

# 컴퓨터공학실험II

## 2장 Verilog 기초



*Be as proud of Sogang  
As Sogang is proud of you*

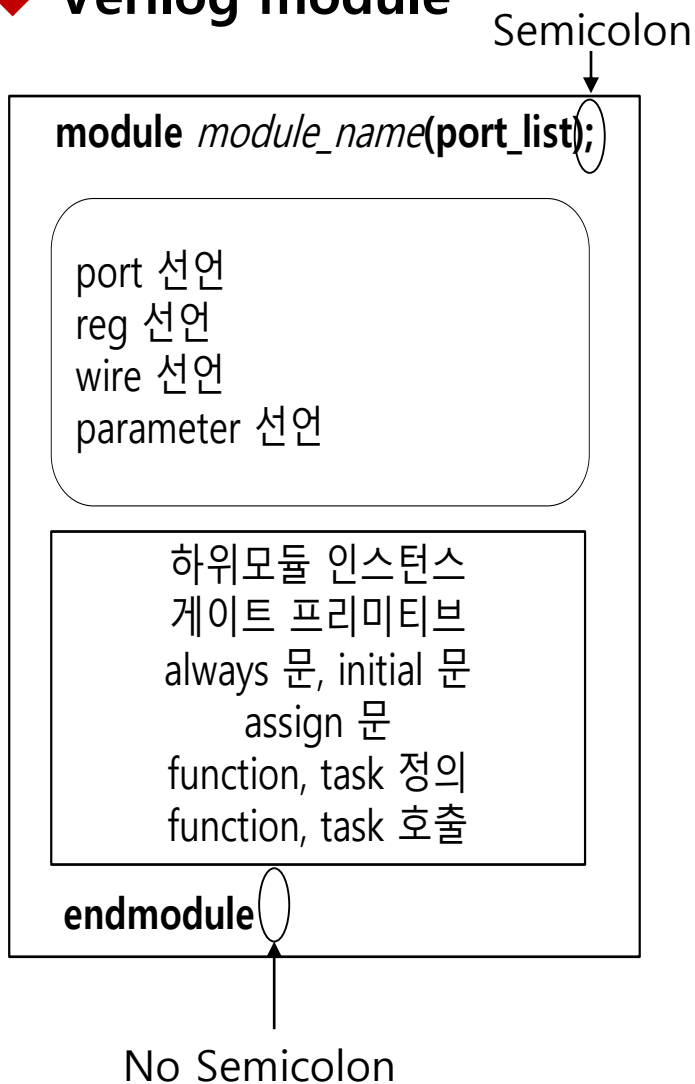
◆ HDL(Hardware Description Language)의 한 종류로 가장 널리 쓰인다.

◆ 회로도 방식과 Verilog 방식 비교

항목	회로도 방식	Verilog 방식
설계 시간	상대적으로 많이 걸림	상대적으로 적게 걸림
설계 용이성	논리 설계 능력 필요 복잡해질수록 많이 어려워짐	동작을 이해하면 비교적 쉽게 기술 가능
설계 이해	설계자 이외에는 이해하기 어려움	다른 사람도 비교적 쉽게 이해
설계 변경	설계자도 변경이 쉽지 않음	다른 사람도 쉽게 변경 가능
설계 이식성	라이브러리나 툴을 바꾸려면 쉽지 않음	비교적 쉽게 다른 것으로 바꿀 수 있음.
문서화 작업	별도의 기술 문서를 작성해야 하는 부담	설계용 Verilog 코드를 기술 문서로서 활용 가능

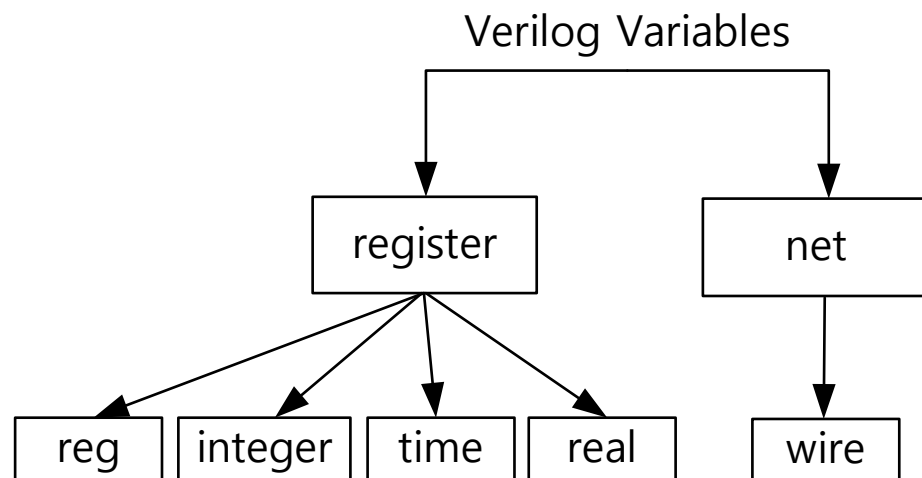
- ◆ Verilog(베릴로그)는 전자 회로 및 시스템에 사용되는 하드웨어 기술 언어로, 회로 설계, 검증, 구현 등 여러 용도로 사용할 수 있다.
- ◆ C 언어와 비슷한 문법을 가져서 사용자들이 쉽게 접근할 수 있도록 만들어졌다. 'if'나 'while'과 같은 제어 구조도 동일하며, 출력 루틴 및 연산자들도 거의 비슷하다.
- ◆ C 언어와 달리, 블록의 시작과 끝을 중괄호 기호를 사용하지 않고, 대신에 Begin과 End를 사용하여 구분하고, HDL의 특징인 시간에 대한 개념이 포함되었다는 것 등의 일반적인 프로그램과의 다른 점도 존재한다.

## ◆ Verilog module



- ◆ Verilog 는 **머리부**, **선언부**, **몸체부** 등 크게 세 부분으로 구성된다.
- ◆ **머리부** 는 키워드 `module` 로 시작하여 모듈 이름, 포트 목록 그리고 `(;)` 으로 끝난다.  
키워드와 동일한 이름을 사용할 수 없다.
- ◆ **선언부** 는 포트 목록에 나열된 포트들의 방향, 비트 폭, `reg` 및 `wire` 선언, `parameter` 선언 등 모듈에서 필요로 하는 것들을 선언한다.
- ◆ **몸체부** 는 회로의 기능, 동작, 구조 등을 표현하는 다양한 Verilog 구문들로 구성된다.

## ◆ Verilog Data Type



◆ **Input** : 입력 신호

◆ **Output** : 출력 신호

## ◆ Register : 추상적인 저장 장치

- reg : 절차형 할당문(always,initial)에 의해 값을 받는 객체.
- integer : 정수형 변수.
- time,realtime : 시간형 변수(타이밍 체크가 필요한 상황에서 시뮬레이션 시간을 처리).
- real : 실수형 변수.

## ◆ Net : 디바이스의 물리적인 연결

- wire : 변수들이 모듈내에서 어떻게 연결되어 있는지 나타내 주는 변수.
- tri : 선을 서로 연결할 때 사용하여 wire와 달리 tri는 3상태(tri-state) net에 사용된다.

## ◆ Verilog HDL의 상수 선언

- 비트수 제한이 있는 reg 값 선언시 형식 : (비트수)'(입력형식)(입력값)
- Size를 정하지 않은 값 (사이즈를 특별히 정하지 않아도 선언이 가능하다.)  
214; [정수 214]  
'h32; [16진수 32]  
'o324; [8진수 324]
- Size를 정한 값(맨 처음에 몇 비트인지 정해주고 선언한다.)  
4'b1111; [4bit의 2진수 1111]  
4'hf; [4bit의 16진수 f(=4'b1111)]  
4'd15; [4bit의 10진수 15(4'b1111)]
- 부호가 있는 수의 처리(음의 값은 2의 보수로 처리 되어 저장된다)  
-8'd6; [8비트의 -6]

## ◆ Verilog 연산자

기호	기능	기호	기능
{},{}	결합, 반복	^	비트단위 exclusive or
+, -, *, /, **	산술	^~ 또는 ~^	비트 단위 등가
%	나머지	&	축약(reduction) and
>, >=, <, <=	관계	~&	축약 nand
!	논리 부정		축약 or

### Example

```
//X=4'b1010, Y=4'b1101, Z=4'b10x1
~X //Negation, Result is 4'b0101
X&Y //Bitwise AND, Result is 4'b1000
X | Y //Bitwise OR, Result is 4'b1111
X^Y //Bitwise XOR, Result is 4'b0111
X^~Y //Bitwise XNOR, Result is 4'b1000
X&Z //Bitwise AND, Result is 4'b10x0
```

## ◆ Shift 연산자

기호	기능	기호	기능
<<	왼쪽 shift	>>	오른쪽 shift

Example

```
//X=4'b1101
```

```
Y=X>>1 // Y is 0110 (0 is filled in MSB position)
```

```
Y=X<<2 //Y is 0100 (0 is filled in LSB position)
```

## ◆ 조건 연산자

conditional\_expression = expression1 ? expression2 : expression3



## ◆ 결합 및 반복 연산자

- Concatenation Operators : { }
- ```
//A = 1'b1 , B=2'b00 , C=2'b10  
Y = {B,C} // Results Y is 4'b0010  
X = {A, B, 3'b110} // Result X is 6'b100110  
Z = {A, B[0], C[1]} // Result Z is 3'b101
```
- Replication Operators : { { } }
- ```
//A = 1'b1 , B = 2'b01 , C = 2'b00  
Y = {4{A}} //Result Y is 4'b1111  
X = {4{A},2{B}} //Result X is 11110101  
Z = {4{A}, {2{B}, C} //Result Z is 1111010100
```

## ◆ Verilog 구문

## ◆ Timescale

- 'timescale <시간단위>/<정밀도>
- <시간단위> : 이 값을 선언하면, 그 파일 내의 모든 시간 단위는 선언한 값으로 바뀐다.  
EX) '1ns'라고 선언하면, 그 파일 내의 모든 시간 단위는 1ns가 된다.
- <정밀도> : 정밀도는 주어진 시간 단위로 구성 할 수 있는 가장 작은 지연을 나타내며, <시간 단위>와 관련하여 사용할 수 있는 소수점의 허용범위를 나타낸다.

```
' timescale 10ns/1ns
    #1.55a=b;
//모든 시간 단위는 10ns로, 1.55*10ns = 15.5ns 이다.
여기서 정밀도 값이 1ns로 소수점 값을 올림하여, 16ns가 된다.
```

```
' timescale 1ns/1ps
    #1.0055a=b;
//모든 시간 단위는 1ns로 1.0055*1ns = 1.0055이다.
여기서 정밀도 값이 1ps로 소수점 값을 올림하여, 1.006ns가 된다.
(1ps=0.001ns)
```

## ◆ Verilog 연속 할당문

### ◆ assign

- **assign** 구문은 입력 피연산자의 값에 변화(**event**)가 발생할 때마다 우변의 식이 평가되고, 그 결과 값이 할당문 주변의 net을 구동(**drive**)하는 하드웨어적 특성을 갖는다. 간단하게 말해서 **net** 변수에 특정 논리 값을 지정하는데 사용한다.
- **deassign** 구문은 variable에 대한 assign 문의 영향을 제거시킬 때 사용한다.

```
wire mynet=enable&data;
```

=

```
wire mynet;  
assign mynet = enable & data;
```

Ex)

```
assign wire1 = reg1; //선을 단순히 연결하는 것.
```

```
assign wire2 = (pin1)?reg2[0] : reg2[1];
```

// 2 to 1 mux 의 역할, 괄호안에 있는것이 참이면 ":" 앞쪽에 있는 것이 거짓이면 ":" 오른쪽에 있는 것 으로 연결.

```
assign wire2 = {reg1,reg2[0]}; // 서로 다른 두개의 신호를 하나의 버스로 집어 넣어줌.
```

## ◆ Verilog 구문

### ◆ always

- **always** 문은 시뮬레이션이 실행되는 동안 반복적으로 실행되며, 따라서 타이밍 제어와 연관된 표현에 유용하게 사용. **@(sensitivity\_list)**는 **always** 문의 실행을 제어하는 역할을 하며, **sensitivity\_list**에 나열된 신호들중 하나 이상에 변화(**event**)가 발생했을 때 **always** 내부에 있는 **begin-end** 블록이 실행된다.

```
always @(sensitivity_list) begin  
Blocking or nonBlocking statements;  
end
```

### ◆ initial

- 시뮬레이션이 진행되는 동안 무한히 반복되는 **always** 구문과는 다르게, **initial** 구문은 시뮬레이션이 실행되는 동안 한번만 실행 된다.  
**initial** 구문의 **begin-end** 블록은 절차형 문장들로 구성되며 이들 절차형 문장들은 나열된 순서대로 실행된다.

```
initial begin  
Blocking or nonBlocking statements;  
end
```

## ◆ Verilog 절차형 할당문

- 문장이 나열된 순서대로 실행(execute)되어 할당 문 좌변의 변수 값을 갱신하는 소프트웨어적 특성을 가짐.  
always 구문, initial 구문 내에서 사용.

## ◆ Blocking statement

- 1) Blocking symbol: =
- 2) Begin ~ end 까지 Line by Line으로 순차적으로 계산과 동시에 저장에 이루어짐
- 3) 할당 까지 다 수행 후 다음 문장 수행->즉 한 문장이 수행이 끝나기 전에 blocking 된다.
- 4) #t 변수 = 연산 ; -> t시간 후, 연산하고 변수에 할당한다.

