REPORT

보고서 작성 서약서

1. 나는 타학생의 보고서를 복사(Copy)하지 않았습니다.
2. 나는 타학생의 보고서를 인터넷에서 다운로드 하여 대체하지 않았습니다.
3. 나는 타인에게 보고서 제출 전에 보고서를 보여주지 않았습니다.
4. 보고서 제출 기한을 준수하였습니다.

나는 보고서 작성시 위법 행위를 하지 않고,
성.균.인으로서 나의 명예를 지킬 것을 약속합니다.

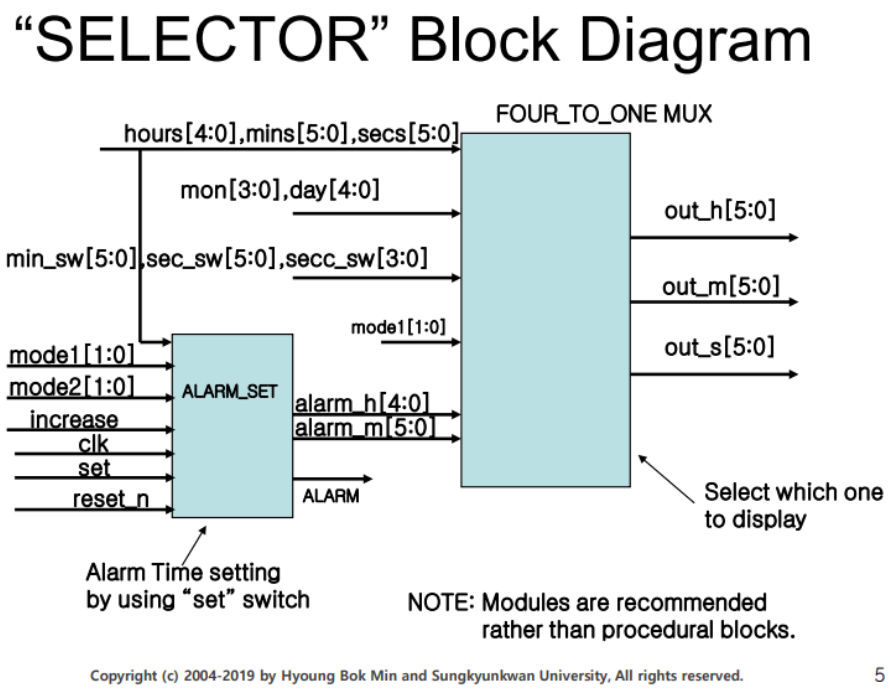
과　　목 : 디지털 시스템
과 제 명 : HW4 Digital clock
담당교수 : 민 형 복
학　　과 : 소프트웨어학과
학　　년 : 3
학　　번 : 2016312761
이　　름 : 여혁수
제 출 일 : 2020/11/30

1. Purpose

이번 실습에서 설계하는 것은 디지털 시계의 기능 중 하나인 알람 기능을 하는 알람 모듈과 시계의 모드에 따라 시계에 보여질 값을 선택하는 SELECTOR 모듈, 그리고 2진수로 표현되어 있는 시계에 보여질 값을 10진법으로 십의 자리 수와 일의 자리 수로 나누는 DIVIDER 모듈이다. 알람 모듈에는 알람이 울릴 시간을 조정하는 것과 알람을 끄고 알람 시간을 초기화하는 리셋 기능이 들어간다.

2. Problem statement

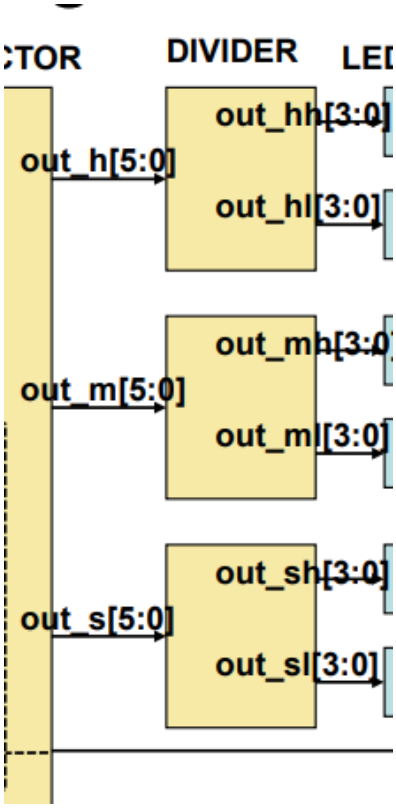
1) Describe what is the problem.



알람 모듈과 SELECTOR 모듈은 SELECTOR의 이름으로 같이 위와 같은 block diagram으로[1] 구현된다. 알람 모듈은 우선 알람 시간 조정하는 기능이 필요하다. 시간과 분을 설정할 수 있고, block diagram상의 increase라는 신호가 들어올 때마다 값을 1씩 증가시키며 조정한다. 알람이 울려야 할 때는 아웃풋 신호를 보내고, set 신호가 들어오면 알람을 멈춘다. set신호를 주기 전까지는 계속해서 알람이 울린다. reset신호가 들어오면 알람이 설정되었다는 것부터 모든 알람

관련 값을 초기화한다. reset신호가 들어오는 것 외에는 알람이 설정되고 나서는 그 알람이 지워지지 않으므로 알람 시간에서 24시간이 지나면 또 다시 알람이 울린다. 이를 위해 알람이 설정되었다는 것을 의미하는 변수가 필요하고, 그 변수는 현재 시계의 모드가 알람 모드이고 그 모드 중에 알람 시간을 조정하는 모드로 들어가면 0에서 1로 바뀐다. 알람을 설정한 시간에 알람이 울리고 set 신호로 알람을 멈추었는데 그 시간도 알람을 설정한 시간과 일치해서 다음 rising edge에서 알람이 또 울리면 안된다.

SELECTOR 모듈은 알람 모듈의 아웃풋인 설정된 알람 시간 값과 다른 시계 모드에서의 아웃풋 값들을 모두 받아서 현재 모드에 따라 실제로 시계에 보여질 값을 결정하여 아웃풋으로 보낸다. 아웃풋은 시간, 분, 초로 나누어 3개가 있는데 알람 모드와 같이 시간, 분만 보여지는 모드라면 초에 대한 아웃풋은 0으로 설정한다.



DIVIDER 모듈은 6비트의 값을 받아서 10진법에서 십의 자리 수와 일의 자리 수 2개로 나눈다. Full chip simulation에서 위 block diagram과[2] 같이 DIVIDER 3개를 붙여서 3개의 아웃풋 각각을 나눌 수가 있다.

2) Describe how do you solve the problem.

