HDL응용설계 프로젝트
결과보고서

과　　목: HDL응용설계
담당교수: ○○○ 교수님
제출일자: 2019.06.10
소　　속: 공과대학 전자공학과
학　　번: 2016○○○○○
이　　름: 장혜정

가. 설계목표

Clock generator 모듈과 알람, 스톱워치 기능이 있는 디지털시계 모듈을 설계하며 지금까지 배운 Verilog HDL을 활용하고 이해하는 것을 목표로 하였다.

프로젝트 스펙을 참고했을 때, 설계하고자 하는 모듈이 실제로 일상생활에서 사용되는 가정용 타이머인 것처럼 생각하였다. 가정용 타이머는 시계, 알람, 스톱워치 기능을 가지고 있으며 어떤 버튼을 누르느냐에 따라 각 기능을 선택하여 사용할 수 있다. 여기서는 각 변수를 가정용 타이머의 각 버튼이라고 생각하면서 모듈을 설계하기로 하였다.

나. 설계 내용

- 코드와 주석

1. Clock Generator 모듈

```
/*clock_gen.v*/
module  clock_gen(Clock_5K, Reset,
                  Clock_1Sec, Clock_1MSec);

input   Clock_5K, Reset;
output reg      Clock_1MSec;
output reg      Clock_1Sec;
reg[11:0]       count;  //Clock_1Sec 신호를 생성하기 위한 추가변수

always @(posedge Clock_5K or negedge Reset) begin
        if(!Reset) begin
                /*Clock_1MSec, Clock_1Sec, count를 모두 0으로 초기화*/
                Clock_1MSec <= 0;
                Clock_1Sec <= 0;
                count <= 0;
        end
        else begin
                count <= count + 1;     //count를 1 증가
                Clock_1MSec <= ~Clock_1MSec;    //Clock_1MSec를 반전
                if(count == 2500) begin
                        /*매 0.5초마다 Clock_1Sec를 발생시킨다(2.5KHz)*/
                        Clock_1Sec <= ~Clock_1Sec;
                        count <= 0;
                end
                else    Clock_1Sec <= Clock_1Sec;
        end
end

endmodule
```

2. Stopwatch 모듈

```
/*stop.v*/
module  stop(
                Clock_1MSec, Reset, Start_S, Stop_S, Reset_S,
                Hours_S, Mins_S, Secs_S, MSecs_S, Control);

input           Clock_1MSec, Reset, Start_S, Stop_S, Reset_S, Control;
output reg[3:0] Hours_S;
output reg[5:0] Mins_S, Secs_S;
output reg[9:0] MSecs_S;
reg             stop_count;     //정지(stop=1) 횟수를 셀 수 있도록 설정한 추가 변수

always @(posedge Clock_1MSec or negedge Reset) begin
        if(!Control) begin      //스톱워치 기능으로 설정되어 있을 때
                if(!Reset) begin
                        /*모두 0으로 초기화*/
                        Hours_S <= 0;
                        Mins_S <= 0;
                        Secs_S <= 0;
                        MSecs_S <= 0;
                        stop_count = 0;
                end
                else begin
                        /*평상시(스톱워치 카운트를 시작하지 않았을 때)에는 MSecs_S의 값이 변하지 않는다.*/
                        MSecs_S <= MSecs_S;
                        if(Stop_S) begin
                                /*
                                Stop_S=1이면 stop_count도 1로 바꿔준다.
                                또한 아무 동작도 하지 않고, 정지 상태를 유지한다.
                                */
                                stop_count = 1;
                        end
                        else if((Start_S==1) && (stop_count!=1)) begin
                                /*정지 상태가 아니고, Start_S=1이면 스톱워치 카운트를 시작한다.*/
                                MSecs_S <= MSecs_S + 1;
                                if(MSecs_S == 999) begin
                                        Secs_S <= Secs_S + 1;
                                        if(Secs_S == 59) begin
                                                Mins_S <= Mins_S + 1;
                                                if(Mins_S == 59) begin
                                                        Hours_S <= Hours_S + 1;
                                                        if(Hours_S == 11)       Hours_S <= 0;
                                                        Mins_S <= 0;
                                                end
                                                Secs_S <= 0;
                                        end
                                        MSecs_S <= 0;
                                end
                        end

                        else if((Reset_S==1) && (stop_count==1)) begin
                                /*Reset_S가 1이 되면 스톱워치의 동작이 모두 초기화된다.*/
                                Hours_S <= 0;
                                Mins_S <= 0;
                                Secs_S <= 0;
                                MSecs_S <= 0;
                                stop_count = 0;
                        end
                end
        end
end

endmodule
```

