

**과 목 : 디지털 시스템**

**과 제 명 : HW3 DigitalClock**

**담당교수 : 민 형 복**

**학 과 : 소프트웨어학과**

**학 년 : 3**

**학 번 : 2016312761**

**이 름 : 여혁수**

**제 출 일 : 2020-11-17**

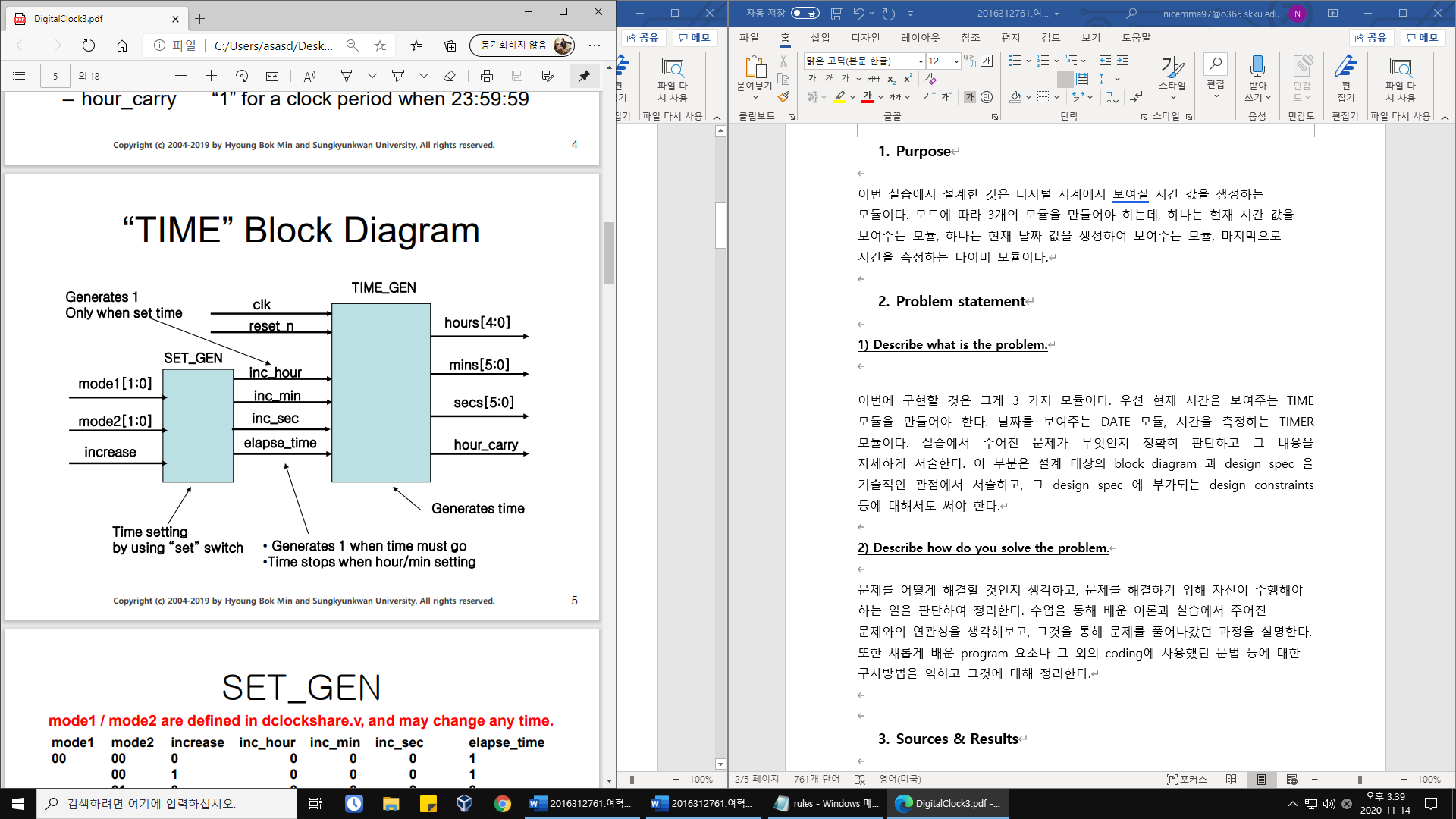
1. **Purpose**

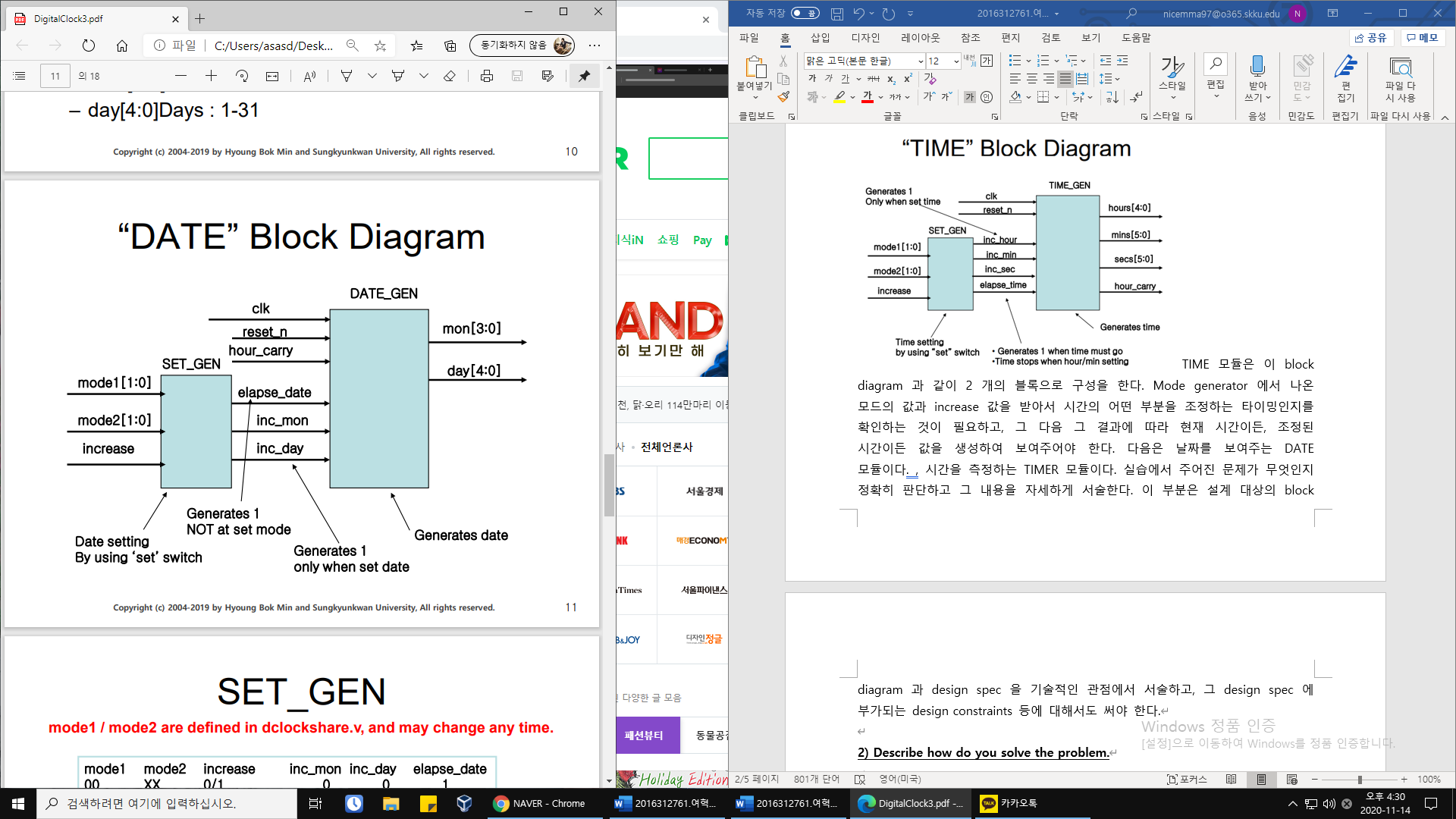
이번 실습에서 설계한 것은 디지털 시계에서 보여질 시간 값을 생성하는 모듈이다. 모드에 따라 3개의 모듈을 만들어야 하는데, 하나는 현재 시간 값을 보여주는 모듈, 하나는 현재 날짜 값을 생성하여 보여주는 모듈, 마지막으로 시간을 측정하는 타이머 모듈이다.

1. **Problem statement**

**1) Describe what is the problem.**

이번에 디지털 시계를 위해 구현해야할 것은 크게 3가지가 있다. 현재 시간을 생성하여 보여주는 TIME 모듈, 날짜를 생성하는 DATE 모듈, 그리고 시간을 측정하는 타이머 역할 TIMER 모듈이다.

TIME모듈은 이 block diagram과[1] 같이 2개의 블록으로 구성을 한다. Mode generator에서 나온 모드의 값과 increase값을 받아서 시간의 어떤 부분을 조정하는 타이밍인지를 확인하는 것이 필요하고, 여기서 나온 결과값에 따라 현재 시간이든, 조정된 시간이든 초, 분, 시의 값을 생성하여 보여주어야 한다. 100개의 rising edge마다 1초가 지나는 것을 기본으로 한다. 또한 하루가 지나는 경우에 날짜를 조정하기 위해 이를 위한 아웃풋 비트도 생성한다. 다음은 날짜를 보여주는 DATE 모듈이다.

TIME모듈과 비슷한 block diagram인데[2] DATE모듈도 역시 모드 값과 increase값을 받아서 날짜를 조정하는 타이밍인지 아닌지를 판단하는 것이 날짜 값을 생성하고 보여주기 전에 필요하다. 날짜를 조정하는 모드일 경우 조정한 날짜의 값을 보내주고, 그게 아니면 흐른 시간에 따라 나온 현재 날짜를 생성하여 내보내면 될 것이다. 날짜의 initial state는 1월 1일이다.

마지막으로 시간을 측정하는 TIMER 모듈이다. TIMER 모듈은 하나의 블록으로만 구성되고, mode generator에서 나온 모드 값을 보고 TIMER 모드 일 때 작동한다. TIMER\_START 모드에서 시간 측정을 시작하고, TIMER\_STOP 모드에서 측정을 중지한다. TIMER\_G로 돌아가면 측정 시간이 리셋된다. 그 외 다른 모드로 변경이 될 때도 측정 시간을 리셋한다. 시간은 100ms 단위까지 측정하는데, 10개의 rising edge 마다 100ms를 증가시키는 방식을 기본으로 한다.

**2) Describe how do you solve the problem.**

TIME 모듈은 교수님의 자료인 time.v 파일을 그대로 활용한다. 나머지 2개의 모듈의 Verilog 코드 또한 time.v의 코드 스타일을 참고하여 작성했다. 이번에 코드를 작성하면서 function에 대해 새롭게 배워 사용하게 되었는데, function도 모듈과 비슷하게 인자를 선언해주고, 그 다음 따로 사용하는 변수를 선언해준다. 그리고 function code를 넣고 endfunction 커맨드로 마무리한다. Verilog 코딩을 마치고, Modelsim[3] 프로그램을 통해 simulation 과정을 거친다. 각 모듈이 정상적으로 simulation이 되는 것을 확인하고, 3개의 모듈을 하나로 합친 것도 simulation 한다. 그리고 Intel Quartus Prime[4] 프로그램을 통해서 각 모듈이 정상적으로 synthesize 됨을 확인함으로서 이번 문제의 해결을 마친다.

1. **Sources & Results**

**1) Analysis of HDL Source Codes of Design**

date.v

if ((mode1 == M1\_DATE) && (increase == 1)) begin

case (mode2)

M2\_DATE\_MON : begin // increment month

inc\_mon = 1;

inc\_day = 0;

end

이 부분은 DATE 모듈에서 날짜 값을 생성하기 전에 날짜를 조정해야하는지 아닌지를 판단하는 부분이다. 우선 모드가 M1\_DATE, 즉 DATE 모드여야 하고, 어떤 시간 값을 1 증가시켜라는 의미의 increase 값에 1이 들어와야한다. 조건이 만족되었을 때 mode2, 즉 세부 모드를 확인하여 month를 조정하는 모드면 위와 같이 inc\_mon에 1을 줌으로서 다음 날짜의 값을 생성할 때 이 inc\_mon값이 영향을 주게 된다. 마찬가지로 day를 조정하는 모드라면 inc\_day에만 1을 주게 된다. 그 외의 모드에서는 두 비트 다 0을 준다.

if (inc\_mon == 1) begin

if (reg\_mon == 12)

reg\_mon <= 1;

else

reg\_mon <= reg\_mon + 1'b1;

이 부분은 날짜 값을 생성하는 블록의 일부분이다. inc\_mon에 1이 들어왔다면 달의 값을 1 증가시켜주면 되는데, 현재 달이 12월일 경우에는 1월로 되돌아가게 했고, 그 외에는 +1 해주었다.

end else if (inc\_day == 1) begin

if (reg\_mon == 2) begin // if February

if (reg\_day == 28)

reg\_day <= 1;

else

reg\_day <= reg\_day + 1'b1;

inc\_day가 1이 들어오면 날짜에서 1일을 증가시켜주면 되는데, 달마다 몇 일이 마지막 날인지가 다르다. 위 코드는 2월인 경우만 보여주었는데, 나머지 달도 이런 식으로 코딩하였다. 2월이라면 현재 날이 28일이면 1로 되돌아가게 하였고, 그게 아니라면 +1 해주었다.

