

第三章 组合逻辑电路建模

3.1 Decoder Design

3-8 Decoder

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY decoder38 IS
    PORT(a,b,c,g1,g2a,g2b: IN std_logic;
         y: OUT std_logic_vector(7 DOWNTO 0));
END decoder38;

ARCHITECTURE behav OF decoder38 IS
    SIGNAL indata: std_logic_vector(2 DOWNTO 0);
BEGIN
    indata <= c&b&a;
    PROCESS (indata, g1,g2a, g2b)
    BEGIN
        IF g1 = '1' AND g2a = '0' AND g2b = '0' THEN
            CASE indata IS
                WHEN "000" => Y <= "11111110"; -- ';' needed for seperation
                WHEN "001" => Y <= "11111101";
                WHEN "010" => Y <= "11111011";
                WHEN "011" => Y <= "11110111";
                WHEN "100" => Y <= "11101111";
                WHEN "101" => Y <= "11011111";
                WHEN "110" => Y <= "10111111";
                WHEN "111" => Y <= "01111111";
                WHEN OTHERS => Y <= "XXXXXXXX"; --note that OTHERS should be cc
            END CASE;
        ELSE
            Y <= "11111111";
        END IF;
    END PROCESS;
END behav;
```

3.2 Encoder Design

Normal Encoder

- has the inverse form of the decoder
- no PROCESS structure needed

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
```

```
ENTITY encoder83 IS
    PORT(x: IN std_logic_vector(7 DOWNTO 0);
         y: OUT std_logic_vector(2 DOWNTO 0));
END encoder83;

ARCHITECTURE behav OF encoder83 IS
    SIGNAL indata: std_logic_vector(2 DOWNTO 0);
BEGIN
    y <= "000" WHEN x="00000001" ELSE -- no ';' needed
    y <= "001" WHEN x="00000010" ELSE
    y <= "010" WHEN x="00000100" ELSE
    y <= "011" WHEN x="00001000" ELSE
    y <= "100" WHEN x="00010000" ELSE
    y <= "101" WHEN x="00100000" ELSE
    y <= "110" WHEN x="01000000" ELSE
    y <= "111" WHEN x="10000000" ELSE
    "ZZZ";
END behav;
```

Priority Encoder

- If multiple input signals are valid, the encoder's output corresponds to the highest priority input

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY encoder83 IS
    PORT(x: IN std_logic_vector(7 DOWNTO 0);
         y: OUT std_logic_vector(2 DOWNTO 0));
END encoder83;

ARCHITECTURE behav OF encoder83 IS
    SIGNAL indata: std_logic_vector(2 DOWNTO 0);
BEGIN
    PROCESS(x)
    BEGIN
        IF X(0) = '0' THEN y<="111"; -- ';' needed for seperation
        ELSIF X(1) = '0' THEN y<="110";
        ELSIF X(2) = '0' THEN y<="101";
        ELSIF X(3) = '0' THEN y<="100";
        ELSIF X(4) = '0' THEN y<="011";
        ELSIF X(5) = '0' THEN y<="010";
        ELSIF X(6) = '0' THEN y<="001";
        ELSIF X(7) = '0' THEN y<="000";
        ELSE y<="XXX";
        END IF;
    END PROCESS;
END behav;
```

3.3 Adder Design

Half Adder

```
sum <= x XOR y;  
co <= x AND y
```

Full Adder

```
sum <= x XOR y XOR ci;  
co <= (x AND y) or (x AND ci) or (y AND ci)
```

Packup as COMPONENT

```
COMPONENT full_adder IS  
  PORT(x,y,ci:IN std_logic;  
        sun, co: OUT std_logic);  
END full_adder;
```

Sequential 4-bit Adder

- Use packaged component
- See P55

Parallel 4-bit Adder

- See P56
- higher speed than sequential adder
- needs more logic resources than sequential adder
- **For 4-bit adder, parallel adder consumes almost the same as the sequential one**

3.4 Multiplexer(MUX)

-See P58 or PPT.4.Combinational Logic using VHDL by Q. P48

3.5 Complement Operation

```
USE IEEE.std_logic_unsigned.all;  
...  
out <= not input + '1';
```

3.6 Tri-state Gate

- Output: '0','1','Z'
- Use en to control the output, when en = '0' then out <= 'Z'

```
PROCESS(in, en)  
BEGIN
```

```
IF en='1' THEN  
  out <= in;  
ELSE  
  out <= 'z';  
END IF;  
END PROCESS;
```

3.7 Buffer

INPUT/OUTPUT Buffer

- See P62

Bi-directional Bus Buffer

```
LIBRARY IEEE;  
USE IEEE.std_logic_.all;  
  
ENTITY tri_bigate IS  
  PORT(a,b:INOUT std_logic_vector(7 DOWNTO 0);  
        en,dr:IN std_logic);  
END tri_bigate;  
  
ARCHITECTURE behav OF tri_bigate IS  
  SIGNAL aout, bout: std_logic_vector(7 DOWNTO 7)  
BEGIN  
  PROCESS(a,dr,en,bout)  
  BEGIN  
    IF (en='0') AND (dr='1') THEN  
      bout<=a;  
    ELSE bout <="ZZZZZZZZ";  
    END IF;  
    b<=bout;  
  END PROCESS;  
  PROCESS(b,dr,en,bout)  
  BEGIN  
    IF (en='0') AND (dr='0') THEN  
      aout<=b;  
    ELSE aout <="ZZZZZZZZ";  
    END IF;  
    a<=aout;  
  END PROCESS;  
END behav;
```

3.8 Comparator

3.9 ROM/RAM