



**THE IMAGINATION UNIVERSITY PROGRAMME**

# **RVfpga Lab 1**

## **Vivado 프로젝트 제작**

## 1. 소개

RVfpga 시스템으로 작업 및 수정하려면 시스템을 정의하는 모든 Verilog, SystemVerilog, 헤더, 구성 및 텍스트 파일을 포함하는 프로젝트를 빌드해야 합니다. 이 실습에서는 이 과정에서 사용된 SoC에서 Digilent의 Nexys A7 FPGA 보드, -100T 버전을 (RVfpgaNexys) 대상으로 Vivado 프로젝트를 생성하는 방법을 보여 줍니다. (Nexys4 DDR 보드가 이미 있다면 이 프로젝트도 사용할 수 있습니다.) 동일한 단계를 수행하여 RVfpgaNexys를 수정하고 크기를 다시 조정할 수 있습니다.

**중요사항:** RVfpga Labs를 시작하기 전에 이미 Imagination University Program에서 제공하는 RVfpga Getting Started Guide를 먼저 이해하여야 합니다. (<https://university.imgtec.com/>).

예를 들어, 아직 설치하지 않은 경우 RVfpga 시작하기 가이드의 지침에 따라 Xilinx의 Vivado 및 Verilator를 설치합니다. 또한 Imagination의 University Program에서 다운로드한 **RVfpga** 폴더를 컴퓨터에 복사했는지 확인하십시오. RVfpga 폴더를 [RVfpgaPath]로 배치하는 디렉토리의 절대 경로를 참조합니다. [RVfpgaPath]/RVfpga/src 폴더에는 RVfpga 시스템에 대한 Verilog 및 SystemVerilog 소스가 포함되어 있으며, RISC-V SoC는 labs에서 사용하고 수정할 예정입니다. [RVfpgaPath]/RVfpga/Labs 폴더에는 Labs 1에서 10까지 사용할 일부 프로그램이 포함되어 있습니다.

## 2. RVfpgaNexys를 위한 Vivado 프로젝트 만들기

Xilinx의 Vivado Design Suite를 사용하여 시스템을 정의하는 Verilog 파일인 RTL을 사용하여 RVfpgaNexys 시스템을 구축합니다. 아래에 자세히 나와 있는 다음 단계에 따라 RVfpgaNexys 시스템을 구축하고 Nexys A7 FPGA 보드를 목표로 합니다.

### Step 1. Vivado 실행

### Step 2. 새 RTL 프로젝트 만들기

### Step 3. RTL 소스 파일과 Constraint 파일 추가

### Step 4. Nexys A7를 타겟 보드에서 선택

### Step 5. RVfpgaNexys를 Top Module로 설정하고 common\_defines.vh를 global로 설정합니다.

### Step 6. 비트스트림 생성

### Step 1. Vivado 실행

RVfpga 시작하기 가이드에 설명된 대로 Vivado를 컴퓨터에 설치하지 않은 경우 지금 설치하십시오. 보드 파일도 설치해야 합니다.

이제 Vivado를 실행합니다(Linux에서는 terminal을 열고 vivado를 입력합니다. Windows에서는 시작 메뉴에서 Vivado를 엽니다). Vivado 환영 화면이 열립니다. Create Project를 클릭합니다(그림 1 참조).

<sup>1</sup> 이 책장에서, 우리는 Vivado 2019.2. 를 사용합니다. 최신 버전의 Vivado 를 사용하더라도, 우리는 지금 버전을 사용하기를 강력하게 추천 드립니다.

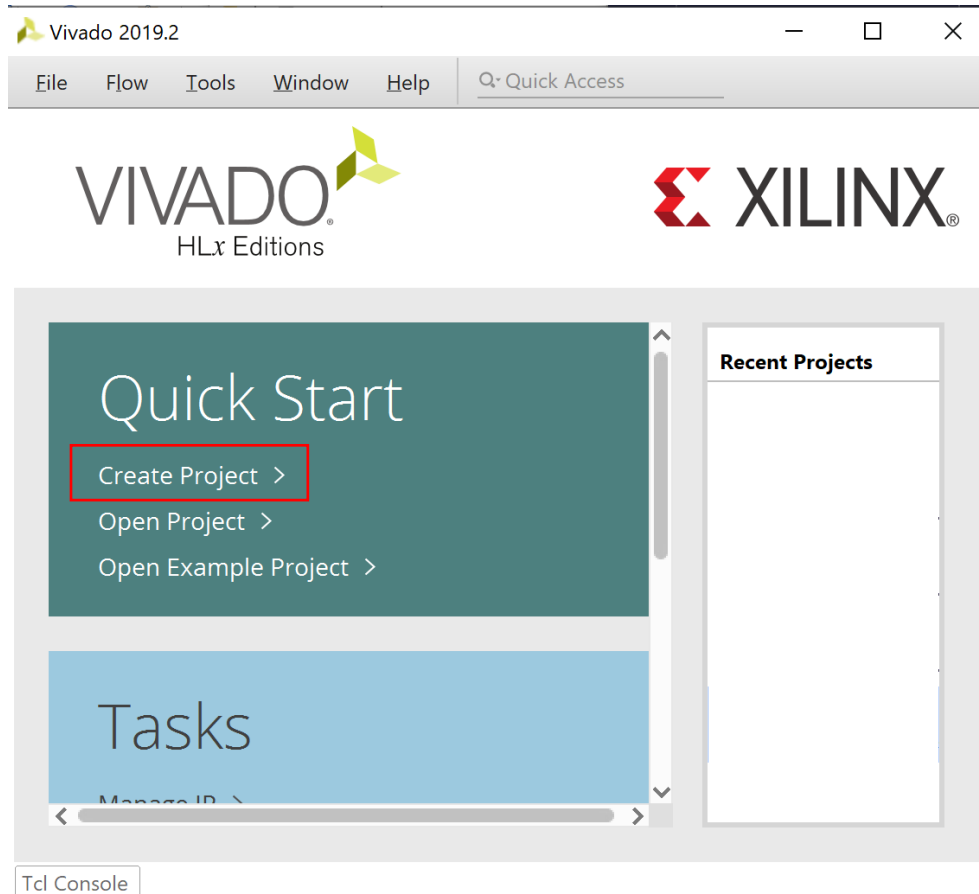


그림 1. Vivado 시작 화면: 프로젝트 생성

## STEP 2. 새 RTL 프로젝트 생성

새 Vivado 프로젝트 만들기 마법사가 열립니다(그림 2 참조). 다음을 클릭합니다.