DATE 모듈에서 이와 같이 날짜를 조정하는 모드가 아닌, 일반적인 시간이 흐르고 있는 것을 가정한 경우도 날짜 값을 생성하는 논리는 동일하다.

timer.v

if (mode1 == M1\_TIMER) begin

if (mode2 == M2\_TIMER\_START) begin

if (count10 == MAX\_COUNT) begin // 100 microseconds elapsed

count10 <= 0;

if (reg\_secc\_sw == 9) begin // 1 second elapsed

reg\_secc\_sw <= 0;

if (reg\_sec\_sw == 59) begin // 1 minute elapsed

reg\_sec\_sw <= 0;

if (reg\_min\_sw == 59) begin // 1 hour elapsed

reg\_min\_sw <= 0;

이 부분은 타이머에서 시간이 측정되고 있을 때를 구현한 부분이다. M1\_TIMER 모드에서 세부 모드가 M2\_TIMER\_START일 때가 타이머 시작 스위치를 눌렀을 때라고 할 수 있다. 이 때 10개의 rising edge가 지났다면 count10이 MAX\_COUNT와 같아져 100 ms가 지났음을 알려준다. 100ms가 지났을 때 100ms 단위 시간 값이 9라면 1초가 된다는 것이므로 100ms 단위 값은 0이 된다. 이 때 초 단위 값 또한 59라면 1분이 지났다는 것이므로 초 단위 값 또한 0이 된다. 같은 방식으로 1시간이 지난 경우 분 단위 값 또한 0이 되어야할 것이다.

end else begin

reg\_min\_sw <= reg\_min\_sw + 1'b1;

end

end else begin

reg\_sec\_sw <= reg\_sec\_sw + 1'b1;

end

end else begin

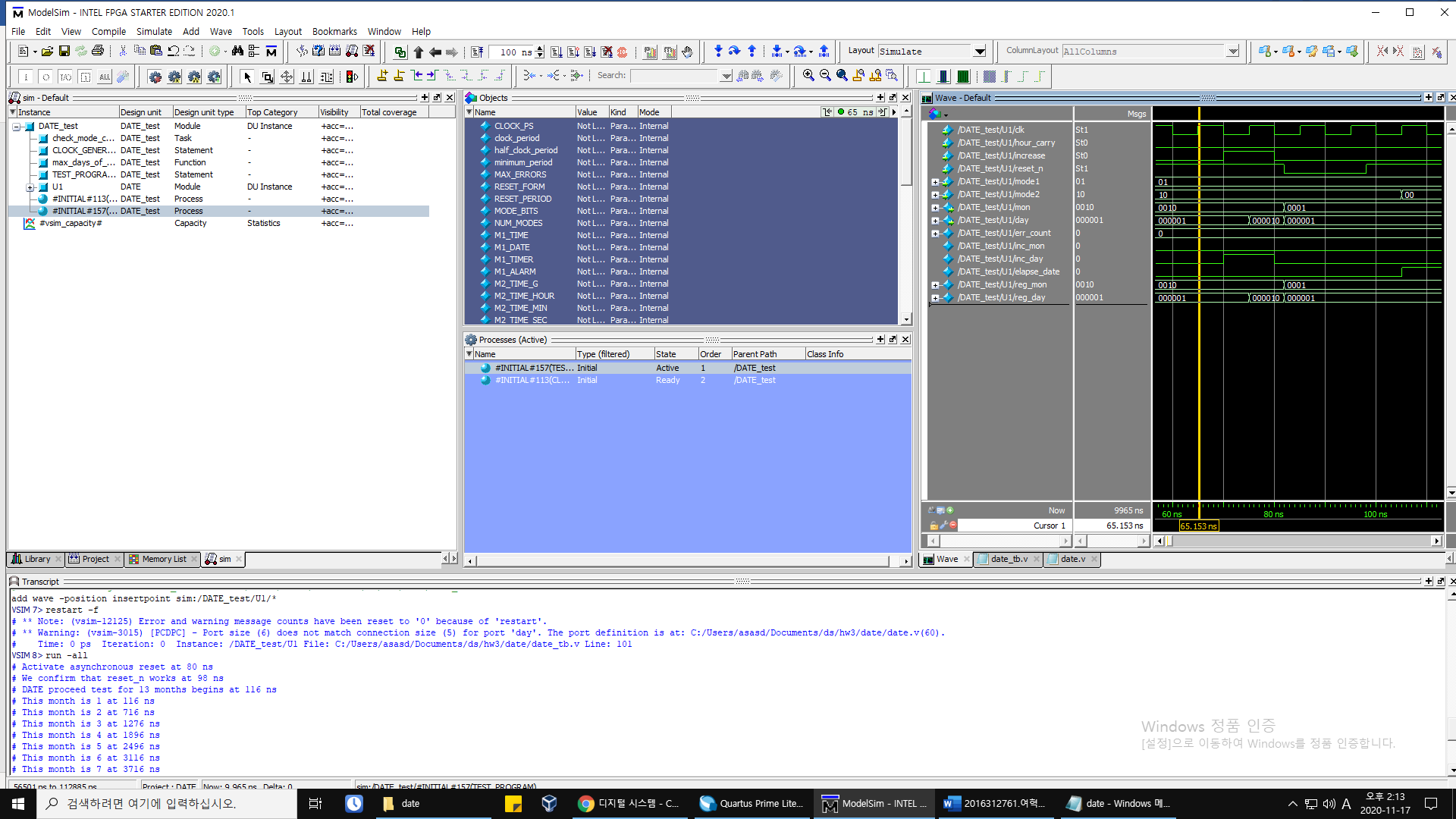
reg\_secc\_sw <= reg\_secc\_sw + 1'b1;

end

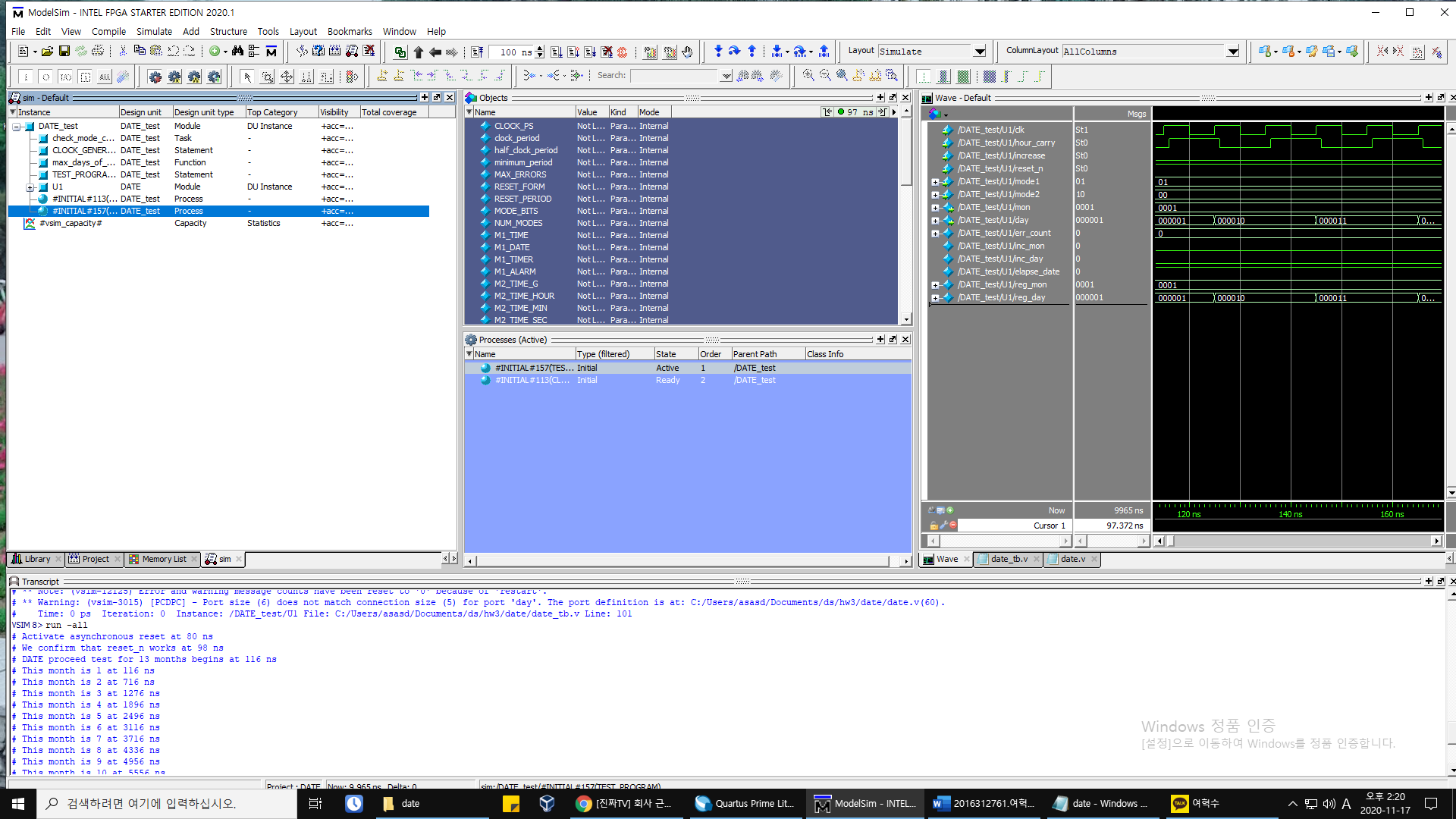
이전의 코드에서 곧바로 이어지는 코드를 발췌한 것이다. 1분이 지났지만 1시간이 지나진 않은 경우 분 단위 값에 +1을 해주면 될 것이고, 마찬가지로 1초는 지났지만 1분이 지난 것은 아니라면 초 단위 값에 +1을 해주면 될 것이다. 1초도 지난 것이 아니라면 100 ms 단위 값을 +1해준다.

**3) Simulation Waveforms**

DATE 모듈)



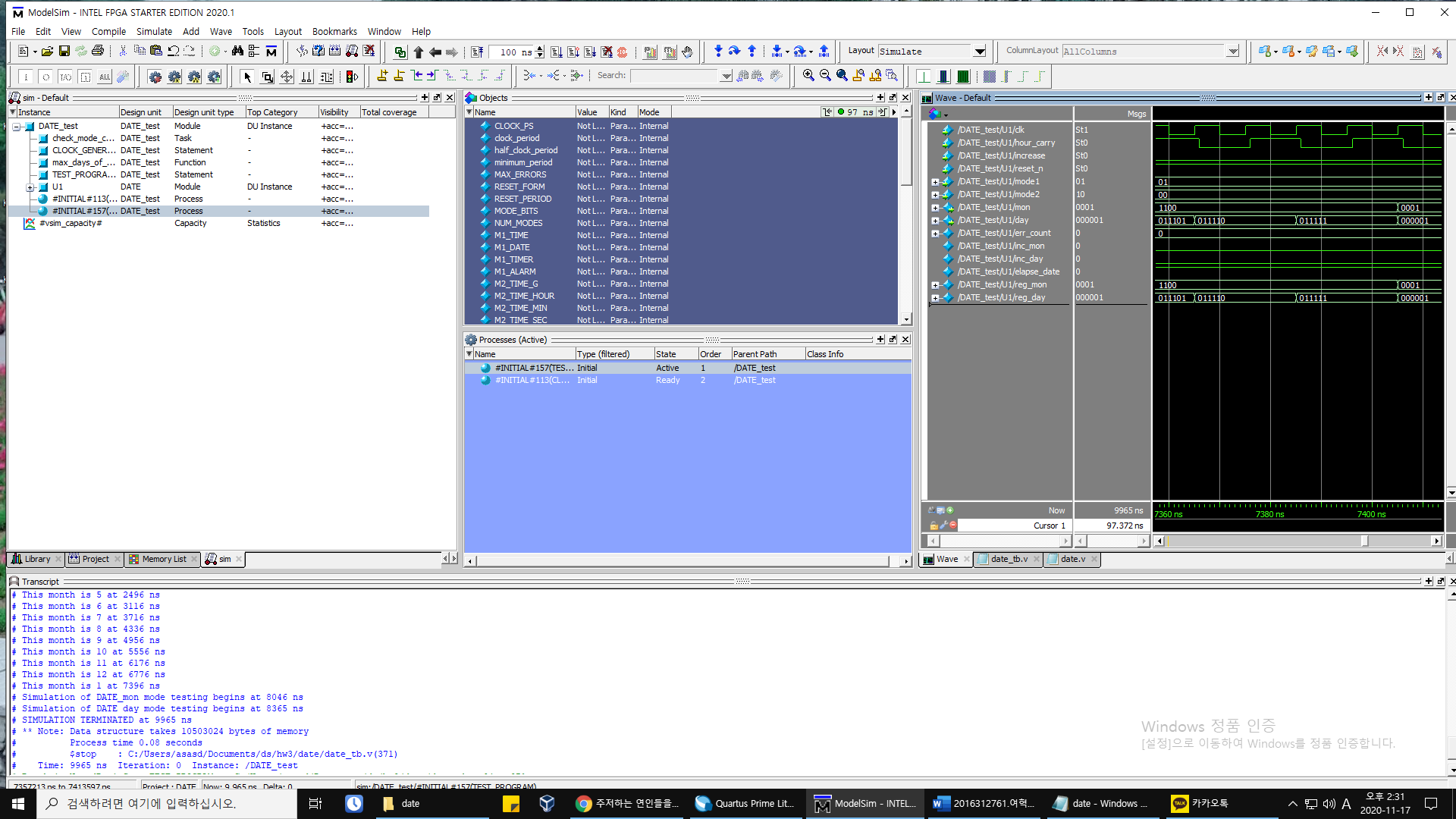
80ns에서 asynchronous reset을 주었을 때의 파형이다. reset\_n이 negedge를 보여주고, 동시에 reg\_mon, reg\_day가 1로 초기화되어 그 값을 유지하는 것을 볼 수 있다.



