



**Universidade Federal de Santa Catarina**  
**Centro Tecnológico**  
Departamento de Informática e Estatística  
Curso de Graduação em Ciências da Computação



# **Sistemas Digitais**

**INE 5406**

## **Aula 4-P**

**Descrição em VHDL, síntese e simulação de latches, flip-flops e registradores.**

**Prof. José Luís Güntzel**  
**[guntzel@inf.ufsc.br](mailto:guntzel@inf.ufsc.br)**

**[www.inf.ufsc.br/~guntzel/ine5406/ine5406.html](http://www.inf.ufsc.br/~guntzel/ine5406/ine5406.html)**

# Latches, Flip-flops e Registradores em VHDL

## ► Os Operadores em VHDL

	Operator Class	Operator
Highest precedence	Miscellaneous	** , ABS , NOT
	Multiplying	* , / , MOD , REM
	Sign	+ , -
	Adding	+ , - , &
	Shift	SLL , SRL , SLA , SRA , ROL , ROR
	Relational	= , /= , < , <= , > , >=
Lowest precedence	Logical	AND , OR , NAND , NOR , XOR , XNOR

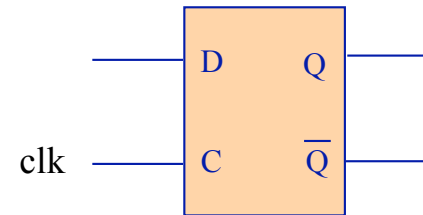
# Latches, Flip-flops e Registradores em VHDL

## ▶ Latch D (ativado por nível lógico alto)

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

ENTITY latchD IS
PORT ( D, clk : IN STD_LOGIC;
      Q :      OUT STD_LOGIC);
END latchD;

ARCHITECTURE comportamento OF latchD IS
BEGIN
  PROCESS (D, clk)
  BEGIN
    IF clk = '1' THEN
      Q <= D;
    END IF;
  END PROCESS;
END comportamento;
```



C	D	$Q_{t+1}$
0	X	$Q_t$
1	0	0
1	1	1

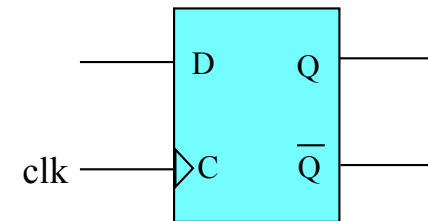
# Latches, Flip-flops e Registradores em VHDL

## ▶ Flip-flop D (disparado pela borda ascendente)

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

ENTITY ffd IS
PORT ( D, clk : IN STD_LOGIC;
      Q : OUT STD_LOGIC);
END ffd;

ARCHITECTURE comportamento OF ffd IS
BEGIN
  PROCESS (clk)
  BEGIN
    IF clk'EVENT AND clk = '1' THEN
      Q <= D;
    END IF;
  END PROCESS;
END comportamento;
```



C	D	$Q_{t+1}$
$\neq \uparrow$	X	$Q_t$
$\uparrow$	0	0
$\uparrow$	1	1

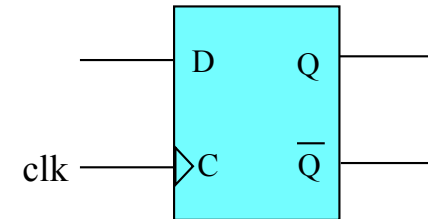
Atributo: refere-se a troca de nível de clk

# Latches, Flip-flops e Registradores em VHDL

## ▶ Flip-flop D (disparado pela borda ascendente) - Versão 2

```
LIBRARY ieee;  
USE ieee.std_logic_1164.all;  
  
ENTITY ffd IS  
  PORT ( D, clk : IN STD_LOGIC;  
        Q :      OUT STD_LOGIC);  
END ffd;
```

```
ARCHITECTURE comportamento OF ffd IS  
BEGIN  
  PROCESS  
  BEGIN  
    WAIT UNTIL clk'EVENT AND clk = '1';  
    Q <= D;  
  END PROCESS;  
END comportamento;
```



A lista de sensibilização é omitida

O Uso de WAIT UNTIL especifica implicitamente que somente o relógio faz parte da lista de sensibilização

