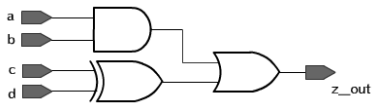
6장 조합논리회로의 설계

제1절 연산자를 이용한 조합논리회로 설계

1. 논리연산자의 이용



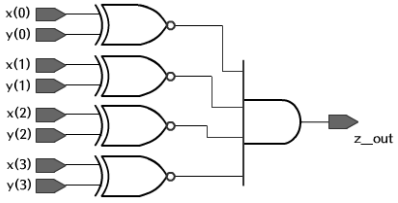
[그림 6.1] 논리연산자를 활용한 조합논리회로

<VHDL 구문>

```
1   entity logic_example is
2       port (a, b, c, d : in std_logic;
3             z_out : out std_logic);
4   end logic_example;
5   architecture combi_logic1 of logic_example is
6    signal x : std_logic;
7   begin
8    z_out <= (a and b) or x;
9       x <= c xor d;
10  end combi_logic1;
```

제1절 연산자를 이용한 조합논리회로 설계

2. 관계 연산자의 이용



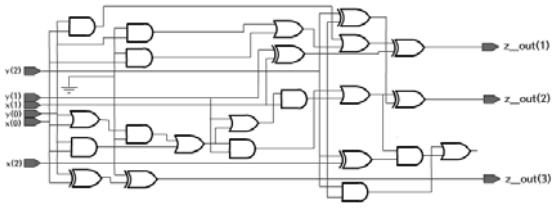
[그림 6.2] 관계연산자를 활용한 조합논리회로

<VHDL 구문>

```
1  entity relation_example is
2     port (x, y : in std_logic_vector(3 downto 0);
3          z_out : out std_logic);
4  end relation_example;
5  architecture combi_logic2 of relation_example is
6   begin
7    z_out <= (x = y);
8  end combi_logic2;
```


제1절 연산자를 이용한 조합논리회로 설계

3. 산술연산자의 이용



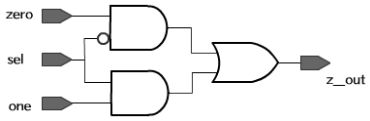
[그림 6.3] 산술연산자를 활용한 조합논리회로

<VHDL 구문>

```
1  package arithmetic_exam_pkg is
2    type medium is range 0 to 7;
3  end arithmetic_exam_pkg;
4  use work.arithmetic_exam_pkg.all;
5  entity arithmetic_example is
6     port (x, y : in medium;
7          z_out : out medium);
8   end relation_example;
9   architecture combi_logic3 of arithmetic_example is
10   begin
11    z_out <= x + y;
12  end combi_logic3;
```

제2절 조건적 논리기능의 조합논리회로

1. 조건적 신호대입의 이용



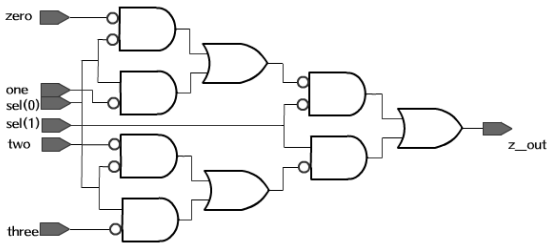
[그림 6.4] 조건적 신호대입의 조합논리회로

<VHDL 구문>

```
1  entity condition_example is
2     port (zero, one, sel : in std_logic;
3           z_out : out std_logic);
4  end condition_example;
5  architecture combi_logic4 of condition_example is
6   begin
7    z_out <= one when sel='1'
8          else zero;
9   end combi_logic4;
```


제2절 조건적 논리기능의 조합논리회로

2. 선택적 신호대입의 이용



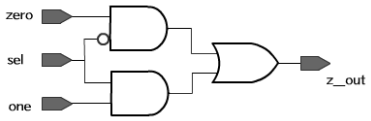
[그림 6.5] 선택적 신호대입의 조합논리회로

<VHDL 구문>

```
1  entity selective_example is
2     port (zero, one, two three : in std_logic;
3           sel : in std_logic_vector(1 downto 0);
4           z_out : out std_logic);
5  end selective_example;
6  architecture combi_logic5 of selective_example is
7   begin
8     with sel select
9     z_out <= zero when "00",
10             one when "01",
11             two when "10",
12             three when others;
13  end combi_logic5;
```

제2절 조건적 논리기능의 조합논리회로

3. if문의 이용



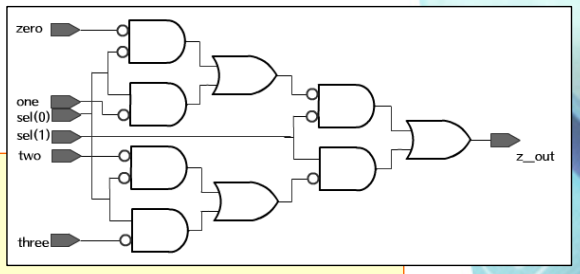
[그림 6.6] if문을 활용한 조합논리회로

<VHDL 구문>

```
1   entity conditional_example is
2      port (zero, one, sel : in boolean;
3            z_out : out boolean);
4   end conditional_example;
5   architecture combi_logic6 of conditional_example is
6    begin
7     process (zero, one, sel)
8      begin
9       if sel then
10           z_out <= one;
11       else
12           z_out <= zero;
13      end if;
14     end process;
15   end combi_logic6;
```


