Sistemas Reconfiguráveis
Eng. de Computação

Profs. Francisco Garcia e Antônio Hamilton Magalhães

Aula 9 – Código sequencial em VHDL

Exemplos: Registradores de deslocamento e contadores

rev. B – 03/10/23

Código sequencial de VHDL

Na aula passada começamos a estudar o **código sequencial** de VHDL, e vimos duas de suas principais construções:

- Construção **IF/THEN/ELSE**:

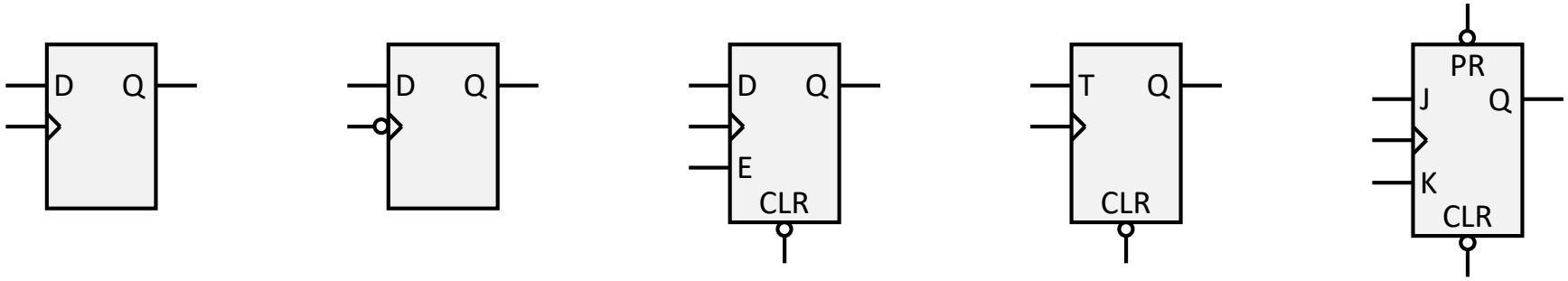
```
[label:] IF condition1 THEN
    expressions;
ELSIF condition2 THEN
    expressions;
ELSE
    expressions;
END IF;
```

- Construção **CASE/WHEN**:

```
[label:] CASE identifier IS
    WHEN value1 =>
        expressions;
    WHEN value2 | value3 =>
        expressions;
    WHEN value4 TO value7 =>
        expressions;
    WHEN OTHERS =>
        expressions;
END CASE;
```

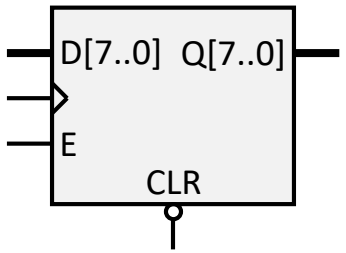
Flip-flops

Com o uso direto dos operadores, além das construções IF/THEN/ELSE e CASE/WHEN do **código sequencial** de VHDL, foram descritos alguns tipos de flip-flops:

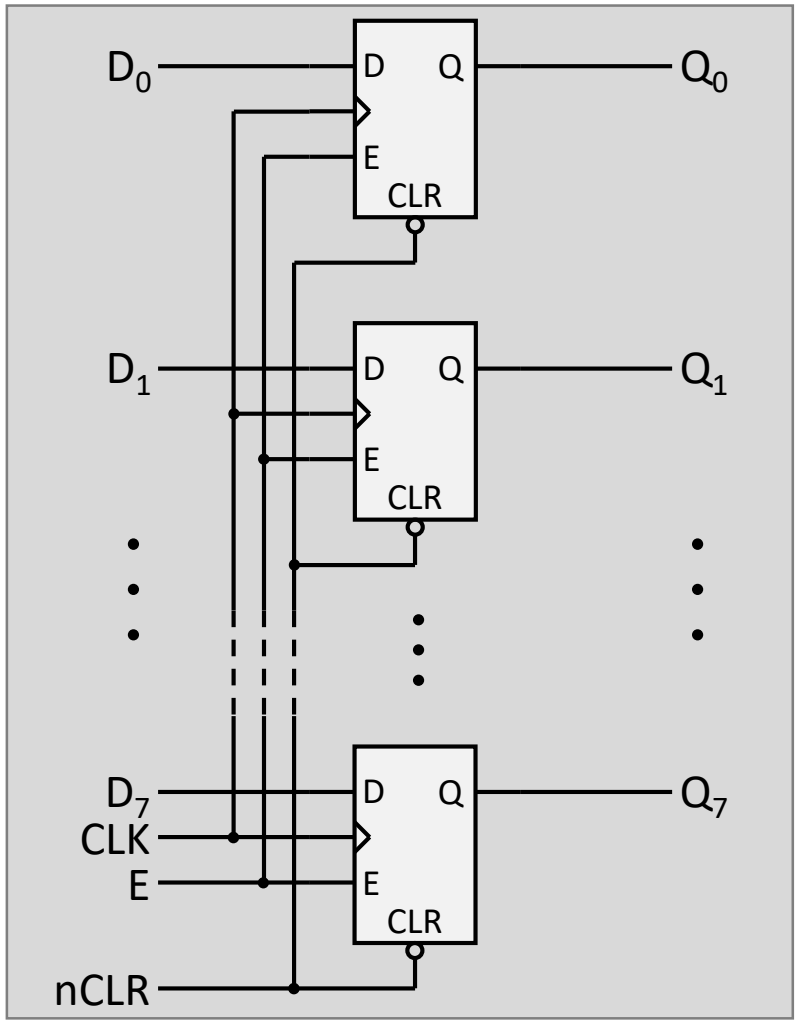


Flip-flops

Vimos também como descrever um registrador de vários bits:

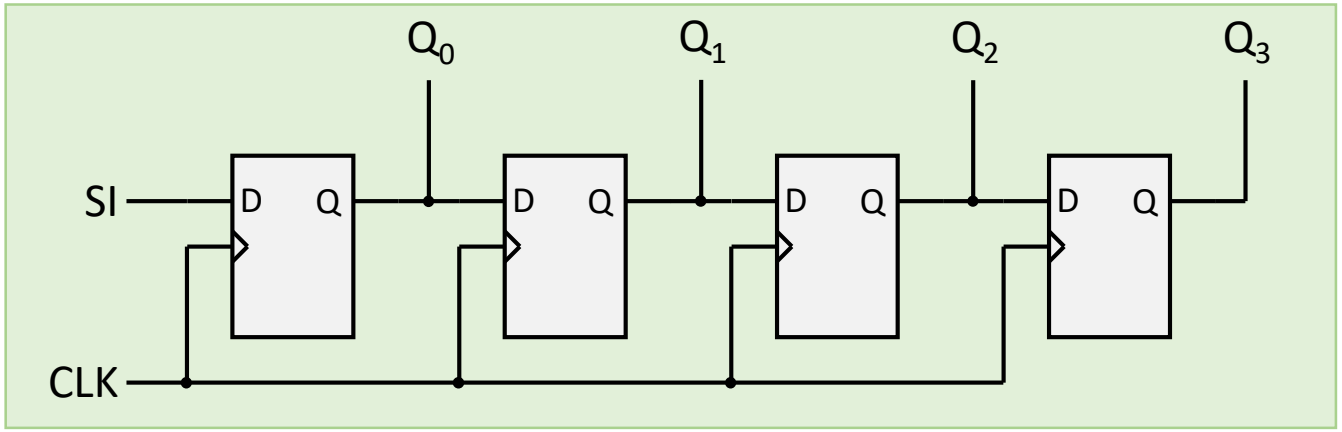


Na aula de hoje vamos estudar mais circuitos usando flip-flops.



Exemplo shf_reg4

Exemplo shf_reg4: **registrador de deslocamento** (*shift register*) de 4 bits, com uma entrada serial e saída paralela. Flip-flops disparados pela borda de subida de *clock*.



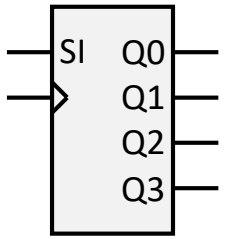
O dado que entra por **SI** é armazenado inicialmente no flip-flop da esquerda.
A cada pulso de **CLK**, o dado se desloca um FF para a direita.

Esse circuito é chamado de **síncrono**, pois todos os flip-flops trabalham com o mesmo *clock*

Esse registrador de deslocamento é chamado de **SIPO** (serial in / parallel out)

Exemplo shf_reg4

Exemplo shf_reg4: **registrador de deslocamento** (*shift register*) de 4 bits, com uma entrada serial e saída paralela. Flip-flops disparados pela borda de subida de *clock*.

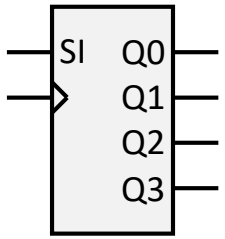



```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY shf_reg4 IS
    PORT (
        clk, si : IN STD_LOGIC;
        q : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)
    );
END ENTITY;
.......
```

Continua na próxima página

Exemplo shf_reg4

Exemplo shf_reg4: **registrador de deslocamento** (*shift register*) de 4 bits, com uma entrada serial e saída paralela. Flip-flops disparados pela borda de subida de *clock*.




```
ARCHITECTURE arch1 OF shf_reg4 IS
BEGIN
    PROCESS(clk)
        VARIABLE reg : STD_LOGIC_VECTOR(3 DOWNTO 0);
    BEGIN
        IF RISING_EDGE(clk) THEN
            reg(3 DOWNTO 1) := reg(2 DOWNTO 0);
            reg(0) := si;
        END IF;
        q <= reg;
    END PROCESS;
END arch1;
```

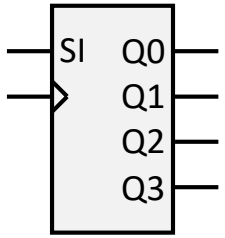
A ordem entre essas expressões é importante!

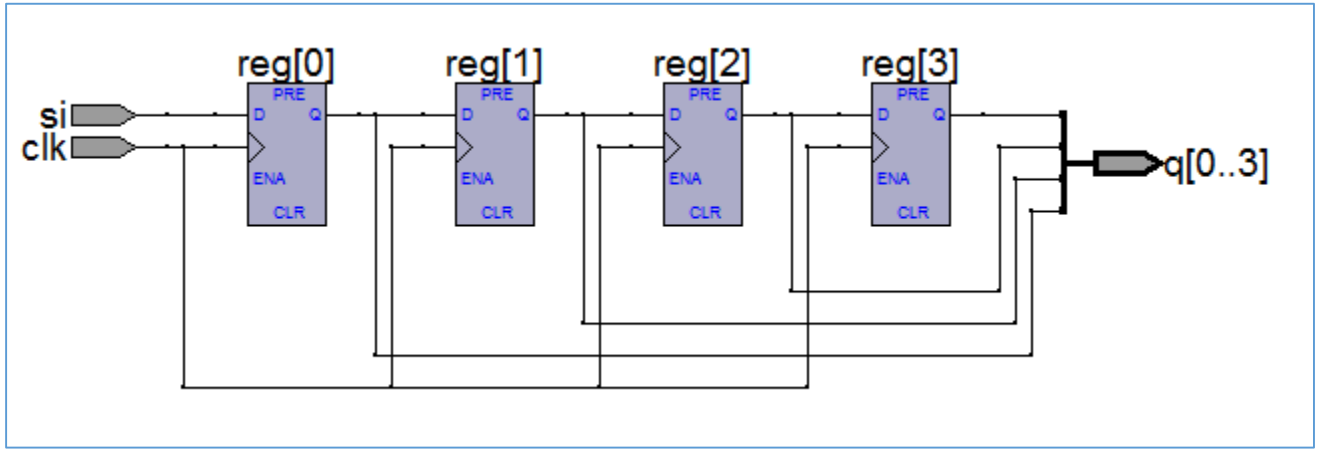
Outra opção:
`reg := reg(2 DOWNTO 0) & si`