각 모듈들의 코드는 모두 verilog 언어로 작성했다. 알람 모듈은 알람 설정 시간을 조정할 때, 즉 increase가 들어올 때 현재 시계의 모드를 확인하고 시간을 증가시키거나, 분을 증가시켜야 한다는 신호를 계속해서 업데이트하는 것이 필요하다. Always block에 increase를 sensitivity list에 줌으로서 구현하였다. 별개로 알람이 울리는 것과 알람의 설정 시간을 만들어 내는 부분을 하나의 always block에 넣어서 구현하였다. 이 부분은 rising edge마다 실행되면서 synchronous하게 작동한다. 리셋이 들어오면 알람이 울리는 것을 의미하는 아웃풋과 설정된 시간을 나타내는 아웃풋들 모두 0으로 초기화한다. 알람 시간과 현재 시간을 비교하여 정확하게 일치한다면 알람 신호를 주고, set 신호가 들어왔다면 다시 알람 신호는 없어진다. 이 때, 알람이 설정된 시간에 알람이 울리고 set 신호로 알람을 멈추었는데 그때의 시간도 알람을 설정한 시간과 일치해서 다음 rising edge에서 알람이 또 울리는 것을 막아야 한다. set신호로 알람을 멈추었을 때의 시간을 변수에 저장하여 다음부터는 그 변수와 현재 시간을 비교하여 같으면 알람을 멈추었는데도 또 울리려 하는 것이므로 알람 신호를 주지 않는 것으로 구현했다. 알람 시간을 조정하는 신호가 들어오면 보통의 경우 시간 값에 1을 더해주고 시간을 나타내는 마지막 값, 시간이면 23, 분이면 59인데 여기서 1을 더해줘야 한다면 1을 더하는 대신 값에 0을 준다.

SELECTOR 모듈의 경우에는 역시 always block을 만들어 시간 관련 값이나 모드가 바뀔 때마다 작동하게 했다. 현재 시계의 모드를 확인하여 그에 맞는 시간 값을 아웃풋으로 선택했다.

DIVIDER 모듈은 6비트의 시간에 대한 값을 받아서 십의 자리와 일의 자리로 나누는 기능을 해야 한다. 받은 값은 시간, 분, 초, 1/10초 중에 하나와 관련된 값인데, 가장 크게 카운팅되는 것이 59까지 가는 초이기 때문에 받은 값의 최댓값이 59라고 할 수 있다. 그래서 값의 범위를 조건으로 잡고 50이상일 때, 40이상일 때... 이렇게 10의 단위로 나누어 조건문을 사용한다. 어떤 조건을 만족한다면 십의 자리는 나온 것이므로 십의 자리는 아웃풋으로 보내준다. 이 때 받은 값에서 알아낸 십의 자리 값을 빼면 일의 자리도 알게 된다. 코드를 완성하고, 각 모듈은 Modelsim 프로그램을[3] 통해 simulation 한다. 또한 Intel Quartus Prime 프로그램으로[4] synthesis과정을 마쳐 정상적으로 회로를 만들 수

있는 코드임을 증명하면서 문제 해결을 마친다.

3. Sources & Results

1) Analysis of HDL Source Codes of Design

selector.v

알람 모듈은 일단 알람 시간을 설정할 수 있어야 한다. 그래서 모드가 바뀌거나 시간 조정을 의미하는 신호가 바뀌면 작동하는 always block 을 하나 만들었다. 모드가 알람을 설정하는 모드인 것을 확인하고, 시간 조정 신호가 들어왔는지 확인한다. 신호가 확인되면 그것이 시간을 조정하는 모드인지, 분을 조정하는 모드인지에 따라 다음과 같이 작동한다.

```
M2_ALARM_HOUR : begin   // increment hours
        inc_hours = 1;
        inc_mins   = 0;
end
```

따로 만든 inc_hours, mins 라는 레지스터 변수에 시간이나 분을 증가시키라는 신호를 줌으로서 실제 시간을 업데이트 하는 것은 synchronous 하게 작동하는 always block 에서 이 별도의 변수를 확인하여 업데이트한다.

알람이 설정된 시간을 나타내는 아웃풋이 초기에 아무 변화가 없다면 모두 0을 나타낼 것이다. 이 때, 현재 시간이 0시 0분이라면, 알람을 설정하지 않았는데도 알람이 울릴 것이다. 따라서 알람을 설정했다는 의미를 가진 변수가 필요하다. alarm_on이라는 변수를 만들었다.

```
if (mode1 == M1_ALARM && (mode2 == M2_ALARM_HOUR || mode2 == M2_ALARM_MIN)) begin
        alarm_on <= 1'b1;
end
```

시계의 모드가 알람 시간을 조정하는 모드라면 1을 준다. 디자인 조건에 따라 리셋이 들어오지 않는 한 알람이 울린 시간에서 24시간이 지나도 똑같이 알람을

울려야 하므로 이 변수는 오직 리셋이 들어올 때만 0이 된다. 처음에는 이 코드가 모드가 바뀔 때 작동하는 always block에 있었는데 simulation에는 문제가 없었지만 synthesis에서 alarm_on이 2개의 always block에 사용되어 에러가 났다. 타이밍에 문제가 있기 때문인 것 같다.

```
if (hours == alarm_h && mins == alarm_m && alarm_on == 1'b1) begin
        // repeat-alarm in same time although we already snooze alarm
        if (alarm_snooze_h == alarm_h && alarm_snooze_m == alarm_m)
                reg_alarm <= 1'b0;   // protecting from repeat-alarm
        else
                reg_alarm <= 1'b1;
end
if (set == 1'b1) begin   // snooze the alarm
        reg_alarm <= 1'b0;
        alarm_snooze_h <= alarm_h;
        alarm_snooze_m <= alarm_m;
end
```

알람을 울리고 멈추는 코드이다. 일단 알람은 현재 시간과 알람의 시간이 일치하고, 알람이 설정되어 있어야 울린다. 그래서 if 조건문을 통해 조건을 위와 같이 써준다. 이 때, 디자인 조건에 따라 알람이 울려서 알람을 멈추었는데, 그 다음 클락에도 시간은 알람이 설정된 시간과 일치할 때, 알람은 울리면 안된다. 그래서 set 신호가 들어오면 알람을 멈추는데, 이 때의 시간을 alarm_snooze_h, m이라는 변수에 기록하여 다음 번에도 알람 시간과 현재 시간이 일치했을 때, 이 alarm_snooze 변수 또한 알람이 설정된 시간과 일치하는지를 확인하여 알람은 이미 멈추었음을 알게 해준다.

그리고 alarm_snooze 변수가 1분의 시간이 지나 현재 시간과 일치하지 않아지면 더 이상 의미를 가지면 안된다. 혹시나 24시간 동안 알람을 건드리지 않아 alarm_snooze와 같은 시간이 또 오게 된다면 alarm_snooze가 기대하지 않는 효과를 낼 수 있다.

```
if ((alarm_snooze_h != hours && alarm_snooze_h != 25) || (alarm_snooze_m !=
```

```
mins && alarm_snooze_m != 60)) begin
        alarm_snooze_h <= 25;
        alarm_snooze_m <= 60;
end
```

이렇게 alarm_snooze의 h, m에 각각 25와 60, 즉 시간 값으로는 나올 수 없는 값을 줘서 혹시 모를 영향을 없앤다.