제2절 조건적 논리기능의 조합논리회로

4. case문의 이용



[그림 6.7] case문을 활용한 조합논리회로

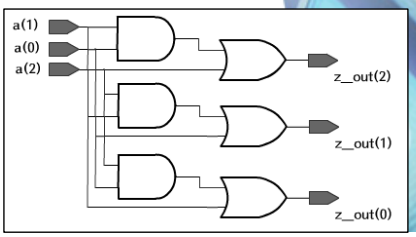
<VHDL 구문>

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   entity conditional_example is
4      port (zero, one, two, three : in std_logic;
5            sel : in std_logic_vector(1 downto 0);
6            z_out : out std_logic);
7   end conditional_example;
8   architecture combi_logic7 of conditional_example is
9    begin
10     process (zero, one, two, three, sel)
11      begin
12       case sel is
13         when "00" => z_out <= zero;
14         when "01" => z_out <= one;
15         when "10" => z_out <= two;
16         when others => z_out <= three;
17       end case;
18     end process;
19   end combi_logic7;
```

제3절 반복논리기능을 활용한 조합논리회로

1. 함수와 프로시저문의 이용

▪ function문



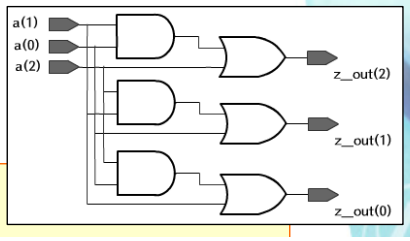
[그림 6.8] 함수를 이용한 조합논리회로

<VHDL 구문>

```
1  entity function_example is
2     port ( a : in bit_vector(2 downto 0);
3           z_out : out bit_vector(2 downto 0);
4  end funcrion_example;
5  architecture combi_logic8 of function_example is
6    function gate (in1, in2, in3 : bit) return bit is
7   begin
8      return (in1 and in2) or in3;
9    end;
10    process ( a )
11     begin
12       z_out(2) <=  gate(a(0), a(1), a(2));
13       z_out(1) <=  gate(a(1), a(2), a(0));
14       z_out(0) <=  gate(a(2), a(0), a(1));
15    end process;
16 end combi_logic8;
```


제3절 반복논리기능을 활용한 조합논리회로

▪ procedure문



[그림 6.9] procedure를 이용한 조합논리회로

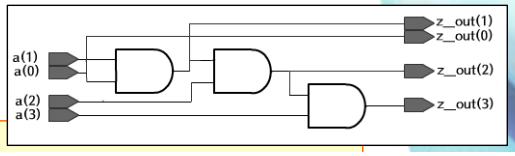
<VHDL 구문>

```
1  entity procedure_example is
2     port ( a : in bit_vector(2 downto 0);
3           z_out : out bit_vector(2 downto 0);
4  end procedure_example;
5  architecture combi_logic9 of procedure_example is
6   procedure gate (in1, in2, in3 : in bit; signal x : out bit) is
7    begin
8       x <= (in1 and in2) or in3;
9   end;
10    process ( a )
11     begin
12       gate(a(0), a(1), a(2), z_out(2));
13       gate(a(1), a(2), a(0), z_out(1));
14       gate(a(2), a(0), a(1), z_out(0));
15    end process;
16 end combi_logic9;
```

제3절 반복논리기능을 활용한 조합논리회로

2. loop문의 이용

- for ~ loop문



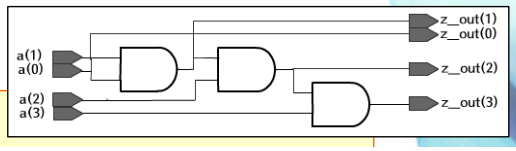
[그림 6.10] loop문을 이용한 조합논리회로

<VHDL 구문>

```
1   entity forloop_example is
2      port ( a : in std_logic_vector(3 downto 0);
3            z_out : out std_logic_vector(3 downto 0);
4   end forloop_example;
5   architecture combi_logic10 of forloop_example is
6   -- process statement
7    begin
8     process ( a )
9       variable s : std_logic;
10      begin
11         s := '1';
12       for i in 0 to 3 loop
13         s := a(i) and s;
14        z_out(i) <= s;
15       end loop;
16     end process;
17   end combi_logic10;
```


제3절 반복논리기능을 활용한 조합논리회로

- while ~ loop문



[그림 6.11] loop문을 이용한 조합논리회로

<VHDL 구문>

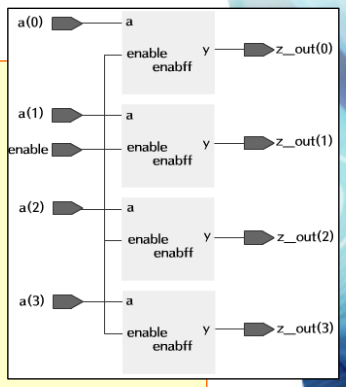
```
1   entity whileloop_example is
2      port ( a : in std_logic_vector(3 downto 0),
3            z_out : out std_logic_vector(3 downto 0);
4   end whileloop_example;
5   architecture combi_logic11 of whileloop_example is
6    begin
7     process ( a )
8       variable s : bit;
9       variable i : integer;
10      begin
11         i := '0';
12       while i < 4 loop
13         s := a(i) and s;
14        z_out(i) <= s;
15          i := i + 1;
16       end loop;
17     end process;
18   end combi_logic11;
```

제3절 반복논리기능을 활용한 조합논리회로

3. generate문의 이용

<VHDL 구문>

