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

Sistemas Digitais

INE 5406

Aula 8-P

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

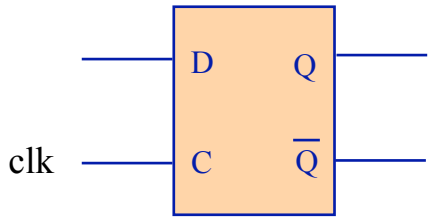
Prof. José Luís Güntzel
guntzel@inf.ufsc.br

www.inf.ufsc.br/~guntzel/ine5406/ine5406.html

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;
```

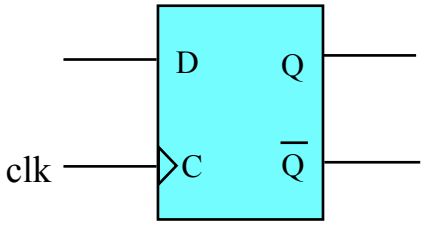


C	D	Q_{t+1}
0	X	Q_t
1	0	0
1	1	1

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;
```



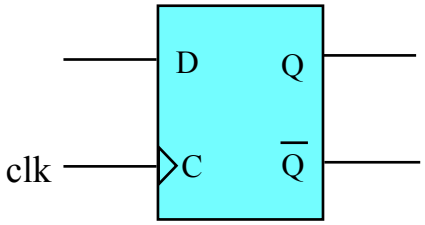
C	D	Q_{t+1}
$\neq\uparrow$	X	Q_t
$\uparrow$	0	0
$\uparrow$	1	1

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';

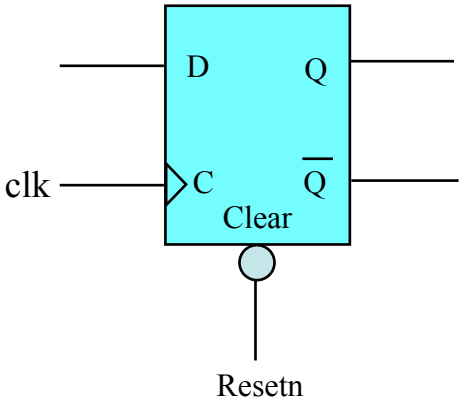
o que se refere ao sinal **clk** se tornar igual a "1"

- 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;
```

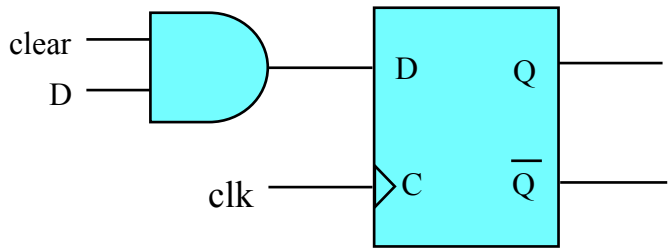


Primeiro testa se o reset assíncrono está ativado (pois é um sinal de mais alta hierarquia)

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' ;
         ELSE
           Q <= D;
         END IF;
      END IF;
   END PROCESS;
END comportamento;
```

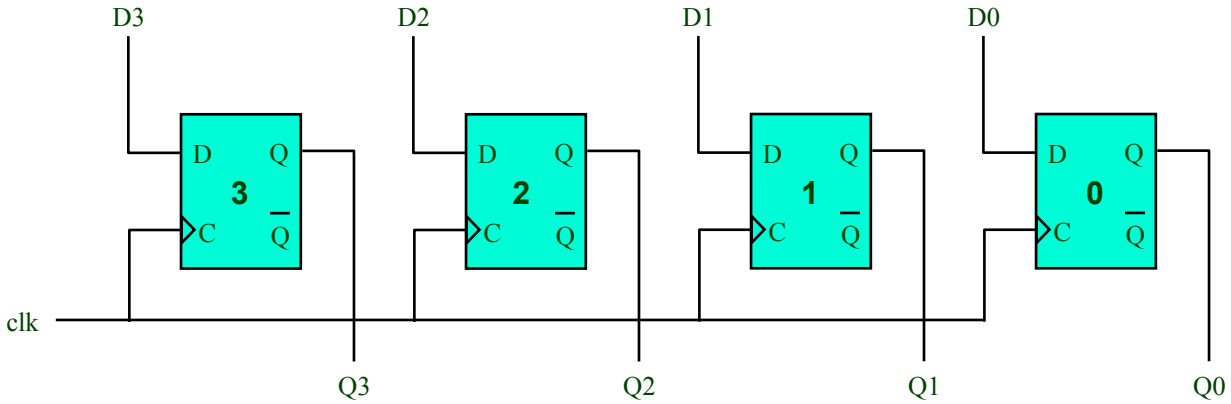


A ativação do reset está condicionada ao sinal clk estar passando pela borda ascendente (ou seja, este sinal é síncrono)

