Laboratório de Eletrônica Digital
SEL-EESC-USP

EESC · USP

# Introdução a VHDL

## Aula 4 – SALA

**Professora Luiza Maria Romeiro Codá**

# Aula 4: Introdução a VHDL

## Conteúdo:

- Comando sequencial: NULL
- Conceito de Atributo: 'EVENT RISING_EDGE(sinal) FALLING_EDGE(sinal)
- Tipo INTEGER
- Cláusula GENERIC
- Mapeamento de GENERIC
- Conversão entre TIPOS
- Prática nº6: FF tipo T
- Prática nº7: contador binário

# Comando NULL

# Comandos em VHDL – Sequenciais NULL

O comando "NULL" não realiza nenhuma operação, a execução é passada para o próximo comando.
Serve para indicar que nenhuma operação deve ser realizada em uma condição de CASE-WHEN ou IF-ELSE.

# Comandos em VHDL – Sequenciais:NULL

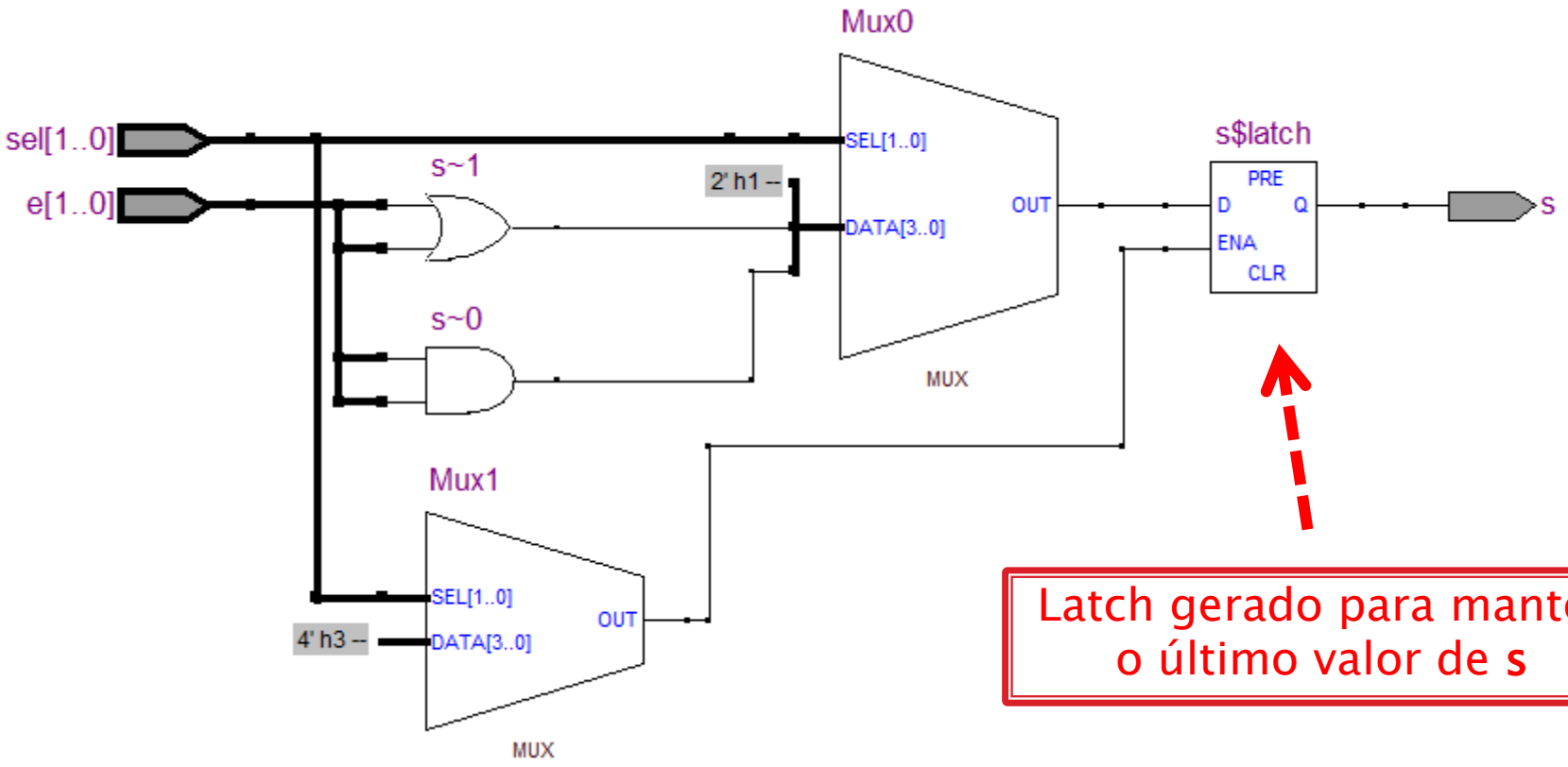
## Exemplo de Aplicação com CASE-WHEN

Na descrição a seguir, a saída **s** recebe o resultado de uma entre duas operações lógicas, selecionadas pelo sinal **sel** ("00" indica AND e "01" indica OR). Caso **sel** seja diferente destes valores, a saída não deve ser alterada.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY ex_mux1 IS
    PORT(e, sel : IN  STD_LOGIC_VECTOR (1 DOWNTO 0);
         s      : OUT STD_LOGIC);
END ex_mux1;