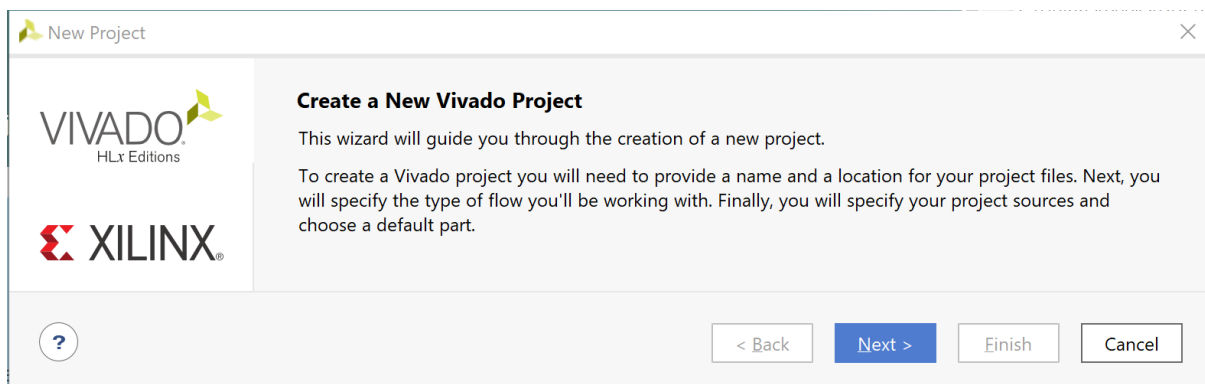
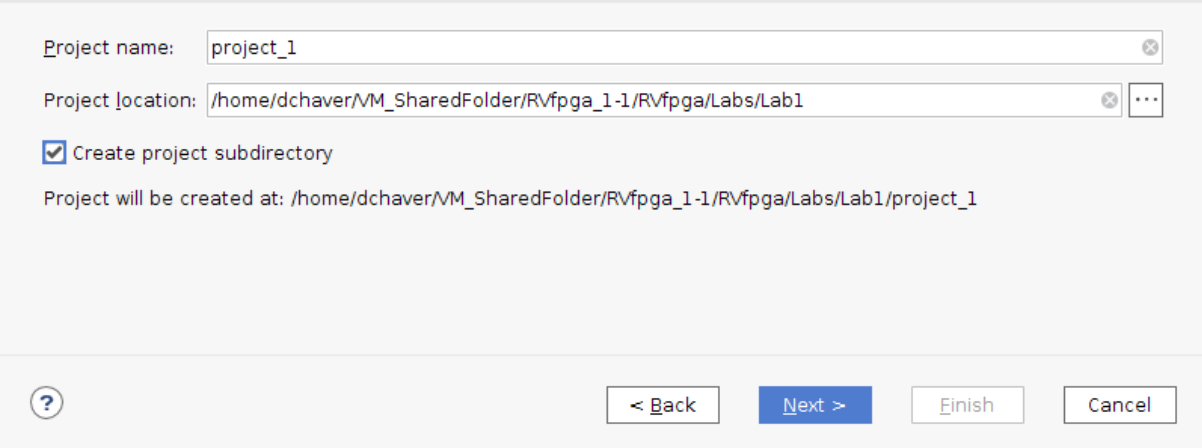


그림 2. 새로운 Vivado 프로젝트 마법사 만들기

Project1 을 호출하여 `[RVfpgaPath]/RVfpga/Labs/Lab1` 폴더에 배치합니다. *Create project subdirectory* 옵션을 선택하시고, Next 를 클릭합니다(그림 3 참조).

### Project Name

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

Project name:

Project location:

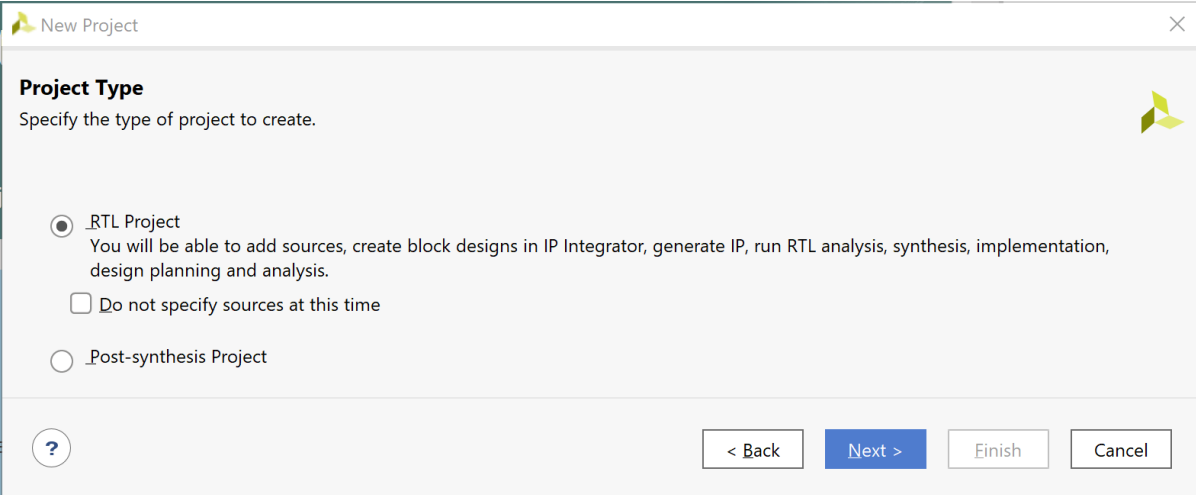
☒ Create project subdirectory

Project will be created at: /home/dchaver/VM\_SharedFolder/RVfpga\_1-1/RVfpga/Labs/Lab1/project\_1

? < Back Next > Finish Cancel

그림 3. 프로젝트 이름

프로젝트 유형을 RTL 프로젝트로 선택하고 다음을 클릭합니다(그림 4 참조).



