COL215 Digital Logic and System Design

VHDL

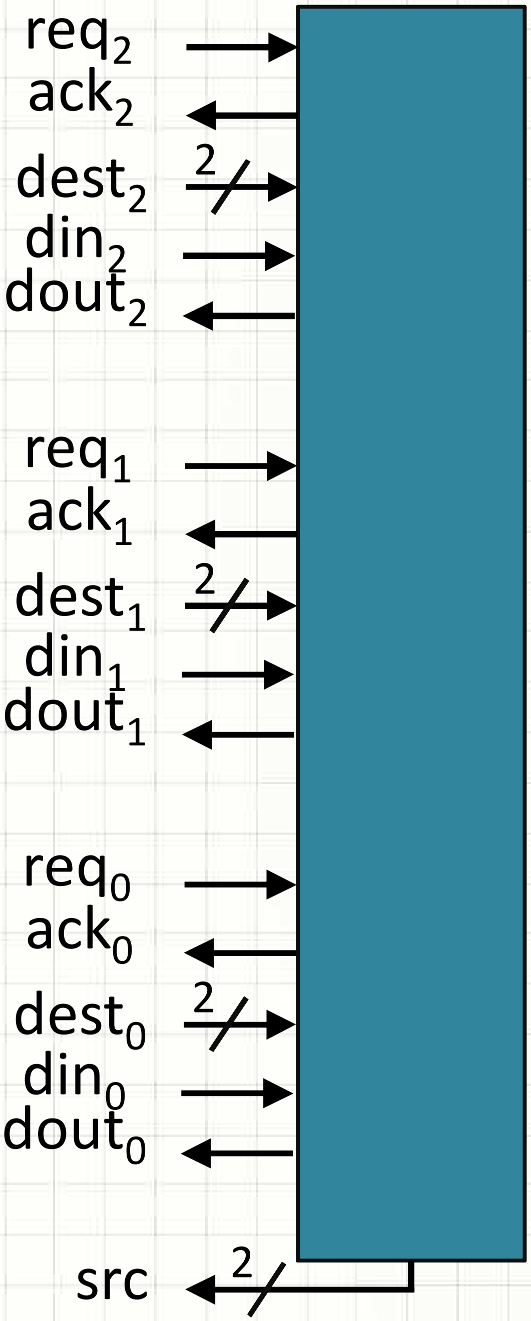
– switch example

– sequential circuits

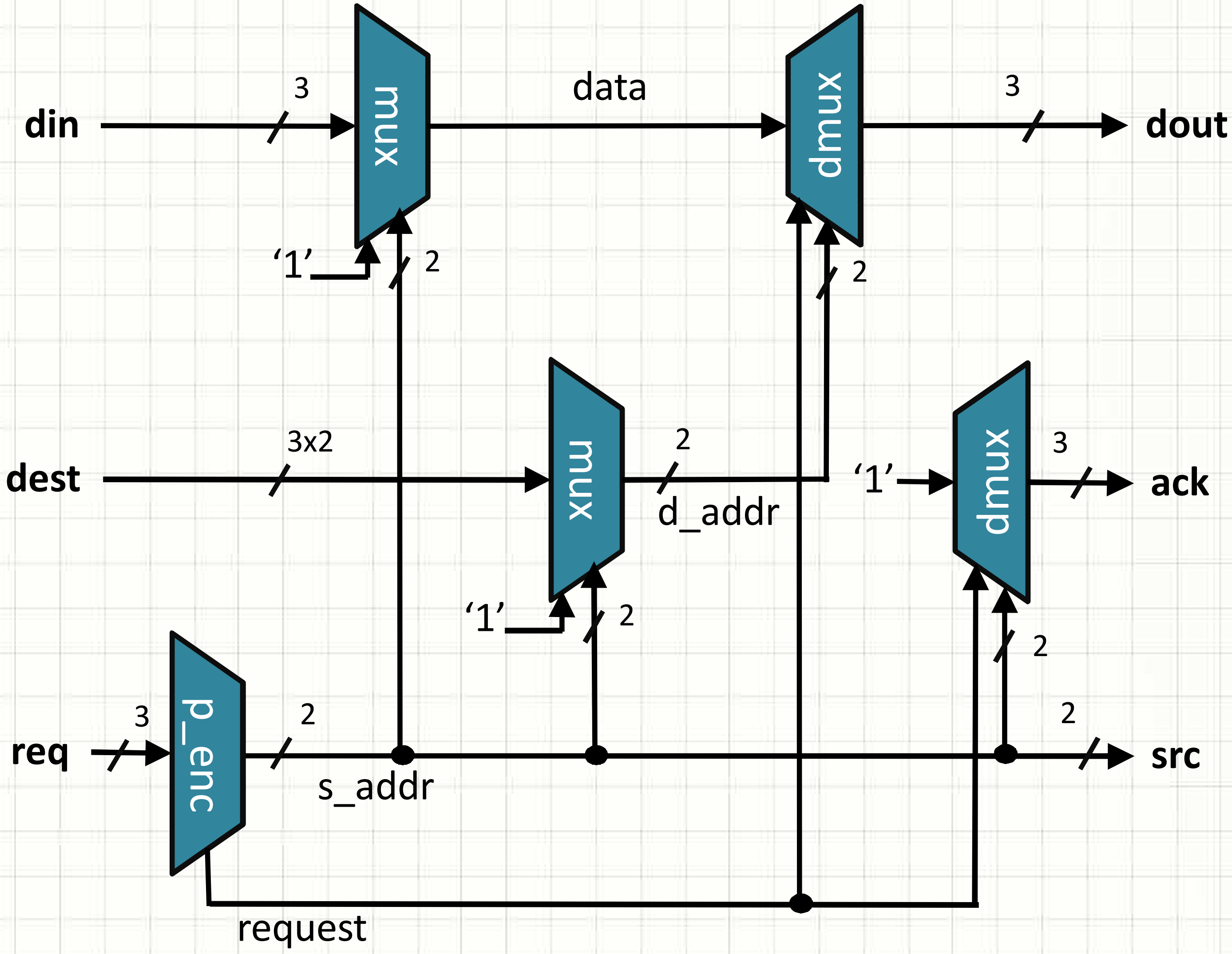
22 August 2017

3-Port Switch

```
ENTITY Switch3 IS
  PORT (
      req : IN bit_vector (2 downto 0);
      ack : OUT bit_vector (2 downto 0);
      din : IN bit_vector (2 downto 0);
      dout : OUT bit_vector (2 downto 0);
      dest2 : IN bit_vector (1 downto 0);
      dest1 : IN bit_vector (1 downto 0);
      dest0 : IN bit_vector (1 downto 0);