Ex)

변수 C = A & B; A 와 B 값을 읽고 & 연산하고 C에 할당. 그리고 다음 문장 수행  
변수 D = A | B; A 와 B 값을 읽고 | 연산하고 D에 할당.  
이때 이들의 수행과정의 delay는 존재 하지 않는다.

## ◆ Non blocking statement

- 1) non Blocking symbol: <=
- 2) Begin ~ end 까지의 모든 계산을 수행한 후 한꺼번에 저장 작업이 수행
- 3) 여러 non-blocking 구문이 동시에 평가 한 후에 할당 된다.
- 4) 변수 <= #t연산 ; -> 연산하고 t후, 변수 할당을 예약해 놓는다.

\*공통적인 사건 발생후, 여러 의 data를 동시에 전송하기 위해 사용된다.

Ex)

변수 C <= A & B; A와 B의 값을 읽고, & 연산하고,  
변수 D <= A | B; 동시에 A와 B의 값을 읽고, | 연산하고,  
그리고 나서 동시에 C와 D에 할당한다.

## ◆ Verilog Blocking 문법 예시

### Nonblocking statement

```

`timescale 1ns / 1ps

module inv;

    reg a,b,clk;

    initial begin
        a=0;
        b=1;
        clk=0;
    end

    always clk = #5 ~clk;

    always @(posedge clk)begin
        a<=b;
        b<=a;
    end

endmodule
    
```

### Blocking statement

```

`timescale 1ns / 1ps

module inv;

    reg a,b,clk;

    initial begin
        a=0;
        b=1;
        clk=0;
    end

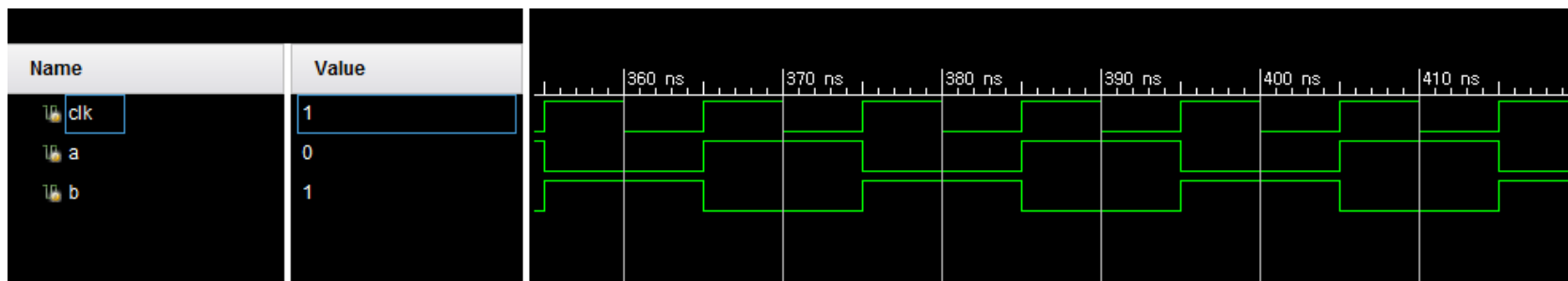
    always clk = #5 ~clk;

    always @(posedge clk)begin
        a=b;
        b=a;
    end

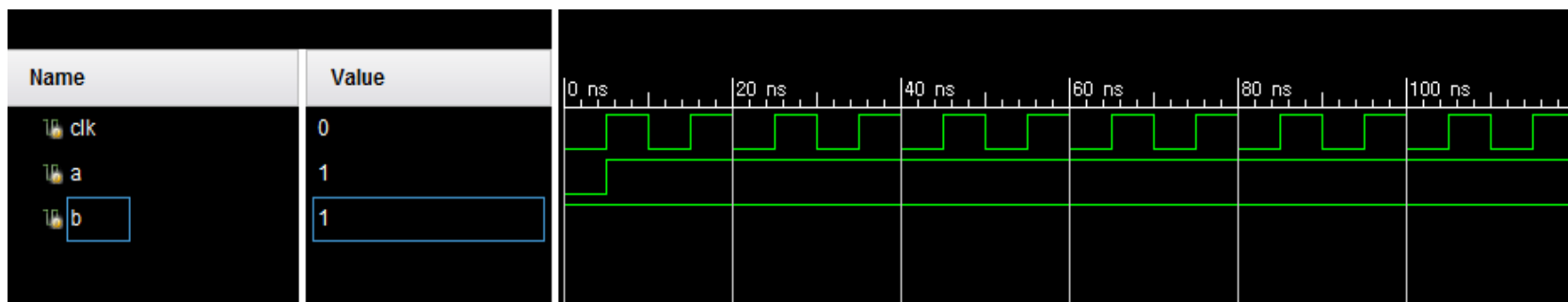
endmodule
    
```