ARCHITECTURE a OF ex_mux1 IS
BEGIN
    PROCESS (e, sel)
    BEGIN
        CASE sel IS
            WHEN "00" => s <= e(0) AND e(1);
            WHEN "01" => s <= e(0) OR  e(1);
            WHEN OTHERS => NULL; -- saída s não se altera
        END CASE;
    END PROCESS;
END a;
```

# Comandos em VHDL – Sequenciais: NULL

## Exemplo de Aplicação com CASE-WHEN




Latch gerado para manter o último valor de **s**

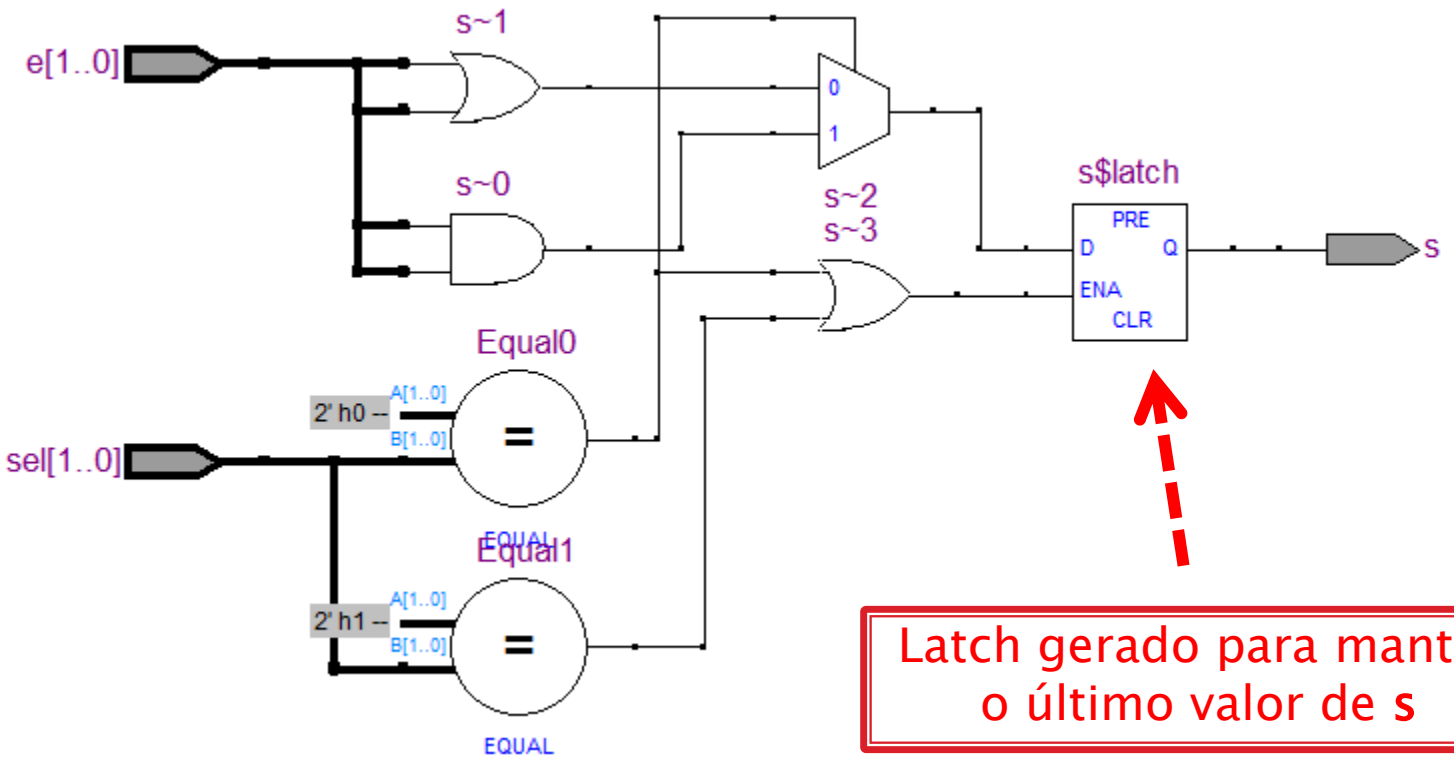
# Comandos em VHDL – Sequenciais:NULL
## Exemplo de Aplicação com IF-THEN-ELSE-END IF

Na descrição a seguir, a saída **s** recebe o resultado de uma entre duas operações lógicas, selecionadas pelo sinal **sel** ("00" indica AND e "01" indica OR). Caso **sel** seja diferente destes valores, a saída não deve ser alterada.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY ex_mux2 IS
    PORT(e, sel : IN  STD_LOGIC_VECTOR (1 DOWNTO 0);
         s      : OUT STD_LOGIC);
END ex_mux2;

ARCHITECTURE a OF ex_mux2 IS
BEGIN
    PROCESS (e, sel)
    BEGIN
        IF    sel = "00" THEN s <= e(0) AND e(1);
        ELSIF sel = "01" THEN s <= e(0) OR e(1);
        ELSE  NULL; -- Não é necessário colocar o ELSE
        END IF;
END PROCESS;
END a;
```