# Latches, Flip-flops e Registradores em VHDL

---

## ▶ WAIT UNTIL

- Para efeitos de síntese de circuitos, **WAIT UNTIL** somente pode ser usado se ele for a **primeira atribuição do processo**
- Na verdade, **'EVENT** é redundante no comando **WAIT UNTIL** e portanto poderíamos simplificar para

**WAIT UNTIL clk='1';**

o que se refere ao sinal **clk** se tornar igual a “1”

- Entretanto, algumas ferramentas de síntese de circuitos a partir de VHDL exigem a inclusão do atributo **“EVENT”**

# Latches, Flip-flops e Registradores em VHDL

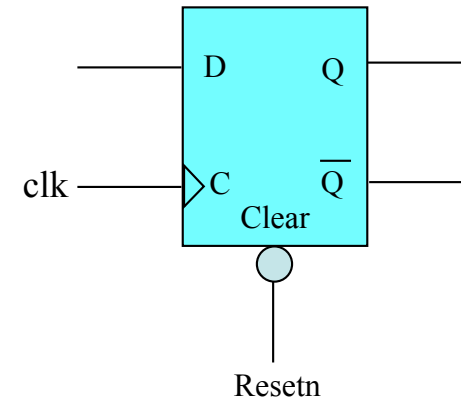
## ► Flip-flop D (com reset assíncrono ativado com lógica negada)

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

ENTITY ffd IS
PORT ( D, Resetn, clk : IN STD_LOGIC;
      Q : OUT STD_LOGIC);
END ffd;

ARCHITECTURE comportamento OF ffd IS
BEGIN
  PROCESS (Resetn, clk)
  BEGIN
    IF Resetn = '0' THEN
      Q <= '0' ;

    ELSIF clk'EVENT AND clk = '1' THEN
      Q <= D;
    END IF;
  END PROCESS;
END comportamento;
```



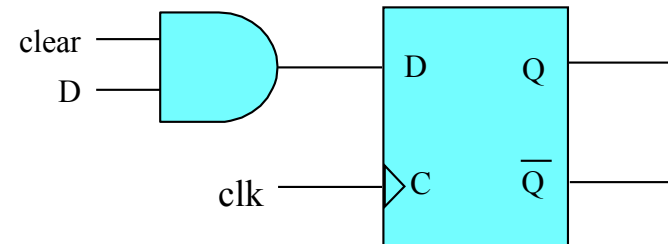
Primeiro testa se o reset assíncrono está ativado (pois é um sinal de mais alta hierarquia)

# Latches, Flip-flops e Registradores em VHDL

## ► Flip-flop D (com reset síncrono ativado com lógica negada)

```
LIBRARY ieee;  
USE ieee.std_logic_1164.all;  
  
ENTITY ffd IS  
PORT ( D, clear, clk : IN STD_LOGIC;  
      Q : OUT STD_LOGIC);  
END ffd;
```

