디지털시스템설계 <텀프로젝트 결과 보고서>

제출일 : 2019-12-09;
이름 : 박상준(2016124103), 양해찬(2016124145), 이정헌(2016124175); 10조

✔ 설계 목표 설정 결과

● 설계 목표 결과

시계 기본 기능(시계, 타이머, 스톱워치) 및 외부 모듈 블루투스 모듈, 심장박동수 모듈을 추가하여 심장박동수를 휴대폰으로 출력하고 휴대폰으로 모드변경, 리셋, 프리셋, 시작, 정지 등의 다양한 기능을 휴대폰으로 컨트롤하는 기능 구현.

- 디지털 시계 및 알람 :

◎ 시(HEX5), 분(HEX4,HEX3), 초(HEX2,HEX1) 0.1초(HEX0) 단위로 순서대로 디스플레이
◎ 임의의 값을 설정하는 프리셋(어플 프리셋 버튼)기능은 시,분 까지 가능
◎ 리셋 (어플 리셋버튼, 음성인식)을 통해 00:00:00으로 초기화
◎ LED[9]로 AM, PM을 구분 AM(LEDR[9] off),PM(LEDR [9] on)
◎ 각 7-Segment에 디스플레이 되는 시,분단위의 현재 시각을 알람으로 프리셋(KEY[1])하고 설정했던 시각이 되면 휴대폰에 알람 출력.

- 스톱 워치 :

◎ 1/100초 단위까지 표현
◎ START : 어플 start버튼을 눌러 Start신호를 주면 카운트 업
◎ STOP : 어플 Stop 버튼을 눌러 Stop신호를 주면 정지.
◎ 리셋 : 어플 Reset 버튼, 음성인식

- 타이머 :

◎ 시(HEX5), 분(HEX4,HEX3), 초(HEX2,HEX1) 0.1초(HEX0)차례로 디스플레이
◎ 임의의 값(시,분 단위) 을 프리셋(어플 프리셋 버튼)하면 그 값부터 00:00:00까지 카운트 다운된다.
◎ Stop(어플 stop버튼)으로 일시 정지, Reset(어플 리셋버튼, 음성인식) 으로 초기화 할 수 있다.
◎ 타이머가 0이 되면 어플에 알람 출력

- 심장박동수 측정 :

◎ 어플 실행화면 상단에 출력된다(BPM)

