



ECE4700J Computer Architecture

Summer 2024

Lab #1 Getting Started with Vivado and SystemVerilog

Due: 11:59pm (Beijing time) May. 28th, 2024

Logistics:

- This lab is an individual exercise.
- Please use the discussion board on Piazza for Q&A.
- All reports and code (if available) MUST be submitted to the assignment of Canvas.
- Internet usage is allowed and encouraged.
- Late submission penalty is 10% per day.

Contents

1 Overview	2
2 Xilinx Vivado Introduction and Tutorial	2
2.1 Introduction	2
2.2 Vmware Horizon Client Usage	2
2.3 Vivado Installation	4
3 Create Project and Do Verilog Programming	7
4 Design a Round Robin Bus Arbiter	11
4.1 Introduction	11
4.2 Assignment	12
4.2.1 Design	12
4.2.2 Test	12
5 Debugging of SystemVerilog Design	14
5.1 Introduction	14
5.2 Assignment	14
6 Deliverables	15
7 Grading policy	15

1 Overview

In this lab, you will install Vivado and set up the environment for SystemVerilog programming, and then do some simple Verilog designs.

- Install the Xilinx Vivado ML Edition 2021.2.
- Set up the environment correctly.
- Be able to implement some simple SystemVerilog designs and synthesize/simulate it using Vivado.
- Study different kinds of bugs in hardware design.
- Be able to implement testbenches that both have good coverage and short execution time.

2 Xilinx Vivado Introduction and Tutorial

2.1 Introduction

Xilinx Vivado ML Edition - 2021.2 is a stable version of Xilinx software. In this semester, it's mainly used for Verilog programming, synthesizing and simulation.

As the version of vivado is somehow important and we do not want any extra trouble on software problems, every student has already got their accounts created in remote server Vmware Horizon Client with vivado 2021.2 and gem5 already downloaded. It is more recommended that everyone can use the remote server.

2.2 Vmware Horizon Client Usage

Vmware Horizon can be easily accessed with the link <https://vdi.ji.sjtu.edu.cn/?includeNativeClientLaunch=true>



VMware Horizon

您可以使用 VMware Horizon Client 或通过浏览器连接到桌面和应用程序。

VMware Horizon Client 提供了更佳的性能和功能。

1

启动本机客户端



选中此处以跳过该屏幕并始终使用本机客户端。

2

VMware Horizon
HTML Access



选中此处以跳过该屏幕并始终使用 HTML Access。

3

单击此处以下载 VMware Horizon Client

- * button 1 to open the vmware horizon if you have already got one.
- * button 2 to open the online vmware horizon webpage.
- * button 3 to download a local vmware horizon client.

Then this is what you shall see.



2.3 Vivado Installation

1. Download Xilinx Vivado ML Edition - 2021.2 from Downloads (<https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/2021-2.html>). You need to first sign up at <https://www.amd.com/en/registration/create-account.html> if this is the first you use a Xilinx software.

Version

2022.1

2021.2

2021.1

2020.3

[Vivado Archive](#)

[ISE Archive](#)

[CAE Vendor Libraries Archive](#)

Vivado ML Edition - 2021.2 Full Product Installation

Important Information

Vivado ML 2021.2 is now available for download:

- New device support for Artix® UltraScale+™: XCAU20P and XCAU25P
- Improved Intelligent Design Runs for push-button timing closure
- New example designs available in Vivado®
- Ease of use enhancements for HLS flows

We **strongly recommend** to use the web installers as it reduces download time and saves significant disk space.

Please see [Installer Information](#) for details.

Note:

- Download verification is only supported with Google Chrome and Microsoft Edge web browsers.
- Beginning this release we will be offering only 2 Editions for Vivado ML. Please go to [product page](#) for more details.
- Vivado ML 2021.1 and later versions require upgrading your license server tools to the Flex 11.17.2.0 versions.

Download Includes

Download Type

Last Updated

Answers

Documentation

Support Forums

Vivado ML Edition

Full Product Installation

Oct 27, 2021

[2021.x - Vivado Known Issues](#)

[Release Notes](#)
[OS Support Update](#)
[What's New in Vivado](#)

[Installation and Licensing](#)

Based on your OS type and network, you can choose to download the “Self-Extracting Web Installer” or “All OS Unified Installer Single-File Download (SFD)”.

