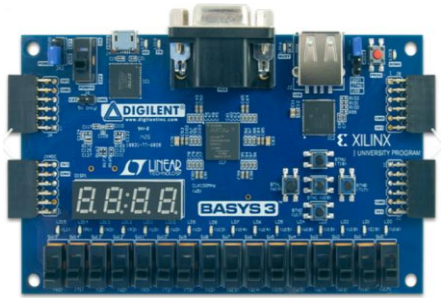
UART+WATCH 프로젝트

AI 시스템반도체설계 2기
7조 발표자: 서윤철

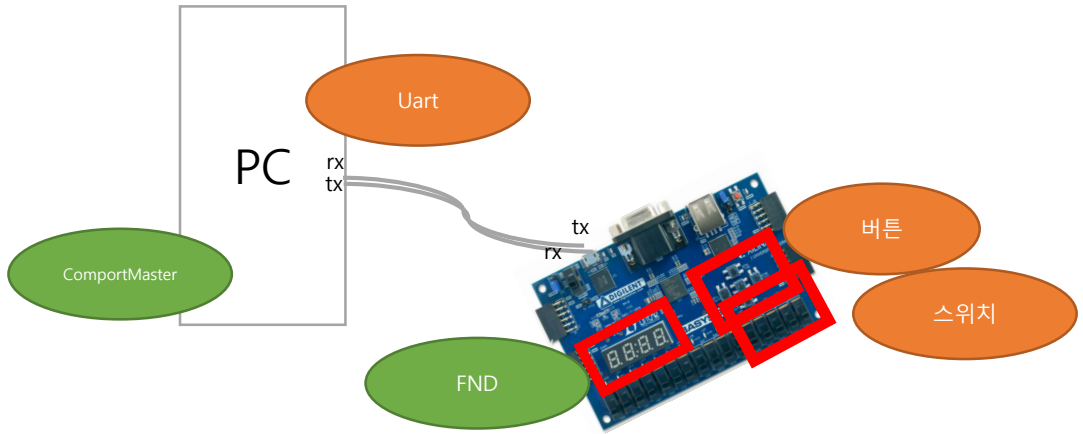
목차

Verilog



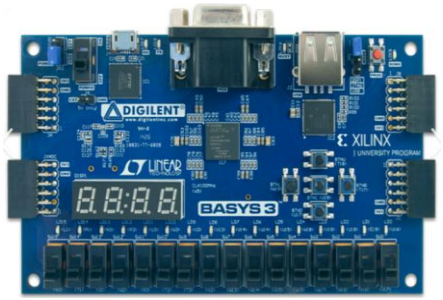
VIVADO™

목표: Uart 및 button, switch를 이용한 시계 제어



input	output
버튼, Uart, 스위치	FND, ComportMaster(tx)