Latches, Flip-flops e Registradores em VHDL

Registradores

Usando explicitamente flip-flops do tipo D

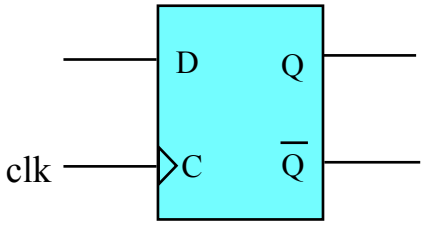


Uma abordagem possível para se descrever um registrador de vários bits é criar uma entidade que instancia flip-flops na quantidade desejada. Vejamos…

Latches, Flip-flops e Registradores em VHDL

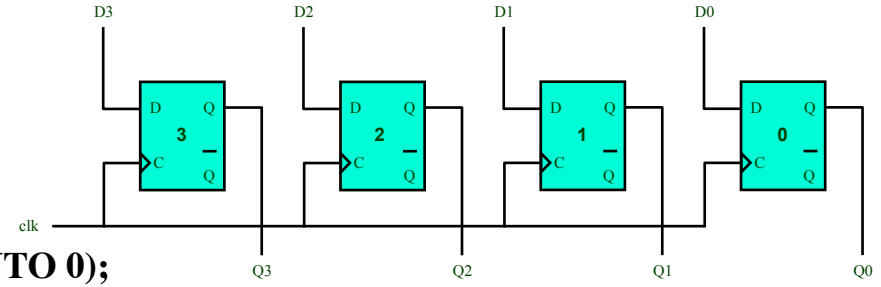
Registradores (Usando explicitamente flip-flops do tipo D)

```
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)

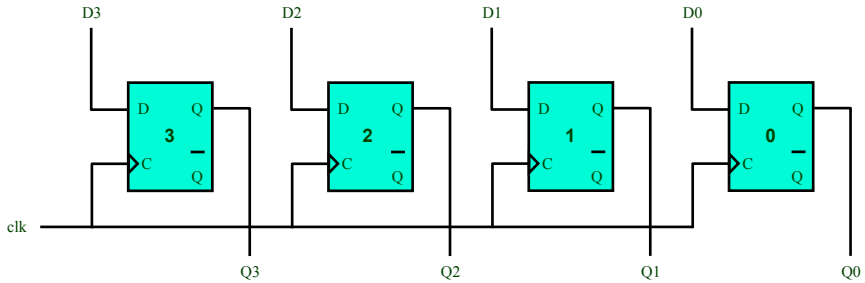



```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

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)

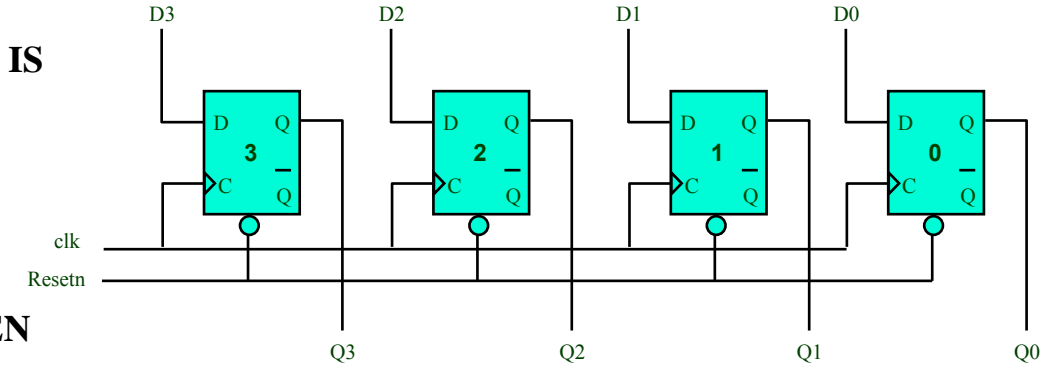
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY reg4 IS
PORT ( clk :  IN STD_LOGIC;
          D :    IN STD_LOGIC_VECTOR (3 DOWNTO 0);
          Q :    OUT STD_LOGIC_VECTOR (3 DOWNTO 0));
END reg4;

ARCHITECTURE comportamento OF reg4 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 não hierárquica com reset assíncrono)

