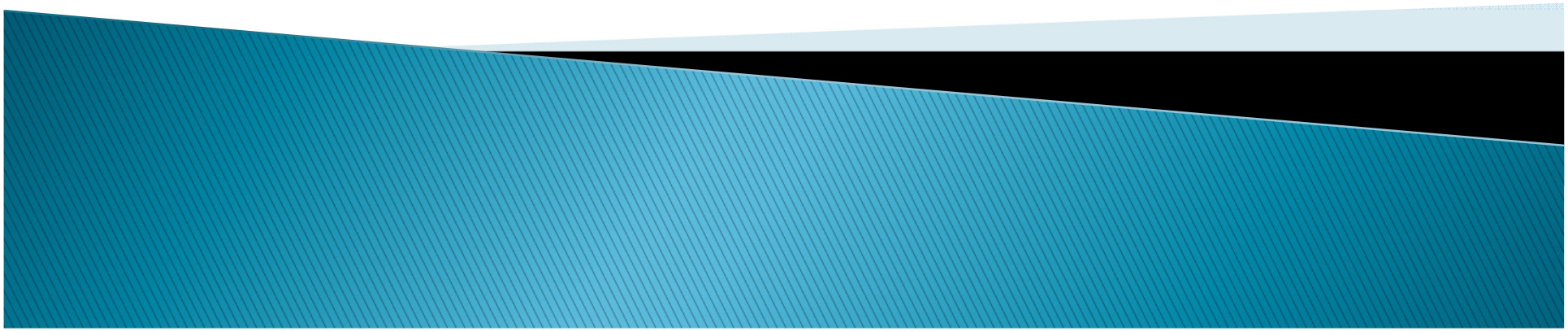
Laboratório de Eletrônica Digital
SEL-EESC-USP

