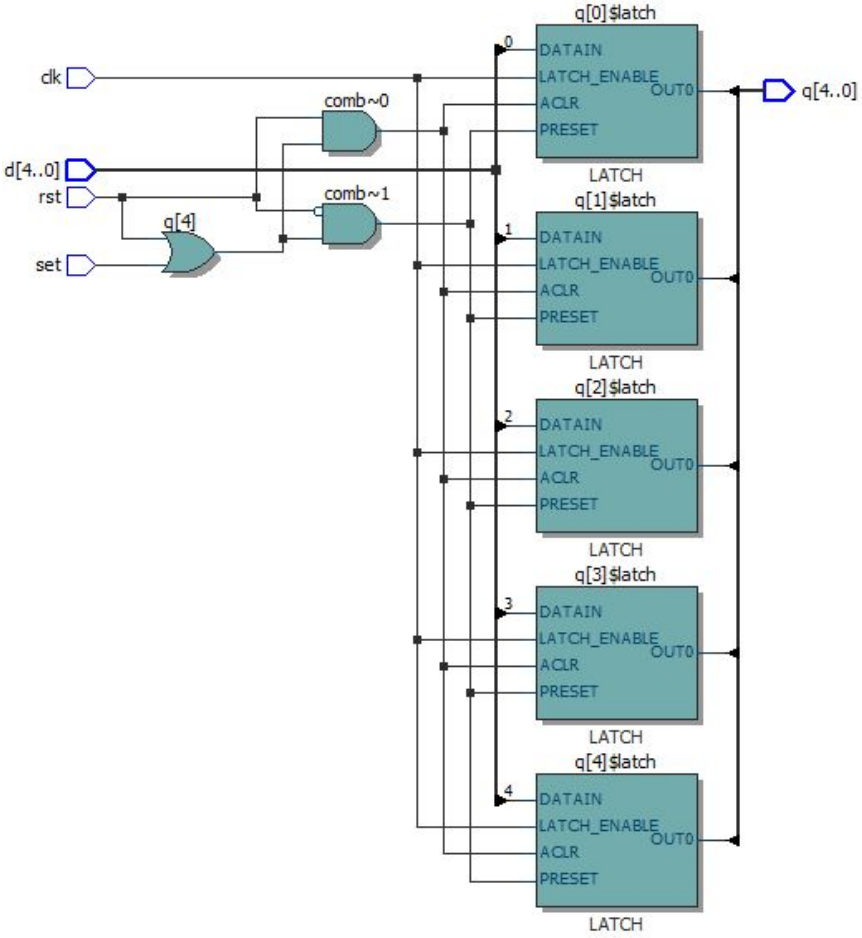
FFs tipo D em paralelo sensíveis a nível alto do clock RESET e SET Assíncronos Usando IF-ELSIF-ELSE, INTEGER e GENERIC

N = 5

```
ENTITY FF_D_generico IS
    GENERIC(n : NATURAL := 5);
    PORT(clk, rst, set : IN  BIT;
         d             : IN  INTEGER RANGE 0 TO (2**n) - 1;
         q             : OUT INTEGER RANGE 0 TO (2**n) - 1);
END FF_D_generico;

ARCHITECTURE a OF FF_D_generico IS
BEGIN
    PROCESS(clk, d, rst, set)
    BEGIN
        IF (rst = '1') THEN
            q <= 0;
        ELSIF (set = '1') THEN
            q <= (2**n) - 1; -- (2^5 - 1) = 11111
        ELSIF (clk = '1') THEN
            q <= d;
        END IF;
    END PROCESS;
END a;
```

FFs tipo D em paralelo sensíveis a nível alto do clock RESET e SET Assíncronos Usando IF-ELSIF-ELSE, INTEGER e GENERIC

Circuito Gerado com N=5



Mapeamento de Genéricos na Solicitação de Componentes -Comando GENERIC MAP

A declaração de Componentes que possuem Genéricos em suas Entidades segue o seguinte formato:

```
COMPONENT <nome_do_componente> IS
GENERIC(<generico_x> : tipo := <valor_inicial_x>;
    < generico_y> : tipo := <valor_inicial_y>;
        <generico_z> : tipo);
PORT(...);
END COMPONENT;
```

