```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY myand IS
PORT ( A, B : IN STD_LOGIC;
         C : OUT STD_LOGIC
        );
END myand;
ARCHITECTURE dataflow OF myand IS
BEGIN
           C <= A and B;
END dataflow;
```

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY fa IS
PORT ( A, B, Cin : IN STD_LOGIC;
        Cout, S : OUT STD_LOGIC
      );
END fa;
ARCHITECTURE dataflow OF fa IS
signal M: std_logic;
BEGIN
      M <= A xor B;
      S <= M xor Cin;
      Cout <= ( M and Cin ) or ( A and B );
END dataflow;
```

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY rca4 IS
Port ( A, B: in std_logic_vector(3 downto 0);
       Cin: in std_logic;
       Cout: out std_logic;
       S: out std_logic_vector(3 downto 0));
END rca4;
ARCHITECTURE structural OF rca4 IS
component fulladder
PORT (A, B, Cin : IN STD_LOGIC;
         Cout, S : OUT STD_LOGIC
       );
end component;
signal CR: STD_LOGIC_VECTOR(3 downto 0);
BEGIN
U0: fulladder port map ( A => A(0), B => B(0),
                         Cin => Cin,
                        Cout=> CR(0), S => S(0)
                        );
U1: fulladder port map ( A => A(1), B => B(1),
                         Cin => CR(0),
                         Cout=> CR(1), S => S(1)
                        );
U2: fulladder port map ( A => A(2), B => B(2),
                         Cin => CR(1),
                        Cout=> CR(2), S => S(2)
                        );
U3: fulladder port map ( A => A(3), B => B(3),
                         Cin => CR(2),
                         Cout=> CR(3), S => S(3)
                          );
END structural;
```

```
--Unsigned 8x4-bit Multiplier
module multiplier(A, B, RES);
input [7:0] A;
input [3:0] B;
output [11:0] RES;
assign RES = A * B;
endmodule
```

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY myand_tb IS
END myand_tb;
ARCHITECTURE beh OF myand_tb IS
component myand
PORT( A, B : IN STD_LOGIC;
         C : OUT STD_LOGIC   );
end component;
signal TA, TB : STD_LOGIC;
signal TC: STD_LOGIC;
BEGIN
                 uut: myand port map (
                 A => TA,   B => TB,
                 C => TC  );
                 Process
                 Begin
                   TA <='0'; TB<='0';
                   Wait for 10 ns;
                   TA <='0'; TB<='1';
                   Wait for 10 ns;
                   TA <='1'; TB<='0';
                   Wait for 10 ns;
                   TA <='1'; TB<='1';
                   Wait for 10 ns;
                    Wait;
                 End process;
END beh;
```

```
Library ieee;
Use ieee.std_logic_1164.all;
Entity dff is
Port ( d, clk: in std_logic;
         q: out std_logic );
End dff;
Architecture beh of dff is
begin
  process( clk )
  begin
     -- if (rising_edge (clk) ) then
        q <= d;
     end if;
  end process;
End beh;
```

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY dff_tb IS
END dff_tb;
ARCHITECTURE beh OF dff_tb IS
component dff
PORT  (
           d, clk : IN STD_LOGIC;
           q : OUT STD_LOGIC
          );
end component;
signal d, clk, q : STD_LOGIC;
BEGIN
uut: dff port map ( d => d,
              clk => clk, q => q );
  Process
  Begin
    clk <= '0'; Wait for 10 ns;
    Clk <= '1'; Wait for 10 ns;
  End process;
  Process
  Begin
    d <='0'; Wait for 8 ns;
    d <='1'; Wait for 20 ns;
   d <='0'; Wait for 8 ns;
   Wait;
 End process;
```