# Comandos em VHDL – Sequenciais:NULL

## Exemplo de Aplicação com IF-THEN-ELSE-END IF




Latch gerado para manter o último valor de **s**

# Atributos

# Atributos – Conceito

Atributos são informações adicionais associadas a objetos (como por exemplo, a sinais) que possibilitam a verificação de transições de sinal e o modelamento de atrasos.

Biblioteca WORK:

| ATRIBUTO | FUNÇÃO |
|---|---|
| `nome_do_sinal'EVENT` | Verdadeiro se ocorreu uma troca de valor do sinal no ciclo corrente de simulação, falso caso contrário. |

# Atributos –Verificação de transição de um sinal

Biblioteca WORK:

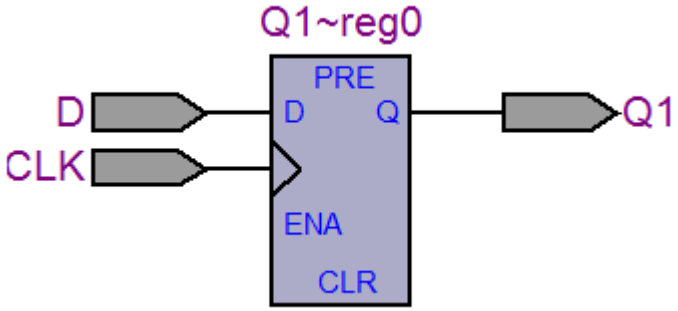
| DESCRIÇÃO VHDL | FUNÇÃO |
|---|---|
| `clk'EVENT AND clk ='1'` | Seleção da transição positiva do sinal clk |
| `clk'EVENT AND clk = '0'` | Seleção da transição negativa do sinal clk |

Biblioteca IEEE pacote 1164:

| DESCRIÇÃO VHDL | FUNÇÃO |
|---|---|
| `RISING_EDGE(clk)` | Seleção da transição positiva do sinal clk |
| `FALLING_EDGE(clk)` | Seleção da transição negativa do sinal clk |

Obs.: As funções RISING_EDGE() e FALLING_EDGE() apenas retornam TRUE se houver uma transição de 0 para 1 ou de 1 para 0, ignorando transições envolvendo outros valores (X, –, H, etc.).

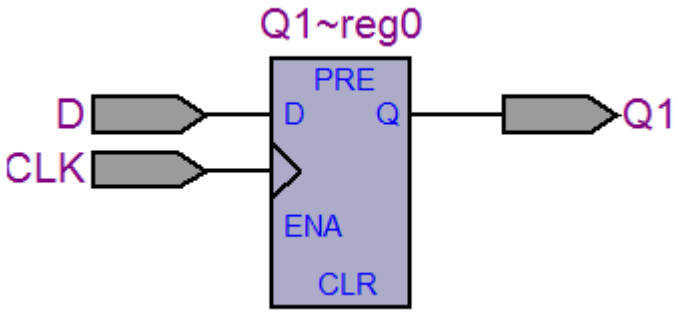
# FF tipo D disparado na borda de subida do clock Usando IF-THEN




```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY FF_tipoD IS
    PORT(D, CLK : IN     STD_LOGIC;
         Q1     : BUFFER STD_LOGIC); -- Saída de
          -- Q1     : OUT STD_LOGIC); opção 2
END FF_tipoD;

ARCHITECTURE a OF FF_tipoD IS
--SIGNAL sinal_Q1: STD_LOGIC; opção 2
BEGIN
    PROCESS(CLK)
    BEGIN
        IF (CLK'EVENT AND CLK = '1') THEN -- Aguarda borda de subida do CLK
            Q1 <= D;
      --sinal_Q1 <=D; opção 2
       -- ELSE Q1 <= Q1; Desnecessário incluir
        END IF;
    END PROCESS;

      --Q1 <= sinal_Q1; opção 2
END a;
```