```
 1  library ieee;
 2  use ieee.std_logic_1164.all;
 3   entity generate_example is
 4      port (enable : in std_logic;
 5              a : in std_logic_vector(3 downto 0);
 6           z_out : out std_logic_vector(3 downto 0);
 7   end generate_example;
 8   architecture combi_logic12 of generate_example is
 9    component enabff
10     port (enable, a : in std_logic;
11           z : out std_logic);
12      end component;
13      begin
14      g1 :  for i in 3 to 0 generate
15          u : enabff port map (enable, a(i) z_out(i));
16        end genertate g1;
17     end process;
18   end combi_logic12;
```

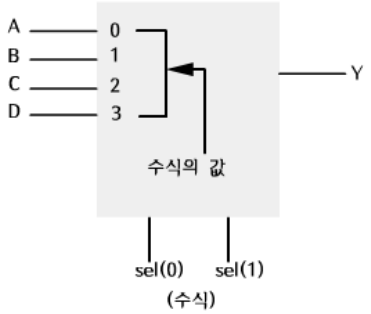


(4x1MUX)

[그림 6.12] generate문을 이용한 조합논리회로

제4절 멀티플렉서(Multiplexer)의 설계

1. 다중 if문의 이용



sel	출 력
00	A
01	B
10	C
11	D

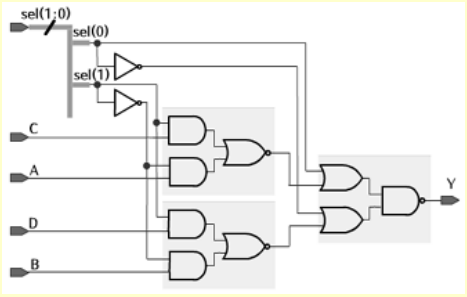
(a) 기호 (b) 진리표

[그림 6.13] MUX의 개념회로

제4절 멀티플렉서(Multiplexer)의 설계

<VHDL구문>

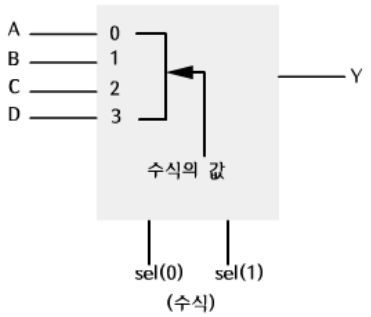
```
library IEEE;
use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;
entity MUX4_1 is
 port (SEL      :  in std_logic_vector(1 downto 0);
       A, B, C, D : in std_logic;
       Y        : out std_logic);
end MUX4_1;
architecture DATAFLOW_example of MUX4_1 is
 begin
 1  process (SEL, A, B, C, D)
 2   begin
 3    if (SEL = "00") then
 4      Y <= A;
 5    elsif (SEL = "01") then
 6      Y <= B;
 7    elsif (SEL = "10") then
 8      Y <= C;
 9    else
10      Y <= D;
11    end if;
12  end process;
end DATAFLOW_example;
```



[그림 6.14] 4x1MUX의 합성회로

제4절 멀티플렉서(Multiplexer)의 설계

2. when ~ else문의 이용



sel	출 력
00	A
01	B
10	C
11	D

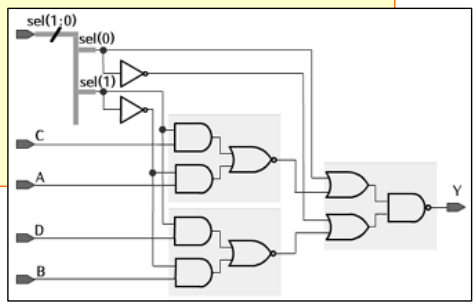
(a) 기호 (b) 진리표

[그림 6.13] MUX의 개념회로

제4절 멀티플렉서(Multiplexer)의 설계

<VHDL구문>

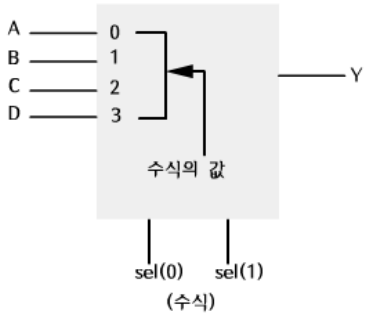
```
library IEEE;
use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;
entity MUX4_1 is
  port (SEL       :  in std_logic_vector(1 downto 0);
        A, B, C, D : in std_logic;
        Y          : out std_logic);
  end MUX4_1;
architecture WHENELSE_example of MUX4_1 is
  begin
1   Y <= A when sel = "00" else
2        B when sel = "01" else
3        C when sel = "10" else
4        D  -- when sel = "11"
  end WHENELSE_example;
```



[그림 6.14] 4x1MUX의 합성회로

제4절 멀티플렉서(Multiplexer)의 설계

3. CASE문의 이용



sel	출 력
00	A
01	B
10	C
11	D

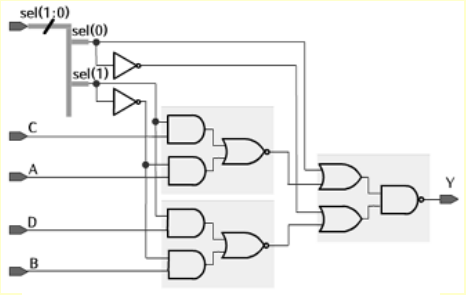
(a) 기호 (b) 진리표

[그림 6.13] MUX의 개념회로

제4절 멀티플렉서(Multiplexer)의 설계

<VHDL구문>

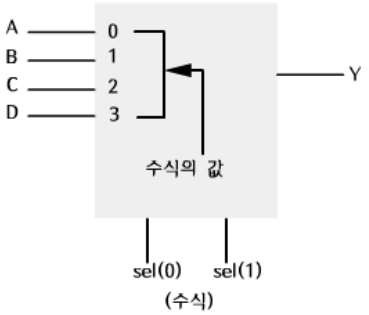
```
library IEEE;
use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;
entity MUX4_1 is
  port (SEL       :  in std_logic_vector(1 downto 0);
        A, B, C, D : in std_logic;
        Y          : out std_logic);
end MUX4_1;
architecture CASE_example of MUX4_1 is
  begin
1   process (sel, A, B, C, D)
2    begin
3     case sel is
4       when "00" => Y <= A;
5       when "01" => Y <= B;
6       when "10" => Y <= C;
7       when others => Y <= D;
8     end case;
9    end process;
   end CASE_example;
```



[그림 6.14] 4x1MUX의 합성회로

제4절 멀티플렉서(Multiplexer)의 설계

4. with ~ select문의 이용



sel	출 력
00	A
01	B
10	C
11	D

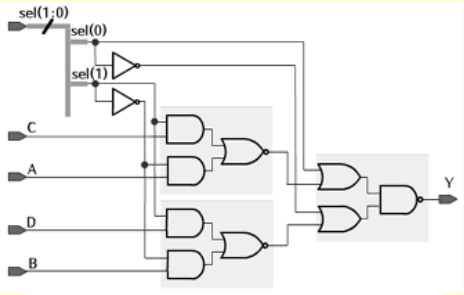
(a) 기호 (b) 진리표

[그림 6.13] MUX의 개념회로

제4절 멀티플렉서(Multiplexer)의 설계

<VHDL구문>

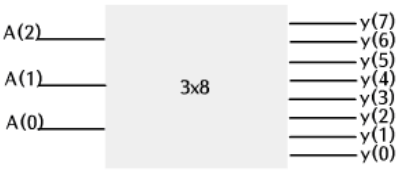
```
library IEEE;
use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;
entity MUX4_1 is
  port (SEL       :  in std_logic_vector(1 downto 0);
        A, B, C, D : in std_logic;
        Y          : out std_logic);
end MUX4_1;
architecture WITHSELECT_example of MUX4_1 is
  begin
1     with sel select
2      Y <= A when "00",
3           B when "01",
4           C when "10",
5           D when "11",
6           A when others;
 end WITHSELECT_example;
```



[그림 6.14] 4x1MUX의 합성회로

제5절 디코더(Decoder)의 설계

1. 다중 if문의 이용



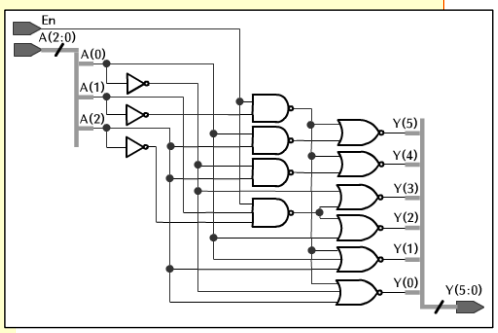
입 력			출 력							
A(2)	A(1)	A(0)	Y(7)	Y(6)	Y(5)	Y(4)	Y(3)	Y(2)	Y(1)	Y(0)
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

[그림 6.15] Decoder의 개념 (a) 기호 (b)진리표

제5절 디코더(Decoder)의 설계

<VHDL구문>

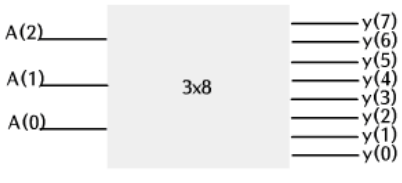
```
library IEEE;
use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;
entity DECODER3_8 is
  port (A :  in integer range 0 to 7;
        Y : out std_logic_vector(7 downto 0));
end DECODER3_8;
architecture DATAFLOW_example of DECODER3_8 is
  begin
1   process (A)
2   begin
3     if (A = 1) then  Y <= "00000001";
4     elsif (A = 2) then Y <= "00000010";
5     elsif (A = 3) then Y <= "00000100";
6     elsif (A = 4) then Y <= "00001000";
7     elsif (A = 5) then Y <= "00010000";
8     elsif (A = 6) then Y <= "00100000";
9     elsif (A = 3) then Y <= "01000000";
10      else  Y <= "10000000";
11      end if;
12    end process;
  end DATAFLOW_example;
```



[그림 6.16] 3x8Decoder의 합성회로

제5절 디코더(Decoder)의 설계

2. WHEN ~ ELSE문의 이용



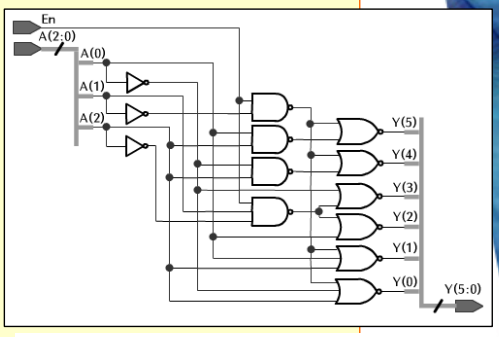
입 력			출 력							
A(2)	A(1)	A(0)	Y(7)	Y(6)	Y(5)	Y(4)	Y(3)	Y(2)	Y(1)	Y(0)
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

[그림 6.15] Decoder의 개념 (a) 기호 (b)진리표

제5절 디코더(Decoder)의 설계

<VHDL구문>

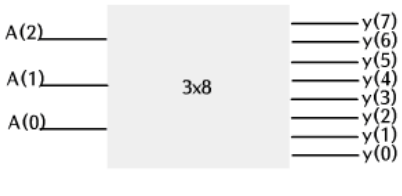
```
   library IEEE;
   use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;
   entity DECODER3_8 is
     port (A :  in integer range 0 to 7;
              Y : out std_logic_vector(7 downto 0));
   end DECODER3_8;
   architecture WHENELSE_example of DECODER3_8 is
     begin
1     Y <= "00000001" when A = 0 else
2          "00000010" when A = 1 else
3          "00000100" when A = 2 else
4          "00001000" when A = 3 else
5          "00010000" when A = 4 else
6          "00100000" when A = 5 else
7          "01000000" when A = 6 else
8          "10000000";
   end WHENELSE_example;
```



[그림 6.16] 3x8Decoder의 합성회로

제5절 디코더(Decoder)의 설계

3. CASE문의 이용



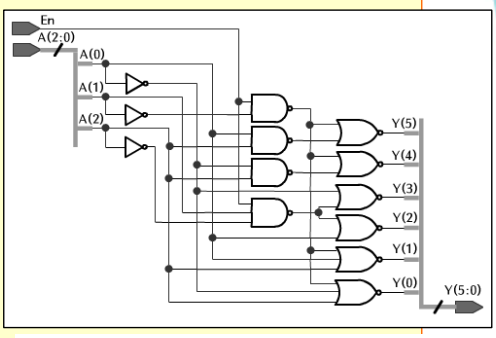
입 력			출 력							
A(2)	A(1)	A(0)	Y(7)	Y(6)	Y(5)	Y(4)	Y(3)	Y(2)	Y(1)	Y(0)
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

[그림 6.15] Decoder의 개념 (a) 기호 (b)진리표

제5절 디코더(Decoder)의 설계

<VHDL구문>

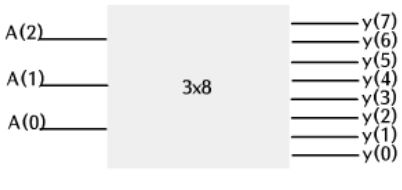
```
library IEEE;
use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;
entity DECODER3_8 is
  port (A :  in integer range 0 to 7;
        Y : out std_logic_vector(7 downto 0));
end DECODER3_8;
architecture CASE_example of DECODER3_8 is
  begin
1   process (A)
2    begin
3     case A is
4      when 0 => Y <= "00000001";
5      when 1 => Y <= "00000010";
6      when 2 => Y <= "00000100";
7      when 3 => Y <= "00001000";
8      when 4 => Y <= "00010000";
9      when 5 => Y <= "00100000";
10      when 6 => Y <= "01000000";
11      when 7 => Y <= "10000000";
12      end case;
13    end process;
  end CASE_example;
```



[그림 6.16] 3x8Decoder의 합성회로

제5절 디코더(Decoder)의 설계

4. WITH ~ SELECT문의 이용



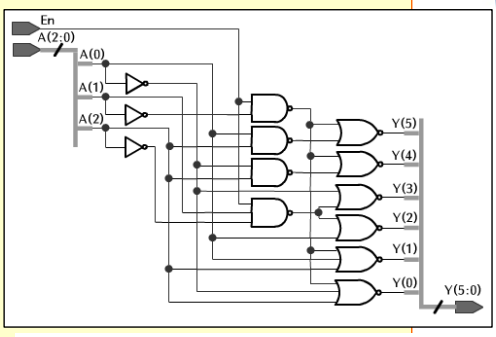
입 력			출 력							
A(2)	A(1)	A(0)	Y(7)	Y(6)	Y(5)	Y(4)	Y(3)	Y(2)	Y(1)	Y(0)
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

[그림 6.15] Decoder의 개념 (a) 기호 (b)진리표

제5절 디코더(Decoder)의 설계

<VHDL구문>

