Chapter

06

디지털 시계 설계

MSL의 설계 _01

DCL의 설계 _02

BSF의 생성과 BDF/Schematic 설계 _03

TL의 설계 _04

AL의 설계 _05

7-세그먼트 드라이버 설계 _06

부가적인 기능의 추가 _07

디지털 시계의 실행 _08

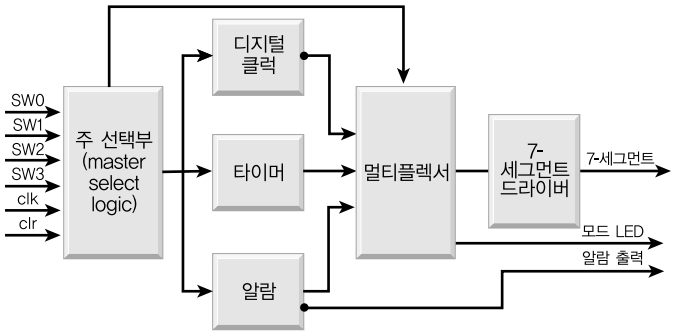
학습목표

- 조합논리회로와 순차논리회로를 사용하여 디지털 회로를 설계하는 프로젝트 과정을 알아본다.
- 디지털 시계의 기능을 정의하고 기능별로 설계한 후 통합 테스트하는 과정을 알아본다.
- 디지털 시계의 설계를 통해 계층적인 설계 과정을 이해한다.

들어가기에 앞서

앞에서는 일반적인 디지털 논리회로를 조합논리회로와 순차회로로 구분하여 VHDL로 설계하는 방법을 설명하였다. 디지털 시계를 설계하는 것은 앞에서 배운 내용들을 종합적으로 실습해 볼 수 있는 가장 좋은 예이다.

디지털 시계의 기능은 크게 디지털 클럭digital clock, 타이머timer 및 알람alarm 기능으로 구성되며 타이머는 스톱워치 기능을 제공한다. 세 기능은 멀티플렉서에 의해 하나가 선택되어 7-세그먼트로 출력되며, 멀티플렉서의 선택 신호는 MSLMaster Select Logic에서 제공한다. 디지털 시계의 기본 블록도는 [그림 6-1]과 같다. 앞에서 설명한 각 기능은 VHDL의 독립된 블록으로 설계된 후 BSFBlock Symbol File로 생성되어 최상위 레벨 설계top level design 단계에서 Block Diagram/Schematic File을 이용하는 설계 방법으로 생성된 BSF 파일들을 연결하여 설계한다. [그림 6-1]은 설계하고자 하는 디지털 시계의 블록도이다.



[그림 6-1] 디지털 시계의 블록도

[표 6-1]은 디지털 시계의 동작모드를 보여준다. 동작모드는 일반적인 디지털 시계의 기능과 매우 유사하게 동작하며 [표 6-1]에서는 동작모드의 동작과 각 모드에서 입력 스위치의 기능을 설명하고 있다. 모드는 선택 스위치 SW0를 누를 때마다 다음 동작모드로 변경이 된다.

초기상태는 모드 0이며, 일반적인 디지털 시계에서 현재의 시간을 보여주는 동작을 한다. 이 모드에서는 선택 스위치 SW1, SW2를 눌러도 다른 변화는 없으며 SW3를 누를 때마다 알람을 ON/OFF한다. SW3는 모든 모드에서 알람을 ON/OFF시킨다. 모드 1에서는 현재의 시간을 설정하는 기능을 제공한다. 이 모드에서 SW1은 설정할 시간의 위치(시, 분 또는 초)를 선택하고, SW2는 선택된 시간 위치의 값을 하나씩 증가시킨다. 알람모드인 모드 2에서 SW1과 SW2도 모드 1에서와 마찬가지로 시간 위치를 선택하고 시간을 하나씩 증가시키는 기능을 제공하나 현재의 시간을 설정하는 것이 아니라 알람시간을 설정하기 위해 사용된다. 마지막으로 모드 3에서는 타이머 동작을 하며 SW1은 누를 때마다 타이머가 스타트/스톱start/stop하며, SW2는 초기값으로 리셋시킨다.

[표 6-1] 디지털 시계의 스위치와 동작모드

모드 선택(SW0)		선택1(SW1)	선택2(SW2)	선택3(SW3)
0	시계모드	현재 시간 출력		알람 On/Off
1	시간 설정	위치 선택 (Hour, Min)	시간 증가	
2	알람 설정	위치 선택 (Hour, Min)	시간 증가	
3	타이머(스톱워치)	스타트/스톱 (Start/Stop)	리셋	

디지털 시계의 설계는 처음부터 모든 기능을 블록별로 설계한 후 통합하는 계층적 설계를 하는 것이 아니라, 시간 설정을 포함하는 시계의 기본 기능을 설계해서 동작을 확인한 후에 이를 기능별로 분리해서 블록별로 설계하는 과정을 따른다. 설계 과정의 자료는 DigitalWatch\DigitalWatchDesign 폴더에서 찾을 수 있으며, DigComV32에 다운로드하여 실행시켜서 동작을 확인하면서 설계를 완성한다.

01 MSL의 설계

MSL Master Select Logic은 디지털 시계, 타이머 및 알람 등 세 가지 기능 가운데 하나를 선택해서 출력시키는 기능을 제공한다. 이 기능에는 디지털 시계의 시간을 설정하고, 타이머의 스타트/스톱 및 리셋 기능을 제공한다.

1.1 클럭의 분주

외부에서 제공되는 클럭은 1MHz이며, 디지털 클럭 로직(DCL)과 알람 로직(AL)은 1Hz를 사용하고, 타이머 로직(TL)은 1/100초 단위로 측정이 되므로 100Hz를 사용한다. 따라서 1MHz 클럭을 1,000,000 분주해서 1Hz 클럭을 DCL과 AL에 제공하고 10,000 분주해서 100Hz 클럭을 TL에 제공한다. 또한 부가 기능을 추가할 때 설계할 시간 설정과 알람 설정 과정에서 설정값이 자동으로 증가하게 하기 위해 2Hz의 클럭도 발생시켜야 한다.