      src : OUT bit_vector (1 downto 0)
  );
END Switch3;
```



3-Port Switch Design



VHDL statements learned so far

Concurrent

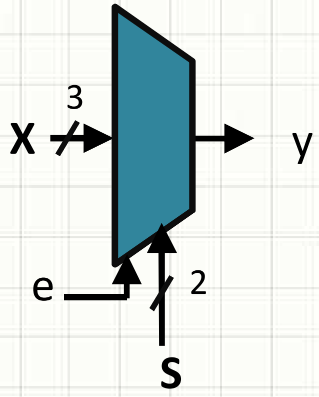
- Concurrent signal assignment
- Selected signal assignment
- Conditional signal assignment
- Process statement
- Component instantiation statement

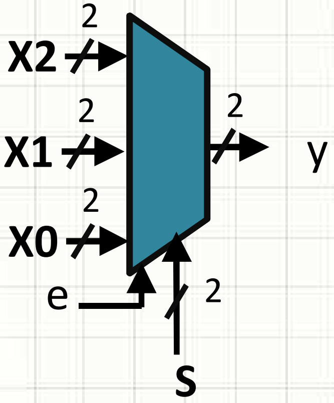
← To be introduced today

Sequential

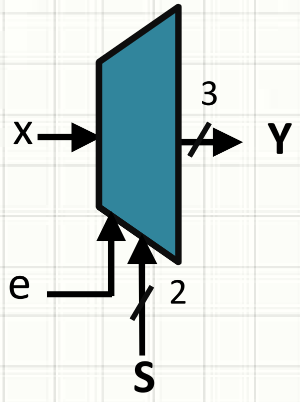
- [Sequential] signal assignment
- Case statement
- If statement