디지털시스템설계 <텀프로젝트 결과 보고서>

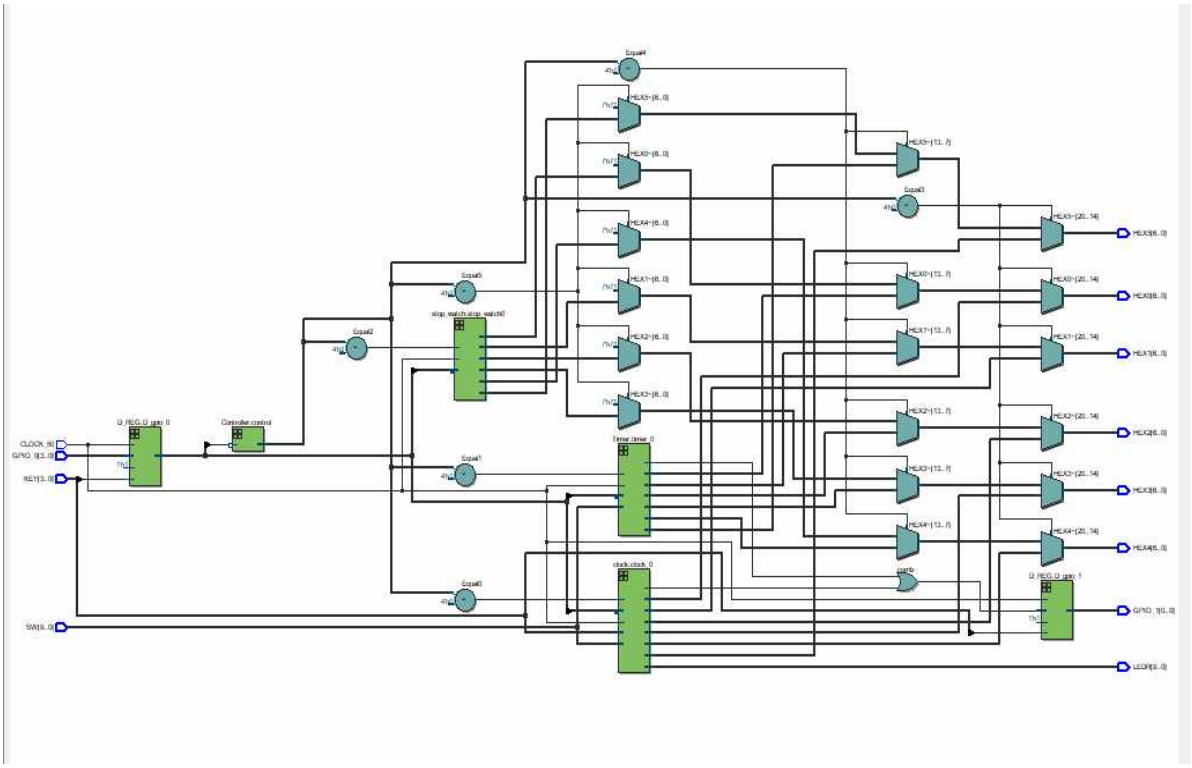
● 구현기능 및 사용부품

목표기술 및 최종결과 및 사용부품	달성여부	비고
1. 전자시계	O	
2. 스탑워치	O	
3. 타이머	O	
4. 알람	O	
5. 어플 보드와 연동	O	
6. 심장박동수 모듈	O	

● 설계 사양

어플을 통해서 시계,타이머,스톱워치 모드를 선택하고 그것이 각 모드의 Enable신호로 입력, Enable신호가 입력된 모드만 출력될 수 있도록 설계

<RTL VIEW>



디지털시스템설계 <텀프로젝트 결과 보고서>

I/O Spec(Top Block)

I/O	Signal Name	Bit	Description
I	SW	10	Input Switch Data
I	KEY[1:0]	2	Input Alarm Preset Signal, Input Alarm Reset Sigmal
I	CLOCK_50	1	Input Clock
I	GPIO_0	4	Input Preset Signal, Reset Signal, Start/Stop Signal, Mode Change Sigmal
O	LEDR	10	Output LED Data
O	GPIO_1	1	Alarm output
O	HEX[5:0]	7	Output Segment Data