3. Alarm 모듈

```
/*alarm_clk.v*/
module  alarm_clk(
                Clock_1sec, Reset, LoadTime, LoadAlm, AlarmEnable,
                Set_AM_PM, Alarm_AM_PM_In,
                SetSecs, SetMins, AlarmMinsIn, SetHours, AlarmHoursIn,
                AM_PM, Alarm, Secs_C, Mins_C, Hours_C);

input           Clock_1sec, Reset, LoadTime, LoadAlm, Set_AM_PM, Alarm_AM_PM_In, AlarmEnable
input[5:0]      SetSecs, SetMins, AlarmMinsIn;
input[3:0]      SetHours, AlarmHoursIn;
output reg      AM_PM, Alarm;
output reg[5:0] Secs_C, Mins_C;
output reg[3:0] Hours_C;

always @(posedge Clock_1sec or negedge Reset) begin
        if(!Reset) begin
        /*초기화*/
                AM_PM <= 1;
                Alarm <= 0;
                Secs_C <= 0;
                Mins_C <= 0;
                Hours_C <= 0;
        end
        else begin
                /*평상시 상태: 시간 유지, 알람 동작 유지*/
                Secs_C <= Secs_C;
                Alarm <= Alarm;

                if(LoadTime) begin      //시간 setting
                        AM_PM <= Set_AM_PM;
                        Secs_C <= SetSecs;
                        Mins_C <= SetMins;
                        Hours_C <= SetHours;
                end


                else if(!LoadTime) begin        //시계 동작
                        Secs_C <= Secs_C + 1;
                        if(Secs_C == 59) begin
                                Mins_C <= Mins_C + 1;   //59초에서 1분으로
                                if(Mins_C == 59) begin
                                        Hours_C <= Hours_C + 1; //59분에서 1시간으로
                                        /*59초이고 59분인 상태에서 11시일 때(11:59:59)*/
                                        if(Hours_C==11) AM_PM <= ~AM_PM;         //오전/오후 바꾸기
                                        if(Hours_C==13) Hours_C <= 1;   //12시가 넘어가면 다시 1시부터
                                        Mins_C <= 0;    //다시 0분부터
                                end
                                Secs_C <= 0;    //다시 0초부터
                        end
                end

                /*알람이 울릴 시각은 테스트벤치에서 setting한다.*/

                if(AlarmEnable) begin   //알람 동작
                        if(!LoadAlm) begin
                                if((AM_PM==Alarm_AM_PM_In)&&(Hours_C==AlarmHoursIn)) begin
                                        /*알람 시각이 된 0초 시점부터 알람이 울린다. (1)*/
                                        if((Mins_C==(AlarmMinsIn-1))&&(Secs_C==59)) begin
                                                Alarm <= 1;
                                        end
                                        /*알람 시각이 되고 59초 시점까지만 알람이 울린다.(1->0)*/
                                        if((Mins_C==AlarmMinsIn)&&(Secs_C==59)) begin
                                                Alarm <= 0;
                                        end
                                end
                        end
                end
        end
end

endmodule
```

4. Top 모듈

Top 모듈 설계에는 앞서 설계한 Clock Generator 모듈, Stopwatch 모듈, Alarm 모듈의 instantiation이 필요하므로, 입출력 변수를 우선 다음과 같이 정리해보았다.