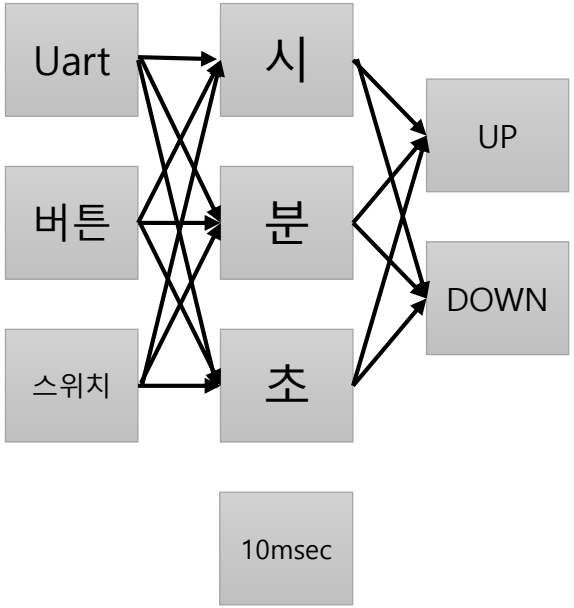
Verilog



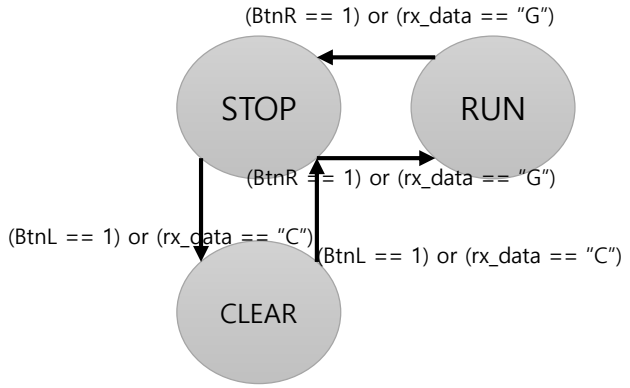
VIVADO™

Watch, Stopwatch 제어

Watch

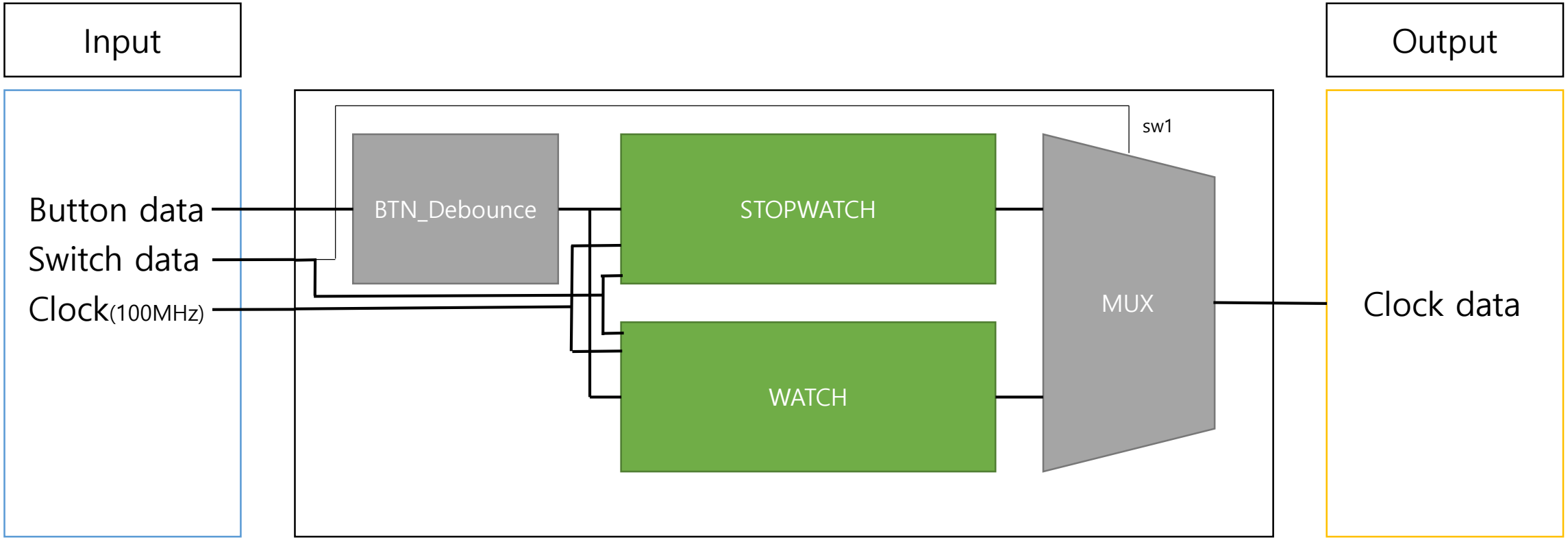


Stopwatch



구현 과정

WATCH

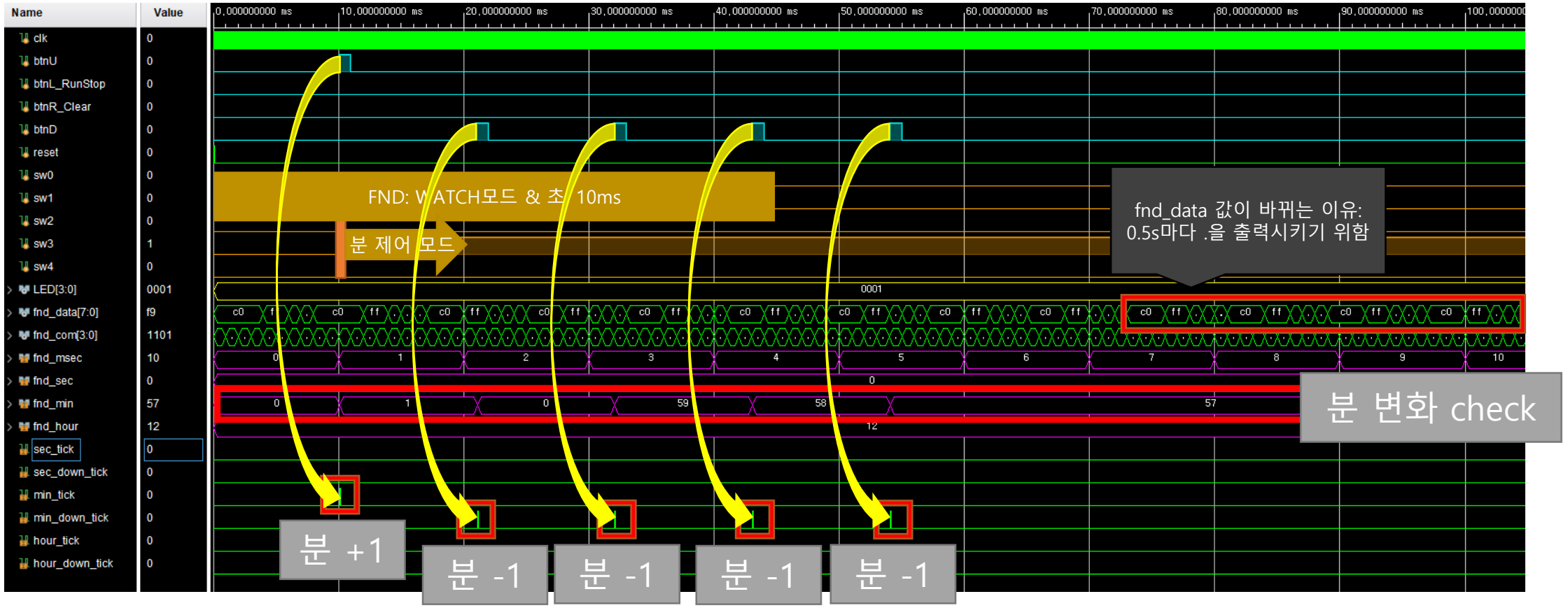


Button data	BtnC	BtnU	BtnD	BtnL	BtnR
Push	Reset	UP	DOWN	RUN/STOP	CLEAR
Switch data	**sw4**	**sw3**	**sw2**	**sw1**	**sw0**
On	Hour_ctrl	Min_ctrl	Sec_ctrl	Mode	HM/SM Mode

