MC542

Organização de Computadores
Teoria e Prática

2006

Prof. Paulo Cesar Centoducatte

ducatte@ic.unicamp.br

www.ic.unicamp.br/~ducatte

MC542

Circuitos Lógicos

Projeto de Flip-Flops, Registradores, Contadores com VHDL

"Fundamentals of Digital Logic with VHDL Design" - (Capítulo 7)

Título do Capítulo Abordado
Sumário

- Memória Implicita
- FF D da Biblioteca
- Gated D latch
- Flip-Flop D
  - Usando Wait Until
  - Com reset Assíncrono
- Módulo lpm_shiftreg
- Registrador
  - 8 bits com Clear Assíncrono
  - N bits com Clear Assíncrono
- Flip-flop D com um mux 2:1 na entrada D

Memória Implicita

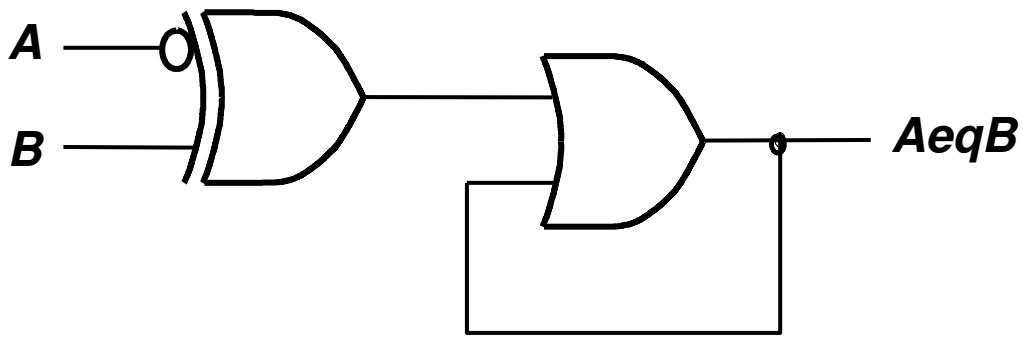
```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY implied IS
      PORT ( A, B      : IN      STD_LOGIC ;
             AeqB      : OUT     STD_LOGIC ) ;
END implied ;

ARCHITECTURE Behavior OF implied IS
BEGIN
      PROCESS ( A, B )
      BEGIN
          IF A = B THEN
              AeqB <= '1' ;
          END IF ;
      END PROCESS ;
END Behavior ;
```

Memória Implicita

```
...
PROCESS ( A, B )
BEGIN
     IF A = B THEN
          AeqB <= '1' ;
     END IF ;
END PROCESS ;
 ...
```



Instanciação de um FF D da Biblioteca

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
LIBRARY altera ;
USE altera.maxplus2.all ;

ENTITY flipflop IS
      PORT (  D, Clock              : IN     STD_LOGIC ;
              Resetn, Presetn       : IN     STD_LOGIC ;
              Q                     : OUT    STD_LOGIC ) ;
END flipflop ;

ARCHITECTURE Structure OF flipflop IS
BEGIN
      dff_instance: dff PORT MAP ( D, Clock, Resetn, Presetn, Q ) ;
END Structure ;
```

Gated D latch

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

ENTITY latch IS
      PORT (  D, Clk     : IN       STD_LOGIC ;
              Q          : OUT      STD_LOGIC) ;
END latch ;

ARCHITECTURE Behavior OF latch IS
BEGIN
      PROCESS ( D, Clk )
      BEGIN
            IF Clk = '1' THEN
                  Q <= D ;
            END IF ;
      END PROCESS ;
END Behavior ;
```

Flip-Flop D

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

ENTITY flipflop IS
      PORT (       D, Clock    : IN    STD_LOGIC ;
                   Q           : OUT STD_LOGIC) ;
END flipflop ;

ARCHITECTURE Behavior OF flipflop IS
BEGIN
      PROCESS ( Clock )
      BEGIN
          IF Clock'EVENT AND Clock = '1' THEN
                Q <= D ;
          END IF ;
      END PROCESS ;
END Behavior ;
```

Flip-Flop D Usando Wait Until

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

ENTITY flipflop IS
        PORT ( D, Clock : IN        STD_LOGIC ;
                  Q          : OUT     STD_LOGIC ) ;
END flipflop ;

ARCHITECTURE Behavior OF flipflop IS
BEGIN
      PROCESS
      BEGIN
            WAIT UNTIL Clock'EVENT AND Clock = '1' ;
            Q <= D ;
      END PROCESS ;
END Behavior ;
```

Flip-Flop D com reset Assíncrono

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

ENTITY flipflop IS
      PORT (  D, Resetn, Clock   : IN      STD_LOGIC ;
              Q                  : OUT     STD_LOGIC) ;
END flipflop ;