Obs.: Todos os sinais assíncronos entram na lista de sensibilidade do PROCESS

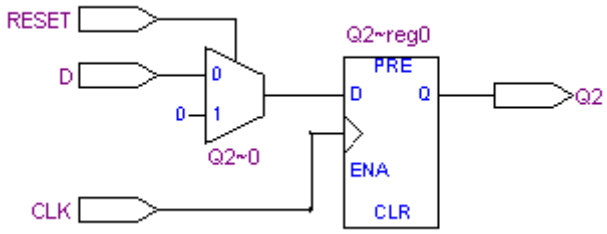
# FF tipo D disparado na borda de subida do clock Usando WAIT UNTIL



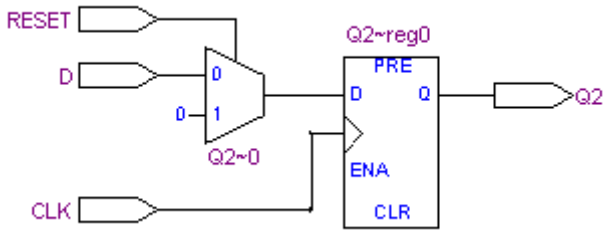

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY FF_tipoD IS
    PORT(D, CLK, : IN     STD_LOGIC;
         Q1      : BUFFER STD_LOGIC);
END FF_tipoD;

ARCHITECTURE a OF FF_tipoD IS
BEGIN
    -- Se WAIT é usado, não há lista de sensibilidade no PROCESS
    PROCESS
    BEGIN
        -- Aguarda borda de subida do clk
        WAIT UNTIL (CLK'EVENT AND CLK = '1');
            Q1 <= D;
    END PROCESS;
END a;
```

# FF tipo D disparado na borda de subida do clock e RESET síncrono Usando IF-THEN-ELSE

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY FF_tipoD_Rsinc IS
    PORT(D, CLK, RESET : IN     STD_LOGIC;
         Q2            : BUFFER STD_LOGIC);
END FF_tipoD_Rsinc;

ARCHITECTURE a OF FF_tipoD_Rsinc IS
BEGIN
    PROCESS(CLK)
    BEGIN
        -- Aguarda borda de subida do clk
        IF (CLK'EVENT AND CLK = '1') THEN
            IF RESET = '1' THEN
                Q2 <= '0';
            ELSE
                Q2 <= D;
            END IF;
        END IF;
    END PROCESS;
END a;
```



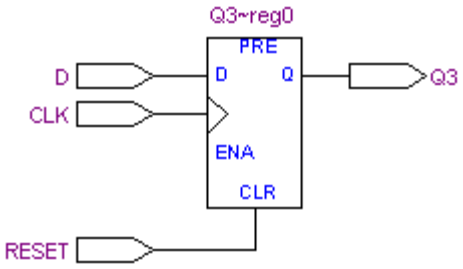
# FF tipo D disparado na borda de subida do clock e RESET síncrono Usando WAIT UNTIL




```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY FF_tipoD_Rsinc IS
    PORT(D, CLK, RESET : IN     STD_LOGIC;
         Q2            : BUFFER STD_LOGIC);
END FF_tipoD_Rsinc;

ARCHITECTURE a OF FF_tipoD_Rsinc IS
BEGIN
    PROCESS -- Se WAIT é usado não é usada lista de sensibilidade
    BEGIN
        -- Aguarda borda de subida do clk
        WAIT UNTIL (CLK'EVENT AND CLK = '1');
            IF RESET = '1' THEN
                Q2 <= '0';
            ELSE
                Q2 <= D;
            END IF;
   END PROCESS;
END a;
```

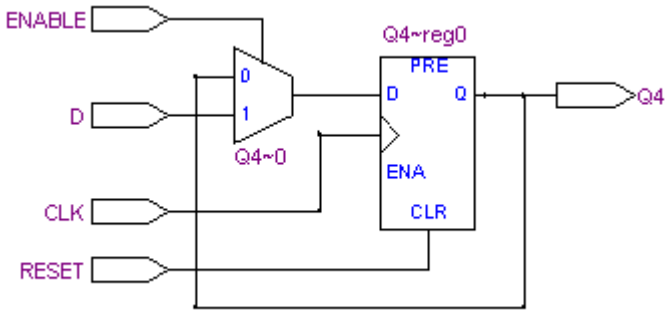
# FF tipo D disparado na borda de subida do clock RESET Assíncrono Usando IF-ELSIF-ELSE

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY FF_tipoD_Rassinc IS
    PORT(D, CLK, RESET : IN     STD_LOGIC;
         Q3            : BUFFER STD_LOGIC);
END FF_tipoD_Rassinc;