코드 6-1

```
-- 1Hz clock generation, ClockDivide.vhd
      process(clk, clk1hz)
            variable    cnt1hz            : integer := 0;
      begin
            if rising_edge(clk) then
                  if cnt1hz >= 499999 then
                        cnt1hz := 0;
                        clk1hz <= not clk1hz;
                  else
                        cnt1hz := cnt1hz + 1;
                  end if;
            end if;
      end process;
```

[코드 6-1]은 1Hz 클럭을 발생시키는 부분이다. 디지털 시계에서 사용된 클럭 분주는 1MHz 클럭의 상승에지에서 카운터를 499,999까지 카운트하다가 500,000번째 클럭에서 출력 클럭을 '0'에서 '1'로 또는 '1'에서 '0'으로 바꿔서 출력한다. 클럭을 분주하는 회로는 DigComV32

실습 키트에서 제공하는 1MHz 클럭을 FPGA의 입력으로 할당하고 출력을 LED에 할당하여 LED가 1초마다 ON/OFF하는 것을 확인할 수 있다(DigitalWatch\DigitalWatchDesign\ClockDevide 프로젝트 참고). 나머지 클럭도 카운터의 값을 변경하여 동일한 방법으로 발생시킬 수 있다.

1.2 모드의 선택

디지털 시계에는 앞에서 설명한 것과 같이 네 가지 모드가 있으며, 각 모드에서는 [표 6-2]와 같은 동작을 한다.

[표 6-2] 디지털 시계의 동작모드

모드	동작
0	현재의 시간을 출력
1	시간을 설정
2	타이머(스톱워치)를 출력
3	알람시간을 출력

MSL에서는 현재의 모드를 저장하는 신호와 SW0 스위치값을 저장하는 sw0_node signal을 선언하고, SW0 스위치값이 clk에 동기가 되어 sw0_node에 저장된다. 따라서 SW0 스위치가 한 번 눌릴 때마다 sw0_node의 상승에지에 동기가 되어 모드가 순환적으로 변경되도록 한다. 2비트 mode signal을 선언하고, 스위치를 한 번 누를 때마다 하나의 클럭으로 인식하여 mode signal을 하나씩 증가시킨다. Mode signal이 "00"→"01"→"10"→"11"로 증가한 후 다시 "00"이 되며, 각각의 값은 모드 0~모드 3에 해당한다. 또한 현재의 모드 정보는 DCL[Digital Clock Logic], TL[Timer Logic] 및 AL[Alarm Logic]로 전달되며, 멀티플렉서의 선택 입력 신호로 사용되므로 멀티플렉서로도 전달된다.

코드 6-2

```
-- modeSelect.vhd
      signal mode       : std_logic_vector(1 downto 0);
      signal sw0_node         : std_logic;

      process(reset, clk, sw0)
      begin
            if reset= '0' then
                  sw0_node <= '0';
```

```
        elsif rising_edge(clk) then
              sw0_node <= sw0;
        end if;
  end process;

  process(reset, sw0_node)
  begin
        if reset = '0' then
              mode <= "00";
        elsif rising_edge(sw0_node) then
              mode <= mode + '1';
        end if;
  end process;
```

[코드 6-2]는 푸시버튼 스위치push button switch를 입력으로 할당하고 2개의 LED를 출력으로 할당하여 실습키트에 다운로드해 실행시킨다. 스위치를 한 번 누를 때마다 두 개의 LED가 "00"→"01"→"10"→"11"을 반복한다(DigitalWatch\DigitalWatchDesign\modeSelect 프로젝트 참고).

02 DCL의 설계

DCL Digital Clock Logic에는 현재의 시간을 저장하는 레지스터가 선언되며, 모드 0에서는 1Hz의 클럭에 동기되어 1초씩 증가하여 출력한다. 그러나 모드 1에서는 증가하는 동작을 멈추고 새로운 시간을 설정하기 위해 SW1와 SW2를 이용해서 시 또는 분값을 설정한다.

2 .1 모드 0에서 시계 동작

모드 0에서는 리셋 스위치가 눌려지면 모든 시간은 초기화되며, 정상적으로 동작할 때는 1초 단위로 증가하며 현재의 시간을 출력한다. 시, 분, 초 단위로 시간을 저장하는 레지스터를 선언하며, 시는 최대 23이므로 5비트를 선언하고, 분과 초는 최대 59이므로 6비트를 선언해야 한다. signal로 선언된 레지스터들은 1Hz 클럭에 동기되어 초 단위 레지스터를 하나씩 증가시키다가 60초가 되면 분 레지스터를 증가시키고 초 레지스터는 0으로 초기화한다. 같은 방법으로 분 레지스터가 60분이 되면 시간 레지스터가 하나 증가하고 분 레지스터는 0으로 초기화한다.

코드 6-3

```
        signal  hour      : std_logic_vector(4 downto 0) := "00000";
        signal  min, sec  : std_logic_vector(5 downto 0) := "000000";

  -- process for second
        process(reset, clk1hz, sec)
        begin
                if reset = '0' then
                    sec <= "000000";
                elsif rising_edge(clk1hz) then
                    if conv_integer(sec) >= 59 then
                          sec <= "000000";
                    else
                          sec <= sec + '1';
                    end if;
                end if;
        end process;

  -- process for minute
```

```
        process(reset, clk1hz, sec, min)
        begin
                if reset = '0' then
                    min <= "000000";
                elsif rising_edge(clk1hz) then
                    if conv_integer(sec) >= 59 then
                          if conv_integer(min) >= 59 then
                              min <= "000000";
                          else
                              min <= min + '1';
                          end if;
                    end if;
                end if;
        end process;

  -- process for hour
        process(reset, clk1hz, sec, min, hour)
        begin
                if reset = '0' then
                    hour <= "00000";
                elsif rising_edge(clk1hz) then
                    if conv_integer(sec) >= 59 then
                          if conv_integer(min) >= 59 then
                              if conv_integer(hour) >= 23 then
                                      hour <= "00000";
                              else
                                      hour <= hour + '1';
                              end if;
                          end if;
                    end if;
                end if;
        end process;
```

[코드 6-3]에서 시간값은 초와 분이 59일 때에만 1Hz의 클럭에 맞춰 하나 증가하고 초와 분값은 0으로 초기화된다.