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY shiftreg IS
PORT ( Din, clk, clr : IN STD_LOGIC;
        Q : OUT STD_LOGIC_VECTOR(3 downto 0)
        );
END shiftreg;
ARCHITECTURE beh OF shiftreg IS
signal W: std_logic_vector(3 downto 0);
BEGIN
  process( clk, clr )
  begin
    if (clr = '1') then
      W <= "0000";
    elsif (rising_edge (clk)) then
      W(3) <= Din;
      W(2) <= W(3);
      W(1) <= W(2);
      W(0) <= W(1);
    end if;
  end process;
  Q <= W;
END beh;
```

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY mux IS
PORT( A, B,C,D : IN STD_LOGIC;
      SEL: IN STD_LOGIC_VECTOR(1 downto 0);
       DOUT : OUT STD_LOGIC
     );
END mux;
ARCHITECTURE design OF mux IS
BEGIN
DOUT <= A WHEN SEL = "00" ELSE
          B WHEN SEL = "01" ELSE
         C WHEN SEL = "10" ELSE
         D;
END design;
```

```
/library ieee;
use ieee.std_logic_1164.all;
package MY_PACK is
 function PARITY (X : std_logic_vector)
            return std_logic;
end MY_PACK;
package body MY_PACK is
 function PARITY (X : std_logic_vector)
            return std_logic is
 variable TMP : std_logic;
 begin
    --TMP:= '0';
    TMP := X(0);
    --for J in X'range loop
    for J in 1 to X'high loop
       TMP := TMP xor X(J);
     end loop;      -- works for any size X
     return TMP;
  end PARITY;
end MY_PACK;
```

```
library ieee;
use ieee.std_logic_1164.all;
use work.MY_PACK.all;

entity PAR is
  port( db:  in std_logic_vector(7 downto 0);
        dw: in std_logic_vector(15 downto 0);
        pb, pw: out std_logic);
end PAR;

architecture ARCH1 of PAR is
begin
 pb <= PARITY(db);
 pw <= PARITY(dw);
end ARCH1;
```

Even Parity and Odd Parity Examples

even

ab_parity (f_even)

00_0

01_1

10_1

11_0

f_even = a xor b

odd

ab_parity (f_odd)

00_1

01_0

10_0

11_1

f_odd = not (a xor b);

ADD/SUBTRACT

Modulo 2 uses XOR

0 ± 0 = 0;

0 ± 1 = 1;

1 ± 0 = 1;

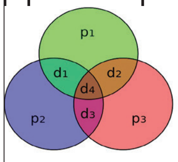
1 ± 1 = 0

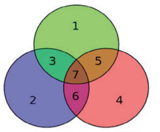
MULTIPLICATION

```
   1 0 1 1
 x 0 1 0 1
   1 0 1 1
  0 0 0 0
 1 0 1 1
0 0 0 0
0 1 0 0 1 1 1
```

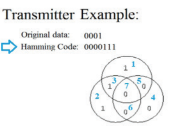
DIVISION

```
               1 0 0 0 1 rem. 1 0 1
1 0 0 1 1|1 0 0 1 0 0 1 1 0
          1 0 0 1 1
                1 0 1 1 0
                1 0 0 1 1
                    1 0 1


             1 rem. 1 0 1 0
1 1 0 0 1|1 0 0 1 1
          1 1 0 0 1
            1 0 1 0


             1 rem.1 0 1 0
1 0 0 1 1|1 1 0 0 1
          1 0 0 1 1
            1 0 1 0
```



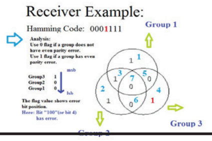


Bit position of the data and parity bits


Transmitter Example:

Original data: 0001

Hamming Code: 0000111


Receiver Example:

Hamming Code: 0001111

Analysis:
Use 0 flag if a group does not have even parity error.
Use 1 flag if a group has even parity error.

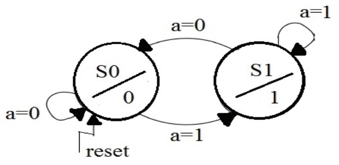
Group3 1
Group2 0
Group1 0

The flag value shows error bit position.
Here: Bit "100"(or bit 4) has error.

