Embedded Systems Lab
Lab Manual

Authors: Mr. Umer Shahid
Miss. Shehzeen Malik

Name: _____

Registration Number: _____

Section: _____

Department of Electrical Engineering

University of Engineering and Technology Lahore

Contents

Important Instructions

- Read out the Manual very carefully

Part A

Experiment

Experiment No. 1

Implementation of SPI Protocol using verilog on Nexys A7

Section 1.1: Objectives

Understanding the SPI protocol for communicating with the memory.

Introduction to SPI Protocol

SPI is the acronym of **S**erial **P**eripheral **I**nterface. SD card modules and card reader modules use SPI to communicate with microcontrollers. Devices communicating via SPI are in a master-slave relationship, with the microcontroller being the master and the memory card being the slave. There can only be one master but multiple slaves (but in our case flash memory is the only slave).
Following lines are used for communication

1. **MOSI** **M**aster **O**utput **S**lave **I**nput line is used for sending data from the master to the slave. MSB is sent first.
2. **MISO** **M**aster **I**nput **S**lave **O**utput line is used for sending data to the master from the slave. LSB is sent first.
3. **SCLK** This is a serial clock signal that is transmitted by the bus master to the slave device(s)
4. **SS** **S**lave **S**elect line is for master to choose between different slaves to send the data.

The SPI bus connections, for point to point communication between master and single slave device are shown in Figure 1.1. In case of multiple devices, a separate slave select (SS) signal is used by the bus master, to enable the desired slave device.

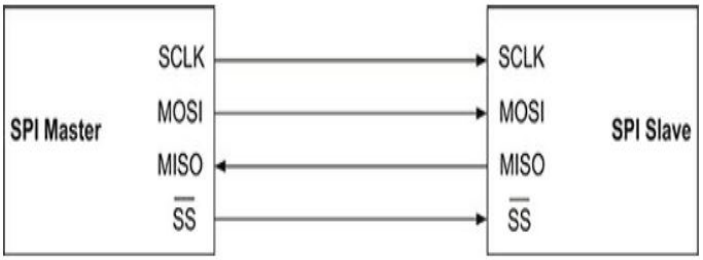

Figure 1.1: SPI Master Connected with SPI slave

SPI Signal Sequence

Due to single bus master protocol followed by the SPI interface, all the bus communications with the slave devices are initiated by the master device. When the SPI master intends to send/receive data to/from a slave device, it pulls the corresponding SS line low. This is followed by the activation of the clock signal at the desired frequency. The master transmits data using MOSI line, while the incoming data from the selected slave is received by sampling MISO line. The timing diagram for SPI communication is shown in Figure 1.2.

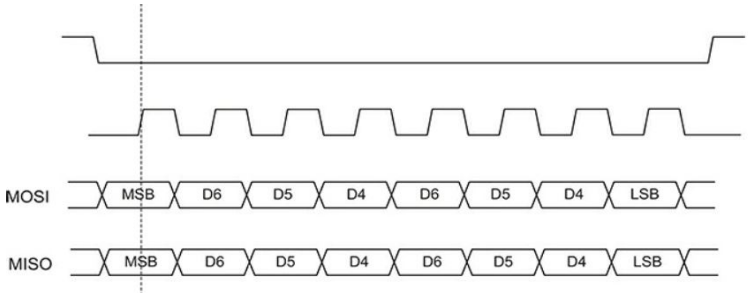

FIGURE 1.2: SPI Timing Diagram

Section 1.2: Experiment Procedure

The guide for creating a project in vivado is as follows:

1.2.1 Creating a Project in Vivado 2019.1

1. Select the *Create Project* in the welcome dialog. (Figure 1.3) Then click *Next* when the next window pops up.

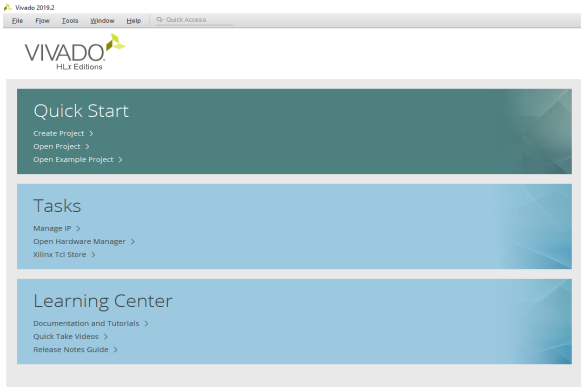

FIGURE 1.3: Welcome Dialog

2. Name your project and choose its location. Then click *Next* to continue.

3. In the New Project dialog choose *RTL Project* and check the *Do not specify sources at the time* box. Click *Next* to continue. (Figure 1.4)