VARIABLEs são atualizadas imediatamente dentro de um PROCESS

Exemplo shf_reg4

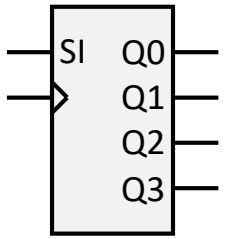
Exemplo shf_reg4: **registrador de deslocamento** (*shift register*) de 4 bits, com uma entrada serial e saída paralela. Flip-flops disparados pela borda de subida de *clock*.

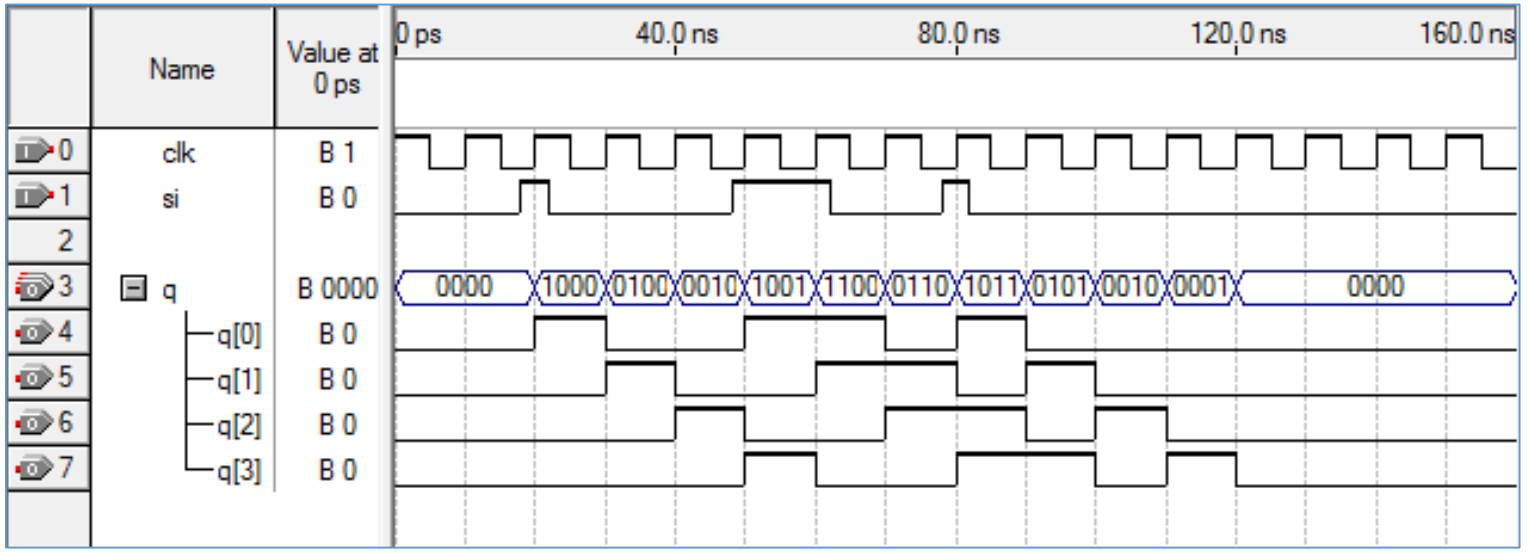




Exemplo shf_reg4

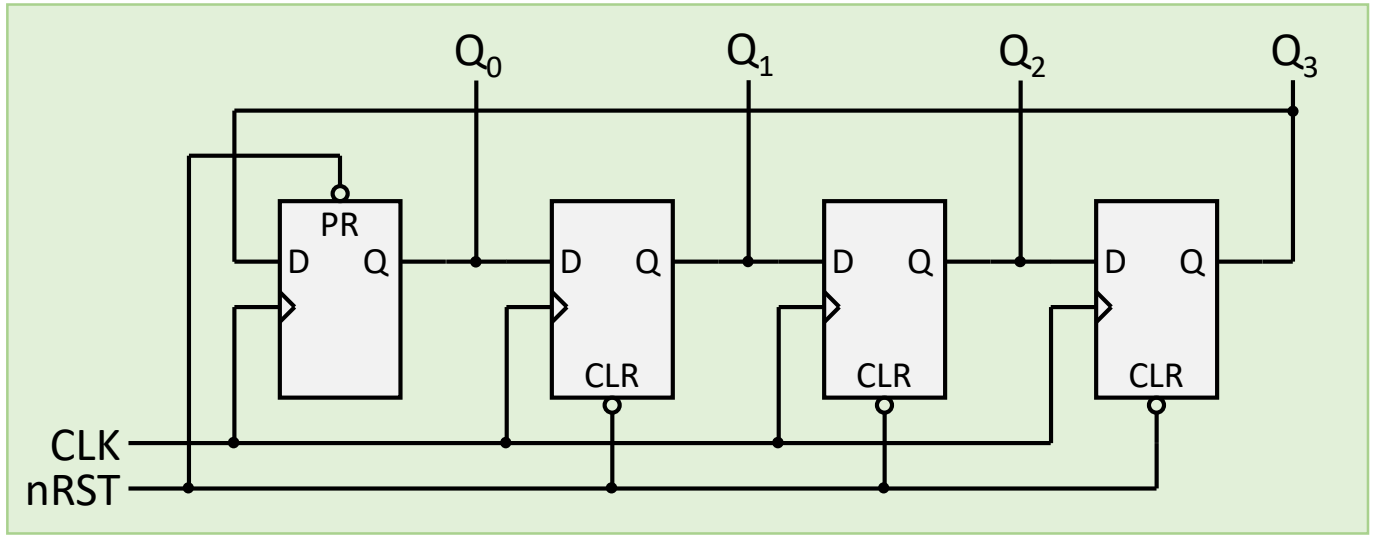
Exemplo shf_reg4: **registrador de deslocamento** (*shift register*) de 4 bits, com uma entrada serial e saída paralela. Flip-flops disparados pela borda de subida de *clock*.





