Circuitos Digitais II - 6882

André Barbosa Verona
Nardênio Almeida Martins

Universidade Estadual de Maringá
Departamento de Informática

Bacharelado em Ciência da Computação

Aula de Hoje

Implementação e simulação dos seguintes circuitos sequenciais:

Latch RS
Latch RS com Entrada Clock

Latch D
Latch D com Entrada Clock

Flip-Flop JK Mestre-Escravo
Flip-Flop JK Mestre-Escravo com Entradas Preset e Clear

Flip-flop D
Flip-Flop T

Aula de Hoje

Criar as seguintes pastas no work:

latch_rs
latch_rs_clk

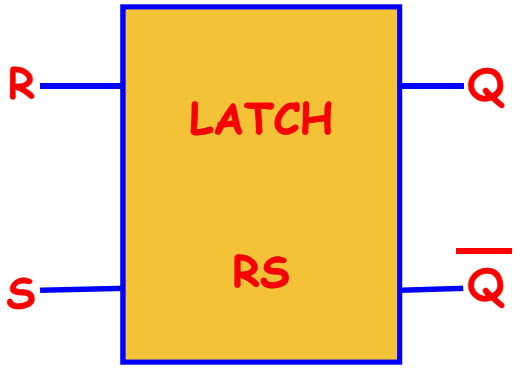
latch_d
latch_d_clk

flipflopjk
flipflopjk_clrpst

flipflopd
flipflopt

Latch RS

Bloco Lógico e Tabela Verdade



S	R	Qa	Qf
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	X
1	1	1	X

Qf=Qa Mantém a saída anterior

Qf=0 Reset da saída anterior

Qf=1 Set da saída anterior

Entradas não permitidas

S	R	Qf
0	0	Qa
0	1	0
1	0	1
1	1	X

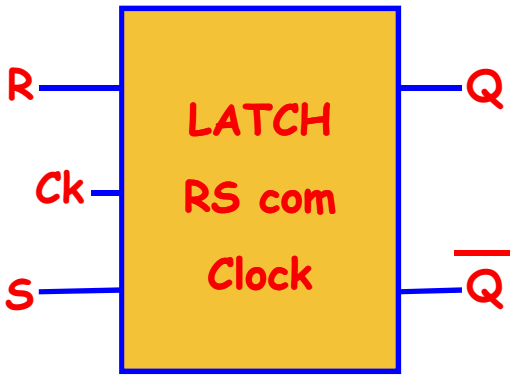
VHDL – Código

Solução: Latch RS

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY latchrs IS
   PORT( set, reset : IN BIT;              -- STD_LOGIC;
              q, qbar : BUFFER BIT);       -- STD_LOGIC;
END latchrs;
ARCHITECTURE comportamental OF latchrs IS
BEGIN
   PROCESS (set, reset)
   BEGIN
          IF     (reset = '1' AND set = '0') THEN q <= '0'; qbar <= NOT q;
          ELSIF (reset = '0' AND set = '1') THEN q <= '1'; qbar <= NOT q;
          ELSIF (reset = '1' AND set = '1') THEN q <= '1'; qbar <= '1';
          ELSE q <= q; qbar <= qbar;
          END IF;
   END PROCESS;
END comportamental;
```

Latch RS com Entrada Clock

Bloco Lógico e Tabela Verdade



Se o clock=0 ⇒ Latch permanece no seu estado anterior, mesmo que variem as entradas S e R

Ck=0 ⇒ Qf = Qa

Se o clock=1 ⇒ Latch funciona como um Latch RS

S	R	Qf
0	0	Qa
0	1	0
1	0	1
1	1	X

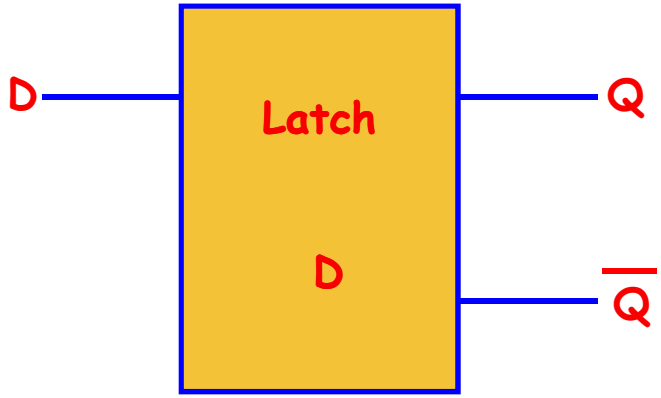
VHDL – Código

Solução: Latch RS com Entrada Clock

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY latchrs_clk IS
   PORT(clk, set, reset : IN BIT;            -- STD_LOGIC;
                q, qbar : BUFFER BIT);       -- STD_LOGIC;
END latchrs_clk;
ARCHITECTURE comportamental OF latchrs_clk IS
BEGIN
   PROCESS (clk, set, reset)
   BEGIN
          IF     (clk = '1' AND reset = '1' AND set = '0') THEN q <= '0'; qbar <= NOT q;
          ELSIF (clk = '1' AND reset = '0' AND set = '1') THEN q <= '1'; qbar <= NOT q;
          ELSIF (clk = '1' AND reset = '1' AND set = '1') THEN q <= '1'; qbar <= '1';
          ELSE q <= q; qbar <= qbar;
          END IF;
   END PROCESS;
END comportamental;
```

