조별 과제: 스톱워치

날짜: 2022. 11. 24

학번, 이름: 21700384 송태웅
21700794 홍준형

1. 목적

- 7 segment와 clock counter 등을 이용하여 스톱워치를 구현할 수 있다.
- PS2키보드와 UART 통신을 이용하여 스톱워치를 제어할 수 있다.

2. 과제 내용

출력	7 segment 6개 (7 segment display의 우측에서부터 1/100초, 1/10초, 1초, 10초, 1분 단위로 표시)
상태	0: 정지 및 초기화 1: 작동(시간이 증가하는 상태)
입력	1) PS2키보드 입력 - ENTER 입력: 스톱워치 reset - s 입력: (초기화 후 누르는 경우) 스톱워치 카운트 시작 (일시정지 후 누르는 경우) 일시정지된 이후부터 카운트 재시작 - t 입력: 스톱워치 일시정지 2) UART 키보드 간접 입력 - (위의 내용과 동일)

3. Flowchart

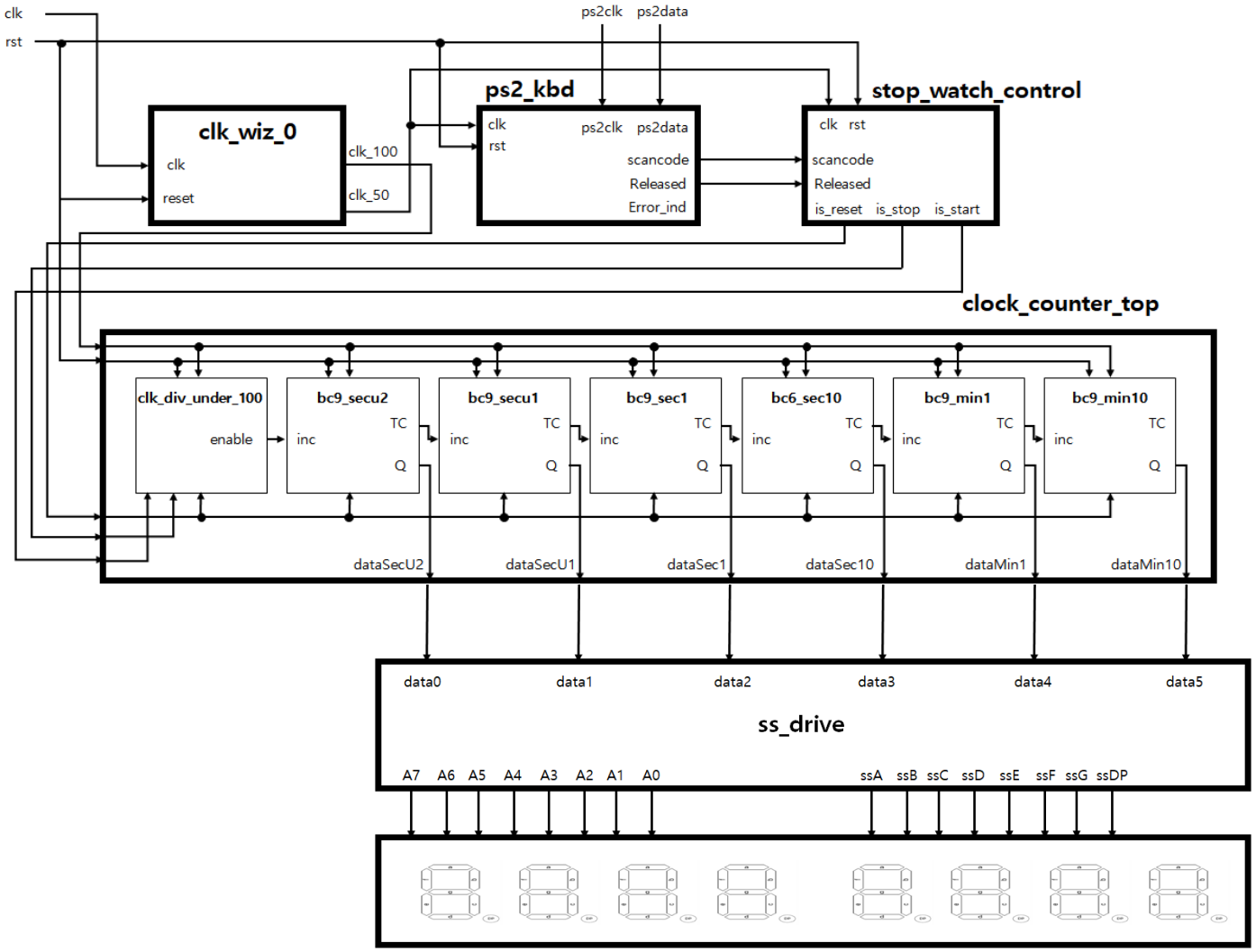


그림 1. 스톱워치 회로도(PS2KEYBOARD)

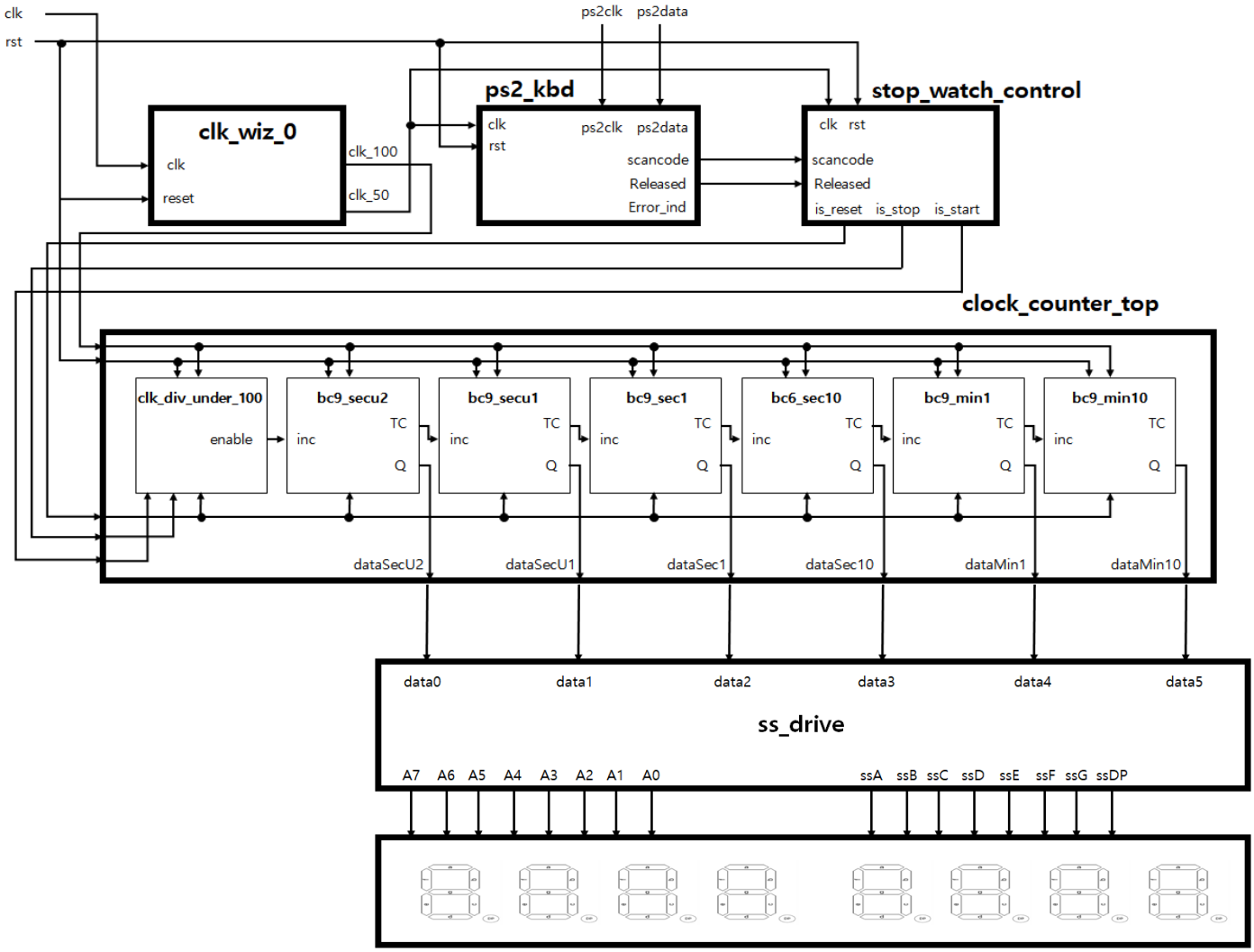


그림 2. 스톱워치 회로도 (UART)

