

# ModelSim 기초 매뉴얼

디지털시스템 2015-1

# ModelSim-Altera Edition 다운로드

- [http://dl.altera.com/?edition=subscription&platform=windows&download\\_manager=direct](http://dl.altera.com/?edition=subscription&platform=windows&download_manager=direct)
- Altera 홈페이지 회원가입이 필요하다.
- ModelSim-Altera Edition (includes Starter Edition) 라고 써진 것을 다운받는다.
  - 설치파일 크기 약 1.1 GB, 설치공간 크기 약 3.7GB

# ModelSim-Altera Edition 다운로드

Combined Files Individual Files DVD Files Additional Software Updates

Download and install instructions: [More](#)

[Read Altera Software v14.1 Installation FAQ](#)

[Quick Start Guide](#)

**Quartus II Subscription Edition**

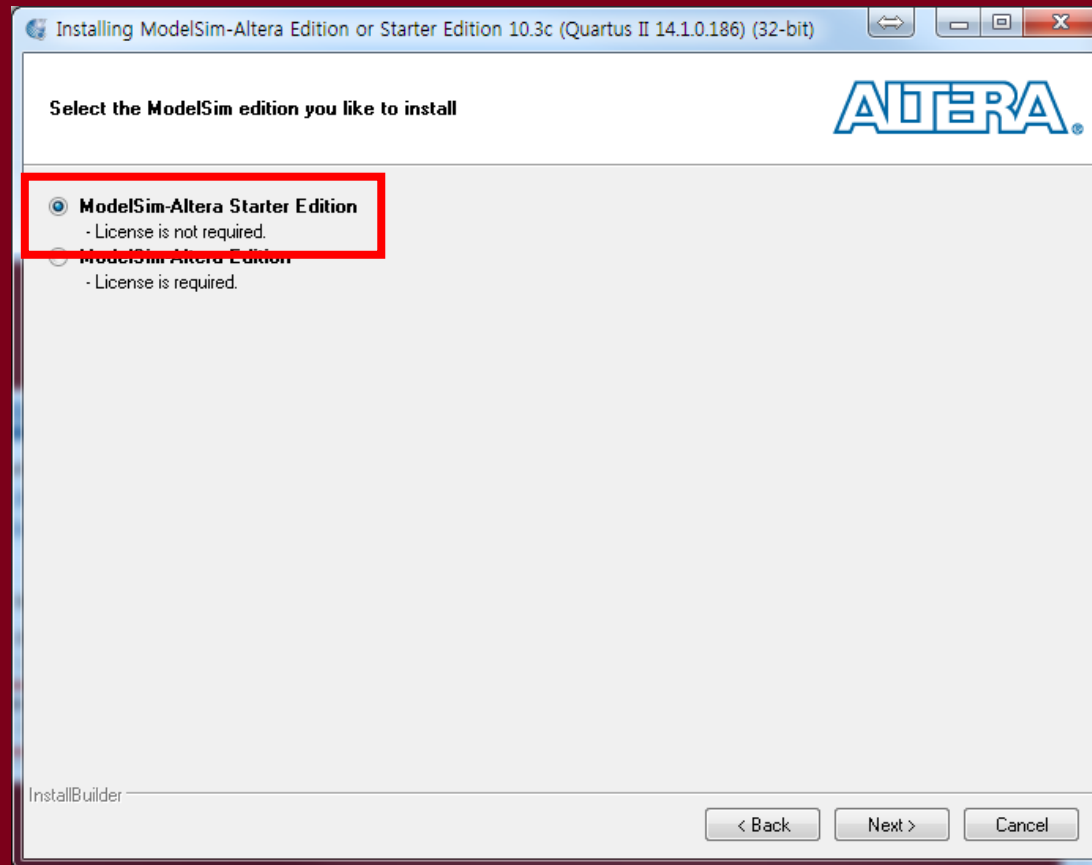
**Quartus II Software (includes Nios II EDS)**  
Size: 1.7 GB MD5: 386032926BB96993E25FA578FDC5F744