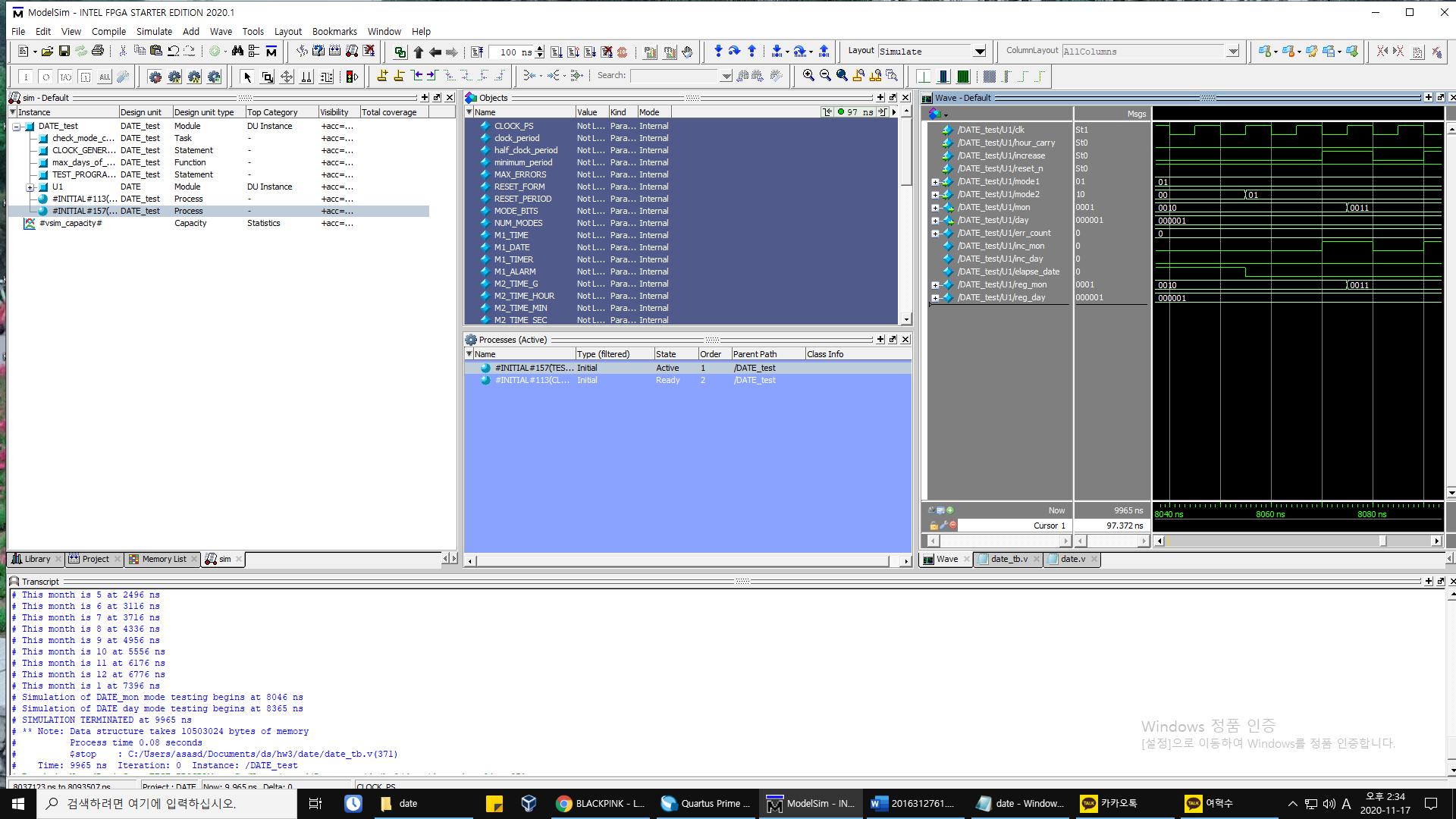
116ns부터 1일씩 날짜를 넘기며 적절한 타이밍에 다음 달로 잘 넘어가는지 테스트를 시작한다. reg\_day를 관찰해보면 1씩 일정 주기로 증가하는 것을 볼 수 있다.



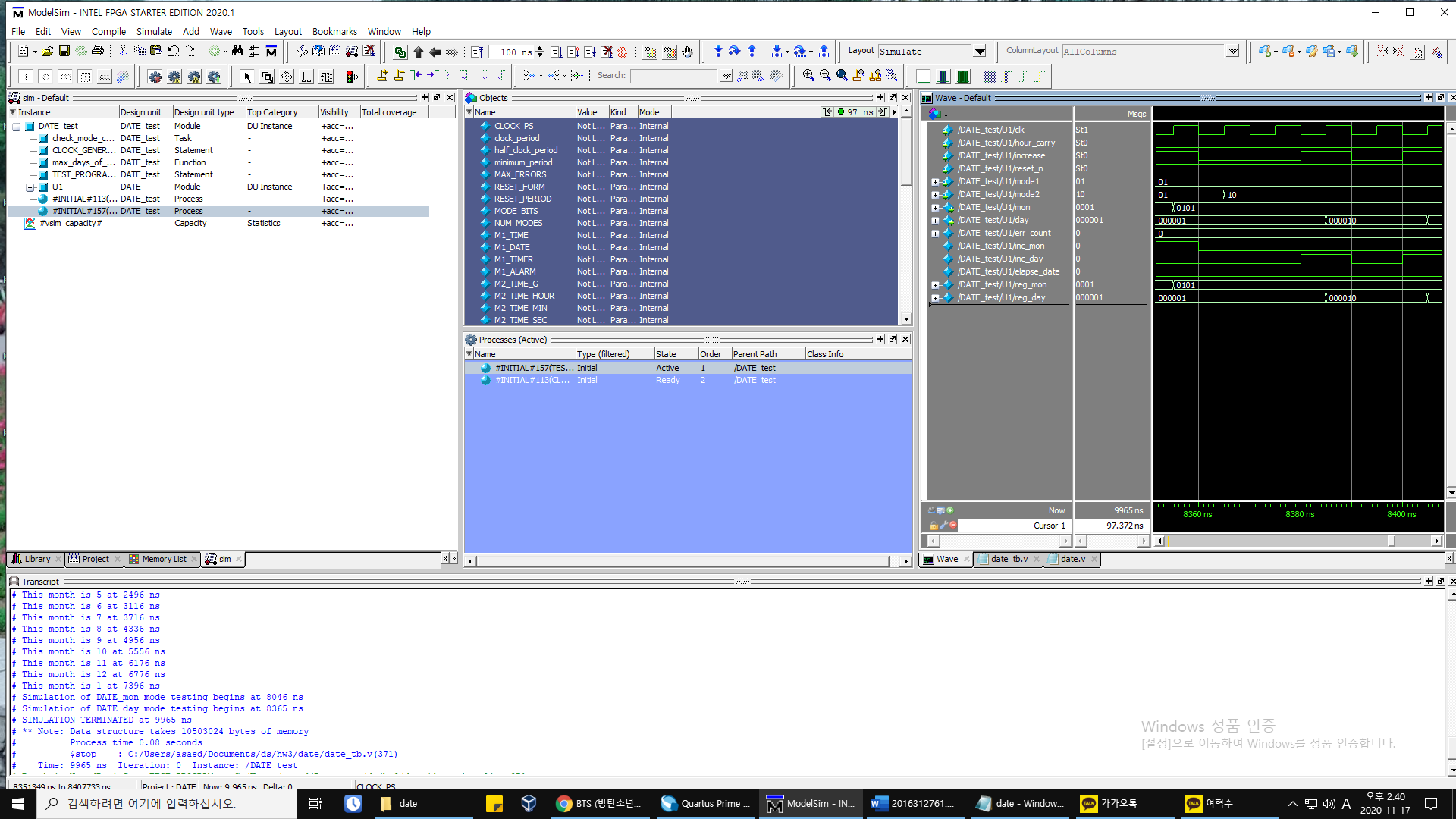
계속 1일씩 지나가다가 2월로 넘어가는 716ns 전후의 파형이다. Reg\_day를 봤을 때 값이 31까지 증가한 것을 볼 수 있고, 그 다음 값이 1로 되는 것을 볼 수 있다. 같은 타이밍에 reg\_mon가 2로 적절히 바뀌는 것을 볼 수 있다.



12월 31일까지 지나는 것을 시뮬레이션하고 1월 1일이 되는 7400ns 전후의 파형이다. Mon, day가 각각 12, 31인 것을 확인할 수 있고, 7400ns가 조금 지나서 둘 다 1로 바뀌는 것을 볼 수 있다.

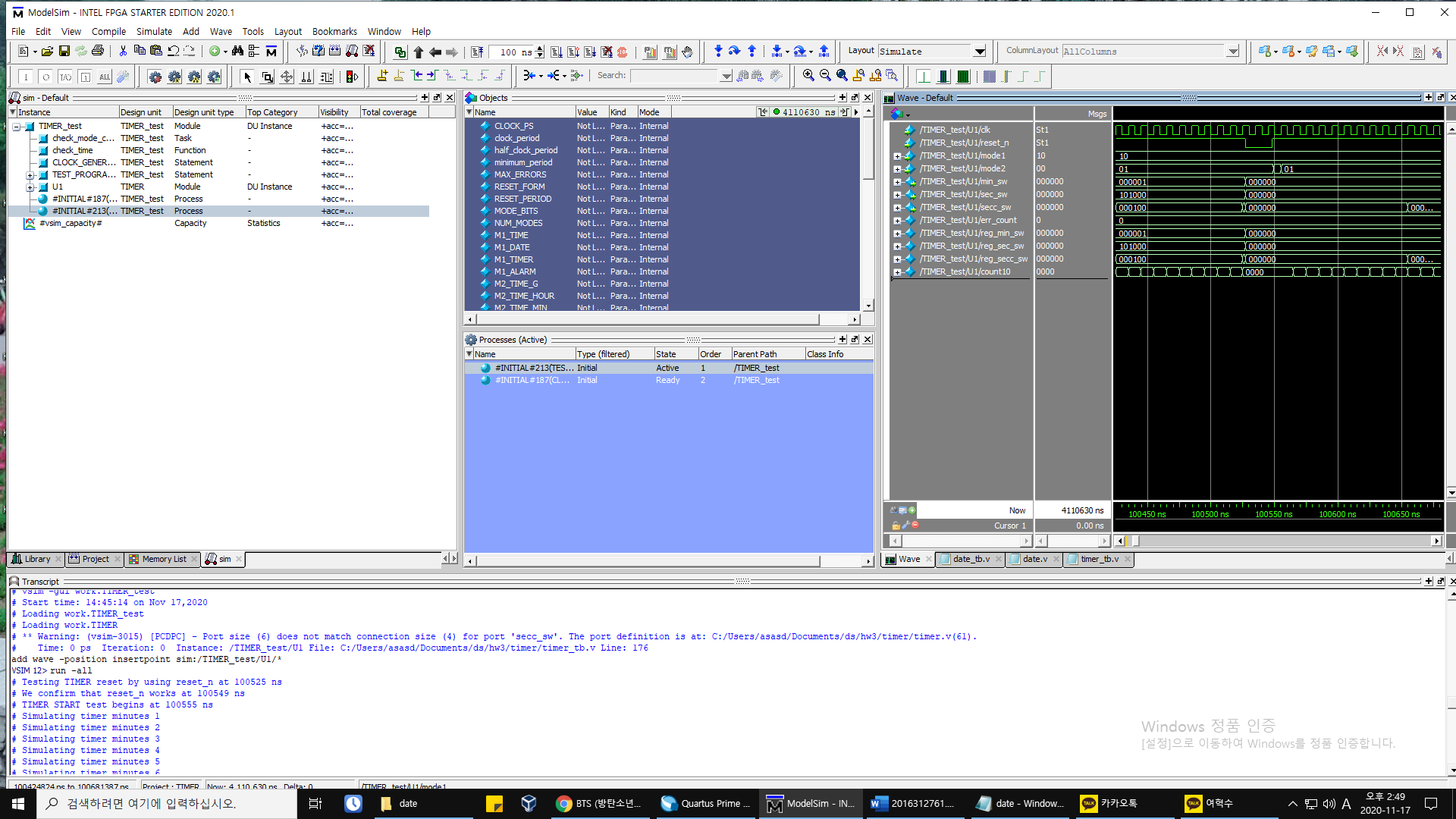


이전까지는 시간이 흐르는 것을 가정했을 때이고, 이것은 날짜에서 달을 조정하는 모드인 DATE\_MON 모드를 시뮬레이션한 파형이다. mode2를 보면 00에서 01로 되면서 DATE\_MON 모드로 바뀌었음을 알 수 있다. 이후 8070ns 에서 inc\_mon에 1을 주었고, 다음 clk의 rising edge에서 inc\_mon을 확인해 reg\_mon이 1 증가하는 것을 볼 수 있다.

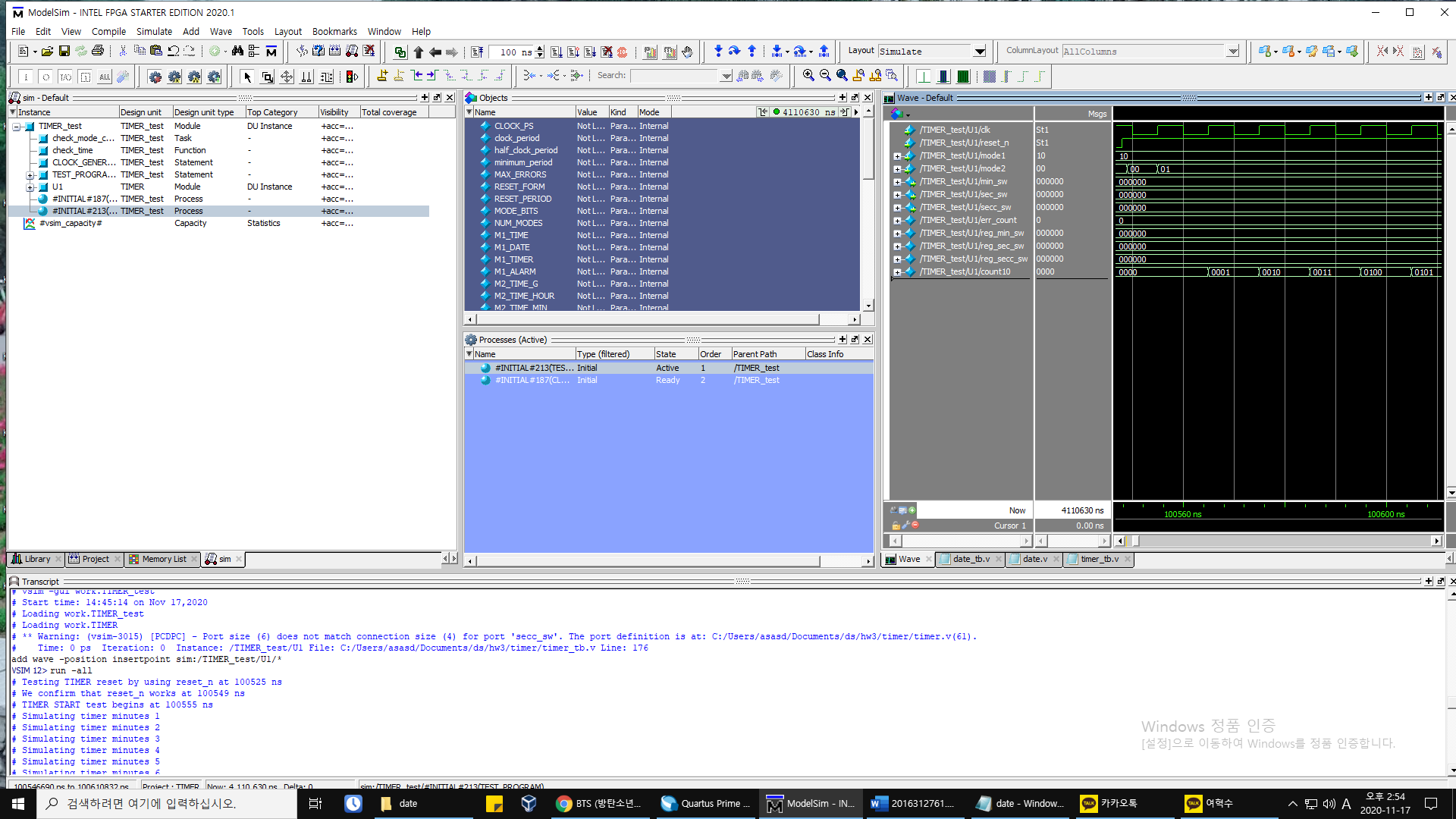


날짜에서 day를 조정하는 모드인 DATE\_DAY 모드를 시뮬레이션 할 때의 파형이다. mode2를 보면 2’b01에서 2’b10으로 되면서 DATE\_DAY 모드로 바뀌었음을 알 수 있다. 이후 8380 ns에서 inc\_day에 1을 주었고, 다음 clk의 rising edge에서 inc\_day 값을 확인해 reg\_day가 1 증가하는 것을 볼 수 있다.