4. 코드 및 기능

- Top module

```
`timescale 1ns / 1ps

module stop_watch_top(
    input clk,
    input rst,
    input RxD,
    input ps2clk,
    input ps2data,
    output TxD,
    output Released,
    output ssA,
    output ssB,
    output ssC,
    output ssD,
    output ssE,
    output ssF,
    output ssG,
    output ssDP,
    output AN7,
    output AN6,
    output AN5,
    output AN4,
    output AN3,
    output AN2,
    output AN1,
    output AN0
    );


wire clk_50;
wire clk_100;

wire [7:0] scancode;
wire err_ind;

wire [7:0] datain_ext, dataout_ext;
wire new_in, new_out;


wire [3:0] dataSecU2 ;
wire [3:0] dataSecU1 ;
wire [3:0] dataSec1  ;
wire [3:0] dataSec10 ;
wire [3:0] dataMin1  ;
wire [3:0] dataMin10 ;

wire is_reset, is_start, is_stop;

wire mode_ps2, mode_uart;
```

```
clk_wiz_0 clk_core(
  // Clock in ports
   .clk_in1(clk),          // input clk_in1

   // Clock out ports
   .clk_out1(clk_100),     // output clk_out1
   .clk_out2(clk_50),      // output clk_out2

   // Status and control signals
   .reset(rst),            // input reset
   .locked()               // output locked
);


ps2_kbd_top ps2_kbd (
    .clk    (clk_50),
    .rst    (rst),
    .ps2clk (ps2clk),
    .ps2data    (ps2data),
    .scancode   (scancode),
    .Released   (Released),
    .err_ind    (err_ind)
    );


stop_watch_control stop_watch_ctrl(
    .clk        (clk_100),
    .rst        (rst),
    .released   (Released),
    .data       (scancode),
    .dataUart   (dataout_ext),
    .is_reset   (is_reset),
    .is_stop    (is_stop ),
    .is_start   (is_start)
    );

clock_counter_top clkCnt(
    .clk        (clk_100),
    .rst        (rst),
    .is_reset   (is_reset),
    .is_stop    (is_stop ),
    .is_start   (is_start),
    .dataMin1   (dataMin1),
    .dataMin10  (dataMin10),
    .dataSec1   (dataSec1),
    .dataSec10  (dataSec10),
    .dataSecU1  (dataSecU1),
    .dataSecU2  (dataSecU2)
    );
```

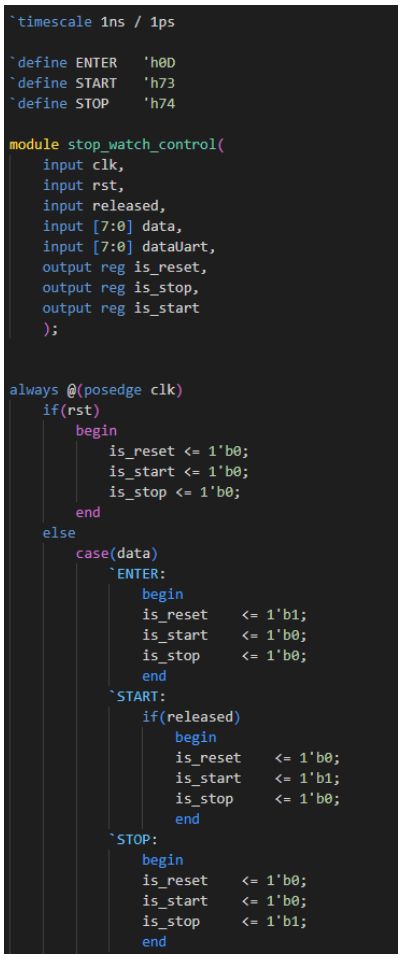
```
ss_drive ss_drive (
    .clk        (clk_100),
    .rst        (rst),
    .data7  (),
    .data6  (),
    .data5  (dataMin10),
    .data4  (dataMin1),
    .data3  (dataSec10),
    .data2  (dataSec1),
    .data1  (dataSecU1),
    .data0  (dataSecU2),
    .mask   (8'b0011_1111),
    .ssA    (ssA),
    .ssB    (ssB),
    .ssC    (ssC),
    .ssD    (ssD),
    .ssE    (ssE),
    .ssF    (ssF),
    .ssG    (ssG),
    .ssDP   (ssDP),
    .AN7    (AN7),
    .AN6    (AN6),
    .AN5    (AN5),
    .AN4    (AN4),
    .AN3    (AN3),
    .AN2    (AN2),
    .AN1    (AN1),
    .AN0    (AN0)
    );

endmodule
```