```
  library IEEE;
  use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;
  entity DECODER3_8 is
    port (A :  in integer range 0 to 7;
          Y : out std_logic_vector(7 downto 0));
  end DECODER3_8;
  architecture WITH_SELECT_example of DECODER3_8 is
    begin
1     with A select
2      Y <= "00000001" when 0,
3           "00000010" when 1,
4           "00000100" when 2,
5           "00001000" when 3,
6           "00010000" when 4,
7           "00100000" when 5,
8           "01000000" when 6,
9           "10000000" when 7,
10            "00000000" when others;
  end WITH_SELECT_example;
```



[그림 6.16] 3x8Decoder의 합성회로

제5절 디코더(Decoder)의 설계

5. Enable 기능의 디코더 설계

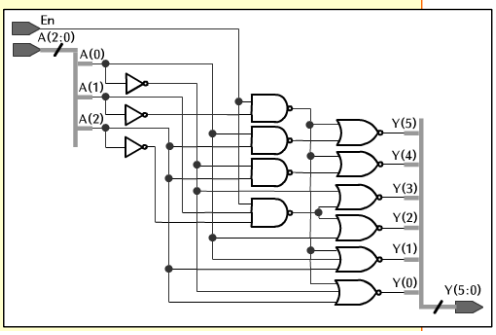
[표 6.1]Enable기능의 3x6디코더 진리표

입 력				출 력					
En	A(2)	A(1)	A(0)	Y(5)	Y(4)	Y(3)	Y(2)	Y(1)	Y(0)
0	X	X	X	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	1	0
1	0	1	0	0	0	0	1	0	0
1	0	1	1	0	0	1	0	0	0
1	1	0	0	0	1	0	0	0	0
1	1	0	1	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0

제5절 디코더(Decoder)의 설계

<VHDL구문>