EESC · USP

Introdução a VHDL

Aula 5

Professora Luiza Maria Romeiro Codá

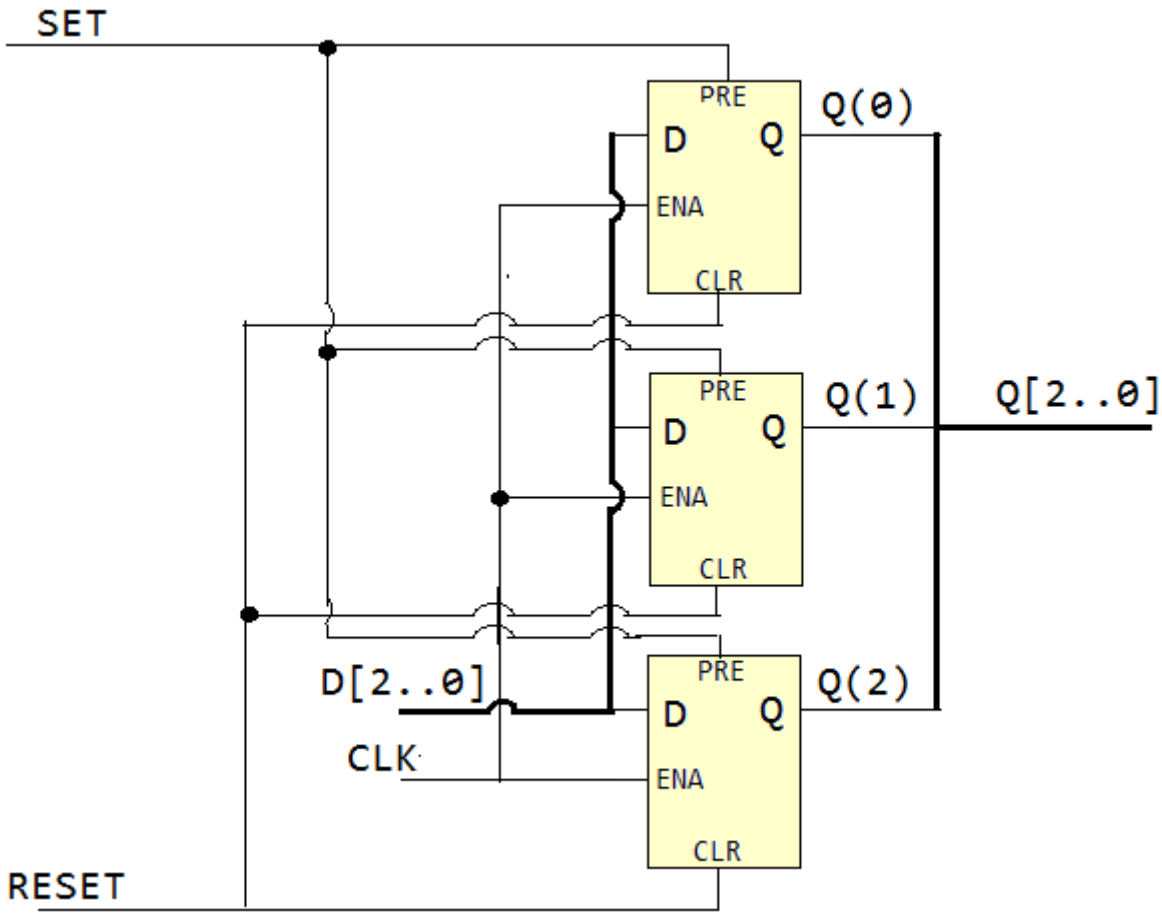


TIPO INTEGER

3 FFs tipo D em paralelo sensíveis a nível alto do clock RESET e SET Assíncronos Usando 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;
```

3 FFs tipo D em paralelo sensíveis a nível alto do clock 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;  -- Y 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.

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

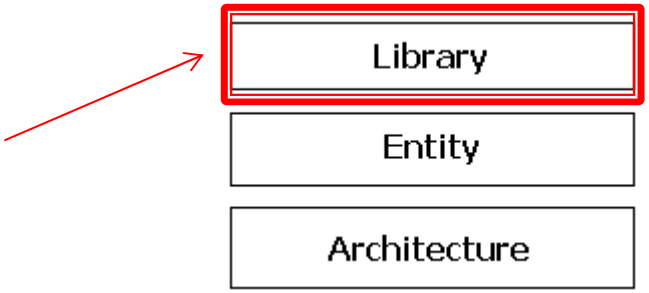
```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