alarm_clk		stop		TOP	
output	input	output	input	output	input
AM_PM	Clock_1Sec		Clock_1MSec	AM_PM	Clock_5K
Alarm	Reset		Reset	Alarm	Reset
Hours_C	LoadTime	Hours_S	Start_S	Hours	LoadTime
Mins_C	LoadAlm	Mins_S	Stop_S	Mins	LoadAlm
Secs_C	Set_AM_PM	Secs_S	Reset_S	Secs	Set_AM_PM
	SetSecs	MSecs_S	Control	MSecs	SetSecs
	SetMins			SW_State	SetMins
	SetHours				SetHours
	Alarm_AM_PM_In				Alarm_AM_PM_In
	AlarmEnable				AlarmEnable
	AlarmMinsIn				AlarmMinsIn
	AlarmHoursIn				AlarmHoursIn
					Control

clock_gen: output - Clock_1MSec, Clock_1Sec / input: Clock_5K, Reset

```
/*top.v*/
module  TOP(Clock_5K, Reset,
            Control,
            Start_S, Stop_S, Reset_S,
            AM_PM, Hours, Mins, Secs, MSecs, Alarm, SW_State,
            LoadTime, LoadAlm, AlarmEnable, Set_AM_PM, Alarm_AM_PM_In,
            SetSecs, SetMins, AlarmMinsIn, SetHours, AlarmHoursIn);

input           Start_S, Stop_S, Reset_S;
input           Clock_5K, Reset, Control, LoadTime, LoadAlm, Set_AM_PM, Alarm_AM_PM_In, AlarmEnable;
input[5:0]      SetSecs, SetMins, AlarmMinsIn;
input[3:0]      SetHours, AlarmHoursIn;
output[3:0]     Hours;
output[5:0]     Mins, Secs;
output[9:0]     MSecs;
output          AM_PM, Alarm;
output reg      SW_State;
wire[3:0]       Hours_C, Hours_S;
wire[5:0]       Mins_C, Mins_S, Secs_C, Secs_S;
wire[9:0]       MSecs_S;

/*모듈 불러오기(instantiation)*/
clock_gen       clkgen(
                        Clock_5K, Reset,
                        Clock_1Sec, Clock_1MSec);
alarm_clk       alarm(
                        Clock_1Sec, Reset, LoadTime, LoadAlm, AlarmEnable,
                        Set_AM_PM, Alarm_AM_PM_In,
                        SetSecs, SetMins, AlarmMinsIn, SetHours, AlarmHoursIn,
                        AM_PM, Alarm, Secs_C, Mins_C, Hours_C);
stop            stopwatch(
                        Clock_1MSec, Reset, Start_S, Stop_S, Reset_S,
                        Hours_S, Mins_S, Secs_S, MSecs_S, Control);
```

```
/*Control 신호 1이면 시계, 0이면 스톱워치*/
assign Hours = (Control==1)?Hours_C:Hours_S;
assign Mins = (Control==1)?Mins_C:Mins_S;
assign Secs = (Control==1)?Secs_C:Secs_S;
assign MSecs = (Control==1)?0:MSecs_S;

always @(posedge Clock_5K or negedge Reset) begin
        if(!Reset)      SW_State <= 0;  //SW_State 초기화
        else if(!Control) begin //스톱워치 기능일 때
                if(Start_S)     SW_State <= 1;  //스톱워치 카운트 시작하면 SW_State에 1 저장
                if(Stop_S)      SW_State <= 0;  //스톱워치 카운트 끝나면 SW_State에 0 저장
        end
end

endmodule
```

다. 테스트벤치 및 시뮬레이션 결과

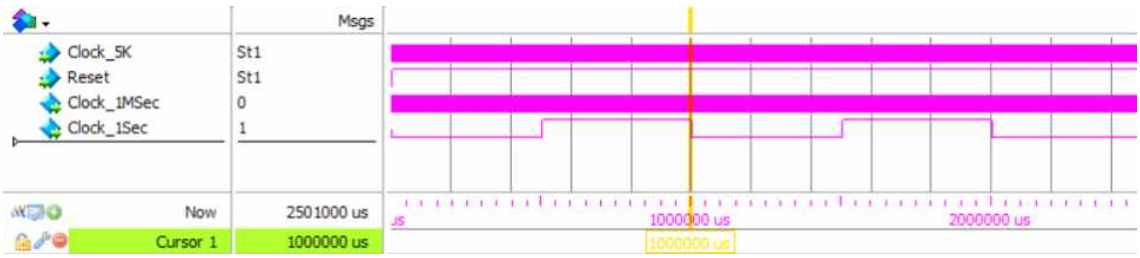
1. Clock Generator 모듈

Clock Generator의 테스트벤치 코드로는 프로젝트 스펙의 코드를 그대로 이용하였다. 테스트벤치 코드로 시뮬레이션 결과, Clock_1Sec가 1초주기를 가지도록 잘 동작하고 있음을 확인하였다.