**ModelSim-Altera Edition (includes Starter Edition)**  
Size: 1.1 GB MD5: C931A4F7F9B4306DD8E8248607993C7C

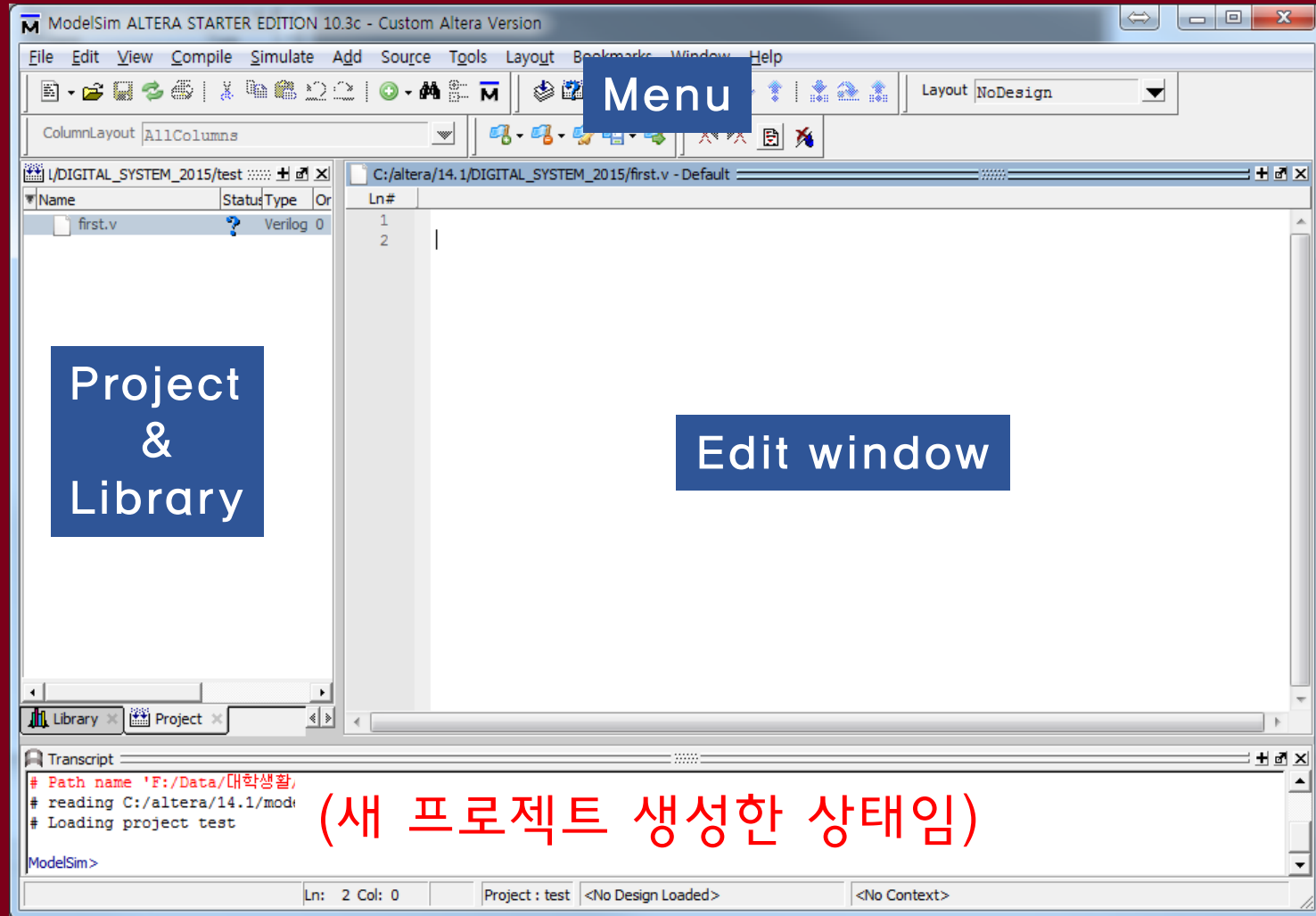
UPDATE

# ModelSim-Altera Starter Edition 설치

- 라이선스가 없어도 되는 Starter Edition으로 설치한다.  
(제한: 코드 라인 10000줄 이하, 약간의 속도 저하)

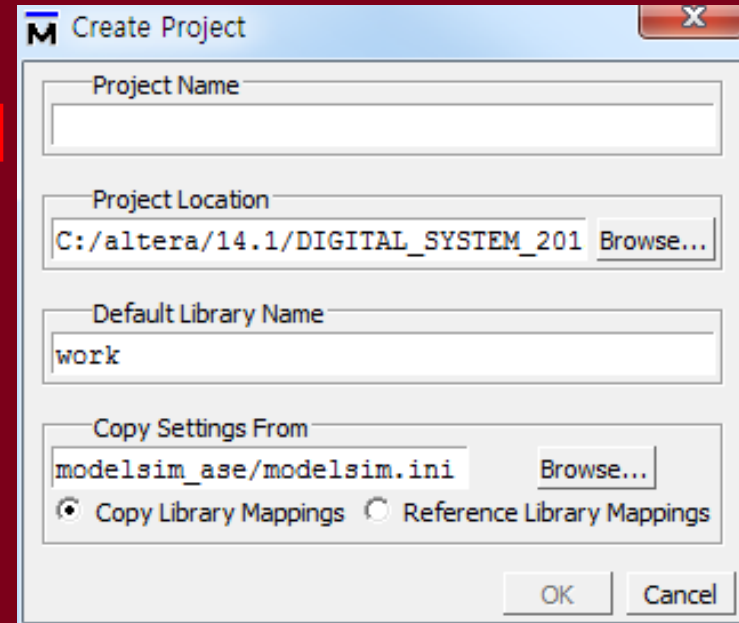
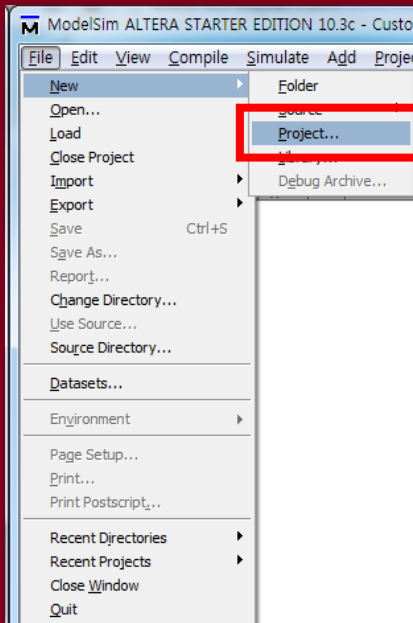
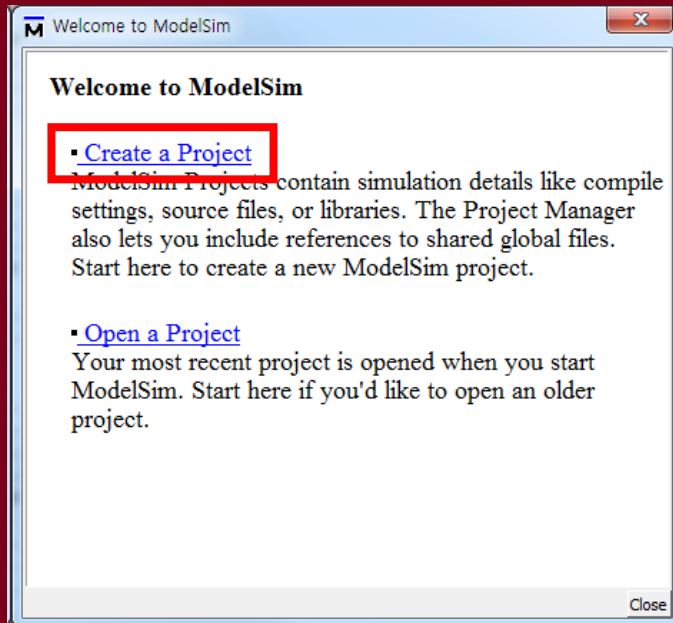