<top-block 코드>

```
module project(
            KEY,
            SW,
            CLOCK_50,
            LEDR,
            HEX0,
            HEX1,
            HEX2,
            HEX3,
            HEX4,
            HEX5,
            GPIO_0,
            GPIO_1

);
parameter I=4; //soc보드 기준 외부에서 받는 input parameter
parameter O=1; //soc보드 기준 외부에서 받는 output parameter
input       [1:0]       KEY; //KEY[1]: clock모듈 preset KEY[0]: clock모듈 reset, GPIO핀 입출력 값 reset
input       [9:0]       SW; //SW[9:8]: clock모듈에서 시 분 초 중 preset 할 항목을 select, SW[4:0]: clock 모듈 preset data
input                   CLOCK_50;
input       [I-1:0]     GPIO_0; //soc보드 기준 input
output      [O-1:0]     GPIO_1; //soc보드 기준 output

output      [6:0]       HEX0,HEX1,HEX2,HEX3,HEX4,HEX5; // HEX5: 시단위 HEX4: 10분단위 HEX3: 1분단위 HEX2: 10초단위 HEX1: 1초단위 HEX0: 0.1초단위
output      [9:0]       LEDR;




wire                    clock_En,timer_En, stop_watch_en,oEN;
wire        [3:0]       state;
wire        [6:0]       c_HEX0,c_HEX1,c_HEX2,c_HEX3,c_HEX4,c_HEX5; //clock 모듈 출력 data 저장
wire        [6:0]       t_HEX0,t_HEX1,t_HEX2,t_HEX3,t_HEX4,t_HEX5; //timer 모듈 출력 data 저장
wire        [6:0]       s_HEX0,s_HEX1,s_HEX2,s_HEX3,s_HEX4,s_HEX5; //stop_watch 모듈 출력 data 저장

wire        [I-1:0]     gpio_0_tmp; //아두이노에서 soc보드로 입력되는 값을 저장
wire        [O-1:0]     gpio_1_tmp; //timer alarm
wire        [0:0]       gpio_1_1_tmp; //clock alarm
assign clock_En = state == 3'b000 ?         1'b1:1'b0;
assign timer_En = state == 3'b001 ?         1'b1:1'b0;
```

```
assign stop_watch_en = state == 3'b010 ? 1'b1 : 1'b0;
//controller 모듈에서 지정한 state에 맞게 clock,timer,stop_watch가 작동할 수 있도록 각각 enable 신호 설정

D_REG#(.WL(I))
D_gpio_0(.iRSTn(KEY[0]),.iCLK(CLOCK_50),.iDATA(GPIO_0[I-1:0]),.iEN(1'b1),.oDATA(gpio_0_tmp[I-1:0]));
D_REG#(.WL(O))
D_gpio_1(.iRSTn(KEY[0]),.iCLK(CLOCK_50),.iDATA(gpio_1_1_tmp[0]|gpio_1_tmp[O-1:0]),.iEN(1'b1),.oDATA(GPIO_1[O-1:0]));
//soc보드와 아두이노의 입출력 사이에 클럭을 맞추기 위함

Controller
control(
        .key(~gpio_0_tmp[3]),
        .state(state)
); //gpio_0_tmp로 state변경

clock
clock_0(
        .KEY(~gpio_0_tmp[1:0]),
        .sw(SW),
        .EN(clock_En),
        .clk(CLOCK_50),
        .oEN(oEN),
        .led(LEDR[9:0]),
        .hex0(c_HEX0),
        .hex1(c_HEX1),
        .hex2(c_HEX2),
        .hex3(c_HEX3),
        .hex4(c_HEX4),
        .hex5(c_HEX5)
        ,.alarm(gpio_1_tmp[0:0]),.key(KEY[1:0]))
;

Timer
timer_0(.Oalarm(gpio_1_1_tmp[0:0]),
        .key(~gpio_0_tmp[2:0]),
        .sw(SW),
        .EN(timer_En),
        .clk(CLOCK_50),
        .hex0(t_HEX0),
        .hex1(t_HEX1),
        .hex2(t_HEX2),
        .hex3(t_HEX3),
        .hex4(t_HEX4),
        .hex5(t_HEX5)
);
stop_watch
stop_watch0(
        .key(~gpio_0_tmp[1:0]),
        .sw(SW[1:0]),
        .EN(stop_watch_en),
        .clk(CLOCK_50),
        .hex0(s_HEX0),
        .hex1(s_HEX1),
        .hex2(s_HEX2),
        .hex3(s_HEX3),
        .hex4(s_HEX4),
        .hex5(s_HEX5)
);
```

```
assign HEX0 = state==3'b000 ? c_HEX0:
                                        state == 3'b001 ?    t_HEX0:
                                    state == 3'b010 ?  s_HEX0:7'b1000111;


assign HEX1 = state==3'b000 ? c_HEX1:
                                        state == 3'b001 ?    t_HEX1:
                                    state == 3'b010 ?  s_HEX1:7'b1000111;

assign HEX2 =  state==3'b000 ? c_HEX2:
                                        state == 3'b001 ?    t_HEX2:
                                    state == 3'b010 ?  s_HEX2:7'b1000111;


assign HEX3 =  state==3'b000 ? c_HEX3:
                                        state == 3'b001 ?    t_HEX3:
                                    state == 3'b010 ?  s_HEX3:7'b1000111;


assign HEX4 = state==3'b000 ? c_HEX4:
                                        state == 3'b001 ?    t_HEX4:
                                    state == 3'b010 ?  s_HEX4:7'b1000111;


assign HEX5 = state==3'b000 ? c_HEX5:
                                        state == 3'b001 ?    t_HEX5:
                                    state == 3'b010 ?  s_HEX5:7'b1000111;



//지정한 state에 맞는 mode의 출력값이 soc보드로 출력될 수 있도록 설정

endmodule
```

디지털시스템설계 <텀프로젝트 결과 보고서>

<Digital Clock+Alarm>

I/0 spec (Clock module)

I/O	Signal Name	Bit	Description
I	key	2	KEY[1]:알람 (시,분)설정,KEY[0]:알람 설정값 리셋
I	sw	10	SW[9:8]=시,10분,1분 프리셋할지 선택 키, SW[4:0]=시간 프리셋용, SW[2:0]=10분 프리셋, SW[3:0]=1분 프리셋
I	clk	1	시계가 작동할 때 사용하는 CLERK
I	EN	1	모드 변경시 각 상태(시계,타이머,스탑워치)에 할당된 EN을 통해 모드 설정.
I	KEY	2	KEY[0]:리셋,KEY[1]:프리셋을 각각 GPIO0[1:0]에 탑블록에서 할당하여 사용.
O	oEN	1	시계의 시 단위 oEN값
O	led	10	LED[9]:AM/PM 표시 나머지 ;LED[8:0]=9'b000000000
O	alarm	1	알람 설정 값이 되면 1출력,아니면 0출력 탑블록관 연결되어 gpio_1_tmp[0]에 연결후 DREG통과후 GPIO1[0]에 전달
O	HEX0	7	원하는 출력 값을 Segment로 표시
O	HEX1	7	
O	HEX2	7	
O	HEX3	7	
O	HEX4	7	
O	HEX5	7	