그림 3. Top module 코드

Top module은 그림 1의 회로도에 맞게 코드를 구성하였다.

- Sub module
  - stop_watch_control



```
`timescale 1ns / 1ps

`define ENTER   'h0D
`define START   'h73
`define STOP    'h74

module stop_watch_control(
    input clk,
    input rst,
    input released,
    input [7:0] data,
    input [7:0] dataUart,
    output reg is_reset,
    output reg is_stop,
    output reg is_start
    );


always @(posedge clk)
    if(rst)
        begin
            is_reset <= 1'b0;
            is_start <= 1'b0;
            is_stop <= 1'b0;
        end
    else
        case(data)
            `ENTER:
                begin
                is_reset    <= 1'b1;
                is_start    <= 1'b0;
                is_stop     <= 1'b0;
                end
            `START:
                if(released)
                    begin
                    is_reset    <= 1'b0;
                    is_start    <= 1'b1;
                    is_stop     <= 1'b0;
                    end
            `STOP:
                begin
                is_reset    <= 1'b0;
                is_start    <= 1'b0;
                is_stop     <= 1'b1;
                end
            default:
                begin
                is_reset <= 1'b0;
                is_start <= 1'b0;
                is_stop <= 1'b0;
                end
        endcase

endmodule
```

그림 4. Sub module 코드 (1)

stop_watch_control 모듈은 스톱워치 제어를 위한 제어신호를 생성한다. 키보드 ‘s’를 눌렀다 뗄 경우, 스톱워치 동작을 의미하는 is_start 제어신호가 켜진다. 키보드 ‘t’를 누르는 경우, 스톱워치 일시정지를 의미하는 is_stop 제어신호가 켜진다. 마지막으로 키보드 ‘ENTER’를 누르는 경우, 스톱워치 초기화를 의미하는 is_reset 제어신호가 켜진다.