A Instanciação, por usa vez, segue o padrão a seguir:

```
-- Instanciação (Solicitação) de Componentes
<rótulo> : <nome_do_componente> GENERIC MAP(<valor_x>,<valor_y>, <valor_z>)
                                PORT MAP(...);
-- Forma alternativa
<rótulo> : <nome_do_componente> GENERIC MAP(<generico_z> => <Valor_z>)
                                PORT MAP(...);
-- Sem alterar nenhum valor caso todos genéricos já tenham valor inicial
<rótulo> : <nome_do_componente> PORT MAP(...);
```

Mapeamento de Genéricos na Solicitação de Componentes - Comando GENERIC MAP - Exemplo

```
COMPONENT Circuito IS -- Declaração do Componente
GENERIC(largura : INTEGER := 3; -- Atribui valor inicial 3
        comprimento : INTEGER); -- Não atribui valor inicial
PORT(...);
END COMPONENT;

-- Instanciação (Solicitação) de Componentes
-- Atribui valores 8 e 12 aos genéricos largura e comprimento
X1 : Circuito GENERIC MAP(8, 12) PORT MAP(...);

-- Atribui valores 8 e 12 aos genéricos largura e comprimento
X2 : Circuito GENERIC MAP(comprimento => 12, largura => 8) PORT MAP(...);

-- Atribui valor 7 ao genérico comprimento
X3 : Circuito GENERIC MAP(comprimento => 7) PORT MAP(...);

-- Sem alterar nenhum Generic caso o generic comprimento tenha valor inicial
X4 : Circuito PORT MAP(...);
```

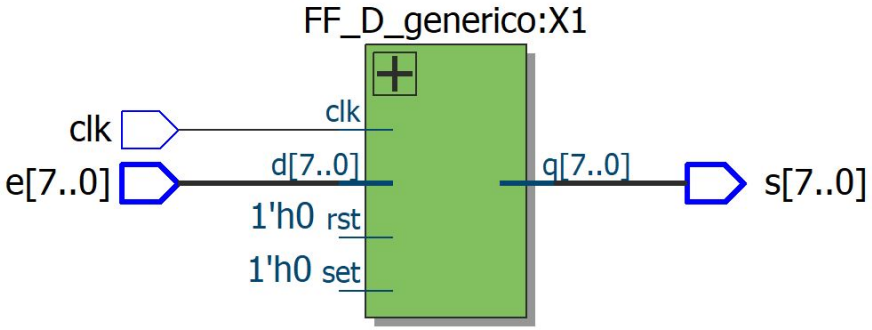
Exemplo: Mapeamento de Genericos utilizando o projeto FFs tipo D como COMPONENT