```
   library IEEE;
   use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;
   entity EN_DECODER3_6 is
    port (A :  in integer range 0 to 7;
          Y : out std_logic_vector(7 downto 0));
   end EN_DECODER3_6;
   architecture CASE_example of EN_DECODER3_6 is
    begin
1   process (A)
2    begin
3     if (En = '0') then
4      Y<= "000000"
5     else
6       case A is
7        when 0 => Y <= "000001";
8        when 1 => Y <= "000010";
9        when 2 => Y <= "000100";
10        when 3 => Y <= "001000";
11        when 4 => Y <= "010000";
12        when 5 => Y <= "100000";
13        when others => Y <= "000000";
14       end case;
15    end if;
16   end process;
   end CASE_example;
```



[그림 6.17] Enable_Decoder의 합성회로

제6절 인코더(Encoder)의 설계

1. 다중 if문의 이용

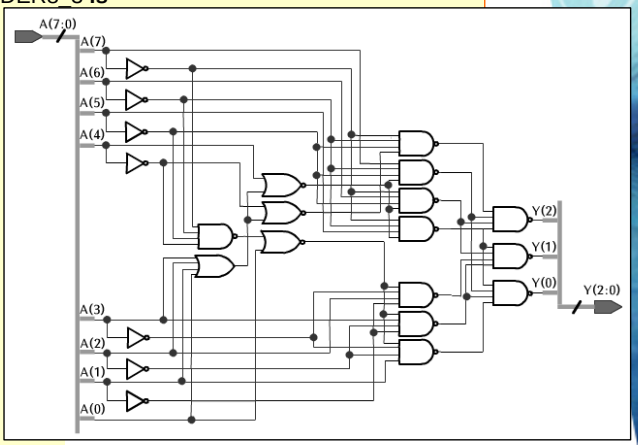
입력								출력		
A(7)	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)	Y(2)	Y(1)	Y(0)
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

[그림 6.18] Encoder의 개념 (a) 기호 (b)진리표

제6절 인코더(Encoder)의 설계

<VHDL구문>