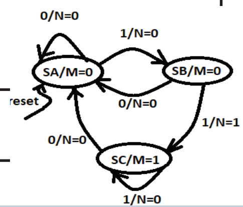
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity hamming is
 Port ( hamdin: in std_logic_vector(3 downto 0);
       hamout: out std_logic_vector(7 downto 1) );
end hamming;
architecture Behavioral of hamming is
signal p: std_logic_vector(4 downto 1);
begin
p(1) <= hamdin(3) xor hamdin(1) xor hamdin(0);
p(2) <= hamdin(3) xor hamdin(2) xor hamdin(0);
p(4) <= hamdin(3) xor hamdin(2) xor hamdin(1);
hamout <= p(1) & p(2) & hamdin(3) & p(4) & hamdin(2) & hamdin(1) & hamdin(0);
end Behavioral;
```

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY moorefsm IS
PORT
        ( reset, a, clk : IN STD_LOGIC;
          y: OUT STD_LOGIC
         );
END moorefsm;
ARCHITECTURE beh OF moorefsm IS
type state_type is (S0, S1);
signal cs, ns: state_type;
BEGIN
   Process(reset, clk)
   Begin
      If (reset = '1') then
         cs <= S0;
      elsif (rising_edge(clk)) then
         cs <= ns;
      end if;
   End process;
   Process(cs, a)
   Begin
      Case (cs) is
         When S0 => y <= ‘0’;
            If (a=’1’) then
               ns <= S1;
            else
               ns <= S0;
            end if;
         When S1 => y <= ‘1’;
            If (a=’1’) then
               ns <= S1;
            else
               ns <= S0;
            end if;
        When others=> y<= ‘0’; ns <= S0;
      end case;
   end process;
END beh;
```

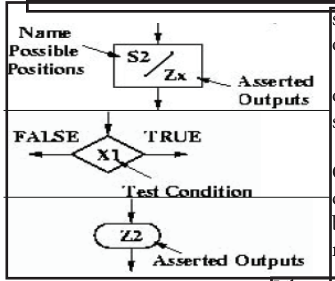
```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY moorefsm_tb IS
END moorefsm_tb;

ARCHITECTURE beh OF moorefsm_tb IS
component moorefsm
PORT (
        reset, a, clk : IN STD_LOGIC;
        y: OUT STD_LOGIC
       );
end component;
signal reset, a, clk : STD_LOGIC;
signal y: STD_LOGIC;
BEGIN
   Uut: moorefsm port map ( reset => reset,
                            a => a, clk => clk,
                            y => y );
   reset <= '1', '0' after 5 ns, '0' after 100 ns;
   clk<= ‘0’, ‘1’ after 10 ns, ‘0’ after 20 ns ,
‘   0’ after 30 ns, ‘1’ after 40 ns,
   ‘0’ after 50 ns, ‘1’ after 60 ns,
   ‘0’ after 70 ns, ‘1’ after 80 ns,
‘   0’ after 90 ns, ‘1’ after 100 ns;
   a <= ‘0’, ‘1’ after 12 ns, ‘0’ after 56 ns;
END beh;
```



```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY mealyfsm IS PORT
   ( reset, a, clk : IN STD_LOGIC;
     y:          OUT STD_LOGIC
   );
END mealyfsm;

ARCHITECTURE beh OF mealyfsm IS
type  state_type is (SO,Sl);
signal   cs, ns: state_type;
BEGIN
   Process(reset, elk)
   Begin
      If(reset = '1') then
         cs <= SO;
      elsif (rising_edge(clk)) then
         cs <= ns;
      end if;
   End process;
   Process(cs, a)
   Begin
      Case (cs) is
         When S0 => If (a=’1’) then
                       ns <= S1; y <= ‘1’;
                    else
                       ns <= S0; y <= ‘0’;
                    end if;
         When S1 => If (a=’1’) then
                       ns <= S1; y <= ‘0’;
                    else
                       ns <= S0; y <= ‘1’;
                    end if;
         When others=> y<= ‘0’;  ns <= S0;

      end case;
   end process;
END beh;
```