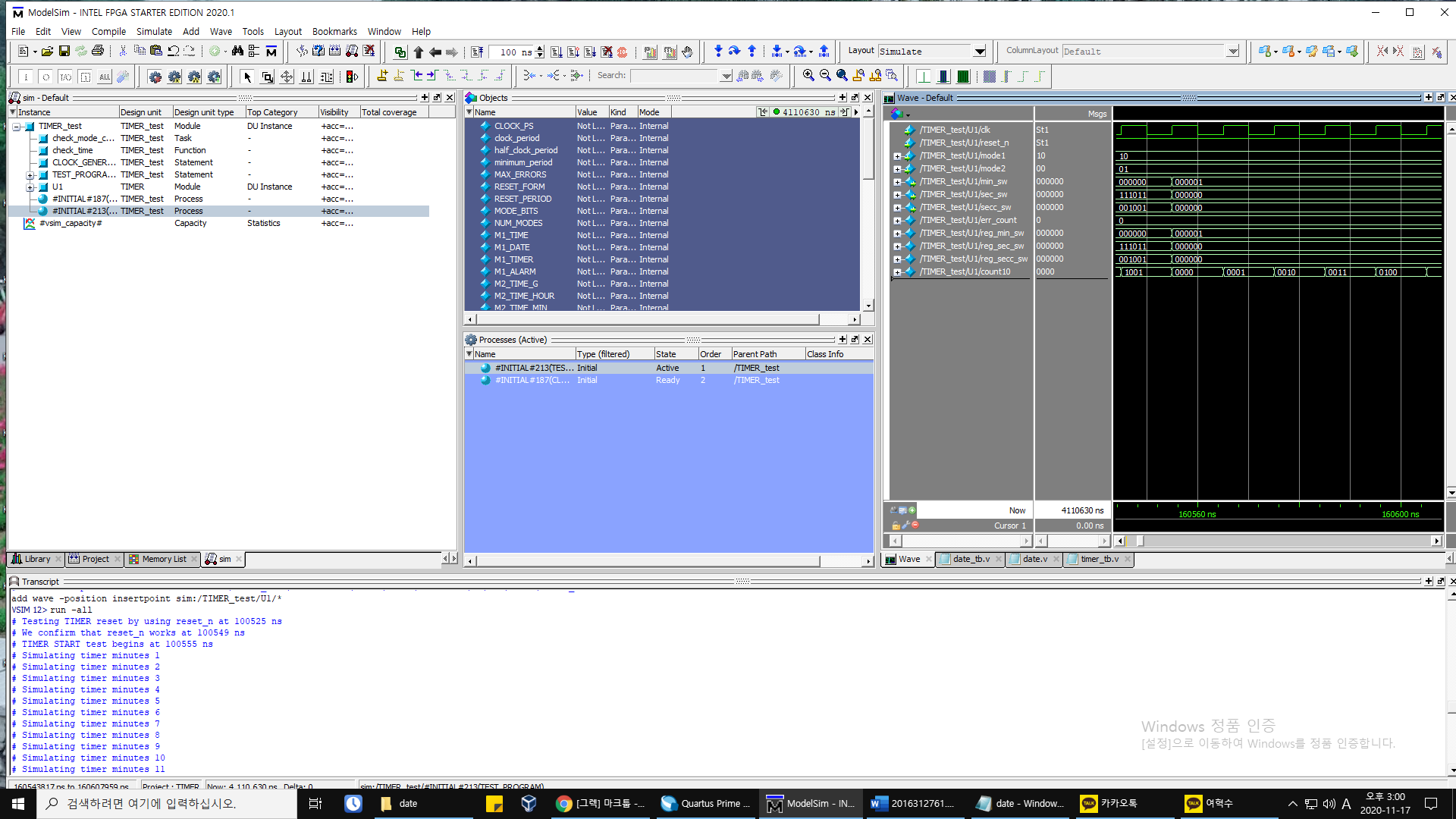
TIMER 모듈)



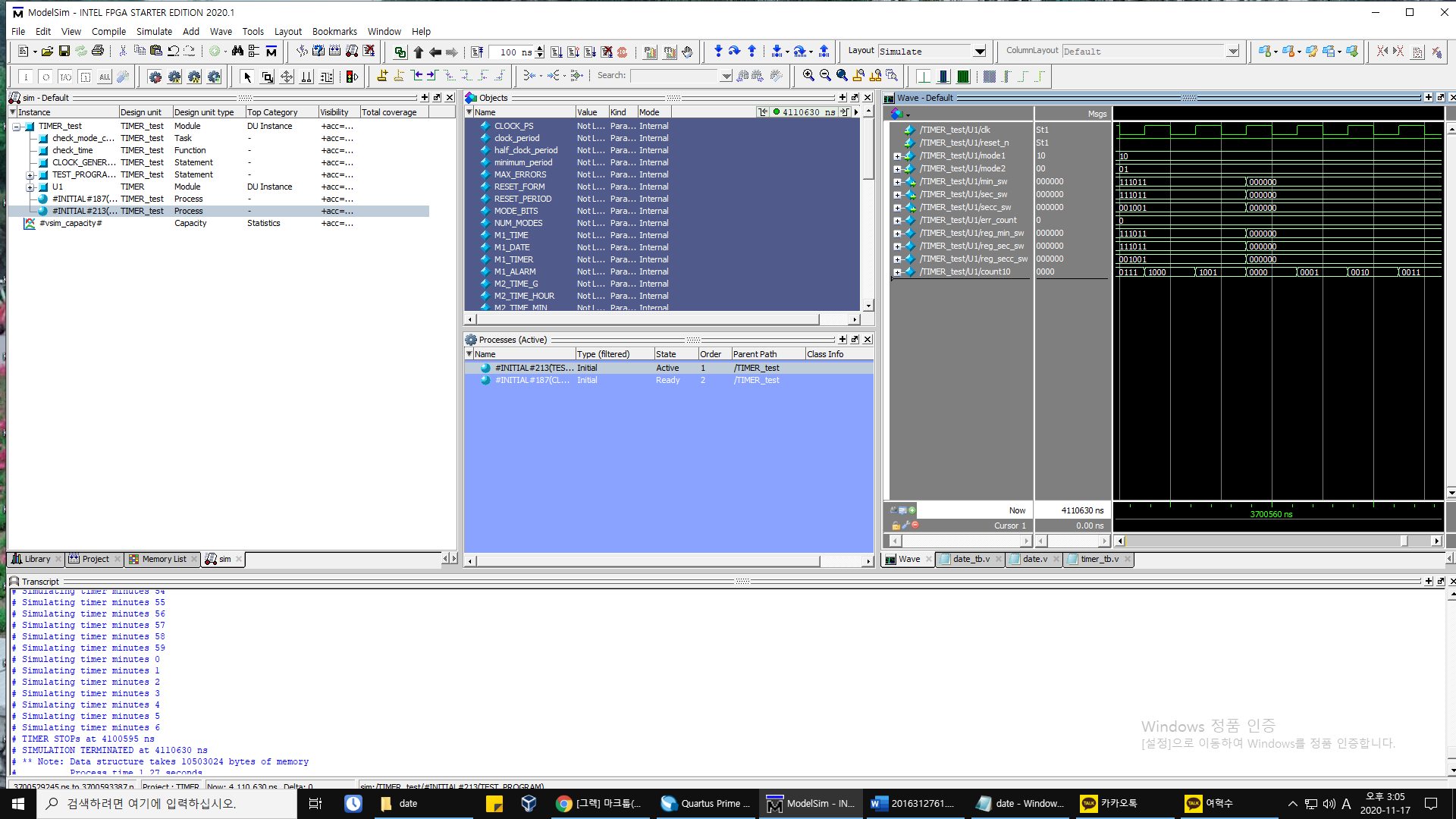
100525ns에 asynchronous reset을 주었을 때의 파형이다. reset\_n이 negedge일 때 동시에 min\_sw, sec\_sw, secc\_sw, 그리고 count10까지 0으로 초기화되는 것을 볼 수 있다.



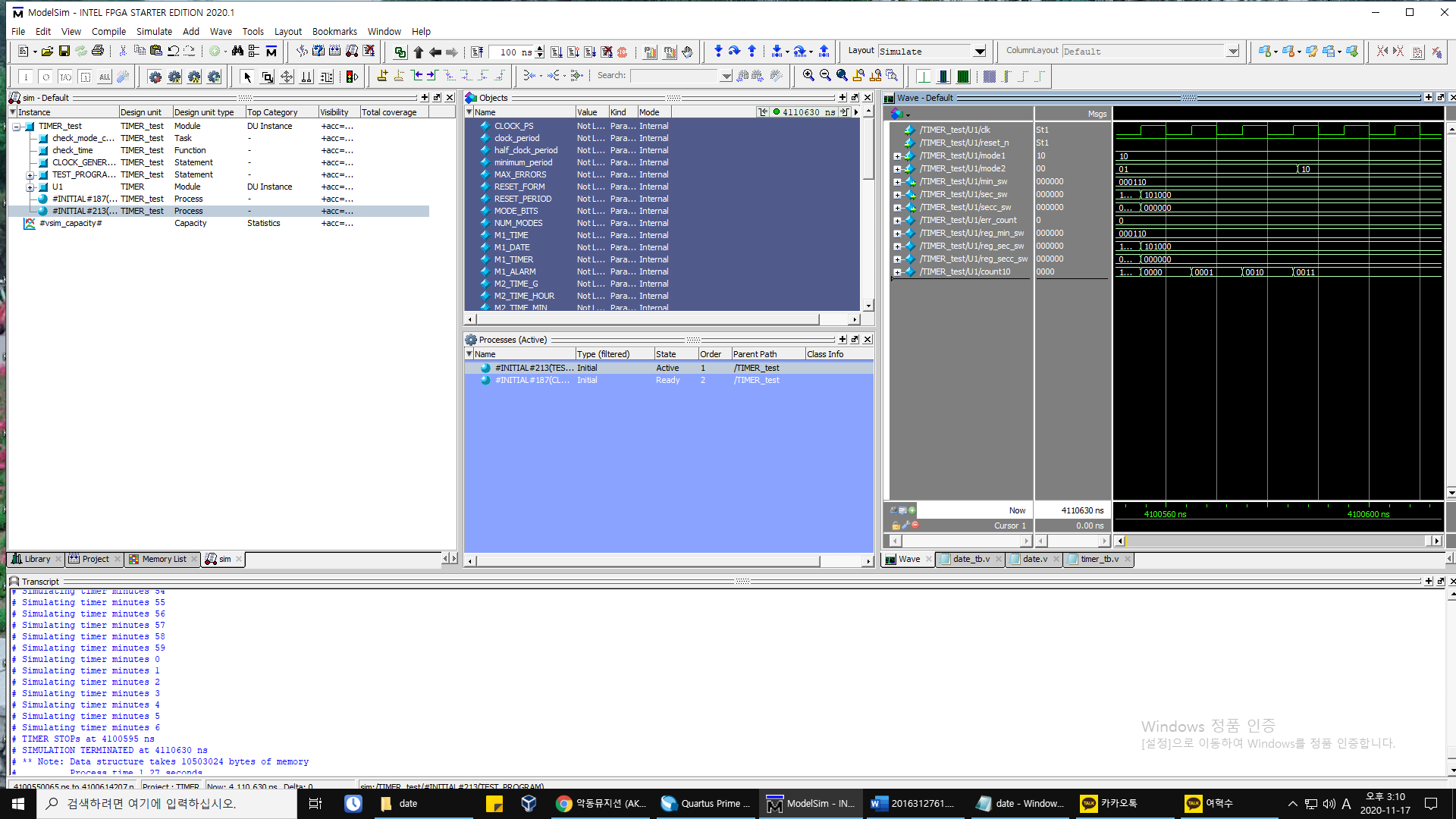
100555 ns에 타이머의 시작 버튼이 눌린 상황을 시뮬레이션한 파형이다. mode2가 2’b01로 바뀌면서 TIMER\_START 모드가 되었음을 알 수 있다. 그 직후의 rising edge에서부터 100ms를 counting하는 count10이 1씩 증가하며 counting되는 것을 볼 수 있다.