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

? < Back Next > Finish Cancel

그림 4. RTL 프로젝트

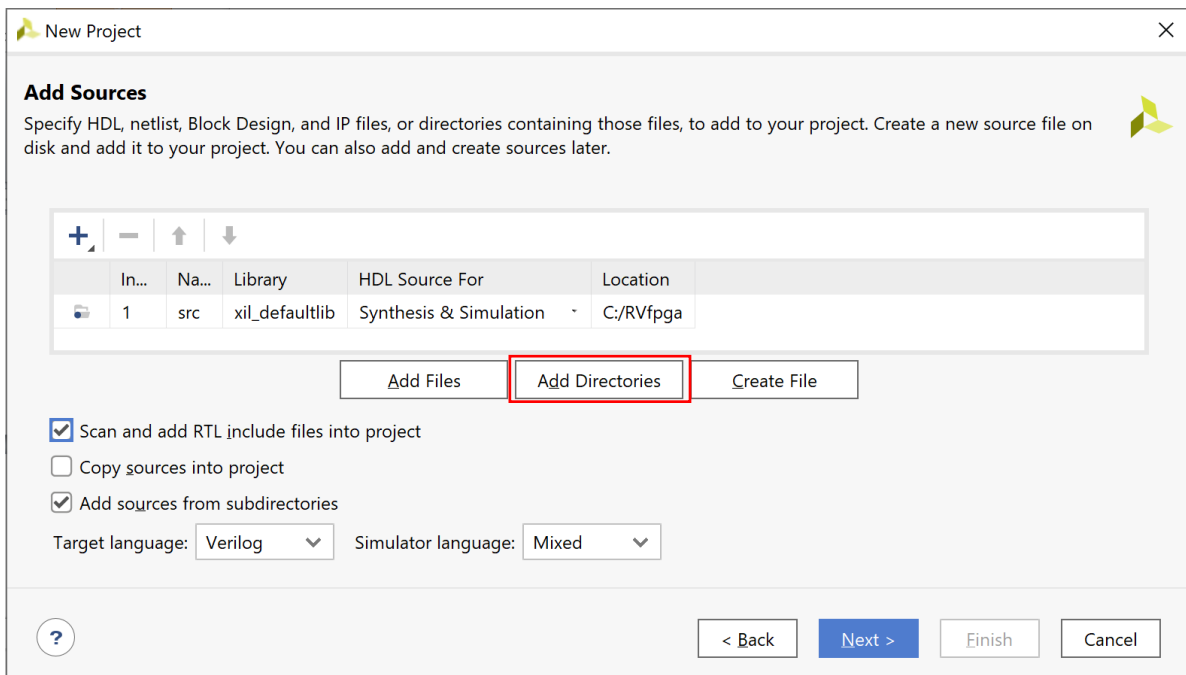
### Step 3. RTL source files 과 Constraint files 추가

소스 추가 창에서 디렉터리 추가를 클릭하고 `[RVfpgaPath]/RVfpga/src` 를 선택합니다(그림 5 참조).

다음 옵션을 모두 선택해야 합니다(그림 5 참조):

- RTL 포함 파일 검색 및 프로젝트에 추가
- 하위 디렉터리의 소스 추가

그리고 다음을 누릅니다.



**그림 5. 소스 추가**

이제 시스템에 대한 제약 조건을 추가합니다. 이러한 파일은 신호 이름을 보드의 핀에 매핑합니다. 예를 들어 Nexys A7 FPGA 보드의 LED는 PCB의 트레이스를 통해 보드의 FPGA 핀에 연결됩니다. Vivado는 이 사실을 알아야 RTL의 올바른 신호 이름을 올바른 FPGA 핀에 매핑할 수 있습니다. 예를 들어, Xilinx 설계 제약 조건 파일인 `[RVfpgaPath]/RVfpga/src/rvfpganexys.xdc` 파일의 다음 줄은 FPGA 핀 H17이 최소의 LED (`o_led[0]`)에 매핑이 되고 LVCMOS 3.3V 신호를 사용함을 나타냅니다:

```
set_property -dict { PACKAGE_PIN H17  IOSTANDARD LVCMOS33 }
[get_ports { o_led[0] }]
```

신호 이름 `o_led`는 Verilog 코드에 Nexys A7 보드의 LED를 구동하는 데 사용되는 이름입니다.

Constraints 추가 창에서 파일 추가를 클릭하고 다음 두 파일을 선택합니다(그림 6 참조):

```
[RVfpgaPath]/RVfpga/src/rvfpganexys.xdc
[RVfpgaPath]/RVfpga/src/LiteDRAM/liteDRAM.xdc
```