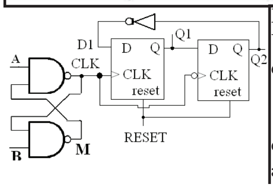
```
library IEEE;
use IEEE.std_logic_1164.all;
entity fsm is
port ( clk, reset, x1 : IN std_logic;
        outp : OUT std_logic
       );
end entity;
architecture beh1 of fsm is
type state_type is (s1,s2,s3,s4);
signal state, next_state: state_type;
   begin
   process1: process (clk, reset)
      begin
         if (reset ='1') then
            state <= s1;
         elsif (clk = '1' and clk'Event) then
            state <= next_state;
         end if;
   end process process1;
   process2 : process (state, x1)
   begin
      case state is
         when s1 =>
                     if x1='1' then
                        next_state <= s2;
                     else
                        next_state <= s3;
                     end if;
         when s2 => next_state <= s4;
         when s3 => next_state <= s4;
         when s4 => next_state <= s1;
      end case;
   end process process2;
   process3 : process (state)
   begin
      case state is
         when s1 => outp <= ‘1’;
         when s2 => outp <= ‘1’;
         when s3 => outp <= ‘0’;
         when s4 => outp <= ‘0’;
      end case;
   end process process3;
end beh1;
```



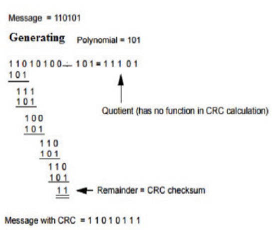
state box- with outputs

optional decision boxes

Optional conditional output box asynchronous/Mealy



5. CRC Error Detection



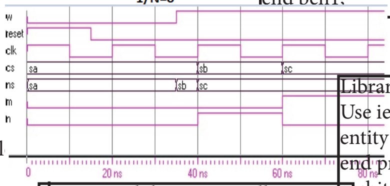
```
Message = 110101
Generating   Polynomial = 101
11010100÷ 101=11101
101
 111
 101
  100
  101
    110
    101
     110
     101
      11  ← Remainder = CRC checksum
```

Quotient (has no function in CRC calculation)

Message with CRC = 11010111

```
Library ieee;
Use ieee.std_logic_1164.all;
entity prob1 is
  port (a, b, reset: in std_logic;
                q1, q2:  out  std_logic );
end prob1;
architecture circuit of prob1 is
signal       clk,  m, d1, t1, t2: std_logic;
begin
   clk <= a  nand m;
   m  <=  b  nand clk;
   d1 <= not t2;
process ( clk , reset)
begin
   if (reset= ‘1’) then t1 <= ‘0’;
   elsif (rising_edge(clk)) then
      t1 <= d1;
   end if;
end process;
process (  clk , reset)
begin
   if (reset= ‘1’) then t2 <= ‘0’;
   elsif (falling_edge(clk)) then
         t2 <= t1;
   end if;
end process;
q1 <= t1;  q2 <= t2;
end circuit;
```

```
Library ieee; --tb for that circuit
Use ieee.std_logic_1164.all;
entity prob1_tb
end prob1_tb;
architecture beh of prob1_tb is
signal a, b, reset:   std_logic;
signal    q1, q2:     std_logic;
component prob1  (a, b, reset: in std_l
                 q1, q2:  out  std_logic );
end component;
begin
uut: prob1 port map ( a=> a, b=>b,
           reset=>reset, q1=>q1, q2=>q2 );
process
begin
   reset <= ‘1’;
   wait for 5 ns;
   reset <= ‘0’;
   wait for 100 ns;
   wait;
end process;
b <= not a;
 process
 begin
   a <= ‘1’;  wait for 10 ns;
   a <= ‘0’;  wait for 5 ns;
   a <= ‘1’;  wait for 5 ns;
   a <= ‘0’;  wait for 5 ns;
   a <= ‘1’;  wait for 5 ns;
   a <= ‘0’;  wait for 5 ns;
   a <= ‘1’;  wait for 5 ns;
   a <= ‘0’;  wait for 5 ns;
   wait;
end process;
end beh;
```