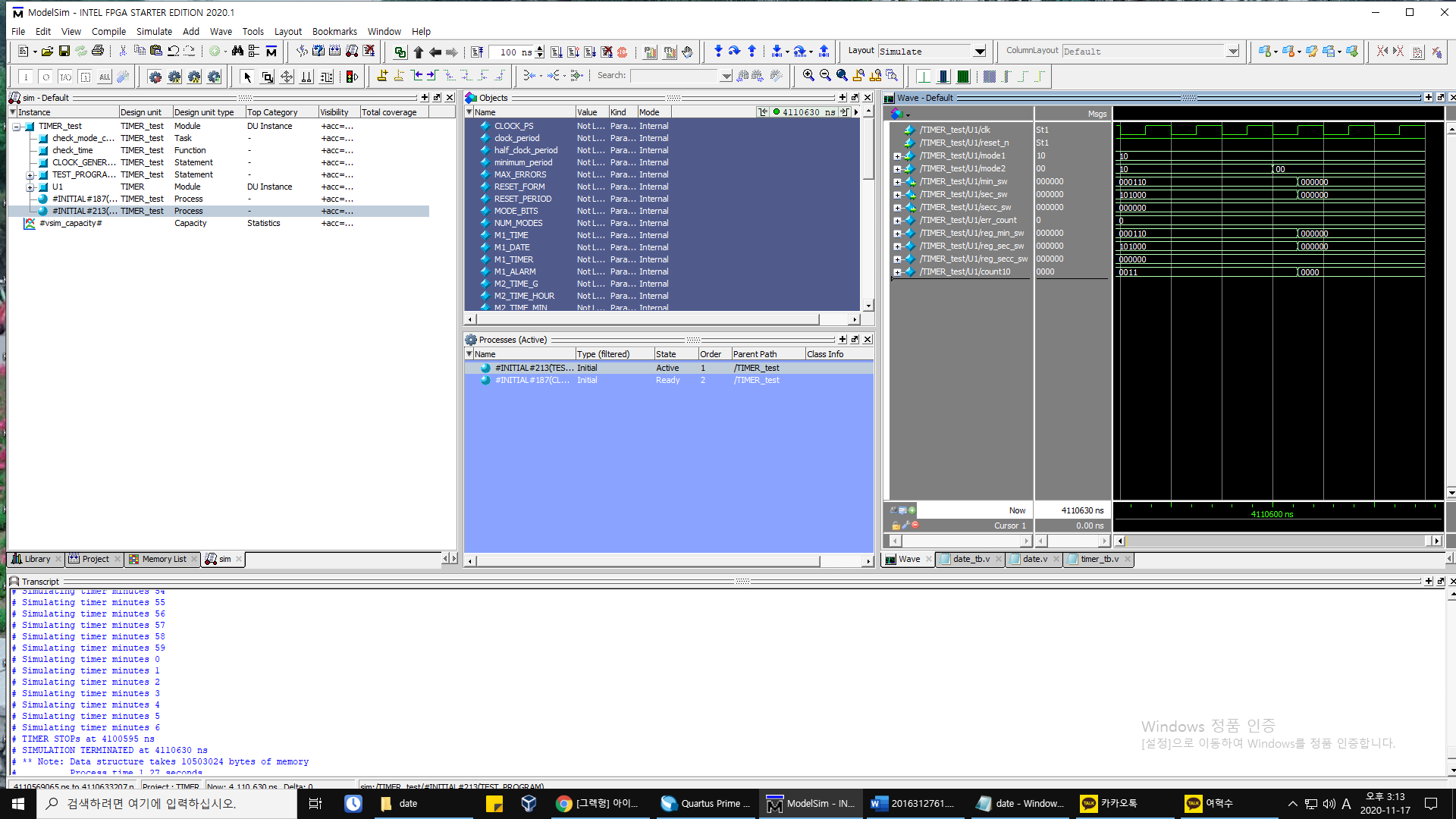
TIMER\_START 모드에서 시간이 계속 흐르다가 1분이 지났을 때의 파형이다. count10에서 10개의 counting을 했을 때, 100ms 단위 값이 9이고, sec 단위 값이 59인 것을 볼 수 있다. 그래서 이후의 값은 min 단위 값이 0에서 1로 바뀌고 나머지는 0으로 바뀌는 것을 볼 수 있다.



다음은 시간이 계속 흘러 1시간이 되기 직전인 59:59:09 일 때 다시 시간이 자동으로 0부터 시작하는 것을 보여주는 파형이다. count10이 9일 때 다음 rising edge에서 모든 시간 값이 0으로 초기화되는 것을 볼 수 있다.

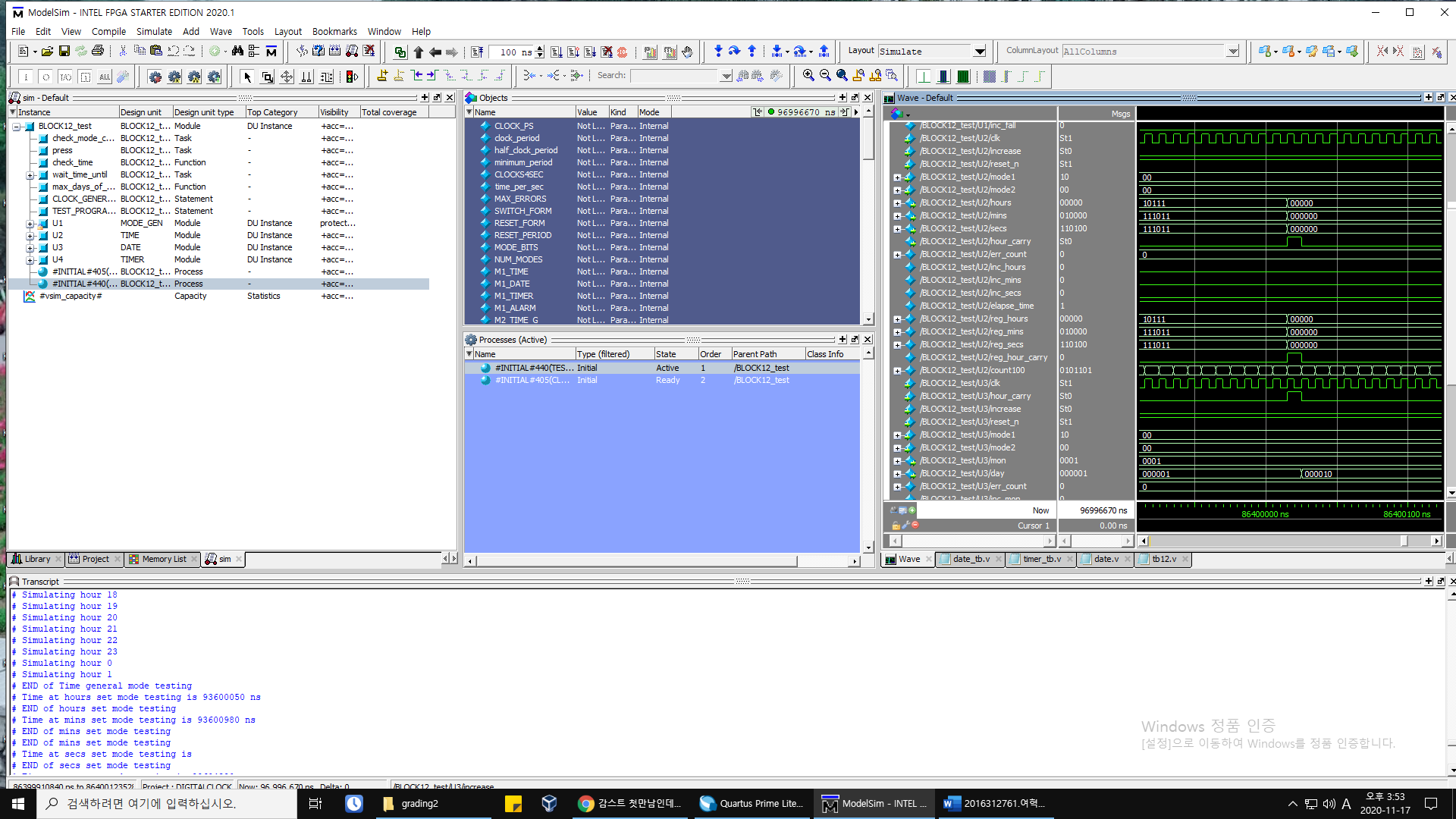


타이머를 중지한 것을 시뮬레이션한 파형이다. mode2가 2’b01에서 2’b10으로 바뀐 것을 보아 TIMER\_STOP 모드로 간 것을 알 수 있다. 그리고 모드 변경 이후 count10을 비롯해 모든 시간 값이 변화가 없는 것을 보아 정상적으로 중지된 것을 볼 수 있다.

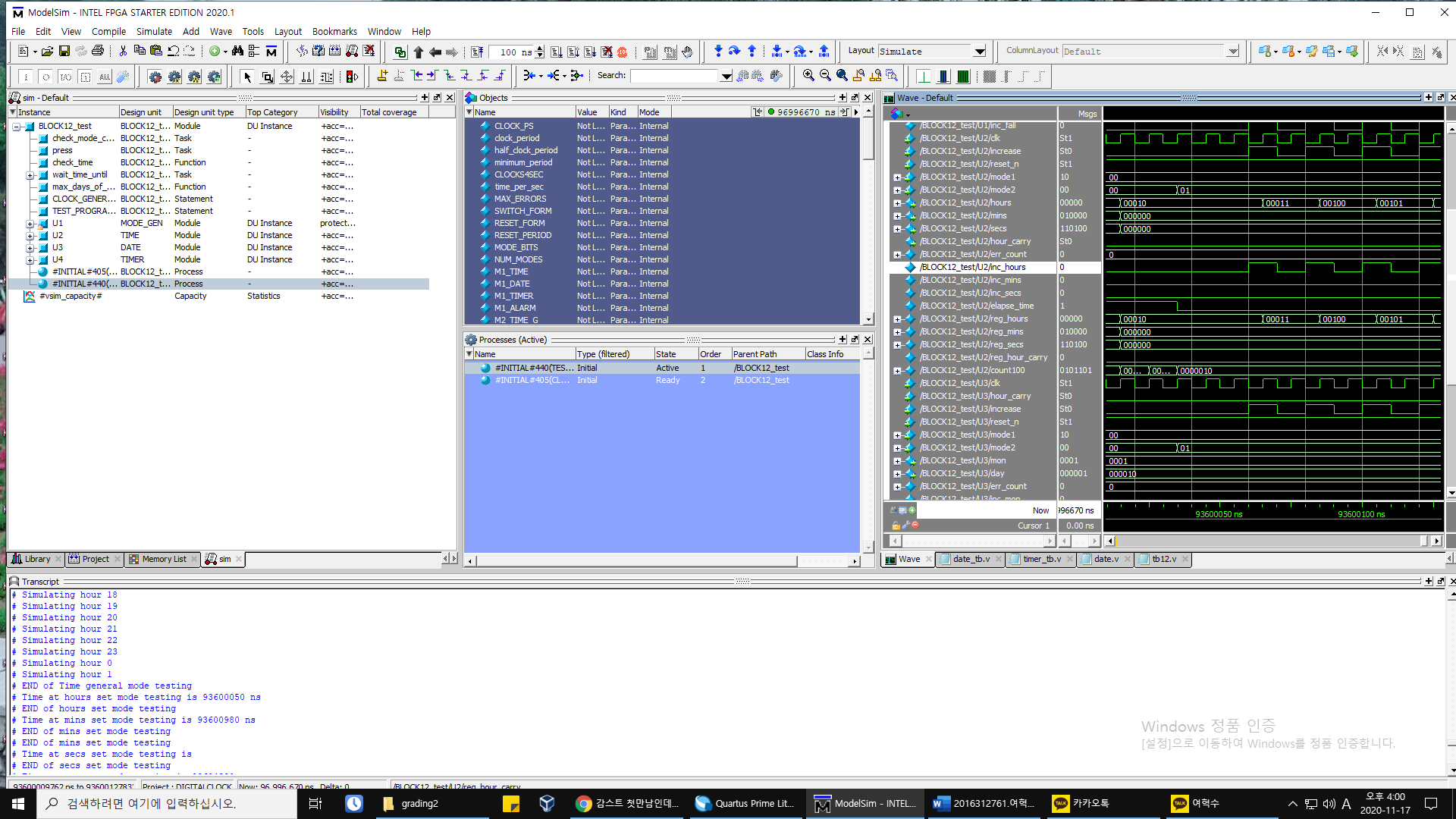


마지막으로 다시 세부 모드를 TIMER\_G 모드로 바꾸면서 시간을 리셋하는 것을 시뮬레이션 한 파형이다. Mode2가 00, 즉 TIMER\_G로 바뀌고 다음 rising edge에서 시간 값이 모두 0으로 바뀌는 것을 볼 수 있다.