그리고 다음을 눌러주세요

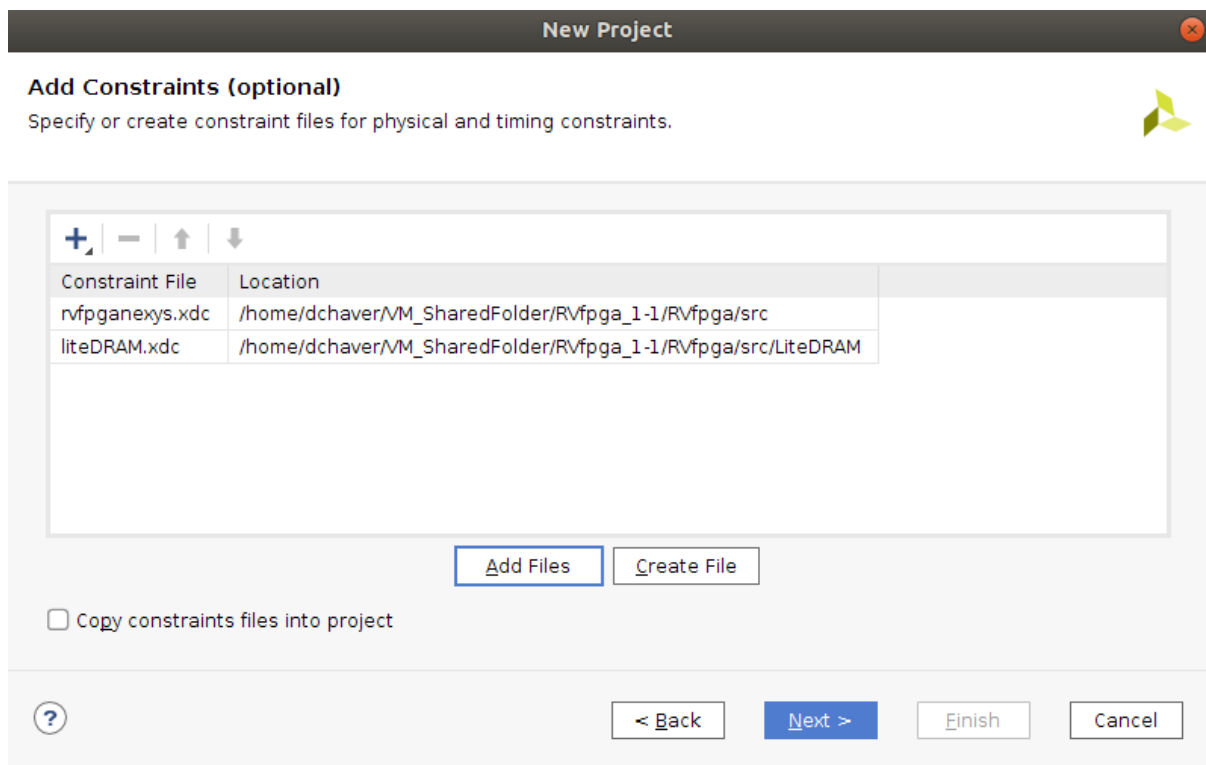


그림 5. Constraints 추가

#### STEP 4. Nexys A7 을 타겟 보드로 선택합니다.

Default Part 창에서 Boards 를 클릭한 다음 Nexys A7-100T 를 선택합니다(그림 7 참조).

검색 상자를 사용하여 결과를 줄일 수 있습니다. 또한 실제 대상 FPGA 의 이름이 부품 열 xx7a100tcsg324-1 에 나열되어 있습니다. 이는 CSG(chip-scale grid) 패키지와 324 핀이 장착된 100k 등가 게이트를 갖춘 Xilinx Artix-7 FPGA 임을 나타냅니다.

다음을 눌러주세요.

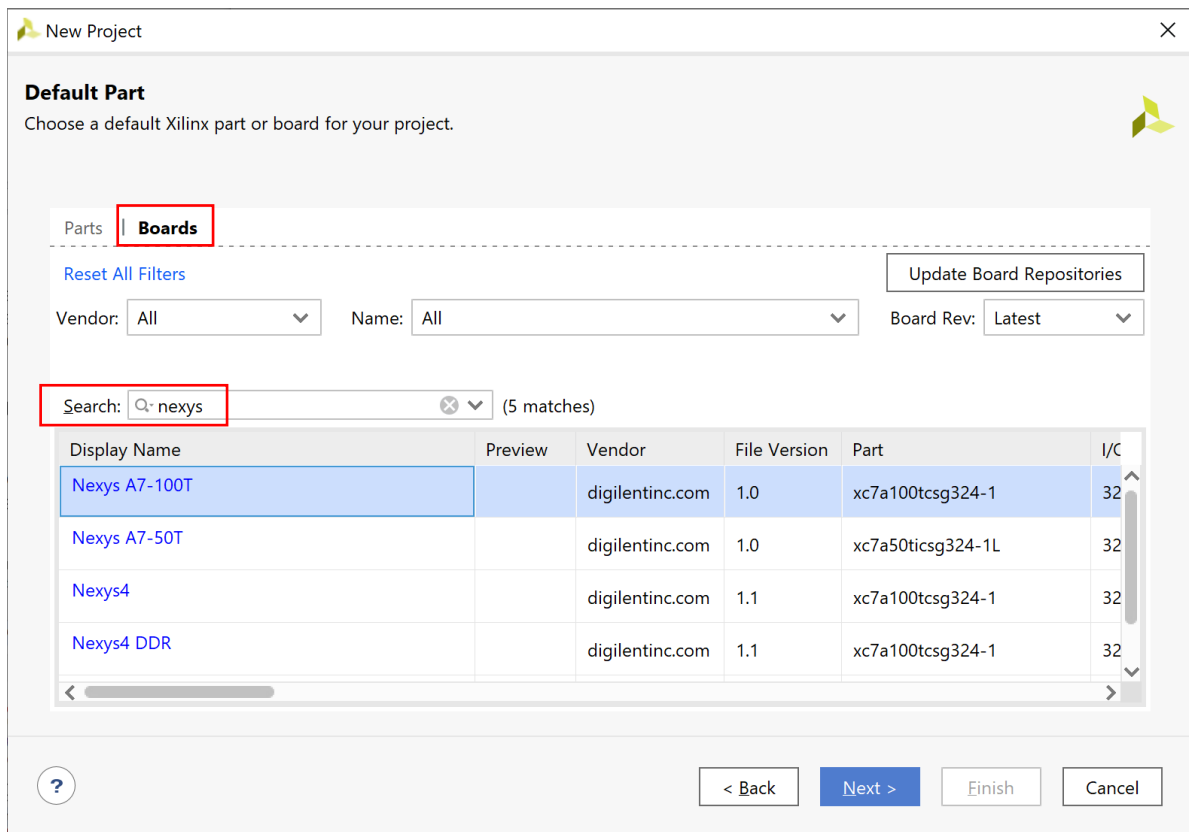


그림 6. 대상 보드 선택: 넥시스 A7-100t

새 프로젝트 요약 창에서 마침을 클릭합니다(그림 8 참조).