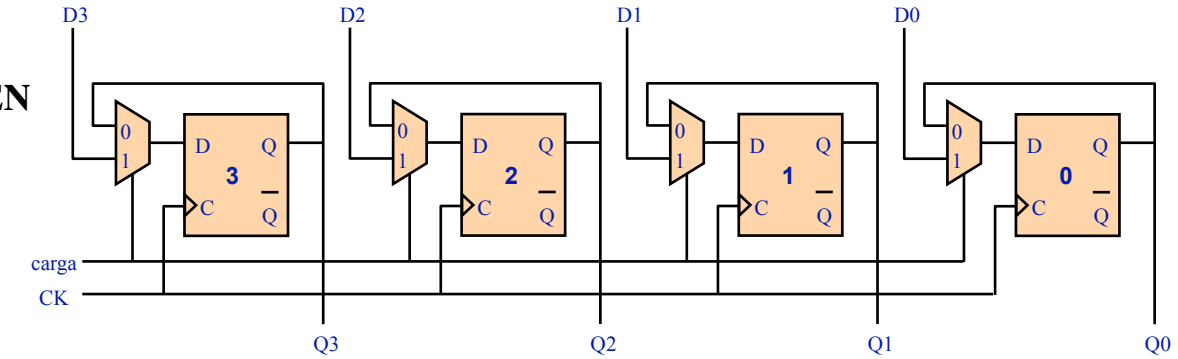
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY reg4 IS
PORT ( Resetn, clk :  IN STD_LOGIC;
          D :    IN STD_LOGIC_VECTOR (3 DOWNTO 0);
          Q :    OUT STD_LOGIC_VECTOR (3 DOWNTO 0));
END reg4;

ARCHITECTURE comportamento OF reg4 IS
BEGIN
   PROCESS (Resetn , 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 habilitação de carga)

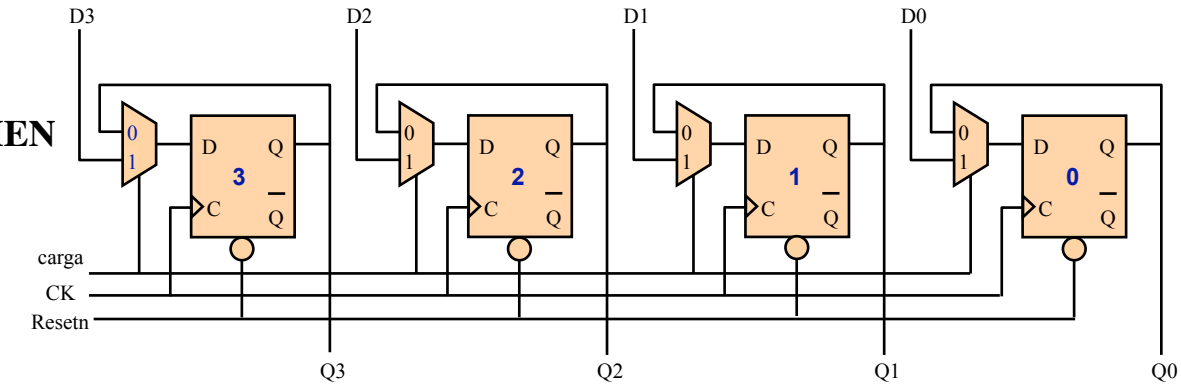
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY reg4c IS
PORT ( clk, carga : IN STD_LOGIC;
          D :           IN STD_LOGIC_VECTOR (3 DOWNTO 0);
          Q :           OUT STD_LOGIC_VECTOR (3 DOWNTO 0));
END reg4c;

ARCHITECTURE comportamento OF reg4c 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 4 bits (versão não hierárquica com habilitação de carga e reset assíncrono)

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

ENTITY reg4c IS
PORT ( clk, carga, Resetn : IN STD_LOGIC;
          D :           IN STD_LOGIC_VECTOR (3 DOWNTO 0);
          Q :           OUT STD_LOGIC_VECTOR (3 DOWNTO 0));
END reg4c;

ARCHITECTURE comportamento OF reg4c IS
BEGIN
   PROCESS (clk, Resetn)
   BEGIN
      IF Resetn = '0' THEN
         Q <= "0000";
      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

Registrador de 32 bits (versão não hierárquica com habilitação de carga e reset assíncrono)

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

ENTITY reg32c IS
PORT ( clk, carga : IN STD_LOGIC;
          D :           IN STD_LOGIC_VECTOR (31 DOWNTO 0);
          Q :           OUT STD_LOGIC_VECTOR (31 DOWNTO 0));
END reg32c;

ARCHITECTURE comportamento OF reg32c IS
BEGIN
   PROCESS (clk)
   BEGIN
      IF Resetn = '0' THEN
         Q <= "00000000000000000000000000000000";
      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

Generalizando para N bits

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

ENTITY regnc IS
GENERIC (N : INTEGER := 4 );
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 regnc;