📄 [Xilinx Unified Installer 2021.2: Windows Self Extracting Web Installer \(EXE - 212.41 MB\)](#)

MD5 SUM Value : 76b60fc6a74338066f4e4dd0a855ec93

Download Verification ⓘ

[Digests](#)[Signature](#)[Public Key](#)

📄 [Xilinx Unified Installer 2021.2: Linux Self Extracting Web Installer \(BIN - 272.8 MB\)](#)

MD5 SUM Value : d4fe2978f735e4353f6ccff3405b488b

Download Verification ⓘ

[Digests](#)[Signature](#)[Public Key](#)

📄 [Xilinx Unified Installer 2021.2 SFD \(TAR/GZIP - 71.9 GB\)](#)

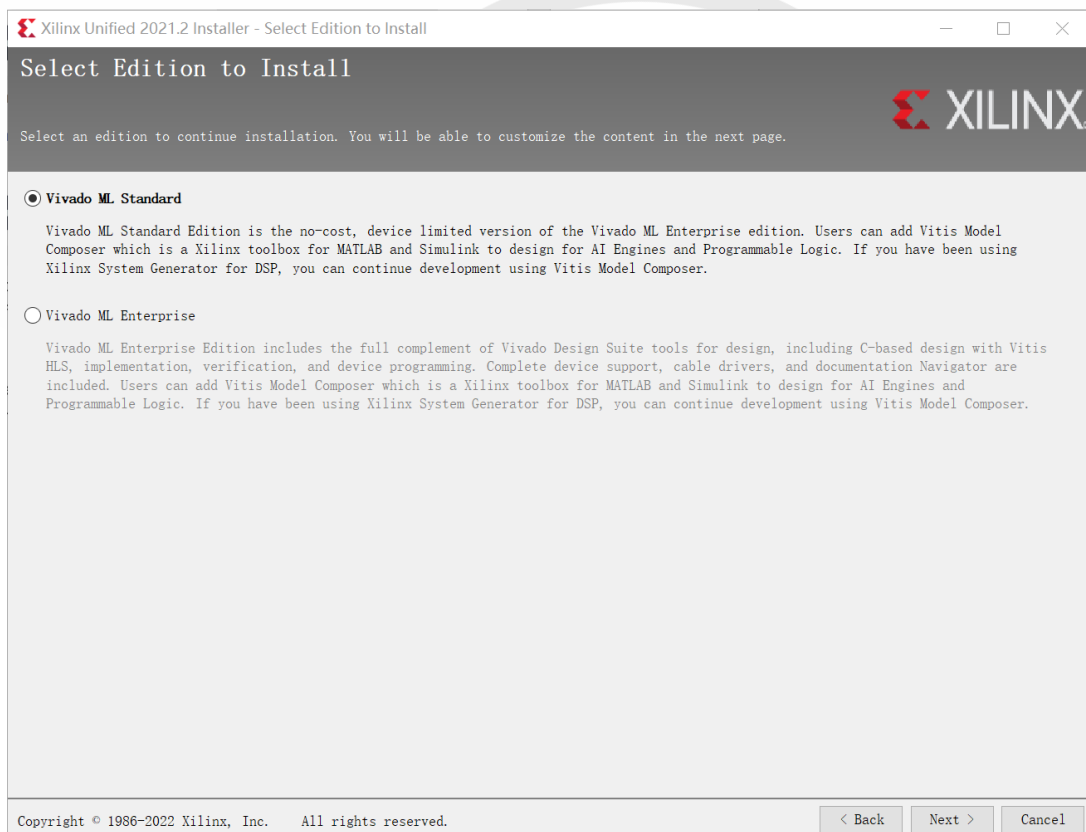
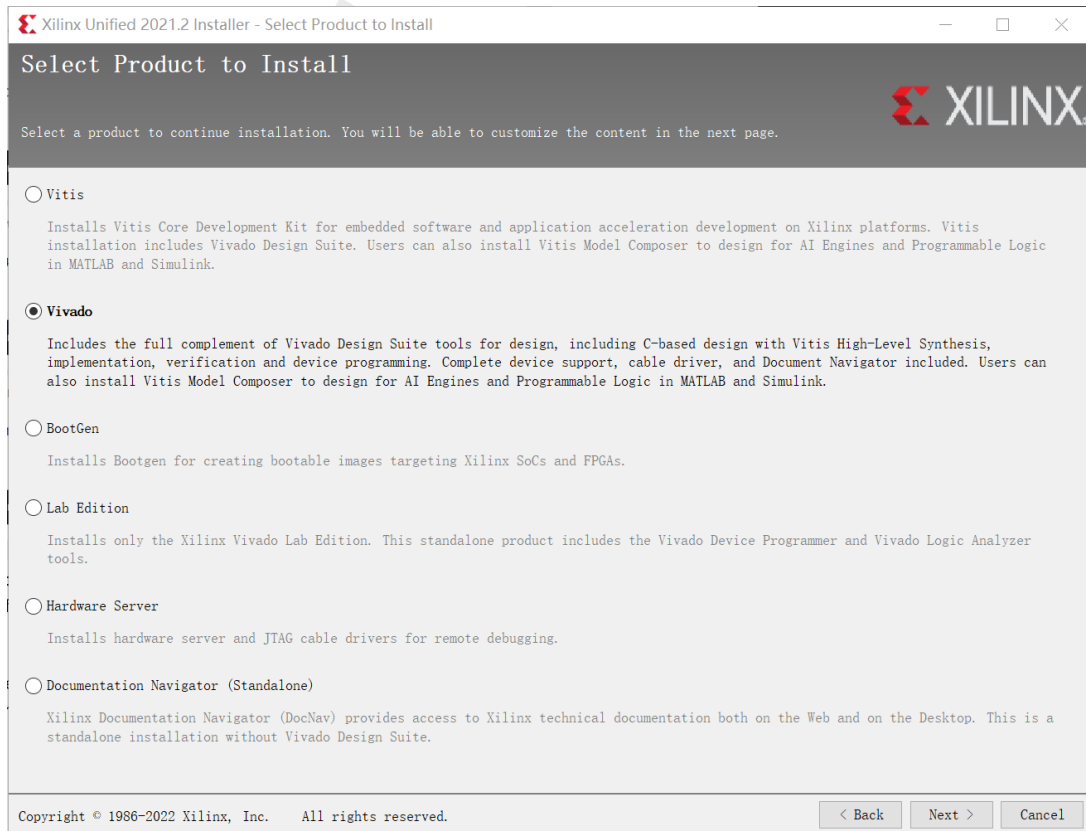
MD5 SUM Value : c6f91186f332528a7b74a6a12a759fb6