Exemplo rng_cnt4

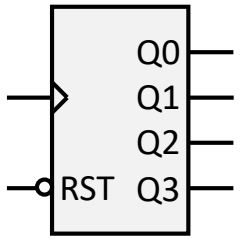
Exemplo rng_cnt4: **contador em anel** (*ring counter*) de 4 bits. Flip-flops disparados pela borda de subida de *clock*.



Após um pulso de nível baixo na entrada assíncrona **nRST**, o FF da esquerda vai para '1', e os demais para '0'. A cada pulso de **CLK**, o dado se desloca um FF para a direita. Quando o '1' chega no último FF da direita, volta para o primeiro da esquerda, em um funcionamento circular.

Exemplo rng_cnt4

Exemplo rng_cnt4: **contador em anel** (*ring counter*) de 4 bits. Flip-flops disparados pela borda de subida de *clock*.

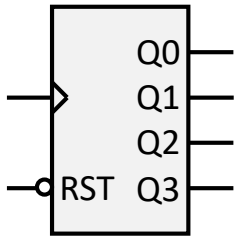



```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY rng_cnt4 IS
    PORT (
        clk, nrst : IN STD_LOGIC;
        q : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)
    );
END ENTITY;
.......
```

Continua na próxima página

Exemplo rng_cnt4

Exemplo rng_cnt4: **contador em anel** (*ring counter*) de 4 bits. Flip-flops disparados pela borda de subida de *clock*.

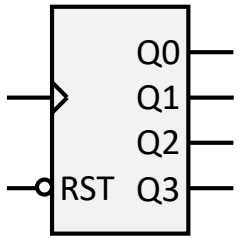


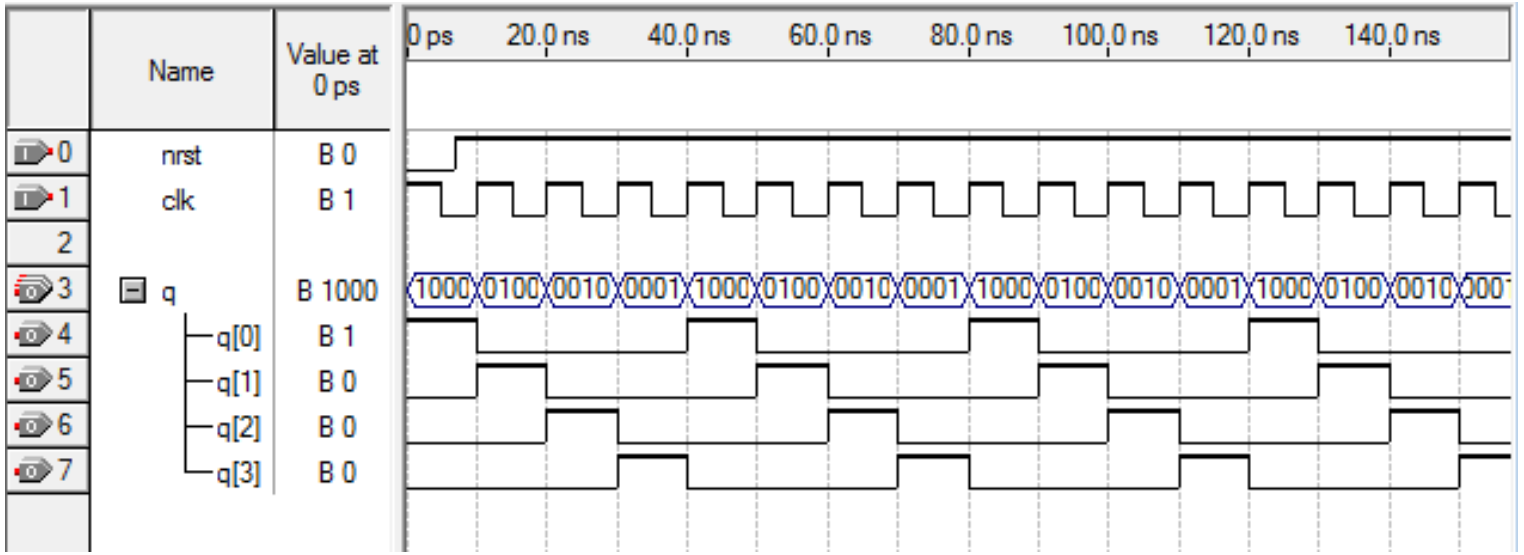

```
ARCHITECTURE arch1 OF rng_cnt4 IS
BEGIN
    PROCESS(clk, nrst)
        VARIABLE reg : STD_LOGIC_VECTOR(3 DOWNTO 0);
    BEGIN
        IF nrst = '0' THEN
            reg := "0001";
        ELSIF RISING_EDGE(clk) THEN
            reg := reg(2 DOWNTO 0) & reg(3);
        END IF;
        q <= reg;
    END PROCESS;
END arch1;
```

Embora no desenho o FF 0 tenha sido representado à esquerda, no vetor **reg** ele está à direita

