DIGITAL LOGIC WITH VHDL DESIGN

Dr Tin Thet Nwe

Fig 6.38 Mux 2 to 1 using if-statement

```
library ieee ;
use ieee.std logic 1164.all ;
entity   mux2to1 is
port (                 w0, w1, s : in std logic ;
                       f : out std logic ) ;
end mux2to1 ;
architecture  behavior  of   mux2to1 is
begin
   process (w0, w1, s )
   begin
   if s='0' then
       f<=w0 ;
   else
        f<=w1 ;
   end if ;
   end process ;
end behavior ;
```

mux 2 to 1 using if-statement fig 6.39

```
LIBRARY ieee ;
USE ieee.std logic 1164.all ;
ENTITY mux2to1 IS
PORT (          w0, w1, s : IN STD LOGIC ;
                f : OUT STD LOGIC ) ;
END   mux2to1 ;
ARCHITECTURE   Behavior   OF      mux2to1 IS
BEGIN
    PROCESS (w0, w1, s )
        BEGIN
            f<=w0;
        IF s='1' THEN
            f<=w1 ;
        END IF ;
    END PROCESS ;
END Behavior ;
```

mux 2 to 1 using case statement Fig 6.45

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY  mux2to1 IS
PORT (w0, w1, s : IN STD LOGIC ;
f : OUT STD LOGIC ) ;
END         mux2to1 ;
ARCHITECTURE       Behavior      OF       mux2to1 IS
BEGIN
PROCESS (w0, w1, s )
    BEGIN
    CASE s IS
             WHEN ’0’  =>
                          f<= w0 ;
             WHEN  OTHERS  =>
                          f<= w1 ;
    END   CASE ;
END   PROCESS ;
END   Behavior ;
```

4 to 1 multiplexer using if-statement

```
LIBRARY IEEE ;
USE IEEE.STD LOGIC 1164.ALL ;
ENTITYMUX4TO1 IS
PORT (              a:IN STD_LOGIC_VECTOR(  3 DOWNTO 0);
                     S : IN STD LOGIC ;
                    F : OUT STD LOGIC ) ;
END MUX4TO1 ;
ARCHITECTURE  BEHAVIOR  OF   MUX4TO1 IS
BEGIN
    PROCESS (S)
        BEGIN
            IF S="00" THEN
                    F<= a(0) ;
            ELSIF  S = "01" THEN
                    F<= a(1)  ;
            ELSIF  S = "10" THEN
                    F<= a(2)  ;
            ELSE
                    F <= a(3);
            END IF ;
    END PROCESS ;       END BEHAVIOR ;
```

home work

- 4 to 1 multiplexer using case-statement

4-bit comparator for unsigned numbers

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
USE IEEE.STD_LOGIC_unsigned.ALL ;
ENTITY MUX4TO1 IS
PORT (    A,B:IN STD_LOGIC_VECTOR(  3 DOWNTO 0);
               AeqB,AgtB,AltB: OUT STD LOGIC ) ;
END MUX4TO1 ;
ARCHITECTURE  BEHAVIOR  OF   MUX4TO1 IS
BEGIN
AeqB <= ‘1’ when  A=B else ‘0’;
AgtB <= ‘1’ when  A<B else ‘0’;
AltB <= ‘1’ when  A>B else ‘0’;

END BEHAVIOR ;
```

4-bit comparator for signed numbers

```
library ieee ;
use ieee.std_logic_1164.all ;
use ieee.std_logic_arith.all;
entity mux4to1 is
port (       a,b:in signed (  3 downto 0);
              aeqb,agtb,altb: out std logic ) ;
end mux4to1 ;
architecture  behavior  of   mux4to1 is
begin
aeqb <= ‘1’ when  a=b else ‘0’;
agtb <= ‘1’ when  a<b else ‘0’;
altb <= ‘1’ when  a>b else ‘0’;

end behavior ;
```