```
ARCHITECTURE comportamento OF ffd IS  
BEGIN  
  PROCESS (clk)  
  BEGIN  
    IF clk'EVENT AND clk = '1' THEN  
      IF clear = '0' THEN  
        Q <= '0' ;  
      ELSIF  
        Q <= D;  
      END IF;  
    END PROCESS;  
  END comportamento;
```



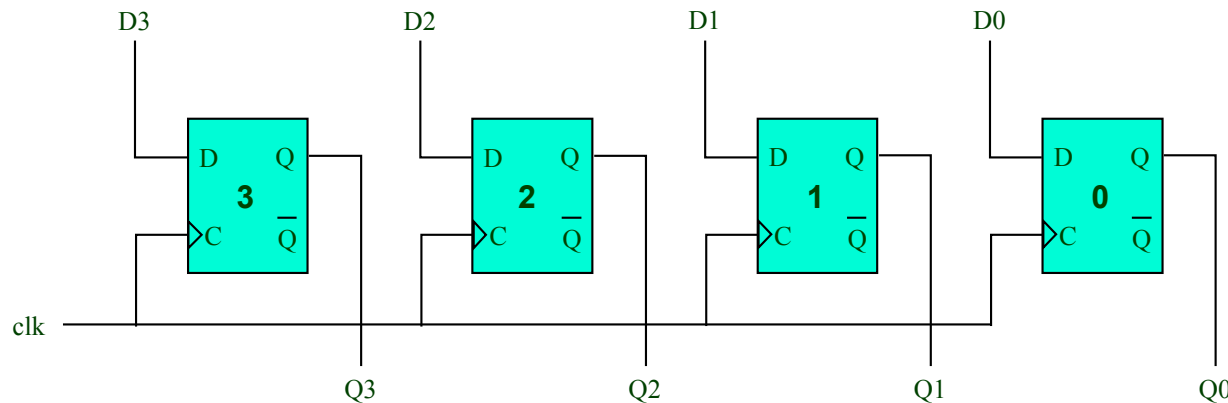
A ativação do reset está condicionada ao sinal clk estar passando pela borda ascendente (ou seja, este sinal é síncrono)



# Latches, Flip-flops e Registradores em VHDL

## ▶ Registradores

Usando explicitamente flip-flops do tipo D



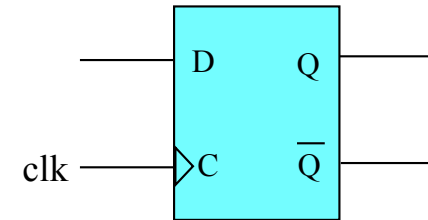
Uma abordagem possível para se descrever um registrador de vários bits é criar uma entidade que instancia flip-flops na quantidade desejada. Vejamos...

# Latches, Flip-flops e Registradores em VHDL

## ▶ Registradores (Usando explicitamente flip-flops do tipo D)

```
LIBRARY ieee;  
USE ieee.std_logic_1164.all;  
  
ENTITY ffD IS  
  PORT ( D, clk : IN STD_LOGIC;  
        Q :      OUT STD_LOGIC);  
END ffD;
```

```
ARCHITECTURE comportamento OF ffD IS  
BEGIN  
  PROCESS (clk)  
  BEGIN  
    IF clk'EVENT AND clk = '1' THEN  
      Q <= D;  
    END IF;  
  END PROCESS;  
END comportamento;
```



# Latches, Flip-flops e Registradores em VHDL

## ▶ Registrador de 4 bits (com flip-flops do tipo D)

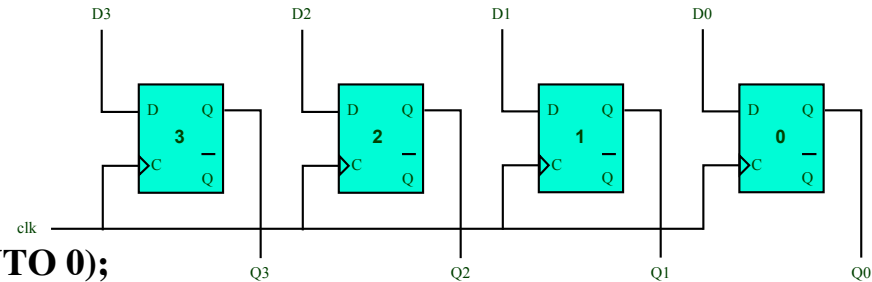
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY reg4b IS
PORT ( clk : IN STD_LOGIC;
      D : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
      Q : OUT STD_LOGIC_VECTOR (3 DOWNTO 0));
END reg4b;

ARCHITECTURE estrutura OF reg4b IS

COMPONENT ffD
PORT ( D, clk : IN STD_LOGIC; Q : OUT STD_LOGIC);
END COMPONENT;

BEGIN
ffD0: ffD PORT MAP (D(0), clk, Q(0));
ffD1: ffD PORT MAP (D(1), clk, Q(1));
ffD2: ffD PORT MAP (D(2), clk, Q(2));
ffD3: ffD PORT MAP (D(3), clk, Q(3));
END estrutura;
```