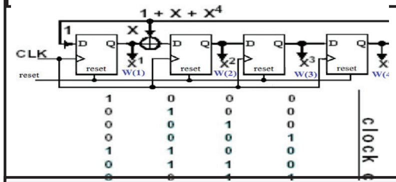
```
Use ieee.std_logic_1164.all;
entity prob2_fsm is
  port( w, Reset, Clk: in std_logic;
                 M, N: out std_logic );
End prob2_fsm;
Architecture beh of prob2_fsm is
type  st is ( SA, SB, SC );
signal  cs, ns:  st;
begin
 process ( w ,  cs )
 begin
  case (cs)
    when SA=>  if (w = ‘0’) then
                  ns <= SA ;
                else
                  ns  <= SB ;
                 end if;
    when SB=>  if(w = ‘0’) then
                  ns  <= SA ;
                else
                  ns <= SC ;
               end if;
     when SC=>  if (w = ‘0’) then
                  ns <= SA ;
                else
                  ns <= SC ;
                   end if;
    when others=> ns <= SA;
  end case;
end process;
```

```
Library ieee;
Use ieee.std_logic_1164.all;
entity prob2_fsm_tb
end prob2_fsm_tb;
architecture beh of prob2_fsm_tb is
signal   w, Reset, Clk:  std_logic;
signal                M, N::    std_logic;
component prob2_fsm ( w, Reset,
                        Clk: in std_logic;
                  M, N: out std_logic );
end component;
begin
uut: prob2_fsm port map ( w,
                  Reset, Clk,  M, N);
reset <= ‘1’, ‘0’ after 15 ns;
w      <= ‘0’, ‘1’ after 35 ns;
process
begin
    Clk <= ‘1’; wait for 10 ns;
    Clk <= ‘0’; wait for 10 ns;
end process;
```

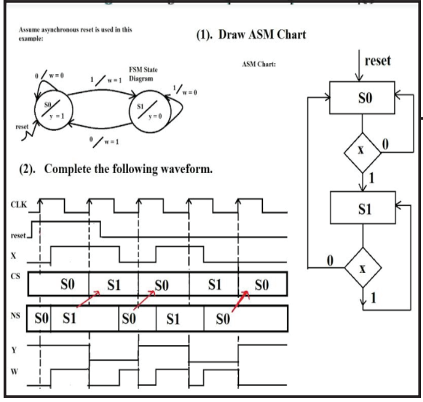
```
process (  Clk ,    Reset)
begin
   if( Reset = ‘1’) then
              cs <= SA ;
  elsif (rising_edge(Clk)) then
              cs <= ns ;
   end if;
 end process;
  M <= ‘1’ when (cs=SC)  else  ‘0’;
  N <= w when (cs=SB)  else ‘0’;
end beh;
```

```vhdl
LIBRARY IEEE; --LFSR
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY lfsr IS
PORT (  reset, clk: IN STD_LOGIC;
           Q : OUT STD_LOGIC_VECTOR(4 downto 1)
          );
END lfsr;
ARCHITECTURE beh OF lfsr IS
signal W: std_logic_vector(4 downto 1);
BEGIN
   process( clk, reset )
   begin
      if (reset='1') then
         W <= ( 1=>'1', others => '0' );
      elsif (rising_edge (clk)) then
         W <= W(3 downto 2) & ( W(1) xor W(4) ) & W(4);
      end if;
   end process;
   Q <= W;
END beh;
```

```vhdl
LIBRARY IEEE; --lfsr tb
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY lfsr_tb is
End lfsr_tb;
ARCHITECTURE beh OF lfsr_tb IS
Component lfsr
PORT (  reset, clk: IN STD_LOGIC;
          Q : OUT STD_LOGIC_VECTOR(4 downto 1)
         );
END Component;
signal reset, clk: std_logic;
signal Q: std_logic_vector(4 downto 1);
BEGIN
uut: lfsr port map( reset=>reset, clk=>clk, Q=>Q );
   Process
   Begin
      Clk <= '0';
      Wait for 10 ns;
      Clk <= '1';
      Wait for 10 ns;
   End process;
   Process
   Begin
      reset <=’1’;
      Wait for 6 ns;
      reset <=’0’;
      Wait for 40 ns;
      Wait;
   End process;
END beh;
```



