Integrated Clock

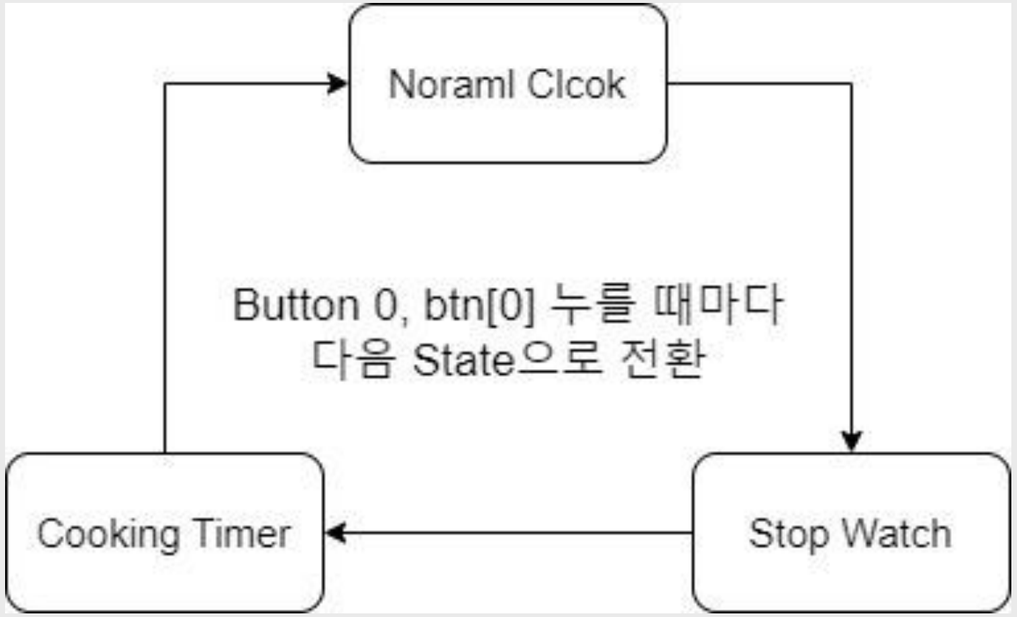
교육생 전 병 혁

Contents

설계 환경

Design Environment

1. State Diagram



- 모듈 전환 버튼인 Button 0, btn[0] 을 누를 때마다 다음 state로 전이된다.
- 이번 구현에서는 next_state 변수 없이 state 변수만 사용

Design Environment

1. Watch mode
- Watch / Set 모드 전환 버튼
- 분 컨트롤 버튼
- 초 컨트롤 버튼

2. Stop Watch mode
- Start 버튼
- Lap 버튼
- Clear 버튼

3. Cooking Timer mode
- Start 버튼
- 분 컨트롤버튼
- 초 컨트롤 버튼
- Alarm off 버튼

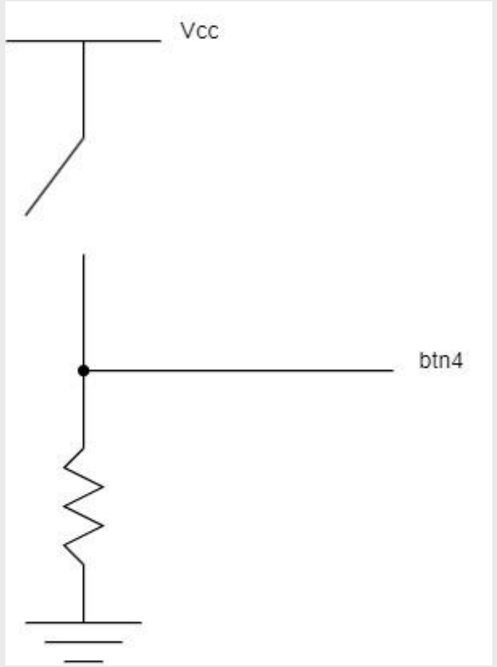
	Btn[0]	Btn[1]	Btn[2]	Btn[3]
Clock	모드 전환 버튼	분 컨트롤	초 컨트롤	Watch Set 버튼
Stop Watch		시작 버튼	Lap 버튼	Clear 버튼
Cooking timer		시작 버튼	초 컨트롤	분 컨트롤