<Clck>

```
module clock(
        KEY,
        sw,
        EN,
        clk,
        led,
        oEN,
        hex0,
        hex1,
        hex2,
        hex3,
        hex4,
        hex5,
        key,
        alarm);

input   [1:0]   KEY; //alarm설정용 KEY
input   [1:0]   key; // reset, preset 등 clock 통제용 key
input   [9:0]   sw; //preset data 입력
input           clk,EN;

output  [6:0]   hex0,hex1,hex2,hex3,hex4,hex5;
output  [9:0]   led;
output          oEN;
output          alarm; //설정한 시간이 되면 1출력
```

```
wire EN_5M, EN_50M, EN_100MS, EN_1S, EN_10S, EN_1MIN, EN_10MIN, EN_1H, EN_3s;
wire      [25:0]          O_50M;
wire      [22:0]          O_5M;
wire      [3:0]           O_100MS, O_1S, O_1MIN;
wire      [2:0]           O_10S, O_10MIN;
wire      [4:0]           O_1H;
wire      [3:0]           hour;
wire      [1:0]           O_3s;
wire      [4:0]           alarm_1H;
wire      [2:0]           alarm_10MIN;
wire      [3:0]           alarm_1MIN;
wire                      preset1,preset2,preset3;
wire                      reset, preset;


reg alarm;
assign preset1 = (sw[9:8] == 2'b01) ? preset : 1'b1;
assign preset2 = (sw[9:8] == 2'b10) ? preset : 1'b1;
assign preset3 = (sw[9:8] == 2'b11) ? preset : 1'b1; //sw[9:8] 로 시,분,초 중 preset 할 항목을 select

assign reset = EN? KEY[0]: 1'b1; //enable 신호가 입력될 때 reset=KEY[0]
assign preset = EN? KEY[1]: 1'b1; //enable 신호가 입력될 때 preset=KEY[1]

assign led[8:1] = 8'b00000000;
assign oEN = EN_1H;



D_REG
#(.WL(3))
U0
(.iRSTn(key[0]),.iCLK(key[1]),.iEN(1'b1),.iDATA(O_1MIN),.oDATA(alarm_1MIN));
D_REG
#(.WL(2))
U1
(.iRSTn(key[0]),.iCLK(key[1]),.iEN(1'b1),.iDATA(O_10MIN),.oDATA(alarm_10MIN));
D_REG
#(.WL(4))
U3
(.iRSTn(KEY[0]),.iCLK(key[1]),.iEN(1'b1),.iDATA(hour),.oDATA(alarm_1H));
//key[1] 입력시 입력 시점의 시 분 초가 알람 설정값으로 저장된다.


COUNTER_LAB
#(.n(23),.k(5000000))]
COUNTER_5M(
          .iCLK(clk),
          .iEN(1'b1),
          .iRSTn(reset),
```

```
        .oEN(EN_5M),
        .oCNT(O_5M)
);

COUNTER_LAB
#(.n(4),.k(10))
COUNTER_100MS(
        .iCLK(clk),
        .iEN(EN_5M),
        .iRSTn(reset),
        .oEN(EN_100MS),
        .oCNT(O_100MS)
);

COUNTER_LAB
#(.n(4),.k(10))
COUNTER_1S(
        .iCLK(clk),
        .iEN(EN_100MS),
        .iRSTn(reset),
        .oEN(EN_1S),
        .oCNT(O_1S)
);

COUNTER_LAB
#(.n(3),.k(6))
COUNTER_10S(
        .iCLK(clk),
        .iEN(EN_1S),
        .iRSTn(reset),
        .oEN(EN_10S),
        .oCNT(O_10S)
);

COUNTER_PRESET
#(.n(4),.k(10))
COUNTER_1M(
        .iCLK(clk),
        .iEN(EN_10S),
        .iRSTn(reset),
        .iPRSTn(preset1),
        .iDATA_Preset(sw[3:0]),
        .oEN(EN_1MIN),
        .oCNT(O_1MIN)
);

COUNTER_PRESET
#(.n(3),.k(6))
COUNTER_10M(
        .iCLK(clk),
        .iEN(EN_1MIN),
```

```
        .iRSTn(reset),
        .iPRSTn(preset2),
        .iDATA_Preset(sw[2:0]),
        .oEN(EN_10MIN),
        .oCNT(O_10MIN)
);

COUNTER_PRESET
#(.n(5),.k(24))
COUNTER_1H(
        .iCLK(clk),
        .iEN(EN_10MIN),
        .iRSTn(reset),
        .iPRSTn(preset3),
        .iDATA_Preset(sw[4:0]),
        .oEN(EN_1H),
        .oCNT(O_1H)
);


AM_PM
am_pm_1(
        .iDATA(O_1H),
        .oDATA(hour),
        .oLED(led[9])
);

Segment_Decoder
Hex0(
        .iDATA(O_100MS),
        .oDATA(hex0)
);

Segment_Decoder
Hex1(
        .iDATA(O_1S),
        .oDATA(hex1)
);

Segment_Decoder
Hex2(
        .iDATA(O_10S),
        .oDATA(hex2)
);

Segment_Decoder
Hex3(
        .iDATA(O_1MIN),
        .oDATA(hex3)
);
```

```
Segment_Decoder
Hex4(
        .iDATA(O_10MIN),
        .oDATA(hex4)
);

Segment_Decoder
Hex5(
        .iDATA(hour),
        .oDATA(hex5)
);
always@*
begin

if((~(alarm_1H==5'b00000)|~(alarm_10MIN==3'b000)|~(alarm_1MIN==4'b0000))&((O_1MIN==alarm_1MIN)&(O_10MI
N==alarm_10MIN)&(O_1H==alarm_1H)))
begin
alarm <= 1'b1;
end
else
begin
alarm <= 1'b0;
end
//알람 설정값과 현재 시간이 같으면 alarm으로 1출력
end
endmodule
```