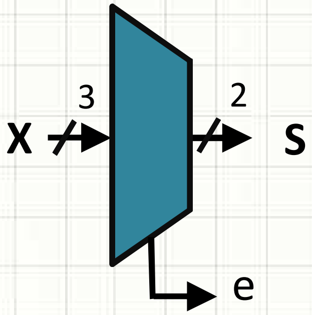
```
ENTITY mux_3_1 IS
  PORT (X: IN bit_vector (2 downto 0);
            S: IN bit_vector (1 downto 0);
            e: IN bit;
            y: OUT bit
           );
END mux_3_1;
ENTITY mux_3_1_2bit IS
  PORT (X2: IN bit_vector (1 downto 0);
            X1: IN bit_vector (1 downto 0);
            X0: IN bit_vector (1 downto 0);
            S: IN bit_vector (1 downto 0);
            e: IN bit;
            y: OUT bit_vector (1 downto 0)
           );
END mux_3_1_2bit;
```





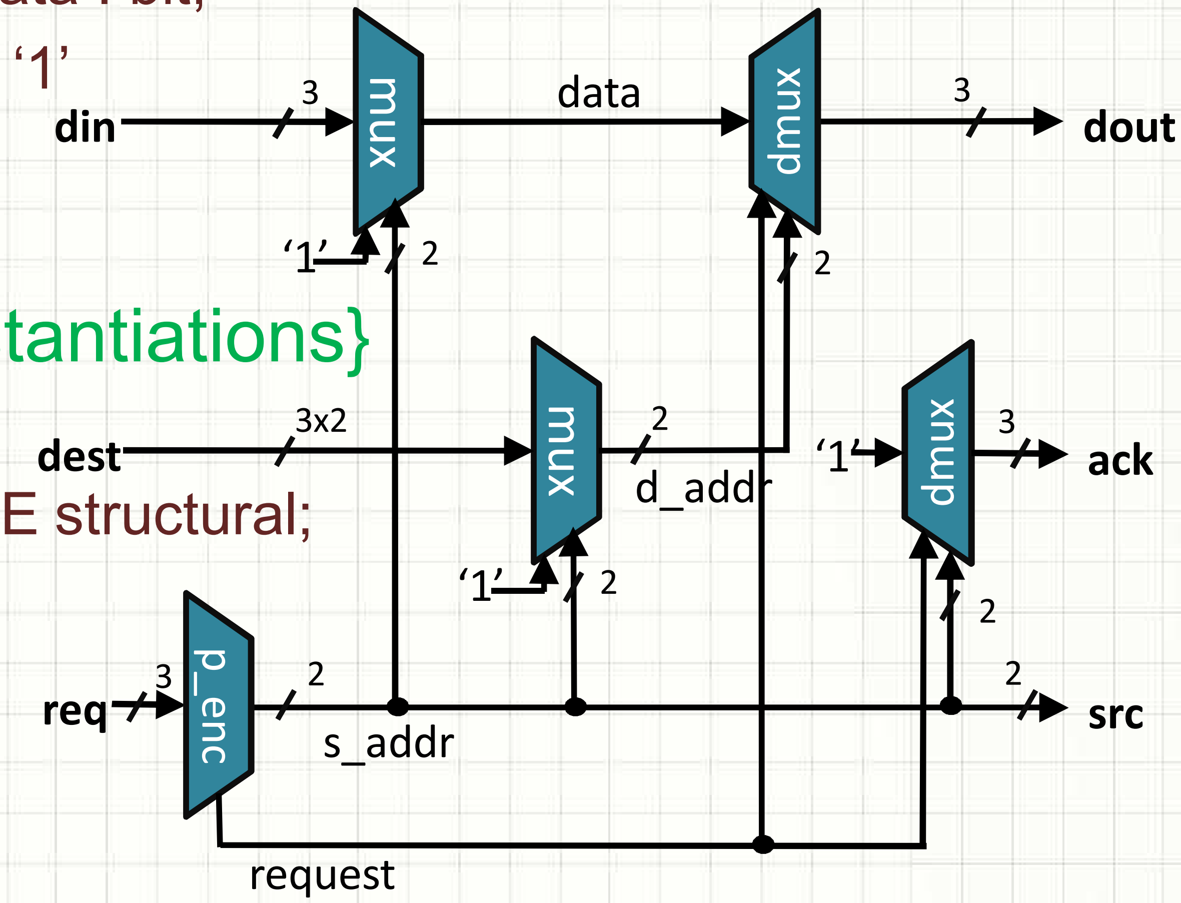
```
ENTITY de-mux_1_3 IS
  PORT (x: IN    bit;
        e: IN    bit;
        S: IN    bit_vector (1 downto 0);
        Y: OUT bit_vector (2 downto 0)
       );
END de-mux_1_3;
ENTITY Priority_3 IS
  PORT (X: IN     bit_vector (2 downto 0);
        S: OUT  bit_vector (1 downto 0);
        e:  OUT  bit
       );
END Priority_3;
```