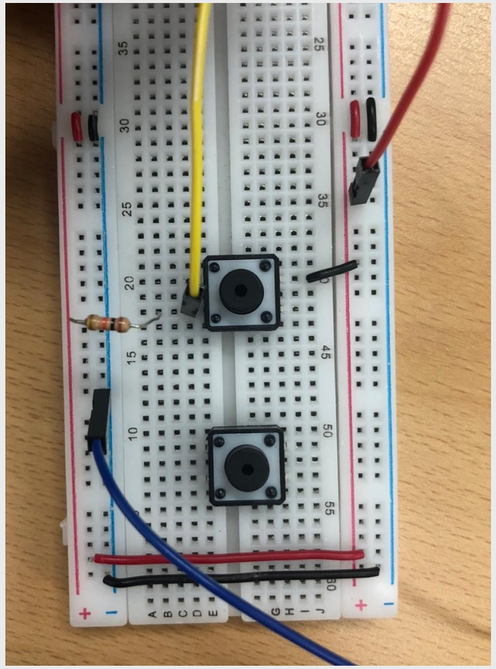
Design Environment

1. Cooking Timer 에서 부족한 1개의 버튼에 대해서는 외부에서 Pull-down Switch 생성하여 사용

2. 외부 버튼의 신호 (btn4) 의 신호를 받기 위해 JC Pin Header의 1번 핀 사용

3. Cooking Timer에서 사용할 Alarm은 JB Pin Header의 4번 핀 사용

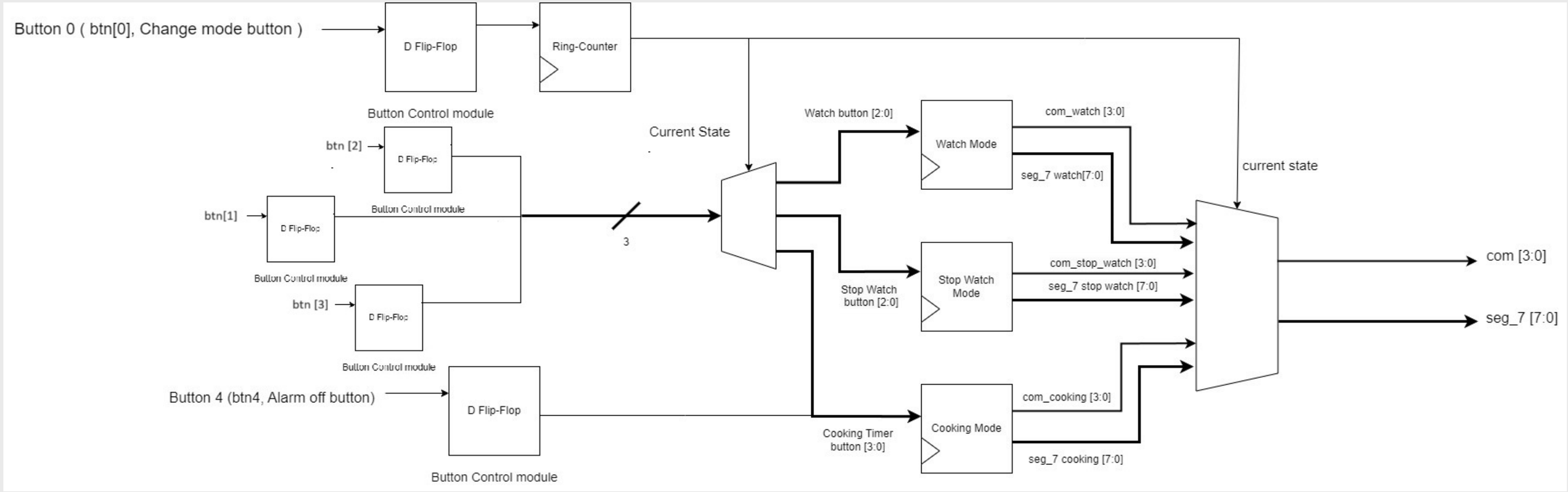




1차 설계

1차 설계

0. Block diagram



1. Declare Input / Output variables of Module

```
module Integrated_Watch_Project(
    input clk, reset_p,
    input [3:0] btn,
    input btn4,
    output [3:0] com,
    output [7:0] seg_7,
    output [2:0] led_debug,
    output buzz
);
```

- btn [0] : 모드 전환 버튼
- btn [1] ~ btn [3] : 모든 모드들이 사용하는 공통 버튼
- btn4 : Cooking Timer 만 사용하는 버튼이며, 알람 끄는 버튼

2. Declare state parameter

```
// declare state parameter
parameter S_CLOCK         = 3'b001;
parameter S_STOP_WATCH    = 3'b010;
parameter S_COOKING_TIMER = 3'b100;

// LED for state
assign led_debug = state;
```

- S_CLOCK : 일반 시계 모드
- S_STOP_WATCH : 스톱 워치 모드
- S_COOKING_TIMER : Cook Timer 모드

3. Get one cycle pulse of button

```
// Get One Cycle pulse of button0
wire clk_btn0, clk_btn1, clk_btn2, clk_btn3, clk_btn4;
button_cntr btn_cntr_btn0(.clk(clk), .reset_p(reset_p), .btn(btn[0]), .btn_pedge(clk_btn0));
button_cntr btn_cntr_btn1(.clk(clk), .reset_p(reset_p), .btn(btn[1]), .btn_pedge(clk_btn1));
button_cntr btn_cntr_btn2(.clk(clk), .reset_p(reset_p), .btn(btn[2]), .btn_pedge(clk_btn2));
button_cntr btn_cntr_btn3(.clk(clk), .reset_p(reset_p), .btn(btn[3]), .btn_pedge(clk_btn3));
button_cntr btn_cntr_btn4(.clk(clk), .reset_p(reset_p), .btn(btn4), .btn_pedge(clk_btn4));
```

- button_cntr 모듈을 통해 1ms delay time을 갖고, button 값을 읽는다.