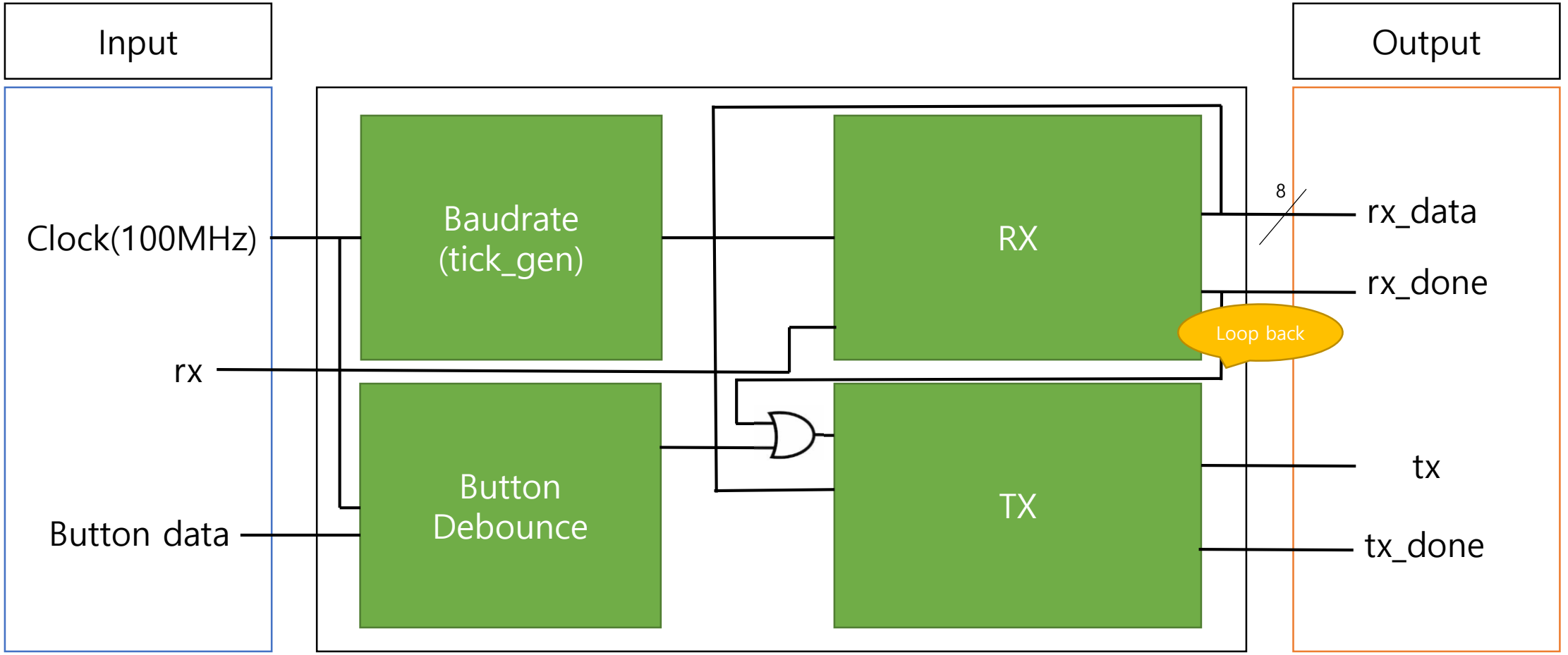
sw1	Clock Data
ON	STOPWATCH
OFF	WATCH

WATCH

FND


Name
Value
clk
btnU
btnL_RunStop
btnR_Clear
btnD
reset
sw0
sw1
sw2
sw3
sw4
LED[3:0]
fnd_data[7:0]
fnd_com[3:0]
fnd_msec
fnd_sec
fnd_min
fnd_hour
sec_tick
sec_down_tick
min_tick
min_down_tick
hour_tick
hour_down_tick
FND: WATCH모드 & 초 10ms
분 제어 모드
fnd_data 값이 바뀌는 이유:
0.5s마다 .을 출력시키기 위함
분 변화 check
분 +1
분 -1
분 -1
분 -1
분 -1