- testbench 코드

```
`timescale 10us/1us

module clk_tb();
  reg Clock_5K, Reset;
  wire Clock_1Sec, Clock_1MSec;

  clock_gen clk_g(Clock_5K, Reset, Clock_1Sec, Clock_1MSec);

  initial
  begin
    Clock_5K = 1; Reset = 1;
    #50 Reset = 0;
    #50 Reset = 1;
    #250000 $finish;
  end

  always #10 Clock_5K = ~Clock_5K;

endmodule
```

- simulation 결과



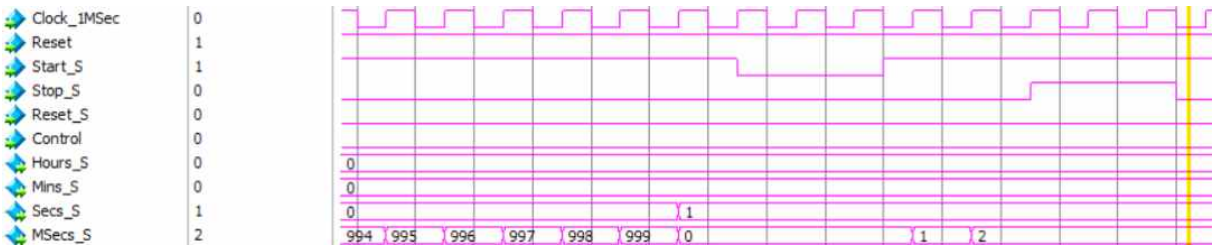
2. Stopwatch 모듈

다음은 스톱워치의 시뮬레이션이다. 테스트벤치 코드는 프로젝트 스펙의 스톱워치 시뮬레이션 결과를 참고하여 작성하였다.

- testbench 코드

```
/*tb_stop.v*/
`timescale 10us/1us

module stop_tb();
reg     Clock_1MSec, Reset, Start_S, Stop_S, Reset_S, Control;
wire[3:0]       Hours_S;
wire[5:0]       Mins_S, Secs_S;
wire[9:0]       MSecs_S;

stop    stop_m(Clock_1MSec, Reset, Start_S, Stop_S, Reset_S,
                Hours_S, Mins_S, Secs_S, MSecs_S, Control);

initial begin
                Clock_1MSec=0; Reset=1; Control=0; Start_S=0; Reset_S=0; Stop_S=0;
        #5      Reset=0;
        #5      Reset=1;
        #25     Start_S=1;
        #25     Start_S=0;
        #25     Start_S=1;
        #25     Stop_S=1;
        #25     Stop_S=0;
        #250000 $finish;
end

always begin
        #5      Clock_1MSec = ~Clock_1MSec;
end

endmodule
```