# ModelSim Interface



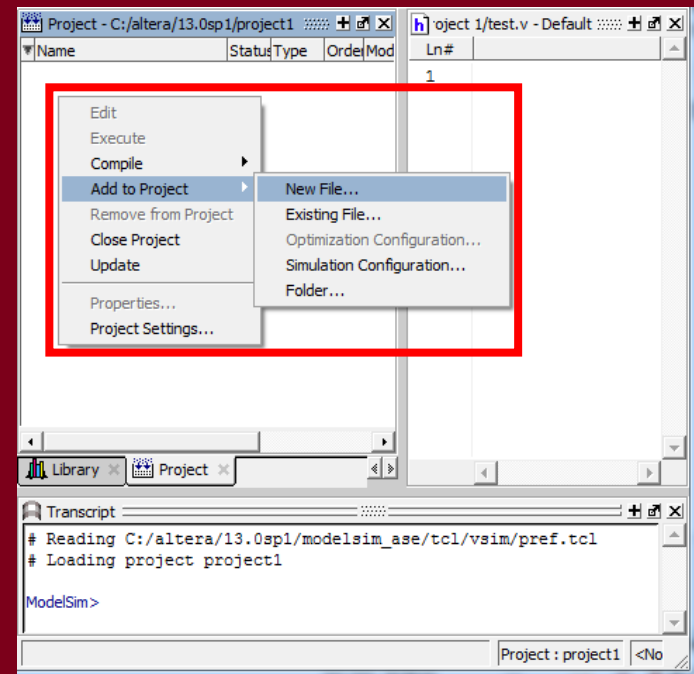
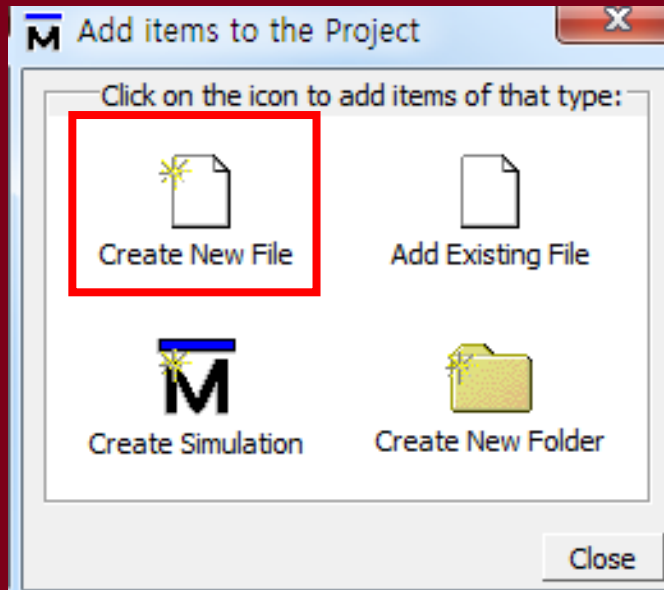
# 새로운 프로젝트 생성

- ModelSim 소프트웨어 실행 후 Jumpstart 메뉴에서 Create a Project를 선택하거나, 메뉴 > File > New > Project... 선택하여 새 프로젝트를 만든다.



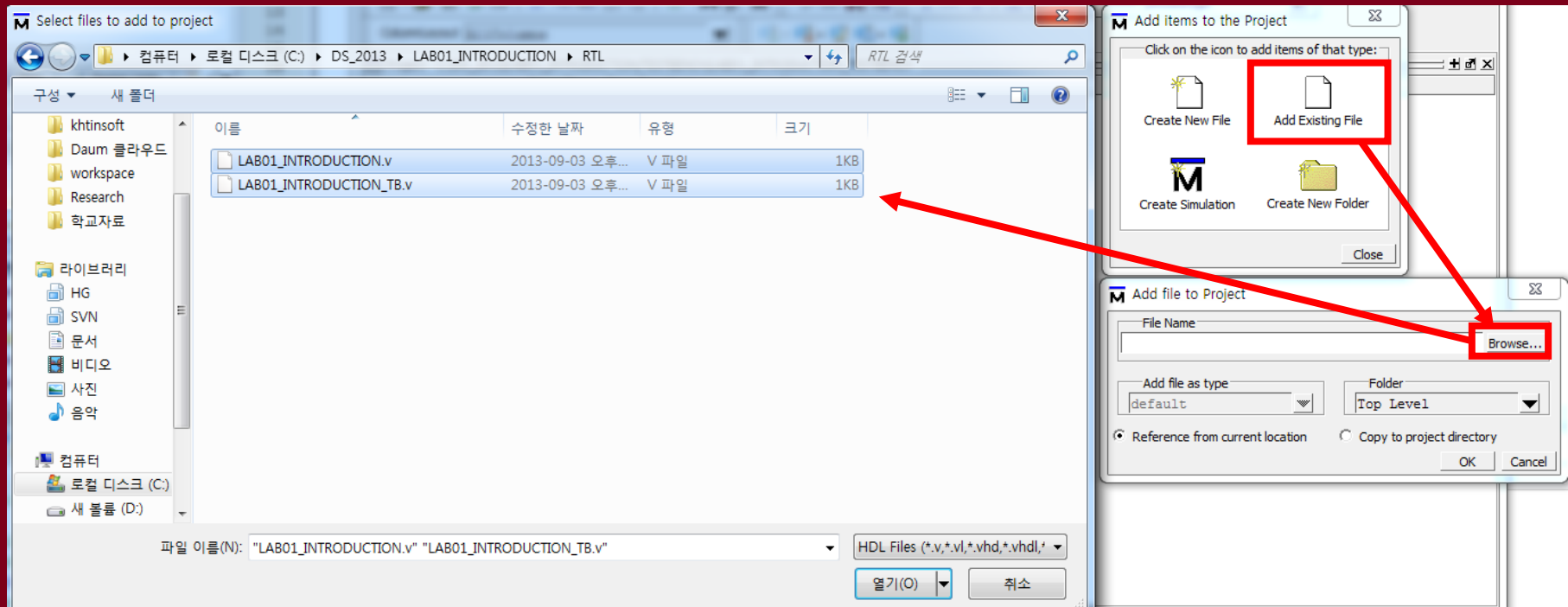
# 프로젝트에 새 소스코드 추가

- Add items to the Project 창에서 Create New File 을 선택한다.
- 또는 Project 탭에서 오른쪽 클릭을 해서 Add to Project > New File... 을 선택한다.
  - 새 파일을 만들 때는 Verilog 파일(.v)을 선택한다. (VHDL ≠ Verilog !!)