```
library IEEE;
use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;
entity ENCODER8_3 is
  port (A :  in std_logic_vector(7 downto 0);
        Y : out std_logic_vector(2 downto 0));
end ENCODER8_3;
architecture DATAFLOW_example of ENCODER8_3 is
  begin
1   process (A)
2    begin
3     if (A = "00000001") then  Y <= "000";
4     elsif (A = "00000010") then Y <= "001";
5     elsif (A = "00000100") then Y <= "010";
6     elsif (A = "00001000") then Y <= "011";
7     elsif (A = "00010000") then Y <= "100";
8     elsif (A = "00100000") then Y <= "101";
9     elsif (A = "01000000") then Y <= "110";
10      elsif (A = "10000000") then Y <= "111";
11      else Y <="xxx"
12      end if;
13    end process;
  end DATAFLOW_example;
```



[그림 6.19] 8x3Encoder회로의 합성결과

제6절 인코더(Encoder)의 설계

2. WHEN ~ ELSE문의 이용

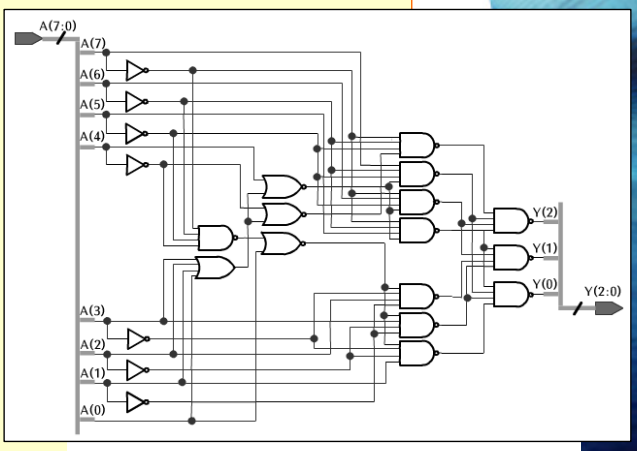
입력								출력		
A(7)	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)	Y(2)	Y(1)	Y(0)
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

[그림 6.18] Encoder의 개념 (a) 기호 (b)진리표

제6절 인코더(Encoder)의 설계

<VHDL구문>