- button를 누르면 버튼 값이 0 -> 1로 변하면서 Positive edge 발생

4. Declaring necessary variables

```
// declare state, next_state variable
reg [2:0] state;

// declare com, seg_7 variables.
wire [3:0] com_clock, com_stop, com_cook;
wire [7:0] seg_7_clock, seg_7_stop, seg_7_cook;

// Declare Button
reg [2:0] clock_button, stop_button;
reg [3:0] cook_button;
```

- 각 모듈의 인스턴스에 버튼 값을 각각 주기 위해 서로 다른 버튼 변수 선언
- 각 모듈의 출력 값인 com, seg_7을 받기 위해 서로 다른 com, seg_7 변수 선언

5. Variable initialization

```
// State transition and button processing logic
    always @(posedge clk or posedge reset_p) begin
        if(reset_p) begin
            state = S_CLOCK;
            clock_button = 3'b000;
            stop_button = 3'b000;
            cook_button = 4'b0000;
       end else begin

                ''' (생략) '''


        end
    end
```

- state 변수와 button 변수 초기화

6. The state changes every time the button 0 is pressed.

1차 설계

```
// State transition and button processing logic
    always @(posedge clk or posedge reset_p) begin
        if(reset_p) begin
            state = S_CLOCK;
            clock_button = 3'b000;
            stop_button = 3'b000;
            cook_button = 4'b0000;
        end else begin
            if(clk_btn0) begin // if pressed button 0
                case(state)
                    S_CLOCK: state = S_STOP_WATCH;
                    S_STOP_WATCH: state = S_COOKING_TIMER;
                    S_COOKING_TIMER: state = S_CLOCK;
                endcase
            end
            else begin
               ''' (생략) '''
            end
        end
    end
```

- clk_btn0 변수가 활성화 될때 마다 다음 state로 전이한다.