3-Port Switch

```
ARCHITECTURE structural OF Switch3 IS
BEGIN
    SIGNAL s_addr, d_addr : bit_vector (1 downto 0);
    SIGNAL request, data : bit;
    SIGNAL one :bit := ‘1’
    src <= s_addr;

   {component instantiations}

END ARCHITECTURE structural;
```



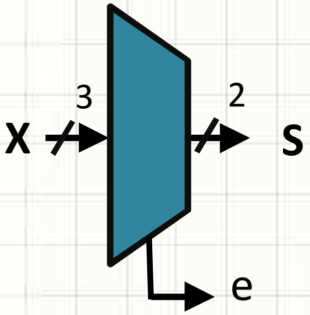
Component instantiations

```
Priority_encoder:
   ENTITY WORK.Priority_3 (cond)
   PORT MAP (req, s_addr, request);
```

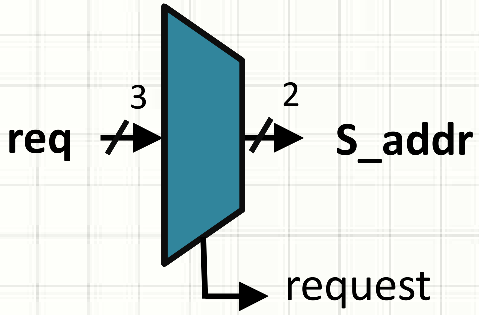
positional association

```
Priority_encoder:
   ENTITY WORK.Priority_3 (cond)
   PORT MAP (
      x => req,
      s => s_addr,
      e => request
   );
```

named association



definition



instance

Component instantiations

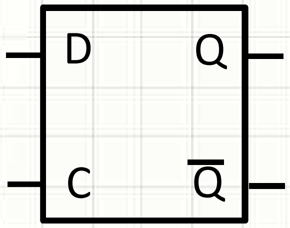
```
Priority_encoder: ENTITY WORK.Priority_3 (cond)
        PORT MAP (req, s_addr, request);

Data_mux: ENTITY WORK.mux_3_1 (ssa)
        PORT MAP (din, s_addr, one, data);

Addr_mux: ENTITY WORK.mux_3_1_2bit (ssa)
        PORT MAP (dest2, dest1, dest0, s_addr, d_addr);

Data_dmux: ENTITY WORK.de-mux_1_3 (ssa)
        PORT MAP (data, request, d_addr, dout);

Ack_dmux: ENTITY WORK.de-mux_1_3 (ssa)
        PORT MAP (one, request, s_addr, ack);
```