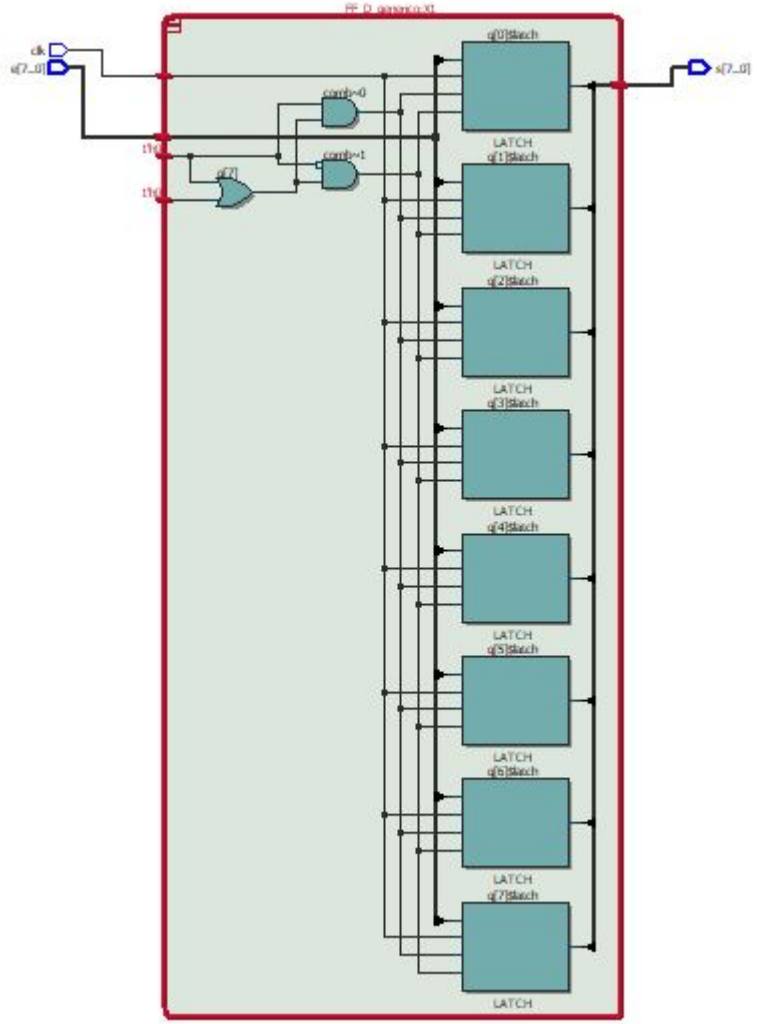
```
ENTITY FF_D_generico IS
    GENERIC(n : NATURAL := 3);
    PORT(clk, rst, set : IN  BIT;
         d             : IN  INTEGER RANGE 0 TO (2**n) - 1;
         q             : OUT INTEGER RANGE 0 TO (2**n) - 1);
END FF_D_generico;

ARCHITECTURE a OF FF_D_genérico IS
BEGIN
    PROCESS(clk, d, rst, set)
    BEGIN
        IF (rst = '1') THEN
            q <= 0;
        ELSIF (set = '1') THEN
            q <= (2**n) - 1; -- (2^3 - 1) = 111
        ELSIF (clk = '1') THEN
            q <= d;
        END IF;
    END PROCESS;
END a;
```

Mapeamento de Genéricos na Solicitação de Componentes - Comando GENERIC MAP – Instanciação de FF tipo D de 8 bits

```
ENTITY exemplo IS
    PORT(clk: IN BIT;
         e  : IN  INTEGER RANGE 0 TO 255;
         s : OUT INTEGER RANGE 0 TO 255
         );
END exemplo;

ARCHITECTURE a OF exemplo IS
    COMPONENT FF_D_generico IS -- Entidade "FF_D_generico" descrita anteriormente.
    -- Atribui o valor 1 ao genérico n, substituindo o valor (3) atribuído
    -- anteriormente na declaração da entidade do componente
        GENERIC(n : NATURAL := 1);
        PORT(clk, rst, set : IN  BIT;
             d             : IN  INTEGER RANGE 0 TO (2**n) - 1;
             q             : OUT INTEGER RANGE 0 TO (2**n) - 1);
    END COMPONENT;

    SIGNAL ground : BIT:= '0';

BEGIN
    -- Atribui o valor 8 ao genérico n, substituindo o valor (1) atribuído
    -- acima na declaração do componente
    X1 : FF_D_generico GENERIC MAP(8) PORT MAP(clk, ground, ground, e,s);


END a;
```

Mapeamento de Genéricos na Solicitação de Componentes - Comando GENERIC MAP – Instanciação de FF tipo D de 8 bits





Prática nº10.b
Utilizar Template de um contador do QUARTUS II
Usar clausula GENERIC