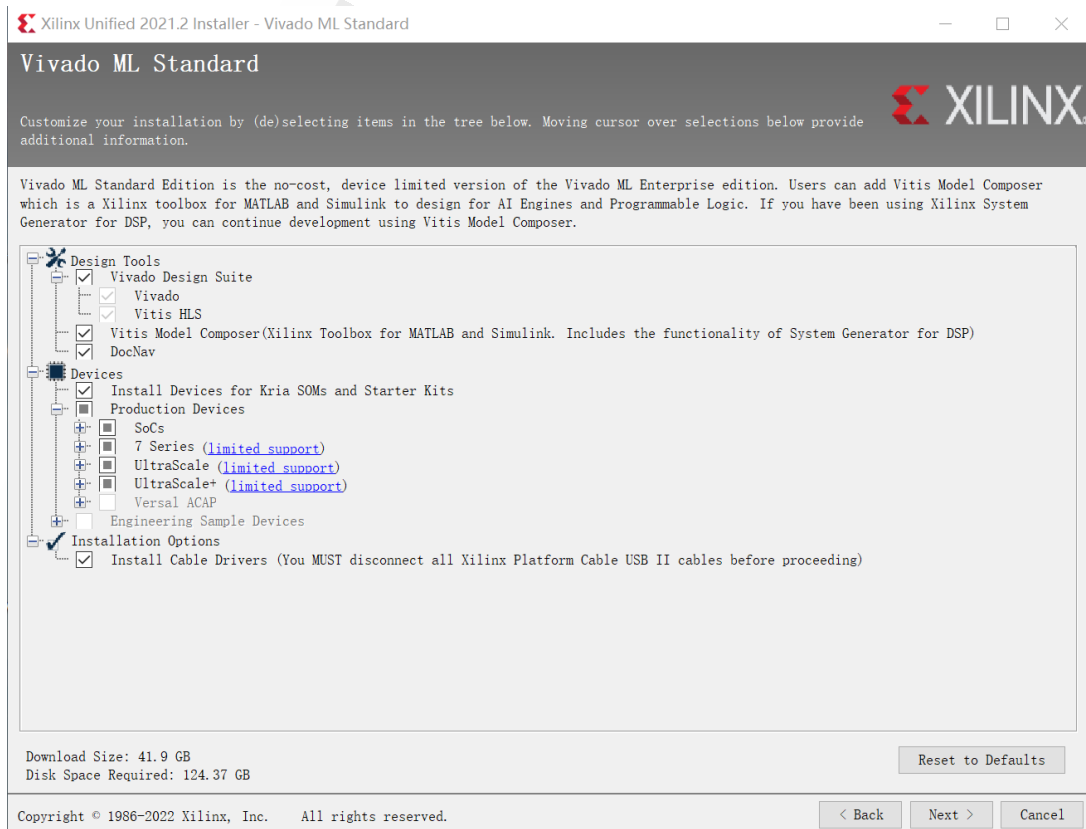
Download Verification ⓘ

[Digests](#)[Signature](#)[Public Key](#)

The former installer will download the main program by itself and the latter one has contained all necessary files and you needn't download anything more. Here, we use the former one to continue demo.

2. Execute the installer and you need to sign in. Choose Vivado ML Standard which is free. Default tools are enough for our labs.

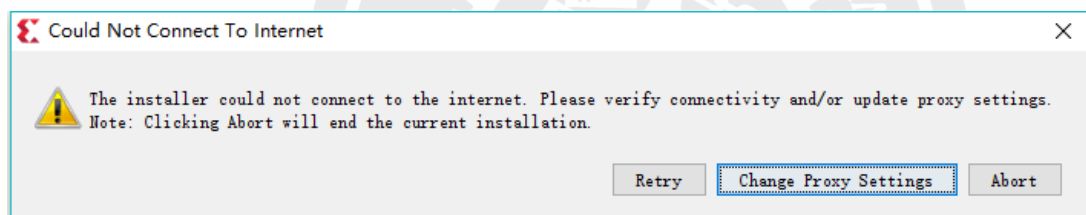




And then you can specify the installation directory and start installing.

Tips:

a) When downloading, there may be warnings as the figure below, if the installer is still running and the progress bar is progressing, please just ignore it or click “Retry” several minutes later, it will not affect the downloading process.



b) If you find the downloading speed to be slow, connect to SJTU VPN may help.

c) Start early. The installation of Vivado does take several hours.

3 Create Project and Do Verilog Programming

1. Open Vivado to open the software. If you are using Windows, just double click the shortcut of Vivado.

If you are using Linux, you need to first source the script by typing this in the

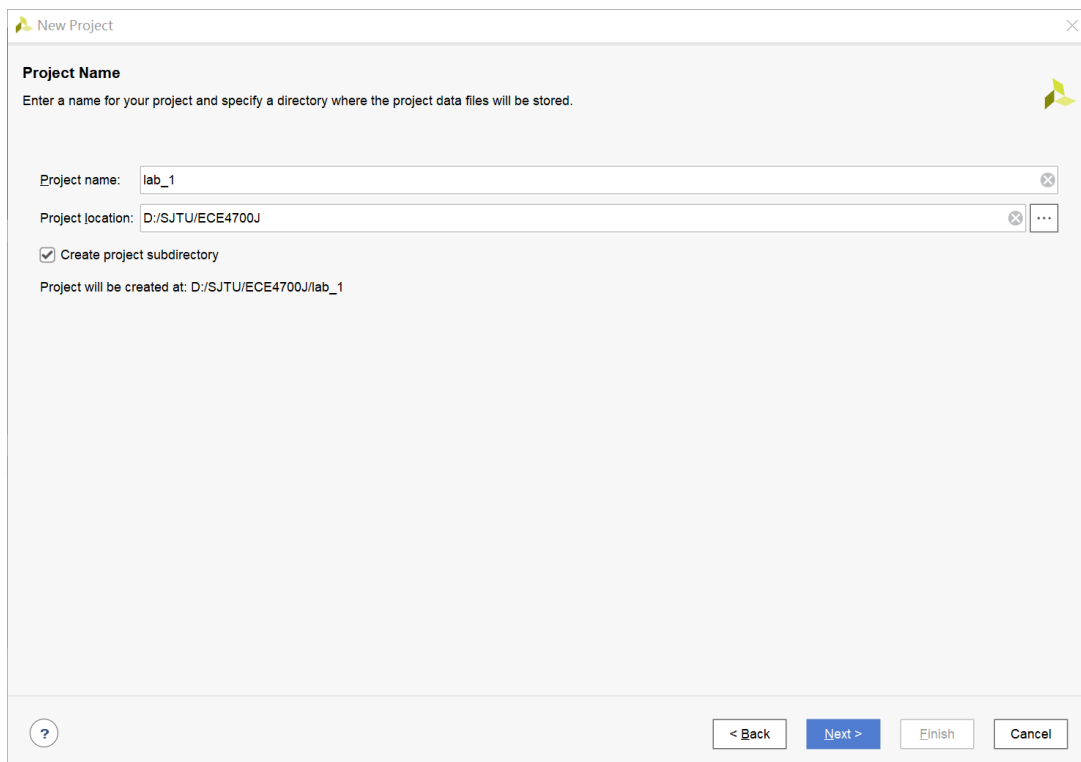
terminal

```
1 source ${PATH_TO_VIVADO_INSTALLATION_PATH}/Vivado/2021.2/settings64.sh
```

