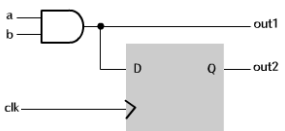
7장 순서논리회로의 설계

제1절 순서논리회로

1. 순서논리의 개념

[그림 7.1] 순서논리의 개념

2. 순서논리회로의 특성



[그림 7.2] 순서논리회로

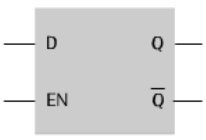
제1절 순서논리회로

<VHDL 구문>

```
1  p1: process(a, b)        -- 조합논리의 process문(감지 리스트: a, b)
2       begin
3            out1  <= a and b;  -- 입력의 변화가 있으면, 바로 출력 out1에 전달
4       end process p1;         -- flip_flop이 합성
5  p2: process( clk )         -- 순서논리 process, 감지문는 clk
6       begin
7            if(clk'event and clk='1') then -- clk의 rising edge에서만 입력 값이 출력에 전달
8                 out2  <= a and b;    -- flip_flop이 합성
9            end if;
10     end process p2;
```


제2절 래치(latch) 회로의 설계

1. 래치회로의 모델링



D	EN	Q	$\overline{Q}$
X	0	Q	$\overline{Q}$
0	1	0	1
1	1	1	0

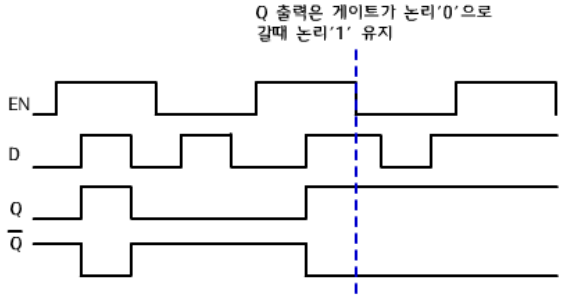


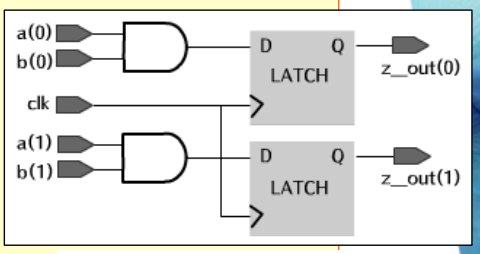
그림 7.3] D_latch (a) 기호 (b) 진리표 (c) 입출력 파형

제2절 래치(latch) 회로의 설계

2. 프로시저(procedure)를 활용한 래치회로

<VHDL 구문>

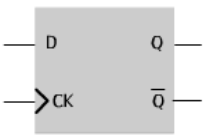
```
    library IEEE
    use IEEE.std_logic_1164.all;
    entity procedure_latch is
      port (clk : in std_logic;
        a, b : in std_logic_vector(1 downto 0);
        y : out std_logic_vector(1 downto 0));
    end procedure_latch;


    architecture latch_example of procedure_latch is
1       procedure latch(clk, a, b : in std_logic;
2                       signal z_out : out std_logic) is
3       begin
4           if clk = '1' then
5               z_out <= a and b;
6           end if;
        end latch;
8   begin
9     latch_1 : latch(clk, a(0), b(0), z_out(0));
10    latch_2 : latch(clk, a(1), b(1), z_out(1));
    end latch_example;
```