디지털시스템설계 <텀프로젝트 결과 보고서>

<stop_watch>

I/O spec

I/O	Signal Name	Bit	Description
I	key	2	key[1]:stop버튼,key[0]:reset버튼 각각은 GPIO0[1:0]에 탑블록에서 할당
I	clk	1	Clock 입력
I	EN	1	각 상태(시계,스톱워치,타이머)에 존재하는 EN신호로서 모드변경시 각 모드가 될 때마다 각 EN이 1이됨.
O	HEX5,HEX4	7	10분,1분단위 Display
O	HEX2, HEX3	2	10초,1초 단위 Display
O	HEX4, HEX5	2	0.1초 0.01초 단위 Display

<Stop watch>

```
module stop_watch(
        key,
        EN,
        clk,
        hex0,
        hex1,
        hex2,
        hex3,
        hex4,
        hex5
);

input           [1:0]   key;
input                   clk,EN;

output          [6:0]   hex0,hex1,hex2,hex3,hex4,hex5;

wire EN_5M, EN_100MS, EN_10MS, EN_1S, EN_1MIN, EN_10MIN, EN_10S;
wire    [18:0]  O_5M;
wire    [3:0]           O_100MS, O_10MS,O_1S, O_1MIN;
wire    [2:0]           O_10S, O_10MIN;
wire                            start,reset,start_key;

assign reset = EN? key[0] : 1'b1;
assign start_key = EN? key[1]  : 1'b1;


T_FF
Start_0(
        .iRSTn(reset),
        .iCLK(start_key),
        .iDATA(1'b1),
        .oDATA(start)
);
```

```
COUNTER_LAB // 0.001초 단위 카운터
#(.n(19),.k(500000))
COUNTER_5M(
          .iCLK(clk),
          .iEN(start),
          .iRSTn(reset),
          .oEN(EN_5M),
          .oCNT(O_5M)
);

COUNTER_LAB// 0.01초 단위 카운터
#(.n(4),.k(10))
COUNTER_10MS(
          .iCLK(clk),
          .iEN(EN_5M),
          .iRSTn(reset),
          .oEN(EN_10MS),
          .oCNT(O_10MS)
);

COUNTER_LAB// 0.1초 단위 카운터
#(.n(4),.k(10))
COUNTER_100MS(
          .iCLK(clk),
          .iEN(EN_10MS),
          .iRSTn(reset),
          .oEN(EN_100MS),
          .oCNT(O_100MS)
);

COUNTER_LAB// 1초 단위 카운터
#(.n(4),.k(10))
COUNTER_1S(
          .iCLK(clk),
          .iEN(EN_100MS),
          .iRSTn(reset),
          .oEN(EN_1S),
          .oCNT(O_1S)
);

COUNTER_LAB// 10초 단위 카운터
#(.n(3),.k(6))
COUNTER_10S(
          .iCLK(clk),
          .iEN(EN_1S),
          .iRSTn(reset),
          .oEN(EN_10S),
          .oCNT(O_10S)
);

COUNTER_LAB 1분 단위 카운터
```

```
#(.n(4),.k(10))
COUNTER_1M(
        .iCLK(clk),
        .iEN(EN_10S),
        .iRSTn(reset),
        .oEN(EN_1MIN),
        .oCNT(O_1MIN)
);

COUNTER_LAB 10분 단위 카운터
#(.n(3),.k(6))
COUNTER_10M(
        .iCLK(clk),
        .iEN(EN_1MIN),
        .iRSTn(reset),
        .oEN(EN_10MIN),
        .oCNT(O_10MIN)
);

Segment_Decoder
Hex0(
        .iDATA(O_10MS),
        .oDATA(hex0)
);

Segment_Decoder
Hex1(
        .iDATA(O_100MS),
        .oDATA(hex1)
);

Segment_Decoder
Hex2(
        .iDATA(O_1S),
        .oDATA(hex2)
);

Segment_Decoder
Hex3(
        .iDATA(O_10S),
        .oDATA(hex3)
);

Segment_Decoder
Hex4(
        .iDATA(O_1MIN),
        .oDATA(hex4)
);

Segment_Decoder
Hex5(
```

```
        .iDATA(O_10MIN),
        .oDATA(hex5)
);


endmodule
```