and then you can just in this command to activate this software

```
1 vivado
```

2. After the GUI is opened, click **Create Project** in the left side panel
3. Follow the navigator to choose **Project location** and **Project name**:



4. Set the **Project type** as **RTL Project** and click **Next**

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.

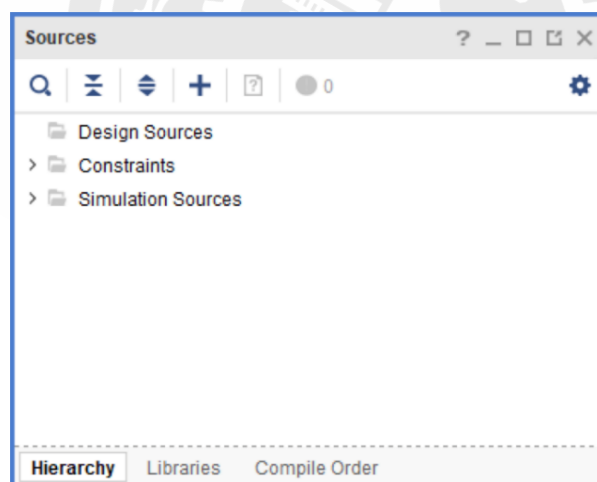
☐ **Imported Project**
Create a Vivado project from a Synplify, XST or ISE Project File.

☐ **Example Project**
Create a new Vivado project from a predefined template.

- you can skip **Add Source** and **Add Constraint** at this moment
- Then you need to choose the **Default part**. Different parts in Vivado means different FPGA target boards. In ECE4700J course, since we only need to synthesize and simulate the design (we don't need to do on-board experiments), you can feel free to select your favorite part and it doesn't matter. However, you must ensure that the resources (i.e., the LUTs, DSPs, RAMs, hardware resources) on this part is enough for your design. We recommend you to choose big boards as the part. For example, we recommend you to use the part **xczu7eg-ffvf1517-2-i** (if you have installed the Ultrascale+ series devices in [2.3](#))

You finish creating a project after all these above steps:)

- In the project, you will need to include your source file. In the following window, click the “+” button to add source.



Choose “Add or create design source”

Add Sources

This guides you through the process of adding and creating sources for your project

- ☐ Add or create constraints
- ☒ Add or create design sources
- ☐ Add or create simulation sources

If you have finished writing the design files in other IDEs (like VSCode), you can directly add these files by clicking “Add Files”.

If you want to create design files, click “Create File”.

Add or Create Design Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those file types to add to your project. Create a new source file on disk and add it to your project.



Use Add Files, Add Directories or Create File buttons below

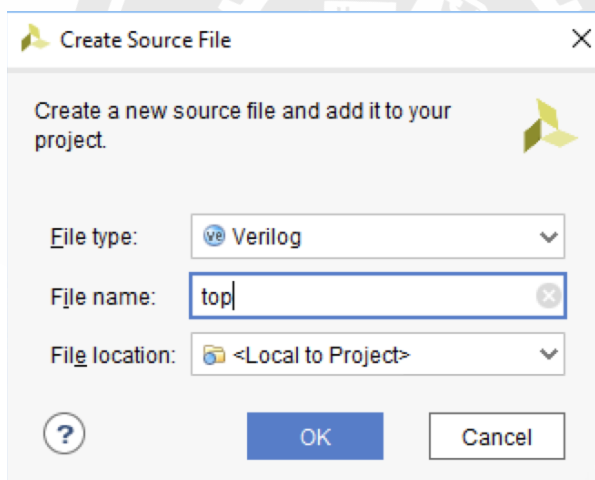
Add Files

Add Directories

Create File

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

Choose **File type** to be Verilog/SystemVerilog (ECE4700J will mainly use SystemVerilog), enter **File name**.