7. Setting button values based on state values

1차 설계

```
case(state)
        S_CLOCK: begin
                clock_button = {clk_btn1, clk_btn2, clk_btn3};
                stop_button = 3'b000;
                cook_button = {clk_btn4, 3'b000};
        end
        S_STOP_WATCH: begin
                clock_button = 3'b000;
                stop_button = {clk_btn3, clk_btn2, clk_btn1};
                cook_button = {clk_btn4, 3'b000};
       end
       S_COOKING_TIMER: begin
                clock_button = 3'b000;
                stop_button = 3'b000;
                cook_button = {clk_btn4, clk_btn3, clk_btn2, clk_btn1};
       end
endcase
```

- Alarm off 버튼는 모든 모드에서 동작해야 하기 때문에 모든 경우의 수에 대해서 Btn4가 눌러졌을 때, 알람을 끄기 위해 cook_button을 위와 같이 설계

8. Declare an instance, then substitute argument

```
// Module instantiations with direct button connections
watch_top watch_mode (.clk(clk), .reset_p(reset_p), .btn(clock_button), .com(com_clock), .seg_7(seg_7_clock));

stop_watch_top stop_watch_mode (.clk(clk), .reset_p(reset_p), .btn(stop_button), .com(com_stop), .seg_7(seg_7_stop));

cook_timer_top cook_timer_mode (.clk(clk), .reset_p(reset_p), .btn(cook_button), .com(com_cook), .seg_7(seg_7_cook), .buzz(buzz));
```

- 각 모드의 버튼 값을 인자 값으로 전달한다.
- 각 모드의 출력 값인 com, seg_7을 전달 받은 뒤, 각 모드의 com, seg_7 변수에 각각 저장한다.

9. According to the current state, com, seg_7 values are selected and output.

```
// Output selection based on current state
assign com = (state == S_CLOCK) ? com_clock :
             (state == S_STOP_WATCH) ? com_stop : com_cook;
assign seg_7 = (state == S_CLOCK) ? seg_7_clock :
               (state == S_STOP_WATCH) ? seg_7_stop : seg_7_cook;
```

- 현재 State 변수 값에 따라 현재 수행 중인 모드의 com, seg_7 값을 선택한 후, FND로 출력한다.

1차 설계

문제 현상 (버튼 1~4 동작하지 않는 문제점 발생)

(문제 현상)

문제 원인 (One Cycle Pulse를 argument 값으로 전달)

(문제 원인)

```
reg [20:0] clk_div = 0; //레지스터에 0주는건 시뮬레이션에서만 가능, 회로에서는 레지스터에 0을 줄수없기때문 리셋으로 0줄수있음
always @(posedge clk)clk_div = clk_div + 1;

wire clk_div_nedge; //와이어에 0주는건 접지시키는것, 거기에 1들어가면 쇼트
edge_detector_p ed_clk(.clk(clk), .reset_p(reset_p), .cp(clk_div[16]), .n_edge(clk_div_nedge));

reg debounced_btn;
always @(posedge clk or posedge reset_p) begin
        if(reset_p) debounced_btn = 0;
        else if(clk_div_nedge) debounced_btn = btn;
end

edge_detector_p ed_btn(.clk(clk), .reset_p(reset_p), .cp(debounced_btn), .n_edge(btn_nedge), .p_edge(btn_pedge));
```

- button_cntr 모듈은 1ms delay time을 갖고, Button 값을 읽기 때문에

  One Cycle Pulse을 argument 값으로 전달하면 button 값을 읽을 수 없다.