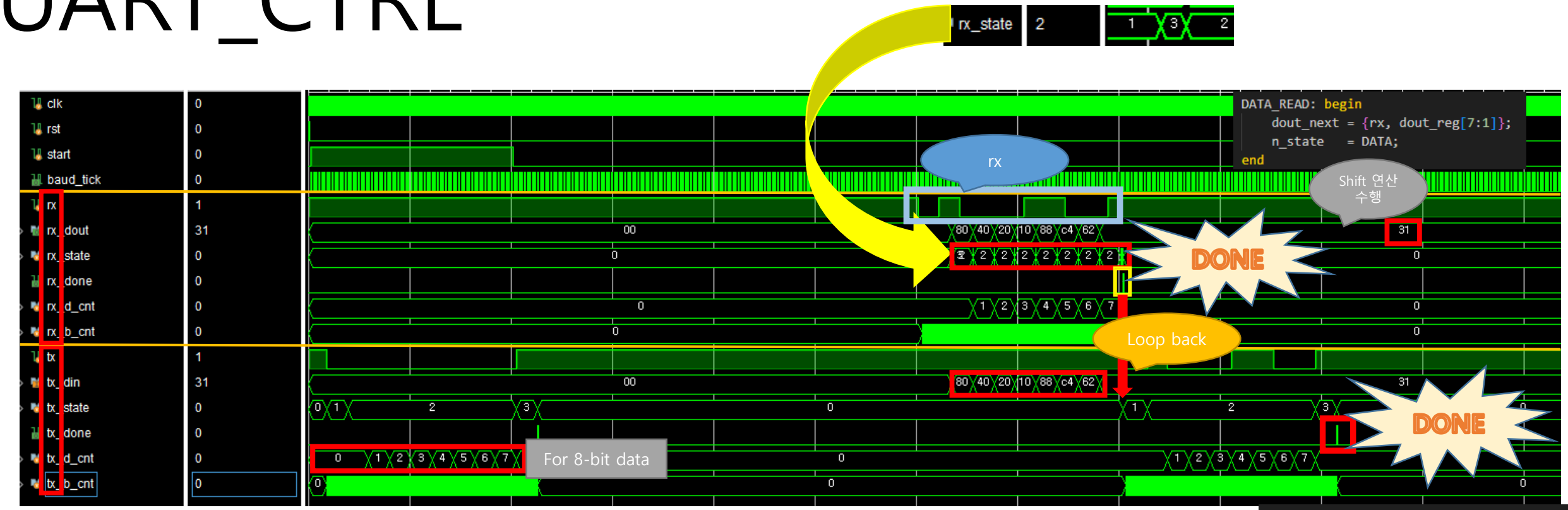
UART_CTRL



Button data	BtnC	BtnU
Push	Reset	tx_start

rx_data	G	C	U	D	M	E
기능	Run/Stop	Clear	Up	Down	Mode	HM/SM

UART_CTRL



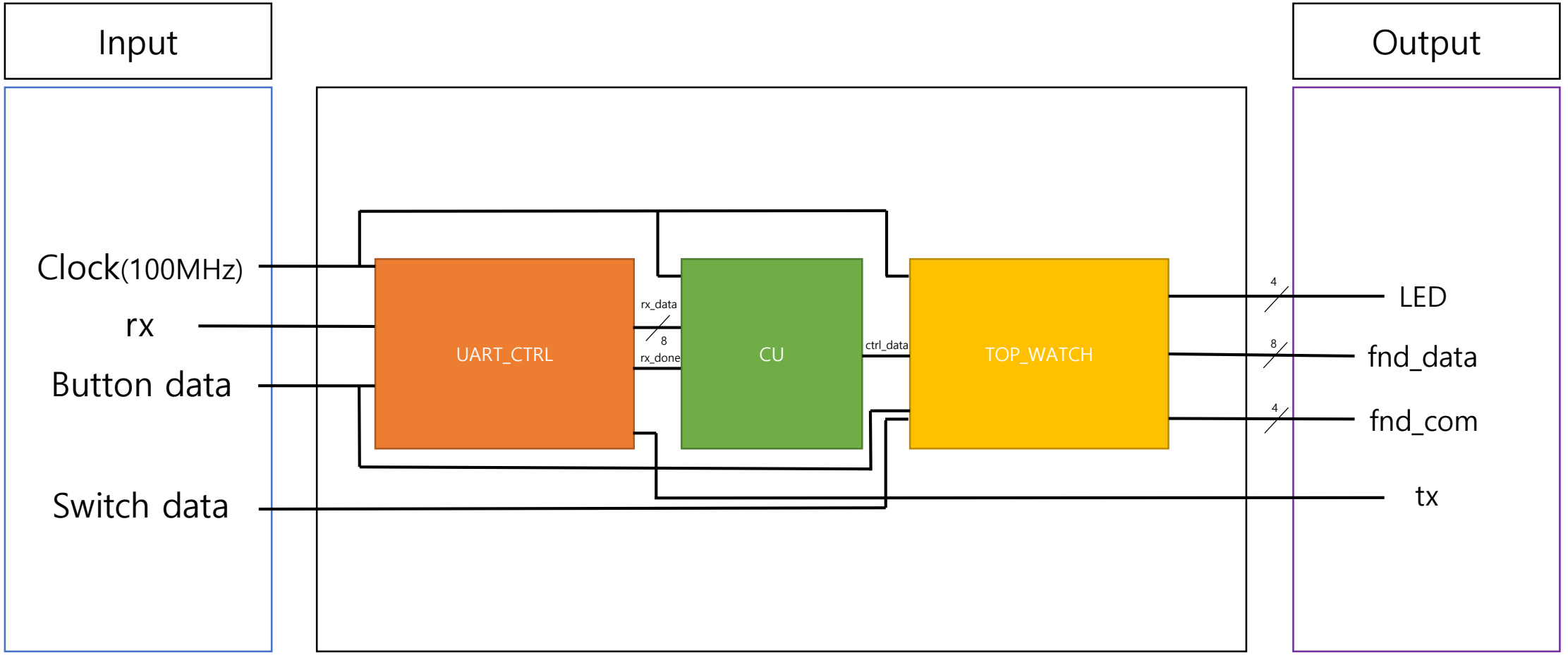
rx_state	STATE
0	IDLE
1	START
2	DATA
3	DATA_READ
4	STOP

tx_state	STATE
0	IDLE
1	START
2	DATA
3	STOP

Binary	00110001
HEX	31

```
initial begin
    #0; clk = 0; rst = 1; start = 0; rx = 1;
    #20; rst = 0;
    #10000; start = 1;//BUTTON PUSH
    #1000000; start = 0;
    #2000000; rx = 0;  //start
      #104160; rx = 1;  //d0
      #104160; rx = 0;
      #104160; rx = 0;
      #104160; rx = 0;
      #104160; rx = 1;
      #104160; rx = 1;
      #104160; rx = 0;
      #104160; rx = 0;  //d7
      #104160; rx = 1;  //stop
    #200000000;
    $stop;
end
```

TOP


Input
Output
Clock(100MHz)
rx
Button data
Switch data
UART_CTRL
rx_data
8
rx_done
CU
ctrl_data
TOP_WATCH
4
LED
8
fnd_data
4
fnd_com
tx

CU

Input

Clock(100MHz)

reset

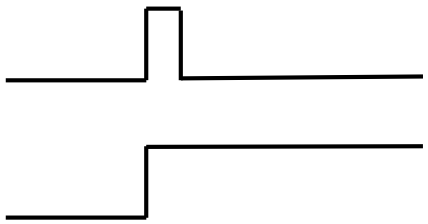
rx_data

rx_done

```
module CU(
    input           clk,
    input           rst,
    //  G, C, U, D, , 1, 2, 3, M, E
    input      [7:0] rx_data,
    input           rx_done,
    // 틱으로 발생하는 output
    output          go_stop,
    output          clear,
    output          up,
    output          down,
    // 긴 신호로 발생하는 output
    output reg      hour_ctrl,
    output reg      min_ctrl,
    output reg      sec_ctrl,
