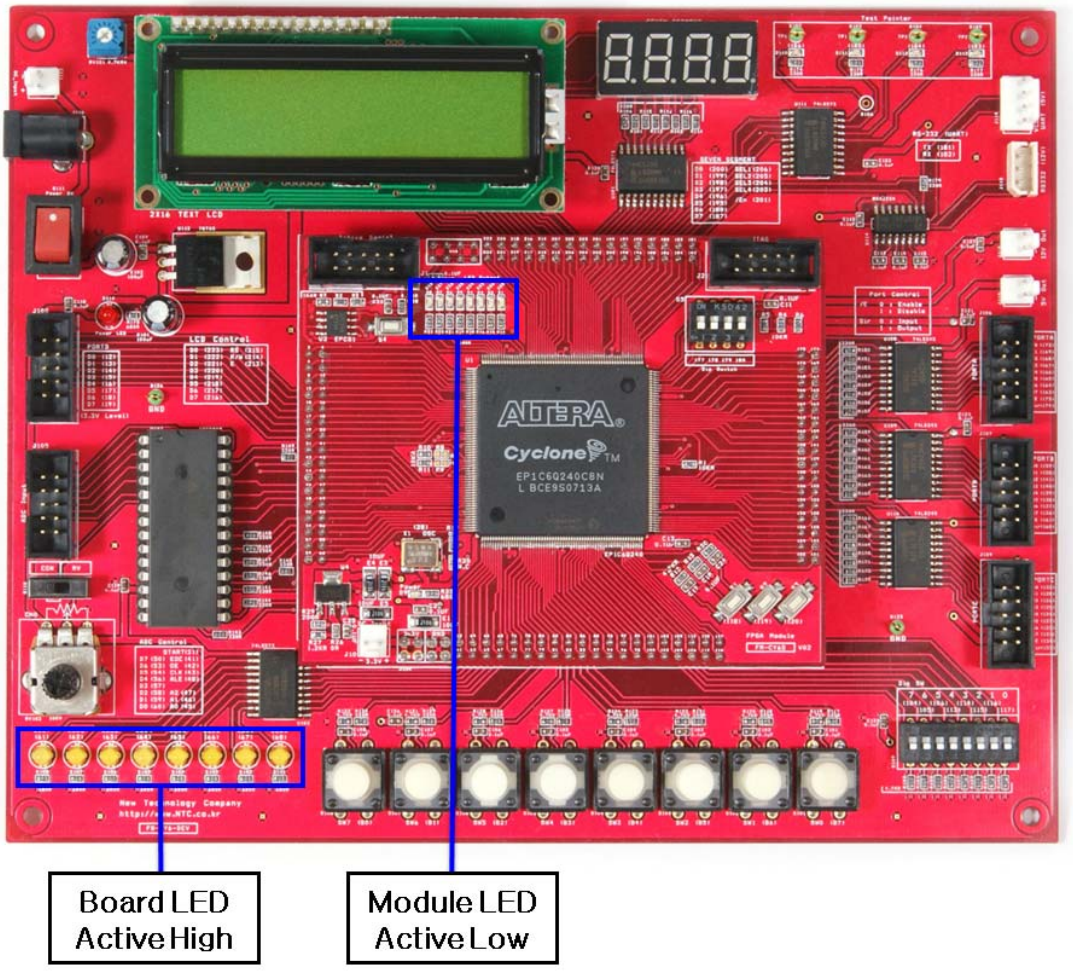
NTC FPGA 강좌 5. LED 켜기

(주) 뉴티씨 (NewTC)
http://www.NewTC.co.kr

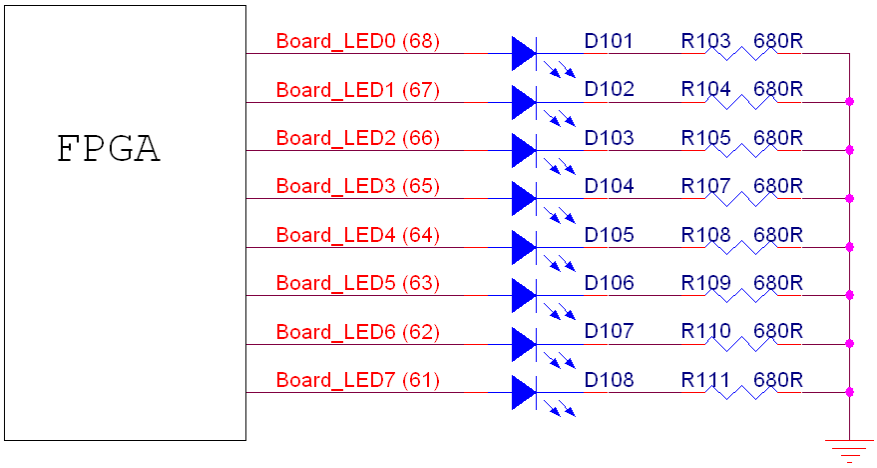
1 LED 구성 및 동작

FM-CY6S (FPGA 개발보드)에는 간단하게 제어할 수 있는 LED 가 모듈에 8개 메인보드에 8개 장착되어 있습니다. 이를 이용하면 내부 레지스터 값을 편리하게 확인할 수 있습니다. FPGA 모듈에 장착되어 있는 LED(Module_LED)는 Active Low 로 동작을 합니다. 해당 포트의 출력이 '0'일 때 LED가 켜지고 '1'일 때 꺼집니다. 메인보드에 장착되어 있는 LED (Board_LED)는 Active High 로 해당 포트의 출력이 '1'일 때 LED가 켜지고 '0'일 때 꺼집니다.



1.1 Board LED 연결 및 핀 번호

Board LED 는 메인보드 상에 장착되어 있는 LED로 내부적으로 버퍼를 통해 출력이 나가고 있습니다. 출력이 '1' 일때 LED가 켜지는 Active High 로 아래 그림과 같이 연결되어 있습니다.



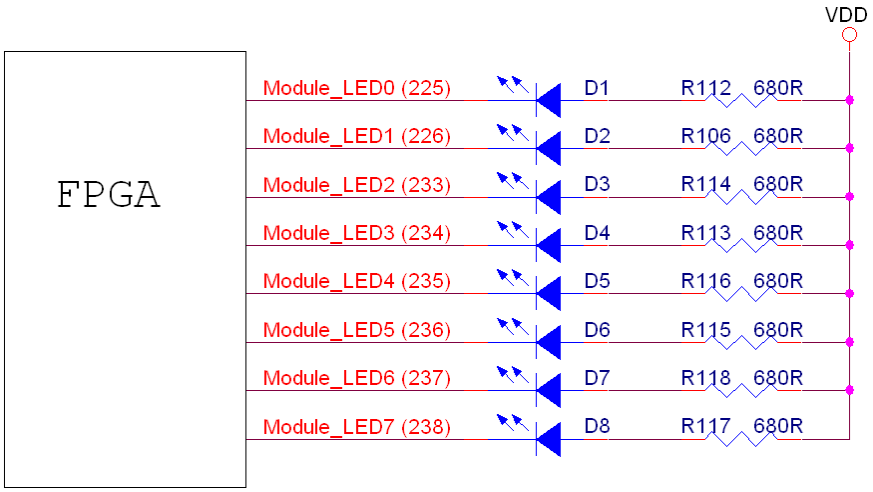
메인보드 LED 출력 핀 번호 (출력이 1일 때 LED가 켜짐)