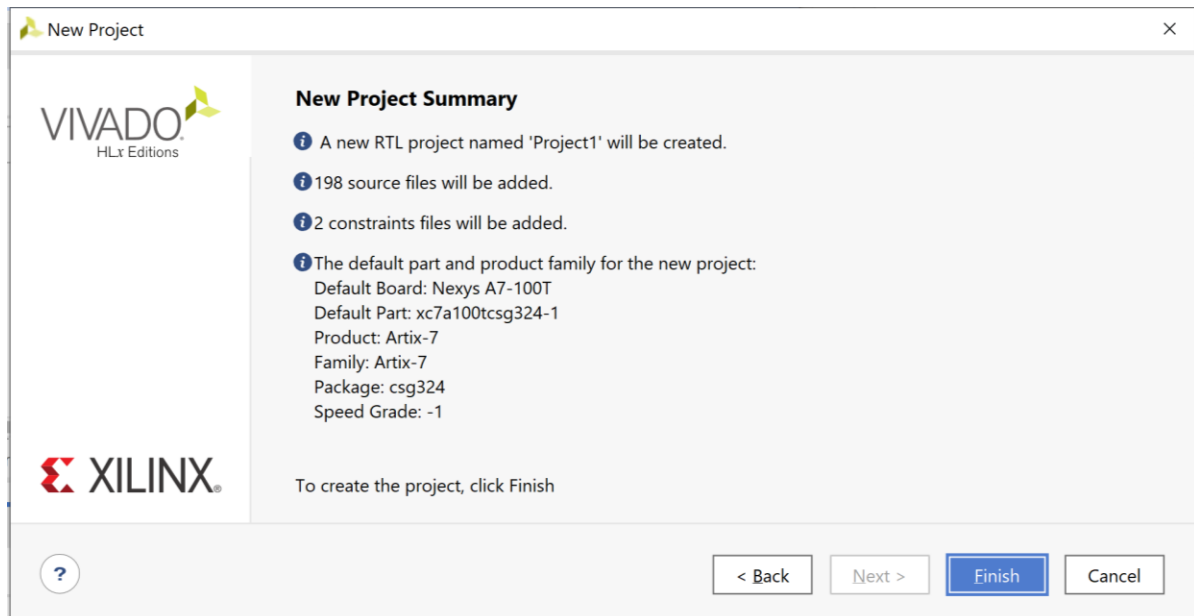


그림 7. 새 프로젝트 요약 창

프로젝트 설정이 완료되면 구문 오류가 있는 파일이 있음을 나타냅니다. 이 오류는 다음 단계에서 해결됩니다.

**Step 5. Project Configuration:** rvfpganexys 를 최상위 모듈로 설정하고 common\_defines.vh 파일을 전역으로 설정하고, boot\_main.mem 을 프로젝트에 추가하여 Pulp Platform 폴더를 포함합니다.

**Set rvfpganexys as Top Module:** 이제 rvfpganexys 모듈은 상단 모듈로 설정합니다. 소스 창의 디자인 소스에서 아래로 스크롤하고 rvfpganexys 모듈을 마우스 오른쪽 버튼으로 클릭한 다음, Top 으로 설정합니다(그림 9 참조). 보여주는 바와 같이 검색 상자에 이 이름을 입력하여 rvfpganexys 모듈을 찾을 수도 있습니다. 이는 rvfpganexys 를 계층 구조에서 가장 높은 수준의 모듈로 설정하고 FPGA 에 합성 및 구현할 대상을 설정합니다. rvfpganexys 를 최상위 모듈로 설정하면 계층이 업데이트됩니다.

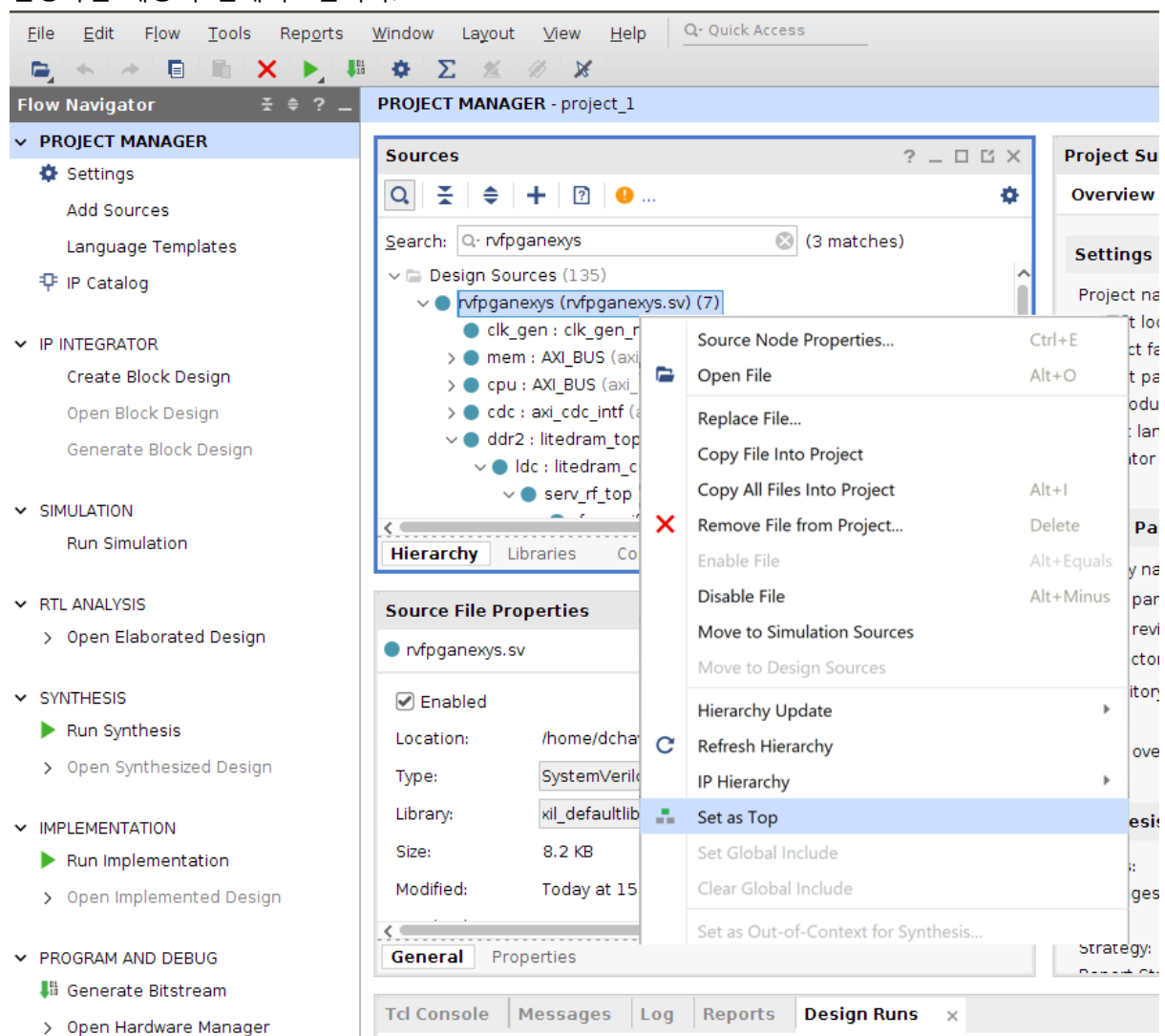


그림 9. rvfpganexys 를 Top Module 로 설정

**Set file common\_defines.vh as Global Include:** 이제 디자인 소스의 소스 창에서 비 모듈 파일 그룹을 확장하고 common\_defines.vh 를 클릭합니다. 그러면 파일의 속성이 원본 창 바로



아래의 원본 파일 속성 창에서 열립니다. Global Include 를 클릭하여 해당 확인란을 선택합니다(그림 10 참조). 이제 계층이 업데이트되고 해당 파일이 디자인 소스/Global Include 에 포함됩니다. 구문 오류 파일은 이후 단계에서 사라집니다.