ARCHITECTURE Behavior OF flipflop IS
BEGIN
      PROCESS ( Resetn, Clock )
      BEGIN
            IF Resetn = '0' THEN
                  Q <= '0' ;
            ELSIF Clock'EVENT AND Clock = '1' THEN
                  Q <= D ;
            END IF ;
      END PROCESS ;
END Behavior ;
```

Flip-Flop D com Reset Síncrono

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

ENTITY flipflop IS
      PORT (  D, Resetn, Clock   : IN      STD_LOGIC ;
              Q                  : OUT     STD_LOGIC) ;
END flipflop ;

ARCHITECTURE Behavior OF flipflop IS
BEGIN
      PROCESS
      BEGIN
            WAIT UNTIL Clock'EVENT AND Clock = '1' ;
            IF Resetn = '0' THEN
                  Q <= '0' ;
            ELSE
                  Q <= D ;
            END IF ;
      END PROCESS ;
END Behavior ;
```

Módulo lpm_shiftreg

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

ENTITY shift IS
      PORT (  Clock          : IN     STD_LOGIC ;
              Reset          : IN     STD_LOGIC ;
              Shiftin, Load  : IN     STD_LOGIC ;
              R              : IN     STD_LOGIC_VECTOR(3 DOWNTO 0) ;
              Q              : OUT    STD_LOGIC_VECTOR(3 DOWNTO 0) ) ;
END shift ;

ARCHITECTURE Structure OF shift IS
BEGIN
      instance: lpm_shiftreg
            GENERIC MAP (LPM_WIDTH => 4, LPM_DIRECTION => "RIGHT")
            PORT MAP (data => R, clock => Clock, aclr => Reset,
                  load => Load, shiftin => Shiftin, q => Q ) ;
END Structure ;
```

Registrador de 8 bits com Clear Assíncrono

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

ENTITY reg8 IS
      PORT (  D                 : IN     STD_LOGIC_VECTOR(7 DOWNTO 0) ;
              Resetn, Clock     : IN     STD_LOGIC ;
              Q                 : OUT    STD_LOGIC_VECTOR(7 DOWNTO 0) ) ;
END reg8 ;

ARCHITECTURE Behavior OF reg8 IS
BEGIN
      PROCESS ( Resetn, Clock )
      BEGIN
            IF Resetn = '0' THEN
                  Q <= "00000000" ;
            ELSIF Clock'EVENT AND Clock = '1' THEN
                  Q <= D ;
            END IF ;
      END PROCESS ;
END Behavior ;
```

Registrador de N bits com Clear Assíncrono

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

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

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

Flip-flop D com um mux 2:1 na entrada D

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

ENTITY muxdff IS
    PORT ( D0, D1, Sel, Clock  : IN          STD_LOGIC ;
           Q                   : OUT  STD_LOGIC ) ;
END muxdff ;

ARCHITECTURE Behavior OF muxdff IS
BEGIN
    PROCESS
    BEGIN
        WAIT UNTIL Clock'EVENT AND Clock = '1' ;
        IF Sel = '0' THEN
            Q <= D0 ;
        ELSE
            Q <= D1 ;
        END IF ;
    END PROCESS ;
END Behavior ;
```

Shift Register
Usando muxdff como Componente

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

ENTITY shift4 IS
    PORT ( R           : IN      STD_LOGIC_VECTOR(3 DOWNTO 0);
           L, w, Clock : IN      STD_LOGIC ;
           Q           : BUFFER  STD_LOGIC_VECTOR(3 DOWNTO 0));
END shift4 ;

ARCHITECTURE Structure OF shift4 IS
    COMPONENT muxdff
        PORT ( D0, D1, Sel, Clock  : IN              STD_LOGIC ;
               Q                   : OUT  STD_LOGIC ) ;
    END COMPONENT ;
BEGIN
    Stage3: muxdff PORT MAP ( w, R(3), L, Clock, Q(3) ) ;
    Stage2: muxdff PORT MAP ( Q(3), R(2), L, Clock, Q(2) ) ;
    Stage1: muxdff PORT MAP ( Q(2), R(1), L, Clock, Q(1) ) ;
    Stage0: muxdff PORT MAP ( Q(1), R(0), L, Clock, Q(0) ) ;
END Structure ;
```

Shift Register
Código Alternativo

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY shift4 IS
    PORT ( R       : IN     STD_LOGIC_VECTOR(3 DOWNTO 0) ;
           Clock   : IN     STD_LOGIC ;
           L, w    : IN     STD_LOGIC ;
           Q       : BUFFER  STD_LOGIC_VECTOR(3 DOWNTO 0) ) ;
END shift4 ;

ARCHITECTURE Behavior OF shift4 IS
BEGIN
    PROCESS
    BEGIN
        WAIT UNTIL Clock'EVENT AND Clock = '1' ;
        IF L = '1' THEN
            Q <= R ;
        ELSE
            Q(0) <= Q(1) ;
            Q(1) <= Q(2);
            Q(2) <= Q(3) ;
            Q(3) <= w ;
        END IF ;
    END PROCESS ;
END Behavior ;
```

Shift Register de N bits