ARCHITECTURE a OF FF_tipoD_Rassinc IS
BEGIN
    PROCESS(RESET, CLK)
    BEGIN
        IF RESET = '1' THEN
            Q3 <= '0';
        ELSIF (CLK'EVENT AND CLK='1‘) THEN -- Aguarda borda de
                                           -- subida do clk
            Q3 <= D;
        ELSE
            Q3 <= Q3;
        END IF;
    END PROCESS;
END a;
```



# FF tipo D disparado na borda de subida do clock com RESET Assíncronos e ENABLE Usando IF-ELSIF-ELSE

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY FF_tipoD_ERassinc IS
    PORT(D, CLK, RESET, ENABLE : IN     STD_LOGIC;
         Q4                    : BUFFER STD_LOGIC);
END FF_tipoD_ERassinc;

ARCHITECTURE a OF FF_tipoD_ERassinc IS
BEGIN
    PROCESS (CLK, RESET)
    BEGIN
        IF RESET = '1' THEN
            Q4 <= '0';
        ELSIF (CLK'EVENT AND CLK = '1') THEN -- Aguarda borda
                                             -- de subida do CLK
            IF ENABLE ='1' THEN
                Q4 <= D;
            ELSE
                Q4 <= Q4;
            END IF;
        END IF;
    END PROCESS;
END a;
```



# Prática nº6
# Descrição de Flip-Flops

Implementar Flip-Flops tipo T sensíveis à borda de subida do clock, utilizando o atributo ‘EVENT e a biblioteca IEEE pacote STD_LOGIC_1164.
Simular e gerar o RTL

Ver roteiro no moodle

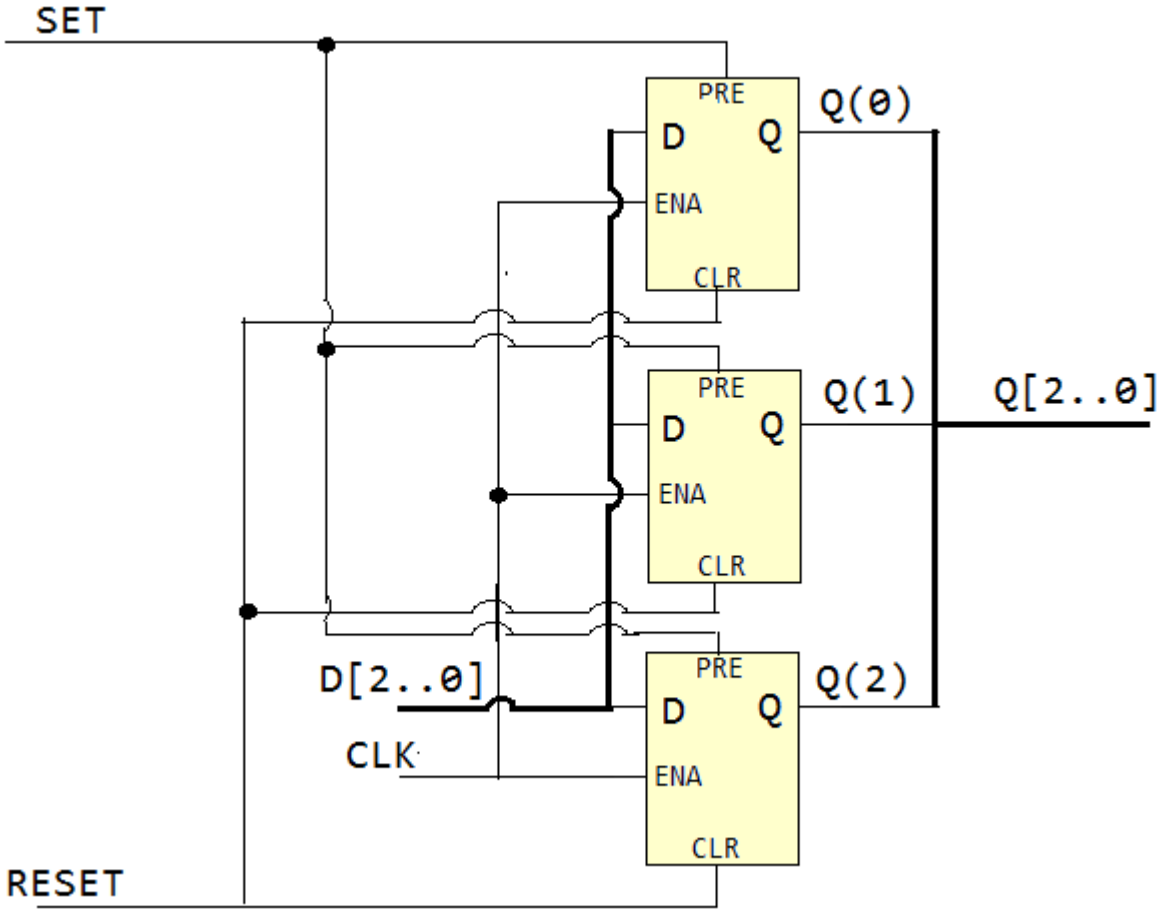
# TIPO INTEGER

# 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

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY FF3_tipoD_nivel IS
    PORT(CLK, RST, SET : IN  STD_LOGIC;
         D             : IN  STD_LOGIC_VECTOR(2 DOWNTO 0);
         Q             : OUT STD_LOGIC_VECTOR(2 DOWNTO 0));
END FF3_tipoD_nivel;