1차 설계

문제 해결 방안 (One Cycle Pulse 대신에 btn pulse wave 전달)

(문제 원인)

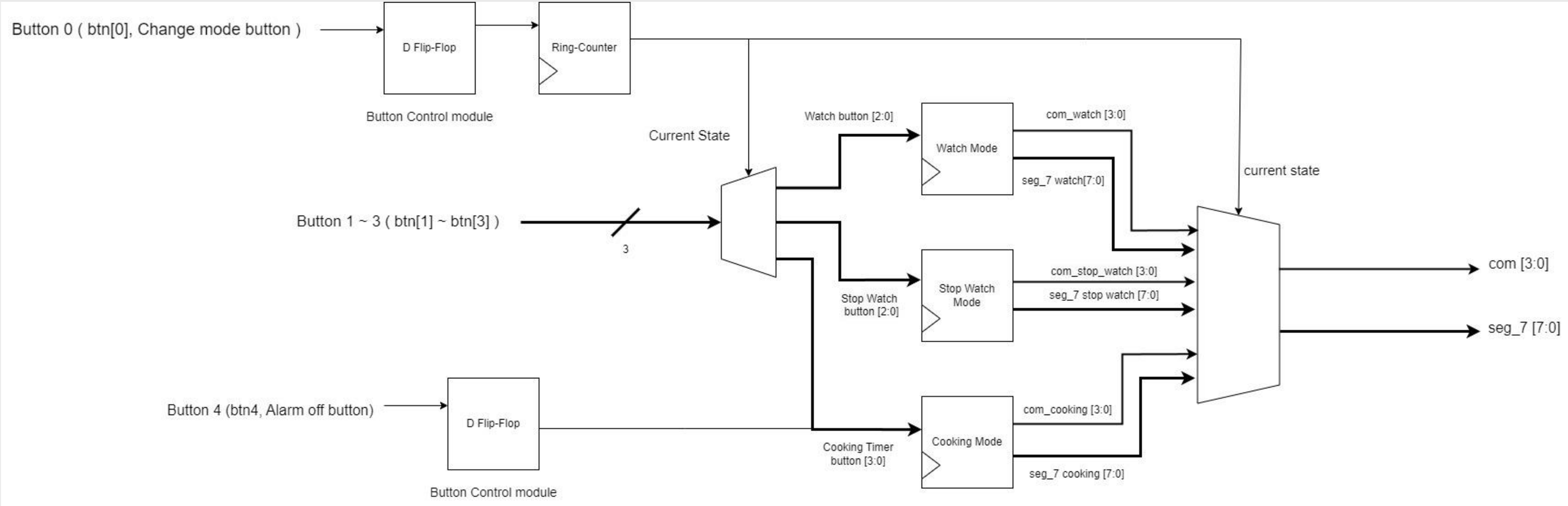
```
case(state)
        S_CLOCK: begin
              clock_button = {btn[1], btn[2], btn[3]};
              stop_button = 3'b000;
              cook_button = {btn4, 3'b000};
        end
        S_STOP_WATCH: begin
              clock_button = 3'b000;
              stop_button = {btn[3], btn[2], btn[1]};
              cook_button = {btn4, 3'b000};
       end
       S_COOKING_TIMER: begin
             clock_button = 3'b000;
             stop_button = 3'b000;
             cook_button = {btn4, btn[3], btn[2], btn[1]};
       end
endcase
```