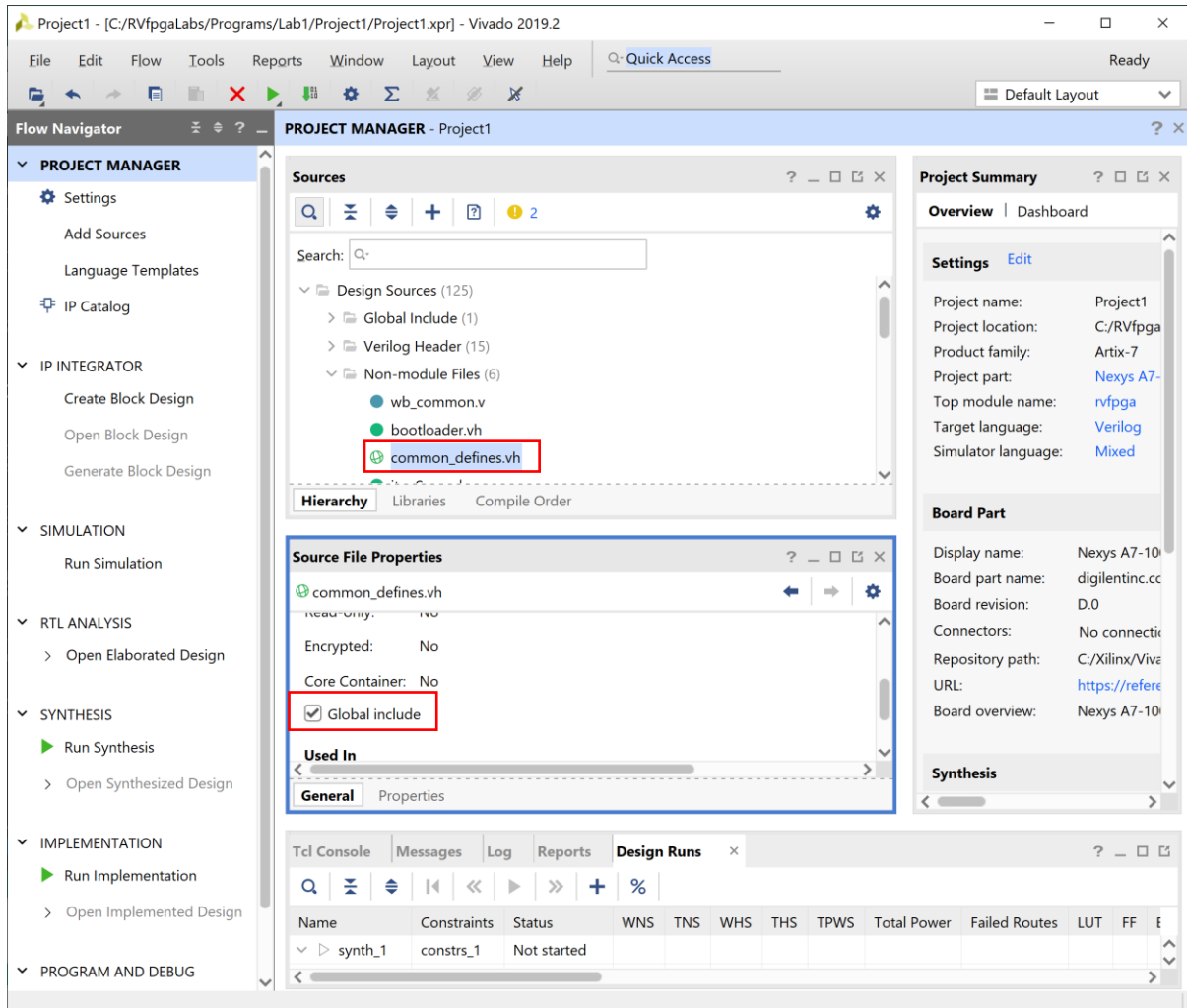
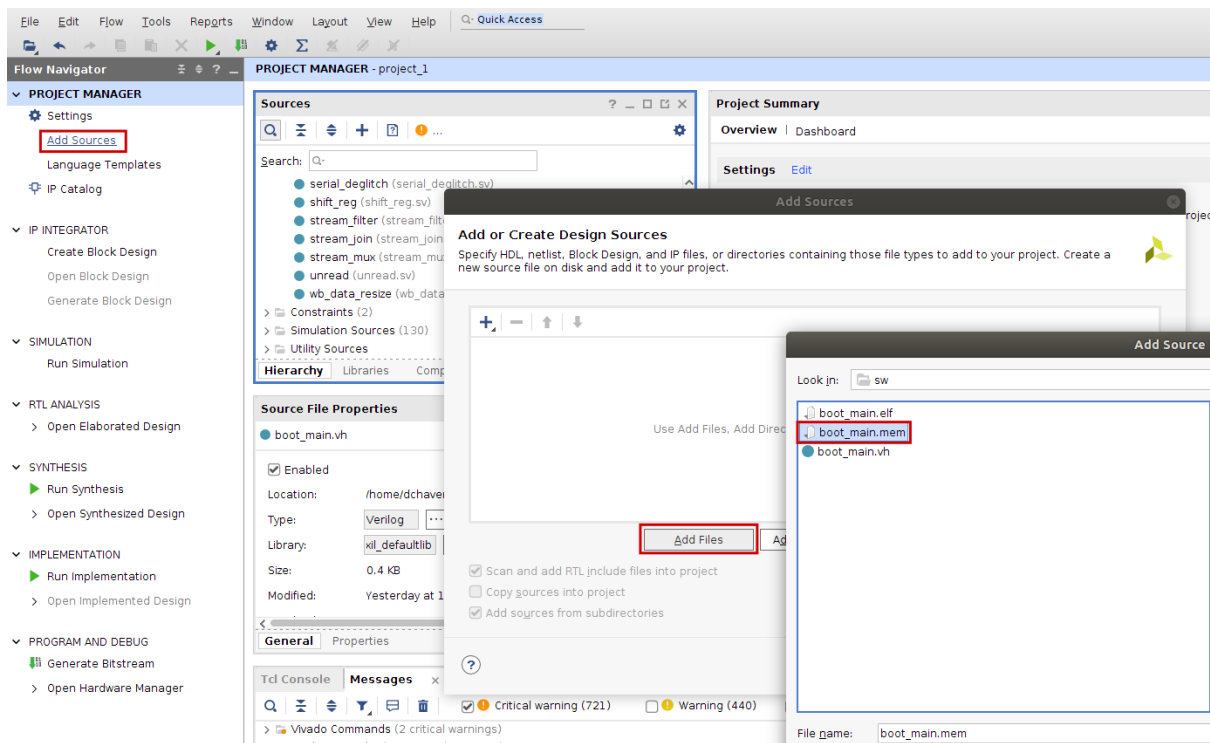