ARCHITECTURE a OF FF3_tipoD_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

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY FF3_tipoD_nivel IS
    PORT(CLK, RST, SET : IN  STD_LOGIC;
         D             : IN  INTEGER RANGE 0 TO 7;
         Q             : OUT INTEGER RANGE 0 TO 7);
END FF3_tipoD_nivel;

ARCHITECTURE a OF FF3_tipoD_nivel 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;
```

# Cláusula GENERIC

# 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  STD_LOGIC;
         Q : OUT STD_LOGIC_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 (instanciação)
2. Declaração do Componente
3. Declaração da Entidade

# GENERIC – Atribuição de valores

Prioridade de atribuição de valor

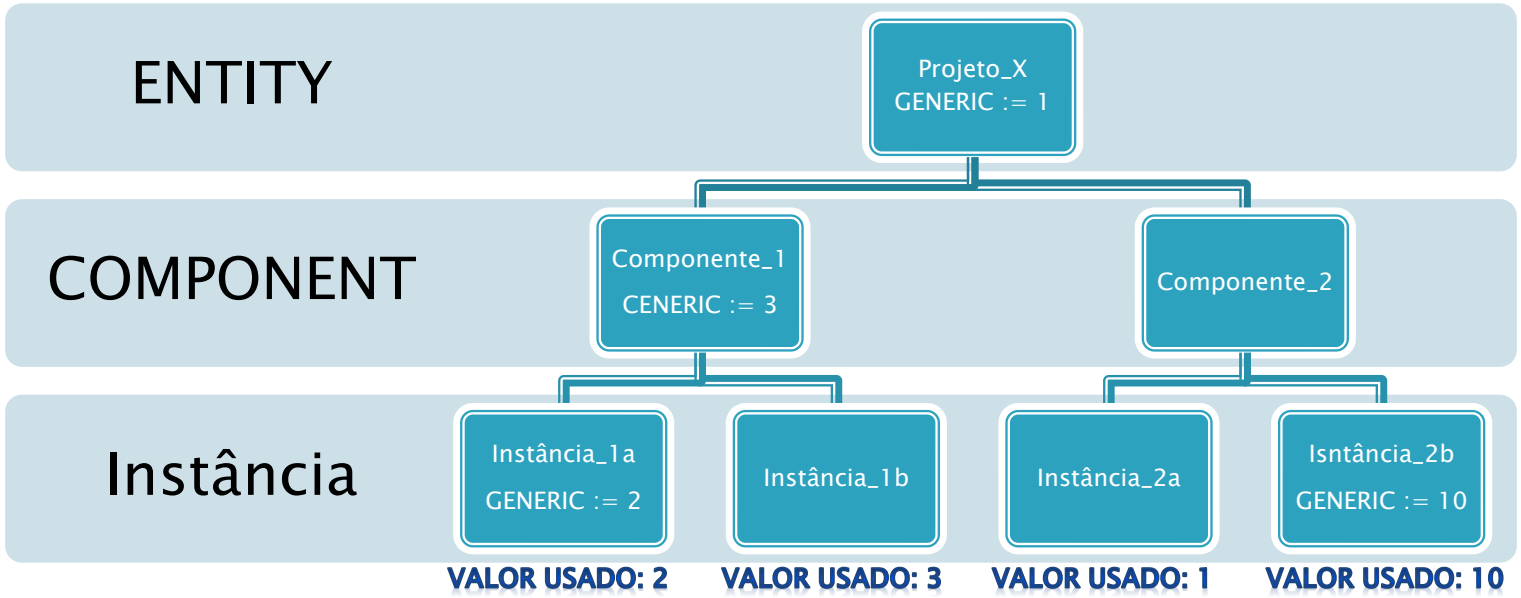
1. Solicitação do Componente
2. Declaração do Componente
3. Declaração da Entidade

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:



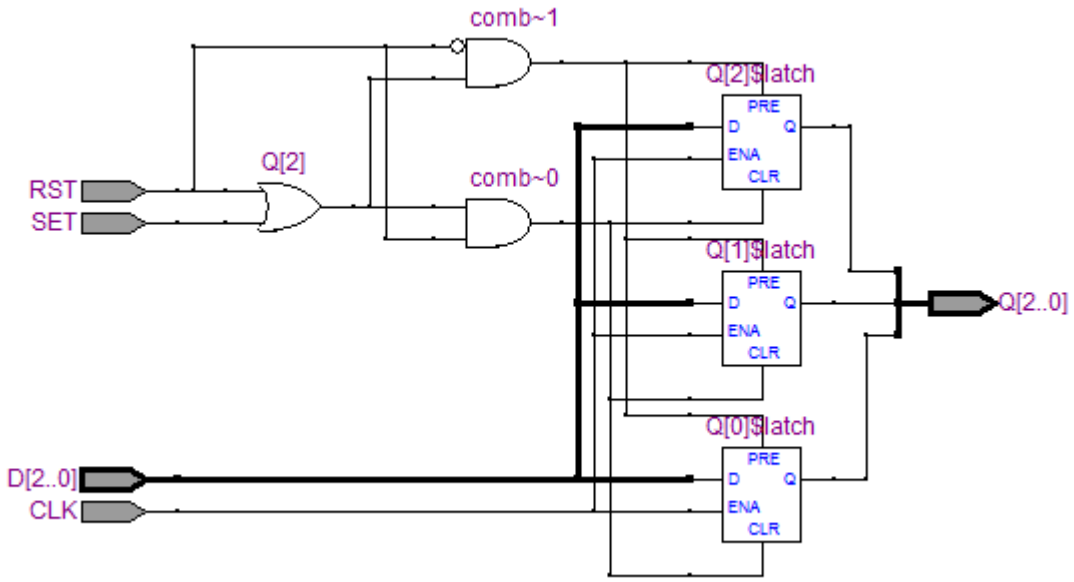
# FFs tipo D em paralelo sensíveis a nível alto do clock RESET e SET Assíncronos Usando IF-ELSIF-ELSE, INTEGER e GENERIC

## N = 3

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY FF_D_inteiro IS
    GENERIC(n : NATURAL := 3);
    PORT(clk, rst, set : IN  STD_LOGIC;
         d             : IN  INTEGER RANGE 0 TO (2**n) - 1;
         q             : OUT INTEGER RANGE 0 TO (2**n) - 1);
END FF3_D_inteiro;

ARCHITECTURE a OF FF_D_inteiro 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



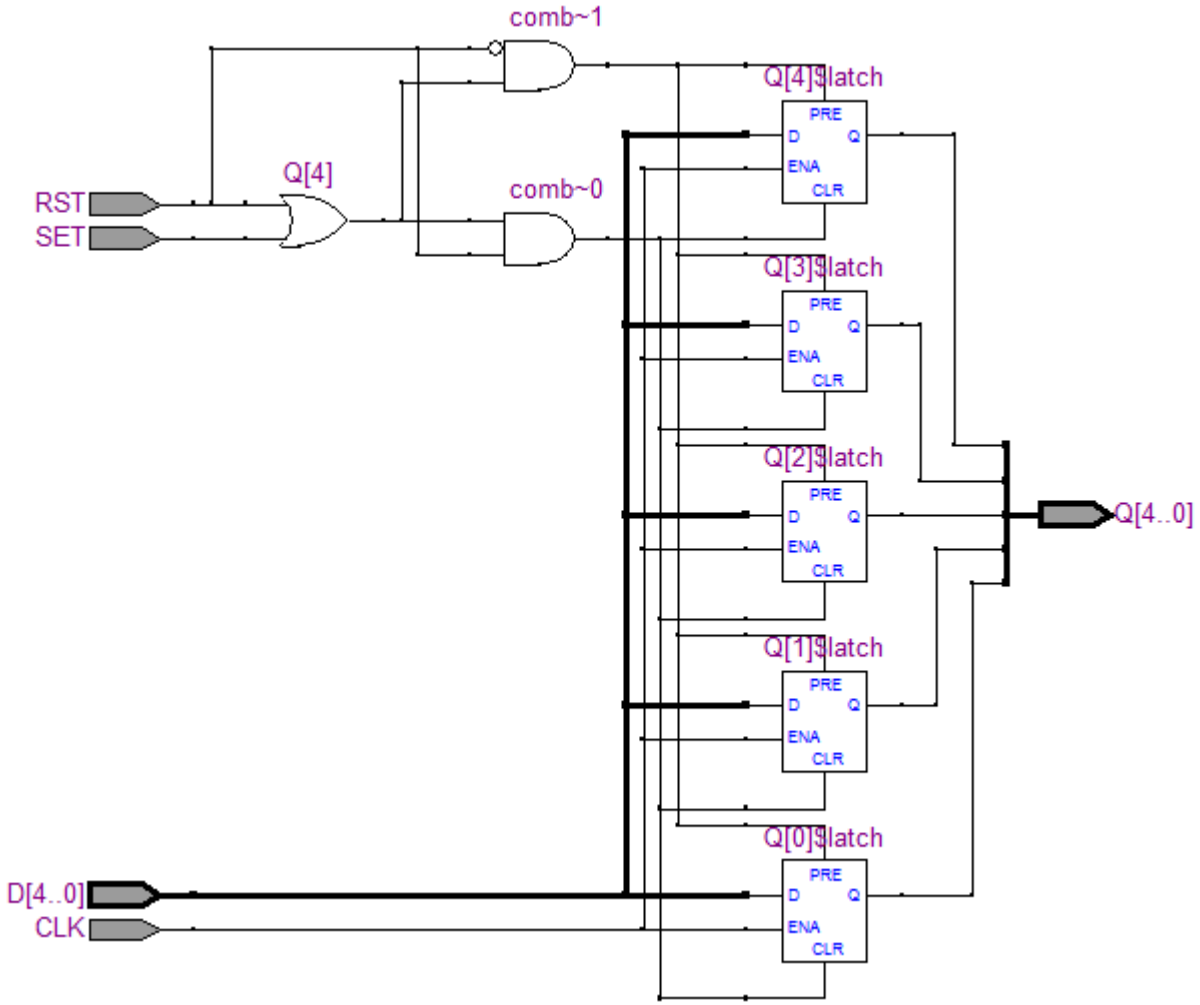
# FFs tipo D em paralelo sensíveis a nível alto do clock RESET e SET Assíncronos Usando IF-ELSIF-ELSE, INTEGER e GENERIC

## N = 5

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY FF_D_inteiro IS