- simulation 결과



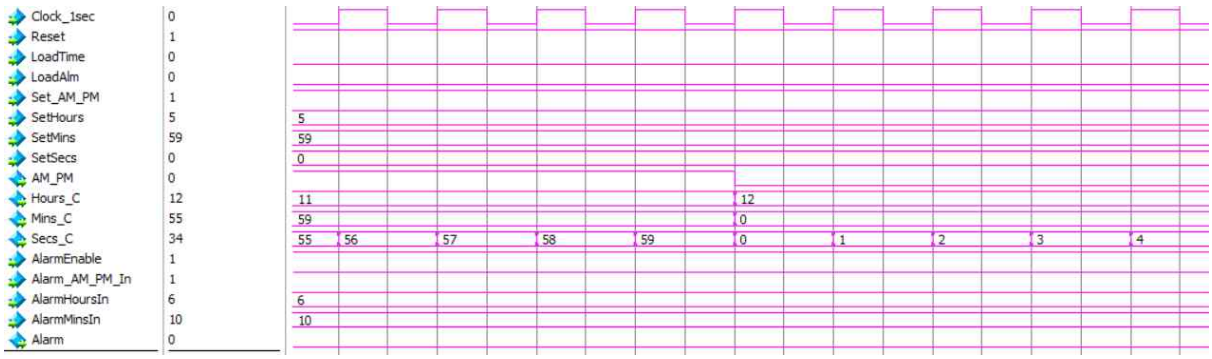
3. Alarm 모듈

알람 모듈의 테스트벤치 코드는 Clock_1sec 주기에 맞춰 직접 작성하였다. 시간은 오전 5시 59분 0초로 설정하였다. 알람이 잘 울리는지 확인하기 위해 알람이 울릴 시간은, 현재 시간과 큰 차이가 나지 않는 오전 6시 10분으로 설정하였다.

- testbench 코드

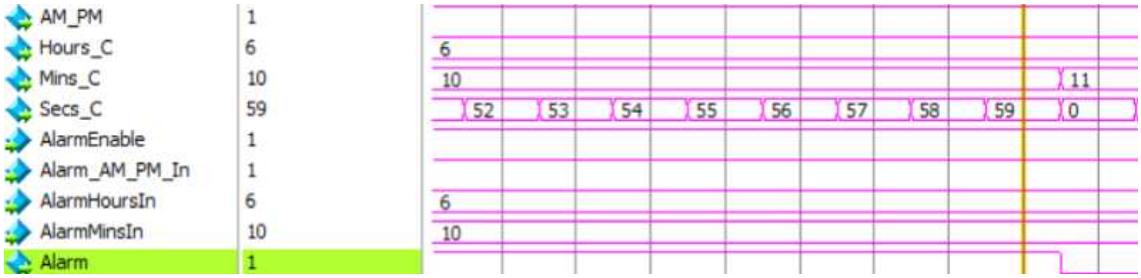
```
/*tb_alarm_clk.v*/
`timescale 10us/1us

module alarm_tb();

reg             Clock_1sec, Reset, LoadTime, LoadAlm, Set_AM_PM, Alarm_AM_PM_In, AlarmEnable;
reg[5:0]        SetSecs, SetMins, AlarmMinsIn;
reg[3:0]        SetHours, AlarmHoursIn;
wire            AM_PM, Alarm;
wire[5:0]       Secs_C, Mins_C;
wire[3:0]       Hours_C;

alarm_clk       alarm_m(Clock_1sec, Reset, LoadTime, LoadAlm, AlarmEnable,
                Set_AM_PM, Alarm_AM_PM_In,
                SetSecs, SetMins, AlarmMinsIn, SetHours, AlarmHoursIn,
                AM_PM, Alarm, Secs_C, Mins_C, Hours_C);

initial begin
                Clock_1sec=0; Reset=1; LoadTime=1; LoadAlm=0; AlarmEnable=1;
        #5      Reset=0;
        #5      Reset=1; Set_AM_PM=1; SetSecs=0; SetMins=59; SetHours=5;        //초기 시각 오전 5:59:00 로 설정
        #10     LoadTime=0;
        #5      LoadAlm=1;
        #5      Alarm_AM_PM_In=1; AlarmMinsIn=10; AlarmHoursIn=6; //알람 시각 오전 6:10 로 설정
        #15     LoadAlm=0;
        #250000 $finish;
end

always begin
        #5      Clock_1sec = ~Clock_1sec;
end

endmodule
```

- simulation 결과①: **오전** 11시 59분 59초 -> **오후** 12시 0분 0초



- simulation 결과②-1: 오전 6시 10분 0초부터 알람 시작(Alarm = 0->1)

- simulation 결과②-2: 오전 6시 10분 59초 끝에 알람 종료(Alarm = 1->0)