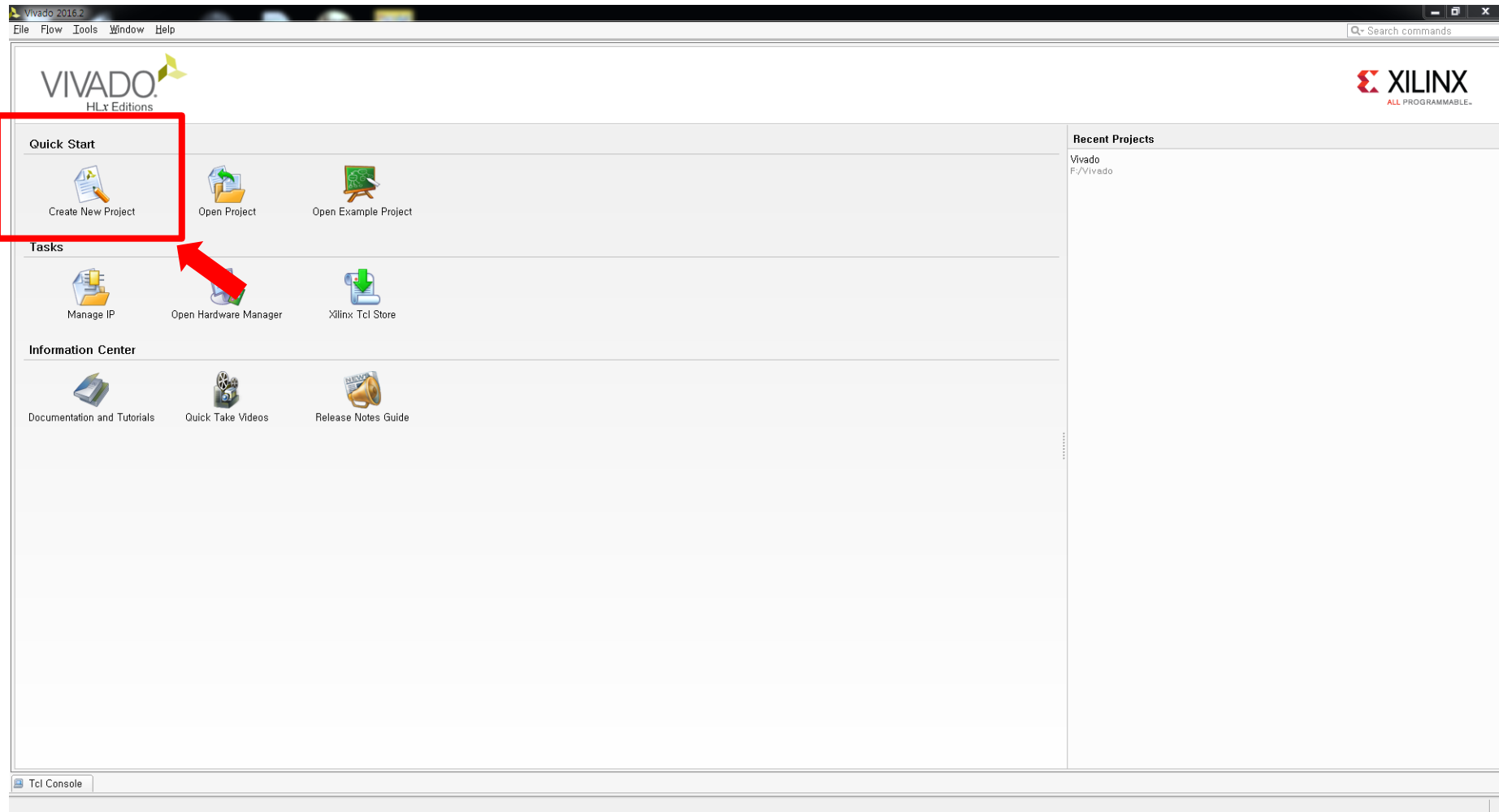
## ◆ Verilog Blocking 문법 예시

### ● Nonblocking statement

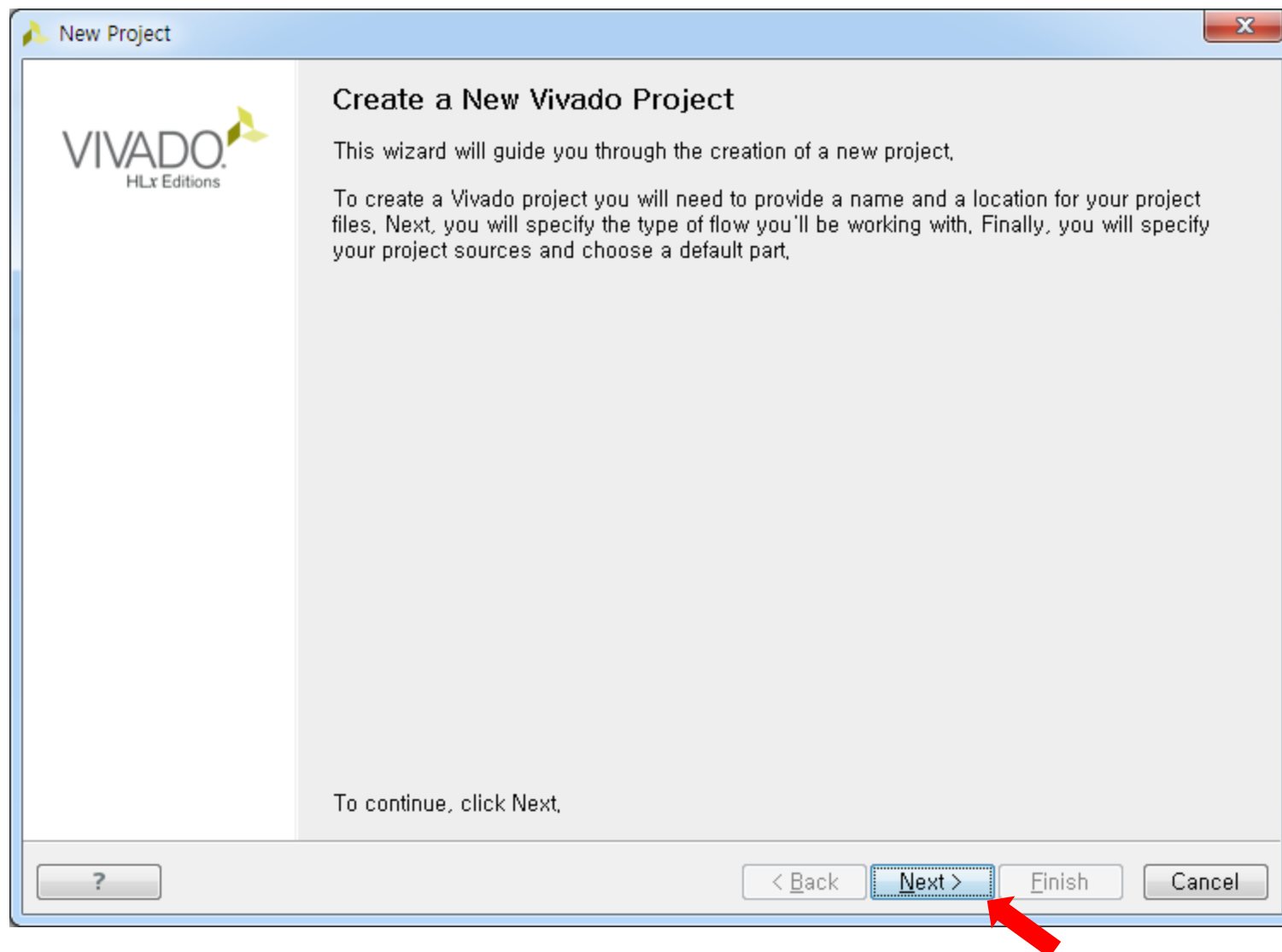


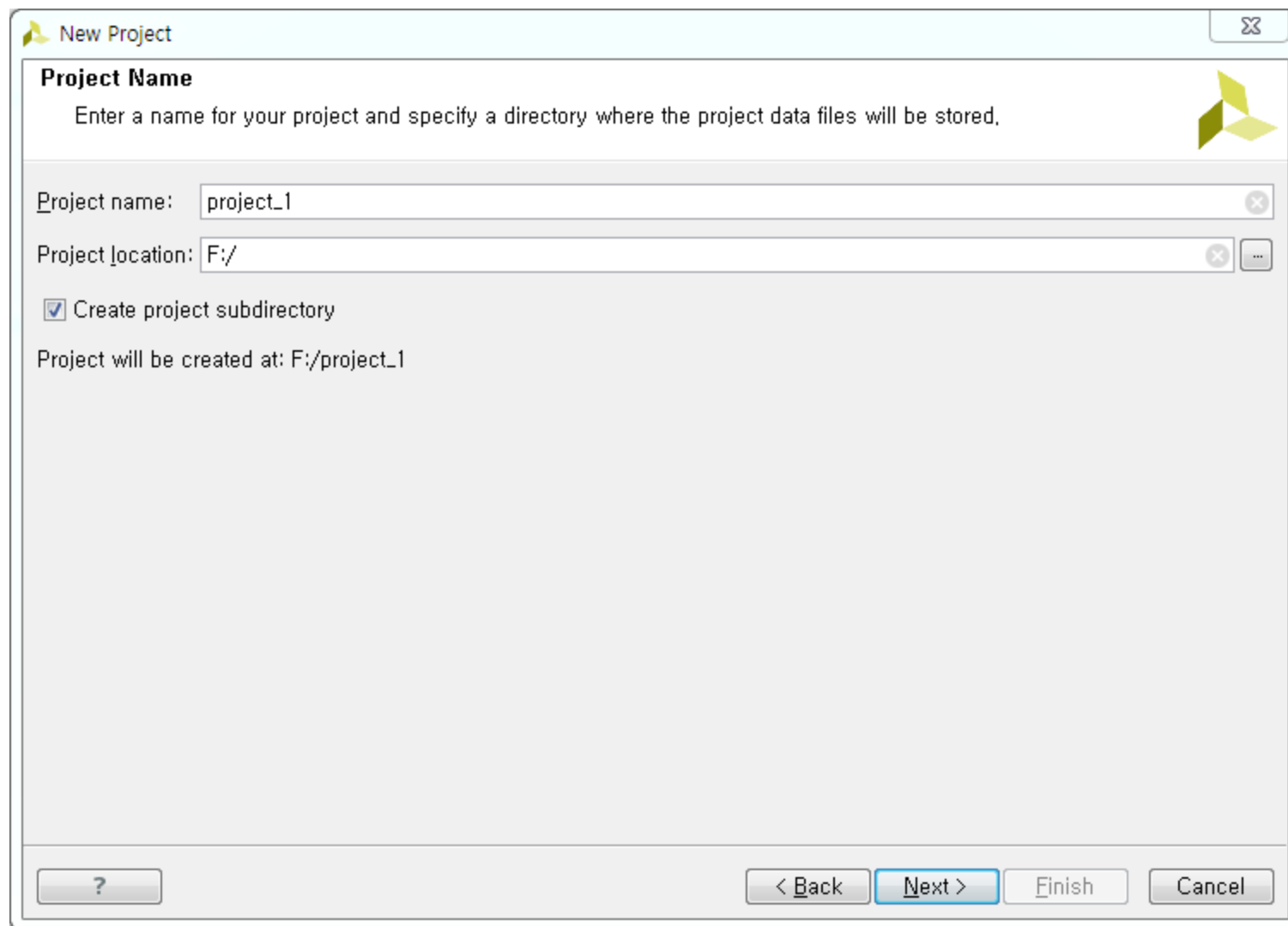
### ● Blocking statement











**New Project**

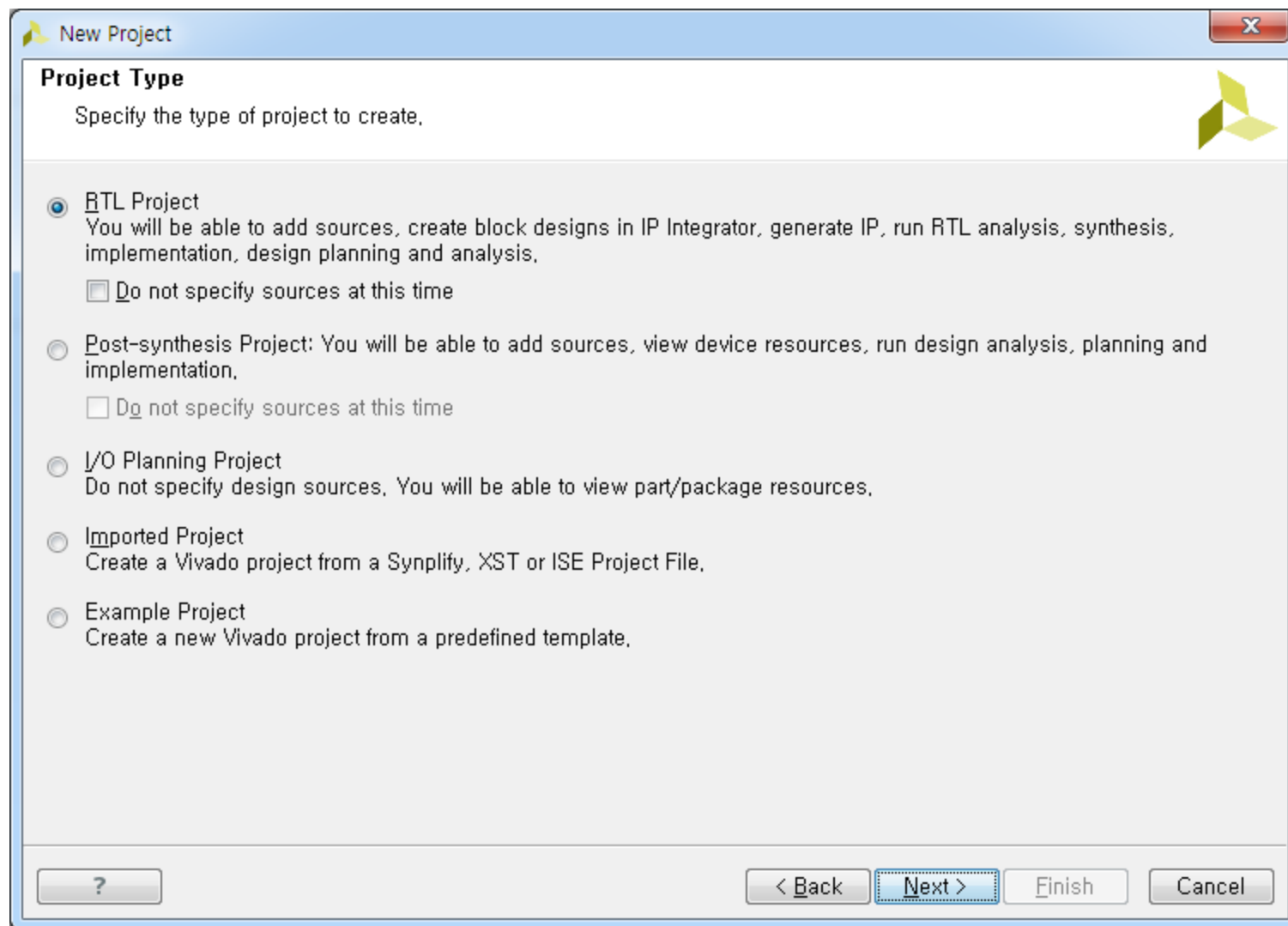
**Project Name**  
Enter a name for your project and specify a directory where the project data files will be stored.

Project name:

Project location:

☒ Create project subdirectory

Project will be created at: F:/project\_1



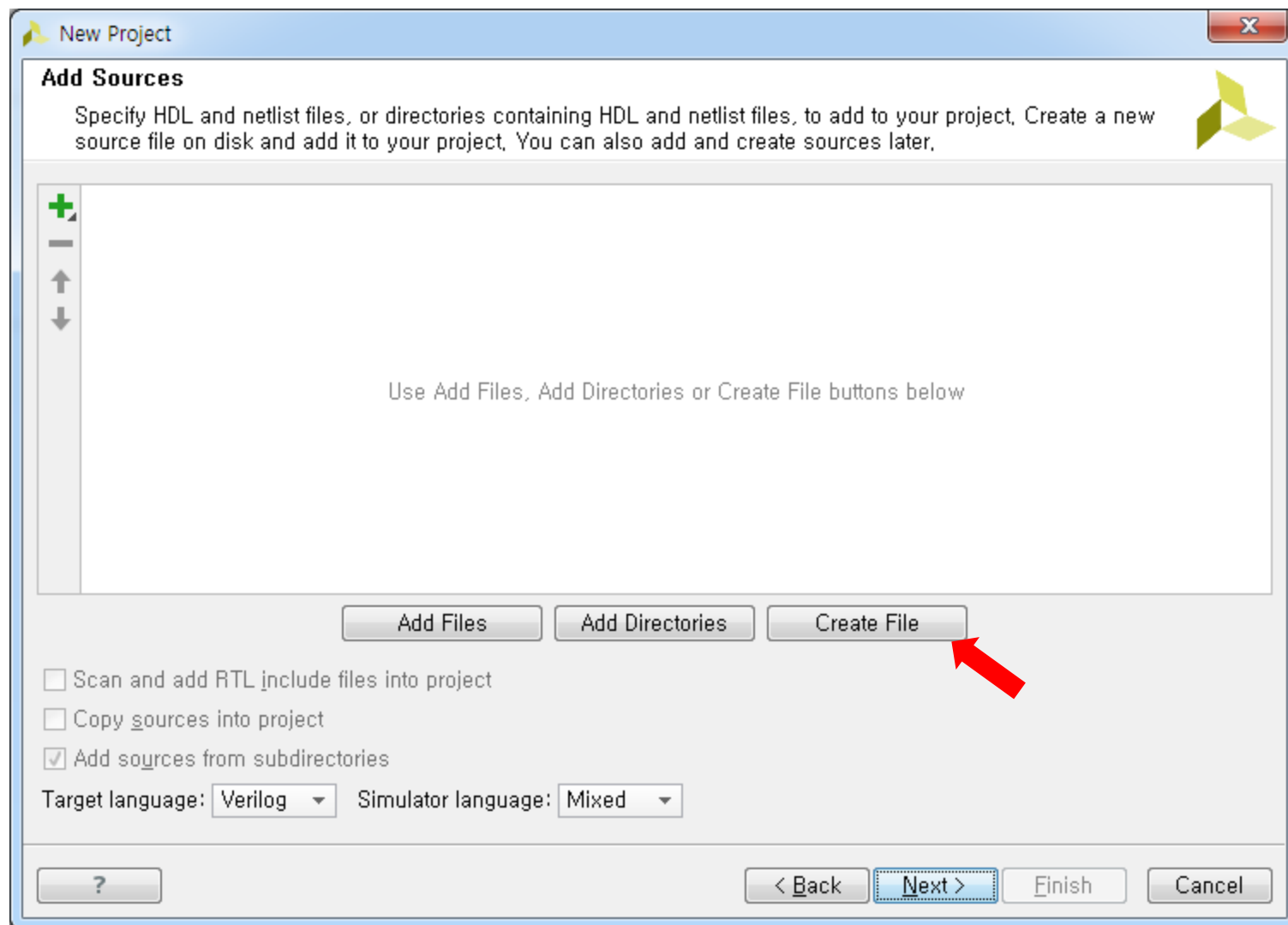
The image shows the 'New Project' wizard in Vivado. The window has a title bar 'New Project' with a close button. The main area is titled 'Project Type' and contains the instruction 'Specify the type of project to create,'. There are five radio button options, each with a description. The first option, 'RTL Project', is selected. Below it is a checkbox 'Do not specify sources at this time'. The other options are 'Post-synthesis Project', 'I/O Planning Project', 'Imported Project', and 'Example Project', each with their respective descriptions. At the bottom, there are four buttons: a help button with a question mark, '< Back', 'Next >' (which is highlighted with a blue border), 'Finish', and 'Cancel'.

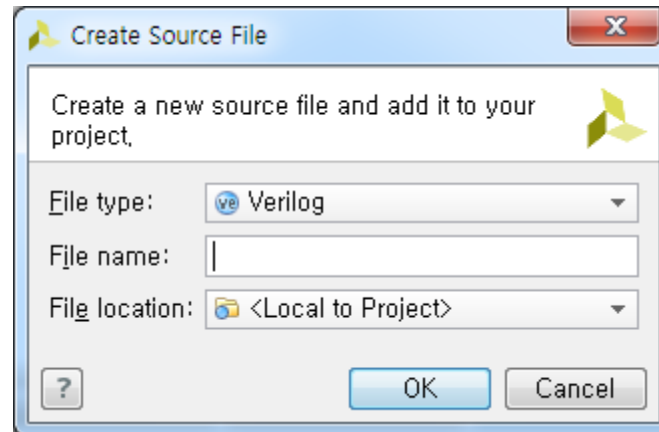
**New Project**

**Project Type**  
Specify the type of project to create.

- ☒ **RTL Project**  
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.  
☐ Do not specify sources at this time
- ☐ **Post-synthesis Project:** You will be able to add sources, view device resources, run design analysis, planning and implementation.  
☐ Do not specify sources at this time
- ☐ **I/O Planning Project**  
Do not specify design sources, You will be able to view part/package resources.
- ☐ **Imported Project**  
Create a Vivado project from a Synplify, XST or ISE Project File.
- ☐ **Example Project**  
Create a new Vivado project from a predefined template.

? < Back Next > Finish Cancel






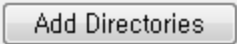



**New Project**

**Add Sources**

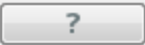
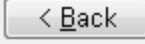

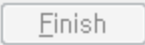
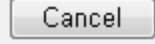
Specify HDL and netlist files, or directories containing HDL and netlist files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.