# 프로젝트에 기존 소스코드 추가

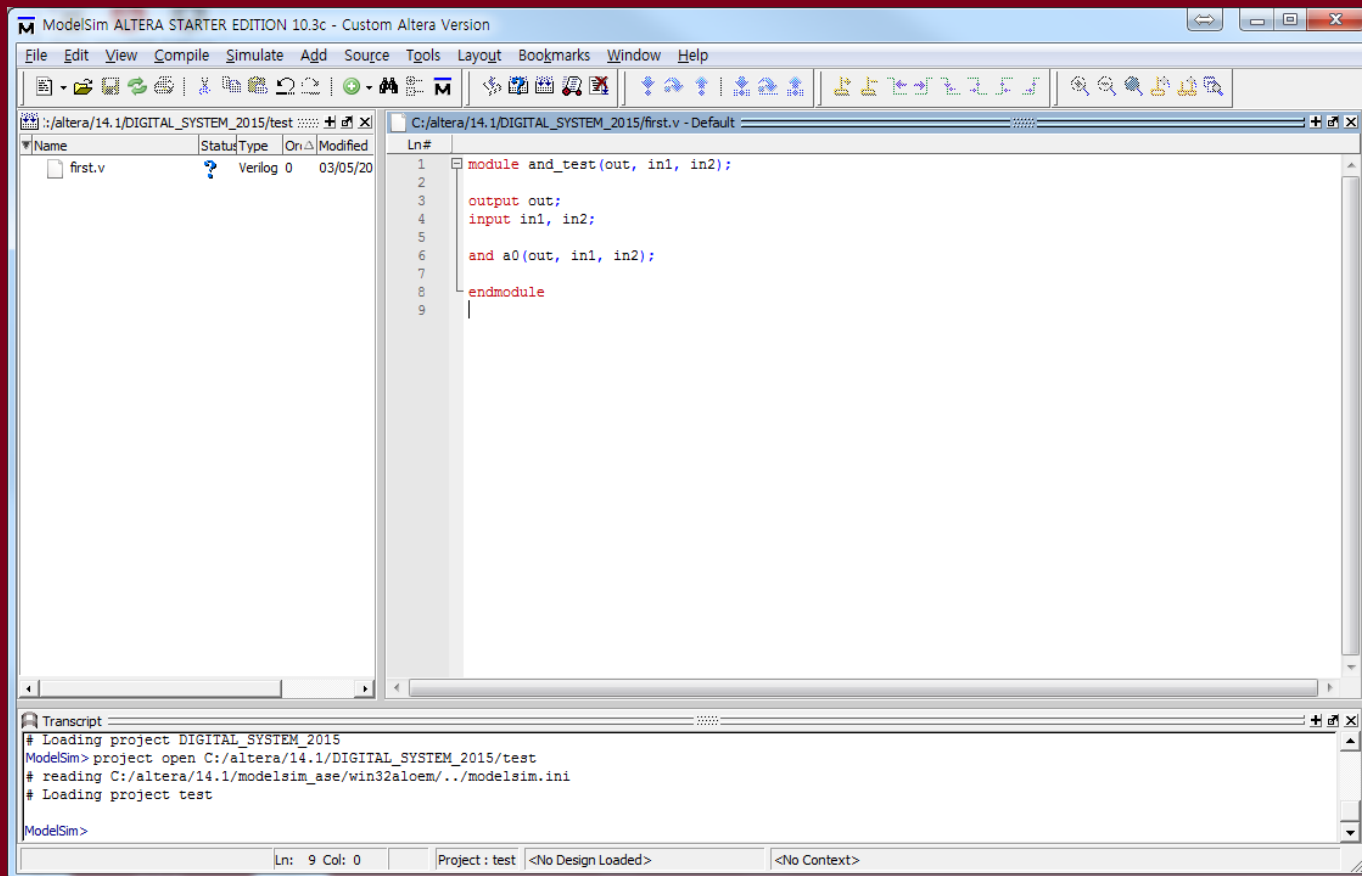
- Add items to the Project 창에서 Add Existing File을 선택한다.
- 또는 Project 탭에서 오른쪽 클릭을 해서 Add to Project > Existing File... 을 선택한다.





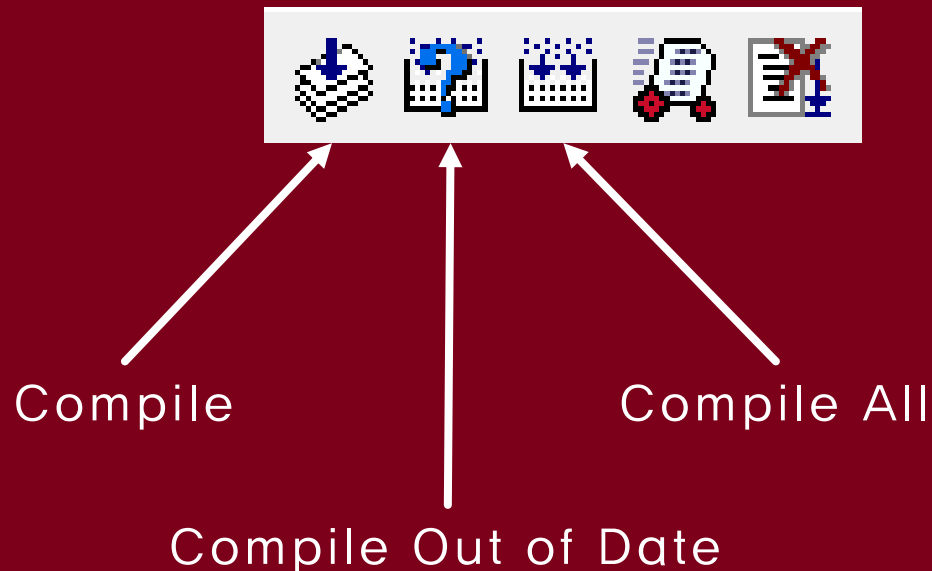
# 생성한 소스코드 파일 수정

- Edit window에서 Verilog 문법에 맞게 소스코드를 작성한다.



# 소스 코드 컴파일 및 에러 메시지

- 소스코드 컴파일을 통해 문법적 에러를 찾아낼 수 있고 시뮬레이션을 통해 논리적 에러를 찾아낼 수 있다.
- 메뉴 > Compile > Compile All 선택하면 프로젝트에 추가된 모든 소스코드를 컴파일하고, 추후에 수정된 코드만 컴파일 하고 싶을 때는 Compile Selected 등을 선택할 수 있다.



<http://compiler.korea.ac.kr>

# 소스 코드 컴파일 및 에러 메시지

- 소스코드 상에 문제가 없다면 Compile했을 때 아래쪽 Transcript 창에 초록색으로 메시지가 뜬다.