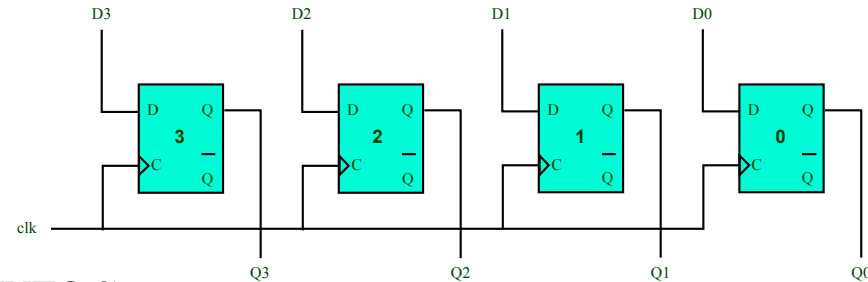
# Latches, Flip-flops e Registradores em VHDL

## ▶ Registrador de 4 bits (versão não hierárquica)

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

```
ENTITY reg4b IS  
  PORT ( clk : IN STD_LOGIC;  
        D : IN STD_LOGIC_VECTOR (3 DOWNTO 0);  
        Q : OUT STD_LOGIC_VECTOR (3 DOWNTO 0));  
END reg4b;
```

```
ARCHITECTURE comportamento OF reg4b IS  
  BEGIN  
    PROCESS (clk)  
      BEGIN  
        IF clk'EVENT AND clk = '1' THEN  
          Q <= D;  
        END IF;  
      END PROCESS;  
    END comportamento;
```



# Latches, Flip-flops e Registradores em VHDL

## ▶ Registrador de 4 bits (versão com reset assíncrono)

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

```
ENTITY reg4b IS  
  PORT ( Resetn, clk : IN STD_LOGIC;  
        D : IN STD_LOGIC_VECTOR (3 DOWNTO 0);  
        Q : OUT STD_LOGIC_VECTOR (3 DOWNTO 0));  
END reg4b;
```

```
ARCHITECTURE comportamento OF reg4b IS  
  BEGIN  
    PROCESS (clk)  
      BEGIN  
        IF Resetn = '0' THEN  
          Q <= "0000";  
        ELSIF clk'EVENT AND clk = '1' THEN  
          Q <= D;  
        END IF;  
      END PROCESS;  
    END comportamento;
```

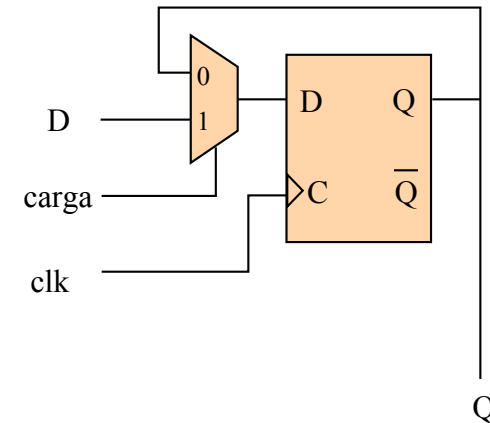
# Latches, Flip-flops e Registradores em VHDL

## ▶ Registrador de 4 bits (versão não hierárquica com carga)

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

```
ENTITY reg4b IS  
  PORT ( clk, carga : IN STD_LOGIC;  
        D :          IN STD_LOGIC_VECTOR (3 DOWNTO 0);  
        Q :          OUT STD_LOGIC_VECTOR (3 DOWNTO 0));  
END reg4b;
```

```
ARCHITECTURE comportamento OF reg4b IS  
  BEGIN  
    PROCESS (clk)  
      BEGIN  
        IF clk'EVENT AND clk = '1' THEN  
          IF carga='1' THEN  
            Q <= D;  
          END IF;  
        END IF;  
      END PROCESS;  
END comportamento;
```



# Latches, Flip-flops e Registradores em VHDL

## ▶ Registrador de 32 bits (versão não hierárquica, c/ carga)

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

ENTITY reg32b IS
PORT ( clk, carga : IN STD_LOGIC;
      D :          IN STD_LOGIC_VECTOR (31 DOWNTO 0);
      Q :          OUT STD_LOGIC_VECTOR (31 DOWNTO 0));
END reg32b;