Latch D

Bloco Lógico e Tabela Verdade



Entradas R e S nunca são iguais

D	S	R	Qf
∉	0	0	-
0	0	1	0
1	1	0	1
∉	1	1	-

D	Qf
0	0
1	1

VHDL – Código

Solução: Latch D

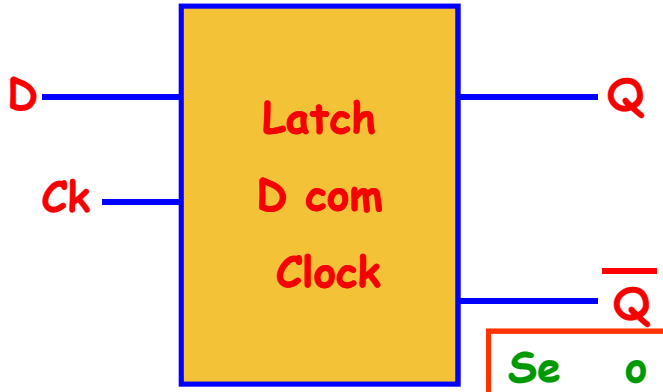
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY latchd IS
        PORT(d : IN BIT;          -- STD_LOGIC;
             q, qbar : BUFFER BIT);  -- STD_LOGIC;
END latchd;

ARCHITECTURE comportamental OF latchd IS
BEGIN
        PROCESS(d)
        BEGIN
                q <= d;
                qbar <= NOT q;
        END PROCESS;
END comportamental;
```

Latch D com Entrada Clock

Bloco Lógico e Tabela Verdade



Se o clock=1 ⇒ Latch funciona como um Latch D

Se o clock=0 ⇒ Latch permanece no seu estado anterior, mesmo que varie a entrada D

Ck=0 ⇒ Qf = Qa

D	S	R	Qf
∉	0	0	-
0	0	1	0
1	1	0	1
∉	1	1	-

D	Qf
0	0
1	1

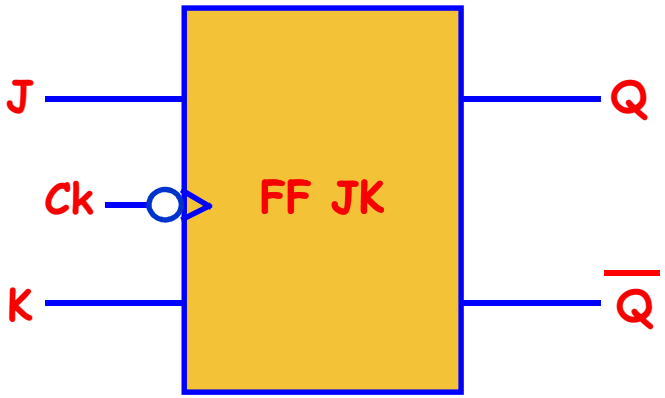
VHDL – Código

Solução: Latch D com Entrada Clock

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY latchd_clk IS
        PORT(clk, d : IN BIT;       -- STD_LOGIC;
             q, qbar : BUFFER BIT);  -- STD_LOGIC;
END latchd_clk;
ARCHITECTURE comportamental OF latchd_clk IS
BEGIN
        PROCESS(clk, d)
        BEGIN
                IF (clk = '1') THEN q <= d; qbar <= NOT d;
                ELSE q <= q; qbar <= qbar;
                END IF;
        END PROCESS;
END comportamental;
```