1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
1	1	0	0
0	1	1	0



```vhdl
library ieee; --Clock division
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity display_counter_vhd is
Port( clk: in std_logic;
        load: in std_logic;
        updown: in std_logic;
        din: in std_logic_vector (3 downto 0);
        cntout: out std_logic_vector (3 downto 0);
        led: out std_logic_vector (3 downto 0);
        clkout: out std_logic
       );
end display_counter_vhd;
architecture behavioral of display_counter_vhd is
signal cnt_div: std_logic_vector (9 downto 0);
signal cnt: std_logic_vector (3 downto 0);
signal clk2: std_logic;
begin
process(clk)
begin
if rising_edge (clk) then
  if(cnt_div = 999) then --(N-1), here N=1000
    cnt_div <= (others => '0');
    clk2 <= '1';
  elsif (cnt_div < 499) then --(N/2  - 1)
    cnt_div <= cnt_div + 1;
    clk2 <= '1';
  else
    cnt_div <= cnt_div + 1;
    clk2 <= '0';
  end if;
end if;
end process;
process (clk2, load, updown)
--process active on event of clk2,load, or updown
begin
   if(load = '1') then
      cnt <= din;
   elsif rising_edge (clk2) then
      if(updown = '1') then
         cnt <= cnt + 1;
      else
         cnt <= cnt - 1;
      end if;
   end if;
end process;

cnt_out <= cnt;  --output count
clkout <= clk2; --output clk
led <= cnt;  --led output value
end behavioral;
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;
entity mytt is
port( A, B: in std_logic_vector(5 downto 0);
        Y: out std_logic_vector(11 downto 0));
end mytt;
architecture arch_tt of mytt is
constant C: std_logic_vector(2 downto 0) := "100";
begin
process(A,B)
begin
   Y(2 downto 0) <= A(2 downto 0);
   Y(3) <= A(3) and B(3);
   Y(5 downto 4) <= (A(5)and B(5))&(A(4) or B(4));
   Y(8 downto 6) <= B(2 downto 0);
   Y(11 downto 9)<= C;
end process;
end arch_tt;
```

```vhdl
library ieee; --pos edg dff
use ieee.std_logic_1164.all;
entity flop is
port( CLK, D : in std_logic;
         Q : out std_logic
       );
end flop;
architecture archi of flop is
begin
process (CLK)
begin
if (rising_edge(CLK)) then
   Q <= D;
end if;
end process;
end archi;
```

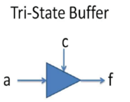
```vhdl
library ieee; --n.edg dff asy rst
use ieee.std_logic_1164.all;
entity flop is
port( C, D, CLR: in std_logic;
         Q : out std_logic
       );
end flop;
architecture archi of flop is
begin
process (C, CLR)
begin
if (CLR = '1')then
   Q <= '0';
elsif (falling_edge(CLK))
then
   Q <= D;
end if;
end process;
end archi;
```

```vhdl
library ieee; --p.edg sync set
use ieee.std_logic_1164.all;
entity flop is
port( C, D, S : in std_logic;
         Q : out std_logic );
end flop;
architecture archi of flop is
begin
process (C)
begin
   if (C'event and C='1') then
      if (S='1') then
         Q <= '1';
      else
         Q <= D;
      end if;
   end if;
end process;
end archi;
```

```vhdl
library ieee; --latch
use ieee.std_logic_1164.all;
entity latch is
port( G, D : in std_logic;
         Q : out std_logic
       );
end latch;
architecture archi of latch is
begin
process (G, D)
begin
   if (G='1') then
      Q <= D;
   end if;
end process;
end archi;
```

```vhdl
library ieee; --4b p.edg clk, asy set, clk en
use ieee.std_logic_1164.all;
entity flop is
port( C, CE, PRE : in std_logic;
         D: in std_logic_vector(3 downto 0);
         Q : out std_logic_vector (3 downto 0)
       );
end flop;
architecture archi of flop is
begin
process (C, PRE)
begin
   if (PRE=’1’) then
      Q <= “1111”;
   elsif (C’event and C=’1’)then
      if (CE=’1’) then
         Q <= D;
      end if;
   end if;
end process;
end archi;
```

```vhdl
library ieee; --tristate
use ieee.std_logic_1164.all;
entity three_st is
port( T, I : in std_logic;
         O : out std_logic
       );
end three_st;
architecture archi of three_st is
begin
process (I, T)
begin
   if (T='0') then
      O <= I;
   else
      O <= 'Z';
   end if;
end process;
end archi;
```