2.2 시간 출력하기

현재의 시간을 7-세그먼트에 출력한다. 출력 부분은 따로 7-세그먼트 드라이버(7SD)[7-Segment Driver] 부분에서 구현하게 되어 있으나 시계의 동작을 확인하기 위해서 이 부분에서 설명한다. 시간을 출력하기 위해서는 시, 분, 초 signal에 저장된 2진수 값을 7-세그먼트에 출력할 수 있도록 10

진수로 변환해야 한다. 예를 들어 초값은 최대 59이므로 6비트에 저장될 수 있으며, 48초는 6비트에 저장된 2진수 "101100"을 두 개의 BCD "0100"과 "1000"으로 변환해야 한다. 즉 하나의 7-세그먼트에는 4비트의 값을 출력할 수 있으므로, 10진수 48로 출력하기 위해서는 4비트에 10진수 4와 8로 각각 변환한 후 7-세그먼트 디코드를 해야 한다. 2진수를 10진수로 변환하는 부분은 [코드 6-4]와 같다.

코드 6-4

```
        signal      hour_buf0, hour_buf1     : integer := 0;
        signal      min_buf0, min_buf1       : integer := 0;
        signal      sec_buf0, sec_buf1       : integer := 0;

  -- process for converting to integer type
        process(hour, min, sec)
        begin
              hour_buf0 <= conv_integer(hour)/10;
              hour_buf1 <= conv_integer(hour) mod 10;
              min_buf0 <= conv_integer(min)/10;
              min_buf1 <= conv_integer(min) mod 10;
              sec_buf0 <= conv_integer(sec)/10;
              sec_buf1 <= conv_integer(sec) mod 10;
        end process;
```

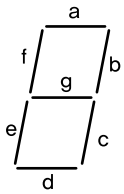
[코드 6-4]에서 standard logic으로 선언된 hour signal에는 출력하고자 하는 시간이 저장되어 있으며, 논리형으로 선언되었으므로 산술연산의 피연산자로 사용할 수 없다. 따라서 conv_integer()함수를 이용해서 정수로 변환한 후 산술연산을 해야 한다. 시간을 10자리와 1자리로 나누기 위해 연산자 '/'를 이용해서 정수 10으로 나눈 몫을 hour_buf0에 저장하고, modulo 연산자 'mod'를 이용해서 10의 modulo를 hour_buf1에 저장한다. 나머지 분과 초도 같은 방법으로 처리한다. BCD로 변환된 시간값은 7-세그먼트에 출력하기 위해 디코딩되어야 한다. 아래는 10자리와 1자리로 분리된 시간을 7-세그먼트에 출력하는 부분을 보여준다. hour_buf0에 저장된 값에 따라 그 값을 표시하기 위해 해당하는 세그먼트를 출력시킨다.

코드 6-5

```
  --    process for 7-segment display
        process(clk, hour_buf0, hour_buf1, min_buf0, min_buf1, sec_buf0, sec_buf1)
        begin
              if rising_edge(clk) then
                    case hour_buf0 is         -- abcdefg-
                          when 0 => seg_hour0 <= "1111110";
                          when 1 => seg_hour0 <= "0110000";
                          when 2 => seg_hour0 <= "1101101";
                          when 3 => seg_hour0 <= "1111001";
```

```
                when 4 => seg_hour0 <= "0110011";
                when 5 => seg_hour0 <= "1011011";
                when 6 => seg_hour0 <= "1011111";
                when 7 => seg_hour0 <= "1110000";
                when 8 => seg_hour0 <= "1111111";
                when 9 => seg_hour0 <= "1110011";
                when others => null;
            end case;
```

7-세그먼트는 [그림 6-2]와 같다. 하나의 숫자를 출력하는 예로 3을 출력하기 위해서는 "abcdefg"에 "1111001"을 출력해야 한다[DigitalWatch\DigitalWatchDesign\DigitalClock 프로젝트 참고].



[그림 6-2] 7-세그먼트의 각 세그먼트 위치

2.3 시간 설정위치의 지정

설정할 시간의 위치를 저장하기 위해서 set_sec, set_min 및 set_hour signal을 선언하고 SW1 스위치가 눌릴 때마다 설정위치를 나타내는 signal이 순환적으로 ON 된다.

코드 6-6

```
signal  set_time      : std_logic_vector := "100";

process(reset, sw1)
begin
        if reset = '0' then
            set_time <= "100";
        elsif rising_edge(sw1) then
            case set_time is
                when "100" =>
                    set_time <= "010";
                when "010" =>
                    set_time <= "001";
                when "001" =>
                    set_time <= "100";
                when others =>
```

```
                        set_time <= null;
                end case;
            end if;
        end process;
```

[코드 6-6]에서 SW1이 입력될 때마다 설정할 시간의 위치를 변경하기 위해 SW1을 눌러서 한 개의 클럭을 발생시키고 클럭의 상승에지에서 설정 위치가 바뀌도록 한다. 키트에서 실행시키면 SW1을 누를 때마다 LED가 순서대로 켜지는 것을 확인할 수 있다[DigitalWatch\DigitalWatch Design\timeSetPosition 프로젝트 참고].

2.4 모드 1에서 시간 설정

모드 1에서는 초 단위로 증가하는 시계 동작을 멈추고 SW2이 눌리면 시, 분, 초 단위로 새로운 시간을 설정한다. 이를 위하여 시간 설정모드(모드 1)에서는 1Hz의 클럭 대신에 SW2 스위치를 클럭소스로 사용한다.

코드 6-7

```
-- process for clock source
      process(clk1hz, mode, sw2)
      begin
            if mode = "01" then
                  timeClock <= sw2;
            else
                  timeClock <= clk1hz;
            end if;
      end process;
```

다음 시계 동작 VHDL 코드를 [코드 6-8]과 같이 추가 수정한다.