    GENERIC(n : NATURAL := 5);
    PORT(clk, rst, set : IN  STD_LOGIC;
         d             : IN  INTEGER RANGE 0 TO (2**n) - 1;
         q             : OUT INTEGER RANGE 0 TO (2**n) - 1);
END FF3_D_inteiro;

ARCHITECTURE a OF FF_D_inteiro 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 : GENERIC MAP

# 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

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY FF_D_inteiro IS
    GENERIC(n : NATURAL := 3);
    PORT(clk, rst, set : IN  STD_LOGIC;
         d             : IN  INTEGER RANGE 0 TO (2**n) - 1;
         q             : OUT INTEGER RANGE 0 TO (2**n) - 1);
END FF3_D_inteiro;

ARCHITECTURE a OF FF_D_inteiro 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

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY exemplo IS
    PORT(CLK: IN STD_LOGIC:
          IN  : IN  INTEGER RANGE 0 TO 255;
         OUT : OUT INTEGER RANGE 0 TO 255;
END exemplo;

ARCHITECTURE a OF exemplo IS
    COMPONENT FF_D_inteiro IS -- Entidade "FF_D_inteiro" 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  STD_LOGIC;
             d             : IN  INTEGER RANGE 0 TO (2**n) - 1;
             q             : OUT INTEGER RANGE 0 TO (2**n) - 1);
    END COMPONENT;

    SIGNAL GROUND, CLK : STD_LOGIC := '0';

BEGIN
    -- Atribui o valor 8 ao genérico n, substituindo o valor (1) atribuído
    -- acima na declaração do componente
    X1 : FF_D_inteiro GENERIC MAP(8) PORT MAP(CLK,GROUND,GROUND,IN,OUT);
    CLK <= NOT CLK;

END a;
```