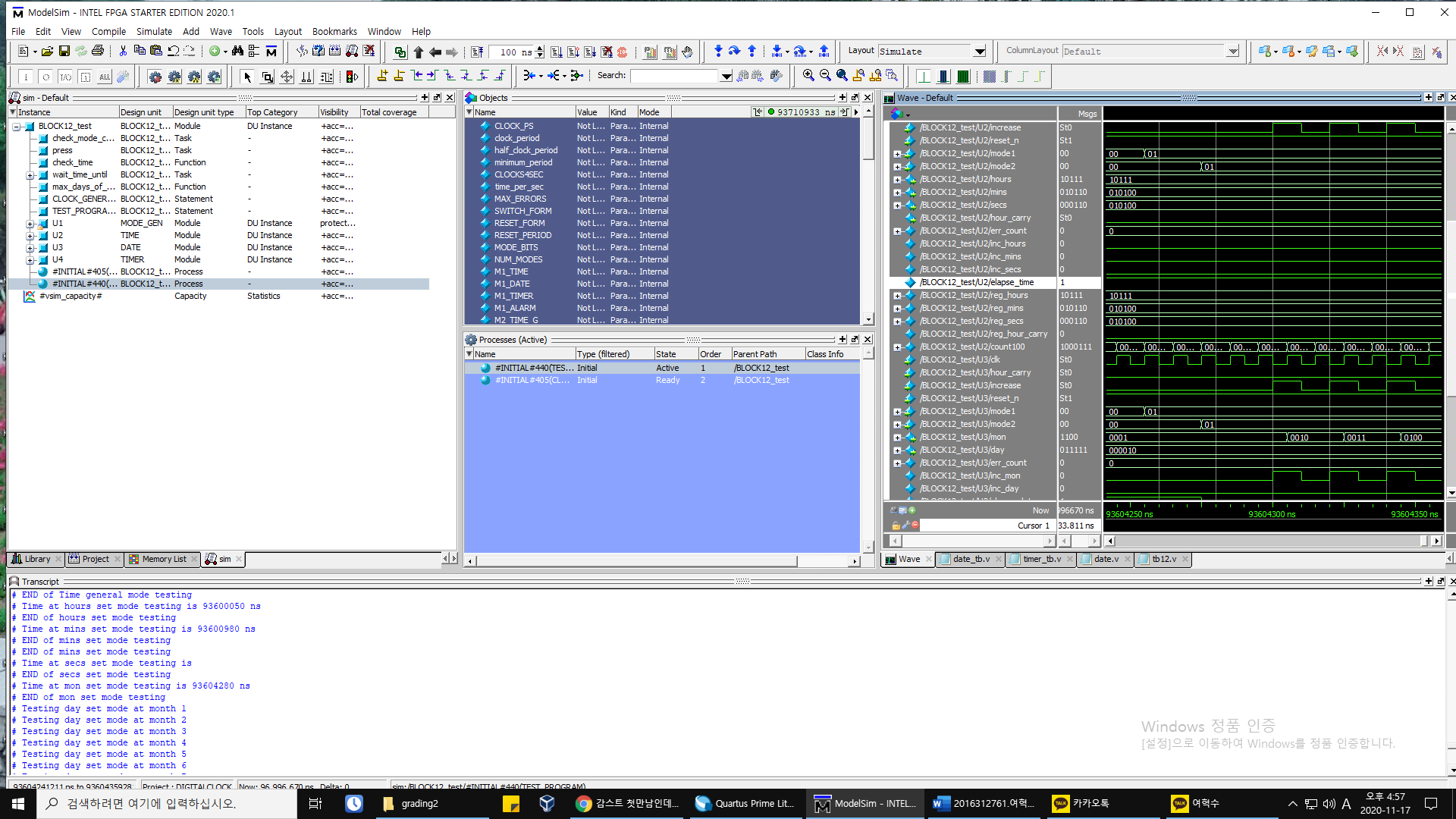
TIME, DATE, TIMER 종합)



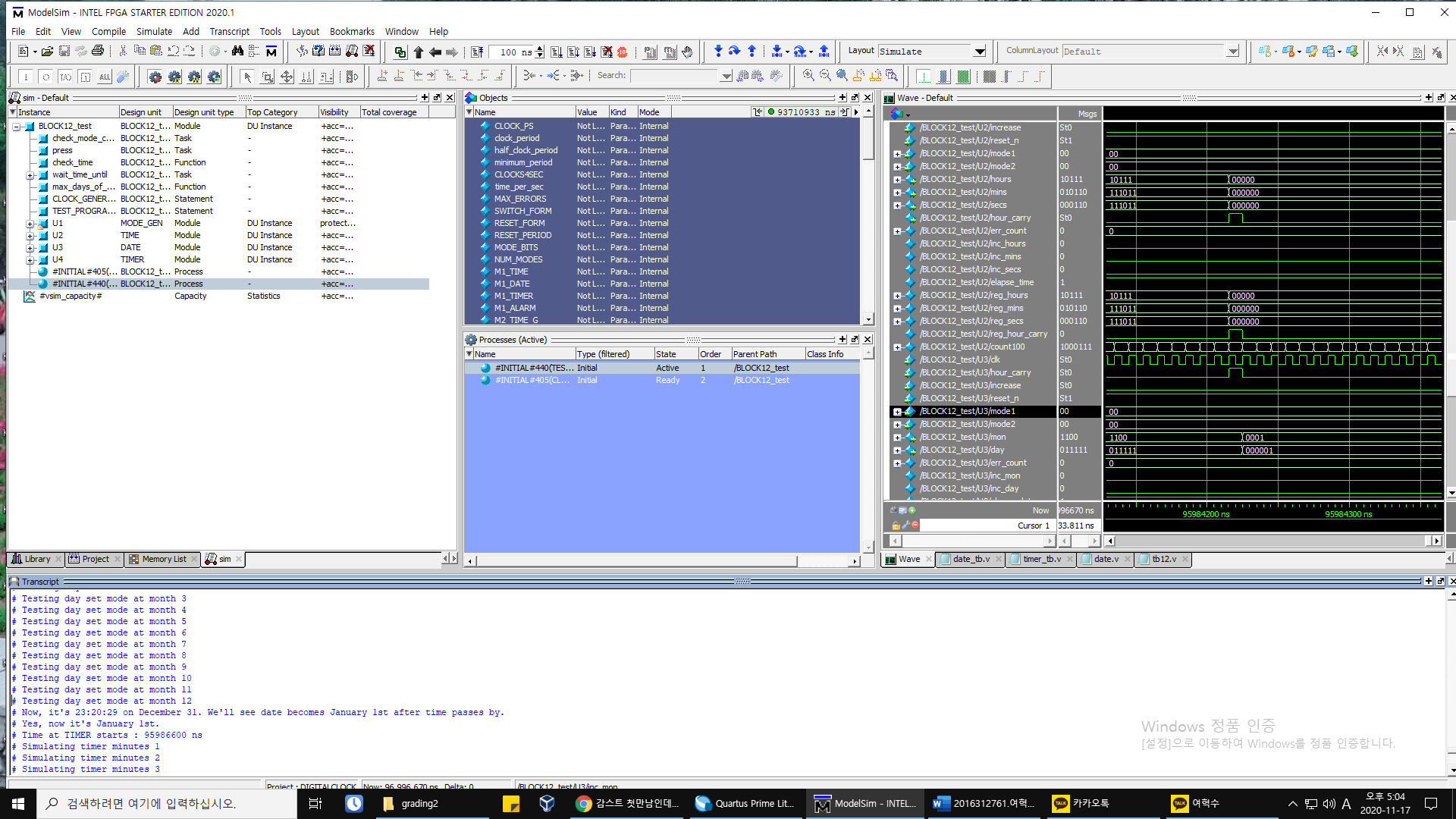
가장 먼저 TIME\_G 모드, 즉 시간이 일반적으로 흐를 때, 총 하루가 지나는 것을 시뮬레이션 하는데, 하루가 넘어갈 때의 파형이다. Mode1, mode2가 모두 0이므로 TIME\_G 모드라는 것을 알 수 있다. U2는 TIME 모듈에서의 변수라는 것을 의미하는데, U2/hours, mins, secs가 각각 0이 되기 직전 값인 23:59:59를 나타내고 있고, count100이 100번의 rising edges를 counting 했을 때 모든 시간 값이 0이 되는 것을 볼 수 있다. 또한 동시에 hour\_carry가 1로 되어 그 값이 U3/day, 즉 DATE 모듈의 day에 전달되어 1월1일에서 1월 2일로 날짜가 업데이트 되는 것을 볼 수 있다.



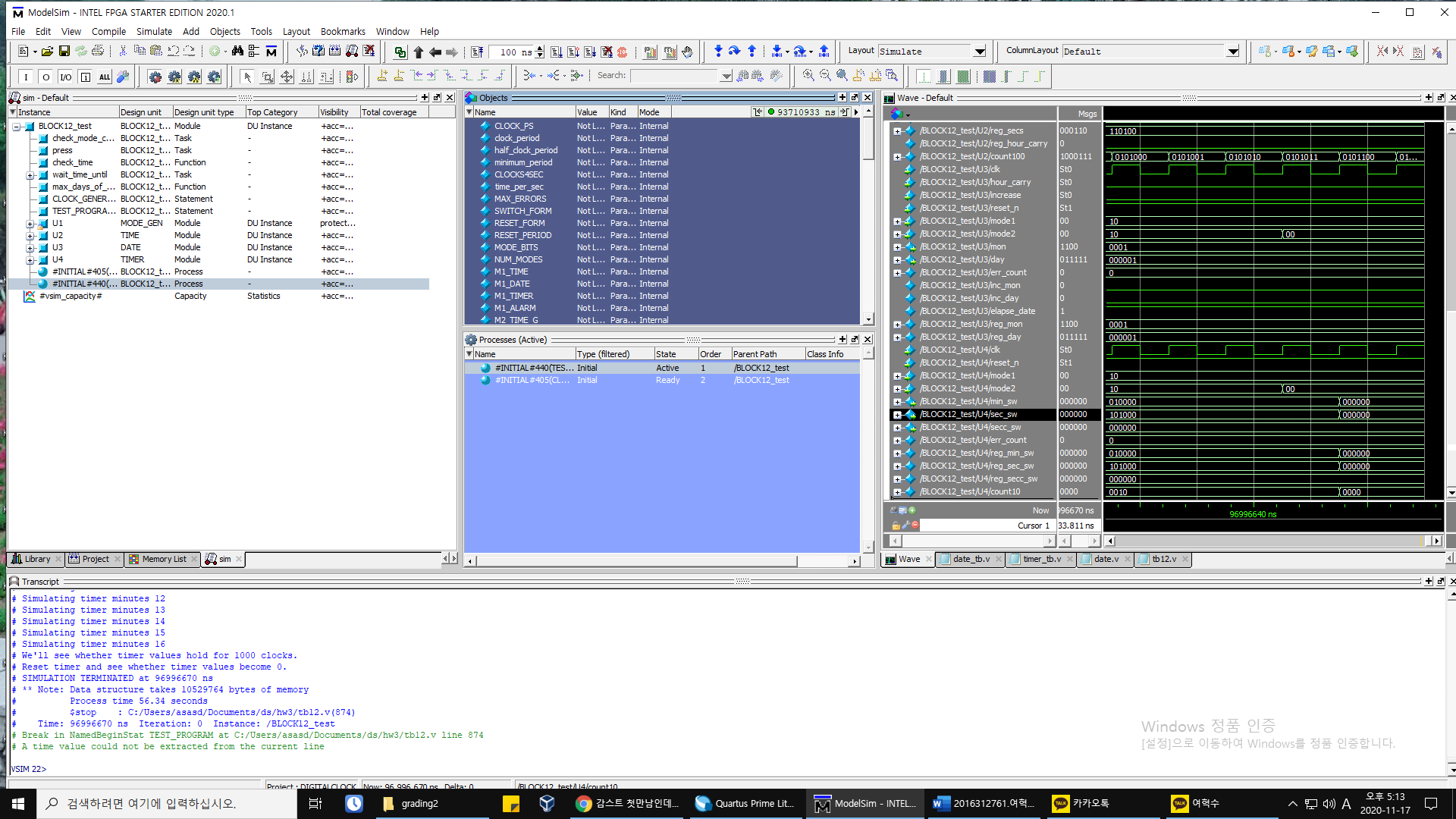
TIME\_G 모드 테스팅이 끝나면 시간을 조정하는 모드인 TIME\_HOUR, TIME\_MIN, TIME\_SEC 모드를 시뮬레이션하는데, TIME\_HOUR 모드일 때를 나타낸 파형이다. Mode2가 2’b01이 되면서 TIME\_HOUR 모드가 된다. 이후 inc\_hours에 1을 주기적으로 주면서 그때마다 다음 rising edge에서 hours 값이 1씩 증가하는 것을 볼 수 있다. 나머지 TIME\_MIN이나 TIME\_SEC도 위와 같이 증가한다.



그 다음으로는 DATE 모드로 가서 날짜를 조정하는 시뮬레이션을 한다. 위 파형을 보면 mode1과 mode2이 각각 2’b00에서 2’b01로 바뀌는 것을 볼 수 있는데, DATE\_MON 모드로 바뀐 것을 의미한다. 모드가 바뀌고 조금 지나 inc\_mon에 주기적으로 1을 주는 것을 볼 수 있다. inc\_mon에 1을 줌으로서 다음 rising edge에서 mon을 증가시키는 것도 볼 수 있다.



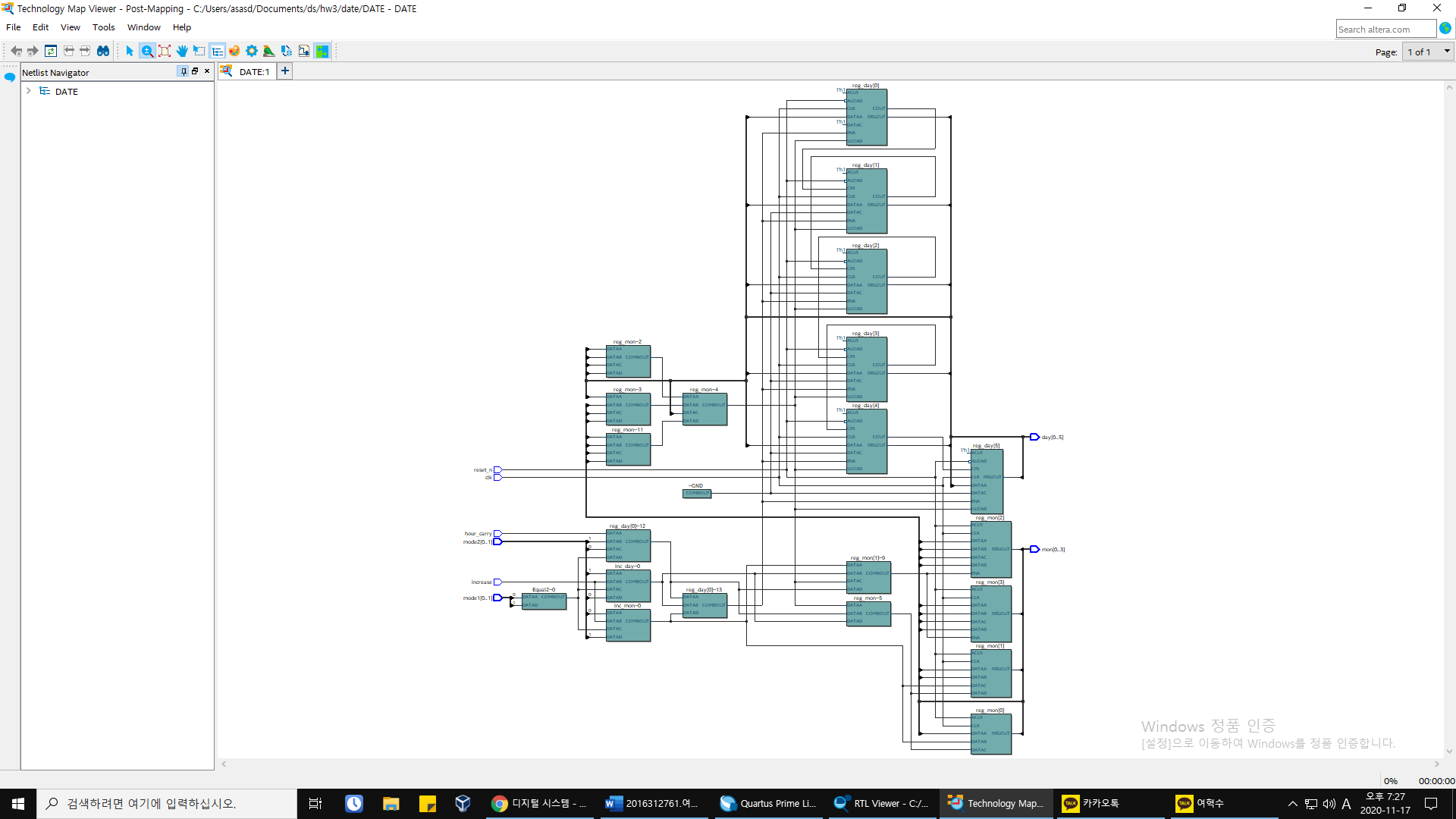
다음으로는 다시 TIME\_G 모드로 돌아가서 시간이 흐르는 것을 가정하고 12월 31일까지 보낸 다음에 1월 1일로 넘어가는지를 시뮬레이션한 것이다. 모든 시간 값은 0이 되면서 날짜는 1월 1일이 되는 것을 볼 수 있다.