Figure 6.36(modified) (page 350)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity mux2To1 is
(    a,b,s: in std_logic;
     c:     out std_logic
);
end mux2To1;
architecture logic_function of mux2To1 is
begin
with s  select
c<= a when ‘0’  ,
        b   when ‘1’;
end logic_function;
```

4 to 1 mux using for generate statement

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity mux4To1 is
( a :in std_logic_vector(3 downto 0);
   s: in std_logic_vector(1 downto 0);
   c:      out std_logic
);
end mux4To1;
architecture logic_function of mux4To1 is
signal m: std_logic_vector ( 1 downto 0);
begin
G1:         for i in 0 to 1 generate
muxes: entity work.mux2To1
port map( a=>a(i), b=> a( i+1) , s=>s(0), c=>m(i)  );
end generate:
mux3:entity work.mux2To1
port map ( a=>m(0),b=>m(1),s=>s(1),c=>c);
end logic_function;
```

page 361

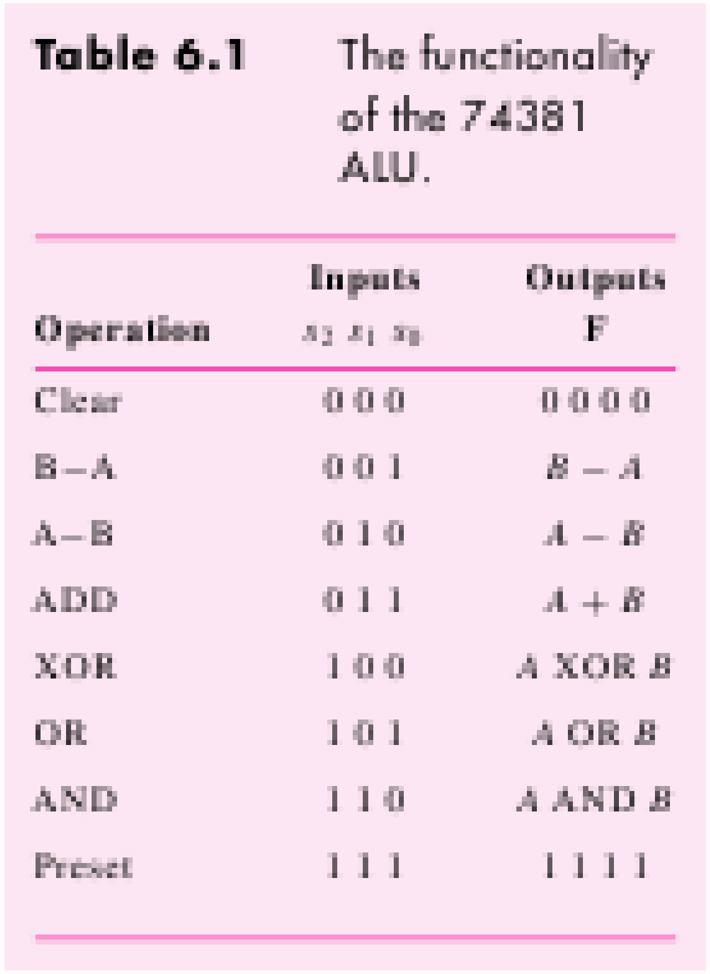

Table 6.1 The functionality of the 74381 ALU.

Operation	Inputs $s_2\ s_1\ s_0$	Outputs F
Clear	0 0 0	0 0 0 0
B−A	0 0 1	$B - A$
A−B	0 1 0	$A - B$
ADD	0 1 1	$A + B$
XOR	1 0 0	A XOR B
OR	1 0 1	A OR B
AND	1 1 0	A AND B
Preset	1 1 1	1 1 1 1

VHDL code representing the functionality of ALU 74381 chip

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
Use IEEE.numeric_std.all;
entity alu is
( A,B :in std_logic_vector(3 downto 0);
   s: in std_logic_vector(2 downto 0);
   f:     out std_logic
);
end alu;
architecture logic_function of alu is
begin
```

```
process( s,A,B)
begin
    case s is
        when “000” =>
            f <= “0000”;
        when “001” =>
            f <= B-A;
        when “010” =>
            f <= A-B;
        when “011” =>
            f <= A+B;
        when “100” =>
            f <= A xor B ;
        when “101” =>
            f <= A or B ;
        when “110” =>
            f <= A and B;
        when “111” =>
            f <= “1111”;
```