3-Port Switch

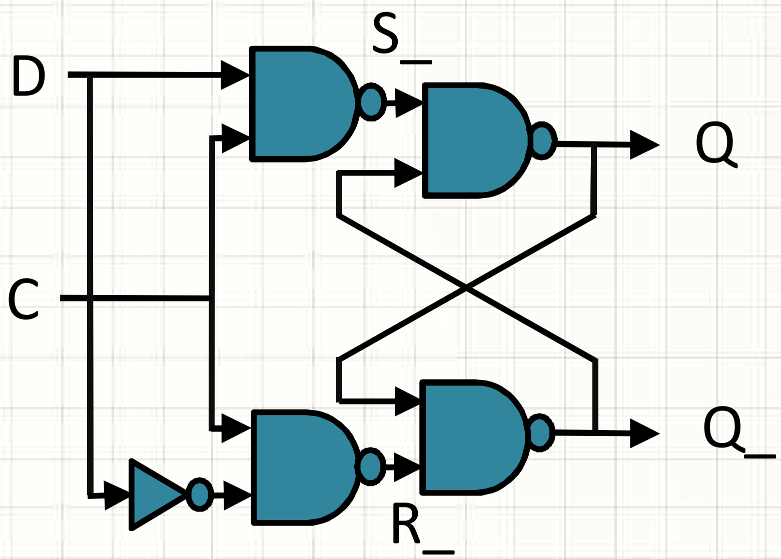
```
ARCHITECTURE combined OF Switch3 IS
BEGIN
   SIGNAL s_addr, d_addr : bit_vector (1 downto 0);
   SIGNAL request, data : bit;

   Priority_encoder_process;

   Data_mux_process;

   Addr_mux_process;

   Data_dmux_process;

   Ack_dmux_process;

   src <= s_addr;
END ARCHITECTURE combined;
```

SEQUENTIAL CIRCUITS

VHDL description of D Latch

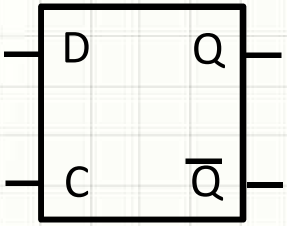


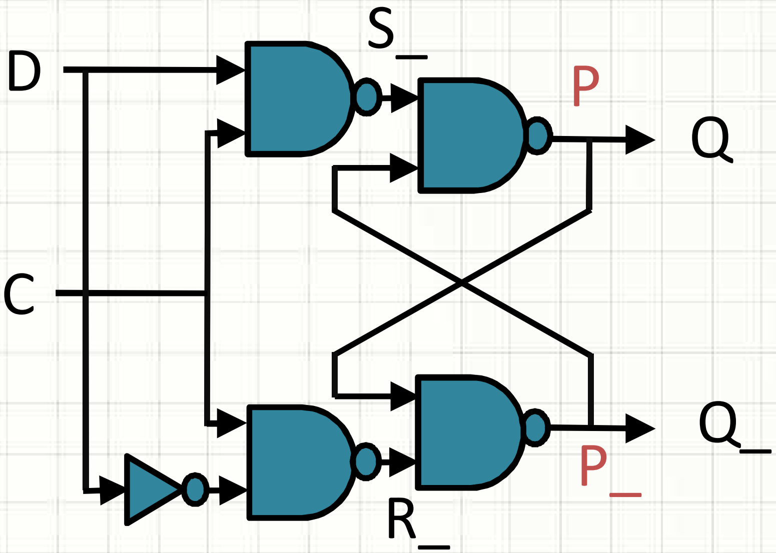
```
ENTITY latch IS
   PORT (D, C: IN BIT; Q, Q_ : OUT BIT);
END latch;

ARCHITECTURE asynch OF latch IS
   SIGNAL S_, R_            : BIT;
BEGIN
   S_ <= C NAND D;
   R_ <= C NAND (NOT D);
   Q   <= S_ NAND Q_ ;
   Q_ <= R_ NAND Q   ;


END asynch;
```