■ clock_counter_top

```
`timescale 1ns / 1ps

module clock_counter_top(
    input clk,
    input rst,
    input is_reset,
    input is_stop ,
    input is_start,
    output [3:0] dataMin1   ,
    output [3:0] dataMin10  ,
    output [3:0] dataSec1   ,
    output [3:0] dataSec10  ,
    output [3:0] dataSecU1  ,
    output [3:0] dataSecU2
    );

wire inc;
wire tcSecU2    ;
wire tcSecU1    ;
wire tcSec1     ;
wire tcSec10    ;
wire tcMin1     ;
wire tcMin10    ;

wire [3:0] secU2    ;
wire [3:0] secU1    ;
wire [3:0] sec1     ;
wire [3:0] sec10    ;
wire [3:0] min1     ;
wire [3:0] min10    ;

clk_div_under_100(
    .clk    (clk), // 100MHz
    .rst    (rst),
    .is_reset   (is_reset),
    .is_stop    (is_stop ),
    .is_start   (is_start),
    .enable (inc)
    );
```

```
BC9 bc9_secu2(
    .clk    (clk),
    .rst    (rst),
    .is_reset   (is_reset),
    .Q          (secU2),
    .inc        (inc),
    .TC         (tcSecU2)
    );

BC9 bc9_secu1(
    .clk        (clk),
    .rst        (rst),
    .is_reset   (is_reset),
    .Q          (secU1),
    .inc        (tcSecU2),
    .TC         (tcSecU1)
    );

BC9 bc9_sec1(
    .clk        (clk),
    .rst        (rst),
    .is_reset   (is_reset),
    .Q          (sec1),
    .inc        (tcSecU1),
    .TC         (tcSec1)
    );

BC6 bc6_sec10(
    .clk        (clk),
    .rst        (rst),
    .is_reset   (is_reset),
    .Q          (sec10),
    .inc        (tcSec1),
    .TC         (tcSec10)
    );

BC9 bc9_min1(
    .clk        (clk),
    .rst        (rst),
    .is_reset   (is_reset),
    .Q          (min1),
    .inc        (tcSec10),
    .TC         (tcMin1)
    );
```

```
BC6 bc6_min10(
    .clk        (clk),
    .rst        (rst),
    .is_reset   (is_reset),
    .Q          (min10),
    .inc        (tcMin1),
    .TC         (tcMin10)
    );

assign dataMin10 = min10;
assign dataMin1  = min1 ;
assign dataSec1  = sec1 ;
assign dataSec10 = sec10;
assign dataSecU1 = secU1;
assign dataSecU2 = secU2;


endmodule
```

그림 5. Sub module 코드 (2)

clock_counter_top 모듈은 clock divider 의 기능을 하는 clk_div_under_100 모듈과 binarry counter BC6 와 BC9 로 이루어져 있다. clk_div_under_100 은 100MHz 인 입력 clock 으로 스톱워치의 1/100 초(100Hz)를 구현하기 위해 clock 의 rising edge 마다 증가하는 count 변수가 10^6번마다 enable 신호를 켜도록 구현하였다. 이 enable 신호를 1/100 초 단위의 시간을 count 하는 BC9 의 increment 입력으로 주었으며, 이를 기준으로 1/10 초, 1 초, 10 초, 1 분, 10 분 단위의 시간을 count 하는 BCD counter 를 자동으로 동기화할 수 있다.

◆ clk_div_under_100

```
module clk_div_under_100(
    input clk, // 100MHz
    input rst,
    input is_reset,
    input is_stop ,
    input is_start,
    output reg enable
    );

localparam MILLS102CNT = 1000000;

integer cnt;

always @(posedge clk)

    if (rst || is_reset)
        begin
            cnt <= 0;
            enable <= 0;
        end

    else if(is_start)
        if (cnt < MILLS102CNT)
            begin
                cnt <= cnt + 1;
                enable <= 0;
            end
        else
            begin
            enable <= 1;
            cnt <= 0;
            end

    else if (is_stop)
        begin
        cnt <= cnt;
        enable <= 0;
        end
```

그림 6. Sub module 코드 (3)

◆ BC6

```
`timescale 1ns / 1ps

module BC6 (clk, rst, is_reset, inc, Q, TC);

input rst, clk;
input is_reset;
input inc;

output reg [3:0] Q ;
output TC ;


wire finl;

always @(posedge clk)
  if(rst || is_reset) Q <= 4'd0;

  else if (inc)
    if(fin) Q <= 4'd0;
    else Q <= Q + 1;

assign fin = (Q == 4'd5) ? 1 : 0;
assign TC = fin & inc ;

endmodule
```

그림 7. Sub module 코드 (4)

◆ BC9

```
`timescale 1ns / 1ps

module BC9 (clk, rst, is_reset, inc, Q, TC);

input rst, clk;
input inc;
input is_reset;
output reg [3:0] Q ;
output TC ;


wire finl;

always @(posedge clk)
  if(rst || is_reset) Q <= 4'd0;

  else if (inc)
    if(fin) Q <= 4'd0;
    else Q <= Q + 1;

assign fin = (Q == 4'd9)?1:0;
assign TC = fin & inc ;

endmodule
```

그림 8. Sub module 코드 (5)