    output reg      mode,
    output reg      hs_mode
);
    reg tick_reg, tick_next, zero;

    assign go_stop = (rx_data == 8'h47) ? tick_reg : 0; // G
    assign clear = (rx_data == 8'h43) ? tick_reg : 0; // C
    assign up = (rx_data == 8'h55) ? tick_reg : 0; // U
    assign down = (rx_data == 8'h44) ? tick_reg : 0; // D

    always @(posedge clk, posedge rst) begin
        if (rst) begin
            tick_reg <= 0;
            hour_ctrl <= 0;
            min_ctrl <= 0;
            sec_ctrl <= 0;
            mode <= 0;
            hs_mode <= 0;
            zero <= 0;
        end else begin
            tick_reg <= rx_done & tick_next;
            if(rx_done) begin
                case (rx_data)
                    8'h31: hour_ctrl <= (hour_ctrl == 0)? 1: 0; // 1
                    8'h32: min_ctrl <= (min_ctrl == 0)? 1: 0; // 2
                    8'h33: sec_ctrl <= (sec_ctrl == 0)? 1: 0; // 3
                    8'h4d: mode <= (mode == 0)? 1: 0; // M
                    8'h45: hs_mode <= (hs_mode == 0)? 1: 0;// E
                    default: zero <= 0;
                endcase
            end
        end
    end

    always @(*) begin
        tick_next = tick_reg;
        case (rx_data)
            8'h47: tick_next = 1'b1;
            8'h43: tick_next = 1'b1;
            8'h55: tick_next = 1'b1;
            8'h44: tick_next = 1'b1;
            default: tick_next = 1'b0;
        endcase
    end
endmodule
```

