NTC FPGA 강좌 3. 로직 설계 및 시뮬레이션

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

1 로직 설계의 기초

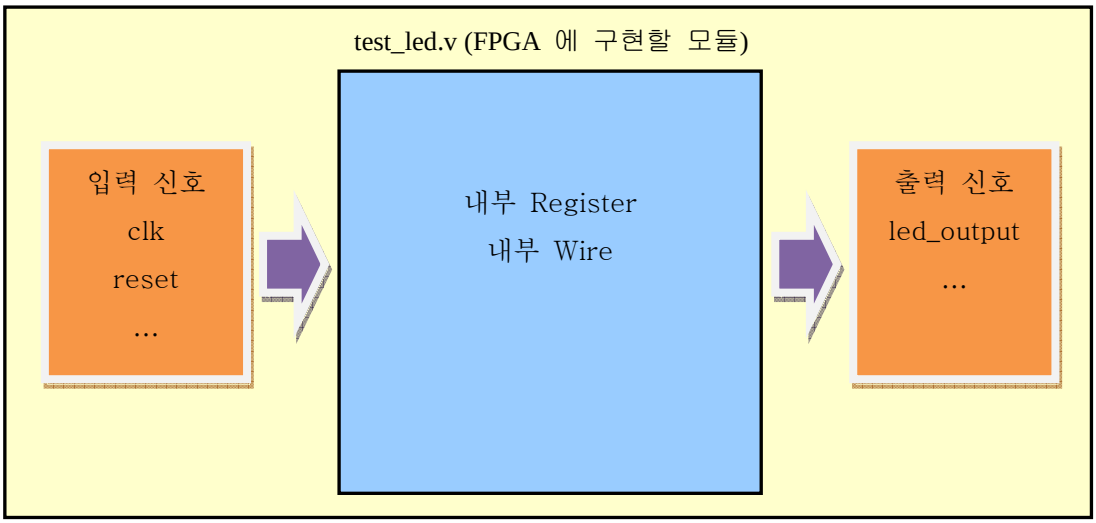
이 장에서는 간단한 로직을 설계하여 시뮬레이션을 해보겠습니다. 칩으로 구현하기 전에 시뮬레이션을 통한 검증은 꼭 필요한 단계이며 복잡한 로직을 구현할수록 더욱 더 중요해 집니다. 간단한 로직의 경우 테스트 핀으로 값을 확인하면서 설계할 수도 있지만 복잡한 로직의 경우 내부에 값들을 하나씩 확인해 볼 수가 없기 때문입니다.

1.1 모듈 설계 및 테스트 벤치 작성

Verilog HDL 로 작성한 로직의 기본 단위를 모듈이라고 합니다. 실제 합성해서 구현하고 싶은 블록도 모듈로 구성하고 이것을 테스트 하기 위한 모듈을 작성하는데 이것을 테스트 벤치라고 부릅니다. 따라서 테스트 벤치는 입출력 포트를 선언할 필요가 없습니다.

앞장에서 시뮬레이션 한 모듈을 보면 실제로 FPGA 로 구현할 모듈은 test_led로 clk, reset, led_out 과 같은 입/출력 포트가 선언되어 있습니다. 이것을 테스트 하기 위해서 tb_test_led 모듈을 만들고 내부에서 입력 신호를 발생 시켰습니다.

tb_test_led.v (test_led 모듈을 위한 Test Bench)



tb_test_led 모듈 (tb_test_led.v)

```
1    `timescale 1ns/10ps

2    module tb_test_led;

3    reg      clk, reset;
4    wire [7:0]    led_out;

5    initial
6    begin
7       #0    reset = 0;                  reset 신호 인가
8       #20 reset = 1;
9    end

10   initial
11   begin
12      #0    clk = 0;                    clk 신호 인가
13      forever
14           #5    clk = ~clk;
15   end

16   test_led    u1 (                      test_led 모듈 생성
17      .clk(clk),
18      .reset(reset),
19      .led_out(led_out) );

20   endmodule
```

소스 설명

위 소스의 2번째 줄에서 tb_test_led 라는 테스트 벤치 모듈을 선언하였습니다. 테스트 벤치 모듈이기 때문에 앞에서 말한 것 과 같이 입/출력 포트 선언이 없습니다.