5. 결과

- Testbench
  - stop_watch_control

```
`timescale 1ns / 1ps


module tb_stop_watch_control;

reg clock, reset;
reg released;
reg [7:0] scancode;
reg [7:0] dataUart;
wire is_reset, is_stop, is_start;

stop_watch_control uut(
    clock,
    reset,
    released,
    scancode,
    dataUart,
    is_reset,
    is_stop,
    is_start
    );

always #5 clock = ~clock;
```

```
initial begin
    clock = 1'b0;
    reset = 1'b1;
    released = 1'b0;
    scancode = 'h00;
    dataUart = 'hxx;

    #10 reset = 1'b0;

    #30 scancode = 'h73; // START
        released = 1'b1;

    #15 released = 1'b0;

    #30 scancode = 'h74; // STOP

    #30 scancode = 'h0D; // ENTER

    #30 scancode = 'h73; // START
        released = 1'b1;

    #30 scancode = 'h74; // STOP
        released = 1'b0;

    #30 scancode = 'h73; // START (x)

    #30 scancode = 'h0D; // ENTER

    #10 $stop;
end



endmodule
```

그림 9. testbench 코드 (1)

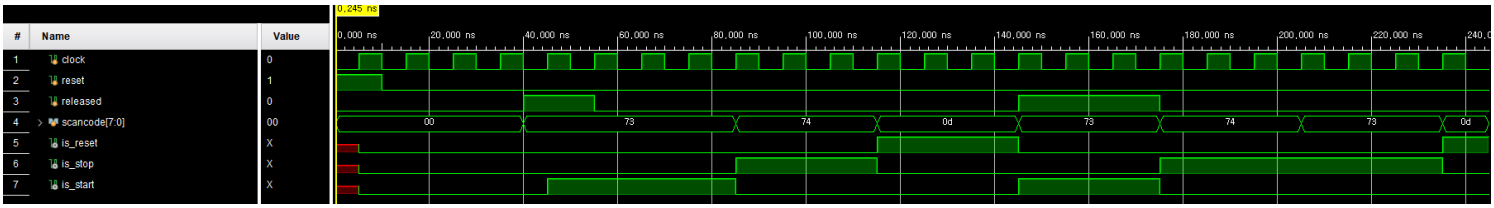


그림 10. 시뮬레이션 결과 (1)

stop_watch_control 모듈의 시뮬레이션을 통해 테스트 입력에 대해 올바른 제어신호가 생성되는지 확인하였다. 모든 값이 초기화 된 상태에서 먼저, 키보드 ‘s’에 해당하는 값(‘h73)과 released bit에 1을 주어 키보드 ‘s’를 눌렀다 뗀 경우의 테스트 입력을 주었을 때, 스톱워치 작동을

의미하는 제어신호인 is_start bit가 1로 켜진 것을 확인할 수 있다. 다음으로 키보드 ‘t’에 해당하는 값(‘h74)을 준 경우, 스톱워치 일시정지를 의미하는 제어신호인 is_stop bit가 1로 켜지고, is_start bit는 0으로 꺼지는 것을 확인할 수 있다. 키보드 ‘ENTER’에 해당하는 값(‘h0D)을 준 경우, 스톱워치 초기화를 의미하는 제어신호인 is_reset bit가 1로 켜지고, 나머지 제어신호는 0으로 꺼지는 것을 확인할 수 있다. 마지막으로, 키보드 ‘s’에 해당하는 값(‘h73)과 released bit가 0인 경우에는 제어신호의 변화가 없음을 확인할 수 있다.

- BC6

```
`timescale 1ns / 1ps


module tb_BC6;

reg clock, reset;
reg is_reset;
reg inc;

wire [3:0] Q;
wire TC;

BC6 uut(
    clock,
    reset,
    is_reset,
    inc,
    Q,
    TC
    );

always #5 clock = ~clock;

initial begin
    clock = 1'b0;
    reset = 1'b1;
    is_reset = 1'b0;
    inc = 1'b0;

    #10 reset = 1'b0;
        inc = 1'b1;

    #85 is_reset = 1'b1;

    #5  is_reset = 1'b0;

    #50 $stop;
end


endmodule
```

그림 11. testbench 코드 (2)