LED[7]	LED[6]	LED[5]	LED[4]	LED[3]	LED[2]	LED[1]	LED[0]
61	62	63	64	65	66	67	68

1.2 Module LED 연결 및 핀 번호

Module LED 는 FPGA 모듈에 장착되어 있는 LED로 출력이 '0' 일때 LED가 켜지는 Active Low 로 아래 그림과 같이 연결되어 있습니다. 내부 로직의 마지막 출력에서 역수를 출력하면 정상적으로 값을 확인할 수 있습니다.

Ex) board_led <= ~ count[15:8];



모듈 LED 출력 핀 번호 (출력이 0일 때 LED가 켜짐)

LED[7]	LED[6]	LED[5]	LED[4]	LED[3]	LED[2]	LED[1]	LED[0]
238	237	236	235	234	233	226	225

2 LED 동작 시키기

2.1 카운터 값 LED로 출력시키기

아래 예제는 LED 가 정상적으로 동작하는지 확인하기 위해 클럭 카운터를 설계하고 카운터 값을 LED로 출력하는 것입니다. board_led 는 레지스터로 선언하여 always 문 안에서 값을 입력하고 있습니다. module_led 출력은 assign 문을 이용했으며 clk_count 값의 역수를 취해서 넣고 있습니다. 이것은 모듈에 장착되어 있는 LED가 Active Low 로 동작하기 때문입니다.