3번째 줄에서 test_led 모듈에서 사용되는 입력 신호를 발생시키기 위해서 clk, reset 신호를 레지스터로 선언하였습니다. 이 신호를 발생시켜 test_led 모듈에 인가하게 됩니다.

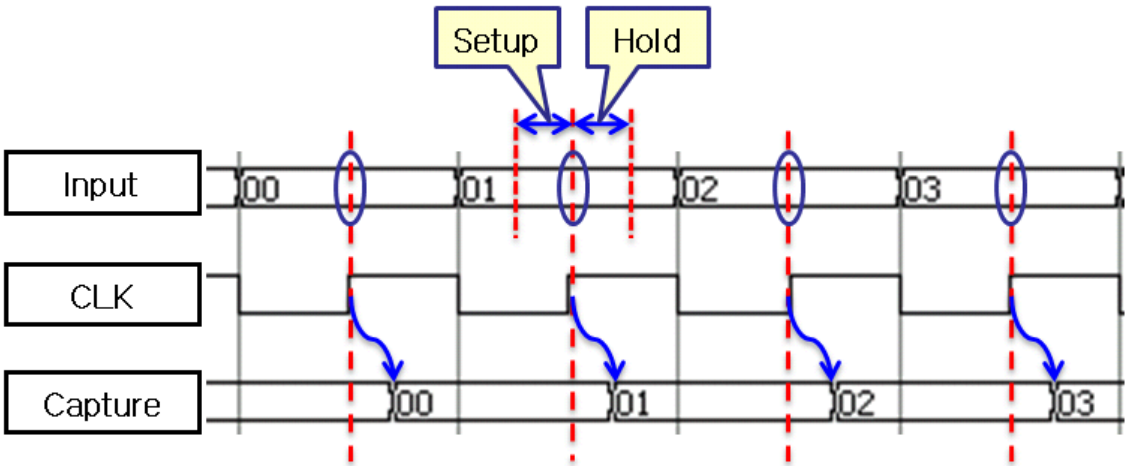
4번째 줄에서는 test_led 모듈의 출력 신호를 wire 형으로 선언하였습니다. led_out 은 테스트 모듈의 출력 신호이기 때문에 wire 형으로 선언해야 합니다.

5~9 번째 줄에서는 reset 신호를 발생 시켰습니다. reset 신호는 한번만 발생하기 때문에 initial 구문으로 구현하였습니다.

10~15 번째 줄에서는 clk 신호를 발생 시켰습니다. clk 신호는 주기적인 신호이기 때문에 initial 구문을 이용하여 초기값을 인가하고 always 구문을 이용하여 주기적인 신호를 구현하였습니다. 여기서는 10ns 마다 토글하므로 50Mhz 클럭이 됩니다.