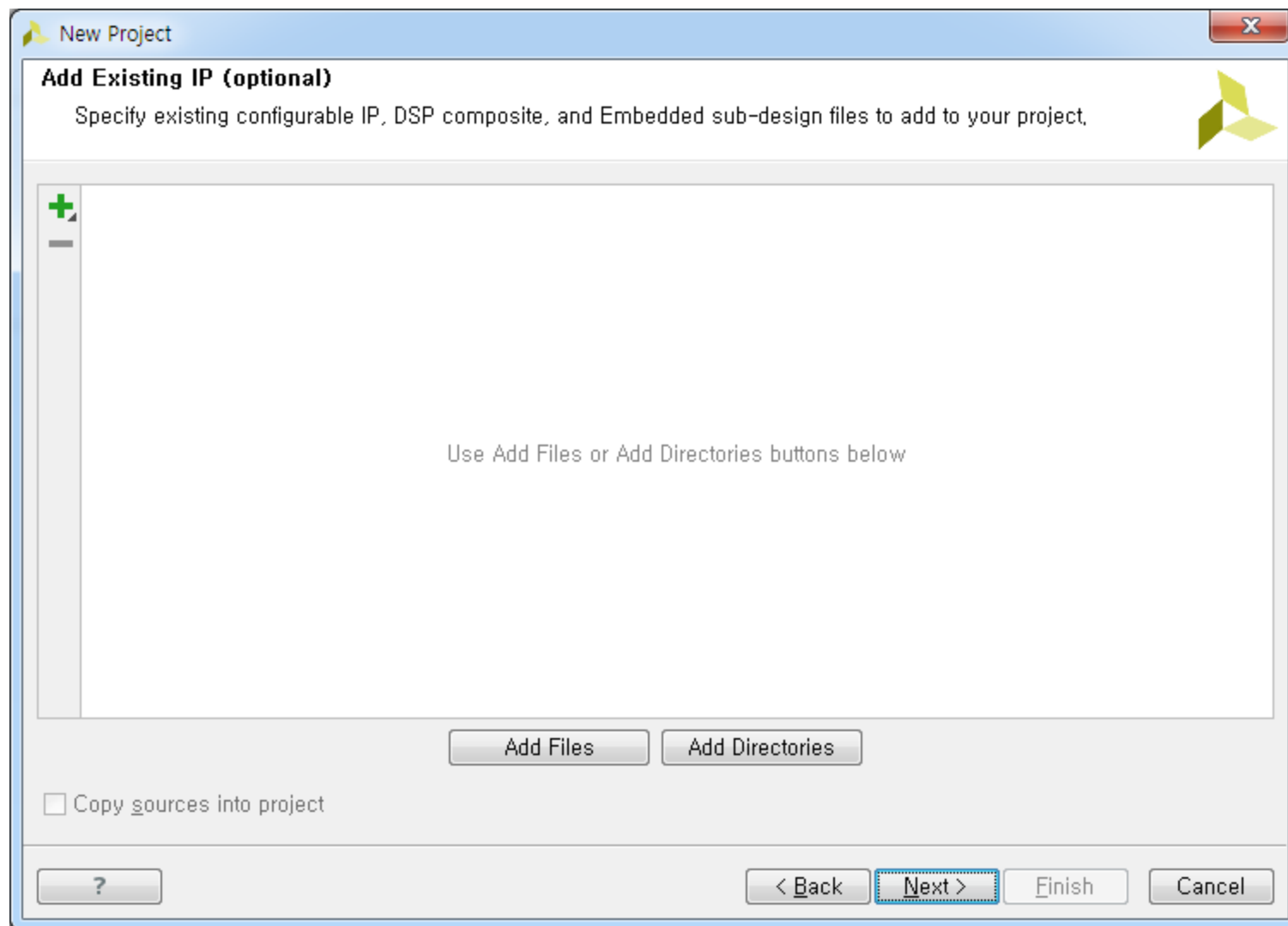
	Index	Name	Library	HDL Source For	Location
	 1	inv.v	xil_defaultlib	Synthesis & Simulati...	<Local to Project>

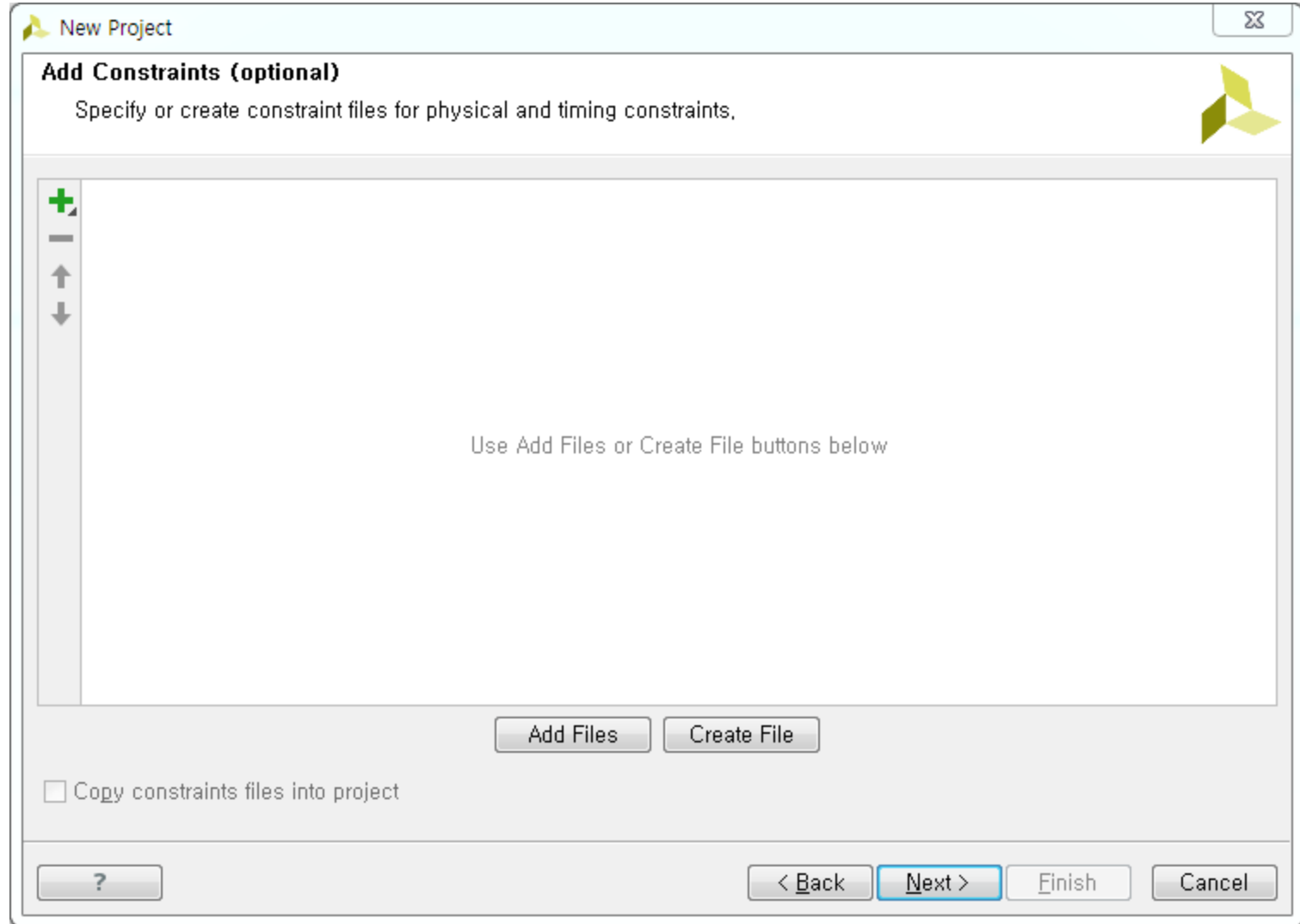
  

☐ Scan and add RTL include files into project  
☐ Copy sources into project  
☒ Add sources from subdirectories

Target language: Verilog Simulator language: Mixed







**New Project**

**Default Part**

Choose a default Xilinx part or board for your project. This can be changed later.

Select: ☒ Parts ☐ Boards





Filter


Product category:  Speed grade:

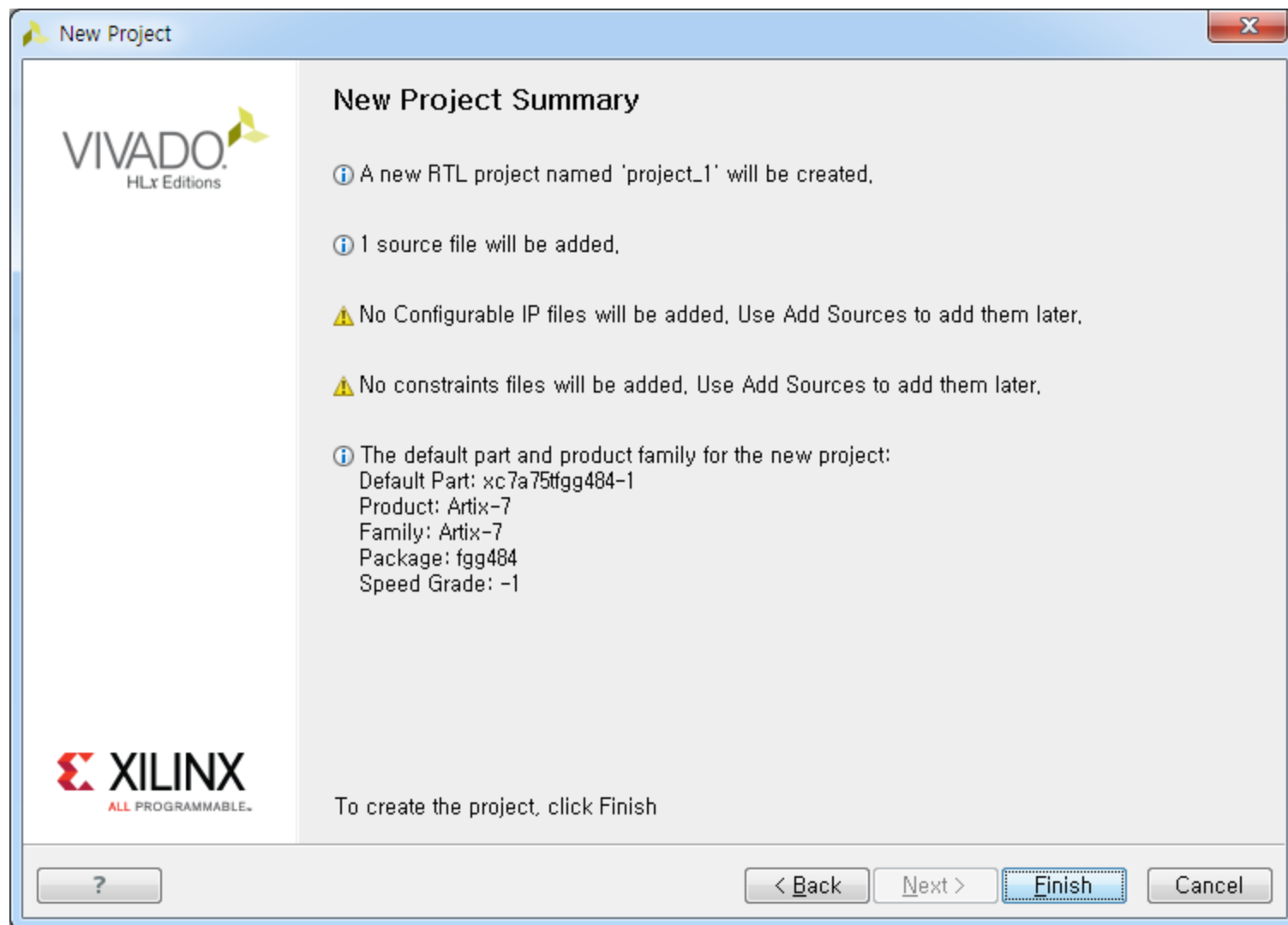
Family:  Temp grade:

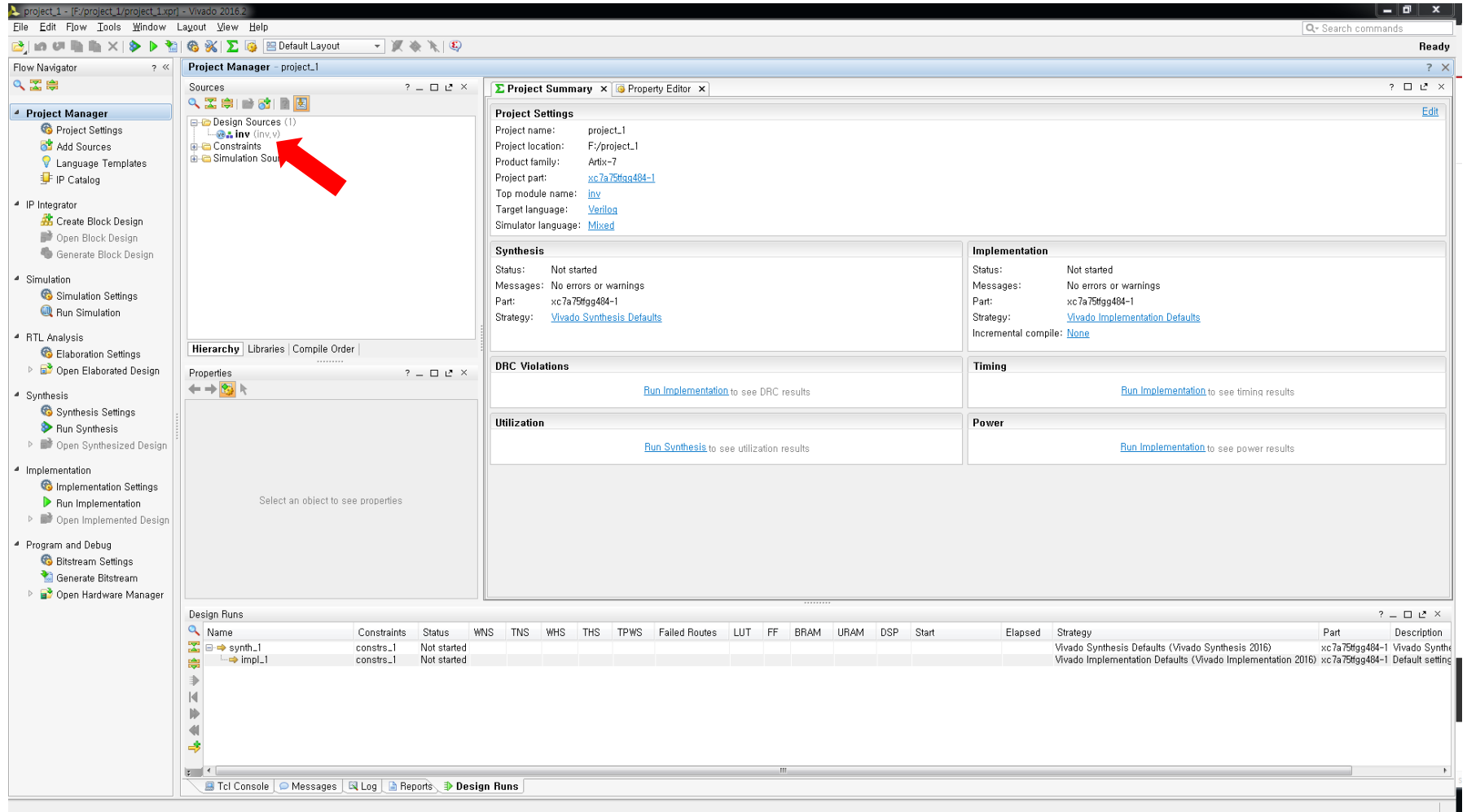
Package:

Search:  (4 matches)

Part	I/O Pin Count	Block RAMs	DSPs	FlipFlops	GTPE2 Transceivers	Gb Transceivers	Available IOBs	LU El
 xc7a75tfgg484-3	484	105	180	94400	4	4	285	472
 xc7a75tfgg484-2	484	105	180	94400	4	4	285	472
 xc7a75tfgg484-2L	484	105	180	94400	4	4	285	472
 xc7a75tfgg484-1	484	105	180	94400	4	4	285	472







The screenshot displays the Vivado 2016.2 IDE interface. The top menu bar includes File, Edit, Flow, Tools, Window, Layout, View, and Help. The left sidebar shows the Flow Navigator with various project management options. The main workspace is divided into several panes:

- Project Manager - project\_1**: Shows the Sources tree with Design Sources (1), Constraints, and Simulation Sources. A red arrow points to the 'inv' source.
- Project Summary**: Contains Project Settings, Synthesis, Implementation, DRC Violations, Utilization, Timing, and Power sections.
- Design Runs**: A table showing the status of various design runs.

**Project Settings**

Project name:	project_1
Project location:	F:/project_1
Product family:	Artix-7
Project part:	xc7a75tgg484-1
Top module name:	inv
Target language:	Venilog
Simulator language:	Mixed

**Synthesis**

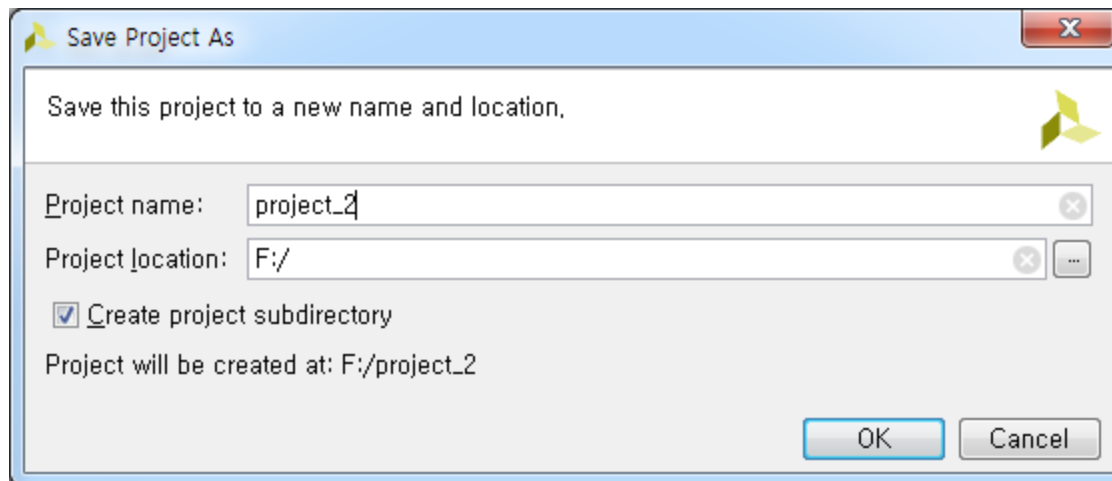
Status:	Not started
Messages:	No errors or warnings
Part:	xc7a75tgg484-1
Strategy:	Vivado Synthesis Defaults

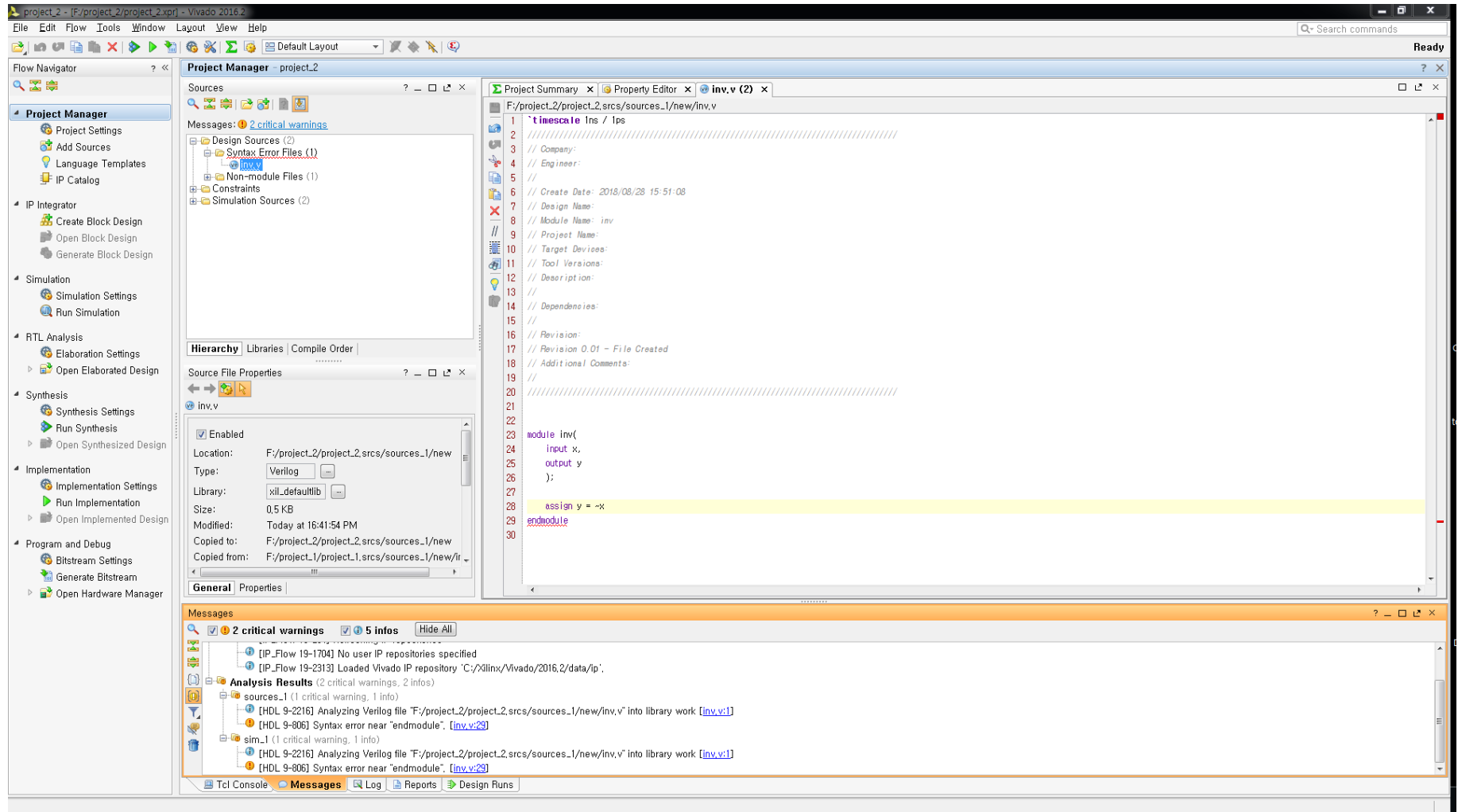
**Implementation**

Status:	Not started
Messages:	No errors or warnings
Part:	xc7a75tgg484-1
Strategy:	Vivado Implementation Defaults
Incremental compile:	None

**Design Runs**

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Failed Routes	LUT	FF	BRAM	URAM	DSP	Start	Elapsed	Strategy	Part	Description
synth_1	constrs_1	Not started														Vivado Synthesis Defaults (Vivado Synthesis 2016)	xc7a75tgg484-1	Vivado Synthesis Defaults (Vivado Synthesis 2016)
impl_1	constrs_1	Not started														Vivado Implementation Defaults (Vivado Implementation 2016)	xc7a75tgg484-1	Vivado Implementation Defaults (Vivado Implementation 2016)





The screenshot displays the Vivado 2016.2 IDE interface for a project named 'project\_2'. The top menu bar includes File, Edit, Flow, Tools, Window, Layout, View, and Help. The Flow Navigator on the left shows the Project Manager with options like Project Settings, Add Sources, Language Templates, IP Catalog, IP Integrator, Simulation, RTL Analysis, Synthesis, Implementation, and Program and Debug. The Project Manager window shows the project hierarchy with Design Sources (2), Syntax Error Files (1), Non-module Files (1), Constraints, and Simulation Sources (2). The Source File Properties window for 'inv.v' shows it is a Verilog file located at 'F:/project\_2/project\_2.srcs/sources\_1/new', with a size of 0.5 KB and a modification time of 'Today at 16:41:54 PM'. The main editor window displays the Verilog code for an inverter module:

```

1 timescale 1ns / 1ps
2 //////////////////////////////////////////////////
3 // Company:
4 // Engineer:
5 //
6 // Create Date: 2018/08/28 15:51:08
7 // Design Name:
8 // Module Name: inv
9 // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////
21
22
23 module inv(
24     input x,
25     output y
26 );
27
28     assign y = ~x;
29 endmodule
30

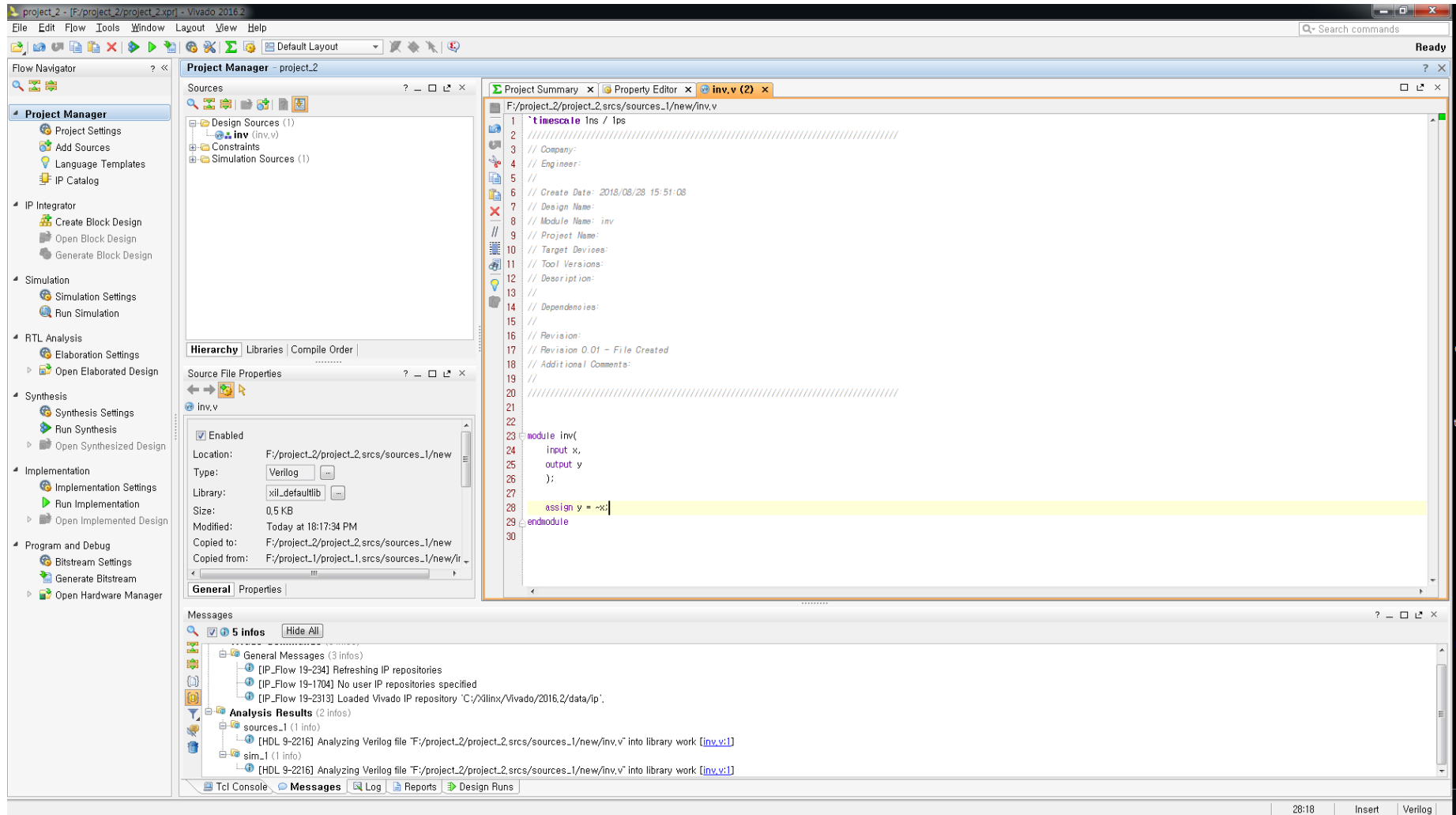
```