Exemplo rng_cnt4

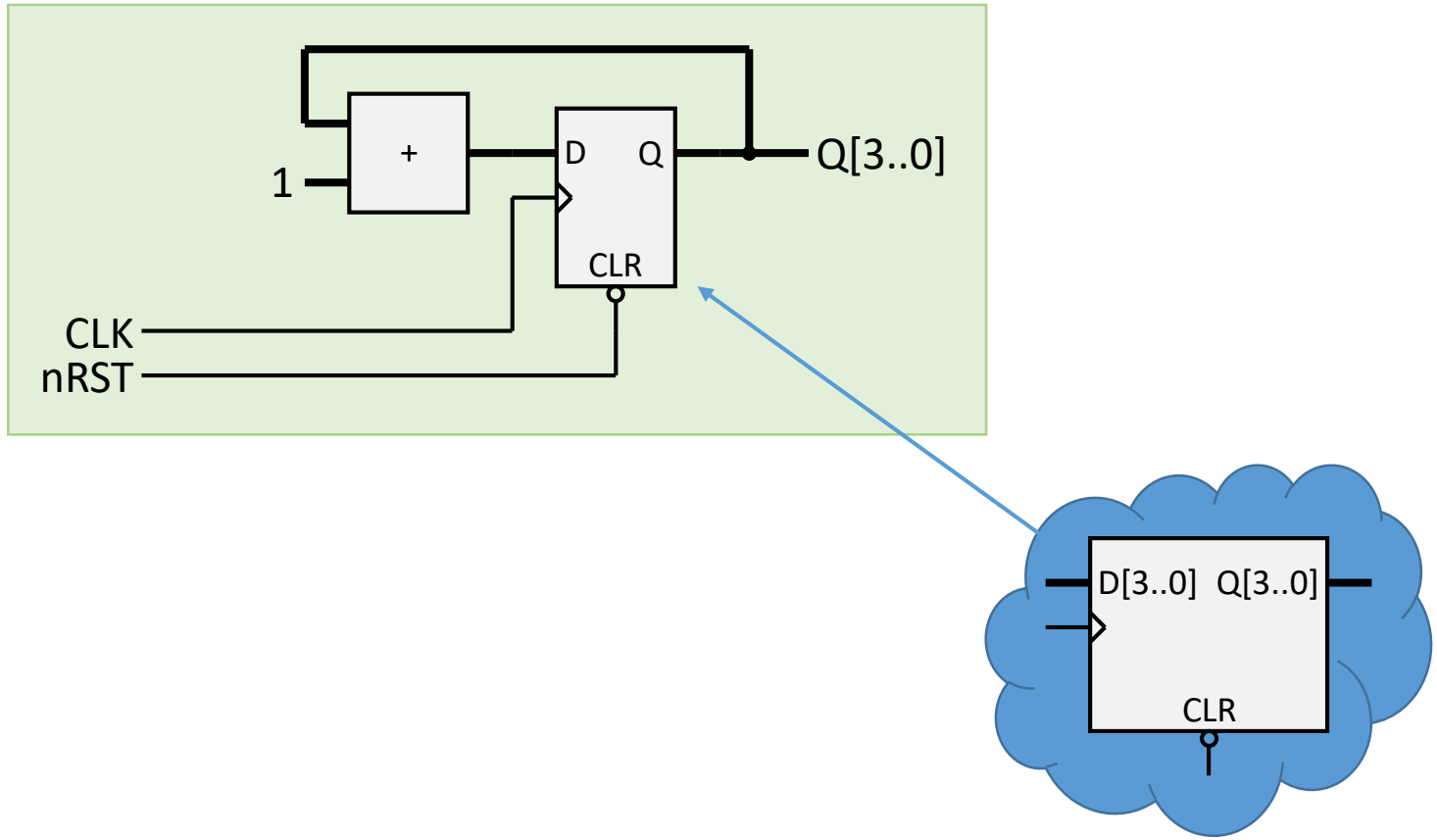
Exemplo rng_cnt4: **contador em anel** (*ring counter*) de 4 bits. Flip-flops disparados pela borda de subida de *clock*.





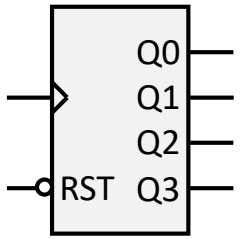
Exemplo cnt_mod16

Exemplo cnt_mod16: **contador módulo 16** com reset assíncrono ativo em nível baixo. Flip-flops disparados pela borda de subida de *clock.*



Exemplo cnt_mod16

Exemplo cnt_mod16: **contador módulo 16** com reset assíncrono ativo em nível baixo. Flip-flops disparados pela borda de subida de *clock*.