앞서 가장 먼저 말한 알람 시간 조정에 대한 변수인 inc_hours, mins를 만들었는데, 이 변수 또한 synchronous하게 확인하며 실제 알람 시간 아웃풋을 업데이트한다. 각 변수의 값이 마지막 카운팅 값인 23, 59라면 +1이 되어 0이 되고, 그게 아니면 값에 +1만 해주면 되는 간단한 알고리즘 이므로 코드는 첨부하지 않았다.

SELECTOR 모듈은 시계의 모드가 바뀌거나 어떤 모드이건 그에 대한 시간 값들이 바뀔 때 작동하여 업데이트된 시계에 보여질 값들을 아웃풋으로 보내는 역할을 한다. 시계에 보여질 값들은 모드에 따라 달라지므로 모드 값을 if 조건문의 조건으로 포함시켜 코드를 작성하였다.

```
if (mode1 == M1_TIME) begin   // show present time
        reg_out_h[4:0] = hours;
        reg_out_h[5] = 1'b0;
        reg_out_m = mins;
        reg_out_s = secs;
```

만약 시계의 모드가 TIME 모드라면, 현재 시간을 보내야 하므로 hours, mins, secs가 각각 아웃풋이 된다.

divider.v

2진법으로 된 6비트의 시간 값을 십진법 상의 십의자리 수와 일의 자리 수로 나누는 기능을 해야한다. 시간에 대한 값이므로 인풋 값은 디자인 스펙 상 최대 59를 가질 수 있다. 그래서 if … else if… 를 붙여서 값이 50이상인가? 아니면 40

이상인가? 이렇게 계속해서 값의 범위를 나누었다. 이렇게 하면 여러 조건 중 어떤 하나에 만족했다면 그 값의 십의 자리를 알게 된다.

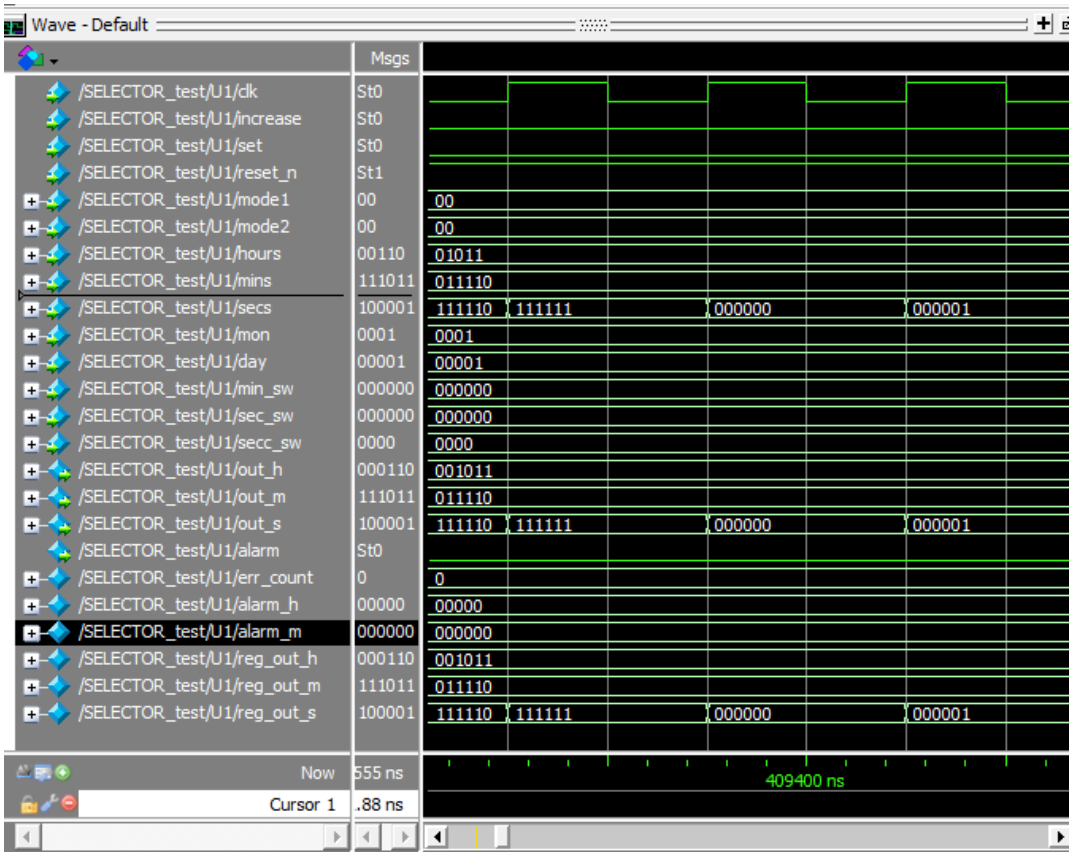
```
…
end else if (binary >= 6'b011110) begin
            reg_bcd_h = 4'b0011;
            reg_ones_digit = binary - 6'b011110;
            reg_bcd_l = reg_ones_digit[3:0];
…
```

연속된 조건문 중 하나를 따온 것이다. 6'b011110은 십진법으로 30이다. 이 조건문을 만족했다면 십의 자리는 3인 것이고, 인풋 값에서 30을 빼준다. 계산한 값은 일의 자리가 될 것이다. 처음에는 binary[3:0] 을 바로 그냥 주려고 했는데 에러가 나서 레지스터 변수를 따로 하나 잡았다.