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;
```

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 <variável_tipo_std_ulogic> := To_StdUlogic(<variável_tipo_bit>);	**Converte bit em std_logic**	std_logic_1164.
-- Dentro de um process <variável_tipo_std_logic_vector>:= 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.

LIBRARY



Pacotes mais utilizados da biblioteca IEEE:

LIBRARY IEEE;	Usada para incluir a biblioteca IEEE;
USE IEEE.std_logic_1164.ALL;	Define o tipo STD_LOGIC (VECTOR), usado em descrições mais fiés a um circuito real.
USE IEEE.std_logic_arith.ALL;	Define os tipos SIGNED e UNSIGNED.
USE IEEE.std_logic_signed.ALL;	Permite tratar sinais do tipo STD_LOGIC_VECTOR como inteiros com sinal (define funções aritméticas, de comparação, etc).
USE IEEE.std_logic_unsigned.ALL;	Permite tratar sinais do tipo STD_LOGIC_VECTOR como inteiros sem sinal (define funções aritméticas, de comparação, etc).
USE IEEE.numeric_std.ALL;	Define os tipos UNSIGNED e SIGNED como matriz de std_logic

Cláusula GENERIC

- Declarado na ENTITY para definir uma constante;
- simimilar 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 – Atribuição de valores

O valor de um Genérico pode ser especificado 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

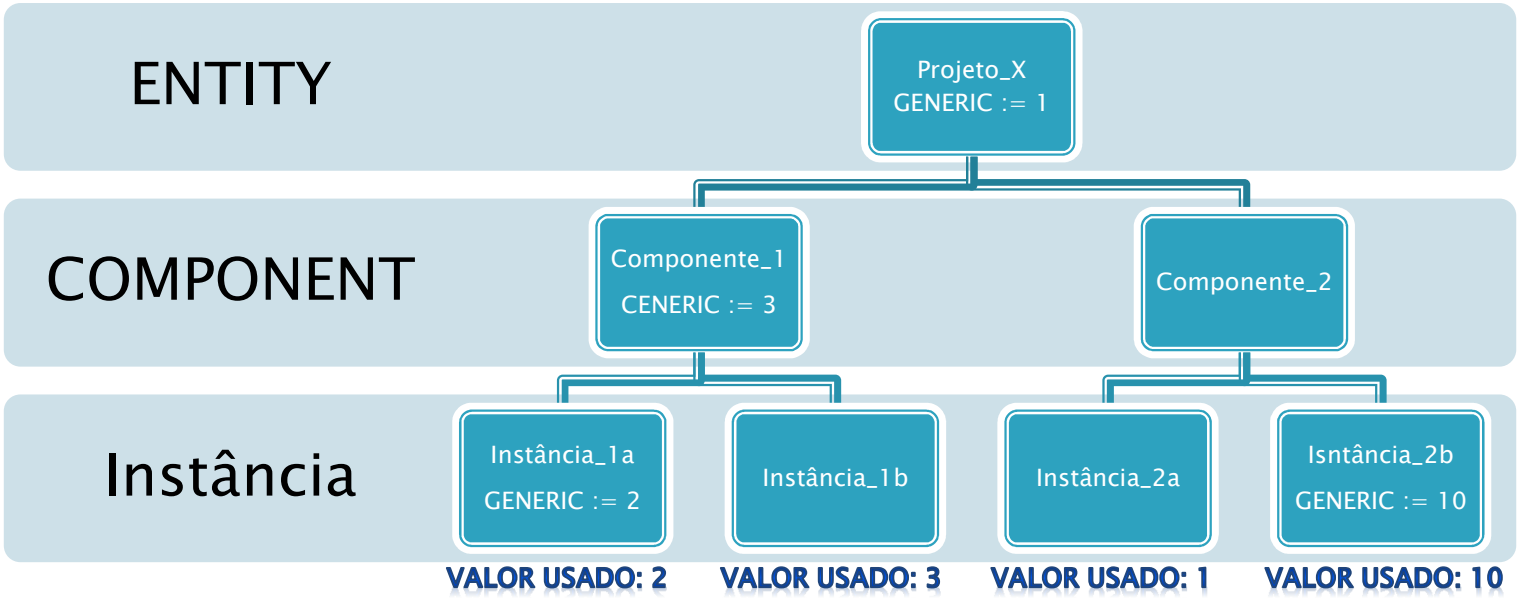
1. Solicitação do Componente
2. Declaração do Componente
3. Declaração da Entidade

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

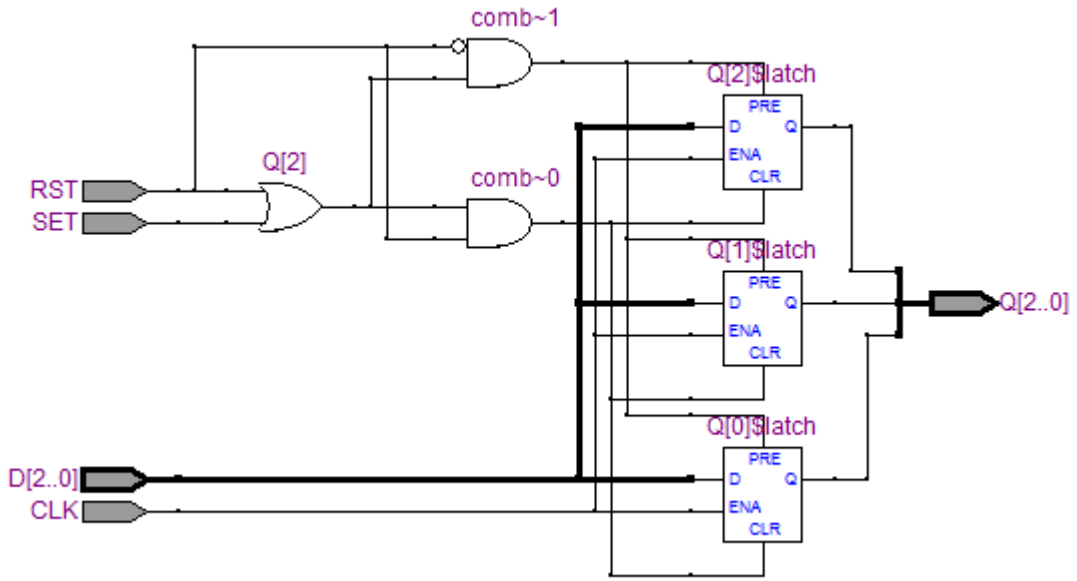
Ilustração do esquema de prioridades:



N = 3 FFs tipo D em paralelo sensíveis a nível alto do clock RESET e SET Assíncronos Usando IF-ELSIF-ELSE, INTEGER e GENERIC