ARCHITECTURE comportamento OF reg32b IS
BEGIN
  PROCESS (clk)
  BEGIN
    IF clk'EVENT AND clk = '1' THEN
      IF carga='1' THEN
        Q <= D;
      END IF;
    END IF;
  END PROCESS;
END comportamento;
```

# Latches, Flip-flops e Registradores em VHDL

## ► Generalizando para N bits

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

ENTITY regn IS
  GENERIC (N : INTEGER := 16);
  PORT ( Resetn, clk, carga : IN   STD_LOGIC;
        D :                   IN   STD_LOGIC_VECTOR (N-1 DOWNTO 0);
        Q :                   OUT  STD_LOGIC_VECTOR (N-1 DOWNTO 0));
END regn;

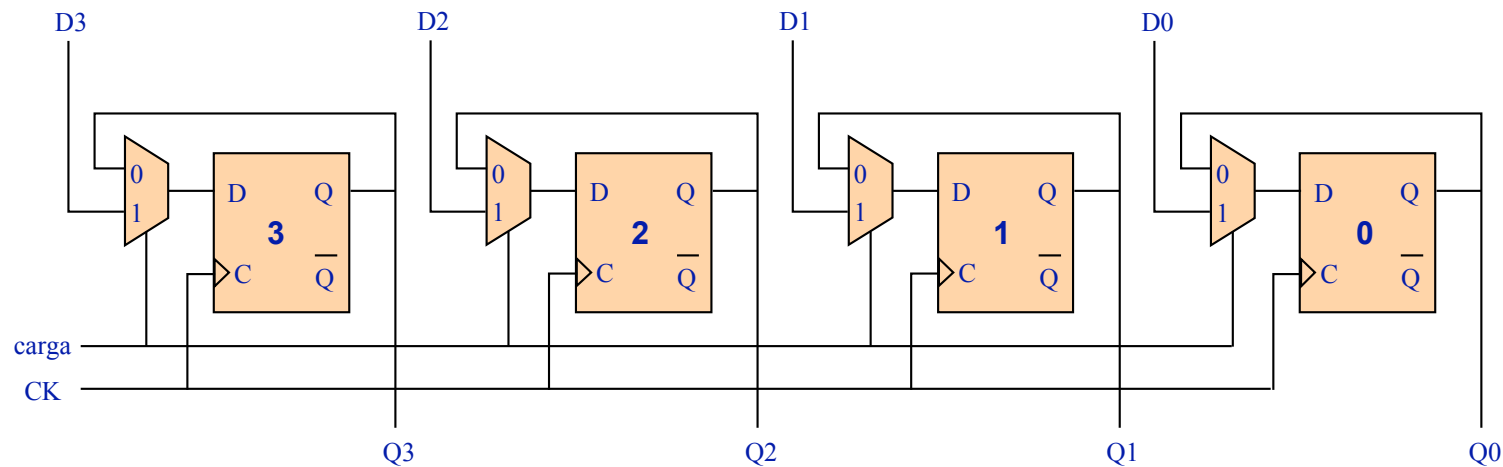
ARCHITECTURE comportamento OF regn IS
BEGIN
  PROCESS (clk)
  BEGIN
    IF Resetn = '0' THEN
      Q <= (OTHERS => '0');
    ELSIF clk'EVENT AND clk = '1' THEN
      IF carga='1' THEN
        Q <= D;
      END IF;
    END IF;
  END PROCESS;
END comportamento;
```



# Latches, Flip-flops e Registradores em VHDL

## ▶ Experimento 1

Descrever em VHDL, sintetizar e simular para FPGAs Stratix o seguinte registrador:



Usar hierarquia e acrescentar um reset assíncrono ativado col lógica direta.

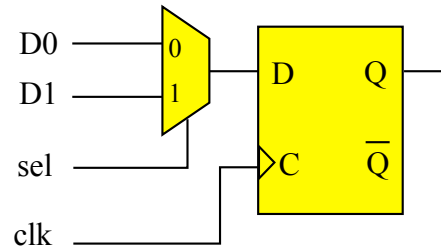
Levantar as seguintes informações:

- Número de LEs
- Características temporais

# Latches, Flip-flops e Registradores em VHDL

## ► Registrador-Deslocador

- Uma alternativa para se descrever um registrador-deslocador é utilizar a hierarquia
- Suponha que se deseje usar o circuito abaixo como componente básico



Então, descrevendo-o em VHDL...