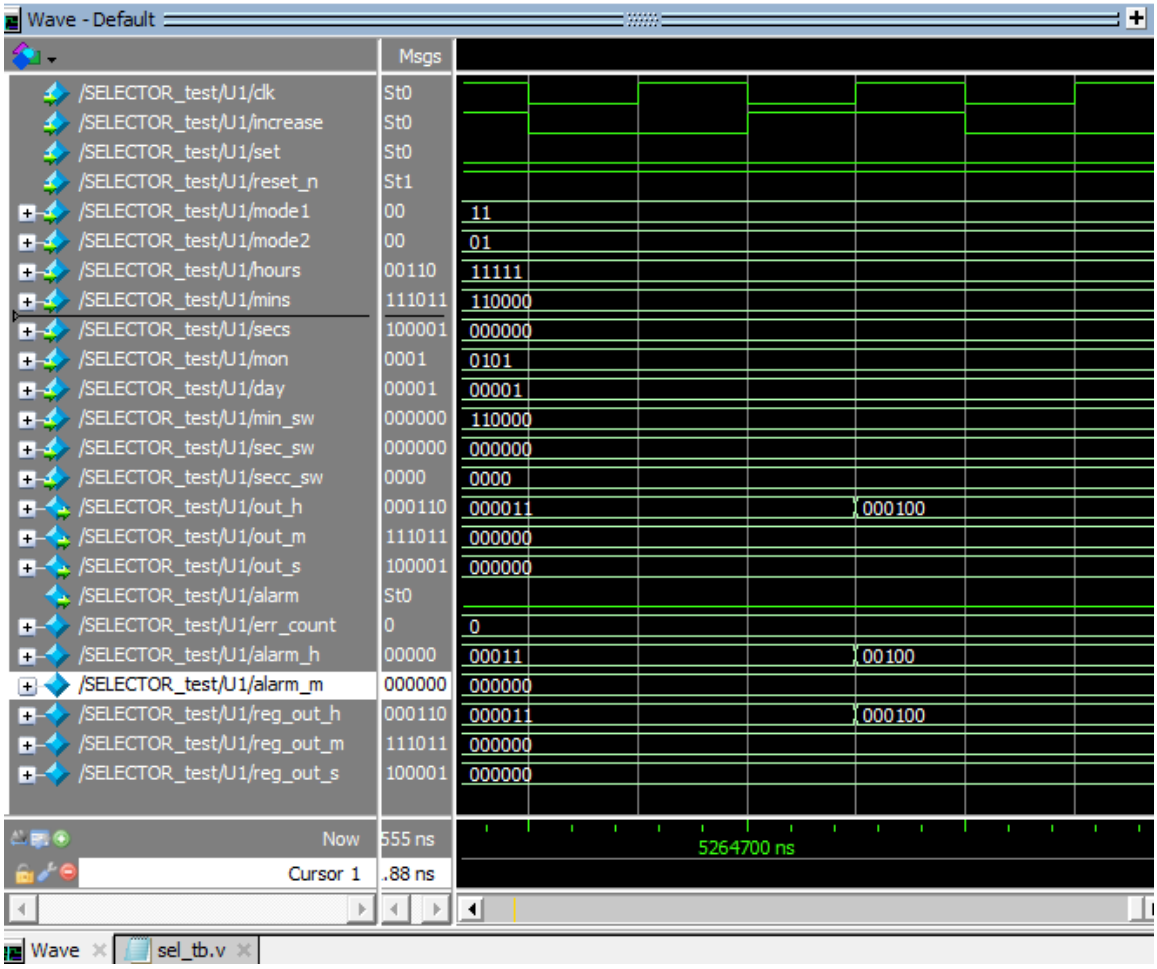
3) Simulation Waveforms

결과 파형을 바탕으로 설명 및 분석을 한다.