Ver Roteiro da prática

Template de um contador

```
ENTITY __entity_name IS
        PORT(   __data_input_name           : IN    INTEGER RANGE 0 TO __count_value;
                __clk_input_name            : IN    BIT;
                __clrn_input_name           : IN    BIT;
                __ena_input_name            : IN    BIT;
                __ld_input_name             : IN    BIT;
                __count_output_name  : OUT  INTEGER RANGE 0 TO __count_value          );
  END __entity_name;
ARCHITECTURE a OF __entity_name IS
        SIGNAL          __count_signal_name  : INTEGER RANGE 0 TO __count_value;
    BEGIN
        PROCESS (__clk_input_name, __clrn_input_name)
        BEGIN
                IF __clrn_input_name = '0' THEN
                        __count_signal_name <= 0;

                ELSIF (__clk_input_name'EVENT AND __clk_input_name = '1') THEN

                IF __ld_input_name = '1' THEN

                        __count_signal_name <= __data_input_name;

                ELSE

                        IF __ena_input_name = '1' THEN

                                __count_signal_name <= __count_signal_name + 1;

                        ELSE

                                __count_signal_name <= __count_signal_name;

                        END IF;

                END IF;

                END IF;

        END PROCESS;
END a;
```

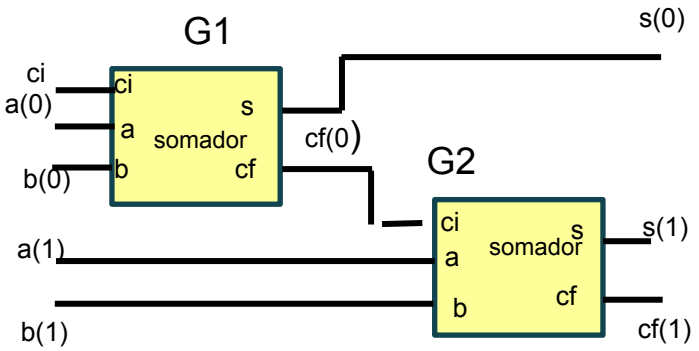
RESOLUCÃO DAS PRÁTICAS

Resolução: Prática Nº10 VHDL

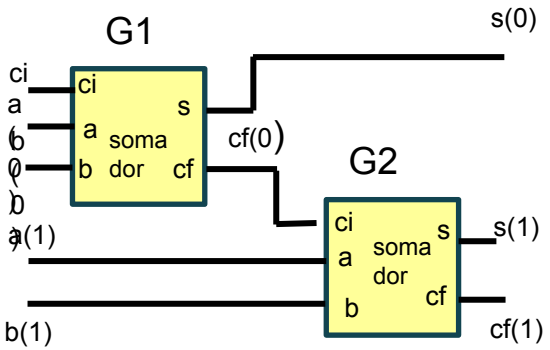
```
ENTITY somador IS
   PORT(ci, a, b : IN  BIT;
                s, cf : OUT  BIT);
END somador;
ARCHITECTURE fluxo_dados OF somador IS
   BEGIN
      s <= (a XOR b XOR ci);
      cf <= (a AND b) OR (ci AND (a OR b ));
END fluxo_dados;
```

Resolução: Prática Nº10 VHDL

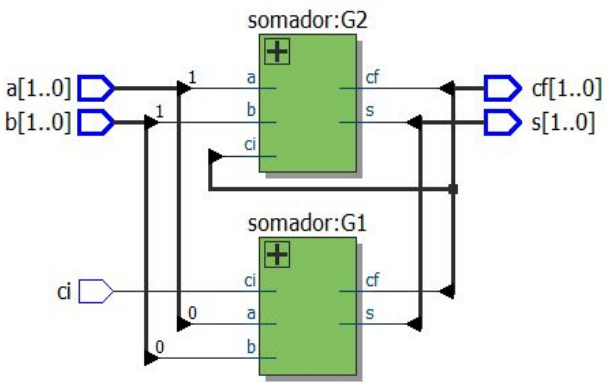
```
ENTITY soma_2bits IS
   PORT(   ci : IN BIT;
           a, b : IN  BIT_VECTOR (1 DOWNTO 0);
           s: OUT BIT_VECTOR (1 DOWNTO 0);
           cf : BUFFER BIT_VECTOR (1 DOWNTO 0)

        );
END soma_2bits;
ARCHITECTURE a1 OF soma_2bits IS

COMPONENT somador IS
   PORT(ci, a, b : IN  BIT;
            s, cf : OUT BIT);
END COMPONENT;

BEGIN
G1: somador PORT MAP (ci, a(0), b(0), s(0), cf(0) );
G2: somador PORT MAP (cf(0), a(1), b(1), s(1), cf(1) );

END a1;
-- Incluir aqui a Descrição VHDL do projeto somador
```