디지털시스템설계 <텀프로젝트 결과 보고서>

<Timer>

I/O spec

I/O	Signal Name	Bit	Description
I	key	3	key[2]:Start,Stop버튼역할 key[1]:타이머 값 설정 key[0]:리셋 각각은 GPIO0[2:0]에 할당하여 블루투스 의 보내는신호에의하여 동작
I	EN	1	각상태에 주어지는 EN값
I	clk	1	타이머에 사용되는 clk
O	Oalarm	1	타이머 값이 00:00:00이 되었을 때 1을출력->gpio_1_tmp[0]전달->D_R EG통과후 GPIO1[0]에 전달하여 어플에 소리 출력
O	HEX5	7	시 단위(5비트)
O	HEX4	7	10분 단위(3비트)
O	HEX3	7	1분 단위(4비트)
O	HEX2	7	10초(3비트)
O	HEX1	7	1초 단위(4비트)
O	HEX0	7	0.1초 단위(4비트)

<timer>

```
module Timer(
        Oalarm,
        key,
        sw,
        EN,
        clk,
        led,
        hex0,
        hex1,
        hex2,
        hex3,
        hex4,
        hex5
);

input   [2:0]   key; // reset, preset 등 clock 통제용 key
input   [9:0]   sw; //preset data 입력
input           clk,EN;

output  [6:0]   hex0,hex1,hex2,hex3,hex4,hex5;
output  [9:0]   led;
output          Oalarm; //0까지 카운트 되면 1출력

wire EN_5M, EN_100MS, EN_10MS, EN_1S, EN_1MIN, EN_10MIN, EN_10S;
wire Z_100MS, Z_10MS, Z_1S, Z_1MIN, Z_10MIN, Z_10S;
wire    [18:0]  O_5M;
```

```
wire      [3:0]     O_100MS, O_10MS,O_1S, O_1MIN;
wire      [2:0]     O_10S, O_10MIN; //카운터 출력값
wire                preset1,preset2,preset3,preset4;
wire                start,reset,preset,start_key;
wire      [2:0]     preset_data;
wire      [3:0]     preset_data9;

assign preset2 = (sw[9:8] == 2'b01) ? preset : 1'b1;
assign preset3 = (sw[9:8] == 2'b10) ? preset : 1'b1;
assign preset4 = (sw[9:8] == 2'b11) ? preset : 1'b1;
assign preset1 = (sw[9:8] == 2'b00) ? preset : 1'b1;  //sw[9:8] 로 시,분,초 중 preset 할 항목을 select


assign reset = EN? key[0] : 1'b1;
assign preset = EN? key[1] : 1'b1;
assign start_key = EN? key[2] : 1'b1;
assign preset_data = (sw[9:8] == 2'b01 || sw[9:8] == 2'b11) ? ((sw[2:0] > 3'd5 )? 3'd5 : sw[2:0] ) : sw[2:0];
//0~5범위 preset data: 10분단위, 10초단위
assign preset_data9 = (sw[9:8] == 2'b00 || sw[9:8] == 2'b10) ? ((sw[3:0] > 4'd9 )? 3'd9 : sw[3:0] ) : sw[3:0];
//0~9범위 preset data: 1분단위, 1초단위
reg      Oalarm;

T_FF
Start(
          .iRSTn(reset),
          .iCLK(start_key),
          .iDATA(1'b1),
          .oDATA(start)
);

COUNTER_LAB
#(.n(19),.k(500000))
COUNTER_5M(
          .iCLK(clk),
          .iEN(start),
          .iRSTn(reset),
          .oEN(EN_5M),
          .oCNT(O_5M)
);

RCOUNTER_LAB// 역카운터
#(.n(4),.k(10))
COUNTER_10MS(
          .iCLK(clk),
          .iEN(EN_5M),
          .iRSTn(reset),
          .iZERO(Z_100MS),