```
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) ;
           Clock   : IN     STD_LOGIC ;
           L, w            : IN     STD_LOGIC ;
           Q       : BUFFER  STD_LOGIC_VECTOR(N-1 DOWNTO 0) ) ;
END shiftn ;
ARCHITECTURE Behavior OF shiftn IS
BEGIN
    PROCESS
    BEGIN
        WAIT UNTIL Clock'EVENT AND Clock = '1' ;
        IF L = '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 Behavior ;
```

Contador de 4 Bits (Crescente)

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_unsigned.all ;
ENTITY upcount IS
    PORT ( Clock, Resetn, E : IN    STD_LOGIC ;
           Q                : OUT   STD_LOGIC_VECTOR (3 DOWNTO 0)) ;
END upcount ;

ARCHITECTURE Behavior OF upcount IS
    SIGNAL Count : STD_LOGIC_VECTOR (3 DOWNTO 0) ;
BEGIN
    PROCESS ( Clock, Resetn )
    BEGIN
        IF Resetn = '0' THEN
            Count <= "0000" ;
        ELSIF (Clock'EVENT AND Clock = '1') THEN
            IF E = '1' THEN
                Count <= Count + 1 ;
            ELSE
                Count <= Count ;
            END IF ;
        END IF ;
    END PROCESS ;
    Q <= Count ;
END Behavior ;
```

Contador Usando Sinal Tipo Integer

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

ENTITY upcount IS
    PORT (  R               : IN      INTEGER RANGE 0 TO 15 ;
            Clock, Resetn, L  : IN    STD_LOGIC ;
            Q               : BUFFER  INTEGER RANGE 0 TO 15 ) ;
END upcount ;

ARCHITECTURE Behavior OF upcount IS
BEGIN
    PROCESS ( Clock, Resetn )
    BEGIN
        IF Resetn = '0' THEN
            Q <= 0 ;
        ELSIF (Clock'EVENT AND Clock = '1') THEN
            IF L = '1' THEN
                Q <= R ;
            ELSE
                Q <= Q + 1 ;
            END IF;
        END IF;
    END PROCESS;
END Behavior;
```

Contador Decrescente

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY downcnt IS
    GENERIC ( modulus : INTEGER := 8 ) ;
    PORT (  Clock, L, E  : IN  STD_LOGIC ;
            Q            : OUT     INTEGER RANGE 0 TO modulus-1 ) ;
END downcnt ;

ARCHITECTURE Behavior OF downcnt IS
    SIGNAL Count : INTEGER RANGE 0 TO modulus-1 ;
BEGIN
    PROCESS
    BEGIN
        WAIT UNTIL (Clock'EVENT AND Clock = '1') ;
        IF E = '1' THEN
            IF L = '1' THEN
                Count <= modulus-1 ;
            ELSE
                Count <= Count-1 ;
            END IF ;
        END IF ;
    END PROCESS;
    Q <= Count ;
END Behavior ;
```

Registrador de N bits

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

ENTITY regn IS
    GENERIC ( N : INTEGER := 8 ) ;
    PORT ( R           : IN   STD_LOGIC_VECTOR(N-1 DOWNTO 0) ;
           Rin, Clock  : IN   STD_LOGIC ;
           Q           : OUT  STD_LOGIC_VECTOR(N-1 DOWNTO 0) ) ;
END regn ;

ARCHITECTURE Behavior OF regn IS
BEGIN
    PROCESS
    BEGIN
        WAIT UNTIL Clock'EVENT AND Clock = '1' ;
        IF Rin = '1' THEN
            Q <= R ;
        END IF ;
    END PROCESS ;
END Behavior ;
```

Registrador N bits com Saída Tri-state

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

ENTITY trin IS
    GENERIC ( N : INTEGER := 8 ) ;
    PORT ( X   : IN    STD_LOGIC_VECTOR(N-1 DOWNTO 0) ;
           E   : IN    STD_LOGIC ;
           F   : OUT   STD_LOGIC_VECTOR(N-1 DOWNTO 0) ) ;
END trin ;

ARCHITECTURE Behavior OF trin IS
BEGIN
    F <= (OTHERS => 'Z') WHEN E = '0' ELSE X ;
END Behavior ;
```

Decclaração de Um Package

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

PACKAGE components IS
    COMPONENT regn -- register
        GENERIC ( N : INTEGER := 8 ) ;
        PORT (  R              : IN     STD_LOGIC_VECTOR(N-1 DOWNTO 0) ;
                Rin, Clock     : IN     STD_LOGIC ;
                Q              : OUT    STD_LOGIC_VECTOR(N-1 DOWNTO 0) ) ;
    END COMPONENT ;

    COMPONENT shiftr -- left-to-right shift register with async reset
        GENERIC ( K : INTEGER := 4 ) ;
        PORT    (  Resetn, Clock, w    : IN          STD_LOGIC ;
                   Q                   : BUFFER      STD_LOGIC_VECTOR(1 TO K) ) ;
    END component ;

    COMPONENT trin -- tri-state buffers
        GENERIC ( N : INTEGER := 8 ) ;
        PORT (  X   : IN    STD_LOGIC_VECTOR(N-1 DOWNTO 0) ;
                E   : IN    STD_LOGIC ;
                F   : OUT       STD_LOGIC_VECTOR(N-1 DOWNTO 0) ) ;
    END COMPONENT ;
END components ;
```