VHDL description of D Latch

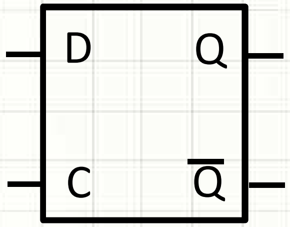
```
ENTITY latch IS
   PORT (D, C: IN BIT; Q, Q_ : OUT BIT);
END latch;

ARCHITECTURE asynch OF latch IS
   SIGNAL S_, R_, P, P_ : BIT;
BEGIN
   S_ <= C NAND D;
   R_ <= C NAND (NOT D);
   P   <= S_ NAND P_ ;
   P_ <= R_ NAND P   ;
   Q   <= P;
   Q_ <= P_;
END asynch;
```





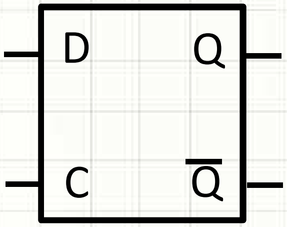
Conditional signal assignment

```
ENTITY latch IS
   PORT (D, C: IN BIT; Q, Q_ : OUT BIT);
END latch;

ARCHITECTURE csa OF latch IS
   SIGNAL P : BIT;
BEGIN
   P   <= D WHEN C =  ‘1’  ELSE P;
   Q   <= P;
   Q_ <= NOT P;
END csa;
```



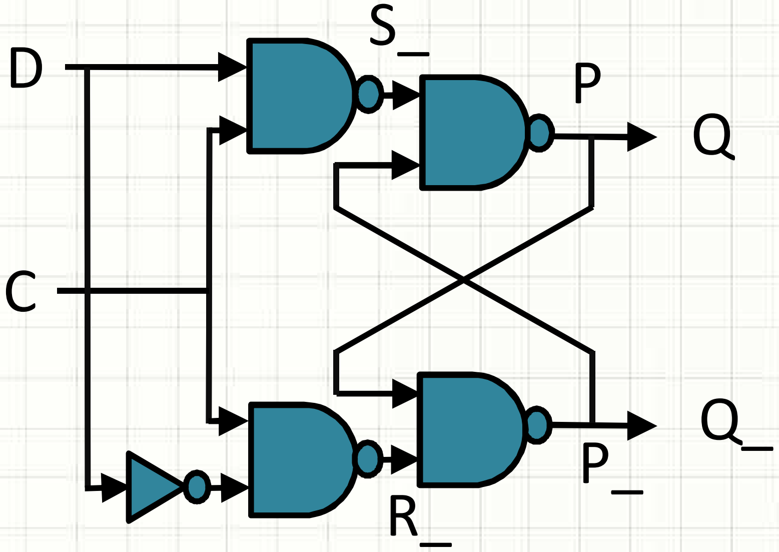
Conditional signal assignment

```
ENTITY latch IS
   PORT (D, C: IN BIT; Q, Q_ : OUT BIT);
END latch;

ARCHITECTURE csa OF latch IS
   SIGNAL P : BIT;
BEGIN
   P   <= D WHEN C =  '1'  ELSE P;
   Q   <= P;
   Q_  <= NOT P;
END csa;
```