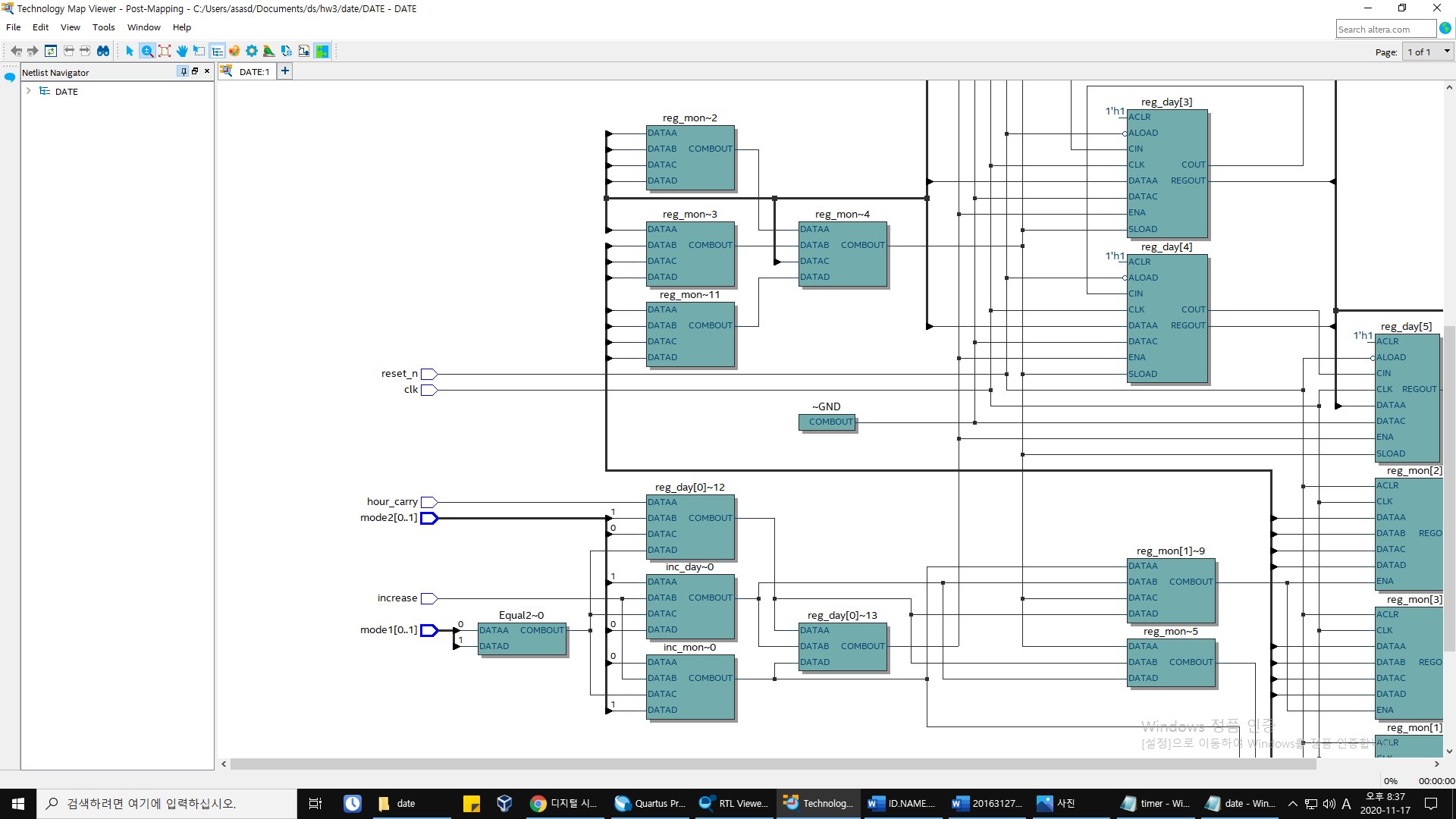
마지막으로 TIMER 모듈로 넘어가는데, 시간 측정을 중지하고, 리셋되는 것을 시뮬레이션 한 것이다. Mode1이 2’b10 이고, mode2가 2’b10 이기 때문에 TIMER\_STOP 모드라고 할 수 있다. 그래서 모든 시간 측정 값이 변하지 않고 그대로 유지되는 것을 볼 수 있다. 또한 mode2가 2’b00으로 바뀌면서 다음 rising edge에서 시간 측정한 값이 0으로 리셋되는 것을 볼 수 있다.

**4) Synthesis Results**

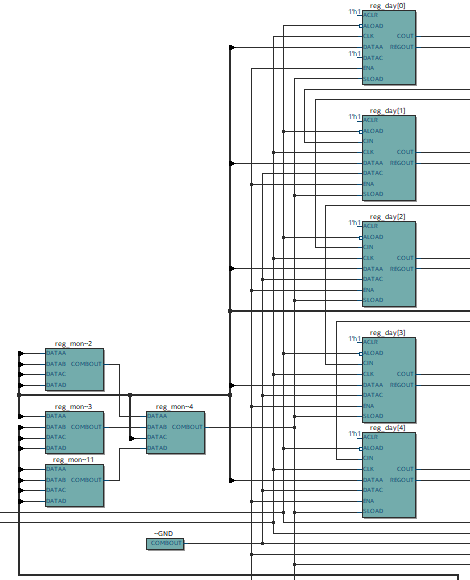
DATE 모듈)



Quartus Prime의 Technology map viewer에서 나온 netlist이다. Mode1,2의 값과 increase 값이 inc\_day, inc\_mon에 영향을 주고, 그것을 통해 day, mon의 비트 자리마다 연산을 해서 day, mon의 아웃풋으로 전달되는 식으로 문제없이 잘 구현된 것 같다.

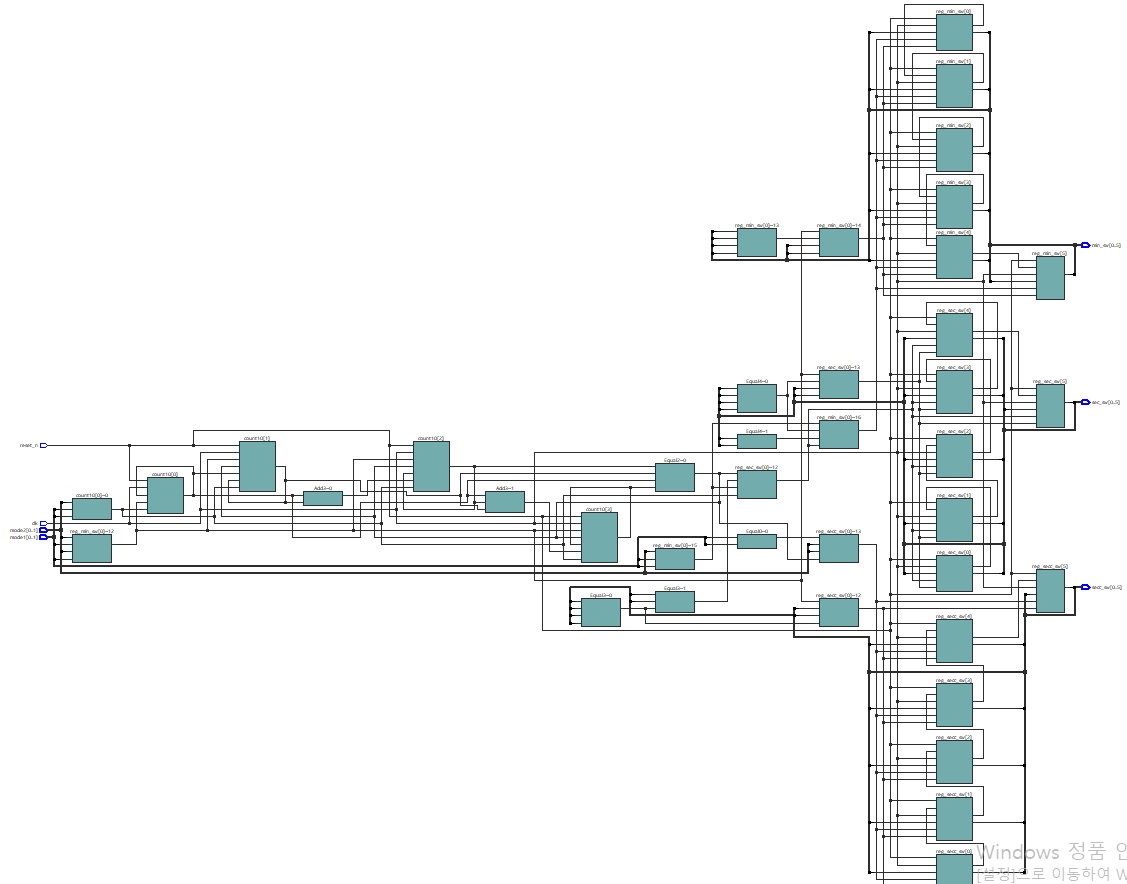


Technology map viewer의 오른쪽 밑에 있는 이 부분이 source code에서 SET\_GEN 블록을 담당하고 있다. SET\_GEN은 inc\_day, inc\_mon을 생성하여 DATE\_GEN에 값을 넘겨주는 역할을 하고 있기 때문이다.

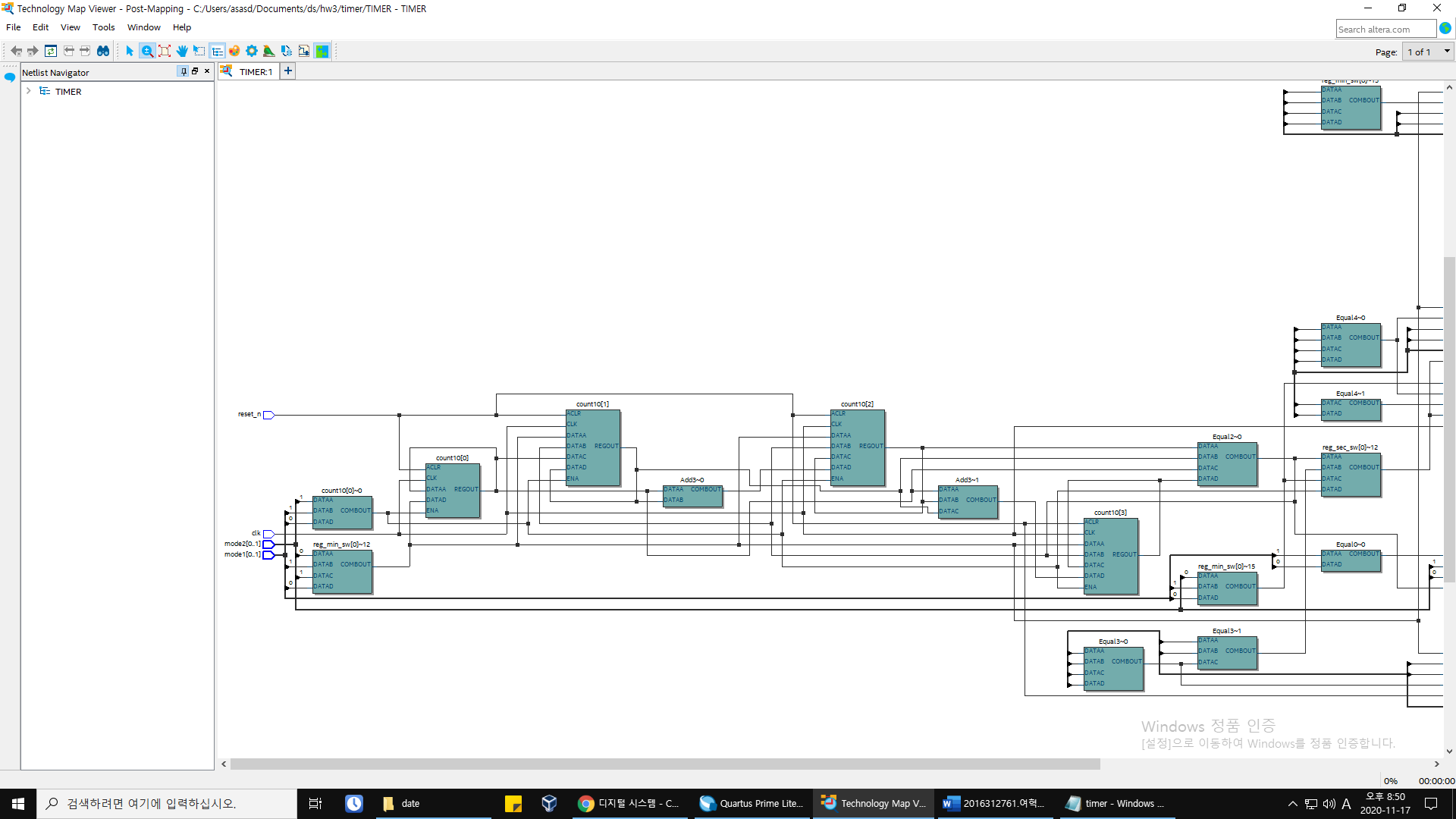


Technology map viewer의 위에 있는 이 부분이 source code에서 DATE\_GEN 블록을 담당하고 있다. DATE\_GEN은 reg\_mon, reg\_day의 값을 연산을 통해 생성하여 아웃풋으로 전달하는 역할을 하는데 바로 이 부분이 연산을 하는 레지스터들이라 할 수 있다.