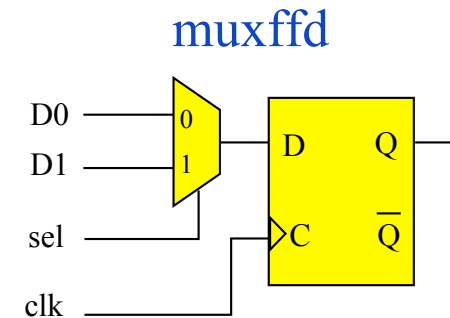
# Latches, Flip-flops e Registradores em VHDL

## ▶ Registrador-Deslocador (componente básico)

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

ENTITY muxffd IS
PORT ( D0, D1, sel, clk: IN STD_LOGIC;
      Q : OUT STD_LOGIC );
END muxffd;

ARCHITECTURE comportamento OF muxffd IS
BEGIN
  PROCESS
  BEGIN
    WAIT UNTIL clk'EVENT AND clk = '1' ;
    IF sel = '0' THEN
      Q <= D0;
    ELSE
      Q <= D1;
    END IF;
  END PROCESS;
END comportamento;
```



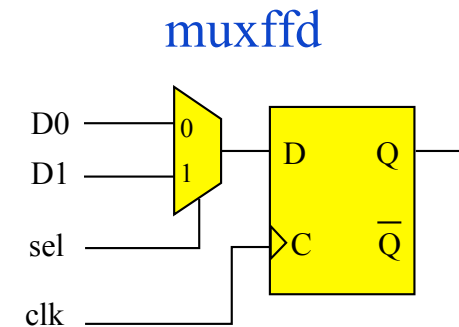
# Latches, Flip-flops e Registradores em VHDL

## ▶ Registrador-Deslocador (de 4 bits, hierárquico)

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

ENTITY shift4 IS
    PORT ( R          : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
          carga, serial, clk : IN STD_LOGIC;
          Q          : BUFFER STD_LOGIC_VECTOR(3 DOWNTO 0) );
END shift4;

ARCHITECTURE estrutura OF shift4 IS
    COMPONENT muxffd
        PORT ( D0, D1, sel, clk : IN STD_LOGIC;
              Q                : OUT STD_LOGIC );
    END COMPONENT;
BEGIN
    estagio3: muxffd PORT MAP ( serial, R(3), carga, clk, Q(3) );
    estagio2: muxffd PORT MAP ( Q(3), R(2), carga, clk, Q(2) );
    estagio1: muxffd PORT MAP ( Q(2), R(1), carga, clk, Q(1) );
    estagio0: muxffd PORT MAP ( Q(1), R(0), carga, clk, Q(0) );
END estrutura;
```



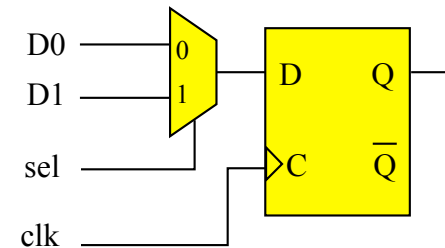
# Latches, Flip-flops e Registradores em VHDL

## ▶ Registrador-Deslocador (de 4 bits, não hierárquico)

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

ENTITY shift4 IS
    PORT ( R          : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
          carga, serial, clk : IN STD_LOGIC;
          Q          : BUFFER STD_LOGIC_VECTOR(3 DOWNTO 0) );
END shift4;