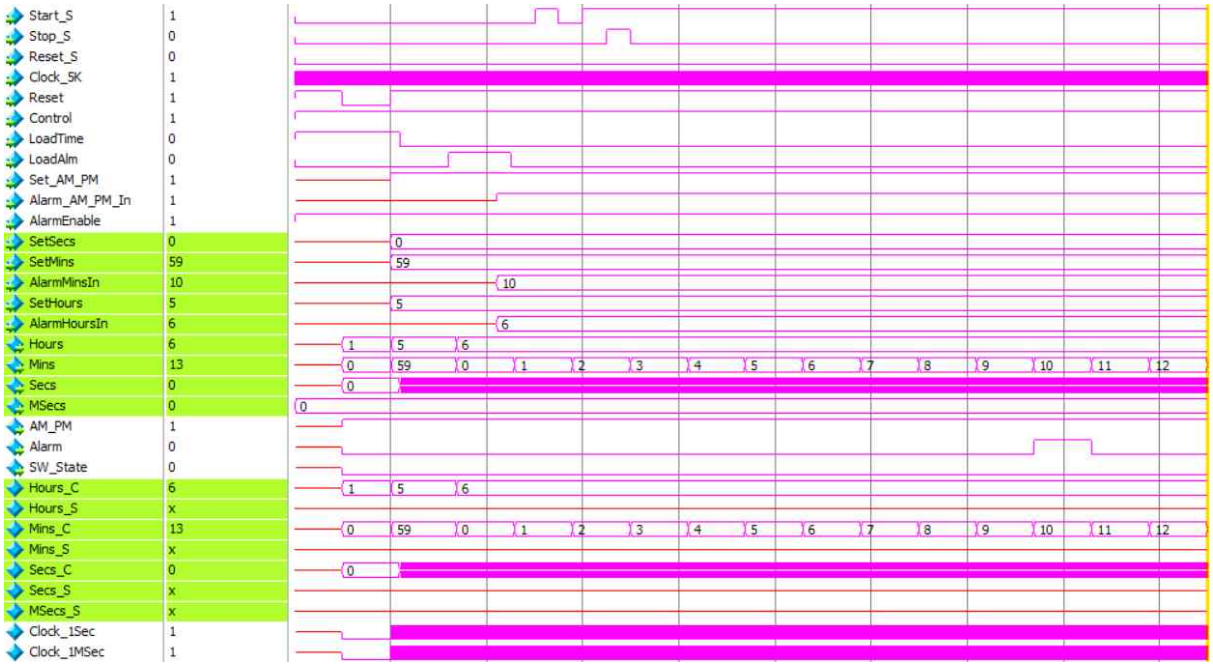
4. Top 모듈

앞서 따로 구현했던 모듈의 테스트벤치를 적절히 조합하였다.

- testbench 코드

```
/*tb_top.v*/
`timescale 10us/1us

module top_tb();

reg             Start_S, Stop_S, Reset_S;
reg             Clock_5K, Reset, Control, LoadTime, LoadAlm, Set_AM_PM, Alarm_AM_PM_In, AlarmEnable;
reg[5:0]        SetSecs, SetMins, AlarmMinsIn;
reg[3:0]        SetHours, AlarmHoursIn;
wire[3:0]       Hours;
wire[5:0]       Mins, Secs;
wire[9:0]       MSecs;
wire            AM_PM, Alarm, SW_State;

TOP             top_m(Clock_5K, Reset,
                Control,
                Start_S, Stop_S, Reset_S,
                AM_PM, Hours, Mins, Secs, MSecs, Alarm, SW_State,
                LoadTime, LoadAlm, AlarmEnable, Set_AM_PM, Alarm_AM_PM_In,
                SetSecs, SetMins, AlarmMinsIn, SetHours, AlarmHoursIn);

initial begin
                Clock_5K=1; Reset=1; Control=1; Start_S=0; Stop_S=0; Reset_S=0;
                LoadTime=1; LoadAlm=0; AlarmEnable=1;
        #5000000        Reset=0;
        #5000000        Reset=1; Set_AM_PM=1; SetSecs=0; SetMins=59; SetHours=5;
        #1000000        LoadTime=0;
        #5000000        LoadAlm=1;
        #5000000        Alarm_AM_PM_In=1; AlarmMinsIn=10; AlarmHoursIn=6;
        #1500000        LoadAlm=0;
        #2500000        Start_S=1;
        #2500000        Start_S=0;
        #2500000        Start_S=1;
        #2500000        Stop_S=1;
        #2500000        Stop_S=0;
        #60000000       $finish;
end

always begin
        #10     Clock_5K = ~Clock_5K;
end

endmodule
```

- simulation 결과



라. 설계 과정에서 겪은 문제와 해결방법

설계는 Clock Generator -> Stopwatch -> Alarm -> Top 모듈 순서로 하였다.