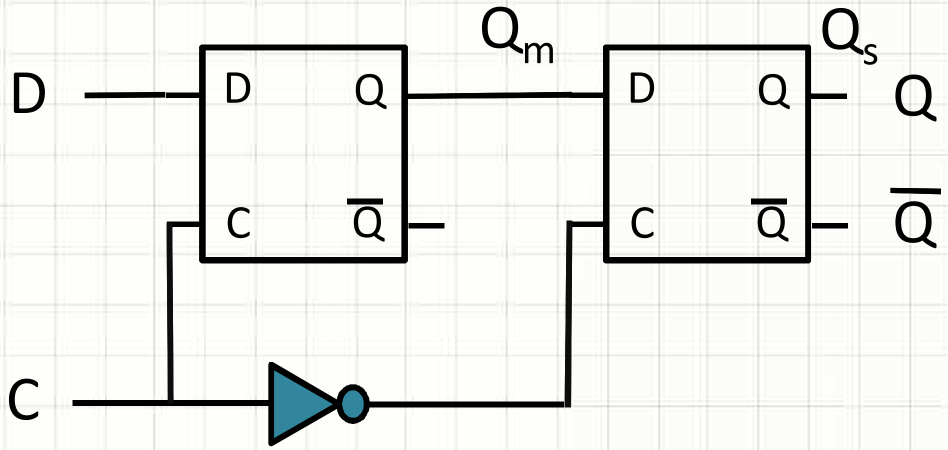
Two descriptions of D Latch

```
ENTITY latch IS
   PORT (D, C: IN BIT;
               Q, Q_ : OUT BIT);
END latch;

ARCHITECTURE asynch
   OF latch IS
   SIGNAL S_, R_, P, P_ : BIT;
BEGIN
   S_ <= C NAND D;
   R_ <= C NAND (NOT D);
   P   <= S_ NAND P_ ;
   P_ <= R_ NAND P    ;
   Q   <= P;
   Q_ <= P_;
END asynch;
```

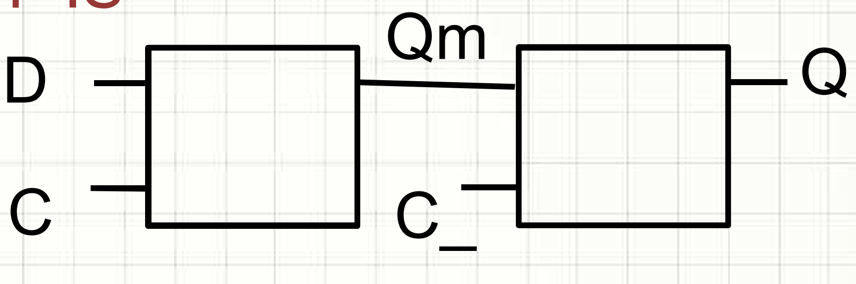


```
ARCHITECTURE csa
   OF latch IS
   SIGNAL P : BIT;
BEGIN
   P   <= D WHEN C =  ‘1’ ;

   Q   <= P;
   Q_ <= NOT P;
END csa;
```

Master-Slave D Flip-Flop



```
ENTITY MSFF IS
   PORT (D, C: IN BIT; Q : OUT BIT);
END MSFF;

ARCHITECTURE dual OF MSFF IS
   SIGNAL Qm : BIT;
BEGIN
   Qm    <= D WHEN C =  '1' ;
   Q    <= Qm WHEN C =  '0' ;
END dual;
```



Clocked circuits

Check for

‘1’ level

```
PROCESS (clk)
  BEGIN
    IF clk = '1’  THEN
    ….. first time on ‘1’ level, later on rising edge
END PROCESS;
```

Check for

Rising edge

```
PROCESS (clk)
  BEGIN
    IF clk = '1' AND clk'EVENT THEN
    ….. only on rising edge
END PROCESS;
```

D Flip-flop with Set/Reset

```
ENTITY DFFsr IS
   PORT (d, clk, s, r: IN BIT;
               q               : OUT BIT);
END DFFsr;
```