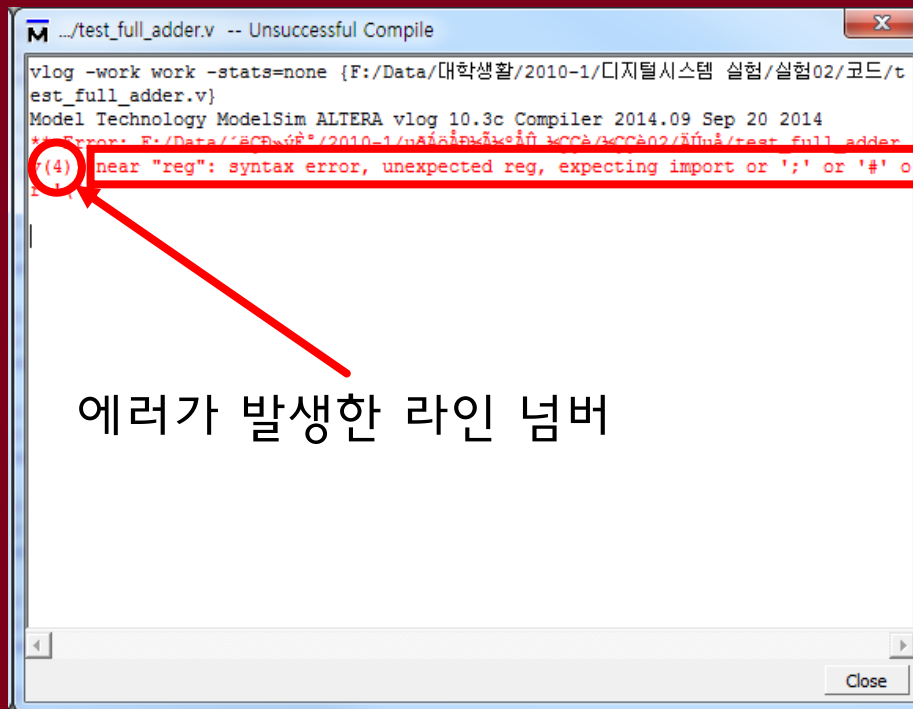
```
# Compile of test_full_adder.v was successful.  
# Compile of full_adder.v was successful.  
# 2 compiles, 0 failed with no errors.
```

- 소스코드 상에 에러가 있다면 (임의로 세미콜론 하나를 삭제한 후 Compile) 빨간색으로 에러 메시지가 뜬다.

```
# Compile of test_full_adder.v failed with 1 errors.  
# Compile of full_adder.v was successful.  
# 2 compiles, 1 failed with 1 error.
```

# 소스 코드 컴파일 및 에러 메시지

- 빨간색 에러 메시지를 더블 클릭하면 소스코드의 어떤 부분에서 에러가 났는지에 대해 힌트를 얻을 수 있다.



```
M .../test_full_adder.v -- Unsuccessful Compile
vlog -work work -stats=none {F:/Data/대학생활/2010-1/디지털시스템 실험/실험02/코드/t
est_full_adder.v}
Model Technology ModelSim ALTERA vlog 10.3c Compiler 2014.09 Sep 20 2014
**Error: F:/Data/대학생활/2010-1/디지털시스템 실험/실험02/코드/test_full_adder
(4) near "reg": syntax error, unexpected reg, expecting import or ';' or '#' o

```

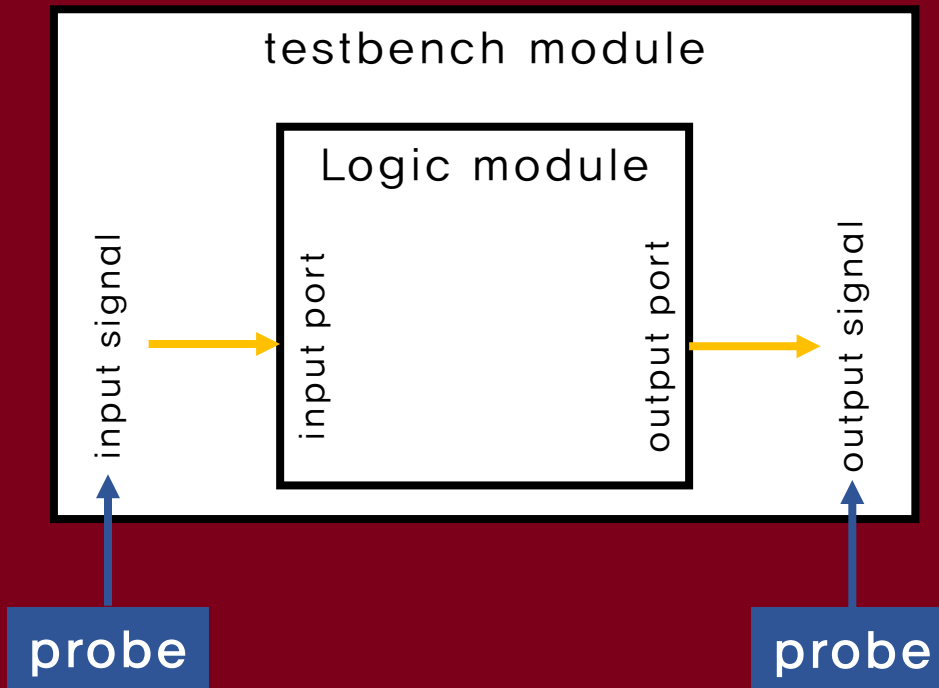
에러가 발생한 라인 넘버

어떤 종류의 에러가 발생했는지  
서술하고 있다.

4번째 줄 “reg”라는 단어에서  
문법 에러가 발생했고, ‘;’이나  
‘#’을 삽입할 것을 권하고 있다.

# Testbench

- testbench는 코딩한 로직 모듈의 동작을 확인해보기 위한 것으로, 신호를 생성하고 다른 모듈에 공급할 수 있는 또 다른 verilog 모듈이다.



# Testbench 작성 예시

```
C:/altera/14.1/DIGITAL_SYSTEM_2015/tb_first.v - Default ==
Ln#
1
2    `timescale 100ps/1ps
3
4    module tb_and_test;
5        parameter test = 8;
6        wire out;
7        reg in1, in2;
8
9        and a0(out, in1, in2);
10
11        initial
12        begin
13            in1 <= 0;
14            in2 <= 0;
15
16            #5
17            in1 <= 1;
18
19            #5
20            in2 <= 1;
21
22            #5
23            in1 <= 0;
24
25            #5
26            in2 <= 0;
27        end
28
29    endmodule
30
```

왼쪽 모듈은 AND gate의 동작을 확인하기 위한 테스트 벤치이다.