The Messages window at the bottom shows 2 critical warnings and 5 infos. The critical warnings include:

- [IP\_Flow 19-1704] No user IP repositories specified
- [IP\_Flow 19-2313] Loaded Vivado IP repository 'C:/xilinx/Vivado/2016.2/data/ip'.
- [HDL 9-2216] Analyzing Verilog file 'F:/project\_2/project\_2.srcs/sources\_1/new/inv.v' into library work [inv.v:1]
- [HDL 9-806] Syntax error near "endmodule". [inv.v:29]

The analysis results show 2 critical warnings and 2 infos, including:

- [sim\_1] (1 critical warning, 1 info)
- [HDL 9-2216] Analyzing Verilog file 'F:/project\_2/project\_2.srcs/sources\_1/new/inv.v' into library work [inv.v:1]
- [HDL 9-806] Syntax error near "endmodule". [inv.v:29]



The screenshot displays the Vivado IDE interface for a project named 'project\_2'. The main editor window shows a Verilog source file 'inv.v' with the following content:

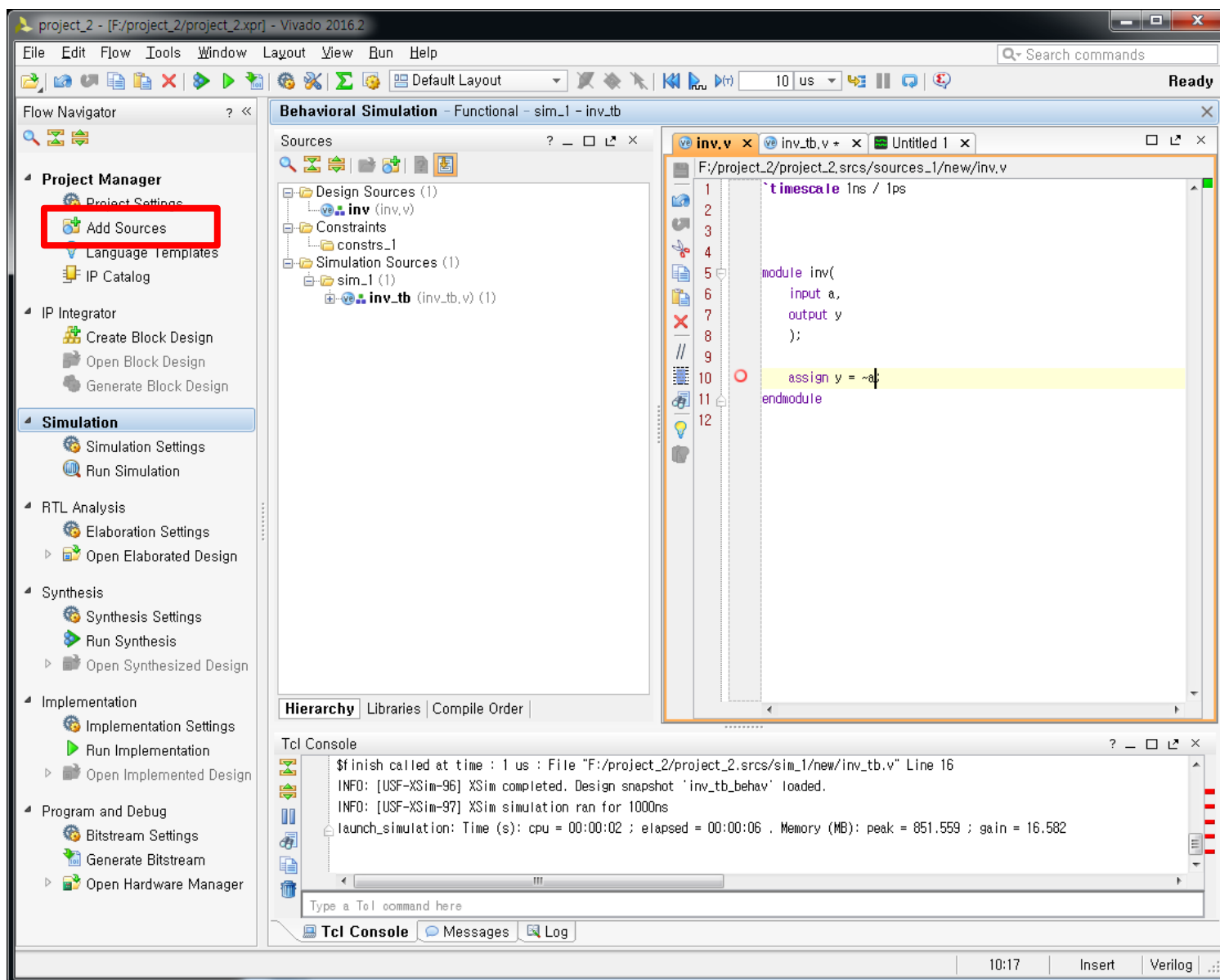
```

1 `timescale 1ns / 1ps
2 //////////////////////////////////////////////////
3 // Company:
4 // Engineer:
5 //
6 // Create Date: 2018/08/28 15:51:08
7 // Design Name:
8 // Module Name: inv
9 // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////
21
22
23 module inv(
24     input x,
25     output y
26 );
27
28     assign y = ~x;
29 endmodule
30

```

The left sidebar contains the Project Manager, IP Integrator, Simulation, RTL Analysis, Synthesis, and Implementation sections. The Project Manager shows the project structure with Design Sources (1), Constraints, and Simulation Sources (1). The IP Integrator shows options for creating, opening, or generating block designs. The Simulation section includes Simulation Settings and Run Simulation. The RTL Analysis section includes Elaboration Settings and Open Elaborated Design. The Synthesis section includes Synthesis Settings, Run Synthesis, and Open Synthesized Design. The Implementation section includes Implementation Settings, Run Implementation, and Open Implemented Design. The Program and Debug section includes Bitstream Settings, Generate Bitstream, and Open Hardware Manager.

The bottom status bar shows the current time as 28:18, and the active window is 'Messages'.



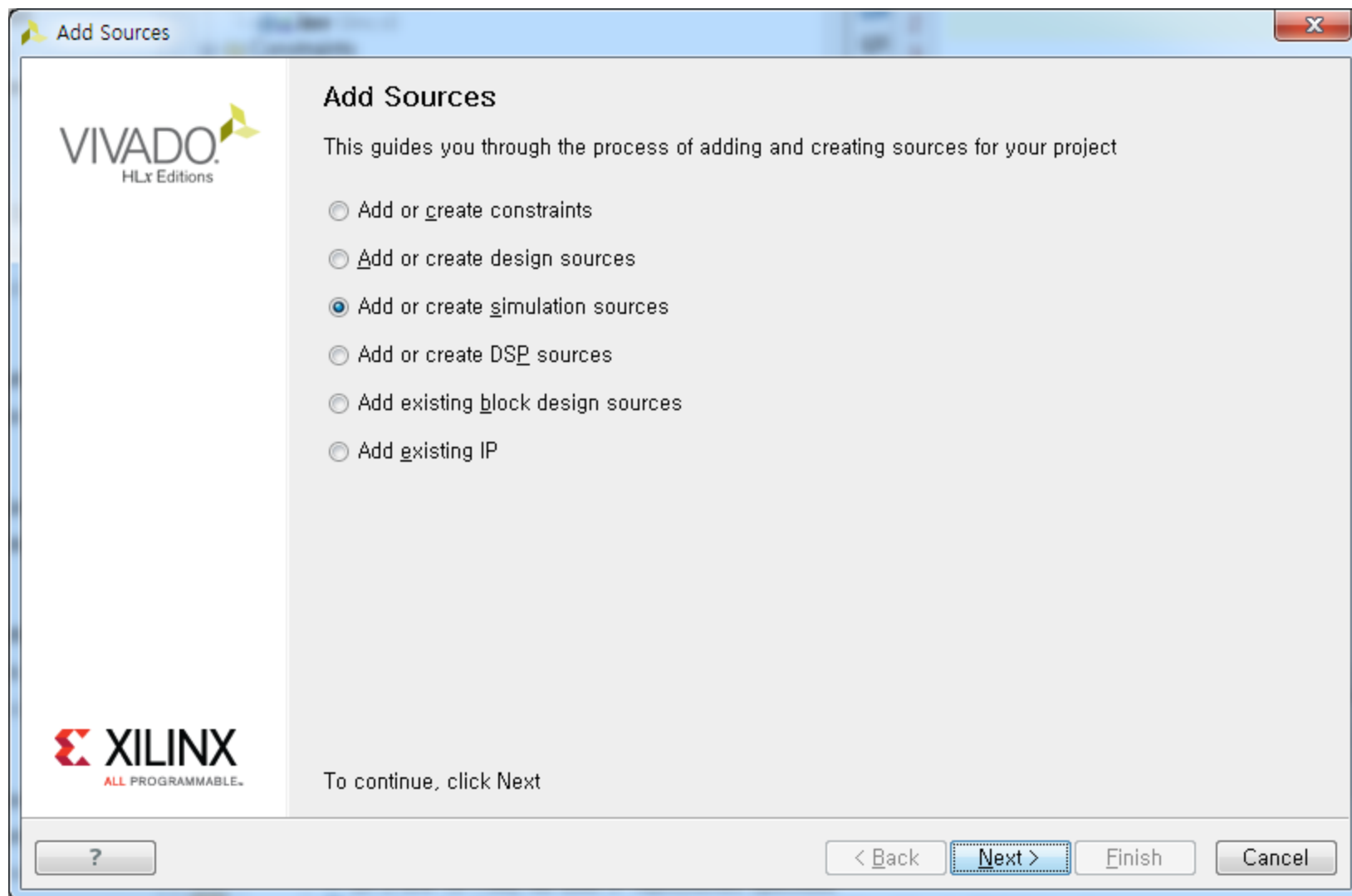
```
`timescale 1ns / 1ps
```

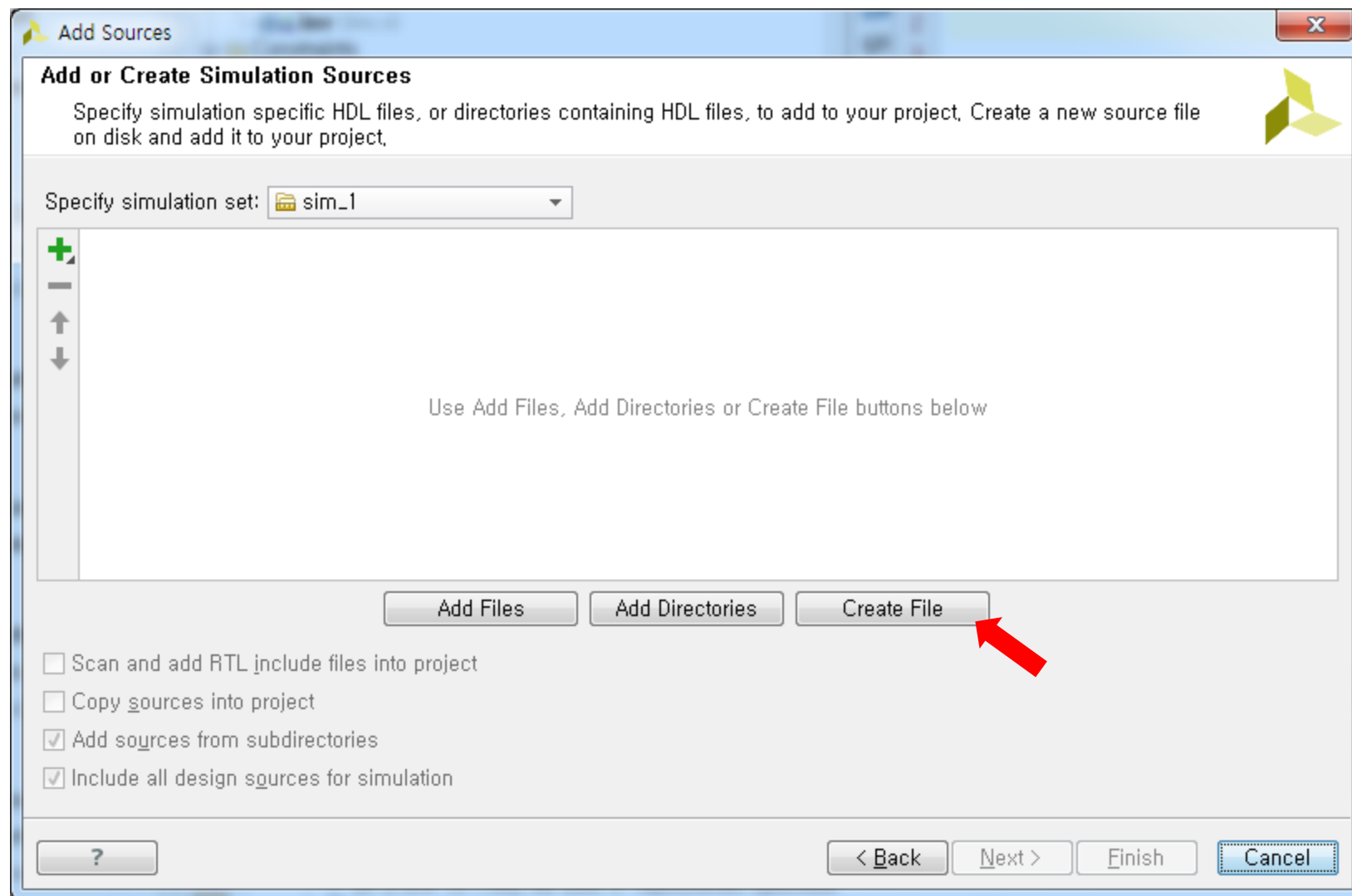
```
module inv(  
    input a,  
    output y  
);
```