D Flip-flop with synch S/R

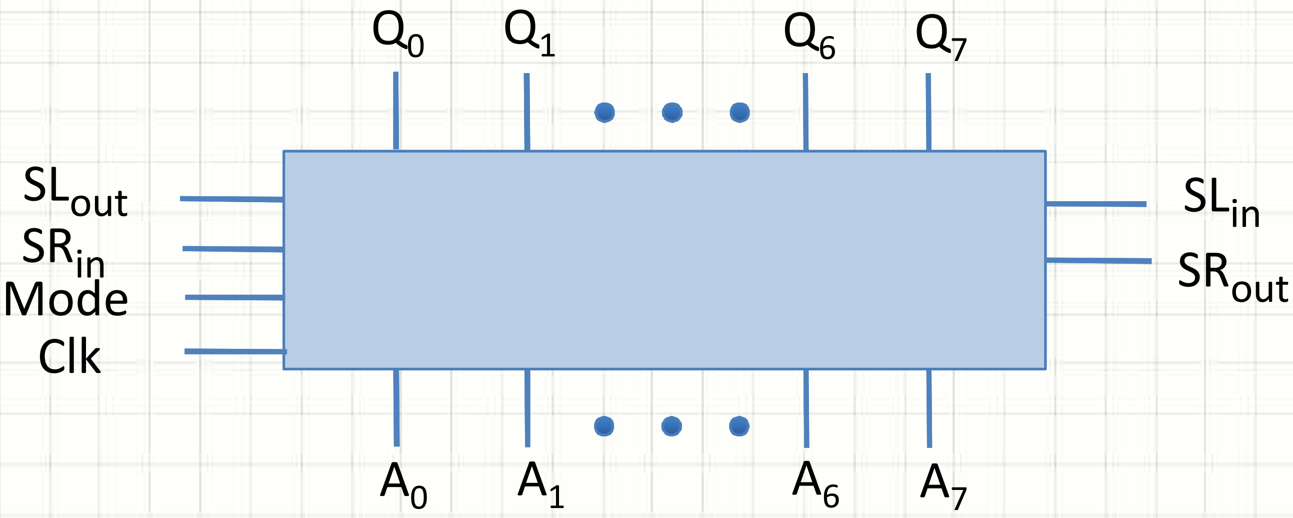
```
ARCHITECTURE synchronous OF DFFsr IS
BEGIN
  PROCESS (clk)
  BEGIN
    IF clk = '1' AND clk'EVENT THEN
      IF s = '1' THEN       q <= '1’ ;
      ELSIF r = '1' THEN    q <= '0’ ;
      ELSE                  q <= d;
      END IF;
    END IF;
  END PROCESS;
END ARCHITECTURE synchronous;
```

D Flip-flop with asynch S/R

```
ARCHITECTURE asynchronous OF DFFsr IS
BEGIN
  PROCESS (clk, s, r)
  BEGIN
    IF s = '1' THEN                              q <= '1’ ;
    ELSIF r = '1' THEN                           q <= '0’ ;
    ELSIF clk = '1' AND clk'EVENT THEN   q <= d;
    END IF;
  END PROCESS;
END ARCHITECTURE asynchronous;
```

Multi-mode Register

```
ENTITY Reg8 IS
  PORT (SLin, SRin, Clk : IN bit; SLout, SRout OUT bit;
                A : IN bit_vector (0 TO 7);
                Mode : IN bit_vector (1 downto 0);
                 Q : OUT bit_vector (0 TO 7)
               );
END ENTITY;
```



Mode	Operation
0 0	No change
0 1	Shift right
1 0	Shift left
1 1	Parallel load

Multi-mode Register

```
ARCHITECTURE beh OF Reg8 IS
  SIGNAL t bit_vector (0 TO 7) := "00000000”;
BEGIN
  PROCESS (Clk) BEGIN
    IF (Clk = ’1' AND clk'EVENT) THEN
      CASE Mode IS
        WHEN "00" => t <= t;
        WHEN "01" => t <= SRin & t (0 TO 6);
        WHEN "10" => t <= t (1 TO 7) & SLin;
        WHEN "11" => t <= A;
      END CASE;
    END IF;
  END PROCESS;
  Q <= t;
  SLout <= Q(0);
  SRout <= Q(7);
END ARCHITECTURE beh;
```

Counter

```
ENTITY counter4 IS
  PORT (
            reset, clk : IN bit_logic;
            count : OUT bit_vector (3 downto 0)
           );
END ENTITY;
```

Counter

```
ARCHITECTURE procedural OF counter4 IS
  SIGNAL t : bit_vector (3 downto 0);
BEGIN
  PROCESS (clk) BEGIN
    IF (clk = '0' AND clk'EVENT) THEN
      IF (reset = '1') THEN t <= "0000”;
      ELSE t <= t + 1;
      END IF;
    END IF;
  END PROCESS;
  count <= t;
END ARCHITECTURE procedural;
```

THANKS