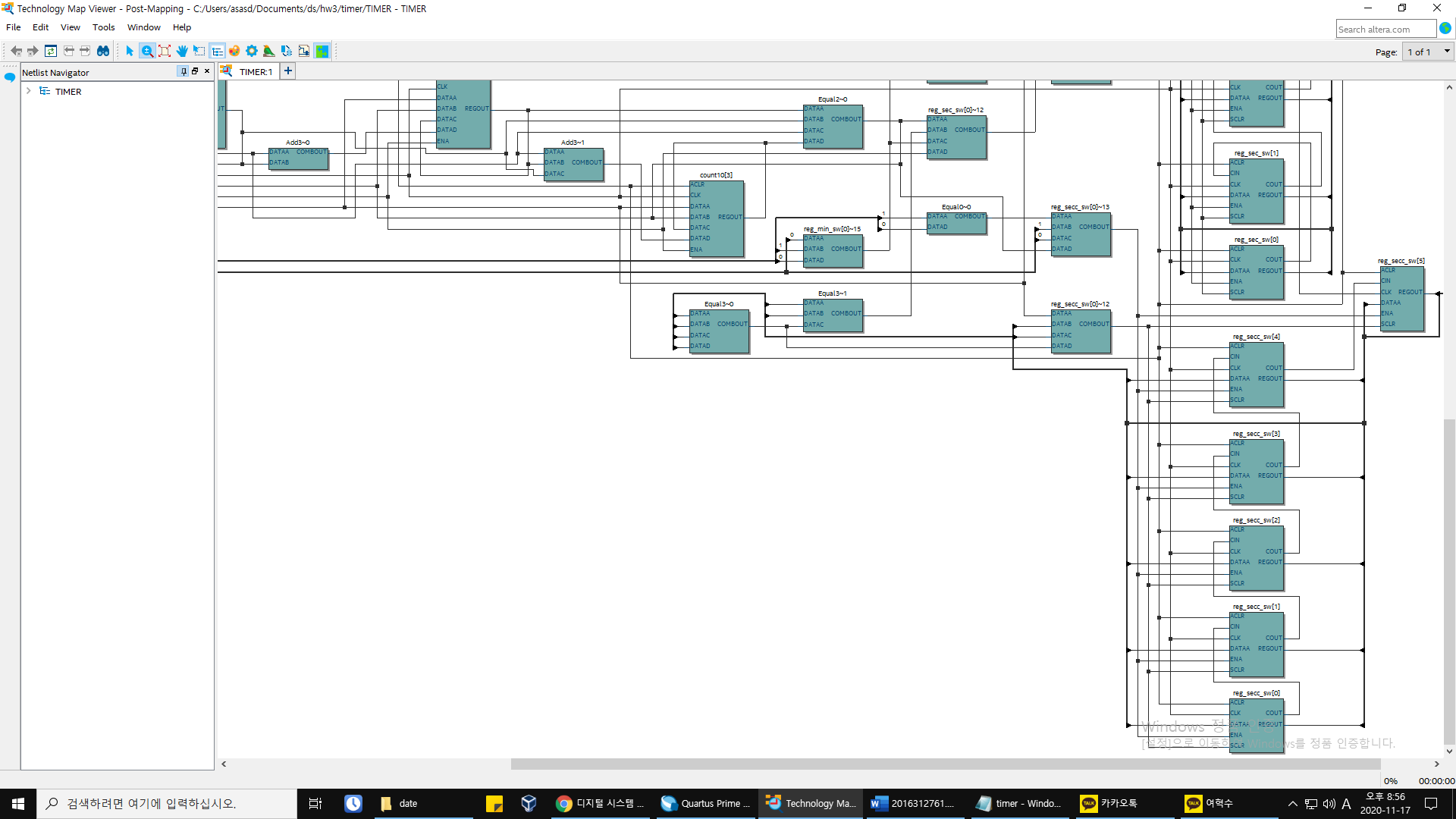
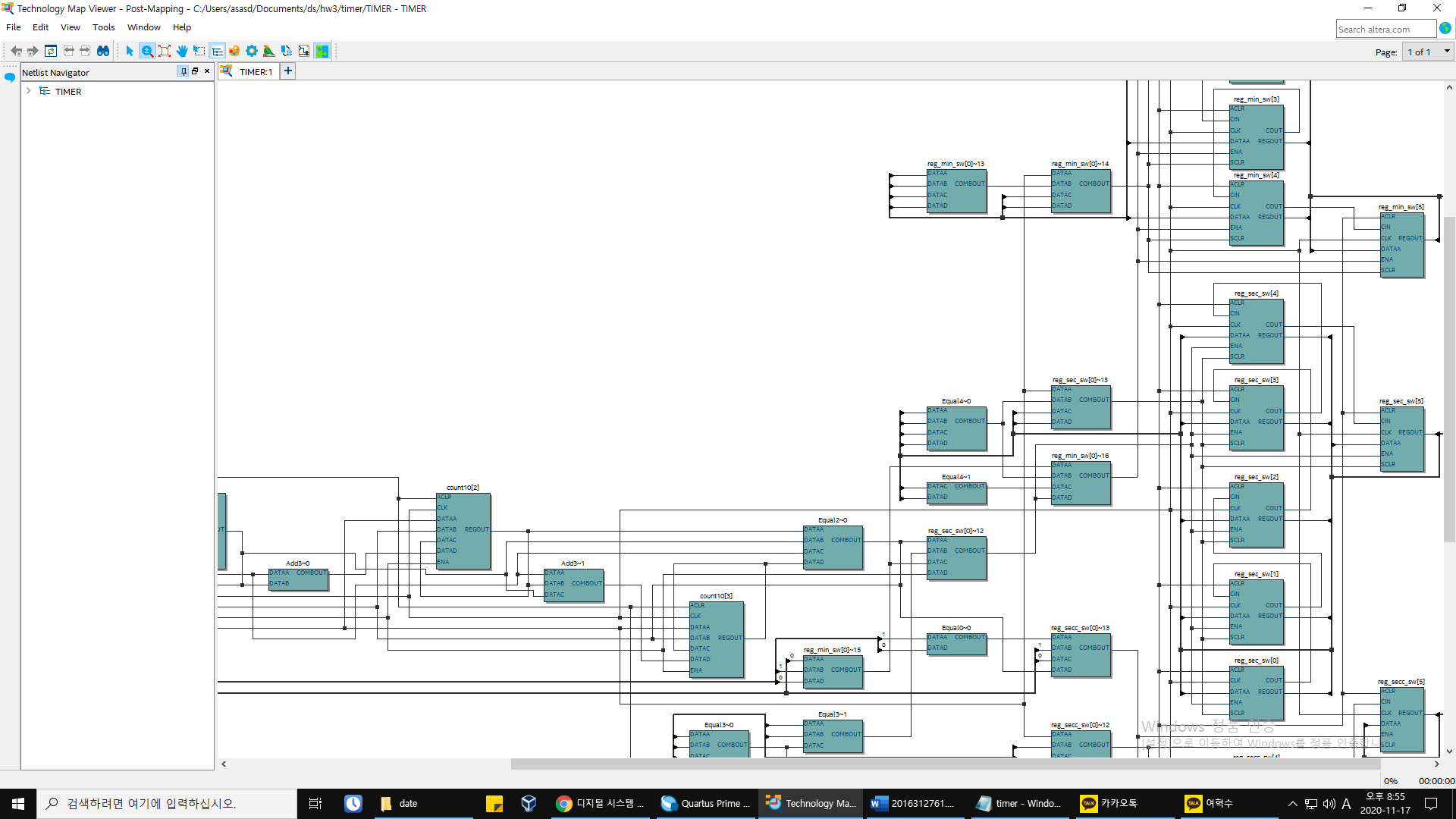
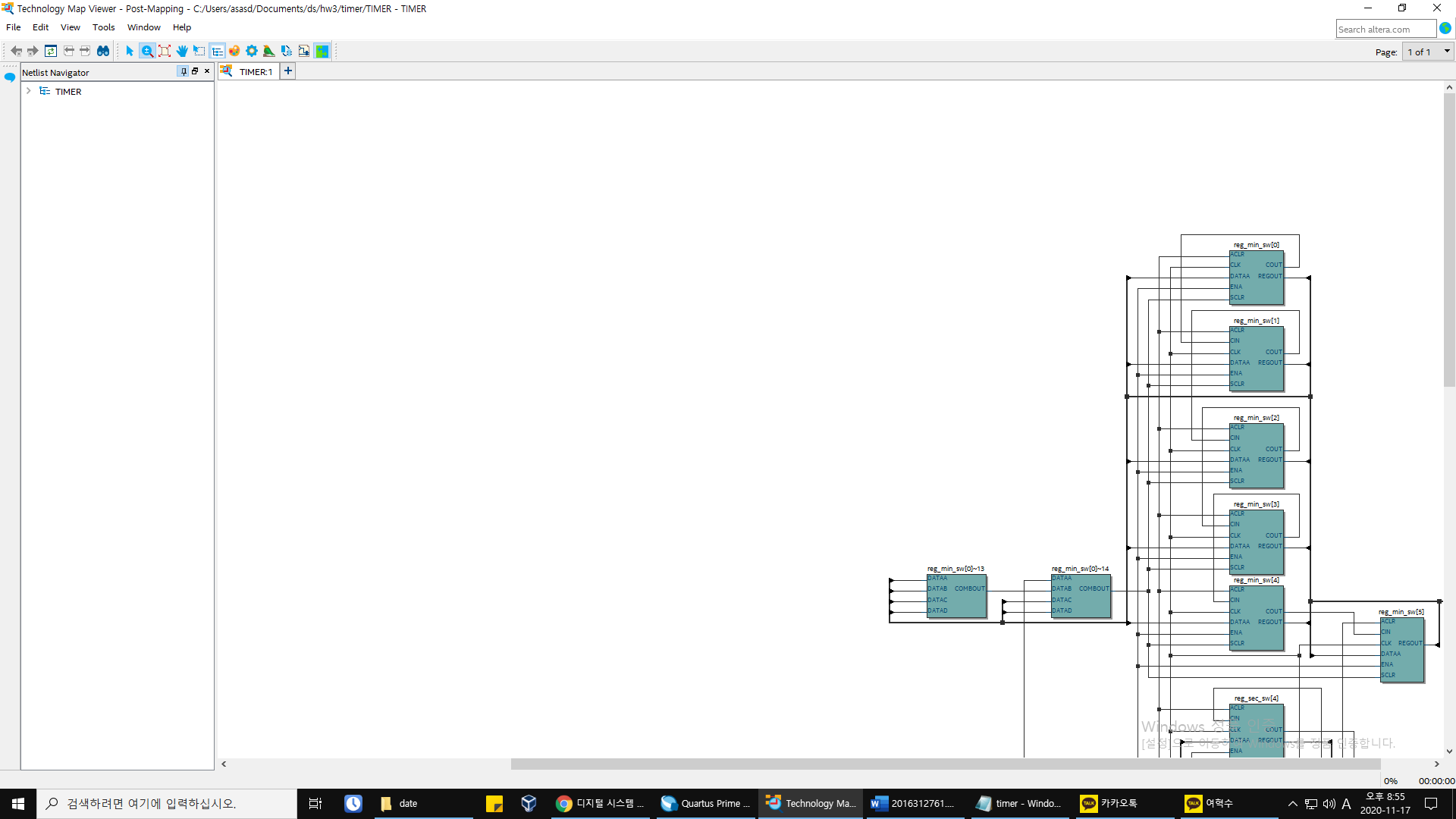
TIMER 모듈)



Quartus Prime의 Technology map viewer에서 나온 netlist이다. 너무 커서 설명하기가 힘들지만 구조적으로 봤을 때 인풋을 받는 부분이 좌측, 시간 측정 관련해서 연산하는 부분이 우측에 세로로 긴 서킷들인 것 같다. 문제 없이 아웃풋으로 값이 전달되게 구현된 것 같다.



TIMER 모듈은 TIMER\_GEN을 이름으로 하는 always 블록 하나로 이루어져 있는데, 이 부분이 그 블록에서 count10이라는 카운터 변수의 연산을 담당하는 레지스터이다. Clk과 연결되어 1씩 더해진다.



netlist에서 파란 동그라미를 친 부분을 3등분해서 보면 다음과 같은 모습인데, 이 부분이 각각 연산된 min, sec, secc의 값을 비트마다 저장하게 되는 레지스터이다.

1. **Conclusion**

이 과제를 통해서 Verilog에서의 function 사용법을 새롭게 알게 되었다. 교수님이 주신 TIME 모듈의 Verilog 코드를 활용하여 나머지 모듈을 쉽게 구현할 수 있었기 때문에 문제 해결에 큰 어려움은 없었다.

1. **참고 문헌**

## [1-2] 민형복, “DigitalClock3.pdf,” [Online]. Available: <http://class.icc.skku.ac.kr/~min/di/> . [Accessed Nov. 11, 2020].

## [3] Modelsim. (FPGA), Intel. [Accessed Nov. 11, 2020]. Available: <https://fpgasoftware.intel.com/20.1/?edition=lite>

## [4] Intel Quartus Prime. (20.1), Intel. [Accessed Nov. 11, 2020]. Available: <https://fpgasoftware.intel.com/20.1/?edition=lite>