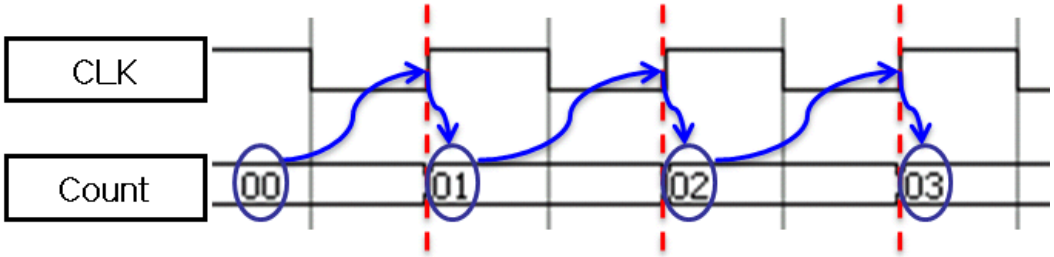
1.2 신호 인가 방법, 타이밍도 보는 방법

테스트 벤치를 작성할 때 신호는 대부분 클럭에 동기하여 인가해야 합니다. 클럭의 상승 에지에서 신호를 캡처 할 수 있도록 상승 에지가 발생하기 전에 신호를 해가해야 합니다. 아래 그림은 입력 신호를 인가한 후 Flip-Flop 으로 캡처하는 동작을 보인 것입니다. 상승 에지가 발생하는 순간에 캡처를 한 후 약간의 딜레이 후 캡처된 것을 볼 수 있습니다.



클럭의 발생 전/후로 입력 값을 일정시간 이상 유지해야 하는데 이것을 셋업/홀드 타임이라고 부릅니다. 이것을 고려하여 신호를 인가해야 하며 시뮬레이션 상에서는 크게 문제 되지 않습니다. 하지만 실제로는 많이 발생하기 때문에 이러한 것을 고려하여 설계해야 합니다.

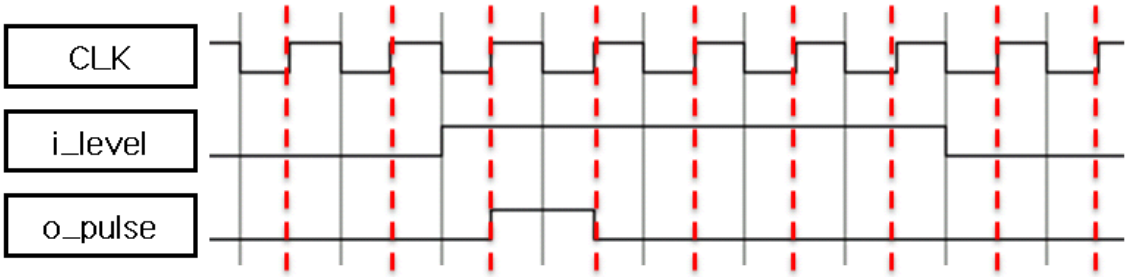
앞에서 시뮬레이션 한 것은 Count 값을 하나씩 증가시키는 것으로 시뮬레이션 결과는 아래와 같습니다.



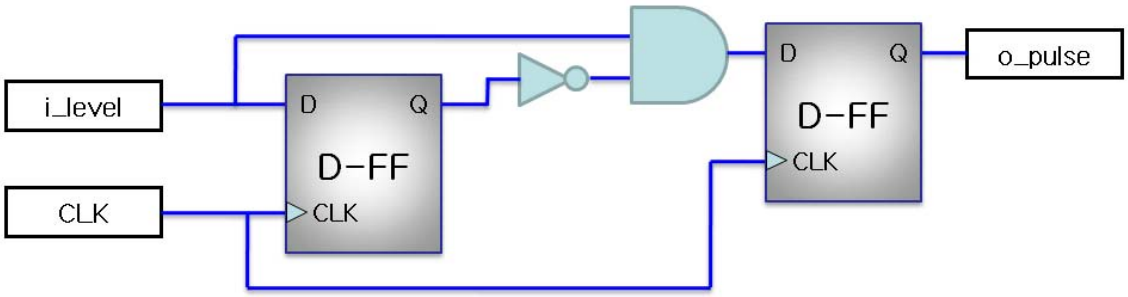
2 로직 설계 실습

2.1 Level To Pulse 컨버터

입력 신호에 따라 0에서 1로 변하는 순간을 감지하여 한번만 실행하거나 인식해야 하는 경우가 있습니다. 이런 경우 필요한 것이 Level to Pulse 변환기 입니다. 여기서는 신호의 상승 에지를 감지하여 1개의 펄스만 발생시키도록 해보겠습니다. 입력과 출력 신호는 아래와 같습니다.