첫 번째로 Clock Generator 모듈을 설계할 때, Clock_1MSec 주기는 어렵지 않게 생성할 수 있었지만, Clock_1Sec 주기 생성에서 고민을 했다. Clock_1MSec 펄스만을 사용해서 Clock_1Sec 신호를 만들어내는 것이 어렵다고 생각하여, count라는 변수를 추가하였다. count는 Clock_1MSec 펄스가 반복되는 횟수에 따라 그 값을 하나씩 늘리는 것으로 생각하여, 바뀌는 값을 계속 저장할 수 있도록 레지스터 reg로 선언하였고, 2.5KHz로 하기 위해 2500까지 저장할 수 있도록 12비트(2^11=2048, 2^12=4096)로 설정하였다. 처음에 Clock_1MSec 펄스만을 사용할 때는 Clock_1Sec의 주기를 정하는데 한계가 있었지만, 이렇게 추가 변수를 주는 방식으로 그 문제를 해결할 수 있었다.

두 번째로 Stopwatch 모듈을 설계할 때는 Stop_S 조건을 구현하는 부분에서 시행착오를 겪었다. 처음에 단순히 Stop_S=1인 경우 시간 변수 Hours_S, Mins_S, Secs_S, MSecs_S가 각각 이전 값을 계속 유지하도록 하는 조건문을 작성하였을 때 스톱워치를 시작(Start_S=1)하고 나서 일시정지(Start_S=0)하는 동작은 잘 실행되었지만, Stop_S가 1에서 0으로 바뀐 뒤에도 프로젝트 스펙에서 주어진 조건대로 정지 상태를 유지하지는 않았다. 이 문제를 해결하기 위해 Clock Generator 모듈을 설계할 때 count라는 추가 변수를 선언했던 것처럼, 이 모듈에서도 추가 변수(stop_count)를 선언하고 조건을 추가해보았다. stop_count는 스톱워치에서 정지 버튼(Stop_S)을 한 번도 누르지 않았을 때는 항상 0이라는 값을 유지하다가, 사용자가 정지 버튼을 누른 순간(Stop_S<=1) 1이 되고, 정지되어 있는 상태에서 사용자가 리셋 버튼을 눌렀을 때(Reset_S<=1) 다시 0이 되는 컨셉으로 생각하였다. 0 또는 1의 값만 가지므로 1비트로 설정하였고, 조건에 따라 값이 바뀌기 때문에 reg로 선언하였다. 조건문은 앞서 서술한 컨셉에 맞추어 작성하였고, 이렇게 수정한

Verilog 코드로 다시 시뮬레이션한 결과, 정지 상태 유지 동작도 잘 실행됨을 확인하였다.

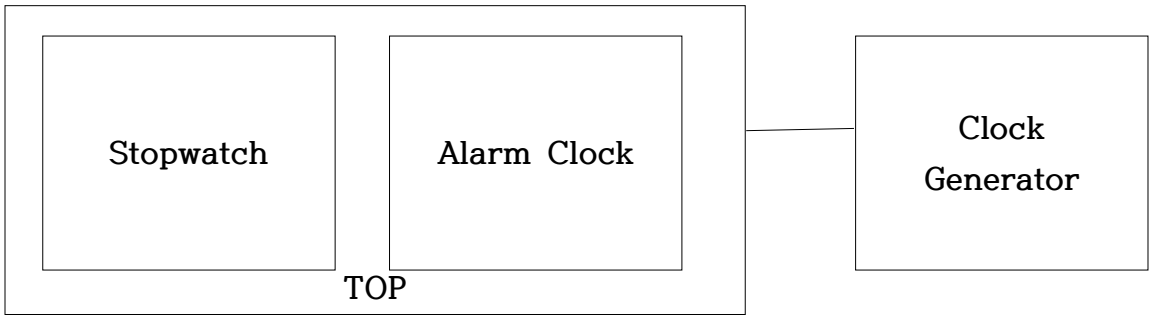
세 번째 Alarm 모듈에서는 매 시간마다 59분을 건너뛰는 문제와 오전/오후가 바뀌는 동작이 제대로 안 되는 문제가 발생하였다. 처음에 시계가 동작하는 코드를 다음과 같이 작성했는데,

```
if(!LoadTime) begin
        Secs_C <= Secs_C + 1;
        if(Secs_C==59) begin
                Mins_C <= Mins_C + 1;
                Secs_C <= 0;
        end
        if(Mins_C==59) begin
                Hours_C <= Hours_C + 1;
                Mins_C <= 0;
        end
        if(Hours_C>12) begin
                AM_PM <= ~AM_PM;
                Hours_C <= 1;
        end
end
```