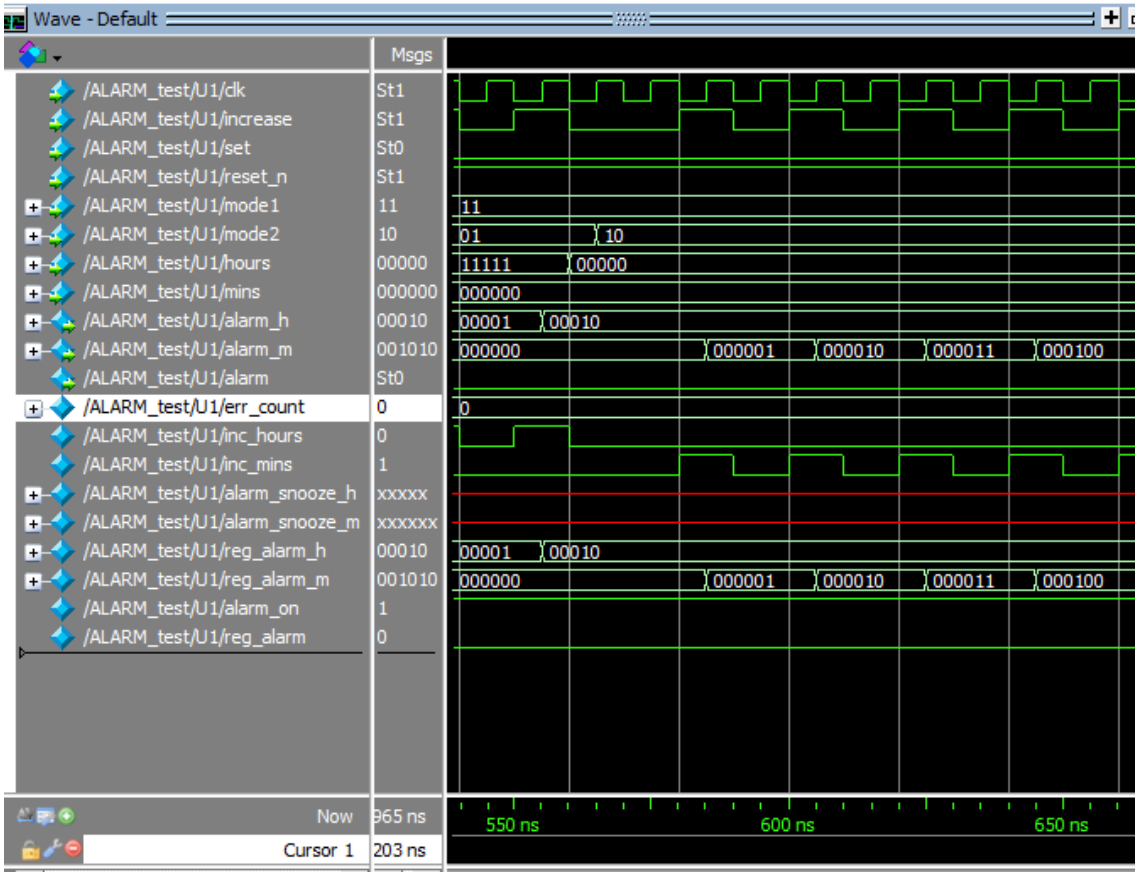
Selector.v



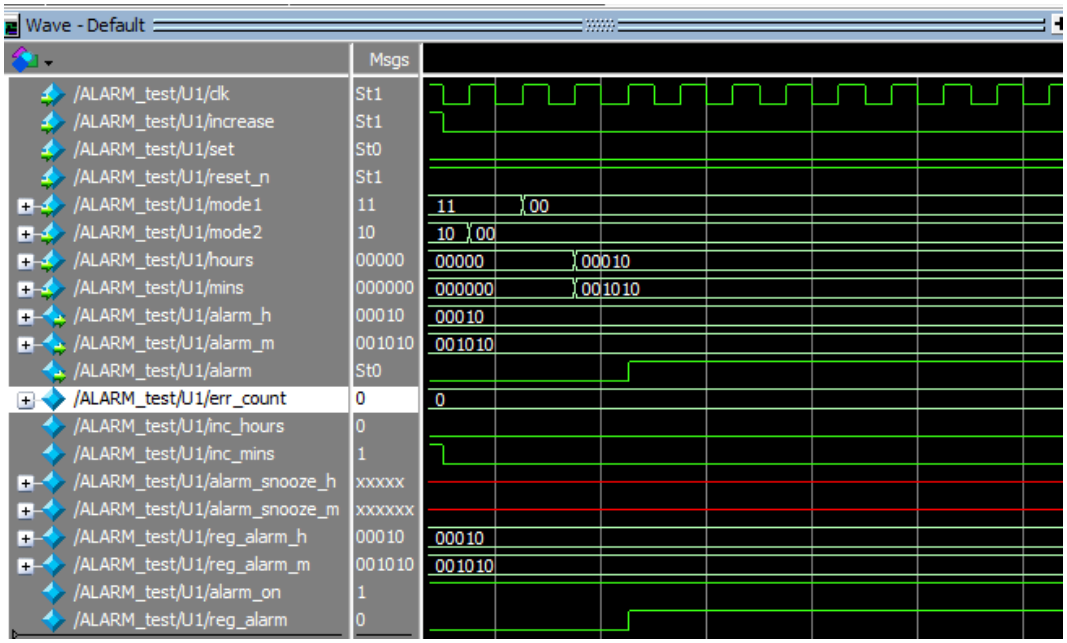
Selector.v의 시뮬레이션에서는 가장 우선으로 시계의 각 모드에 대해 시간이 흐르는 것을 가정하고 아웃풋이 제대로 나타나는지 관찰한다. 위 파형은 TIME 모드일 때이고, 시간이 흐름에 따라 TIME 모드의 시간 값인 hours, mins, secs가 모듈의 아웃풋으로 동시에 일치하게 바뀌는 것을 볼 수 있다. DATE 모드, TIMER 모드 역시 동일하게 잘 나타난다.



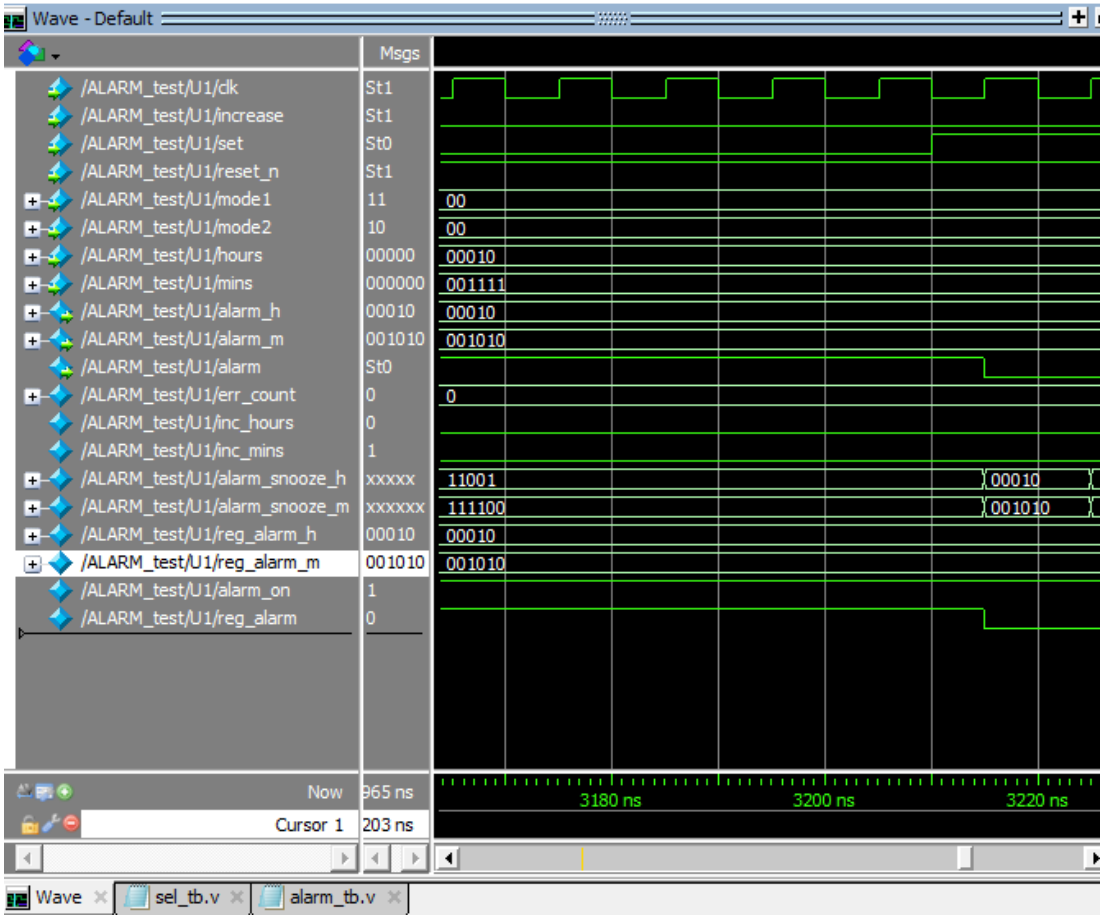
알람 모드도 다른 모드와 비슷한 방식으로 시뮬레이션한다. 알람 모드에서는 알람이 설정된 시간이 시계에 보여지는 아웃풋이 되기 때문에 알람 시간을 1씩 증가 조정하는 것을 가정한다. alarm_h, alarm_m에 값을 주면서 동시에 아웃풋이 동일하게 바뀌는 것을 볼 수 있다.



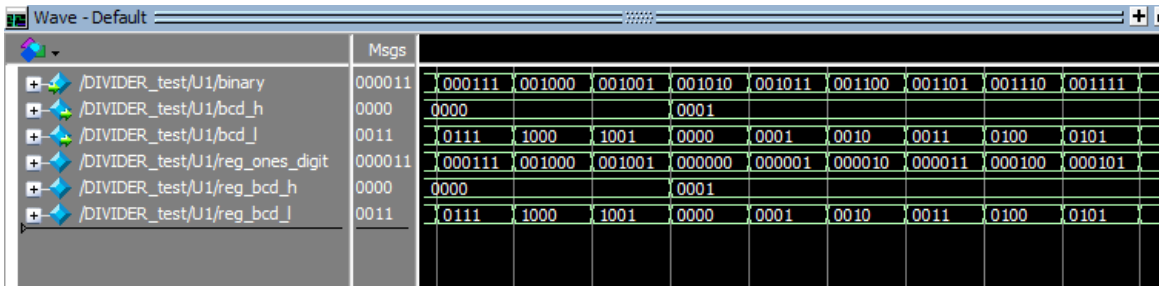
위 파형은 알람 설정 시간이 어떻게 잘 조정이 되는지 보여주는 것이다. inc_hours에 1이 들어오면 그 직후의 rising edge에서 reg_alarm_h에 +1이 되어 그 값이 동시에 alarm_h로 전달되는 것을 볼 수 있다. Inc_mins에 1이 들어왔을 때도 동일하게 alarm_m이 업데이트되는 것을 볼 수 있다.