그림 10. common\_defines.vh 를 Global include 파일로 설정

**Add boot main.mem to the project:** Flow Navigator 창에서 Add Sources 를 클릭하고 기본 옵션 ("Add or create design sources")을 그대로 둔 다음 Add Files 를 클릭합니다 (그림 11 참조). [RVfpgaPath]/RVfpga/src/SweRVolfSoC/BootROM/sw 로 이동하고 boot\_main.mem 을 선택합니다 (그림 11 참조). 계층이 업데이트되고 해당 파일을 디자인 소스/메모리 파일에 포함합니다.





**Figure 8. Add Memory File boot\_main.mem**

이 파일 (boot\_main.mem)은 `[RVfpgaPath] /RVfpga/src/rvfpganexys.rv` 파일의 매개 변수로 호출하여 SoC의 부팅 ROM을 초기화하는 데 사용됩니다.

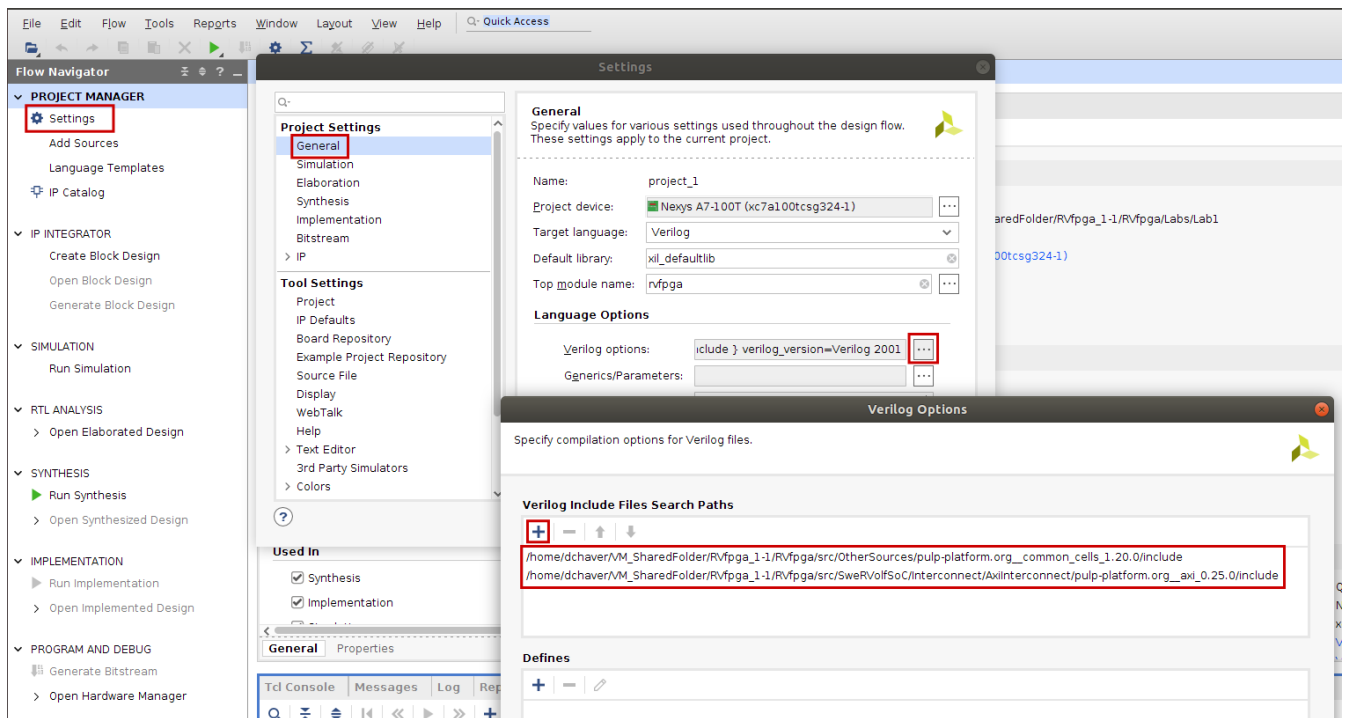
```
25 module rvfpga
26   #(parameter bootrom_file = "boot_main.mem")
```

시작 안내서의 섹션 6.A에는 이 파일에 대한 자세한 정보가 있습니다.

**Include folders:** 마지막으로 Pulp 플랫폼용 폴더 2개를 포함합니다 (그림 12 참조). Flow Navigator 창에서 설정을 클릭하고 열리는 창에서 General을 클릭한 다음 Verilog option

()을 클릭합니다. 새 창에서 를 클릭하고 디렉토리를 찾아 다음 두 개의 포함 디렉토리를 추가하십시오.

```
[RVfpgaPath] /RVfpga/src/SweRVolfSoC/Interconnect/AxiInterconnect/pulp-platform.org__axi_0.25.0/include
[RVfpgaPath] /RVfpga/src/OtherSources/pulp-platform.org__common_cells_1.20.0/include
```



**Figure 12. Include folders for the Pulp Platform**

## Step 6. 비트스트림 생성

이제 그림 13 과 같이 Flow → Generate Bitstream(비트 스트림 생성)을 클릭합니다. 사용 가능한 구현 결과가 없다는 창이 나타나고 합성 및 구현을 시작할 것을 요청할 수 있습니다(그림 14 참조). 예를 클릭합니다. 그런 다음 실행 창에서 확인을 클릭합니다(그림 15 참조). 이 단계는 RVfpgaNexys(프로젝트의 Verilog 및 System Verilog 파일로 정의됨)를 합성하고 이를 FPGA 에 매핑한 후 비트 스트림을 생성합니다. 이 프로세스는 일반적으로 컴퓨터 속도에 따라 20-50 분이 걸립니다.