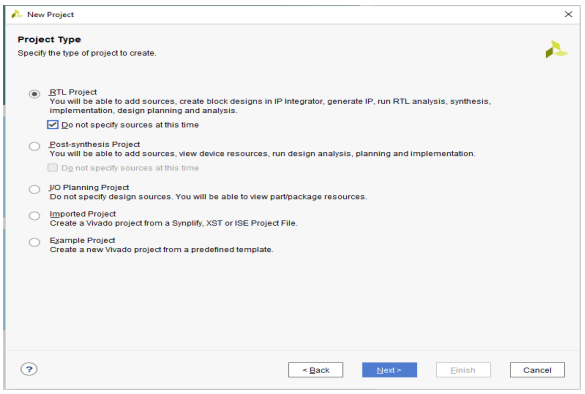

FIGURE 1.4: Choosing New Project

4. In the next window select the *Boards* tab and it will display a list of boards. Select your board Nexys A7 100T Click *Next* to Continue.(Figure 1.5)

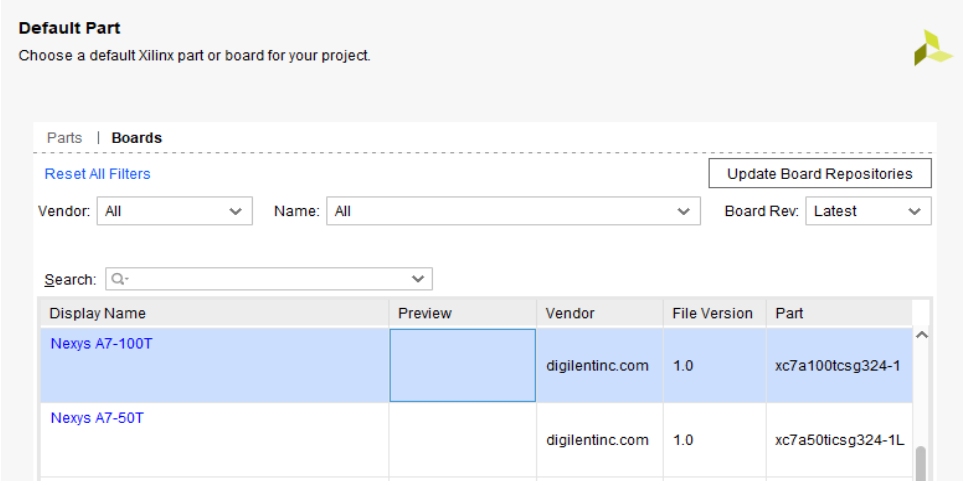

FIGURE 1.5: Choosing Board

5. Click Finish in the *New Project Summary* dialog. (Figure 1.6)

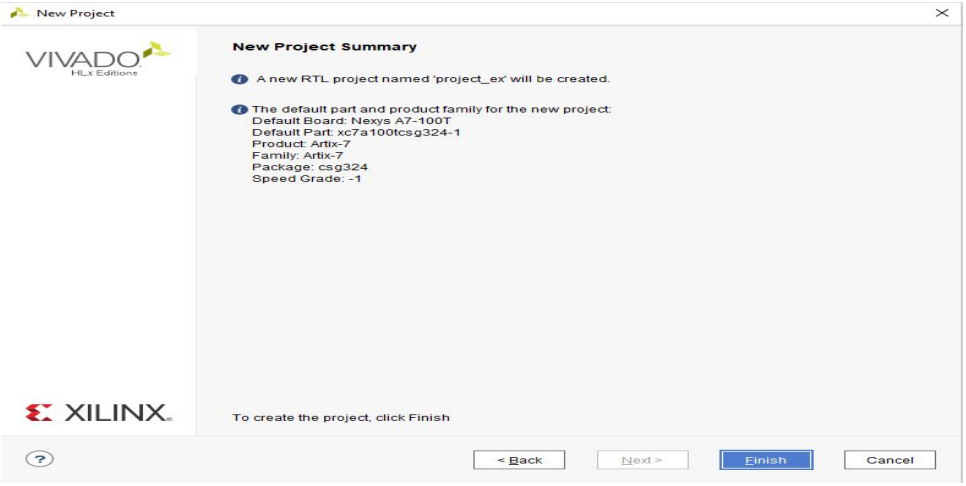

FIGURE 1.6: Project Summary

6. In the *Project Manager*, under the *Sources* tab, right click on Design Sources and choose ***Add Sources***. (Figure 1.7)