LED 출력 예제

```
1   module test_led  ( clk, reset, board_led, module_led );

2   input            clk, reset;
3   output  [7:0]    board_led;
4   output  [7:0]    module_led;

5   reg     [7:0]    board_led;
6   reg     [31:0]   clk_count;

7   always @(posedge clk)
8           if(!reset)
9                   clk_count <= 0;
10          else
11                  clk_count <= clk_count + 1;

12  always @(posedge clk)
13          if(!reset)
14                  board_led <= 0;
15          else
16                  board_led <= clk_count[29:22];

17  assign  module_led = ~clk_count[29:22];

18  endmodule
```

16, 17 번째 줄은 clk_count 값의 22번째 비트부터 29번째 비트까지를 LED 로 출력하는 것입니다. 원하는 범위의 값을 LED로 출력해 볼 수 있습니다.

2.2 LED Shift 시키기

LED 1개를 켜고 Shift 시키는 로직을 설계합니다. 앞에서 설계한 클럭 카운터를 이용하여 1초에 한번씩 동작 하도록 합니다. 내부 클럭이 50Mhz 이기 때문에 클럭을 50,000,000 번 카운트 하면 1초를 만들 수 있습니다.

LED Shift 예제

```
module   Sec_Counter      (  clk,  reset,  board_led,  module_led  );
input              clk, reset;
output   [7:0]     board_led;
output   [7:0]     module_led;

reg      [7:0]     board_led;
reg      [31:0]    clk_count;
parameter          SEC_CNT  =  50000000;

always @(posedge clk)
        if(!reset)
                clk_count <= 0;
        else
                if(clk_count < 50000000)
                        clk_count <= clk_count + 1;
                else
                        clk_count <= 0;

always @(posedge clk)
        if(!reset)
                board_led <= 0;
        else
                if(clk_count == 0)
                        if(board_led == 0)
                                board_led <= 8'h01;
                        else
                                board_led <= board_led << 1;

assign   module_led = ~board_led;
endmodule
```

2.3 시뮬레이션 하기

앞의 LED Shift 예제를 실제와 같은 상황으로 시뮬레이션 하면 클럭이 50,000,000 번 들어 올 때까지 기다린 후 LED가 1번 Shift하게 됩니다. 따라서 시뮬레이션에 시간이 많이 소비되며 디버깅 하기도 불편하게 됩니다. 다른 방법으로는 파라미터를 사용하여 시뮬레이션 할 때마다 바꾸는 것인데 이 방법도 매우 불편합니다.

합성 시에는 파라미터가 50,000,000 으로 셋팅되고 시뮬레이션 시에는 50 으로 셋팅된다면 이러한 불편이 해소 될 것입니다.

Verilog 문법에서는 이를 지원하는데 defparam 이라는 명령이 있습니다. 테스트벤치에서 defparam 을 이용하여 특정 모듈에 parameter 값을 변경할 수 있습니다.

defparam 문을 사용할 때는 인스턴스 이름을 포함하여 지정해야 합니다. 아래 예제에서 Sec_Counter 모듈는 u1 이라는 이름으로 생성되었기 때문에 u1.SEC_CNT 라는 전체 계층 이름을 적어준 것입니다.

Test Bench 예제

```
…
defparam    u1.SEC_CNT = 50;

Led_Shift    u1 (
    .clk(clk),
    .reset(reset),
    .board_led(board_led),
    .module_led(module_led)
   );
…
```

실습 과제

1. 2.1장의 30비트 클럭 카운터에서 clk_count 가 50,000,000 번 카운트를 하면 1초를 만들 수 있습니다. 이것을 이용하여 LED 출력 값이 1초에 하나씩 증가하도록 로직을 설계하고 시뮬레이션 후 보드에서 동작을 확인합니다. (아래 소스는 1초에 한번씩 sec_flag 가 1clk 사이클 동안 1이 됩니다.)

```
always @(posedge clk)
    if(!reset)
        clk_count <= 0;
    else
        if(clk_count < 50000000) begin
            clk_count <= clk_count + 1;
            sec_flag <= 0;
        end else begin
            clk_count <= 0;
            sec_flag <= 1;
        end
```

2. 2.2장의 LED Shift 시키는 로직을 응용하여 LED를 좌/우로 움직이도록 설계합니다.