```
library IEEE;
use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;
entity ENCODER8_3 is
  port (A :  in std_logic_vector(7 downto 0);
        Y : out std_logic_vector(2 downto 0));
end ENCODER8_3;
architecture WHEN_ELSE_example of ENCODER8_3 is
  begin
1   Y <= "000" when A = "00000001" else
2        "001" when A = "00000010" else
3        "010" when A = "00000100" else
4        "011" when A = "00001000" else
5        "100" when A = "00010000" else
6        "101" when A = "00100000" else
7        "110" when A = "01000000" else
8        "111" when A = "10000000" else
          "xxx";
end WHEN_ELSE_example;
```



[그림 6.19] 8x3Encoder회로의 합성결과

제6절 인코더(Encoder)의 설계

3. CASE문의 이용

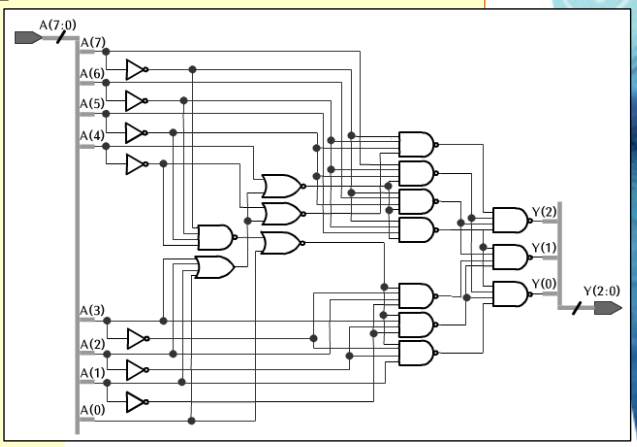
입력								출력		
A(7)	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)	Y(2)	Y(1)	Y(0)
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

[그림 6.18] Encoder의 개념 (a) 기호 (b)진리표

제6절 인코더(Encoder)의 설계

<VHDL구문>

```
library IEEE;
use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;
entity ENCODER8_3 is
  port (A :  in std_logic_vector(7 downto 0);
        Y : out std_logic_vector(2 downto 0));
end ENCODER8_3;
architecture CASE_example of ENCODER8_3 is
 begin
1   process (A)
2    begin
3     case A is
4      when "00000001" => Y <= "000";
5      when "00000010" => Y <= "001";
6      when "00000100" => Y <= "010";
7      when "00001000" => Y <= "011";
8      when "00010000" => Y <= "100";
9      when "00100000" => Y <= "101";
10      when "01000000" => Y <= "110";
11      when "10000000" => Y <= "111";
12      when others     => Y <= "xxx";
13      end case;
14    end process;
end CASE_example;
```



[그림 6.19] 8x3Encoder회로의 합성결과

제6절 인코더(Encoder)의 설계

4. WITH ~ SELECT문의 이용

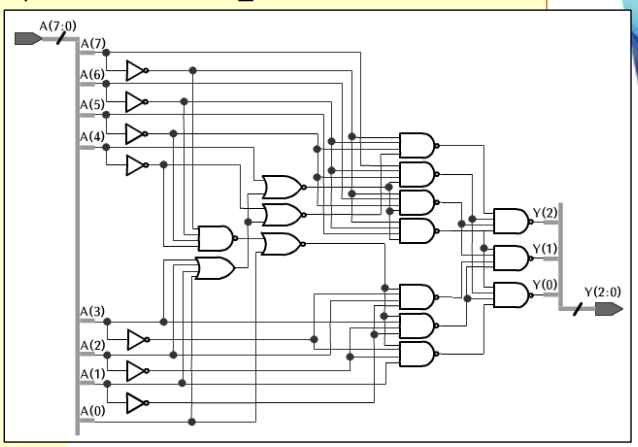
입력								출력		
A(7)	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)	Y(2)	Y(1)	Y(0)
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

[그림 6.18] Encoder의 개념 (a) 기호 (b)진리표

제6절 인코더(Encoder)의 설계

<VHDL구문>

```
  library IEEE;
  use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;
  entity ENCODER8_3 is
    port (A :  in std_logic_vector(7 downto 0);
          Y : out std_logic_vector(2 downto 0));
  end DECODER3_8;
  architecture WITH_SELECT_example of ENCODER8_3 is
    begin
1     with A select
2       Y <= "000" when "00000001",
3       Y <= "001" when "00000010",
4       Y <= "010" when "00000100",
5       Y <= "011" when "00001000",
6       Y <= "100" when "00010000",
7       Y <= "101" when "00100000",
8       Y <= "110" when "01000000",
9       Y <= "111" when "10000000",
10        Y <= "XXX" when others;
  end WITH_SELECT_example;
```



[그림 6.19] 8x3Encoder회로의 합성결과

제6절 인코더(Encoder)의 설계

5. FOR ~ LOOP문의 이용

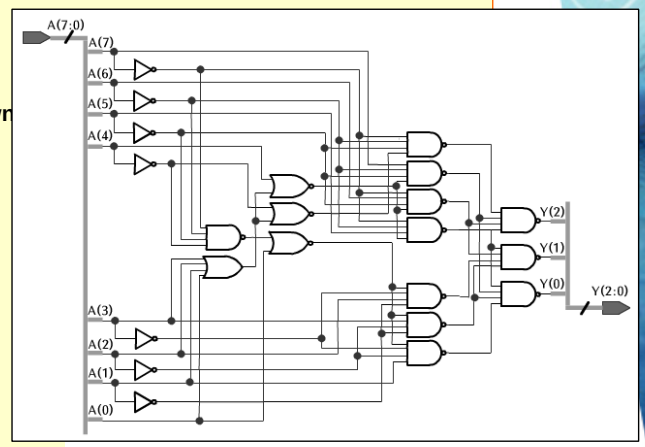
입력								출력		
A(7)	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)	Y(2)	Y(1)	Y(0)
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

[그림 6.18] Encoder의 개념 (a) 기호 (b)진리표

제6절 인코더(Encoder)의 설계

<VHDL 구문>