Button

Switch



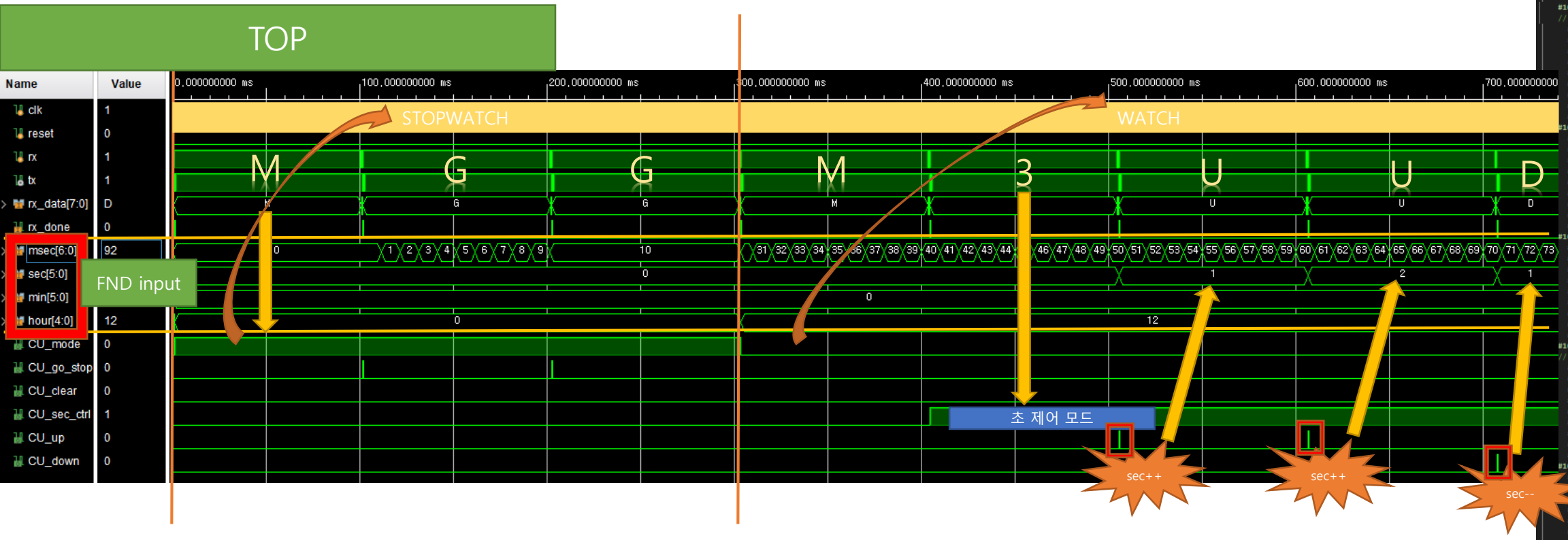
rx_data	output	특징
G	go_stop	button
C	clear	button
U	up	button
D	down	button
1	hour_ctrl	switch
2	min_ctrl	switch
3	sec_ctrl	switch
M	mode	switch
E	hs_mode	switch