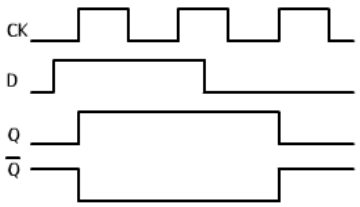
[그림 7.4] latch의 합성회로

제3절 Flip Flop의 설계

1. Flip_Flop의 기능



D	CK	Q	$\overline{Q}$
0	1	0	1
1	1	1	0
X	0	Q	$\overline{Q}$
X	1	Q	$\overline{Q}$

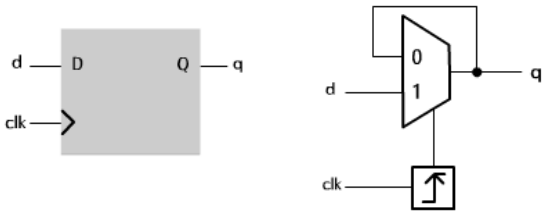


(a) 기호 (b) 진리표 (c) clk에 의한 입・출력 파형

[그림 7.8] Edge trigger D Flip_Flop

제3절 Flip Flop의 설계

2. D Flip_Flop



[그림 7.9] D Flip_Flop과 MUX등가회로

<VHDL구문>

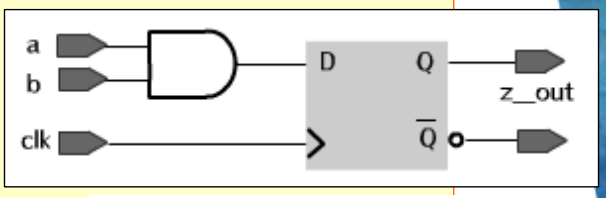
```
1   process (clk)
2    begin
3     if (clk'event and clk='1') then    -- clk의 상승 에지에서
4        q <= d;                         -- d입력이 q에 전달
5        end if;
6   end process;
```


제3절 Flip Flop의 설계

2. if문을 사용한 하나의 Flip_Flop 설계

<VHDL 구문>

```
   library IEEE;
   use IEEE.std_logic_1164.all;
   entity FlipFlop_EX1 is
     port (clk, a, b : in std_logic;
          z_out : out std_logic);
   end FlipFlop_EX1;


   architecture FlipFlop_example of FlipFlop_EX1 is
    begin
1    process (clk)
2     begin
3      if clk'event and clk = '1') then
4       z_out <= a and b;
5      end if;
6    end process;
   end FlipFlop_example;
```



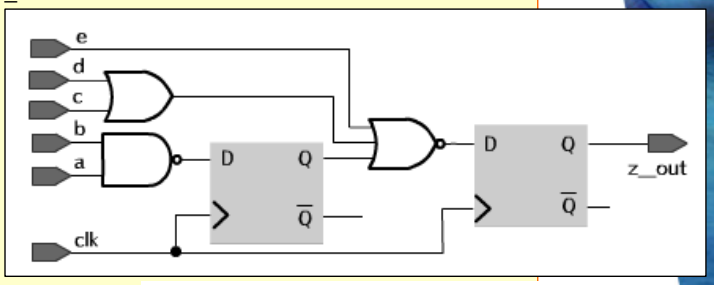
[그림 7.10] Flip_Flop의 합성결과

제3절 Flip Flop의 설계

3. if문을 사용한 두 개의 Flip_Flop 설계

<VHDL 구문>

```
    library IEEE;
    use IEEE.std_logic_1164.all; IEEE.numeric_std.all;
    entity FlipFlop_EX2 is
      port (clk, a, b, c, d, e : in std_logic;
          z_out : out std_logic);
    end FlipFlop_EX2;


    architecture combi_FlipFlop of FlipFlop_EX2 is
1       signal M : std_logic;
2     begin
3     process (clk)
4       variable N : std_logic;
5      begin
6       if rising_edge(clk) then
7         M <= (a nand b);
8         N := (c or d);
9         z_out <=  not (M or N or e);
10       end if;
11    end process;
    end combi_FlipFlop;
```



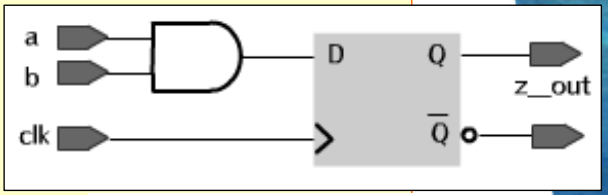
[그림 7.11] Flip_Flop의 합성결과

제3절 Flip Flop의 설계

4. wait until문을 사용한 Flip_Flop 설계

<VHDL 구문>

```
    library IEEE;
    use IEEE.std_logic_1164.all;
    entity FlipFlop_EX3 is
      port (clk, a, b : in std_logic;
          z_out : out std_logic);
    end FlipFlop_EX3;

    architecture FlipFlop_example of FlipFlop_EX3 is
     begin
1     process
2       begin
3        wait until clk'event and clk = '1';
4          z_out <= a and b;
5     end process;
    end FlipFlop_example;
```