Create a new source file and add it to your project.

File type: Verilog

File name: top

File location: <Local to Project>

? OK Cancel

And then you can edit this file in Vivado or any other IDEs you like:)

4 Design a Round Robin Bus Arbiter

In this section, you will use the knowledge of finite state machine (FSM) we learned in previous courses to design a simplified round robin bus arbiter. You will need to use SystemVerilog in this part and you can refer to the lab lecture slides for an overview of SystemVerilog programming.

4.1 Introduction

A bus arbiter has n pairs of `request` and `grant` lines to communicate with n devices. In this lab, n equals to 3. A diagram of the arbiter we are to implement in this lab is shown in Figure 1. These devices are all connected to the same bus. If a device wants to access that bus, it need to send a request to the arbiter. Only when a grant signal is received from the arbiter, can the device really use that bus. And the arbiter can only grant one device at a time. What if there are several devices acquiring the bus at the same time? In fact, each request signal has its priority (It depends on some rules or can be customized by users) and the logic to make decision according to the priority is embedded in the arbiter.

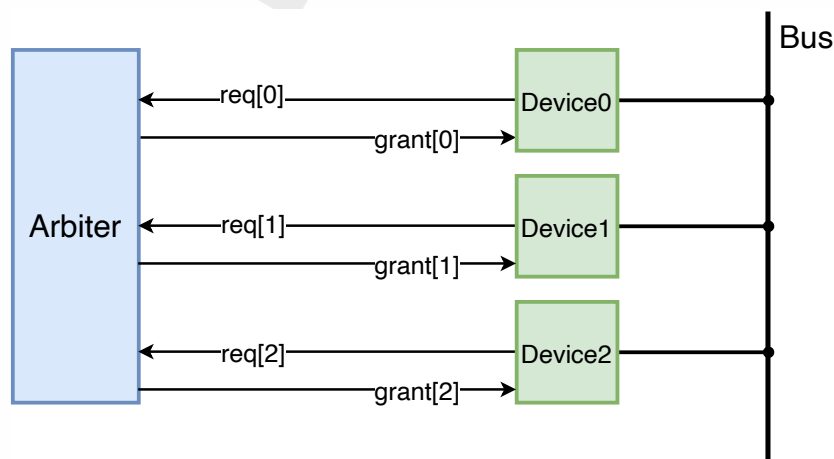


Figure 1: A simple example of arbiter with three devices connected

We are to implement a round robin arbiter. It is a commonly used arbitration strategy, where each device takes turn to access the resources (if it acquires that resource) within a fixed period of time.

To reduce your workload, this lab has several assumptions on the round robin arbiter and the devices (But real life is hard so a real arbiter can be much more complicated:) and we also provide you a FSM diagram in Figure 4

- The round-robin pattern goes in Device2 -> Device1 -> Device0 -> Device2
- If it is DeviceX's turn but it does not request, then the baton to access the bus goes to Device(X+1).

- The **grant** signal is sent on positive clock edge and will hold until the corresponding request signal goes to 0 or the next clock cycle arrives.
- The **request** signal will hold until the corresponding device receives its **grant** signal.
- The **reset** signal will send the state machine back to IDLE state, and also set all **grant** signals to 0 asynchronously.