코드 6-8

```
-- process for second
      process(reset, timeClock)
      begin
            if reset = '0' then
                  sec <= "000000";
            elsif rising_edge(timeClock) then
                  if mode = "01" then              -- time set mode
                        if set_time = "001" then   -- second time set
                              if conv_integer(sec) >= 59 then
```

```
                                    sec <= "000000";
                              else
                                    sec <= sec + '1';
                              end if;
                        end if;
                  else
                        if conv_integer(sec) >= 59 then
                              sec <= "000000";
                        else
                              sec <= sec + '1';
                        end if;
                  end if;
            end if;
      end process;
```

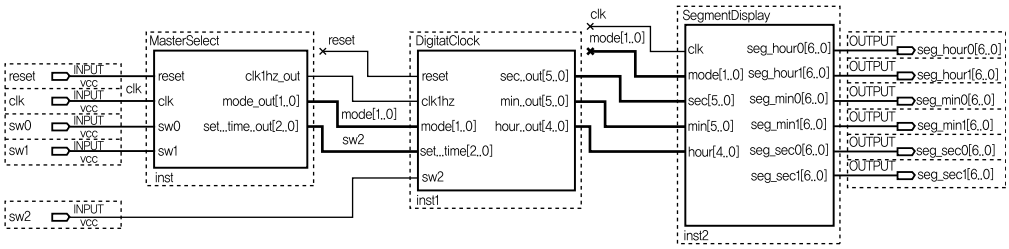
[코드 6-8]은 초 단위 시간을 설정할 때 추가한 부분이며, 시간 설정모드에서 초 단위의 시간을 설정하는 부분을 추가한 것이다. 분 단위와 시 단위도 동일하게 수정한다. 그러나 시 단위를 설정할 때는 23에서 다시 0으로 초기화된다[DigitalWatch\DigitalWatchDesign\Clock Setting 프로젝트 참고].

03 BSF의 생성과 BDF/Schematic 설계

앞에서 설명한 부분들은 디지털 시계의 전체 블록도에서 MSL과 DCL에 관한 부분이며, 일부 7-세그먼트 드라이버(7SD)를 포함하고 있다. 앞에서는 모든 기능을 하나의 모듈로 작성했기 때문에 굳이 BDF/Schematic 설계를 할 필요가 없다. 코드의 길이가 짧은 경우에는 기능 단위로 구분하지 않고 하나의 모듈로 설계하는 것이 편리하나 설계의 양이 많아지면 기능별로 구분을 해서 블록단위로 설계하는 것이 바람직하고 수정하기에도 용이하다. 따라서 위에서 설계한 내용을 MSL, DCL 및 7SD의 BSF를 생성하고, 생성된 BSF를 이용하여 BDF/Schematic 설계를 한다.

BSF를 생성하기 위해서는 앞에서 설명한 클럭 분주와 모드선택 VHDL 코드를 하나의 MSL BSF로, 시계 동작 부분과 시간 설정 부분을 DCL BSF로 생성하고, 출력하는 부분을 7SD BSF로 생성한다. 추가적으로 Timer Logic과 Alarm Logic을 설계하는 과정에서 출력 부분에서 추가되어야 할 기능은 나중에 추가로 구현한다.

위에서 설명한대로 기능별로 블록을 나눠서 설계를 했을 때 최상위 설계의 블록도는 [그림 6-3]과 같다.



[그림 6-3] MSL, DCL 및 7SD의 블록도

[그림 6-3]의 설계는 전체 블록도에서 Timer Logic과 Alarm Logic을 제외한 부분이며, 각 BSF는 VHDL로 설계되어 있어 마우스로 더블클릭을 하면 VHDL 소스를 볼 수 있어 코드를 수정할 수 있다[DigitalWatch\DigitalWatchBDF\DigitalWatch 프로젝트 참고].

04 TL의 설계

이제부터는 필요한 로직을 앞에서 설계한 BDF/Schematic 설계에 추가하는 방법으로 설계를 진행한다. 타이머 로직(TL) Timer Logic을 설계할 때 다른 부분에서 발생되는 출력 신호 중에서 필요한 신호를 입력받아 VHDL로 설계한 후, BSF를 생성해서 BDF/Schematic 설계에 추가한다.

4.1 1/100초 단위 시간 증가 동작

TL의 가장 기본적인 동작은 1/100초 단위로 시간을 증가시키는 것이다. 따라서 100Hz 클럭을 입력받아서 하나의 클럭이 들어올 때마다 1/100초씩 증가시키고 그 값을 출력하는 것이다.

코드 6-9

```
signal  mmsec              : std_logic_vector(6 downto 0) := "0000000";
-- process for 1/100 second
        process(reset, clk100hz)
        begin
            if reset = '0' then
                  mmsec <= "0000000";
            elsif rising_edge(clk100hz) then
                  if conv_integer(mmsec) >= 99 then
                        mmsec <= "0000000";
                  else
                        mmsec <= mmsec + '1';
                  end if;
            end if;
        end process;
```

[코드 6-9]에서 mmsec는 1/100로 단위의 시간을 저장하는 signal이며, 100Hz의 클럭에 동기가 되어 10msec씩 증가하고 99를 카운트하면 다시 0으로 초기화되며 1초 증가한다.

4.2 리셋과 스타트, 스톱 기능 추가

TL의 모드 3 상태에서 현재 1/100초 단위로 카운트 동작상태에 있으면 SW1을 눌렀을 때 증가

동작으로 멈추고, 중지상태이면 다시 카운트 동작을 해야 한다. 또한 중지상태에서 SW2를 누르면 모든 카운터를 리셋하여 초기화한다. 이를 위하여 현재의 상태(카운트상태 또는 정지상태)를 저장하는 signal을 선언하고 이 신호의 값에 따라 동작을 제어한다.

코드 6-10

```
        signal  state     : std_logic;   -- 0 : stop state, 1 : up count state
  -- process for state control
        process(reset, mode, sw1)
        begin
                if reset = '0' then
                    state <= '0';-- stop state
                elsif rising_edge(sw1) then
                    if mode = "11" then
                          if state = '0' then
                              state <= '1';
                          else
                              state <= '0';
                          end if;
                    end if;
                end if;
        end process;
```

state signal의 초기값은 '0'으로 카운트 정지상태이며, 현재의 동작모드가 타이머모드(mode = "11") 이면, SW1을 누를 때마다 현재의 상태가 카운트상태이면 정지상태로, 정지상태이면 다시 카운트상태로 바뀐다. 또한 1/100초 단위로 증가하는 시간 부분을 [코드 6-11]과 같이 수정한다.