알람이 울리는 것을 시뮬레이션한 것이다. 알람 시간을 나타내는 alarm_h, m은 2시 10분을 가리키고 있다. 이 때, 현재 시간을 나타내는 hours, mins에 2시 10분을 준다. 알람 시간과 현재 시간이 일치함을 확인하고 다음 rising edge에서 alarm 비트에 1이 들어옴으로서 알람이 울렸음을 알 수 있다.



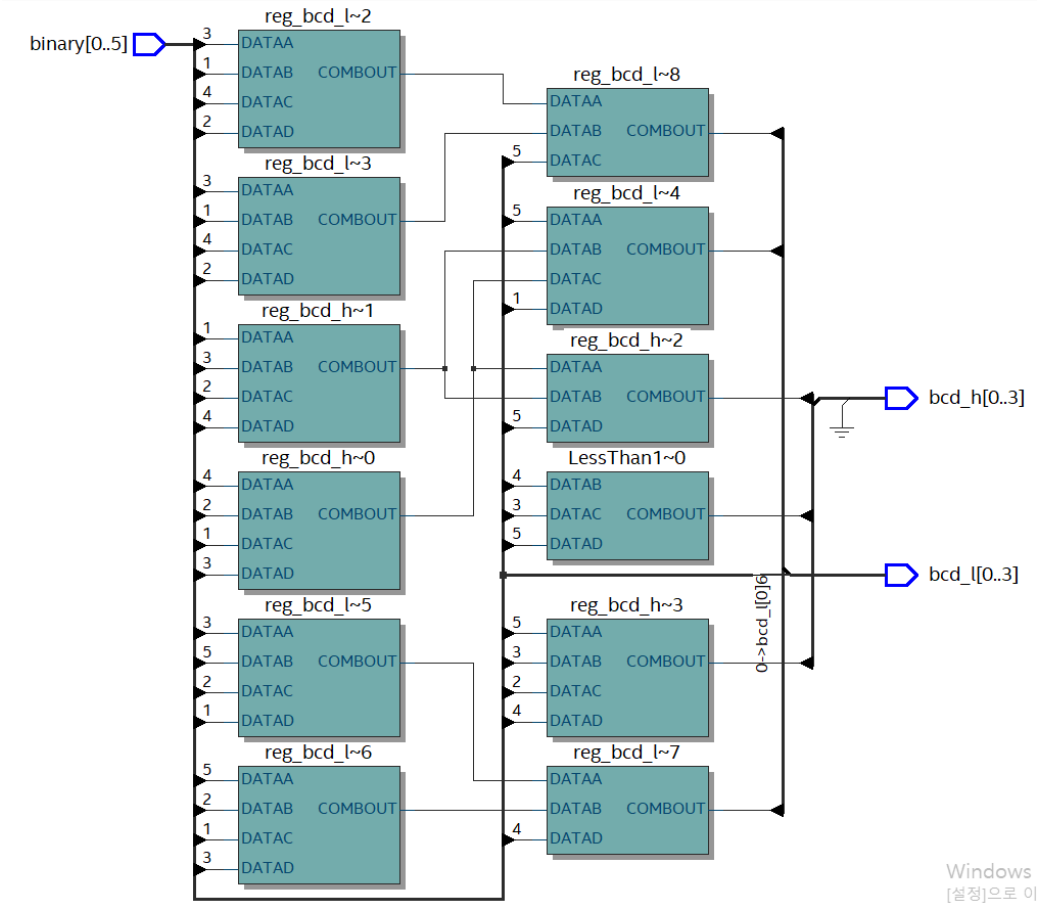
알람이 꺼지는 것을 시뮬레이션한 것이다. Alarm 비트가 계속 1이었다가, 알람을 끄는 신호인 set에 1을 주게 되면서 alarm은 0이 되고, 알람을 멈춘 시간을 나타내는 alarm_snooze_h, m도 알람이 울린 2시 10분을 저장하는 것을 볼 수 있다.



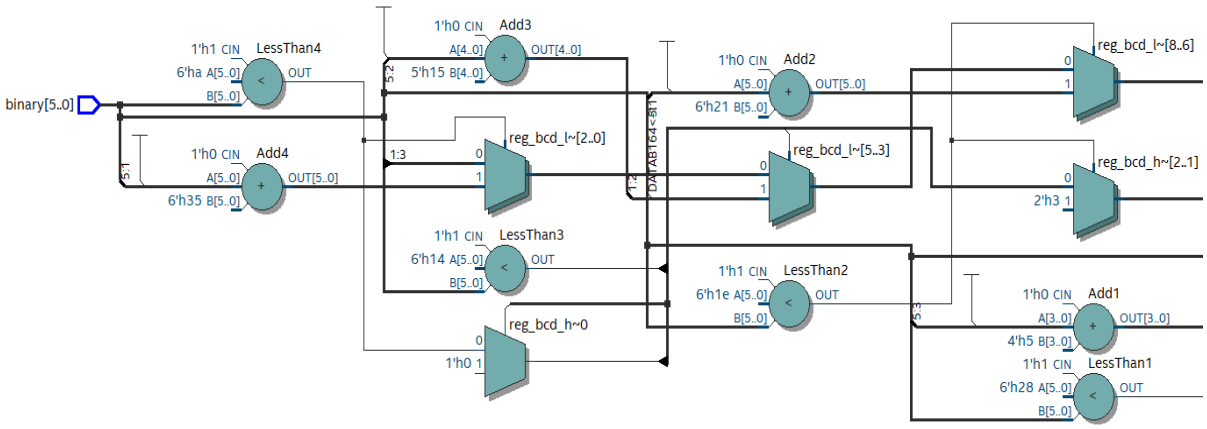
DIVIDER 모듈의 시뮬레이션은 인풋 값인 binary 비트에 0에서 59까지 값을 줘보면서 아웃풋을 관찰하는 것으로 시작한다. 파형을 보면 binary 값이 십진법으로 한 자리일 때는 십의 자리가 계속 0이다가 binary가 10일 때 십의 자리인 bcd_h에 1이 들어오는 것을 볼 수 있다. 일의 자리인 bcd_l도 값을 보면 알맞게 바뀐다.