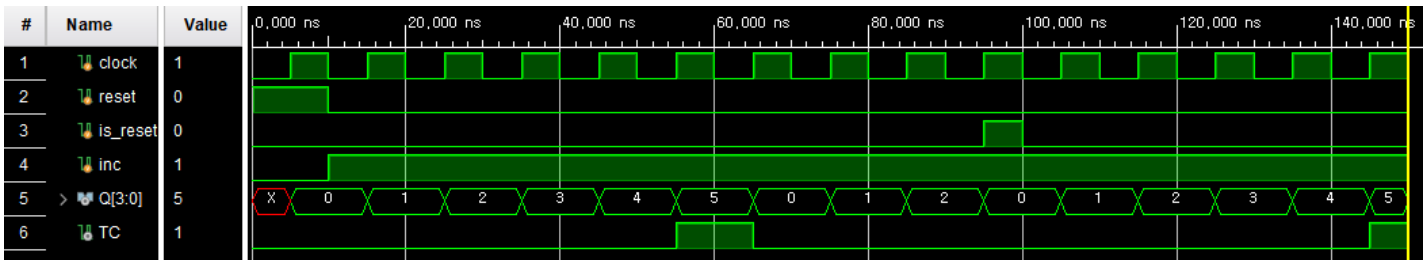


그림 12. 시뮬레이션 결과 (2)

위 그림은 4 bit BCD counter에 modulo 5 연산을 적용하여 0부터 5까지 카운트하는 BCD counter에 대한 시뮬레이션 결과이다. increment를 의미하는 inc 신호는 항상 1로 켜진 상태를 가정하고, 시뮬레이션 하였다. 먼저, counter 출력인 Q가 0부터 5까지 증가한 후, TC bit가 1로 켜진다. 이후 다시 Q는 0으로 돌아가고, TC bit도 0으로 꺼지는 것을 확인할 수 있다. 추가적으로 스톱워치 컨트롤러에서 출력되는 is_reset 신호가 1로 켜진 경우, Q가 0으로 초기화 되는 것을 확인할 수 있다.

■ BC9

```
`timescale 1ns / 1ps


module tb_BC9;

reg clock, reset;
reg is_reset;
reg inc;

wire [3:0] Q;
wire TC;

BC9 uut(
    clock,
    reset,
    is_reset,
    inc,
    Q,
    TC
    );

always #5 clock = ~clock;

initial begin
    clock = 1'b0;
    reset = 1'b1;
    is_reset = 1'b0;
    inc = 1'b0;

    #10 reset = 1'b0;
        inc = 1'b1;

    #150 is_reset = 1'b1;

    #10  is_reset = 1'b0;

    #100 $stop;
end


endmodule
```

그림 13. testbench 코드 (3)

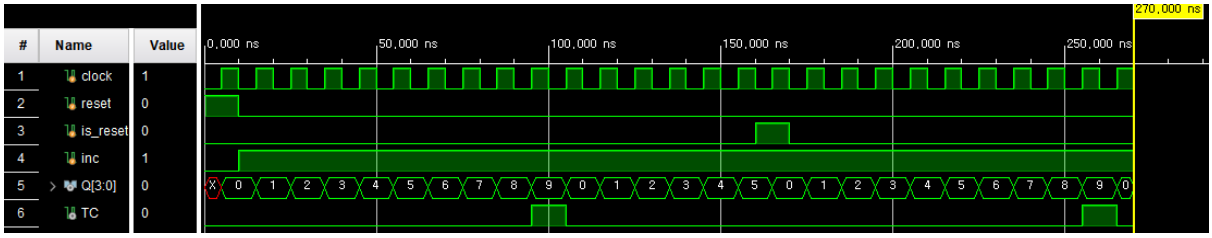


그림 14. 시뮬레이션 결과 (3)

위 그림은 4 bit BCD counter에 modulo 9 연산을 적용하여 0부터 9까지 카운트하는 BCD counter에 대한 시뮬레이션 결과이다. 위에서 기술한 BC6와 동일한 원리로 작동하며, 정상적으로 동작하는 것을 확인할 수 있다.

- 데모영상
  - PS2KEYBOARD
  - UART