ARCHITECTURE comportamento OF shift4 IS
BEGIN
    PROCESS
    BEGIN
        WAIT UNTIL clk'EVENT AND clk = '1' ;
        IF carga = '1' THEN
            Q <= R;
        ELSE
            Q(0) <= Q(1);
            Q(1) <= Q(2);
            Q(2) <= Q(3);
            Q(3) <= serial;
        END IF;
    END PROCESS;
END comportamento;
```



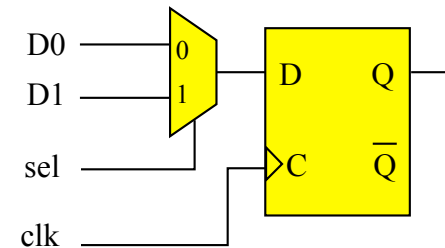
# Latches, Flip-flops e Registradores em VHDL

## ▶ Registrador-Deslocador (com n bits, não hierárquico)

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

ENTITY shiftn IS
    GENERIC ( N : INTEGER := 8 );
    PORT ( R          : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
          carga, serial, clk : IN STD_LOGIC;
          Q          : BUFFER STD_LOGIC_VECTOR(N-1 DOWNTO 0) );
END shiftn;

ARCHITECTURE comportamento OF shiftn IS
BEGIN
    PROCESS
    BEGIN
        WAIT UNTIL clk'EVENT AND clk = '1' ;
        IF carga = '1' THEN
            Q <= R;
        ELSE
            Genbits: FOR i IN 0 TO N-2 LOOP
                Q(i) <= Q(i+1);
            END LOOP;
            Q(N-1) <= w;
        END IF;
    END PROCESS;
END comportamento;
```



# Latches, Flip-flops e Registradores em VHDL

---

## ▶ Experimento 2

Descrever em VHDL, sintetizar e simular para FPGAs Stratix o um registrador de deslocamento para a direita da transparência anterior.

Acrescentar um reset assíncrono ativado com lógica direta.

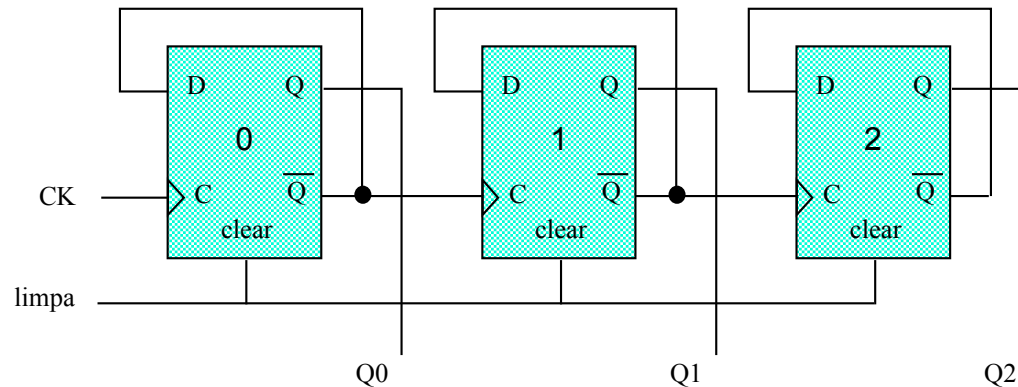
Levantar as seguintes informações:

- Número de LEs
- Características temporais

# Latches, Flip-flops e Registradores em VHDL

## ▶ Contador Assíncrono (*ripple*) de 3 bits

Usando flip-flops D – versão 2



- Para descrever este tipo de contador em VHDL usando hierarquia, iremos instanciar o flip-flop D
- Porém, a descrição do flip-flop D em VHDL deve conter ambas saídas, Q e NQ



# Latches, Flip-flops e Registradores em VHDL

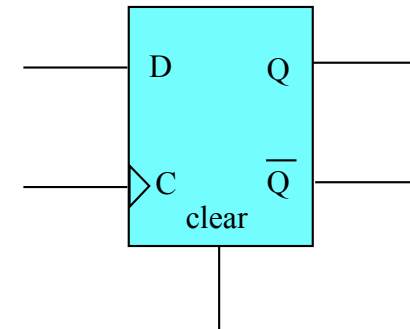
## ► Flip-flop D (versão 2)

Com saídas Q e NQ

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

ENTITY ffDv2 IS
PORT ( D, clk, limpa : IN STD_LOGIC;
      Q, NQ : OUT STD_LOGIC);
END ffDv2;