Flip-Flop JK Mestre-Escravo

Bloco Lógico e Tabela Verdade



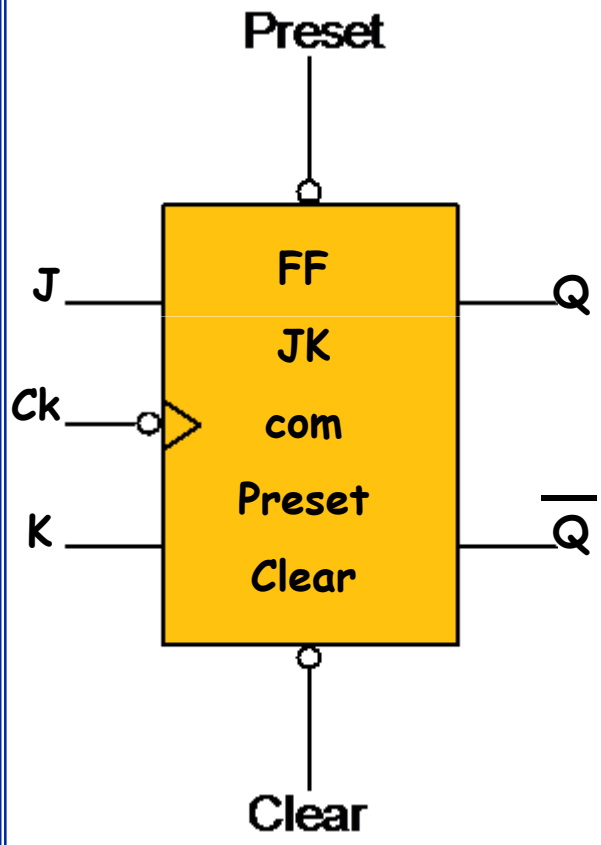
J	K	Qf
0	0	Qa
0	1	0
1	0	1
1	1	$\overline{Qa}$

VHDL – Código

Solução: Flip-Flop JK

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY flipflopjk IS
            PORT(clk, j, k : IN STD_LOGIC;          -- BIT; j = set e k = reset
                  q, qbar : BUFFER STD_LOGIC);     -- BIT;
END flipflopjk;
ARCHITECTURE comportamental OF flipflopjk IS
BEGIN
   PROCESS(clk, j, k)
   VARIABLE qv,qbarv : STD_LOGIC;     -- BIT;
   BEGIN
         -- IF (rising_edge(clk)) THEN              -- IF (clk'EVENT AND clk = '1') THEN
            IF (falling_edge(clk)) THEN             -- IF (clk'EVENT AND clk = '0') THEN
                     IF     (j = '1' AND k = '0') THEN qv := '1'; qbarv := NOT qv;
                     ELSIF (j = '0' AND k = '1') THEN qv := '0'; qbarv := NOT qv;
                     ELSIF (j = '1' AND k = '1') THEN qv := NOT qv; qbarv := NOT qv;
                     ELSE qv := qv; qbarv := NOT qv;
                     END IF;
            END IF;
   q <= qv; qbar <= qbarv;
   END PROCESS;
END comportamental;
```

Flip-Flop JK com Entradas Clear e Preset



Preset e Clear são entradas que operam independentemente das entradas de clock e de dados

1. $\overline{Preset} = \overline{Clear} = 0 \Rightarrow$ Entradas não podem ser usadas
2. $\overline{Preset} = 0$ e $\overline{Clear} = 1 \Rightarrow$ Q é "setada" (Q=1)
3. $\overline{Preset} = 1$ e $\overline{Clear} = 0 \Rightarrow$ Q é "resetada" (Q=0)
4. $\overline{Preset} = \overline{Clear} = 1 \Rightarrow$ FF responde às entradas J e K

$\overline{Preset}$	$\overline{Clear}$	Qf
0	0	X
0	1	1
1	0	0
1	1	FF JK

J	K	Qf
0	0	Qa
0	1	0
1	0	1
1	1	$\overline{Qa}$

VHDL – Código

Solução: Flip-Flop JK com Entradas Preset e Clear

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

ENTITY flipflopjk_clrpst IS
   PORT(pst, clr, clk, j, k : IN STD_LOGIC;                    -- BIT; j = set e k = reset
                    q, qbar : BUFFER STD_LOGIC);               -- BIT;
END flipflopjk_clrpst;

ARCHITECTURE comportamental OF flipflopjk_clrpst IS
BEGIN
   PROCESS(pst, clr, clk, j, k)
   VARIABLE qv,qbarv : STD_LOGIC;
   BEGIN
     IF (pst = '0' AND clr = '0') THEN qv := '1'; qbarv := '1'; -- qv <= '0'; -- qbarv <= '0';
     ELSIF (pst = '0' AND clr = '1') THEN qv := '1'; qbarv := NOT qv;
     ELSIF (pst = '1' AND clr = '0') THEN qv := '0'; qbarv := NOT qv;
```