4) Synthesis Results

DIVIDER 모듈

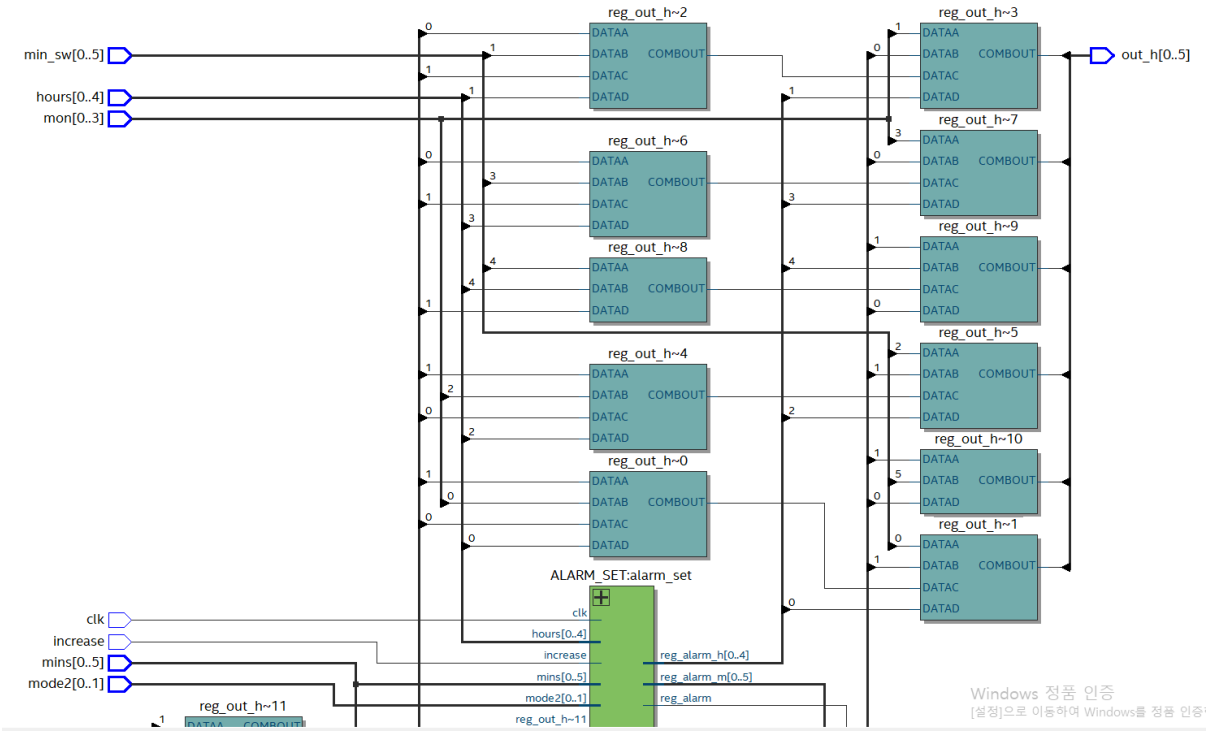


Quartus Prime에서 Technology map viewer를 선택하면 나타나는 것이다. bcd_h와 bcd_l이 각각 3개의 레지스터에 아웃풋을 받아서 만들어지는 것을 볼 수 있고, 그렇기 때문에 문제 없이 synthesis된 것 같다.

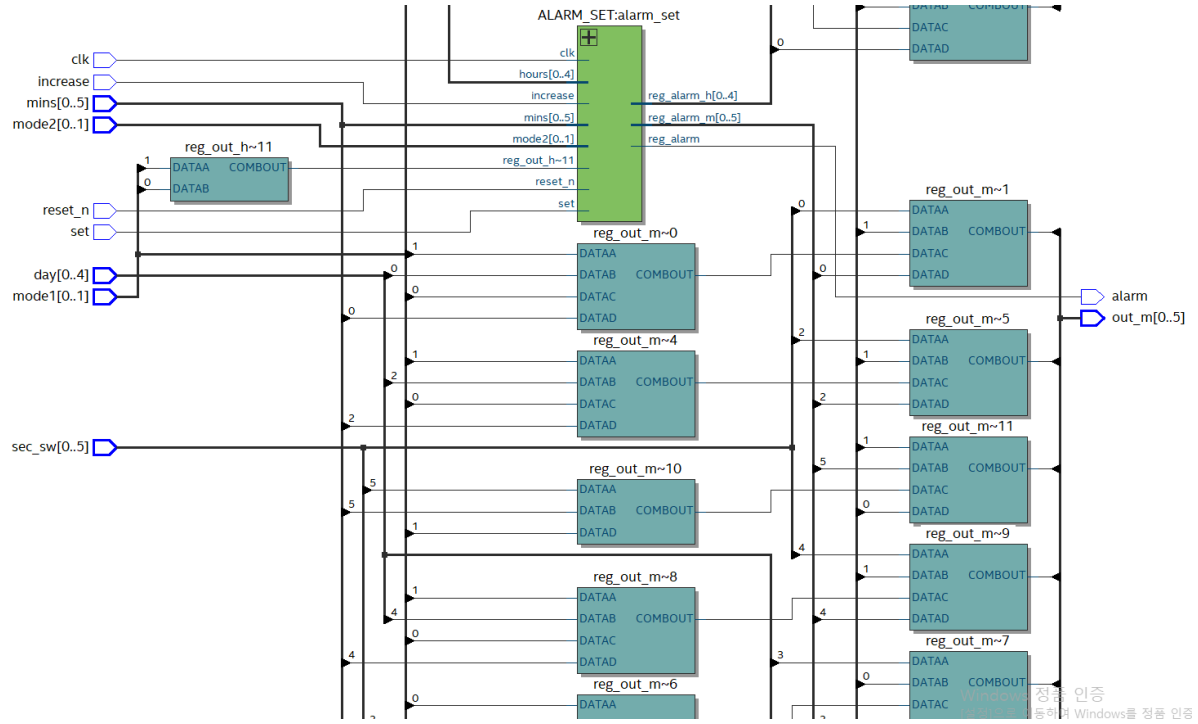


이것은 RTL Design의 일부이다. 아웃풋까지 가는 구조가 규칙적이라 일부만 첨부했다. Add4, Add3.. 와 같은 것들이 일의 자리 수의 아웃풋 중 1비트씩을 담당하고 있고, LessThan..와 같은 것들은 십의 자리 수를 표현할 때 50보다 작은 것, 40보다 작은 것, 이런 식으로 분류하기 위해 쓰인 것 같다. Add와 reg_bcd_l이 합쳐져서 technology map viewer에서 하나의 reg_bcd_l 레지스터가 된다고 할 수 있다. LessThan과 reg_bcd_h 또한 합쳐서 하나의 reg_bcd_h 레지스터라 할 수 있다.