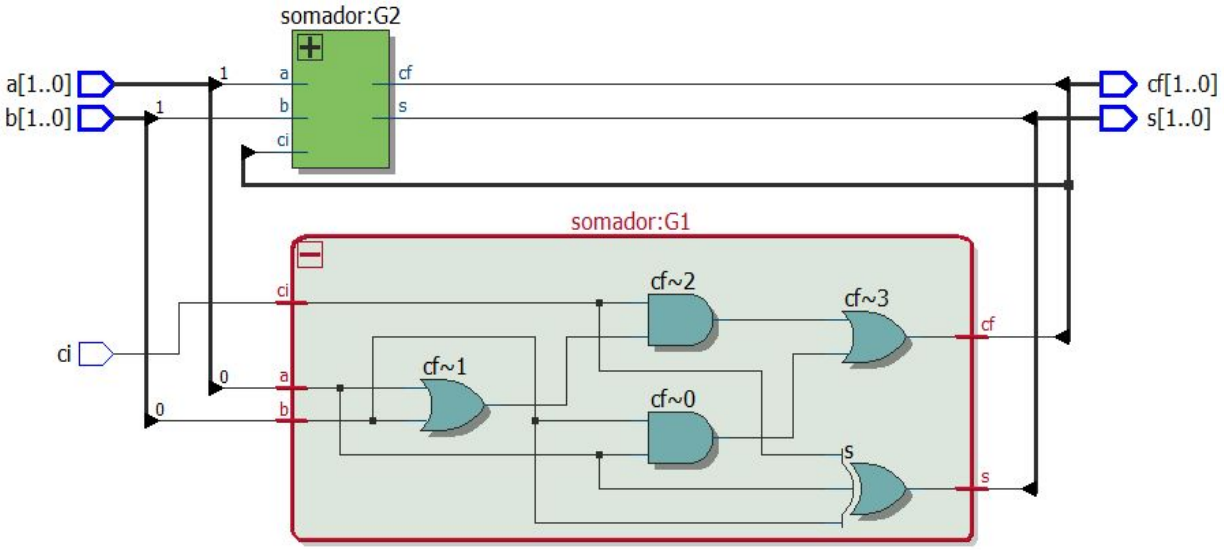


Resolução: Prática Nº10 VHDL



Circuito Gerado:





RESOLUÇÃO: Prática nº10.b

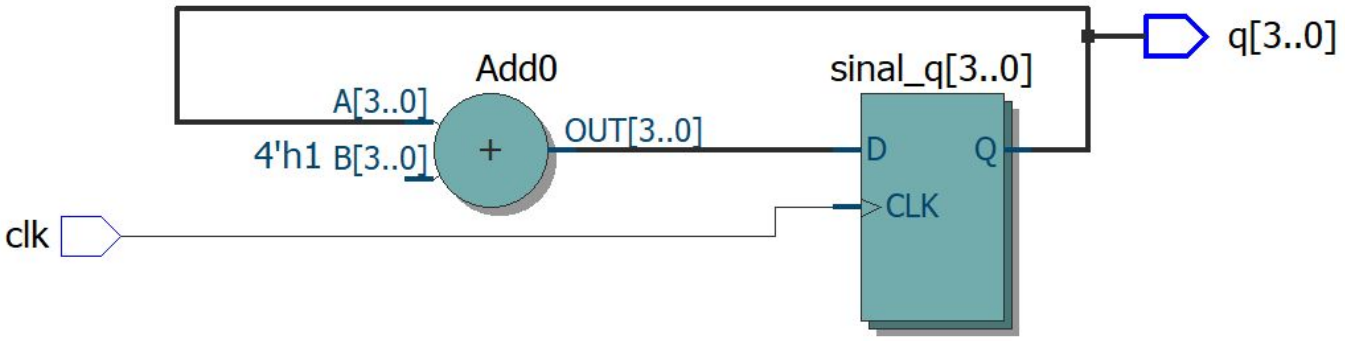
Item 1 : Contador Binário de 4 bits

```
ENTITY upcount IS
        PORT
        (       clk     : IN    BIT;
                q       : OUT   BIT_VECTOR(3 DOWNTO 0));
END upcount;

ARCHITECTURE a OF upcount IS
        SIGNAL sinal_q  : INTEGER RANGE 0 TO 15;
BEGIN
        P1:PROCESS (clk)
        BEGIN
                IF (clk'EVENT AND clk = '1') THEN
                        sinal_q <= sinal_q + 1;
                ELSE
                        sinal_q <= sinal_q;
                END IF;
        END PROCESS P1;

        P2: PROCESS (sinal_q)
            BEGIN
           CASE sinal_q IS
              WHEN 0 => q <= "0000";
              WHEN 1 => q <= "0001";
              WHEN 2 => q <= "0010";
              WHEN 3 => q <= "0011";
              WHEN 4 => q <= "0100";
              WHEN 5 => q <= "0101";
              WHEN 6 => q <= "0110";
              WHEN 7 => q <= "0111";
              WHEN 8 => q <= "1000";
             WHEN 9 => q <= "1001";
              WHEN 10 => q <= "1010";
              WHEN 11 => q <= "1011";
              WHEN 12 => q <= "1100";
              WHEN 13 => q <= "1101";
              WHEN 14 => q <= "1110";
              WHEN 15 => q <= "1111";
           END CASE;
        END PROCESS P2;
END a;
```

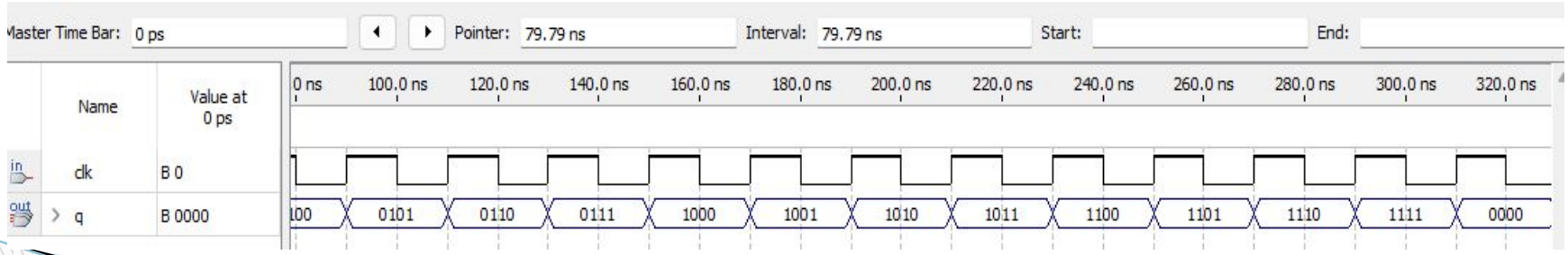
RESOLUÇÃO: Prática nº10.b

Item 1 : Contador Binário de 4 bits

Circuito gerado:



Simulação do contador binário de 4 bits



RESOLUÇÃO: Prática nº10.b

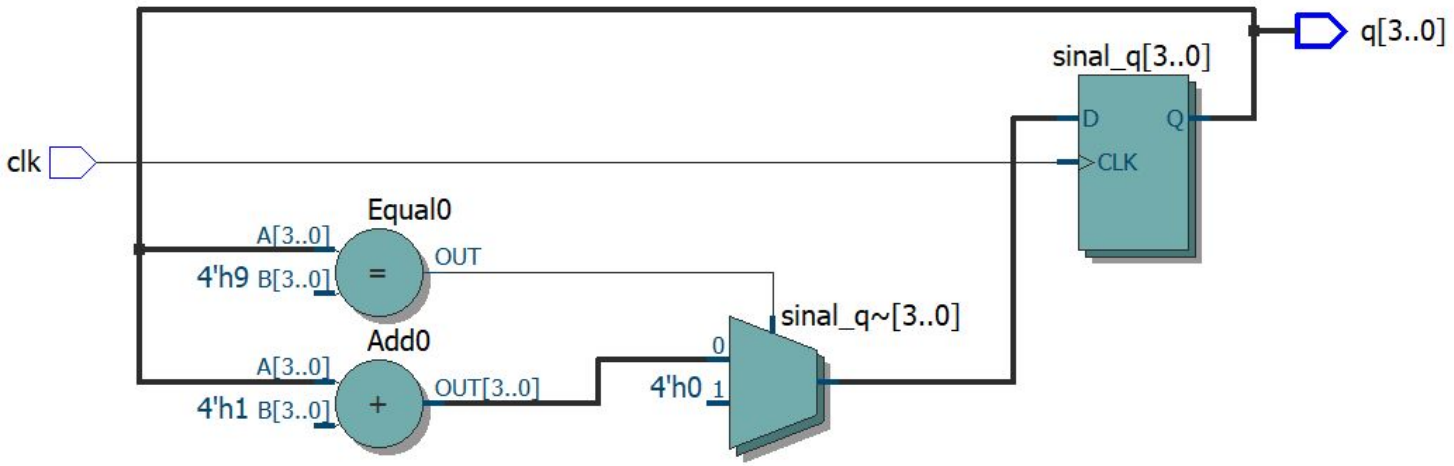
Item 2 : Contador de década

```
ENTITY upcount IS
        PORT
        (       clk     : IN    BIT;
                q       : OUT   BIT_VECTOR(3 DOWNTO 0));
END upcount;

ARCHITECTURE a OF upcount IS
        SIGNAL sinal_q  : INTEGER RANGE 0 TO 15;
BEGIN
        P1:PROCESS (clk)
        BEGIN
            IF (clk'EVENT AND clk = '1') THEN
               IF sinal_q = 9 THEN
                        sinal_q <= 0;
               ELSE
                        sinal_q <= sinal_q + 1;
                                END IF;
             ELSE
                                sinal_q <= sinal_q;
            END IF;
        END PROCESS P1;

        P2: PROCESS (sinal_q)
            BEGIN
           CASE sinal_q IS
              WHEN 0 => q <= "0000";
              WHEN 1 => q <= "0001";
              WHEN 2 => q <= "0010";
              WHEN 3 => q <= "0011";
              WHEN 4 => q <= "0100";
              WHEN 5 => q <= "0101";
              WHEN 6 => q <= "0110";
              WHEN 7 => q <= "0111";
              WHEN 8 => q <= "1000";
             WHEN 9 => q <= "1001";
              WHEN 10 => q <= "1010";
              WHEN 11 => q <= "1011";
              WHEN 12 => q <= "1100";
              WHEN 13 => q <= "1101";
              WHEN 14 => q <= "1110";
              WHEN 15 => q <= "1111";
           END CASE;
        END PROCESS P2;
END a;
```