VHDL – Código

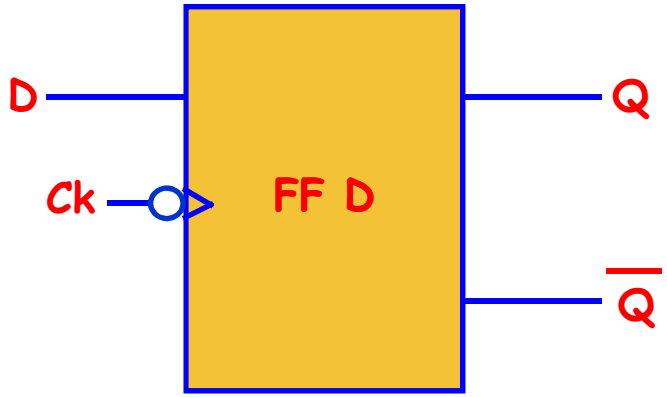
Solução: Flip-Flop JK com Entradas Preset e Clear

```
-- continuacao


        -- ELSIF (rising_edge(clk)) THEN
        -- ELSIF (clk'EVENT AND clk = '1') THEN
        ELSIF (pst = '1' AND clr = '1') AND (falling_edge(clk)) THEN
        -- ELSIF (clk'EVENT AND clk = '0') THEN
                IF     (j = '1' AND k = '0') THEN qv := '1'; qbarv := NOT qv;
                ELSIF (j = '0' AND k = '1') THEN qv := '0'; qbarv := NOT qv;
                ELSIF (j = '1' AND k = '1') THEN qv := NOT qv; qbarv := NOT qv;
                ELSE qv := qv; qbarv := NOT qv;
                END IF;
        END IF;
  q <= qv;
  qbar <= qbarv;
  END PROCESS;
END comportamental;
```

Flip-Flop D

Bloco Lógico e Tabela Verdade



Entradas J e K são sempre diferentes

J	K	D	Qf
0	0	∉	-
0	1	0	0
1	0	1	1
1	1	∉	-

D	Qf
0	0
1	1

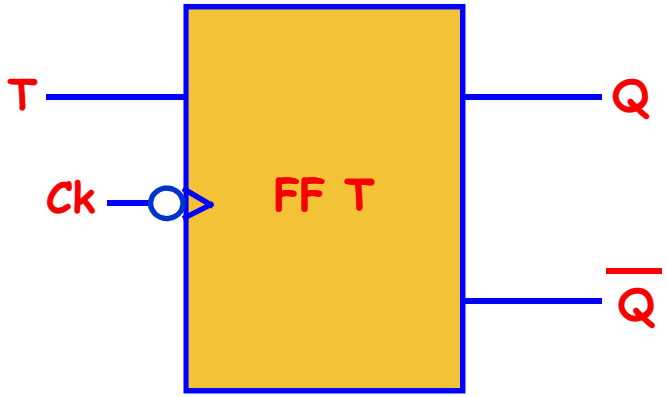
VHDL – Código

Solução: Flip-Flop D

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY flipflopd is
   PORT(clk, d : IN STD_LOGIC;                  -- BIT;
        q, qbar : BUFFER STD_LOGIC);            -- BIT;
END flipflopd;
ARCHITECTURE comportamental OF flipflopd IS
BEGIN
   PROCESS(clk, d)
   VARIABLE qv,qbarv : STD_LOGIC;
   BEGIN
            --IF (clk'EVENT AND clk = '0') THEN
            IF (rising_edge(clk)) THEN              -- IF (clk'EVENT AND clk = '1') THEN
            -- IF (falling_edge(clk)) THEN          -- IF (clk'EVENT AND clk = '0') THEN
                        qv := d;
                        qbarv := NOT qv;
            END IF;
            q <= qv;
            qbar <= NOT q;
   END PROCESS;
END comportamental;
```

Flip-Flop T

Bloco Lógico e Tabela Verdade



Entradas J e K são sempre iguais

J	K	T	Qf
0	0	0	Qa
0	1	∉	-
1	0	∉	-
1	1	1	$\overline{Qa}$

T	Qf
0	Qa
1	$\overline{Qa}$

VHDL – Código

Solução: Flip-Flop T

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY flipflopt IS
   PORT(clk, t : IN STD_LOGIC;              -- BIT; j = set e k = reset
        q, qbar : BUFFER STD_LOGIC);        -- BIT;
END flipflopt;
ARCHITECTURE comportamental OF flipflopt IS
BEGIN
   PROCESS(clk, t)
   VARIABLE qv,qbarv : STD_LOGIC;
   BEGIN
            IF (rising_edge(clk)) THEN                  -- ELSIF (clk'EVENT AND clk = '1') THEN
         -- ELSIF (falling_edge(clk)) THEN             -- ELSIF (clk'EVENT AND clk = '0') THEN
                        IF (t = '1') THEN qv := NOT qv; qbarv := NOT qv;
                        ELSE  qv := qv; qbarv := NOT qv;
                        END IF;
            END IF;
   q <= qv;
   qbar <= qbarv;
   END PROCESS;
END comportamental;
```