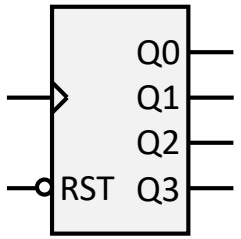

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY cnt_mod16 IS
    PORT (
        clk, nrst : IN STD_LOGIC;
        q : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)
    );
END ENTITY;
.......
```

Pacote necessário para convesão de tipo.

Continua na próxima página

Exemplo cnt_mod16

Exemplo cnt_mod16: **contador módulo 16** com reset assíncrono ativo em nível baixo. Flip-flops disparados pela borda de subida de *clock*.

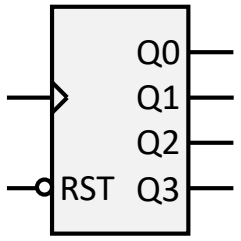


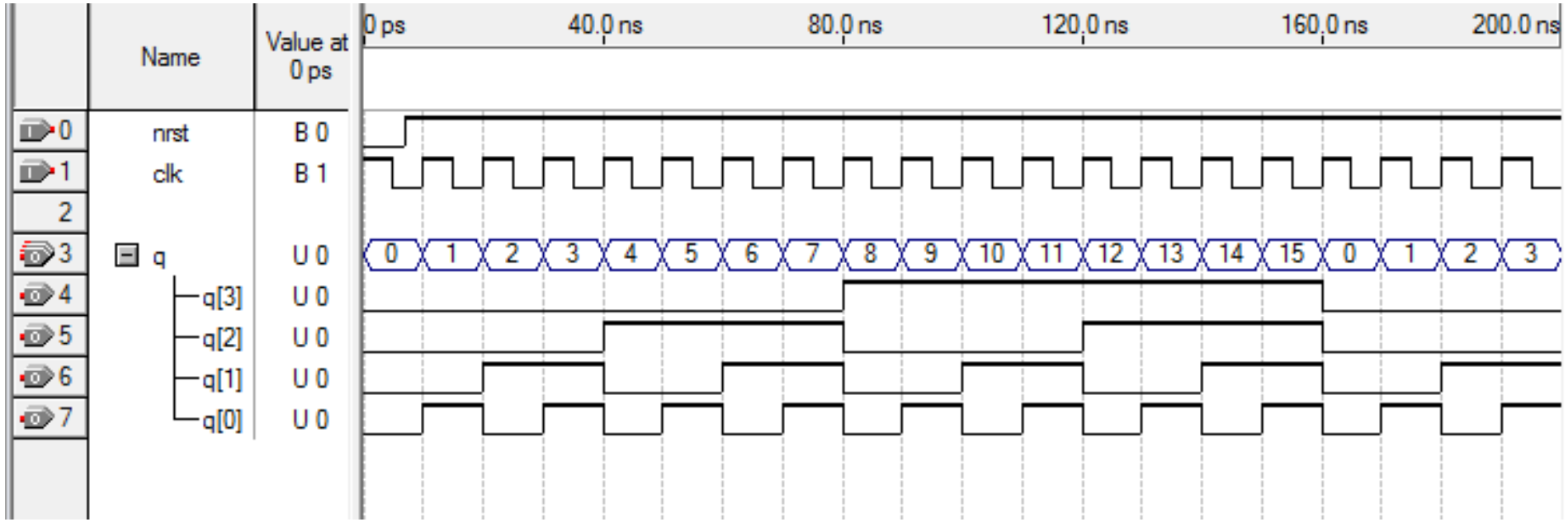

```
ARCHITECTURE arch1 OF cnt_mod16 IS
BEGIN
    PROCESS(clk, nrst)
        VARIABLE cnt : INTEGER RANGE 0 TO 15;
    BEGIN
        IF nrst = '0' THEN
            cnt := 0;
        ELSIF RISING_EDGE(clk) THEN
            cnt := cnt + 1;
        END IF;
        q <= STD_LOGIC_VECTOR(TO_UNSIGNED(cnt,4));
    END PROCESS;
END arch1;
```

Operação de adição é válida para o tipo INTEGER.

Exemplo cnt_mod16

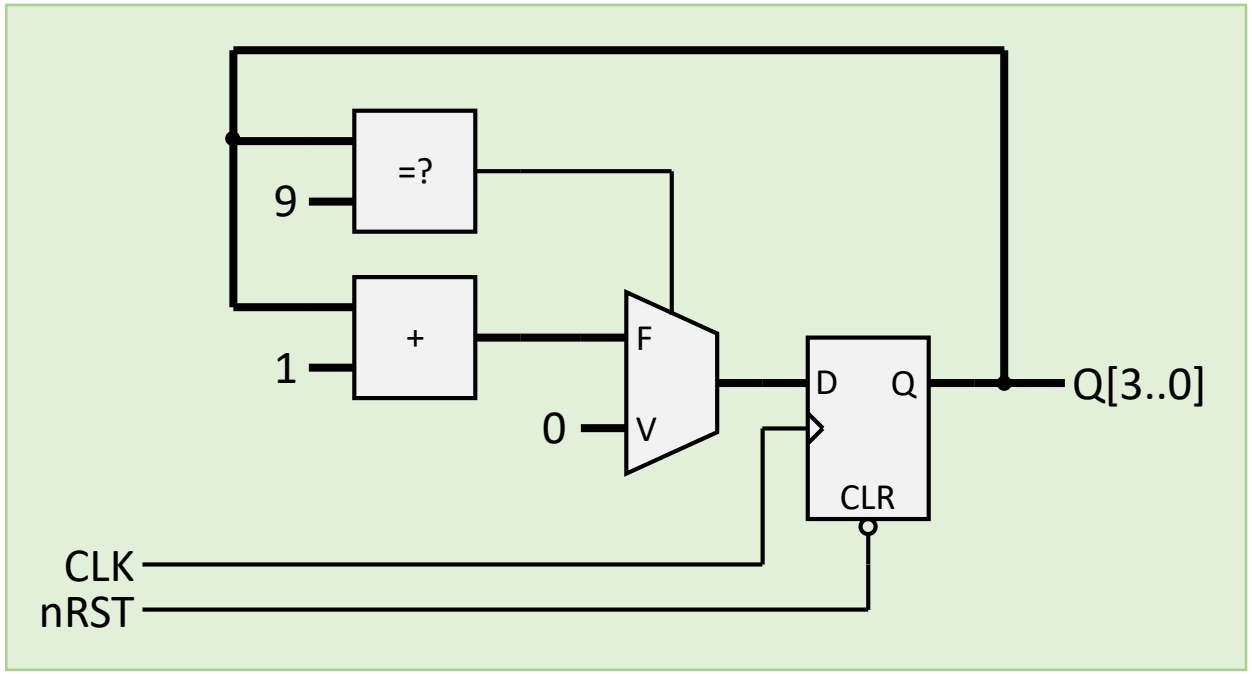
Exemplo cnt_mod16: **contador módulo 16** com reset assíncrono ativo em nível baixo. Flip-flops disparados pela borda de subida de *clock*.





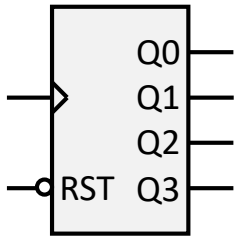
Exemplo cnt_mod10

Exemplo cnt_mod10: **contador módulo 10** com reset assíncrono ativo em nível baixo. Flip-flops disparados pela borda de subida de *clock.*



Exemplo cnt_mod10

Exemplo cnt_mod10: **contador módulo 10** com reset assíncrono ativo em nível baixo. Flip-flops disparados pela borda de subida de *clock*.

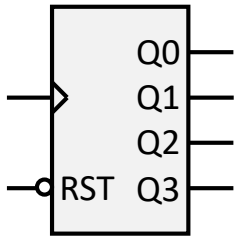