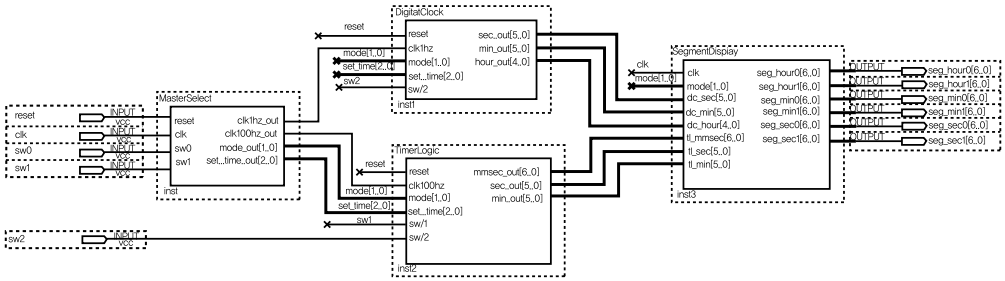
코드 6-11

```
        process(reset, clk100hz, state, mode, sw2)
        begin
                if reset = '0' then
                    mmsec <= "0000000";
                elsif state = '0' then
                    if mode = "11" then
                          if sw2 = '1' then
                              mmsec <= "0000000";
                          end if;
                    end if;
                else
                    if rising_edge(clk100hz) then
                          if conv_integer(mmsec) >= 99 then
                              mmsec <= "0000000";
                          else
```

```
                                mmsec <= mmsec + '1';
                            end if;
                        end if;
                    end if;
            end process;
```

[코드 6-11]에서 모드 3인 경우에 카운트가 정지된 상태(state = '0')에서 SW2를 누르면 시간이 리셋되며, 카운트상태에서는 모드에 상관없이 100Hz 클럭에 동기되어 시간을 증가시킨다.

TL을 포함한 전체 블록 설계도는 [그림 6-4]와 같다.



[그림 6-4] MSL, DCL, TL 및 7SD의 블록도

[그림 6-4]의 설계는 전체 디지털 시계에서 Alarm Logic을 제외한 부분이다. [그림 6-4]에서 SegmentDisplay 블록이 수정되어야 하며 수정되는 내용은 7-세그먼트 출력 부분에서 상세하게 설명한다[DigitalWatch\DigitalWatchBDF_Timer\Digitalwatch 프로젝트 참고].

05 AL의 설계

AL Alarm Logic에서는 알람 기능을 ON/OFF하기 위해 SW3를 사용한다. SW3를 누를 때마다 알람이 ON/OFF된다. 또한 시간 설정 기능과 마찬가지로 설정할 시간의 위치를 나타내는 신호를 MSL로부터 입력 받아 SW2를 이용하여 알람시간을 설정한다.

5.1 알람 ON/OFF 기능

코드 6-12

```
        signal  alarm         : std_logic:= '0';
  -- process for Alarm on/off
        process(reset, alarm, sw3)
        begin
                if reset = '0' then
                    alarm <= '0';
                    alarm_on <= "0000000";
                elsif rising_edge(sw3) then
                    if alarm = '0' then
                          alarm <= '1';
                          alarm_on <= "1111111";
                    else
                          alarm <= '0';
                          alarm_on <= "0000000";
                    end if;
                end if;
        end process;
```

SW3를 누를 때마다 알람 기능이 ON/OFF된다.

5.2 알람 시간 설정

알람 시간을 설정하는 것은 시간을 설정하는 과정과 거의 유사하다. 모드 2에서 해당 시간 단위

알람 설정이 인에이블 enable 되었을 때 SW2를 눌러서 알람 시간을 설정할 수 있다. SW2를 한 번 누를 때마다 시, 분, 초값이 하나씩 증가한다.

코드 6-13

```
-- process for second
      process(reset, sw2, mode, set_time)
      begin
            if reset = '0' then
                  sec <= "000000";
            elsif rising_edge(sw2) then
                  if mode = "10" and set_time = "001" then-- sec alarm set
                        if conv_integer(sec) >= 59 then
                              sec <= "000000";
                        else
                              sec <= sec + '1';
                        end if;
                  end if;
            end if;
      end process;
```

[코드 6-13]은 초 단위 알람 시간을 맞추는 것이며, 모드가 알람모드(mode = “10”)이고, 초 단위 시간을 설정하기 때문에 알람 설정 위치가 초 단위(set_time = “001”)여야 한다. SW2를 누를 때마다 1초씩 증가하며, 59 다음에는 다시 0으로 초기화된다. 분과 시의 설정도 동일하게 한다.

5.3 알람 기능

현재의 시간과 설정된 알람 시간이 일치할 경우 알람 신호를 출력한다. 알람 신호는 알람 기능이 ON 상태이고 알람 시간을 설정하는 모드가 아닐 경우에만 알람 신호를 출력한다.

코드 6-14

```
-- process for comparing between current time and alarm setting time
process(reset, clk100hz, mode, clk_hour, clk_min, hour, min, sec,alarm)
begin
      if rising_edge(clk100hz) then
            if hour = clk_hour then
                  if min = clk_min then
                        if mode /= "10" and alarm = '1' then
                              alarm_out <= "11111111";
                        else
```

```
                                alarm_out <= "00000000";
                        end if;
                else
                        alarm_out <= "00000000";
                end if;
        else
                alarm_out <= "00000000";
        end if;
    end if;
end process;
```

[코드 6-14]에서 현재 시간과 알람 설정 시간은 100Hz 클럭의 상승에지에서 비교되며, 분까지만 비교한다(시간 설정과 알람 설정은 초 단위까지 할 수 있지만 실제로 초 단위까지 시계를 맞추거나 알람을 맞추는 경우는 거의 없다). 그리고 알람 설정모드가 아니고 알람이 ON 상태일 때만(mode /= “10” and alarm = ‘1’) 알람 신호가 출력된다. 그리고 시간이 경과해서 분값이 달라지면 알람 신호는 자동적으로 꺼진다.

06 7-세그먼트 드라이버 설계

Digital clock logic을 설계하는 과정에서 현재의 시간을 출력하는 부분은 이미 설계되었다. 그러므로 시간 설정, 타이머 및 알람 시간을 출력시키기 위해서 앞에서 설계한 출력 부분을 수정한다. 기본적으로 모드 0과 모드 1에서는 디지털 클럭의 시간을 출력하고, 모드 2에서는 알람 로직의 출력을, 그리고 모드 3에서는 타이머 로직의 시간을 출력한다.