```
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; -- (2³ - 1) = 111
        ELSIF (clk = '1') THEN
            q <= d;
        END IF;
    END PROCESS;
END a;
```

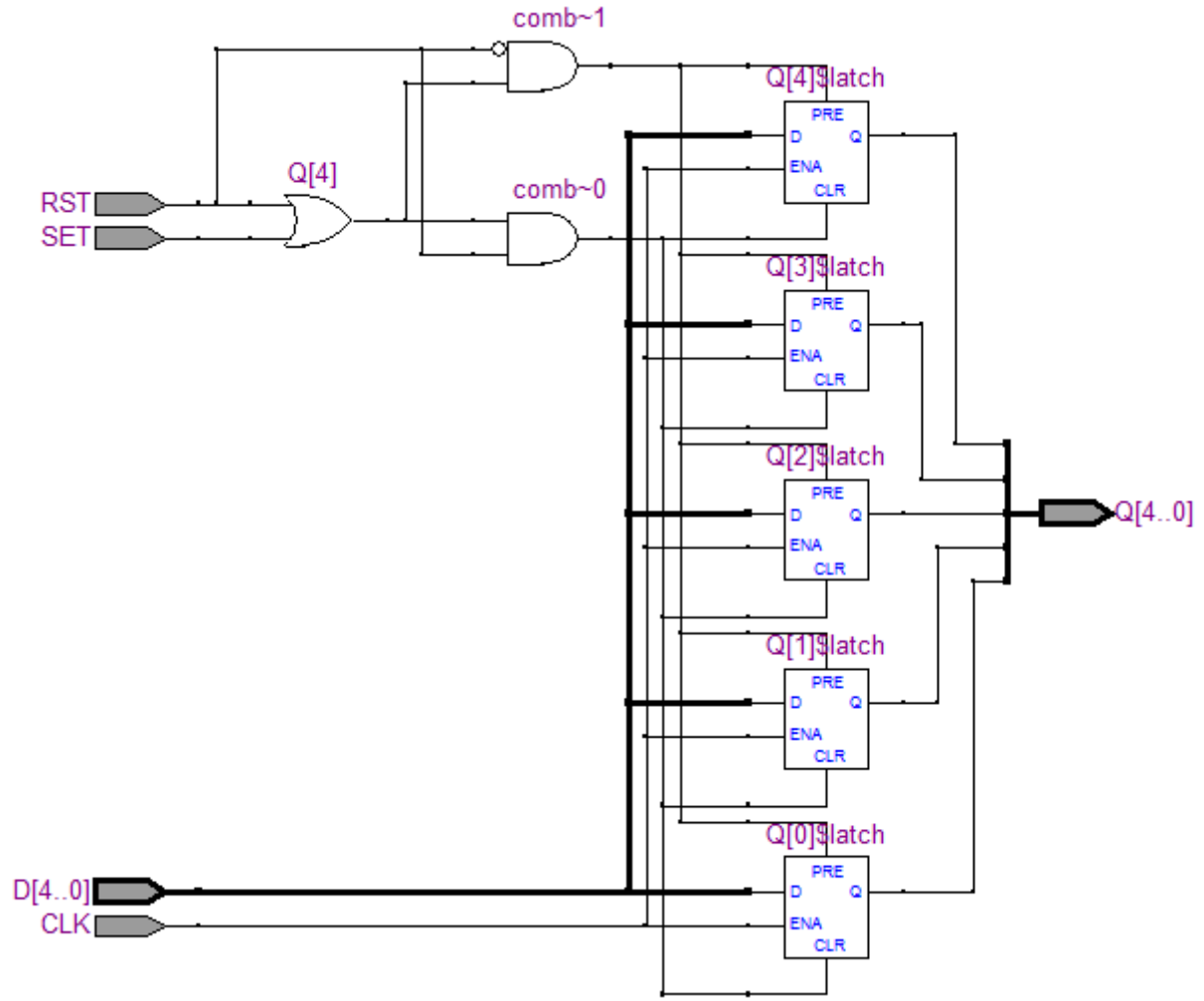
N = 3 FFs tipo D em paralelo sensíveis a nível alto do clock
RESET e SET Assíncronos
Usando IF-ELSIF-ELSE, INTEGER e GENERIC



N = 5 FFs tipo D em paralelo sensíveis a nível alto do clock RESET e SET Assíncronos Usando IF-ELSIF-ELSE, INTEGER e GENERIC