```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY cnt_mod10 IS
    PORT (
        clk, nrst : IN STD_LOGIC;
        q : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)
    );
END ENTITY;
.......
```

Continua na próxima página

Exemplo cnt_mod10

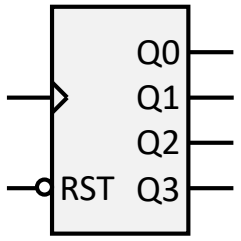
Exemplo cnt_mod10: **contador módulo 10** com reset assíncrono ativo em nível baixo. Flip-flops disparados pela borda de subida de *clock*.

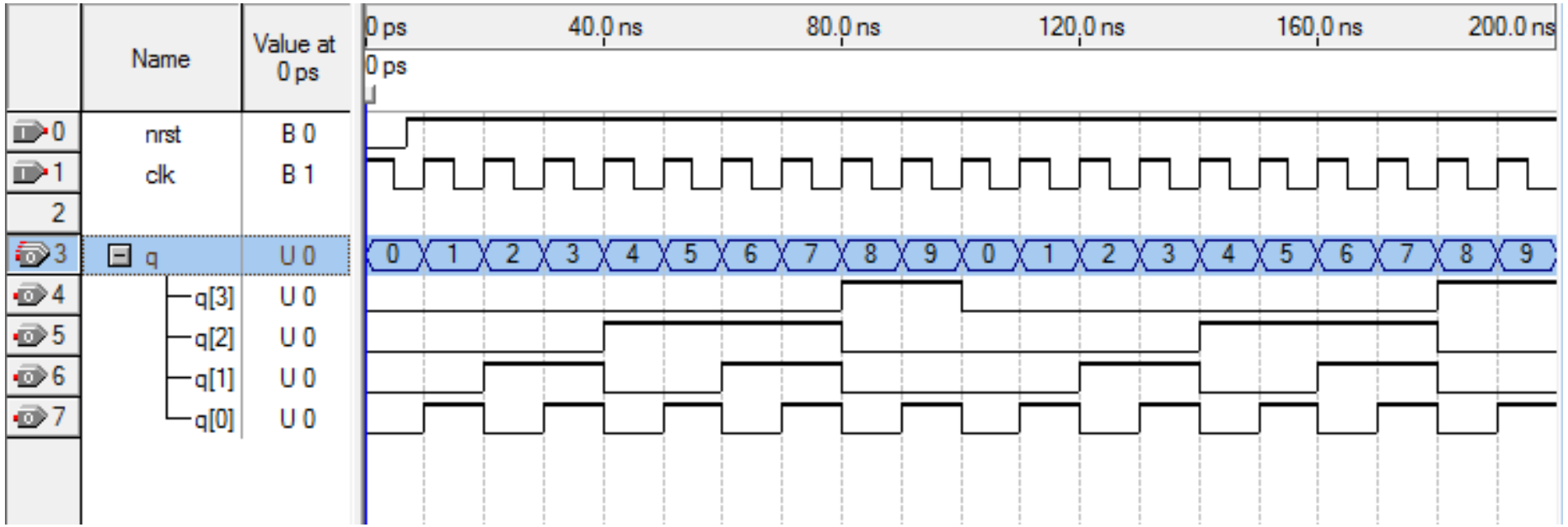