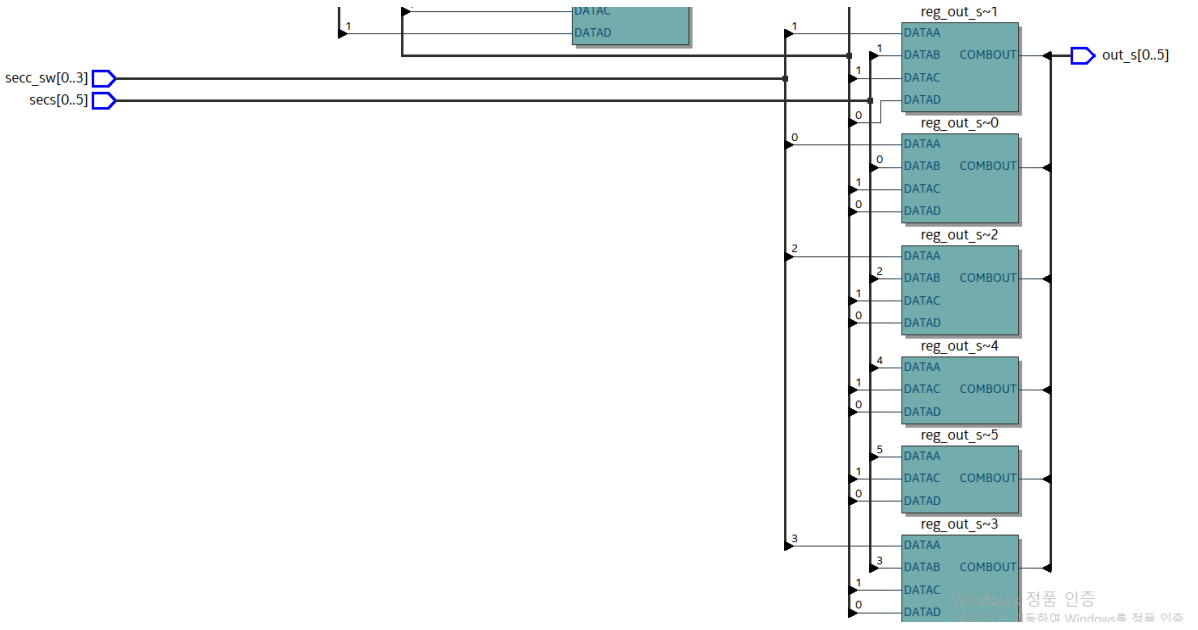
SELECTOR 모듈



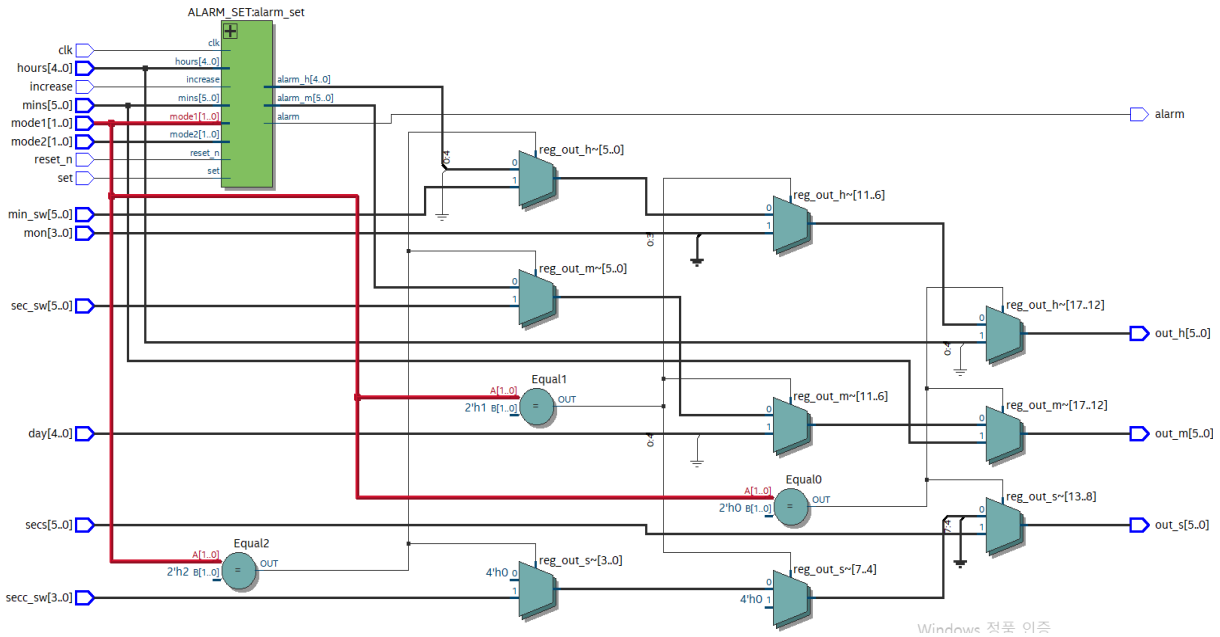
Technology map viewer의 상단부이다. Out_h의 아웃풋으로 들어갈 수 있는 인풋 후보인 TIME 모드의 hours, DATE 모드의 mon, TIMER 모드의 min_sw가 적절하게 연결이 된 것을 볼 수 있다.



Technology map viewer의 중반부이다. 모듈에서 사용되지 않는 인풋과 아웃풋은 포트의 테두리가 얇게 표현되어 있는 것을 볼 수 있다. Synthesis상 문제는 없어 보인다.



Technology map viewer의 하단부도 상단부와 마찬가지로 out_s에 들어갈 수 있는 아웃풋의 후보인 TIME 모듈의 secs와 TIMER 모듈의 secc_sw가 인풋으로 연결되어 있다.



SELECTOR의 RTL design이다. Mode1 인풋이 전달되는 회로를 보면 equal0, 1, 2에 우선 연결이 되어 있는데, mode1 값이 0, 1, 2 중 하나임에 따라 SELECTOR의 아웃풋이 달라지기 때문이다. Equal에서의 선이 연결되는 것을 보면 각각 다른 reg_out_h, m, s 와 연결이 되어있는 것을 볼 수 있다. 왜냐하면 아웃풋으로 보내야할 변수가 다르기 때문이다. 따라서 equal 레지스터와 reg_out 레지스터 하나씩이 합쳐져서 Technology map viewer에서의 reg_out_h나 m, s의 비트당 레지스터들을 이룬다고 할 수 있다.

4. Conclusion

디지털 시계라는 단순하고 친숙한 소재를 통해 어떤 기계의 회로를 낯설지만 그나마 재미있게 만들어본 것 같다. 기계에 들어가는 회로 칩이 만들어지는 과정의 일부를 잘 알게 된 시간이었다. Synthesis 할 때의 에러를 해결하는 데 많은 어려움이 있었다. 완벽하게 코드를 짰다고 생각했으나 latch가 여러 개

유도가 되었고, 2개의 always block에서 하나의 변수를 같이 사용하는 것도 안되어서 코드를 고쳤다. Latch는 변수 값이 hold되기 때문에 일어나는데 문제가 있는 코드 라인을 synthesis 프로그램에서 알려주어서 그래도 무사히 해결할 수 있었다.

5. 참고 문헌

[1-2] 민형복, "DigitalClock4.pdf," [Online]. Available: http://class.icc.skku.ac.kr/~min/di/ . [Accessed Nov. 27, 2020].

[3] Modelsim. (FPGA), Intel. [Accessed Nov. 27, 2020]. Available: https://fpgasoftware.intel.com/20.1/?edition=lite

[4] Intel Quartus Prime. (20.1), Intel. [Accessed Nov. 27, 2020]. Available: https://fpgasoftware.intel.com/20.1/?edition=lite