          .oZERO(Z_10MS),
          .oEN(EN_10MS),
          .oCNT(O_10MS)
);
```

```
RCOUNTER_LAB
#(.n(4),.k(10))
COUNTER_100MS(
          .iCLK(clk),
          .iEN(EN_10MS),
          .iRSTn(reset),
          .iZERO(Z_1S),
          .oZERO(Z_100MS),
          .oEN(EN_100MS),
          .oCNT(O_100MS)
);

RCOUNTER_PRESET
#(.n(4),.k(10))
COUNTER_1S(
          .iCLK(clk),
          .iEN(EN_100MS),
          .iRSTn(reset),
          .iPRSTn(preset1),
          .iDATA_Preset(preset_data9),
          .iZERO(Z_10S),
          .oZERO(Z_1S),
          .oEN(EN_1S),
          .oCNT(O_1S)
);

RCOUNTER_PRESET
#(.n(3),.k(6))
COUNTER_10S(
          .iCLK(clk),
          .iEN(EN_1S),
          .iRSTn(reset),
          .iPRSTn(preset2),
          .iDATA_Preset(preset_data),
          .iZERO(Z_1MIN),
          .oZERO(Z_10S),
          .oEN(EN_10S),
          .oCNT(O_10S)
);

RCOUNTER_PRESET
#(.n(4),.k(10))
COUNTER_1M(
          .iCLK(clk),
          .iEN(EN_10S),
          .iRSTn(reset),
          .iPRSTn(preset3),
          .iDATA_Preset(preset_data9),
          .iZERO(Z_10MIN),
          .oZERO(Z_1MIN),
```

```
            .oEN(EN_1MIN),
            .oCNT(O_1MIN)
);


always@(negedge O_10MS[0])
begin
if({O_10MIN,O_1MIN,O_10S,O_1S,O_100MS,O_10MS}=={21'b0,1'b1})
begin
Oalarm<=1'b1;
end
else begin
Oalarm<=1'b0;
end
end


RCOUNTER_PRESET
#(.n(3),.k(6))
COUNTER_10M(
            .iCLK(clk),
            .iEN(EN_1MIN),
            .iRSTn(reset),
            .iPRSTn(preset4),
            .iDATA_Preset(preset_data),
            .iZERO(1'b1),
            .oZERO(Z_10MIN),
            .oEN(EN_10MIN),
            .oCNT(O_10MIN)
);

Segment_Decoder
Hex0(
            .iDATA(O_10MS),
            .oDATA(hex0)
);

Segment_Decoder
Hex1(
            .iDATA(O_100MS),
            .oDATA(hex1)
);

Segment_Decoder
Hex2(
            .iDATA(O_1S),
            .oDATA(hex2)
);

Segment_Decoder
Hex3(
```

```
            .iDATA(O_10S),
            .oDATA(hex3)
);

Segment_Decoder
Hex4(
            .iDATA(O_1MIN),
            .oDATA(hex4)
);

Segment_Decoder
Hex5(
            .iDATA(O_10MIN),
            .oDATA(hex5)
);

endmodule
```

디지털시스템설계 <텀프로젝트 결과 보고서>

<Controller>

I/O	Signal Name	Bit	Description
I	key	1	각상태를 변경해주는 역할을 하는 것으로 posedge 가 될 때마다 현재state에서 다음 state로 바뀌게 되고 각각의 state들은 state machine을 통해 현재 ,다음 상태들을 할당시켜줌
O	state	3	3비트로 할당된 각 상태들을 변수 지정.