Output

ctrl data

결과

시뮬레이션

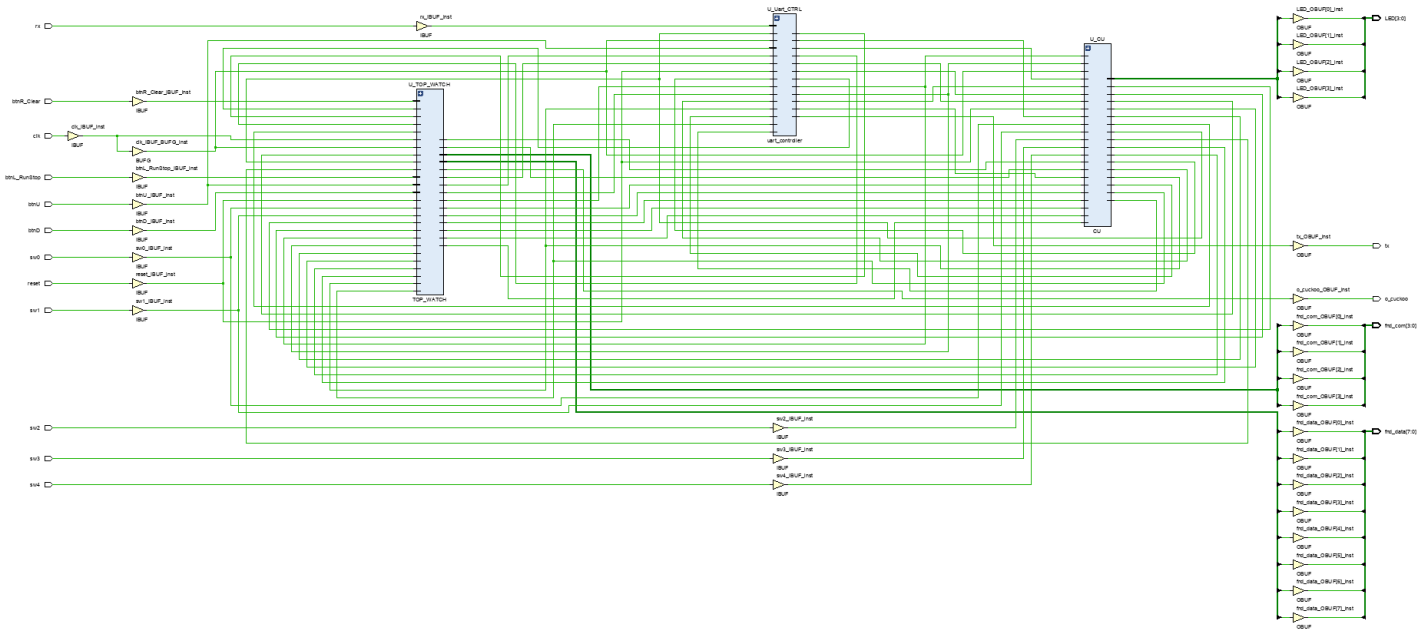


rx_data	G	C	U	D	M	E
제어	Go/Stop	Clear	Up	Down	Mode	HM/SM

rx_data	1	2	3
제어	시	분	초

```
initial begin
    #0; clk = 0; reset = 1; btnU = 0; btnL_R
    sw3 = 0; sw4 = 0; rx = 1;
    #20; reset = 0;
    #10; rx = 0;//start
      #104160; rx = 1;
      #104160; rx = 0;
      #104160; rx = 1;
      #104160; rx = 1;
      #104160; rx = 0;
      #104160; rx = 0;
      #104160; rx = 1;
      #104160; rx = 0;// M => MODE 변경
      #104160; rx = 1;// STOP
    #100000000; rx = 0;//start
    // STOP WATCH 모드
      #104160; rx = 1;
      #104160; rx = 1;
      #104160; rx = 1;
      #104160; rx = 0;
      #104160; rx = 0;
      #104160; rx = 0;
      #104160; rx = 1;
      #104160; rx = 0;// G => START
      #104160; rx = 1;// STOP
    #100000000; rx = 0;//start
      #104160; rx = 1;
      #104160; rx = 1;
      #104160; rx = 1;
      #104160; rx = 0;
      #104160; rx = 0;
      #104160; rx = 0;
      #104160; rx = 1;
      #104160; rx = 0;// G
      #104160; rx = 1;// STOP
    #100000000; rx = 0;//start
      #104160; rx = 1;
      #104160; rx = 0;
      #104160; rx = 1;
      #104160; rx = 1;
      #104160; rx = 0;
      #104160; rx = 0;
      #104160; rx = 1;
      #104160; rx = 0;// M => MODE 변경
      #104160; rx = 1;// STOP
    #100000000; rx = 0;//start
    // WATCH 모드
      #104160; rx = 1;
      #104160; rx = 1;
      #104160; rx = 0;
      #104160; rx = 0;
      #104160; rx = 1;
      #104160; rx = 1;
      #104160; rx = 0;
      #104160; rx = 0;// 3 => 초 제어 모드
      #104160; rx = 1;// STOP
    #1000000; rx = 0;//start
      #104160; rx = 1;
      #104160; rx = 0;
      #104160; rx = 1;
      #104160; rx = 0;
      #104160; rx = 1;
      #104160; rx = 0;
      #104160; rx = 1;
      #104160; rx = 0;// U => UP
      #104160; rx = 1;// STOP
    #10000; rx = 0;//start
      #104160; rx = 1;
      #104160; rx = 0;
      #104160; rx = 1;
      #104160; rx = 0;
      #104160; rx = 1;
      #104160; rx = 0;
      #104160; rx = 1;
      #104160; rx = 0;// U => UP
      #104160; rx = 1;// STOP
    #10000; rx = 0;//start
      #104160; rx = 0;
      #104160; rx = 0;
      #104160; rx = 1;
      #104160; rx = 0;
      #104160; rx = 0;
      #104160; rx = 0;
      #104160; rx = 1;
      #104160; rx = 0;// D => UP
      #104160; rx = 1;// STOP
    #1000000000;
    $stop;
end
```