코드 6-15

```
-- process for converting to integer
process(mode, dc_hour, dc_min, dc_sec, al_hour, al_min, al_sec, tl_mmsec,
tl_sec, tl_min)
begin
      case mode is
            when "00" =>-- digital clock mode
                  hour_buf0 <= conv_integer(dc_hour)/10;
                  hour_buf1 <= conv_integer(dc_hour) mod 10;
                  min_buf0 <= conv_integer(dc_min)/10;
                  min_buf1 <= conv_integer(dc_min) mod 10;
                  sec_buf0 <= conv_integer(dc_sec)/10;
                  sec_buf1 <= conv_integer(dc_sec) mod 10;
            when "01" =>-- clock setting mode
                  hour_buf0 <= conv_integer(dc_hour)/10;
                  hour_buf1 <= conv_integer(dc_hour) mod 10;
                  min_buf0 <= conv_integer(dc_min)/10;
                  min_buf1 <= conv_integer(dc_min) mod 10;
                  sec_buf0 <= conv_integer(dc_sec)/10;
                  sec_buf1 <= conv_integer(dc_sec) mod 10;
            when "10" =>-- alarm mode
                  hour_buf0 <= conv_integer(al_hour)/10;
                  hour_buf1 <= conv_integer(al_hour) mod 10;
                  min_buf0 <= conv_integer(al_min)/10;
                  min_buf1 <= conv_integer(al_min) mod 10;
                  sec_buf0 <= conv_integer(al_sec)/10;
                  sec_buf1 <= conv_integer(al_sec) mod 10;
            when "11" =>-- timer mode
                  hour_buf0 <= conv_integer(tl_min)/10;
                  hour_buf1 <= conv_integer(tl_min) mod 10;
                  min_buf0 <= conv_integer(tl_sec)/10;
```

```
                    min_buf1 <= conv_integer(tl_sec) mod 10;
                    sec_buf0 <= conv_integer(tl_mmsec)/10;
                    sec_buf1 <= conv_integer(tl_mmsec) mod 10;
                when others => null;
        end case;
  end process;
```

[코드 6-15]에서는 디지털 클럭, 타이머 및 알람 블록으로부터 입력되는 시, 분 및 초를 모드에 따라서 정수형으로 변환해서 출력 버퍼에 저장하는 과정을 보여준다. 예로 알람모드(모드 3)에서는 알람 블록에서 입력되는 시간을 정수형으로 변환해서 출력 버퍼에 저장한다.

07 부가적인 기능의 추가

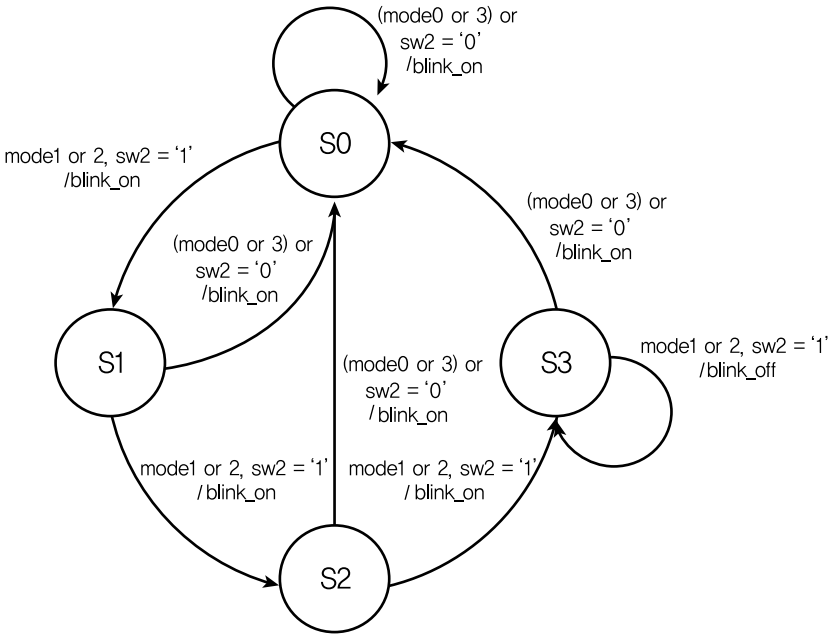
앞에서는 디지털 시계의 기본적인 동작만을 설계하였다. 그러나 일반적으로 동작하는 디지털 시계와 동일하게 보이기 위해서는 몇 가지 기능을 추가할 필요가 있다. 예를 들어서 현재 시간을 설정할 때 설정 대기시간 동안에 설정되는 시간 위치의 7-세그먼트가 일정한 간격으로 ON/OFF를 반복하며[blink], 설정을 위해 스위치를 누르는 동안에는 블링크를 멈추게 된다. 또한 시간 설정을 위해서 SW2를 누르면 시간이 증가하며, 일정한 시간 동안 누른 상태를 유지하면 시간이 자동으로 증가한다.

7.1 시간 설정 시 7-세그먼트 블링크 동작

이 기능은 7-세그먼트 출력에서 현재의 모드, 시간 설정의 위치 및 설정 스위치가 눌린 상태를 감지하여 일정한 간격으로 7-세그먼트를 ON/OFF하게 하며, 이러한 동작을 블링크[blink]라고 한다.

이 기능을 설계하는 것은 앞에서 설계한 것보다 좀 더 복잡할 수 있다. 우선 세그먼트가 블링크되는데 입력으로 고려해야 할 조건이 복잡하기 때문에 단순히 조합논리회로만으로 설계하는 것이 쉽지는 않다. 따라서 순차논리회로를 적용해서 설계를 한다. 순차논리를 사용하지 않고 설계할 수도 있겠지만 이 기회를 통해서 순차논리회로를 적용해서 설계하는 것도 매우 좋은 연습이 될 것이다.

먼저 signal blink를 정의한다. 이 signal은 세그먼트가 블링크되는 조건을 만족하지 못하면 OFF되며, 초기 조건은 블링크가 ON되는 것이다. 시간 설정모드(모드 1)또는 알람 설정모드(모드 2)가 되면 초기 조건은 blink = '1' 이므로 해당 설정 시간 단위의 세그먼트는 블링크를 시작한다. 이 상태에서는 시간 설정을 위해서 SW2를 누를 때마다 1씩 증가하지만 SW2를 4초 이상 누르고 있으면 블링크 동작을 멈추고 시간은 0.5초마다 1씩 자동적으로 증가하도록 한다. [그림 6-5]는 blink signal을 제어하기 위한 상태도를 나타낸다.