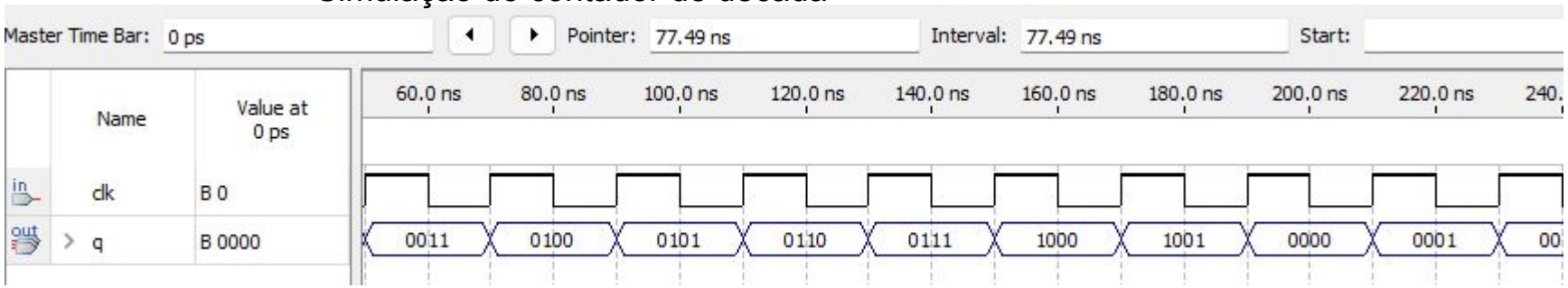
RESOLUÇÃO: Prática nº10.b

Item 2 : Contador de Década

Circuito gerado:



Simulação do contador de década



RESOLUÇÃO: Prática nº10.b

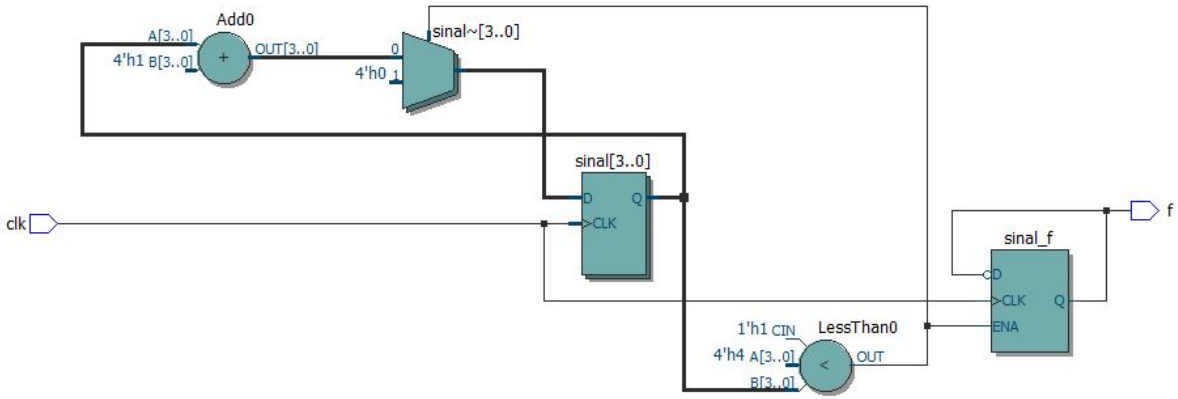
Item 3 : Divisor de frequência Obter 1Hz a partir de 5Mhz

```
ENTITY divfreq IS
GENERIC( maxvalue1:INTEGER:=10 ); - se maxvalue = 50 000 000  e a freq. Do clock =50MHz, f= 1Hz
        PORT(
                clk     : IN    BIT;
                   f    : OUT BIT);
END divfreq;

ARCHITECTURE a OF divfreq IS
        SIGNAL      sinal   : INTEGER RANGE 0 TO maxvalue1;
        SIGNAL   sinal_f : BIT;
BEGIN
   PROCESS (clk)
        BEGIN
                IF (clk'EVENT AND clk = '1') THEN
                        IF (sinal >= (2*maxvalue1-1)/2 ) THEN
                                sinal <=0;
                                sinal_f<= NOT sinal_f;
                        ELSE
                                        sinal <= sinal + 1;
                        END IF;
                ELSE
                        sinal <= sinal;

                END IF;

   END PROCESS;

                f <= sinal_f;
END a;
```

RESOLUÇÃO: Prática nº10.b

Item 3 : Divisor de frequência : com maxvalue = 10 divide a frequência por 10

Circuito gerado:



Simulação do divisor de frequência