```
    library IEEE;
    use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;
     entity ENCODER8_3 is
        port ( a : in std_logic_vector(7 downto 0);
              y : out std_logic_vector(2 downto 0);
     end ENCODER8_3;
     architecture FOR_LOOP_example of ENCODER8_3 is
      begin
 1     process ( a )
 2       variable n : integer range 0 to 7;
 3       variable test : std_logic_vector(7 downt
 4      begin
 5         test := "00000001";
 6           y <= "xxx"
 7       for n in 0 to 7 loop
 8         if (a = test) then
 9           y <= to_unsigned(n, 3);
10            exit;
11          end if;
12         test := shift_left(test, 1);
13       end loop;
14     end process;
15   end FOR_LOOP_example;
```



[그림 6.19] 8x3Encoder회로의 합성결과

제7절 ALU(Arithmetic Logic Unit)의 설계

[표 6.3] ALU의 기능표

S4	S3	S2	S1	S0	Cin	동 작	기 능	관련 장치
0	0	0	0	0	0	Y <= A	전달 A	산술 장치
0	0	0	0	0	1	Y <= A + 1	증가	
0	0	0	0	1	0	Y <= A + B	덧셈	
0	0	0	0	1	1	Y <= A + B +1	캐리더하기	
0	0	0	1	0	0	Y <= A + $\overline{B}$	B의 1의 보수 더하기	
0	0	0	1	0	1	Y <= A + $\overline{B}$ + 1	뺄셈	
0	0	0	1	1	0	Y <= A - 1	감소 A	
0	0	0	1	1	1	Y <= A+	전달 A	
0	0	1	0	0	0	Y <= A and B	AND	논리 장치
0	0	1	0	1	0	Y <= A or B	OR	
0	0	1	1	0	0	Y <= A xor B	XOR	
0	0	1	1	1	0	Y <= $\overline{A}$	A의 부정	
0	0	0	0	0	0	Y <= A	전달 A	쉬프터 장치
0	1	0	0	0	0	Y <= shl A	좌측 이동 A	
1	0	0	0	0	0	Y <= shr A	우측 이동 A	
1	1	0	0	0	0	Y <= 0	전달 0	

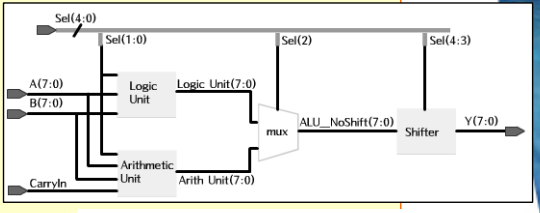
제7절 ALU(Arithmetic Logic Unit)의 설계

<VHDL 구문>

```
   library IEEE;
   use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;

   entity combi_ALU is
     port (sel      : in std_logic_vector(4 downto 0);
           carryin : in std_logic;
           A, B     : in std_logic_vector (7 downto 0);
           Y        : out std_logic_vector (7 downto 0));
   end combi_ALU;
1   architecture ALU_example of combi_ALU is
2     begin
3      process (sel, A, B, carryin)
4        variable sel0_1_carryin: std_logic_vector(2 downto 0);
5        variable LogicUnit, ArithUnit, ALU_NoShift: std_logic_vector (7 downto 0);
6       begin
7    Logic_Unit: case sel(1 downto 0) is     -- logic unit에 대한 동작
8          when "00"  => LogicUnit := A and B;
9          when "01"  => LogicUnit := A or B;
10         when "10"  => LogicUnit := A xor B;
11         when "11"  => LogicUnit := not A;
12         when others => LogicUnit := (others => 'X');
13       end case Logic_Unit;
14      sel0_1_carryin := sel(1 downto 0) & carryin;
```


제7절 ALU(Arithmetic Logic Unit)의 설계

```
15    Arith_Unit:  case sel0_1_carryin) is     -- arithmetic unit에 대한 동작
16          when "000"  => ArithUnit := A;
17          when "001"  => ArithUnit := A + 1;
18          when "010"  => ArithUnit := A + B;
19          when "011"  => ArithUnit := A + B + 1;
20          when "100"  => ArithUnit := A + not B;
21          when "101"  => ArithUnit := A - B;
22          when "110"  => ArithUnit := A - 1;
23          when "111"  => ArithUnit := A;
24          when others => ArithUnit := (others => 'X');
25       end case Arith_Unit;


26    LA_Multi: if (sel(2) = '1') then   --Logic Unit와 Arithmetic Unit의 복합기능의 기술
27          ALU_NoShift := LogicUnit;
28     else
29          ALU_NoShift := ArithUnit;
30     end if LA_Multi;


31    Shift: case sel(4 downto 3) is      -- shift 동작기능의 기술
32       when "00"  => Y <= ALU_Noshift;
33       when "01"  => Y <= shift_left(ALU_Noshift, 1);
34       when "10"  => Y <= shift_right(ALU_Noshift, 1);
35       when "11"  => Y <= (others => '0');
36       when others => Y <= (others => 'X');
37     end case Shift;
38    end process;
39  end ALU_example;
```



[그림 6.23] ALU 회로의 구조