[그림 6-5] blink signal을 제어하기 위한 상태도

위 상태도는 1Hz 클럭에 동기되어 상태천이를 하며, 4개의 상태를 정의한다. 각 상태는 1초단위로 경과한 시간을 기준으로 정해진다. 즉, S0는 초기상태이며 S1은 1초가 경과한 상태이고, S2와 S3는 각각 2초와 3초가 경과한 상태를 나타낸다. 초기상태(S0)에서 모드 1(시간 설정) 또는 2(알람 설정)이고, SW2를 누르고 있으면 다음 상태로 천이하며, 이것은 시간 설정 또는 알람 설정 모드에서 4초 이상 누른 상태를 유지하면 블링크를 멈추게 하기 위해 1초 단위로 상태를 천이시키는 것이다. 따라서 모드가 0 또는 3이거나 SW2를 누른 상태가 아니라면 초기상태로 되돌아간다. 이와 같이 상태천이를 반복하여 S3가 된 후에 SW2가 눌린 상태면 블링크를 멈추고 계속 S3 상태를 유지하며, 그렇지 않을 때는 초기상태 S0로 되돌아간다. 위 상태도는 단지 blink signal 값을 결정하기 위한 것이며, blink가 '1' 이고 동시에 모드가 1 또는 2일 때만 해당 출력 장치가 블링크 동작을 한다는 것을 보여준다.

코드 6-16

```
        type state_type is (s0, s1, s2, s3);
        signal  state                        : state_type;

  -- process for 7-segment blink on/off
        process(reset, clk2hz, mode, set_time, blink, sw2)
        begin
                if reset = '0' then
                    blink <= '1';
                elsif rising_edge(clk2hz) then
                    case state is
```

```
                when s0 =>
                    -- time set or alarm set mode
                    if mode = "01" or mode = "10" then
                            if sw2 = '1' then
                                    state <= s1;
                                    blink <= '1';
                            else
                                    state <= s0;
                                    blink <= '1';
                            end if;
                    else
                            state <= s0;
                            blink <= '1';
                    end if;
                when s1 =>
                    if mode = "01" or mode = "10" then
                            if sw2 = '1' then
                                    state <= s2;
                                    blink <= '1';
                            else
                                    state <= s0;
                                    blink <= '1';
                            end if;
                    else
                            state <= s0;
                            blink <= '1';
                    end if;
                when s2 =>

                        이하 생략
```

[코드 6-16]에서 state_type의 signal을 선언하고 case~when 구문을 이용하여 위 상태도의 조건에 따라 상태천이를 하도록 한다.

실제로 블링크 동작이 일어나기 위해서는 segment display에서 출력하는 부분을 수정해야 한다. 우선 1초마다 ON/OFF를 반복하는 signal을 시, 분, 초 단위로 정의하고, 현재의 동작모드와 설정 시간 위치 조건이 맞을 경우에 블링크가 ON이면 이 signal들이 ON/OFF를 반복한다.

코드 6-17

```
-- process for display on/off
      process(clk2hz, mode, set_time, blink, disp_hour, disp_min, disp_sec)
      begin
              if rising_edge(clk2hz) then
                  if mode = "01" or mode = "10" then
```

```
                if blink = '1' then
                      case set_time is
                         when "100" =>
                                 disp_hour <= not disp_hour;
                                 disp_min <= '1';
                                 disp_sec <= '1';
                         when "010" =>
                                 disp_hour <= '1';
                                 disp_min <= not disp_min;
                                 disp_sec <= '1';
                         when "001" =>
                                 disp_hour <= '1';
                                 disp_min <= '1';
                                 disp_sec <= not disp_sec;
                         when others => null;
                      end case;
                else
                      disp_hour <= '1';
                      disp_min <= '1';
                      disp_sec <= '1';
                end if;
            else
                disp_hour <= '1';
                disp_min <= '1';
                disp_sec <= '1';
            end if;
      end if;
end process;
```

[코드 6-17]에서 모드가 시간 설정 또는 알람 설정모드(모드 1 또는 2)이고 블링크가 ON이면 해당하는 시간 위치의 display signal(disp_sec, disp_min, disp_hour)을 0.5초 간격으로 ON/OFF시킨다. 그러나 시간 설정모드 또는 알람 설정모드가 아니거나, blink signal이 ON이 아니면 display signal은 항상 ON 상태를 유지한다.

코드 6-18

```
            if rising_edge(clk100hz) then
                if disp_hour = '1' then
                      case hour_buf0 is --          abcdefg-
                            when 0 => seg_hour0 <= "1111110";
                            when 1 => seg_hour0 <= "0110000";
                            when 2 => seg_hour0 <= "1101101";
                            when 3 => seg_hour0 <= "1111001";
                            when 4 => seg_hour0 <= "0110011";
                            when 5 => seg_hour0 <= "1011011";
```

```
                        when 6 => seg_hour0 <= "1011111";
                        when 7 => seg_hour0 <= "1110000";
                        when 8 => seg_hour0 <= "1111111";
                        when 9 => seg_hour0 <= "1110011";
                        when others => null;
                    end case;

                    case hour_buf1 is --abcdefg-
                        when 0 => seg_hour1 <= "1111110";
                        when 1 => seg_hour1 <= "0110000";
                        when 2 => seg_hour1 <= "1101101";
                        when 3 => seg_hour1 <= "1111001";
                        when 4 => seg_hour1 <= "0110011";
                        when 5 => seg_hour1 <= "1011011";
                        when 6 => seg_hour1 <= "1011111";
                        when 7 => seg_hour1 <= "1110000";
                        when 8 => seg_hour1 <= "1111111";
                        when 9 => seg_hour1 <= "1110011";
                        hen others => null;
                    end case;
                else
                    seg_hour0 <= "0000000";
                    seg_hour1 <= "0000000";
                end if;
            end if;
```

앞에서 정의한 display signal이 ON이면 출력 버퍼에 저장된 데이터를 세그먼트에 출력하고 OFF이면 세그먼트에 아무 값도 출력하지 않는다. 따라서 0.5초 간격으로 블링크 동작을 한다.

7.2 시간 설정 시 시간의 자동 증가

이 기능을 추가하기 위해서는 블링크가 OFF된 시점에서부터 2Hz 클럭에 동기되어 설정시간이 1씩 증가되도록 한다. 이를 위하여 DCL부분에서 시간이 증가하거나 설정할 때 사용하는 클럭 소스를 정하는 부분을 [코드 6-19]와 같이 수정한다. 시간 설정모드(모드 1)에서 블링크가 OFF 상태이면 시간이 자동적으로 0.5초에 1씩 증가되도록 2Hz 클럭을 사용한다.