c	a	f
0	0	Z
0	1	Z
1	0	0
1	1	1

```
library ieee; --4b  up/down counter, asy. reset
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
entity counter is
port( C, CLR, UP_DOWN : in std_logic;
        Q: out std_logic_vector(3 downto 0)
       );
end counter;
architecture archi of counter is
signal tmp: std_logic_vector(3 downto 0);
begin
process (C, CLR)
begin
   if (CLR='1') then
      tmp <= "0000";
   elsif (C'event and C='1') then
      if (UP_DOWN='1') then
         tmp <= tmp + 1;
      else
         tmp <= tmp - 1;
      end if;
   end if;
end process;
Q <= tmp;
end archi;
```

```
library ieee; --8b Sft L Reg, P.Edg Clk, S.i/O
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
entity shift is
port( C, SI : in std_logic;
        SO : out std_logic
       );
end shift;
architecture archi of shift is
signal tmp: std_logic_vector(7 downto 0);
begin
process (C)
begin
   if (C'event and C='1') then
      for i in 0 to 6 loop
         tmp(i+1) <= tmp(i);
      end loop;
      tmp(0) <= SI;
   end if;
end process;
SO <= tmp(7);
end archi;
```

```
library ieee; --unsigned 8b adder/subtractor
use ieee.std_l ogic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
entity addsub is
port( A, B : in std_logic_vector(7 downto 0);
        OPER : in std_logic;
        RES : out std_logic_vector(7 downto 0) );
end addsub;
architecture archi of addsub is
begin
   RES <= A + B when OPER='0'
   else A - B;
end archi;
```

```
library ieee; --Concatenation
use ieee.std_logic_1164.all;
entity mytt2 is
port( A, B: in std_logic_vector(2 downto 0);
Y: out std_logic_vector(14 downto 0));
end mytt2;
architecture arch_tt of mytt2 is
constant C: std_logic_vector(2 downto 0) := "100";
begin
Y <= A & B & C & C & "110";
end arch_tt;
```

```
library ieee; -- 8b Sft-L Reg, Pedg Clk, Asy Clr, S I/O
use ieee.std_logic_1164.all;
entity shift is
port( C, SI, CLR : in std_logic;
        SO : out std_logic
       );
end shift;
architecture archi of shift is
signal tmp: std_logic_vector(7 downto 0);
begin
process (C, CLR)
begin
   if (CLR='1') then
      tmp <= (others => '0');
   elsif (C'event and C='1') then
      tmp <= tmp(6 downto 0) & SI;
   end if;
end process;
SO <= tmp(7);
end archi;
```

```
library ieee; --8b Sft L Reg, PEdg Clk, Asy Load, S I/O
use ieee.std_logic_1164.all;
entity shift is
port( C, SI, ALOAD : in std_logic;
        D : in std_logic_vector(7 downto 0);
        SO : out std_logic
       );
end shift;
architecture archi of shift is
signal tmp: std_logic_vector(7 downto 0);
begin
process (C, ALOAD, D)
begin
   if (ALOAD='1') then
      tmp <= D;
   elsif (C'event and C='1') then
      tmp <= tmp(6 downto 0) & SI;
   end if;
end process;
SO <= tmp(7);
end archi;
```

```
library ieee; --4-1 mux if statement
use ieee.std_logic_1164.all;
entity mux is
port ( a, b, c, d : in std_logic;
         s : in std_logic_vector (1 downto 0);
         o : out std_logic );
end mux;
architecture archi of mux is
begin
process (a, b, c, d, s)
begin
   if (s = "00") then
      o <= a;
   elsif (s = "01") then
      o <= b;
   elsif (s = "10") then
      o <= c;
   else
      o <= d;
   end if;
end process;
end archi;
```

```
library ieee; --comparator
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity compar is
port( A, B : in std_logic_vector(7 downto 0);
        CMP : out std_logic
);