ARCHITECTURE comportamento OF ffDv2 IS
BEGIN
  PROCESS (clk, limpa)
  BEGIN
    IF limpa='1' THEN
      Q <= '0';  NQ <= '1';
    ELSIF clk'EVENT AND clk = '1' THEN
      Q <= D;  NQ <= not D;
    END IF;
  END PROCESS;
END comportamento;
```



# Latches, Flip-flops e Registradores em VHDL

## ▶ Contador Assíncrono (*ripple*) de 3 bits

(Usando o flip-flop versão 2)

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

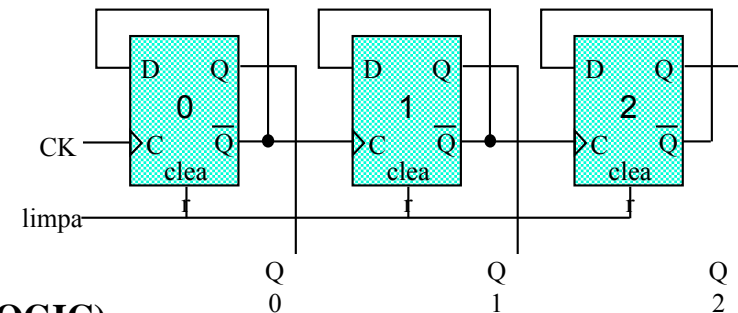
ENTITY cont3b IS
PORT ( clk, limpa : IN STD_LOGIC;
      Aout : OUT STD_LOGIC_VECTOR (2 DOWNTO 0));
END cont3b;

ARCHITECTURE estrutura OF cont3b IS

    SIGNAL NQ0, NQ1, NQ2 : STD_LOGIC;

    COMPONENT ffDv2
    PORT ( D, clk, limpa: IN STD_LOGIC; Q, NQ : OUT STD_LOGIC);
    END COMPONENT;

BEGIN
    ffD0: ffDv2 PORT MAP (NQ0, clk, limpa, Aout(0), NQ0);
    ffD1: ffDv2 PORT MAP (NQ1, NQ0, limpa, Aout(1), NQ1);
    ffD2: ffDv2 PORT MAP (NQ2, NQ1, limpa, Aout(2), NQ2);
END estrutura;
```



# Latches, Flip-flops e Registradores em VHDL

## ▶ Contador Assíncrono (*ripple*) de 4 bits

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

ENTITY contador IS
PORT ( clk, limpa , carga : IN    STD_LOGIC;
      Q : OUT  STD_LOGIC_VECTOR( 3 DOWNTO 0 ) );
END contador ;

ARCHITECTURE comportamento OF contador IS
    SIGNAL conta : STD_LOGIC_VECTOR ( 3 DOWNTO 0 ) ;
BEGIN
    PROCESS (clk, limpa)
    BEGIN
        IF limpa = '0' THEN
            conta <= "0000";
        ELSIF ( clk'EVENT AND clk = '1' ) THEN
            IF carga = '1' THEN
                conta <= conta + 1;
            ELSE
                conta <= conta;
            END IF;
        END IF;
    END PROCESS;
    Q <= conta;
END comportamento;
```

# Latches, Flip-flops e Registradores em VHDL

## ▶ Contador Assíncrono (*ripple*) de n bits

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;      (Com carga paralela e usando sinais do tipo “integer”)

ENTITY contnb IS
PORT ( R           : IN      INTEGER RANGE 0 TO 15;
      clk, limpa , carga : IN      STD_LOGIC;
      Q           : BUFFER INTEGER RANGE 0 TO 15 );
END contnb ;

ARCHITECTURE comportamento OF contnb IS
BEGIN
  PROCESS (clk, limpa)
  BEGIN
    IF limpa = '0' THEN
      Q <= 0;
    ELSIF ( clk'EVENT AND clk = '1' ) THEN
      IF carga = '1' THEN
        Q <= R;
      ELSE
        Q <= Q + 1;
      END IF;
    END IF;
  END PROCESS;
END comportamento;
```

# Latches, Flip-flops e Registradores em VHDL

---

## ▶ Experimento 3

Descrever em VHDL, sintetizar e simular para FPGAs Stratix o um registrador contador da transparência anterior.

Levantar as seguintes informações:

- Número de LEs
- Características temporais