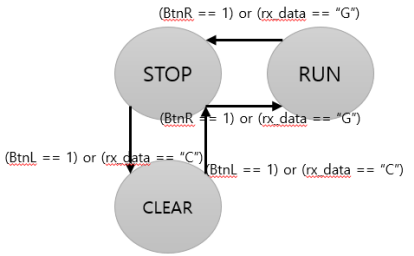
합성



Name	Slice LUTs (20800)	Slice Registers (41600)	Bonded IOB (106)	BUFGCTRL (32)
ToP	345	278	30	1
U_CU (CU)	16	6	0	0
U_TOP_WATCH (TOP_WATCH)	251	208	0	0
U_Uart_CTRL (uart_controller)	78	64	0	0

시연영상

rx_data	기능
G	Run/Stop
C	Clear
U	Up
D	Down
M	Mode
E	HM/SM
1	시 제어
2	분 제어
3	초 제어



참고: 기능 추가 전 영상

추가 기능

SPEC

정각이 되면 지정된 LED 1분 동안 ON

단, 00:00~07:00 동작 X for 수면시간 보장
WATCH, STOPWATCH 둘다 동작 가능

Module

기능만 구현한 코드

```
module cuckoo(
    input rst,
    input [4:0] hour,
    input [5:0] min,
    output o_cuckoo
);
    reg r_cuckoo;
    reg [5:0] prev_min, r_min;
    assign o_cuckoo = r_cuckoo;

    always @(*) begin
        r_min = min;
        if(rst) begin
            r_cuckoo = 1'b0;
            prev_min = r_min;
        end else begin
            if(prev_min != r_min) begin
                if((prev_min == 6'd59) && (r_min == 6'd00)) begin
                    r_cuckoo = 1'b1;
                end else begin
                    r_cuckoo = 1'b0;
                    prev_min = r_min;
                end
            end
            if((5'd0 <= hour) && (hour < 5'd7)) begin
                r_cuckoo = 1'b0;
            end
        end
    end
endmodule
```

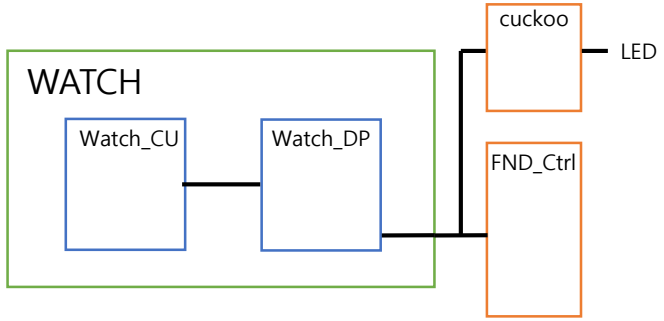
Latch 발생

오동작 발생

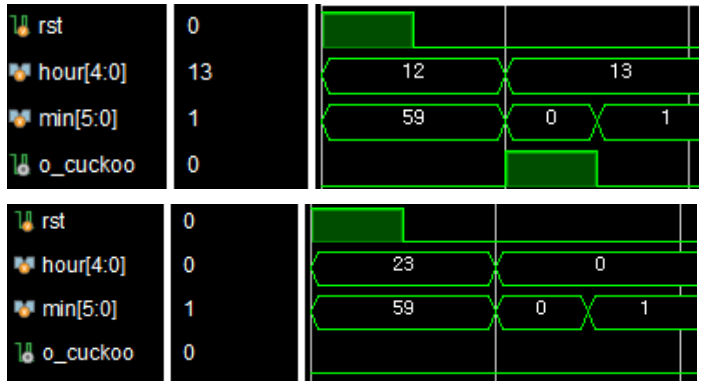
문제해결한 코드

```
`timescale 1ns / 1ps

module cuckoo (
    input  [4:0] hour,
    input  [5:0] min,
    output       o_cuckoo
);
    assign o_cuckoo = (min == 6'd0) & (hour>=7);

endmodule
```

해결



Simulation



시행착오

파일 저장	Vivado simulation	파일 정리
꼭 필요한 파일은 미리미리 저장하자!	Simulation이 동작하지 않는다면, 잠시 쉬는 시간	함부로 파일 경로를 변경하면 안된다.

느낀 점

1. 파일은 미리미리 저장하자!
2. 처음부터 완벽하게 구성하기보다는 구현하고자 하는 기능을 만든 후 최적화시키자!
3. 막히는 부분이 생기면, 잠시 머리를 식히자!

프로젝트를 하며 배운 점

1. 막히는 부분이 생기면, 잠시 머리를 식히자!
2. 파일은 미리미리 저장하자!
3. 처음부터 완벽하게 구성하기보다는 구현하고자 하는 기능을 만든 후 최적화시키자!

감사합니다.