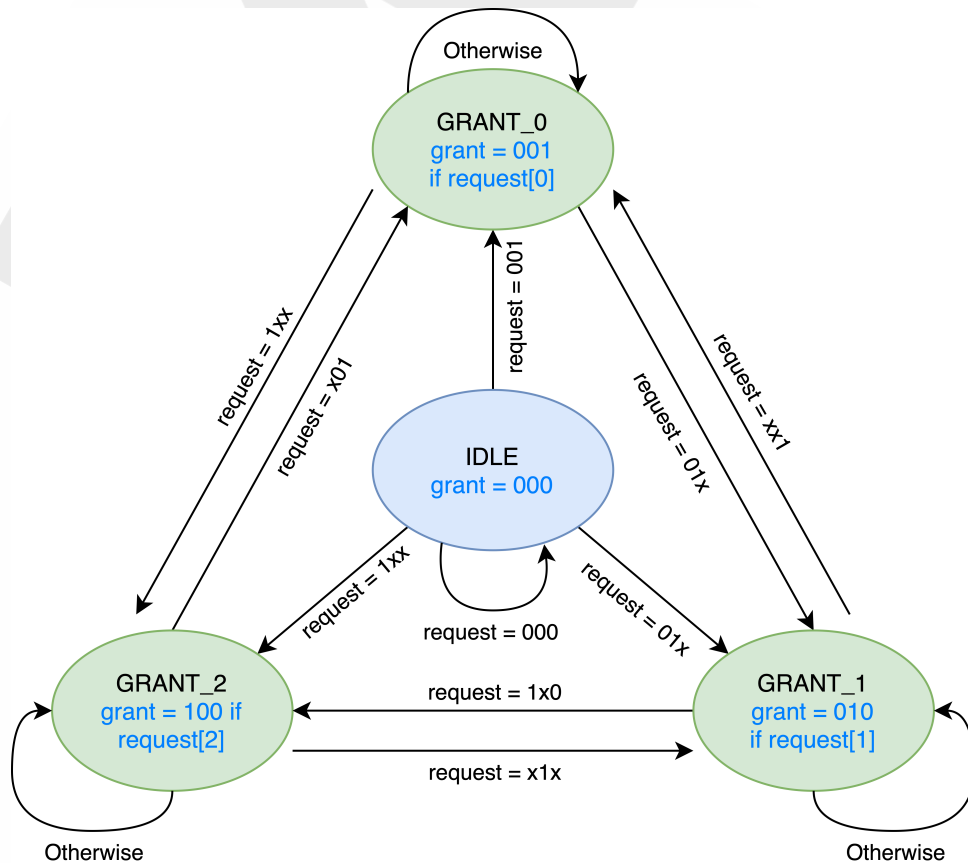


Figure 2: FSM diagram of the simplified Round Robin Arbiter

4.2 Assignment

4.2.1 Design

We provide you a starter file `arbiter.sv` to fill in the code. Please use SystemVerilog to implement the design described in 4.1. You only need to edit the “TODO” part in `arbiter.sv`.

4.2.2 Test

`arbiter_test.sv` is provided to let you have a simple test yourself. We include five simple test cases in it. You are supposed to implement a series of random test cases using the knowledge you learned from the lab lecture. Also, this is a good way to find some potential bugs in your code. You only need to edit the “TODO” part of the test file and submit it. Just a reminder, We will have hidden cases to test your code during grading.

So please test your code as precisely as possible. (Hint: Your Vivado's default simulation time 1000ns may not be enough for your testbench to finish. You need to extend Vivado's simulation time in "Settings".)