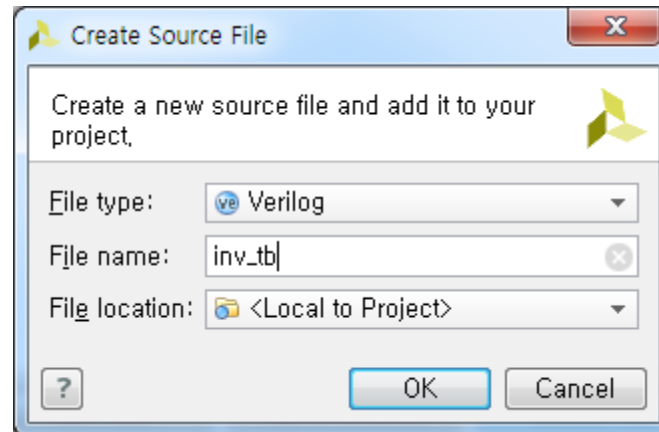
```
    assign y=~a;
```


```
endmodule
```











**Add Sources**
✕

**Add or Create Simulation Sources**


Specify simulation specific HDL files, or directories containing HDL files, to add to your project. Create a new source file on disk and add it to your project.

Specify simulation set: sim\_1



	Index	Name	Library	Location
+	1	inv_tb.v	xil_defaultlib	<Local to Project>

+
−
↕

Add Files
Add Directories
Create File

☐ Scan and add RTL include files into project  
☐ Copy sources into project  
☒ Add sources from subdirectories  
☒ Include all design sources for simulation

?
< Back
Next >
Finish
Cancel


**Define Module**



Define a module and specify I/O Ports to add to your source file.  
 For each port specified:  
 MSB and LSB values will be ignored unless its Bus column is checked.  
 Ports with blank names will not be written.

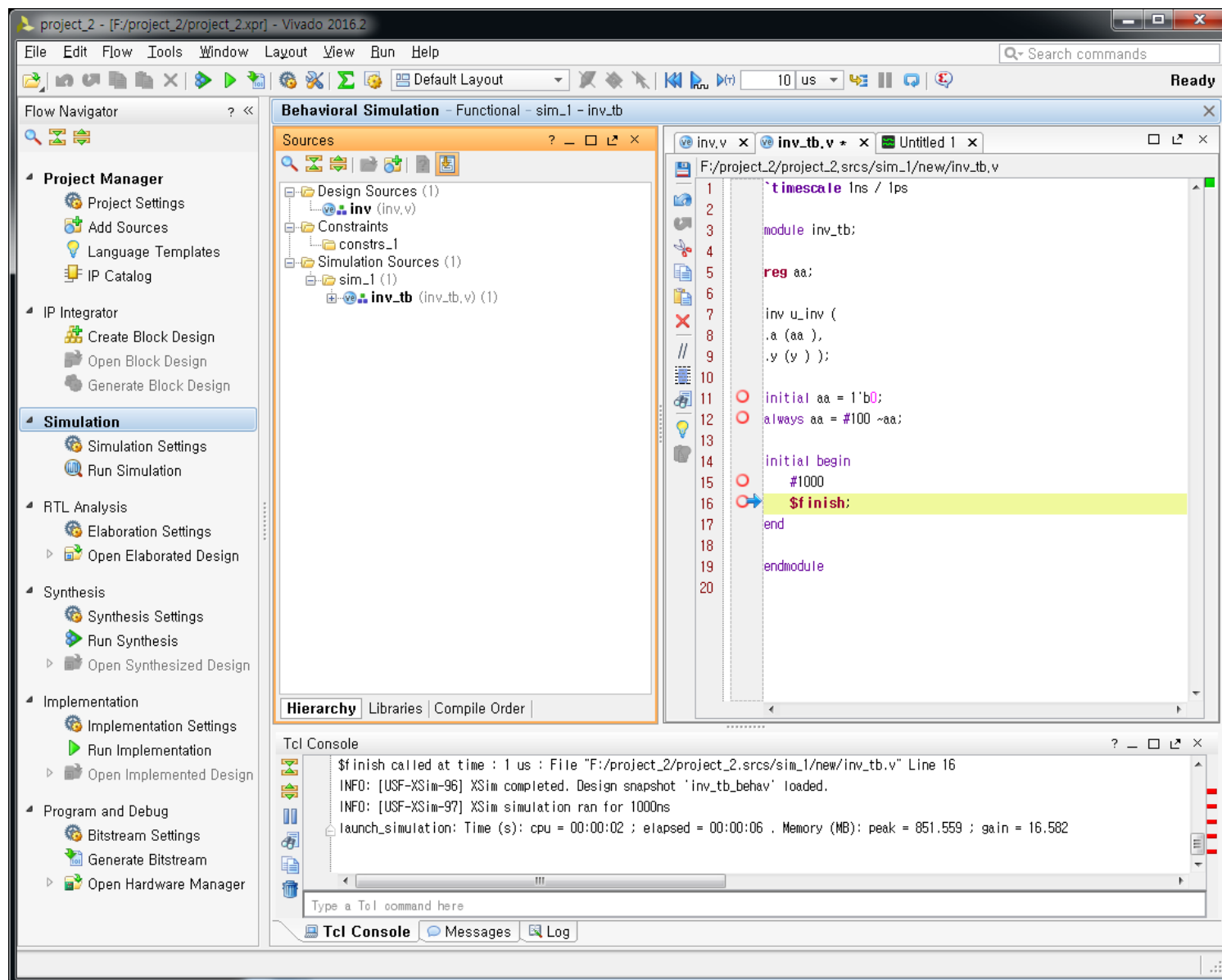
**Module Definition**

Module name:

**I/O Port Definitions**

	Port Name	Direction	Bus	MSB	LSB
+		input	<input checked="" type="checkbox"/>	0	0
-					
↑					
↓					





```
`timescale 1ns / 1ps

module inv_tb;

reg aa;

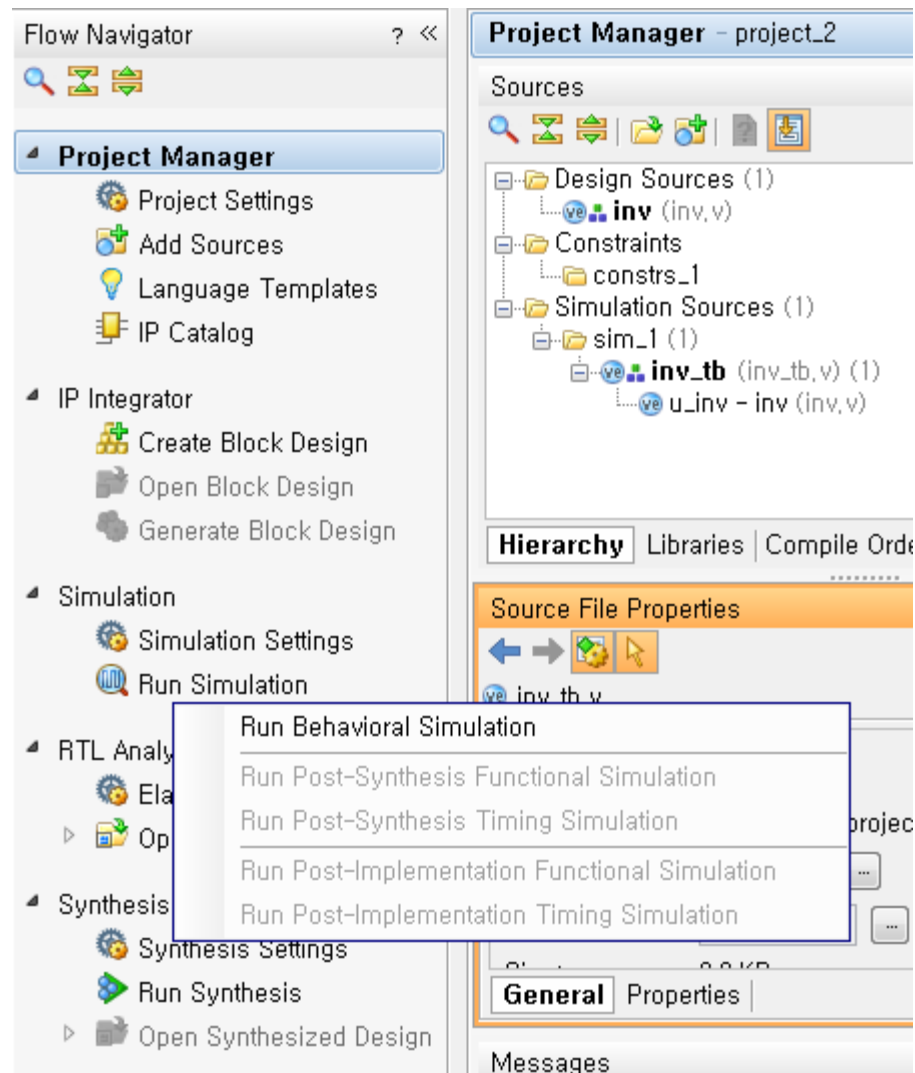
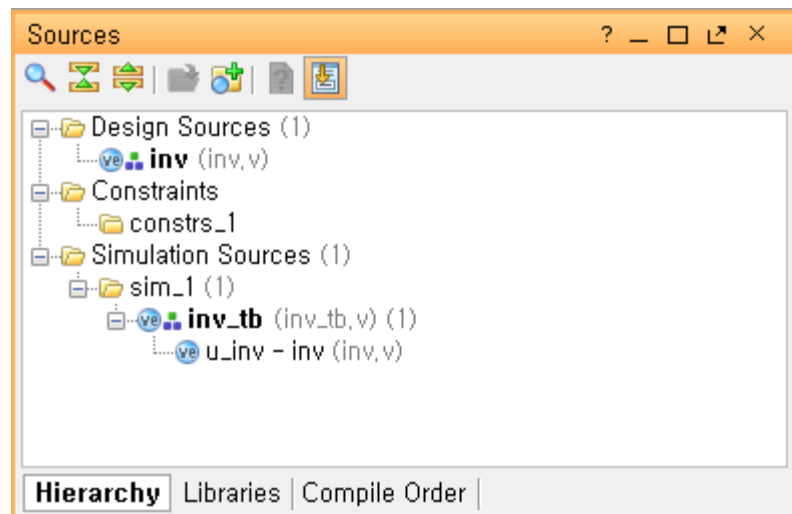
wire y;

inv u_inv (
.a (aa ),
.y (y ) );

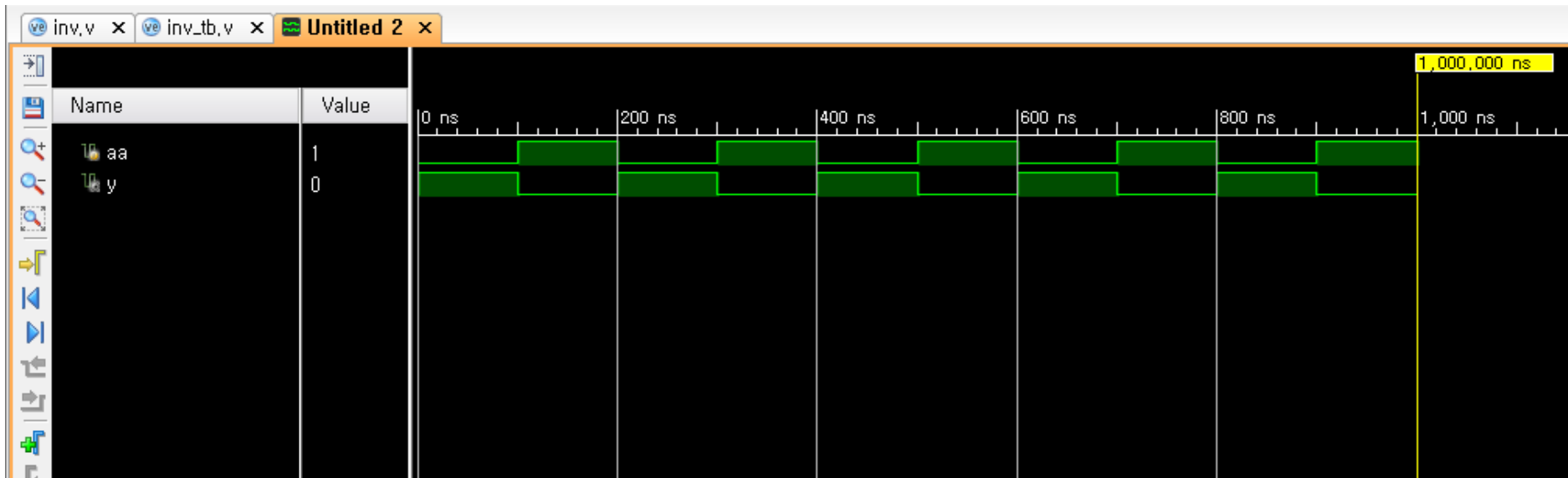
initial aa = 1'b0;
always aa = #100 ~aa;

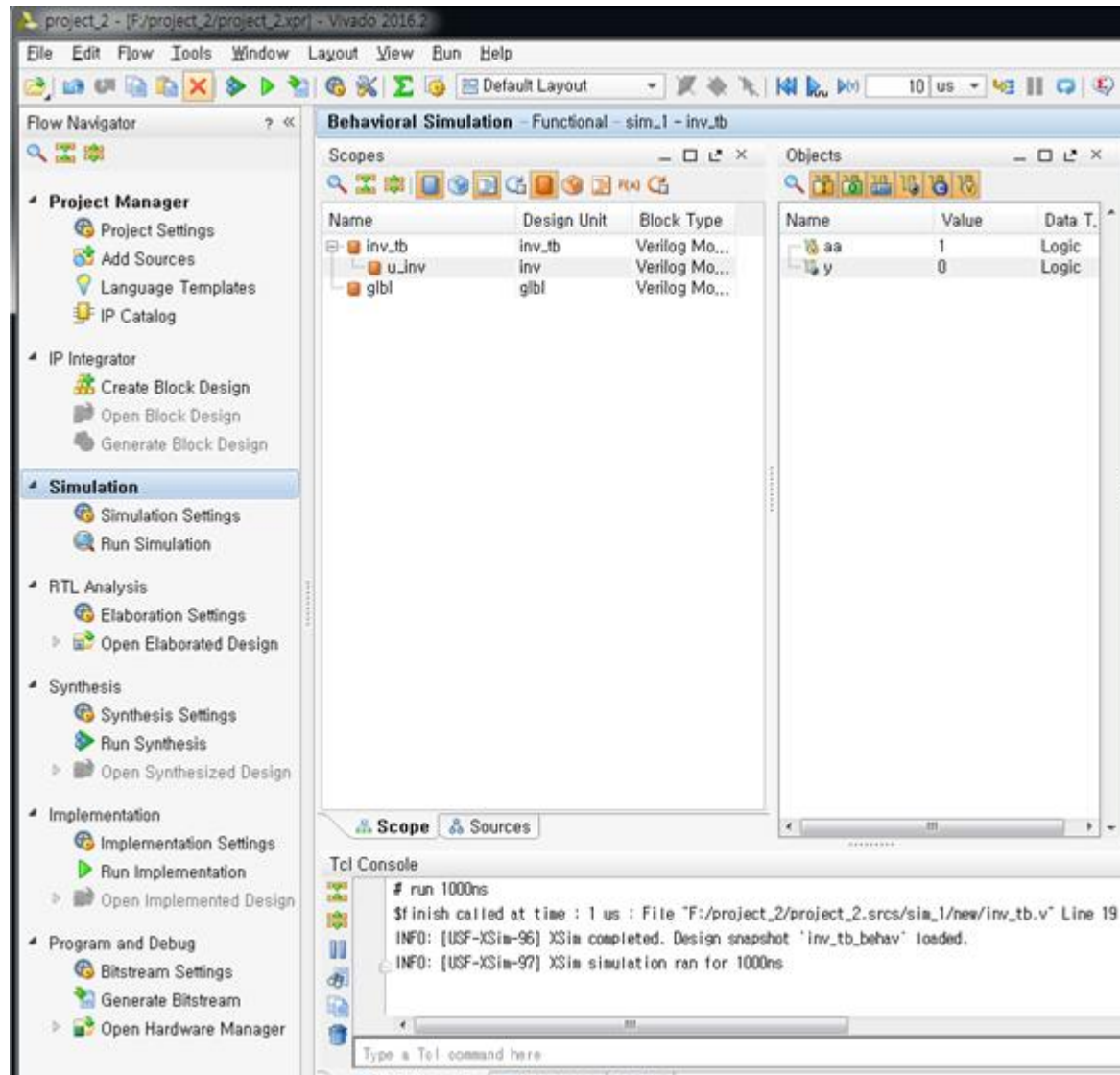
initial begin
    #1000
    $finish;
end

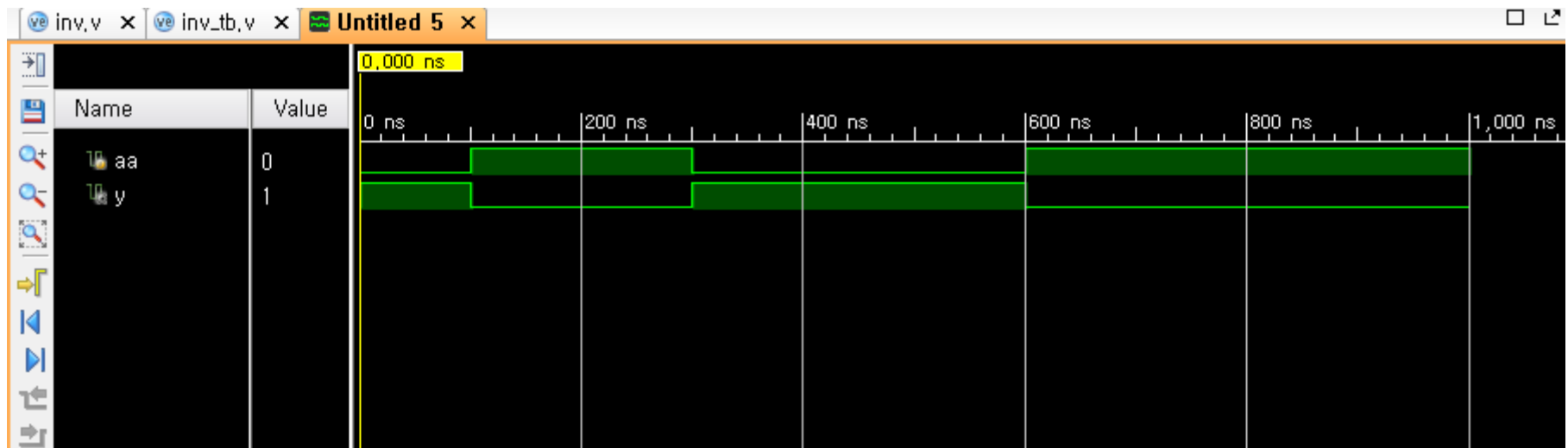
endmodule
```