ARCHITECTURE comportamento OF regnc IS
BEGIN
   PROCESS (clk, Resetn )
   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;
```

Daqui para frente, este será o estilo de codificação VHDL que iremos adotar para registradores!

Latches, Flip-flops e Registradores em VHDL

Experimento 1: registrador 4 bits s/ sinal de carga

1. Na pasta Meus_documentos, criar uma pasta com o seu nome (p. ex., "Paulo"). Na pasta "Paulo", criar uma pasta com nome de "reg4bits".
2. Acessar o sítio "www.inf.ufsc.br/~guntzel/ine5406/aula6P" e baixar para a pasta "reg4bits" os seguintes arquivos:
   -> reg4bits.vhd
   -> estimulosReg4bits.do
3. Abrir o Quartus II e criar um projeto na pasta "reg4bits", usando "reg4bits.vhd" como toplevel. Escolher o dispositivo FPGA EP2C35F672C6 e selecionar o ModelSim-Altera como EDA Simulation Tool.
4. Compilar o projeto criado.
5. Anotar os parâmetros tsu, th e tco resultantes da compilação.

Latches, Flip-flops e Registradores em VHDL

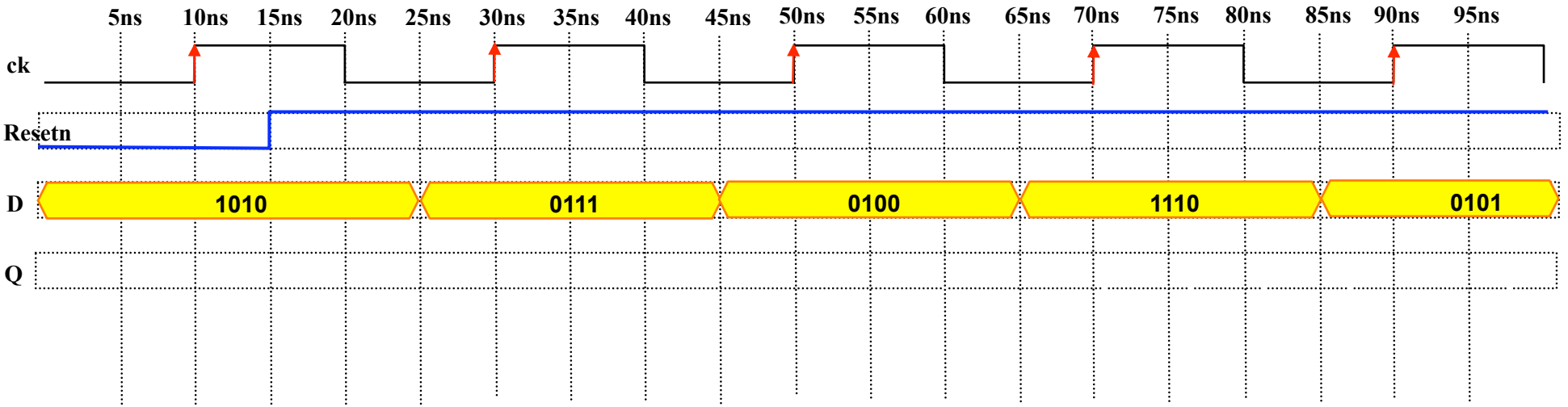
Experimento 1: registrador 4 bits s/ sinal de carga

6. Chamar o ModelSim-Altera a partir da interface do Quartus II (em Tools)
7. Realizar uma simulação "gate-level" com o ModelSim-Altera utilizando o arquivo de estímulos "estimulosReg4bits.do". (ver slides 1P.42 a 1P.45.)

Latches, Flip-flops e Registradores em VHDL

Experimento 1: registrador 4 bits s/ sinal de carga

Análise da simulação



Latches, Flip-flops e Registradores em VHDL

Experimento 2: registrador 4 bits c/ sinal de carga

1. Em Meus_documentos, na pasta criada com seu nome (p.e.x, "Paulo"), criar uma pasta com nome de "reg4bitsc".
2. Acessar o sítio "www.inf.ufsc.br/~guntzel/ine5406/aula6P" e baixar para a pasta "reg4bitsc" os seguintes arquivos:
   -> reg4bitsc.vhd
   -> estimulosReg4bitsc.do
3. Abrir o Quartus II e criar um projeto na pasta "reg4bitsc", usando "reg4bitsc.vhd" como toplevel. Escolher o dispositivo FPGA EP2C35F672C6 e selecionar o ModelSim-Altera como EDA Simulation Tool.
4. Compilar o projeto criado.
5. Anotar os parâmetros tsu, th e tco resultantes da compilação.

Latches, Flip-flops e Registradores em VHDL

Experimento 2: registrador 4 bits c/ sinal de carga

6. Chamar o ModelSim-Altera a partir da interface do Quartus II (em Tools)
7. Realizar uma simulação "gate-level" com o ModelSim-Altera utilizando o arquivo de estímulos "estimulosReg4bitsc.do". (ver slides 1P.42 a 1P.45.)

Latches, Flip-flops e Registradores em VHDL

Experimento 2: registrador 4 bits c/ sinal de carga

Análise da simulação