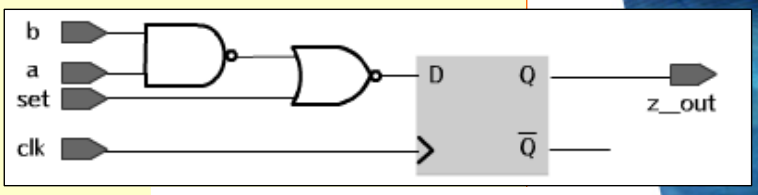
[그림 7.12] Flip_Flop의 합성결과

제3절 Flip Flop의 설계

5. 동기식 SET기능의 Flip_Flop 설계

<VHDL 구문>

```
   library IEEE;
   use IEEE.std_logic_1164.all; IEEE.numeric_std.all;
   entity FlipFlop_EX4 is
     port (clk, set, a, b : in std_logic;
           z_out : out std_logic);
   end FlipFlop_EX4;


   architecture FlipFlop_example of FlipFlop_EX4 is
    begin
1     process (clk)
2       begin
3        if (clk'event and clk = '1') then
4          if set = '1' then
   5          z_out <= '1';
6          else
7           z_out < a and b;
8          end if;
9        end if;
10     end process;
   end FlipFlop_example;
```



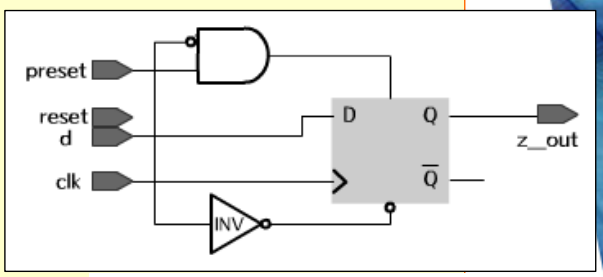
[그림 7.13] Flip_Flop의 합성결과

제3절 Flip Flop의 설계

6. 비동기식 RESET기능의 Flip_Flop 설계

<VHDL 구문>

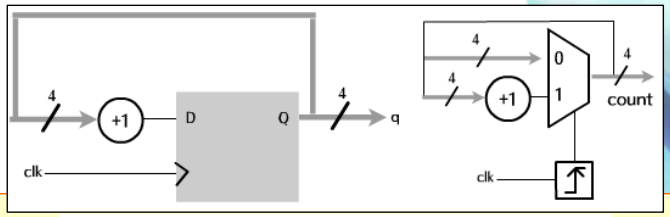
```
   library IEEE;
   use IEEE.std_logic_1164.all; IEEE.numeric_std.all;
   entity FlipFlop_EX5 is
     port (clk, reset, preset, d : in std_logic;
           z_out : out std_logic);
   end FlipFlop_EX5;

   architecture FlipFlop_example of FlipFlop_EX5 is
    begin
1     process (clk, reset, preset)
2       begin
3         if reset = '1' then
4           z_out <= '0';
5         elsif preset ='1' then
6           z_out <= '1';
7         elsif clk'event and clk = '1') then
8           z_out <= 'd';
9         end if;
10     end process;
   end FlipFlop_example;
```



[그림 7.14] Flip_Flop의 합성결과

제4절 카운터(Counter)의 설계

1. 기본 카운터(counter)의 설계



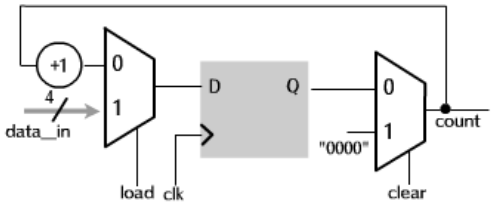
[그림 7.15] (a)16진 카운터 (b)MUX등가표현

<VHDL 구문>

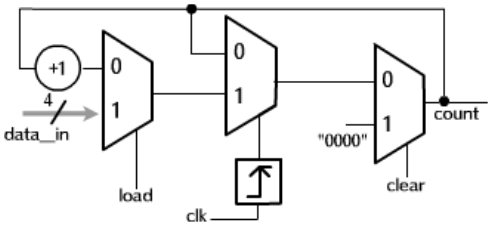
```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use WORK.std_logic_arith.all;        -- VHDL 산술연산자에 의존하는 패키지
4  entity COUNT16_EX is
5      port (  clk    : in std_logic;
6              count : buffer std_logic_vector(3 downto 0));  -- 증가를 위해 count의 mode는 buffer로 지정
7  end COUNT16_EX;
8  architecture COUNT_example of COUNT16_EX is
9   begin
10      process (clk)
11       begin
12        if (clk'event and clk='1') then  -- clk의 상승시점에서
13          count <= count+1;              -- count가 1씩 증가 됨
14        end if;
15     end process;
   end COUNT_example;
```