# Prática nº7
# Contador Binário – Usando Template do QUARTUS II
# Usar clausula GENERIC

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

ENTITY __entity_name IS

        PORT
        (
                __data_input_name   : IN        INTEGER RANGE 0 TO __count_value;
                __clk_input_name    : IN        STD_LOGIC;
                __clrn_input_name   : IN        STD_LOGIC;
                __ena_input_name    : IN        STD_LOGIC;
                __ld_input_name                 : IN        STD_LOGIC;
                __count_output_name: OUT INTEGER RANGE 0 TO __count_value
        );

END __entity_name;
```

# CONVERSÃO ENTRE TIPOS:

| DESCRIÇÃO VHDL | *FUNÇÃO* | Biblioteca |
|---|---|---|
| Inteiro <= conv_integer (vetor); | **Converte um vetor em inteiro** | std_logic_arith. |
| Vetor <= conv_std_logic_vector (inteiro, nºbits); | **Converte um inteiro em vetor** | std_logic_arith. |
| Vetor <= std_logic_vector (to_unsigned(inteiro, nºbits)); | **Converte um inteiro em vetor** | numeric_std |
| Inteiro<=to_integer( unsigned(vetor_std_logic_vector) ) | **Converte um vetor em inteiro** | numeric_std |
| Inteiro<=to_integer( signed(vetor_std_logic_vector) ) | **Converte um vetor em inteiro** | numeric_std |
| -- Dentro de um process<br><variável_tipo_std_ulogic> := To_StdUlogic(<variável_tipo_bit>); | **Converte bit em std_logic** | std_logic_1164. |
| -- Dentro de um process<br><variável_tipo_std_logic_vector>:=<br>to_stdLogicVector(<variável_tipo_bit_vector>); | **Converte std_logic_vector em bit_vector** | std_logic_1164 |

**Observação:** Não é permitida a transferência de valores entre objetos de tipos diferentes.

# FIM