이 테스트벤치는 a0라는 이름이 붙은 AND gate하나를 생성하여 in1, in2 시그널을 input으로 넣어서 out 시그널이 output으로 나오게 된다.

in1, in2의 값은 (0, 0) → (1, 0) → (1, 1) → (0, 1) → (0, 0)의 값으로 변화한 후 이후 같은 값을 계속 유지한다.

자세한 코드 설명은 뒤쪽에서 이어 나간다.

# Testbench 작성 예시

```
C:/altera/14.1/DIGITAL_SYSTEM_2015/tb_first.v - Default =
Ln#
1
2 `timescale 100ps/1ps
3
4 module tb_and_test;
5     parameter test = 8;
6     wire out;
7     reg in1, in2;
8
9     and a0(out, in1, in2);
10
11     initial
12     begin
13         in1 <= 0;
14         in2 <= 0;
15
16         #5
17         in1 <= 1;
18
19         #5
20         in2 <= 1;
21
22         #5
23         in1 <= 0;
24
25         #5
26         in2 <= 0;
27     end
28
29 endmodule
30
```

``timescale 100ps/1ps`

작성한 모듈의 시간 단위를 정의한다.  
주의: `은 따옴표가 아니라 숫자 1 옆  
의 문자이다.

사용 방법:

``timescale` 시간단위/오차범위

#5

정의된 시간 단위의 배수로 시간 흐  
름을 정의한다. #5는 현재 시간 단위  
가 100ps이므로 500ps의 흐름을  
나타낸다.

사용 방법:

#k

# Testbench 작성 예시

```
C:/altera/14.1/DIGITAL_SYSTEM_2015/tb_first.v - Default =
Ln#
1
2 `timescale 100ps/1ps
3
4 module tb_and_test;
5     parameter test = 8;
6     wire out;
7     reg in1, in2;
8
9     and a0(out, in1, in2);
10
11     initial
12     begin
13         in1 <= 0;
14         in2 <= 0;
15
16         #5
17         in1 <= 1;
18
19         #5
20         in2 <= 1;
21
22         #5
23         in1 <= 0;
24
25         #5
26         in2 <= 0;
27     end
28
29 endmodule
30
```

## module

사용할 모듈 이름을 정의한다. 테스트벤치는 독립적인 모듈이므로 포트 없이 정의할 수 있다.

## wire, reg

테스트벤치에서 reg는 input port로 값을 넣을 시그널, wire는 output port에서 로직 결과를 확인할 시그널로 지정한다.

## parameter

모듈 내에서 값이 변하지 않을 상수를 정의할 수 있다.



# Testbench 작성 예시

```
C:/altera/14.1/DIGITAL_SYSTEM_2015/tb_first.v - Default =
Ln#
1
2    `timescale 100ps/1ps
3
4    module tb_and_test;
5        parameter test = 8;
6        wire out;
7        reg in1, in2;
8
9        and a0(out, in1, in2);
10
11    initial
12    begin
13        in1 <= 0;
14        in2 <= 0;
15
16        #5
17        in1 <= 1;
18
19        #5
20        in2 <= 1;
21
22        #5
23        in1 <= 0;
24
25        #5
26        in2 <= 0;
27    end
28
29    endmodule
30
```

`module_name instance_name(port, ...);`

이미 정의된 로직 모듈의 인스턴스를 만든다.

ModelSim 기본 라이브러리에 정의된 and 모듈을 불러와서 a0라는 인스턴스명을 붙이고 and 모듈이 요구하는 (output, input0, input1) 포트에 각각 out, in1, in2 시그널을 넣어준 것이다.

주의: 시그널을 넣을 때 임의의 순서대로 넣으면 안되고 로직 모듈을 설계할 때의 포트 순서를 그대로 지켜줘야 한다.

# Testbench 작성 예시

```
C:/altera/14.1/DIGITAL_SYSTEM_2015/tb_first.v - Default =
Ln#
1
2   `timescale 100ps/1ps
3
4   module tb_and_test;
5       parameter test = 8;
6       wire out;
7       reg in1, in2;
8
9       and a0(out, in1, in2);
10
11   initial
12   begin
13       in1 <= 0;
14       in2 <= 0;
15
16       #5
17       in1 <= 1;
18
19       #5
20       in2 <= 1;
21
22       #5
23       in1 <= 0;
24
25       #5
26       in2 <= 0;
27   end
28
29   endmodule
30
```

## initial

모듈을 실행할 때 처음 한 번만 실행되는 코드 블록을 정의한다. 보통 시그널의 초기화에 사용한다.

cf. always 내부에 정의된 블록은 시뮬레이션 하는 도중 loop를 돌면서 계속 실행되어야 하는 구간을 정의한다.

참고자료: <http://www.asic-world.com/verilog/vbehave1.html>

# Testbench 작성 예시

```
C:/altera/14.1/DIGITAL_SYSTEM_2015/tb_first.v - Default =
Ln#
1
2   `timescale 100ps/1ps
3
4   module tb_and_test;
5       parameter test = 8;
6       wire out;
7       reg in1, in2;
8
9       and a0(out, in1, in2);
10
11      initial
12  begin
13      in1 <= 0;
14      in2 <= 0;
15
16      #5
17      in1 <= 1;
18
19      #5
20      in2 <= 1;
21
22      #5
23      in1 <= 0;
24
25      #5
26      in2 <= 0;
27  end
28
29  endmodule
30
```

## begin - end

다른 프로그래밍 언어에서의 중괄호 ({})와 같은 역할을 한다. 즉 코드 블록의 범위를 지정한다.

블락 안에 한 줄 이상의 코드가 들어가는 경우 반드시 사용해야 한다.

이 코드에서는 begin - end 사이 코드가 그 위에 있는 initial 블록에 포함된다는 것을 나타낸다.