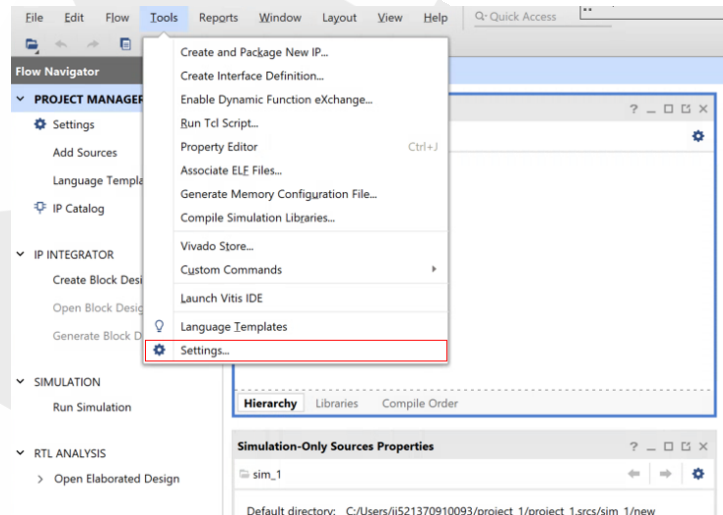


Figure 3: Open Setting Panel

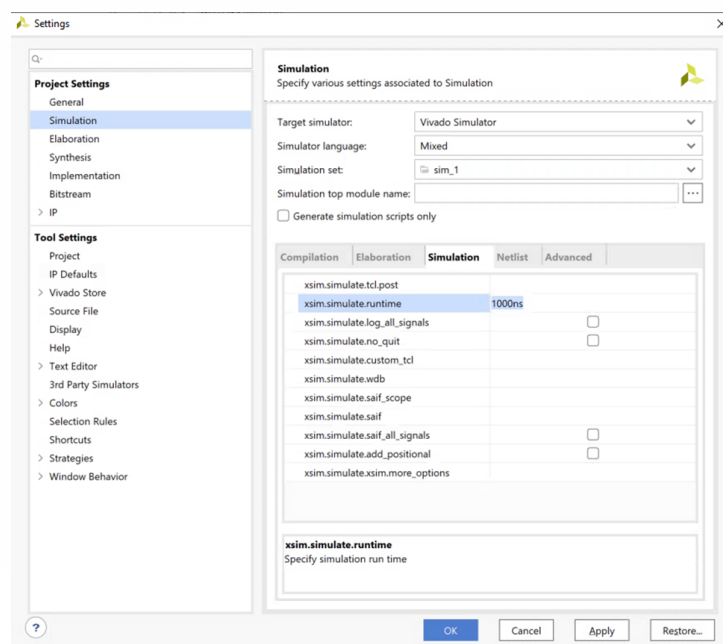
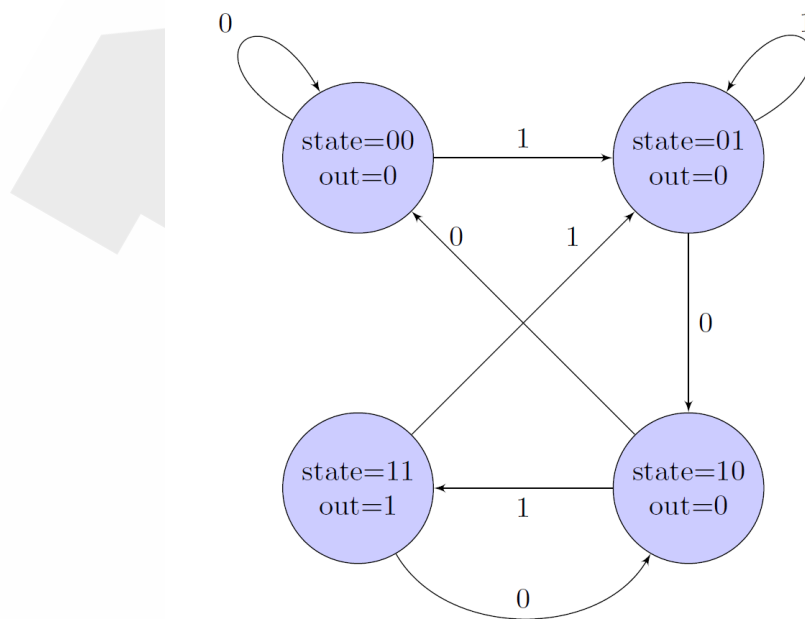


Figure 4: Change Simulation Runtime

5 Debugging of SystemVerilog Design

5.1 Introduction

Hardware debuggers will meet bugs very often. Some bugs are due to bad syntax, wrong logical expressions, or bad coding styles. These bugs are relatively simple to be found. Your assignment is to fix the errors in an FSM as shown below.



The reset signal should send the state machine back to state=00. The transition signal for this state machine is labeled in. The buggy design can be found in `fsm_ab.sv`. We also provided you a testbench, `test_ab.sv`. Read and understand how this testbench is used.

5.2 Assignment

You need to fix the bug inside the design (`fsm_ab.sv`), and also make it to be in good style. (Vivado is an advanced tool, it sometimes can automatically fix your syntax problems, like using blocking assignments in sequential logic). You need to pass post-synthesis timing simulation of the design by using the given testbench, i.e., your simulation must end normally after passing all tests. (Hint: Your Vivado's default simulation time 1000ns may not be enough for your testbench to finish. You need to extend Vivado's simulation time in "Settings".)

6 Deliverables

Please compress all the following files into a single .zip file named with your student ID, e.g. ECE4700lab1_519370910000.zip

- Section 4.2: arbiter.sv file.
- Section 4.2: arbiter_test.sv file.
- Section 5.2: fsm_ab.sv file.

7 Grading policy

Factors	Percentage
Section 4.2 design (provided test cases)	20%
Section 4.2 design (hidden test cases)	40%
Section 4.2 random test	20%
Section 5.2 debugging	20%

References

1. UM-SJTU JI ECE4700J SU 2022 Lab1
2. UM-SJTU JI ECE4700J SU 2022 Lab2
3. Umich EECS470 WN 2021 Lab1
4. Umich EECS470 WN 2021 Lab2
5. Umich EECS470 WN 2021 Lab3
6. Umich EECS470 WN 2021 Project1

Acknowledgement

- Haoyang Zhang (ECE4700J SU 2022 TA)
- Jon Beaumont (University of Michigan)
- VE270 TA Group
- Xilinx