<controller code>

```
module Controller(
        key,
        state
);

input           key;

output  [3:0]   state;

reg             [3:0]   current_state, next_state;

parameter       [3:0]   A = 3'b000 , B = 3'b001, C =3'b010; //A: clock mode B: timer mode C: stop_watch mode

always @(posedge key)
begin
        current_state <= next_state; // key를 누르면 저장되어있는 다음 상태가 현재상태로 대입된다
end

always@* // 현재상태가 A일때 key 누르면 B가, 현재상태가 B일때 key 누르면 C가, 현재상태가 C일때 key 누르면 A가 다음 상태에 저장
begin
        case(current_state)
                A : next_state = B;
                B : next_state = C;
                C : next_state = A;
                default : next_state = A;
        endcase
end

assign state = current_state;

endmodule
```

디지털시스템설계 <텀프로젝트 결과 보고서>

✔ 합성 / 분석

● 관련 기술

디지털 시계

-Lab 4 part2 에서 배운 카운터 프리셋 함수를 주로 사용하여 설계하였다.

-클락 0.05MHz를 세는 함수로 1/10초를 발생시키고 1/10초가 0에서 9까지 카운트 되었을 때 rollover 값을 출력 값으로 받아 그 값을 1초 단위의 카운트를 수행할 함수에 enable로 넣어준다. 같은 방법으로 시 부분까지 차례대로 enable값을 넣어주며 카운트해 올라온다. 이 때 시간 카운터에서 0시부터 23시까지 세는데, Segment Decoder를 수정하여 1부터 12까지는 그대로 hexadecimal 값으로 출력하였고 13시부터 23시까지는 각 시간 값에서 12를 뺀 수를 출력하는 동시에 AM, PM을 LEDR[9] on/off 여부로 구분할 수 있도록 했다.

-시계를 원하는 시간으로 프리셋 할 때에는 SW[9:8]을 이용해 시, 분의 10의 자리, 분의 1의 자리 일 경우를 11, 01, 10로 입력하여 프리셋 값을 넣을 부분을 설정해준 후 스위치로 값을 정하고 어플의 프리셋키를 누르면 해당 자리가 설정값으로 프리셋 된다.

Stop Watch

핸드폰의 실제 스탑 워치를 참고하여 $\frac{1}{100}$초 단위에서 분 단위까지 측정할 수 있도록 설계하였다. 디지털 클럭에서 어플의 모드변경 버튼을 누르면 변경 후 분, 초, $\frac{1}{100}$초 모두 00으로 reset 되어 있는 상태에서 어플의 start 버튼을 누르면 start 하고 시간이 경과 후 멈출수도 있다. 그리고 리셋 버튼으로 리셋도 가능하다.

- Timer

스톱워치에서 어플의 모드변경 버튼을 누르면 타이머 모드로 변경된다. 초기 값은 모두 0이며, SW[9:8]을 조정함으로써 프리셋 위치를 설정하고 설정값을 SW로 정하여 프리셋 하고 스타트 버튼을 누르면 그값이 0이 될 때 까지 카운트 하고 카운트 된후에 00:00:00이 되면 어플에 알람 소리가 출력되게된다.처음 시간을 설정할 때, D Register를 이용하여 입력 받은 값을 저장한 후, 그 output값을 RCOUNTER_LAB,RCOUNTER_PRSET 두 가지로 타이머의 시간이 다운 카운트되게 하고 설정한 시간이 모두 지나면 oALARM 신호가 1이 되고 그것을 gpio_1_tmp로 전달을 하고 그것을 DREG을 통해서 GPIO1에 출력을 한다. 이것은 값의 전송을 클럭에 맞춰 더욱 정확하게 작동하기 위함이다.

디지털시스템설계 <텀프로젝트 결과 보고서>

✔ 일정 및 역할 분담

● 일정

1주차(11/20~27) : 기본기능 구현 완료, 아이디어 회의, 설계계획서 작성

2주차(11/28~12/5) : 심장박동센서 작동 확인, 앱인밴터를 통한 블루투스 모듈 이용 익히기, 블루투스 묘듈 을 이용하여 보드 제어 구현 완료, 알람기능 추가, 동영상 촬영.

3주차(12/6~12/9) : 마지막 작동 확인, 결과 보고서 작성.

● 역할 분담

이름	역할
박상준	시계 기본기능 설계, 아두이노 코드 수정 및 어플제작, 동영상 편집
양해찬	Top block설계, 시계 기본기능 설계, 어플제작 및 수정
이정헌	시계 기본기능 설계, 레포트 작성, PPT 제작