# Testbench 작성 예시

```
C:/altera/14.1/DIGITAL_SYSTEM_2015/tb_first.v - Default =
Ln#
1
2    `timescale 100ps/1ps
3
4    module tb_and_test;
5        parameter test = 8;
6        wire out;
7        reg in1, in2;
8
9        and a0(out, in1, in2);
10
11        initial
12        begin
13            in1 <= 0;
14            in2 <= 0;
15
16            #5
17            in1 <= 1;
18
19            #5
20            in2 <= 1;
21
22            #5
23            in1 <= 0;
24
25            #5
26            in2 <= 0;
27        end
28
29    endmodule
30
```


`input <= value;`

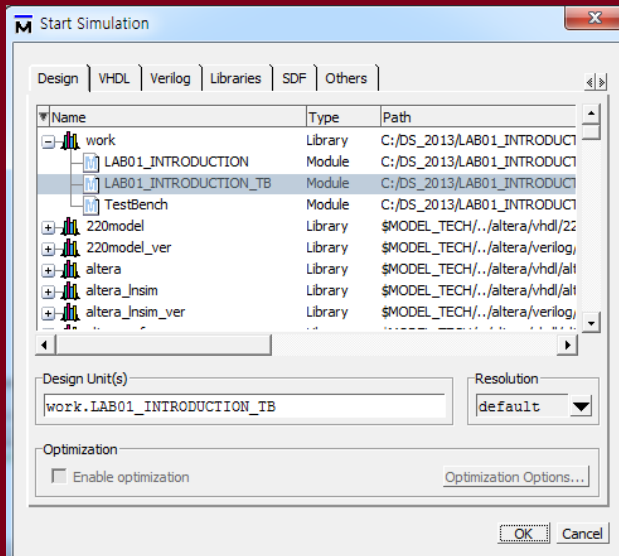
input에 값을 할당한다.

= (blocking assignment) 로 할당하는 것과 <= (non-blocking assignment) 로 할당하는 것의 결과가 다를 수 있으니 주의한다.

참고자료: [ext. link](#) p.6-7

# ModelSim을 이용한 시뮬레이션

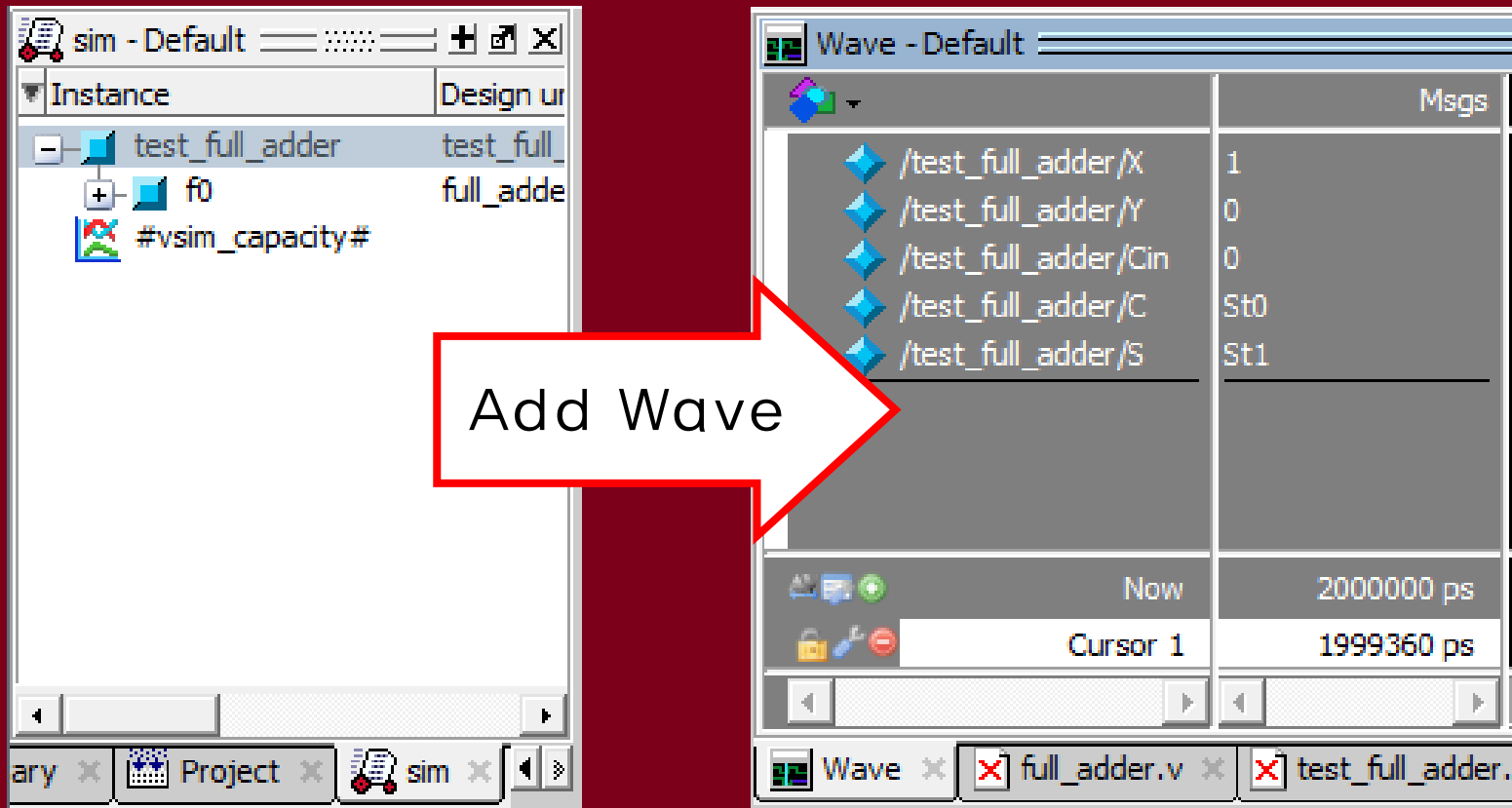
- 메뉴 > Simulate > Start Simulation 선택하거나 Compile All 버튼 옆의  버튼을 눌러 Start Simulation 창을 연다.
- Design 탭에서 work 하단의 testbench 모듈 이름을 선택하고 OK를 누른다.



- 프로젝트 생성할 때 Default Library Name을 work로 지정하지 않았다면 다른 라이브러리 하단에 생성되었을 수 있다.
- 테스트벤치도 Verilog 모듈이므로 컴파일해야 볼 수 있다.