```
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;
```

N = 5 FFs tipo D em paralelo sensíveis a nível alto do clock
RESET e SET Assíncronos
Usando IF-ELSIF-ELSE, INTEGER e GENERIC



Prática nº11
Contador Binário – Usando Template do QUARTUS II

-- Contador com entrada abilitadora do clock (ena), entrada que zera as saídas (clrn) carregável, habilitável e entrada para carregar o valor das saídas(data e ld)

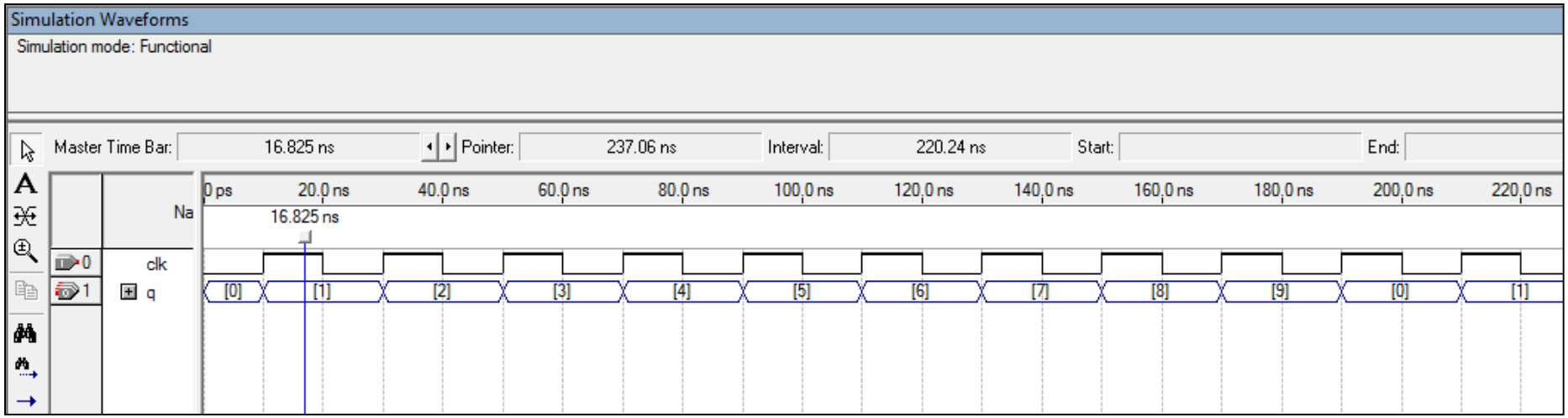
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

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;
```

Resolução da Prática 11 usando GENERIC

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY cont10simples IS

GENERIC(max_count : NATURAL := 9);

PORT(clk : IN  STD_LOGIC;
     q   : OUT INTEGER RANGE 0 TO max_count);
END cont10simples;

ARCHITECTURE a OF cont10simples IS
    SIGNAL sinal_q : INTEGER RANGE 0 TO max_count;
BEGIN
    PROCESS(clk)
    BEGIN
        IF (clk 'EVENT AND clk = '1') THEN -- Verificação da borda de subida de clk
            IF sinal_q >= max_count THEN
               sinal_q <= 0; -- O zero não está entre apóstrofe pois é um INTEGER
            ELSE
               sinal_q <= sinal_q + 1;
            END IF;
        ELSE
            sinal_q <= sinal_q;
        END IF;
    END PROCESS;
    q  <= sinal_q; -- Saída q recebe valor do sinal_q
END a;
```

Resolução da Prática 11 usando GENERIC

Resultado da simulação sem considerar o tempo de atraso do dispositivo:



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);
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>)
                                PORT MAP(...);
-- Forma alternativa
<rótulo> : <nome_do_componente> GENERIC MAP(<generico_y> => <Valor_y>)
                                PORT MAP(...);
-- Sem alterar nenhum valor
<rótulo> : <nome_do_componente> PORT MAP(...);
```

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

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

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

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

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

-- Sem alterar nenhum Generic
X4 : Circuito PORT MAP(...);
```

FFs tipo D em paralelo sensíveis a nível alto do clock
RESET e SET Assíncronos
Usando INTEGER e GENERIC

```
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; -- (2³ - 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
generic
    PORT(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 FF_D_inteiro;

    SIGNAL GROUND, CLK : BIT := '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;
```