이러한 동작을 위해서는 아래와 같은 회로가 필요합니다. i_level 신호가 D-FF을 통과한 후 신호는 i_level 신호의 1클럭 이전값이 됩니다. 이 신호가 0이고 현재 i_level 신호가 1인 경우 상승 에지라는 것을 알 수 있습니다.

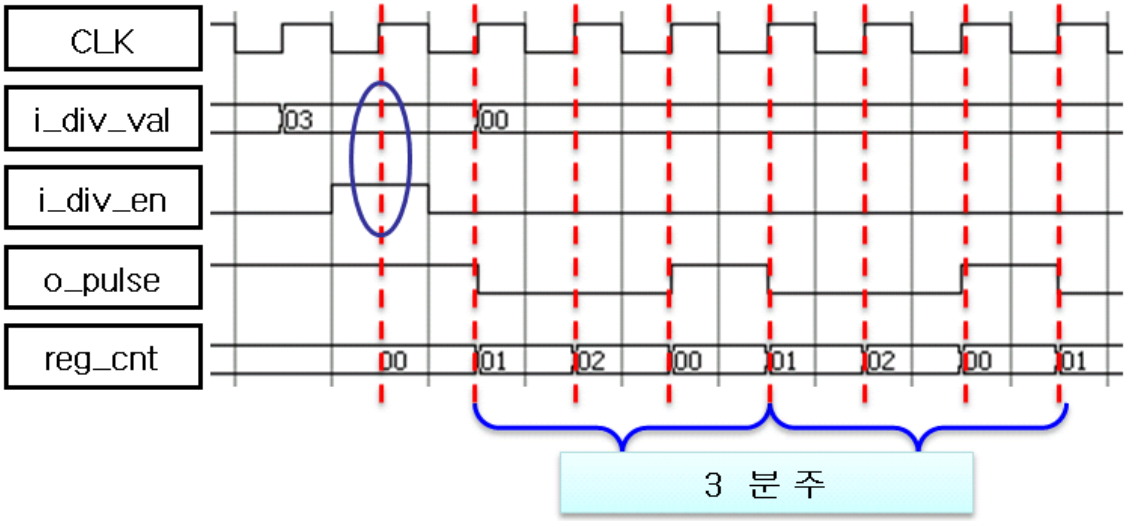


이것을 Verilog HDL로 구현하면 아래와 같습니다.

```
always @(posedge clk)
        if(!reset)
                i_level_buff <= 0;
        else
                i_level_buff <= i_level;
always @(posedge clk)
        if(!reset)
                o_pulse <= 0;
        else
                if((i_level_buff == 0) && (i_level == 1))
                        o_pulse <= 1;
                else
                        o_pulse <= 0;
```

2.2 분주회로 설계

클럭을 원하는 만큼 분주하는 회로를 설계합니다. 입력은 8비트의 i_clk_div[7:0] 신호와 입력 인에이블 신호 i_div_en이며 출력은 i_clk_div 만큼 분주된 신호입니다. reg_cnt 는 내부 로직에서 사용한 값입니다. 아래 그림과 같이 i_div_val 값이 3 이라면 o_pulse 신호는 3클럭마다 1이 되게 됩니다.



실습 과제

1. 2-1장의 레벨 to 펄스 컨버터를 설계하여 상승 에지를 감지 하도록 설계합니다.

2. 실습 1을 응용하여 상승 에지와 하강 에지 모두를 감지하도록 변경합니다.

3. 2-2장의 분주 회로를 설계하고 시뮬레이션 합니다.

4. 하드웨어 적으로 스위치 입력을 받을 때 glitch가 발생하게 됩니다. glitch 는 고주파로 발생하며 연속되는 0또는 1신호가 4개 미만이라고 가정합니다. 이런 glitch를 제거할 수 있는 glitch_filter 를 설계하고 시뮬레이션 합니다.
(1 신호가 4개 이상 지속될 경우 스위치 입력을 1로 인식하고 0 신호가 4개 이상 지속될 경우 0으로 인식하도록 합니다.)