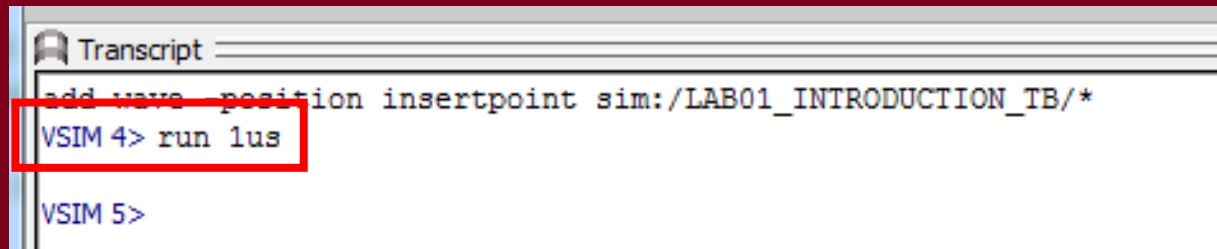
# ModelSim을 이용한 시뮬레이션

- Testbench의 인스턴스 선택 후 오른쪽 클릭하여 Add Wave 항목 선택해서 Wave 창에 파형을 띄울 수 있다. (단축키 Ctrl+W)

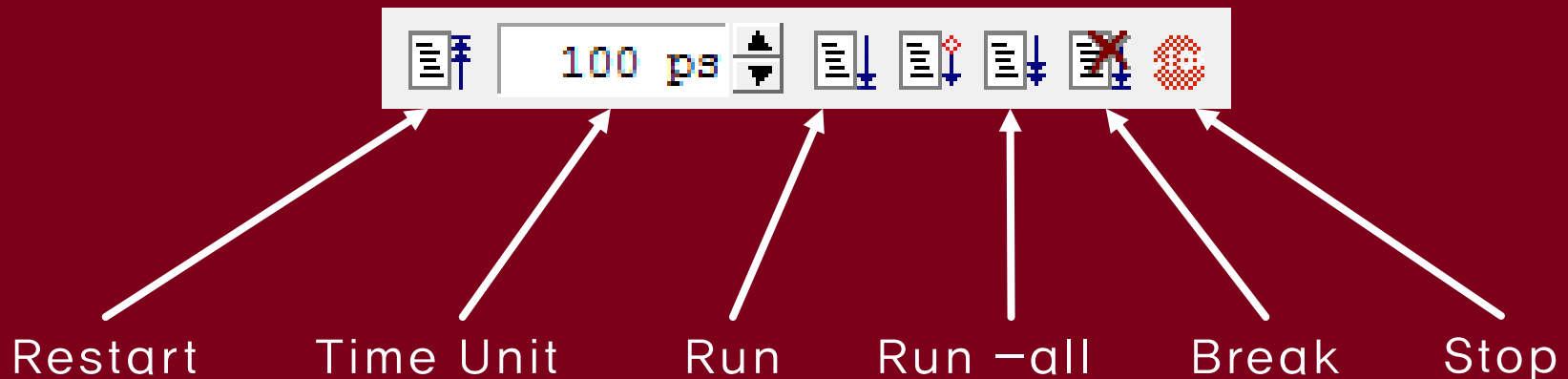


# ModelSim을 이용한 시뮬레이션

- 아래쪽 Transcript 프롬프트에 run 1us 입력하여 시뮬레이션을 시작할 수 있다. (testbench의 timescale 설정에 따라 더 긴 시간 동안 실행해야 할 수도 있다.)

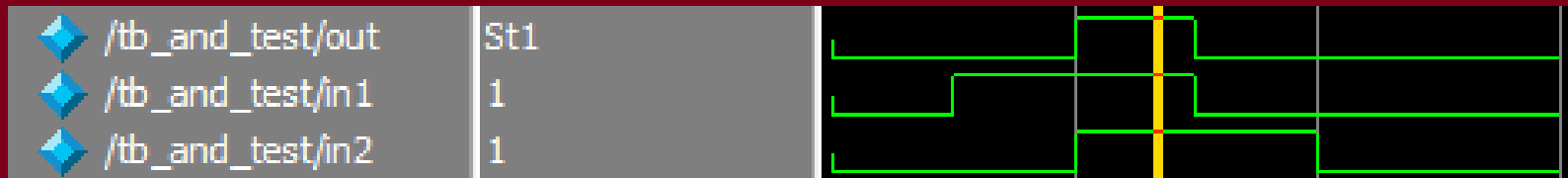


- 또는 메뉴의 아이콘을 눌러서 시뮬레이션을 시작할 수도 있다.



# 시뮬레이션 예시: 1bit AND gate

- 시그널 설명:
  - out: AND gate output, in1/in2: AND gate input





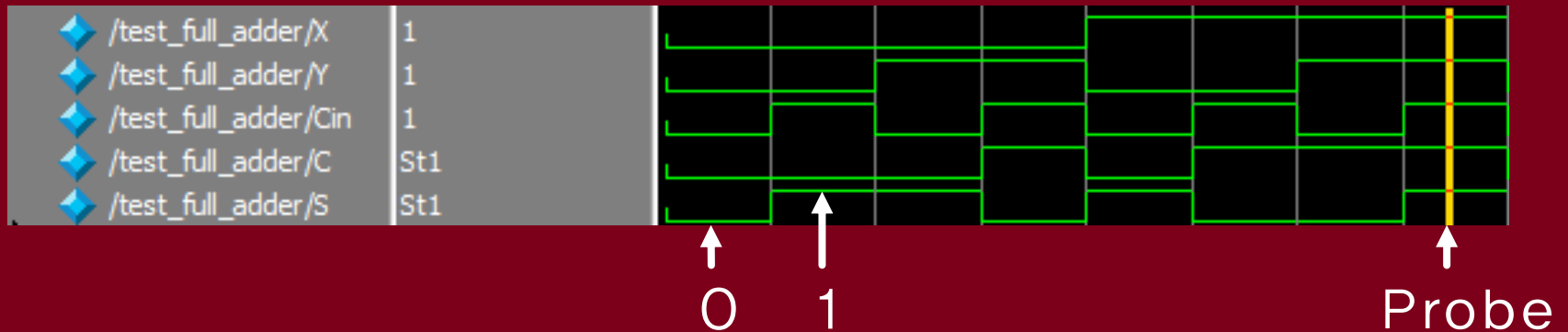
# 시뮬레이션 예시: 1bit Full Adder

- 1bit Full Adder의 Truth table

Input 0	Input 1	Carry in	Carry out	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

# 시뮬레이션 예시: 1bit Full Adder

- 시그널 설명:
  - X: input0, Y: input1, Cin: carry in,
  - C: carry out, S: sum



# Q&A