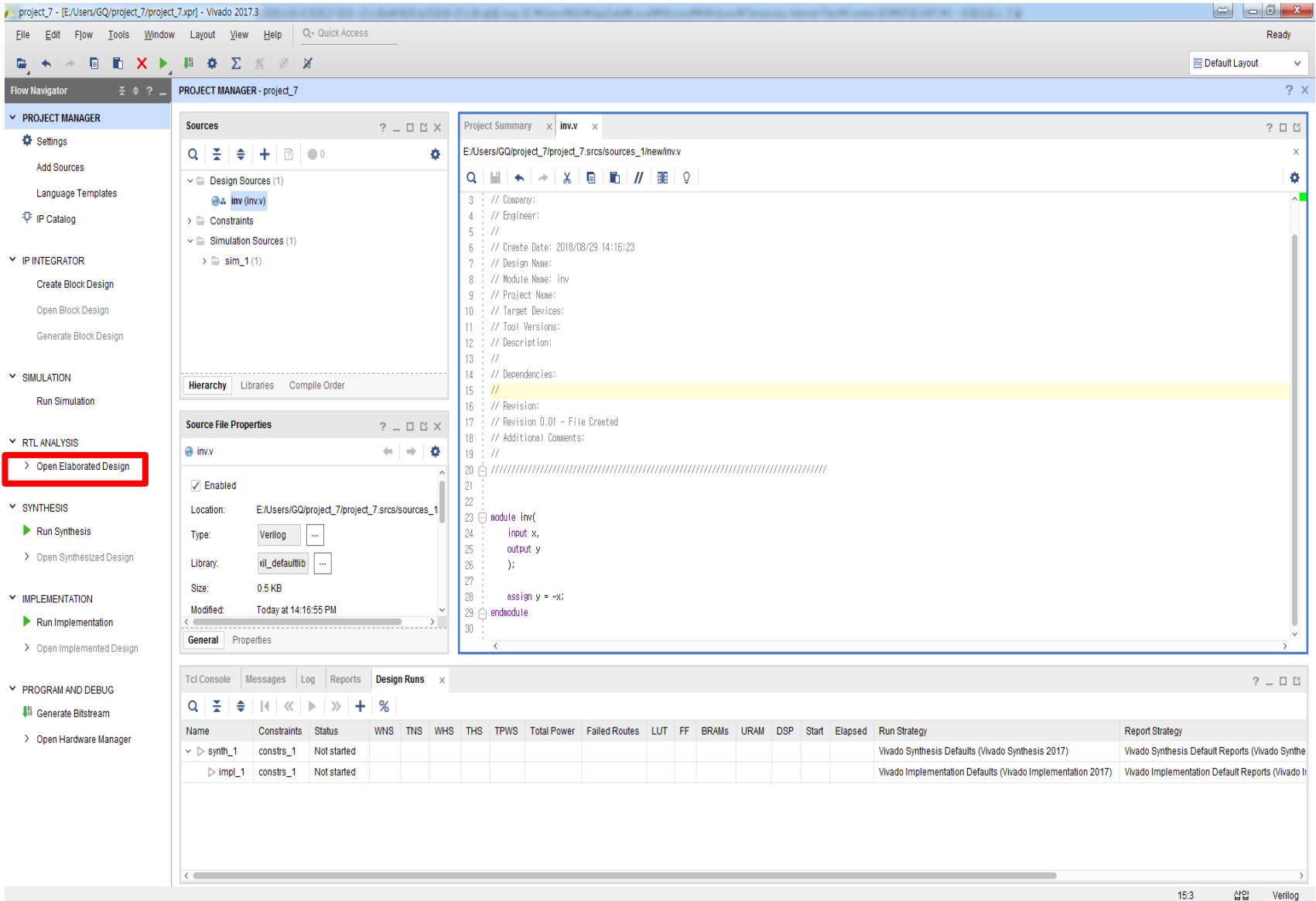












The screenshot displays the Vivado 2017.3 IDE interface. The **PROJECT MANAGER** panel on the left shows the project structure with **inv.v** selected under **Design Sources**. The **Source File Properties** panel for **inv.v** shows it is enabled and located at `E:/Users/GQ/project_7/project_7.srsrcs/sources_1`. The **Design Runs** panel at the bottom shows the status of the design runs.

**Design Runs Table:**

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMS	URAM	DSP	Start	Elapsed	Run Strategy	Report Strategy
synth_1	constrs_1	Not started															Vivado Synthesis Defaults (Vivado Synthesis 2017)	Vivado Synthesis Default Reports (Vivado Synthe
impl_1	constrs_1	Not started															Vivado Implementation Defaults (Vivado Implementation 2017)	Vivado Implementation Default Reports (Vivado Im

project\_8 - [E:/Users/GQ/project\_8/project\_8.xpr] - Vivado 2017.3

File Edit Flow Tools Window Layout View Help Q Quick Access

Flow Navigator

- Language Templates
- IP Catalog
- Package IP
- IP INTEGRATOR
  - Create Block Design
  - Open Block Design
  - Generate Block Design
- SIMULATION
  - Run Simulation
- RTL ANALYSIS**
  - Open Elaborated Design
    - Report Methodology
    - Report DRC
    - Report Noise
    - Schematic**
- SYNTHESIS
  - Run Synthesis
  - Open Synthesized Design
    - Constraints Wizard
    - Edit Timing Constraints
    - Set Up Debug
    - Report Timing Summary
    - Report Clock Networks
    - Report Clock Interaction
    - Report Methodology
    - Report DRC
    - Report Noise

ELABORATED DESIGN - xc7a75tfgg484-1 (active)

Sources Netlist x

- inv
  - Nets (2)
  - Leaf Cells (1)
    - y\_i (RTL\_INV)

Cell Properties

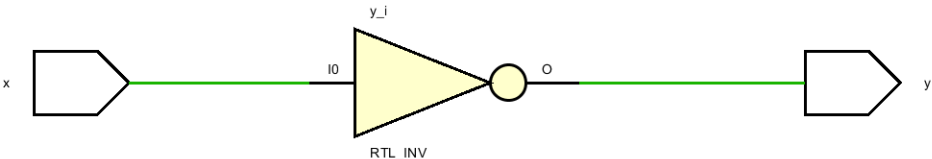
y\_i

Name: y\_i  
Reference name: RTL\_INV  
Type: RTL Gate  
Number of cell pins: 2  
Number of nets: 2

General Properties Nets Cell Pins

Project Summary x inv.v x Schematic x

1 Cell 2 I/O Ports 2 Nets



Tcl Console Messages Log Reports Design Runs x

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed	Run Strategy	Report Strategy
✓ synth_1	constrs_1	synth_design Complete!								1	0	0.00	0	0	8/29/18 5:05 PM	00:00:36	Vivado Synthesis Defaults (Vivado Synthesis 2017)	Vivado Synthesis Default
▶ impl_1	constrs_1	Not started															Vivado Implementation Defaults (Vivado Implementation 2017)	Vivado Implementation Default

## ◆ Buf gate

- (A)와 (B)의 Boolean 식을 비교
- (A)와 (B)의 Verilog 코딩
- (A)와 (B)의 Simulation을 통해 출력 결과 비교