여기서 (1) 59분이 되었을 때(Mins_C==59) 미처 초를 세기도 전에 Mins_C에 0을 대입해버리고, (2) 12시가 지나서야(Hours_C>12) 오전/오후가 바뀐다(AM_PM<=~AM_PM)는 점에서 잘못된 코드라는 것을 알게 되었다. 그래서 초(Secs_C), 분(Mins_C), 시(Hours_C)에 대한 조건을 독립적으로 하는 것이 아니라, 큰 단위의 변수에 대한 조건이 작은 단위의 변수에 대한 조건에 포함되도록 수정하여 문제를 해결했다.

마. 결론 및 고찰

시계의 전체 모듈에 대한 블록도를 그리면 다음과 같다.



블록도를 따라서 Clock Generator 모듈을 먼저 설계하고, Stopwatch와 Alarm 모듈을 구현한 뒤, Stopwatch와 Alarm 모듈이 합쳐진 Top 모듈을 설계하였다.

Clock Generator는 1초 주기의 클락 신호를 주기 위해 존재하는 모듈이고, Top 모듈은 시계의 여러 기능(Stopwatch, Alarm)을 디스플레이하는 모듈이다. '설계목표'에서 서술한 것처럼, 버튼을 눌러 여러 가지 기능을 수행하는 실제 가정용 타이머와 동작이 같다. 이 동

작을 보이기 위해, 스톱워치와 알람 각각의 기능이 제대로 구현되는 것도 중요하지만 무엇보다 전체 모듈이 합쳐졌을 때 사용자의 선택(input 값)을 오류 없이 받아들이도록 synthesize하는 과정이 중요하다.

TOP 모듈이라는 베이스에 Stopwatch와 Alarm Clock을 합성하기 위해, TOP의 Verilog 코드에서는 각 모듈을 instance로 불러왔다. 여기서 TOP의 시간변수는 Stopwatch로 동작하느냐 Alarm Clock으로 동작하느냐에 따라 다른 값을 디스플레이할 수 있다. 어떤 기능으로 동작할지는 Control의 값에 따라 달라지는데, Control=1일 때는 알람(알람 기능은 시계 기능까지도 포함하고 있다.), Control=0일 때는 스톱워치로 동작한다. 이를 이용하여, 3항 조건 연산자(<조건식>?<참 조건>:<거짓 조건> 형태)를 써서 시간변수 Hours, Mins, Secs, MSecs, AM_PM에 어떤 값을 인가할지에 대한 코드를 작성했다. TOP의 output 변수 중에서 Alarm 같은 경우는, Alarm 모듈을 불러옴으로써 값을 받아오기 때문에 따로 조건을 더 지정할 필요가 없다. TOP에서 추가로 생각해야할 것은 SW_State인데, 나머지 출력 포트는 instatiation을 통해 불러온 다른 모듈과 연결되기 때문에 wire로 선언했지만 SW_State는 TOP 모듈의 독립적인 포트이기 때문에 reg로 선언하였다. 그리고 always문 내에서 SW_State가 스톱워치 동작의 시작과 끝을 알리도록 조건에 따라 다른 값을 인가하였다.

TOP 모듈에 대해 시뮬레이션을 한 결과, 알람 동작까지는 잘 실행하고 있지만 SW_State의 역할이 모호함을 확인하였으며 이는 아직 해결하지 못한 문제이다.

프로젝트를 수행하며 testbench 코드도 module 코드 못지않게 중요함을 절실히 느꼈는데, testbench에서 초기 값을 적절하게 인가하지 않으면 설계한 module 코드의 구현을 잘 확인할 수 없기 때문이다.