```
ARCHITECTURE arch1 OF cnt_mod10 IS
BEGIN
    PROCESS(clk, nrst)
        VARIABLE cnt : INTEGER RANGE 0 TO 9;
    BEGIN
        IF nrst = '0' THEN
            cnt := 0;
        ELSIF RISING_EDGE(clk) THEN
            IF cnt = 9 THEN
                cnt := 0;
            ELSE
                cnt := cnt + 1;
            END IF;
        END IF;
        q <= STD_LOGIC_VECTOR(TO_UNSIGNED(cnt,4));
    END PROCESS;
END arch1;
```

Exemplo cnt_mod10

Exemplo cnt_mod10: **contador módulo 10** com reset assíncrono ativo em nível baixo. Flip-flops disparados pela borda de subida de *clock*.





Exercício 1

Tendo como base o exemplo anterior, implementar um contador síncrono bidirecional módulo 10 com as seguintes entradas e saídas:

Entradas

- **nrst** (*reset*): *Reset* assíncrono ativo em nível lógico baixo. Leva a saída **q** para 0. Tem preferência sobre todas as outras entradas.
- **clk** (*clock*): as operações de contagem ocorrem na borda de subida desse sinal.
- **cnt_en** (*count enable*): quando ativado (nível alto), permite a contagem. A contagem é síncrona, ou seja, ocorre na transição de subida de **clk**.
- **dir** (*direction*): quando em nível alto o contador conta para cima (crescente) e em nível baixo conta para baixo (decrescente)

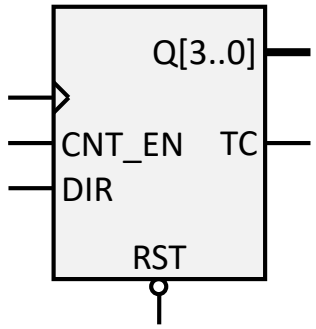
Saídas

- **q**: Saída da contagem com 4 bits
- **tc** (*terminal count*): É ativado (nível lógico alto) quando **cnt_en** estiver ativo **e** a contagem chegar ao fim, ou seja, quando estiver no valor 9 para **dir** = '1' ou 0 para **dir** = '0'. Essa saída é puramente combinacional, ou seja, não depende de *clock*.

Quando **dir** = '1' e a contagem estiver em nove, o próximo valor é 0. Se **dir** = '0' e a contagem estiver em 0, o próximo valor é 9.

Exercício 1

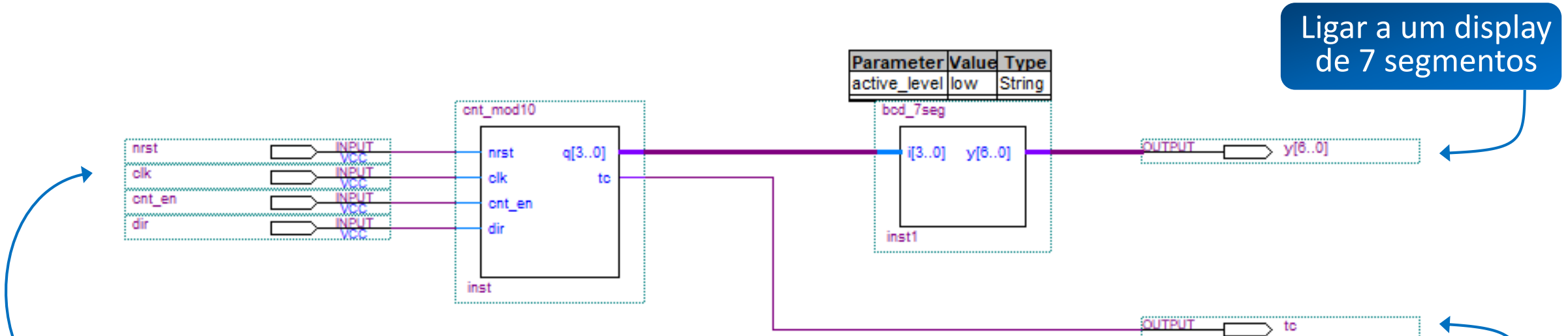
Implementar um contador síncrono bidirecional módulo 10




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

ENTITY cnt_mod10 IS
    PORT (
        nrst    : IN STD_LOGIC;     -- reset assinc. ativo baixo
        clk     : IN STD_LOGIC;     -- clock borda de subida
        cnt_en  : IN STD_LOGIC;     -- habilita contagem
        dir     : IN STD_LOGIC;     -- direção: 1 => crescente
        q : OUT STD_LOGIC_VECTOR(3 DOWNTO 0); -- saída
        tc: OUT STD_LOGIC           -- contagem terminal
    );
END ENTITY;
.......
```

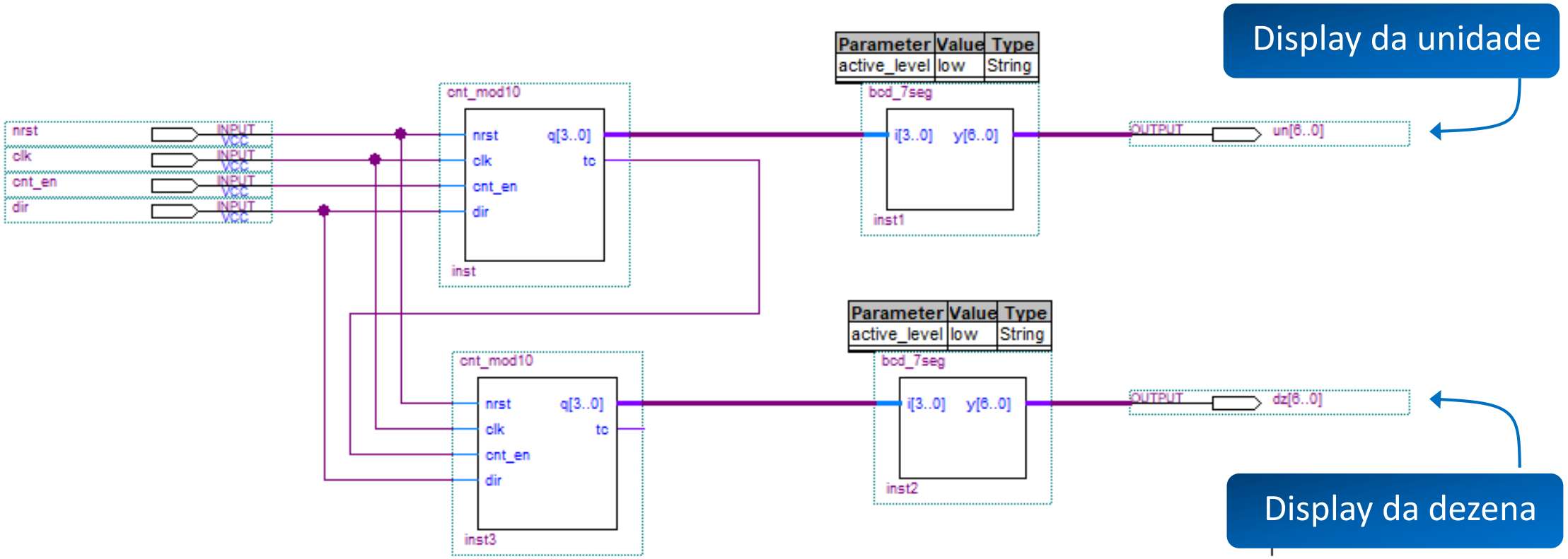
Exercício 1

Para testar esse exercício, interligar ao conversor de BCD para 7 segmentos desenvolvido na aula 8:



Exercício 2

Interligar dois contadores módulo 10 para fazer um contador decimal de dois dígitos (conta de 0 a 99 ou de 99 a 00, dependendo da direção)



Fim