- 버튼의 One Cycle Pulse 대신 버튼의 Pulse wave를 버튼 값에 저장하여 argument로 전달

최종 설계

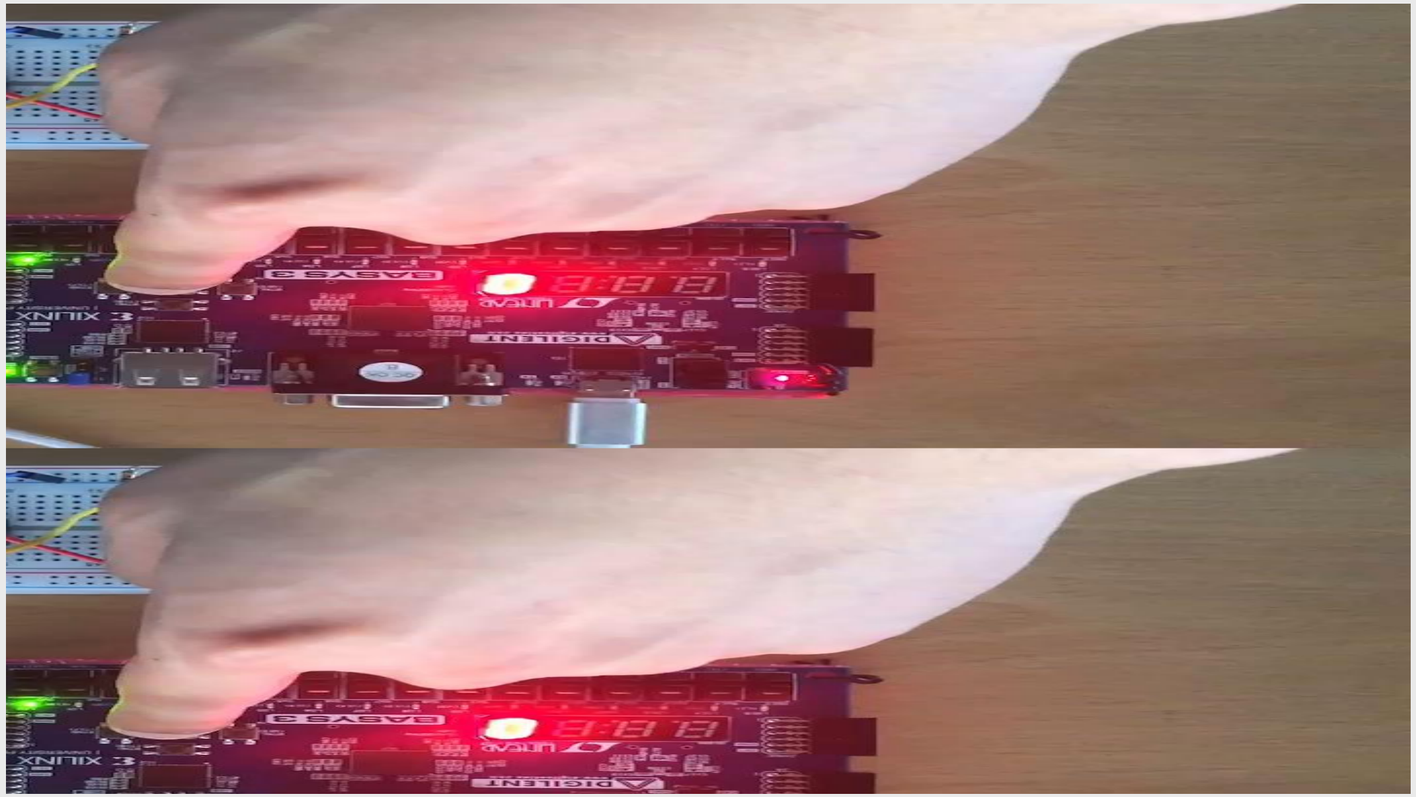
최종 설계

Block diagram



구현 영상

구현 영상



결론

결론

Positive Slack

Name	Constraints	Status	WNS
✔ synth_1	constrs_1	synth_design Complete!	
✔ impl_1	constrs_1	write_bitstream Complete!	2.138

Number of LUT, Flip-Flop

Total Power	Failed Routes	LUT	FF
		277	298
0.080	0	273	298

Q & A

들어주셔서 감사합니다.

SoC 반도체 회로설계 아카데미

전 병 혁