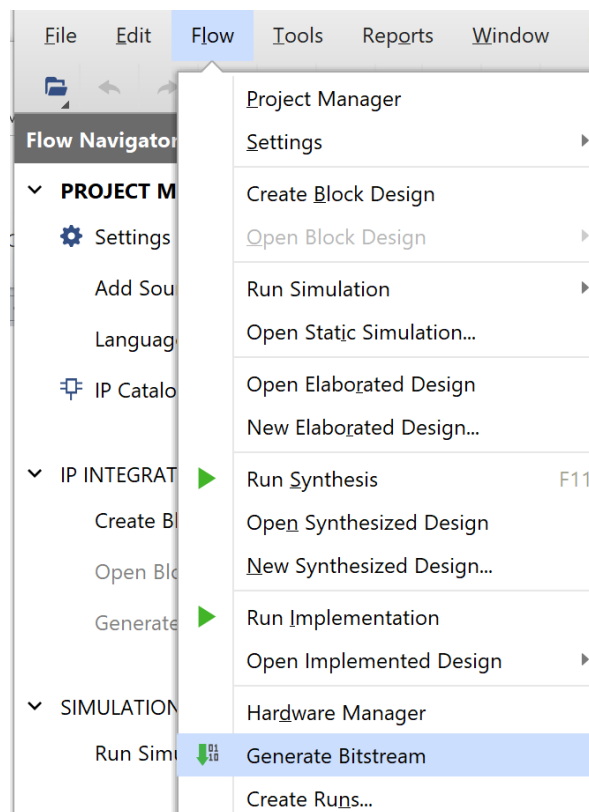


그림 13. 비트스트림 생성

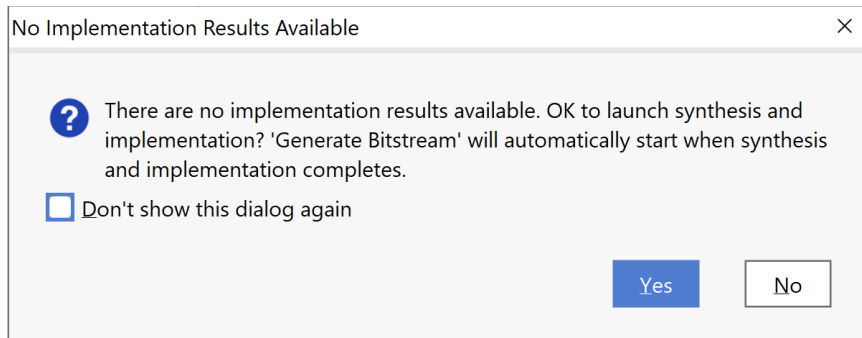
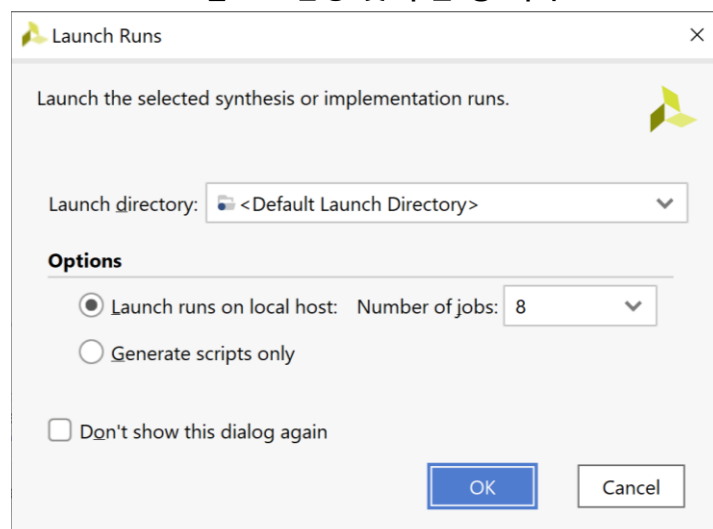


그림 14. 합성 및 구현 창 시작



## 그림 15. 실행 시작

비트 스트림 생성 후 그림 14 와 같이 창이 나타납니다. 오른쪽 상단 모서리에 있는 ✕ 버튼을 클릭하여 창을 닫습니다.

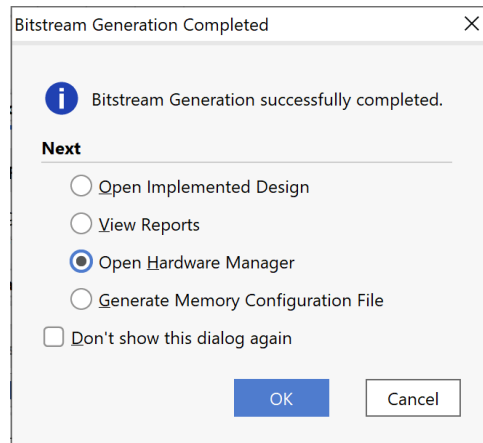


그림 16. 비트스트림 생성 완료

이제 RVfpgaNexys 시스템을 직접 build 했으므로 Labs 6-10 에서 RVfpgaNexys 를 수정하여 rebuild 할 수 있습니다. 지금은 PlatformIO 를 사용하여 프로그램을 다운로드하고 실행함으로써 방금 구축한 RVfpgaNexys 시스템을 사용할 수 있습니다.

Nexys A7 보드에 RVfpgaNexys 를 다운로드하기 위하여 플랫폼 IO 를 사용하는 것이 좋습니다. 자세한 내용은 RVfpga Getting Starting Guide(GSG)의 섹션 6.A 에 자세히 설명되어 있습니다. GSG 의 다른 예제 (6.B ~ 6.H) 에서 설명한 대로, RVfpgaNexys 시스템을 Nexys A7 보드의 FPGA 에 다운로드한 후, PlatformIO 를 사용하여 RVfpgaNexys 에 프로그램을 다운로드하고 실행/디버깅할 수 있습니다.

또한 Verilator(HDL 시뮬레이터)를 사용하여 RVfpga Getting Starting Guide 의 섹션 7 에 설명한 대로 RVfpgaSIM 에서 실행되는 프로그램을 시뮬레이션할 수 있습니다. 이러한 RTL 레벨 시뮬레이션을 통해 소프트웨어 프로그램이 실행될 때 low 레벨의 하드웨어 신호를 확인할 수 있습니다. RVfpga 시스템을 확장하고 변경 사항을 테스트 및 디버깅할 때 Labs 6-10 에서 Verilator 를 많이 사용하게 됩니다.