코드 6-19

```
-- process for clock source
process(clk1hz, clk2hz, mode, blink, sw2)
begin
    if mode = "01" then-- time set mode
        if blink = '1' then
```

```
                timeClock <= sw2;
            else
                timeClock <= clk2hz;
            end if;
        else-- clock mode
            timeClock <= clk1hz;
        end if;
    end process;
```

또한 alarm logic에서도 SW2에 의해서만 증가하던 시간을 블링크가 OFF되면 0.5초에 1씩 증가하도록 수정한다.

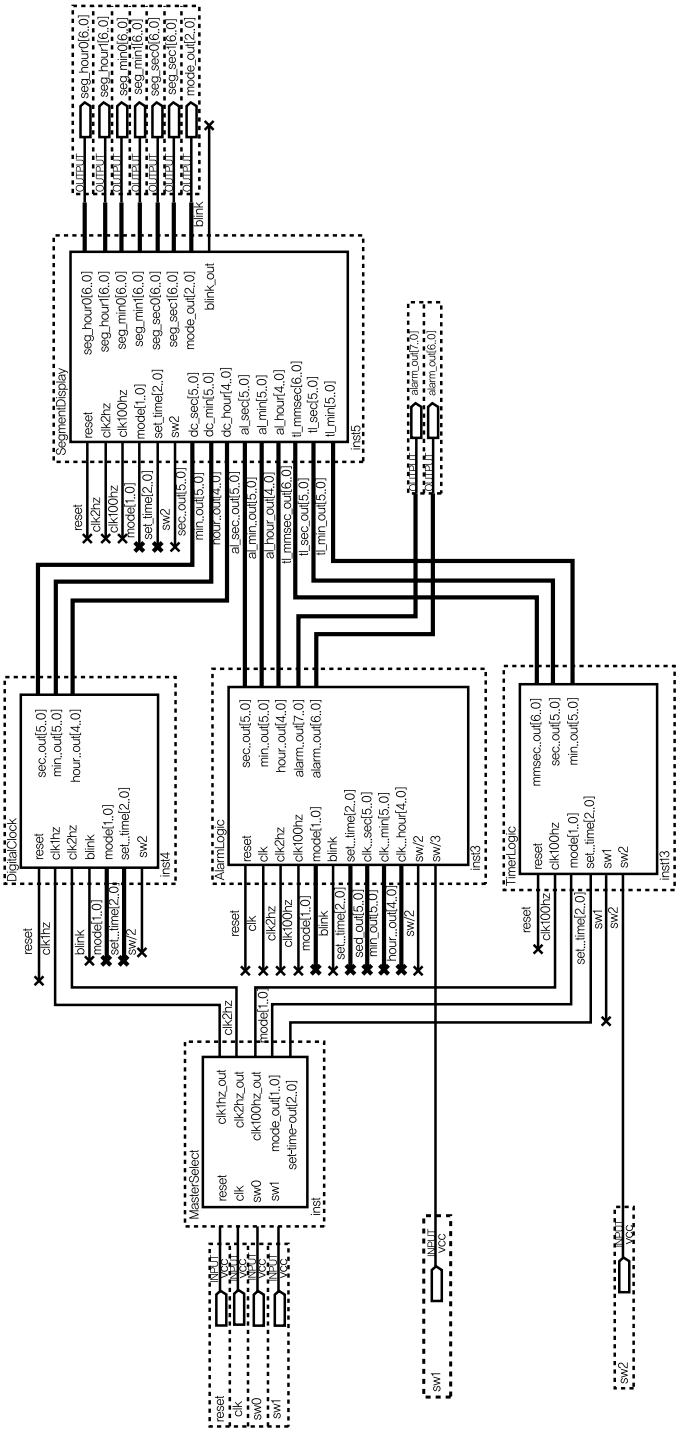
코드 6-20

```
-- process for Alarm clock source
    process(clk2hz, mode, blink, sw2)
    begin
        if mode = "10" then
            if blink = '1' then
                AlarmClock <= sw2;
            else
                AlarmClock <= clk2hz;
            end if;
        end if;
    end process;

-- process for second
    process(reset, AlarmClock, mode, set_time)
    begin
        if reset = '0' then
            sec <= "000000";
        elsif rising_edge(AlarmClock) then
            if mode = "10" and set_time = "001" then-- sec alarm set
                if conv_integer(sec) >= 59 then
                    sec <= "000000";
                else
                    sec <= sec + '1';
                end if;
            end if;
        end if;
    end process;
```

[코드 6-20]에서는 초 단위 설정하는 부분만 보여주고 있으며 분, 시간 단위도 동일하게 수정한다.

[그림 6-6]은 앞에서 추가한 기능을 모두 포함한 블록도이다.



[그림 6-6] 기능을 추가한 디지털 시계의 블록도

08 디지털 시계의 실행

지금까지 설명한 것처럼 디지털 시계를 설계하면 핀을 할당한 후에 다시 컴파일하여 실습키트에 다운로드하여 실행시켜야 한다. 입력으로 사용된 것은 1MHz 클럭, 리셋 스위치 및 선택스위치(SW0～SW3)이며, 출력으로 사용된 것은 시간값을 나타내는 6개의 7-세그먼트와 알람 출력 장치이다. 그러나 소리를 내는 출력 장치가 없으므로 알람 시간이 됐을 때 LED가 ON되도록 하였다. 또한 현재의 동작모드를 나타내는 mode 신호는 모드 표시 LED에 출력하여 현재의 모드를 알 수 있도록 하였다. [표 6-3]은 입력과 출력으로 사용된 장치를 보여준다[DigitalWatch\DigitalWatch\Digitalwatch 프로젝트 참고].

[표 6-3] 디지털 시계의 입출력 장치

입출력	입출력 장치
clk	1MHz 클럭
reset	리셋 스위치
SW0~SW3	입력 키 스위치 Key0~Key3
시, 분 및 초	7-세그먼트 SEG1~SEG7
알람 ON/OFF 상태	LED D9~D15
알람 신호	LED D1~D8
모드	LED D16~D18