제4절 카운터(Counter)의 설계

2. Load/Clear 기능의 카운터 설계



[그림 7.16] (a)Load/Clear기능의 카운터



[그림 7.16] (b) MUX의 등가 표현

제4절 카운터(Counter)의 설계

<VHDL 구문>

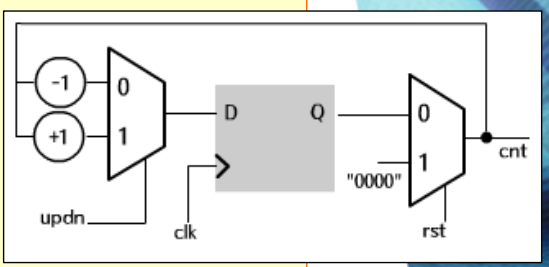
```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.all;
3  use WORK.STD_LOGIC_ARITH.all;
4  entity COUNT16_EX is
5    port ( clk, clear, load : in std_logic;
6           data_in : in std_logic_vector(3 downto 0);
7           count   : buffer std_logic_vector(3 downto 0));
8  end COUNT16_EX;
9  architecture LOCL_example of COUNT16_EX is
10   begin
11    process (clk, clear)                  -- clear가 감지신호 리스트문에 포함
12     begin
13      if clear='1'  then
14       count <= "0000";
15      elsif (clk'event and clk='1') then  -- clk의 상승시점에서
16       if  load='1' then                  --load가 '1'상태이면
17       count <= data_in;                  -- count에 입력 data_in을 대입
18        else                              --그렇지 않으면,
19        count <= count+1;                  -- count에 1증가를 대입
20       end if;
21     end if;
22     end process;
23   end LOCL_example;
```


제4절 카운터(Counter)의 설계

3. UP/DOWN 기능의 카운터 설계

<VHDL 구문>

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.all; use IEEE.STD_LOGIC_UNSIGNRD.all;
3   entity UPDN_EX is
4        port ( rst,clk,updn  : in std_logic;
5             cnt     : buffer std_logic_vector(3 downto 0));
6   end UPDN_EX;
7   architecture updncount_example of UPDN_EX is
8     begin
9       process (rst, clk)
10            begin
11              if rst='1' then
12                cnt <= (others=>'0');
13              elsif rising_edge(clk) then  -- clk의 상승 시점에서
14                if  updn='1'  then         -- updn=1 이면,
15                   cnt <= cnt+1 ;          -- cnt의 1씩 증가
16                else                       -- 그렇지 않으면,
17                   cnt <= cnt-1;           -- cnt의 1씩 감소
18                end if;
19              end if;
20          end process;
21   end updncount_example;
```



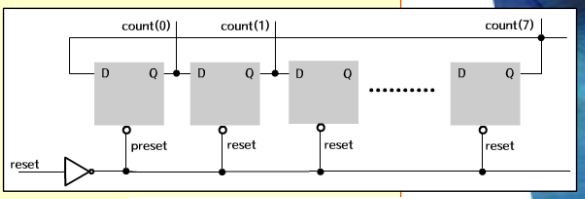
[그림 7.17] UP/DOWN 카운터 회로

제4절 카운터(Counter)의 설계

4. Ring_Counter 카운터의 설계

<VHDL 구문>

```
1    library IEEE;
2    use IEEE.STD_LOGIC_1164.all; use IEEE.STD_LOGIC_UNSIGNED.all;
3    entity RING_COUNTER is
4      port (clk, reset : in std_logic;
5           count : buffer std_logic_vector(7 downto 0));
6    end RING_COUNTER;
7    architecture ring_example of RING_COUNTER is
8      begin
9       process (clk, reset)
10       begin
11          if reset = '1' then
12           count <= "00000001";
13          elsif (clk'event and clk= '1') then
14              count <= count(6 downto 0) & count(7);
15          end if;
16      end process;
17     end ring_example;
```



[그림 7.18] Ring_Counter