end compar;
architecture archi of compar is
begin
   CMP <= ‘1’ when A >= B else ‘0’;
end archi;
```

```
library ieee; --decoder
use ieee.std_logic_1164.all;
entity dec is
port (
        sel: in std_logic_vector (2 downto 0);
        res: out std_logic_vector (7 downto 0)
       );
end dec;
architecture archi of dec is
begin
   res <= “00000001” when sel = “000”
   else “00000010” when sel = “001”
   else “00000100” when sel = “010”
   else “00001000” when sel = “011”
   else “00010000” when sel = “100”
   else “00100000” when sel = “101”
   else “01000000” when sel = “110”
   else “10000000”;
end archi;
```

```
library ieee; --decoder
use ieee.std_logic_1164.all;
entity dec is
port (
        sel: in std_logic_vector (2 downto 0);
        res: out std_logic_vector (7 downto 0)
       );
end dec;
architecture archi of dec is
begin
with sel select
   res <= "00000001" when "000",
             "00000010" when "001",
             "00000100" when "010",
             "00001000" when "011",
             "00010000" when "100",
             "00100000" when "101",
             "01000000" when "110"
             "10000000" when others;
end archi;
```

```
library ieee;
use ieee.std_logic_1164.all;
entity mux is
port ( a, b, c, d : in std_logic;
         s : in std_logic_vector (1 downto 0);
         o : out std_logic
       );
end mux;
architecture archi of mux is
begin
process (a, b, c, d, s)
begin
   case s is
      when "00" => o <= a;
      when "01" => o <= b;
      when "10" => o <= c;
      when others => o <= d;
   end case;
end process;
end archi;
```

```
library ieee;
use ieee.std_logic_1164.all;
entity my_block is
port( I1 : in std_logic; I2 : in std_logic;
        O : out std_logic );
end my_block;
architecture arch1 of my_block is
. . .
end arch1;

library ieee;
use ieee.std_logic_1164.all;
entity top is
port( DI_1, DI_2, DI_3, DI_4 : in std_logic;
        DOUT1, DOUT2 : out std_logic
);
end top;
architecture top_arch of top is
component my_block
port ( I1 : in std_logic; I2 : in std_logic;
        O : out std_logic
       );
end component;
begin
inst1: my_block port map ( I1=>DI_1,
                      I2=>DI_2, O=>DOUT1
                      );
Inst2: my_block port map ( DI_3, DI_4,
DOUT2 );
end top;
```

```
library ieee; --priority encoder
use ieee.std_logic_1164.all;
entity priority is
port ( sel : in std_logic_vector (7 downto 0);
         code :out std_logic_vector (2 downto 0)
       );
end priority;
architecture archi of priority is
begin
code <= "000" when sel(0) = '1' else
           "001" when sel(1) = '1' else
           "010" when sel(2) = '1' else
           "011" when sel(3) = '1' else
           "100" when sel(4) = '1' else
           "101" when sel(5) = '1' else
           “110” when sel(6) = ‘1’ else
           “111” when sel(7) = ‘1’ else
           “000”;
end archi;
```

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity adder is
port( A, B : in std_logic_vector(7 downto 0);
        SUM : out std_logic_vector(7 downto 0);
        CO : out std_logic );
end adder;
architecture archi of adder is
signal tmp: std_logic_vector(8 downto 0);
begin
 tmp <= conv_std_logic_vector((conv_integer(A) +
                                          conv_integer(B)),9);
 SUM <= tmp(7 downto 0);
 CO <= tmp(8);
end archi;
-- Or: Res <= ("0" & A) + ("0" & B);
```