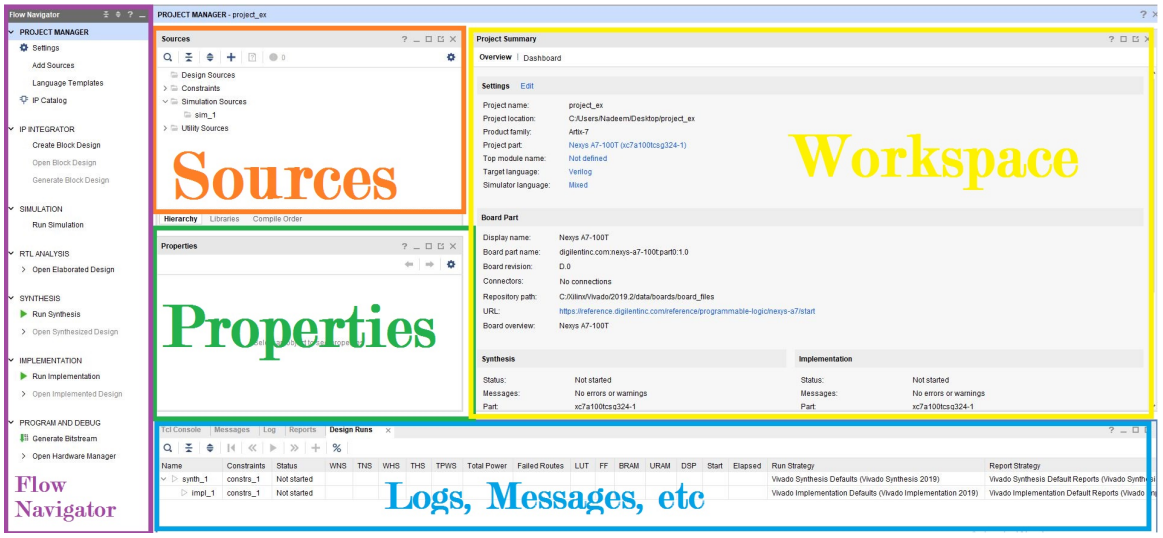

FIGURE 1.7: Project Manager

7. In the window that pops up, select *Add or create design sources*, then click *Next* to continue.

8. In the Add Sources dialog, click *Create File*.

9. You will be prompted to select a *File type, File name, and File location*. Make sure to pick *Verilog* and *Local to project* for the type and location. Give your file a name ending in **.v**. (Figure 1.8)

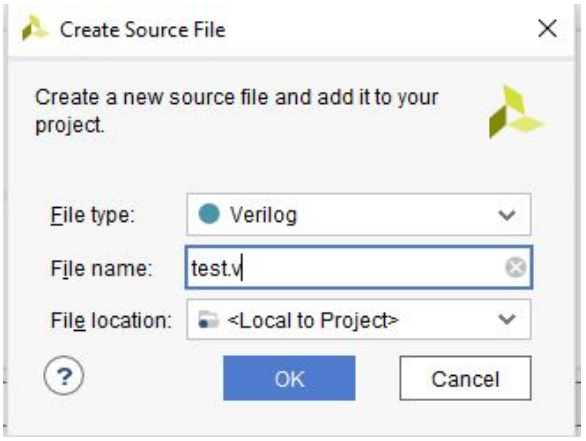

FIGURE 1.8: Naming Source File

10. In the *Define Module* dialog that appears, you can define your inputs and outputs if you want. Otherwise they will be defined in the Verilog file.

11. Open the file from design sources and write your code.

12. If you are adding the file directly to your project, make sure to check **Copy sources into project**. (Figure 1.9)

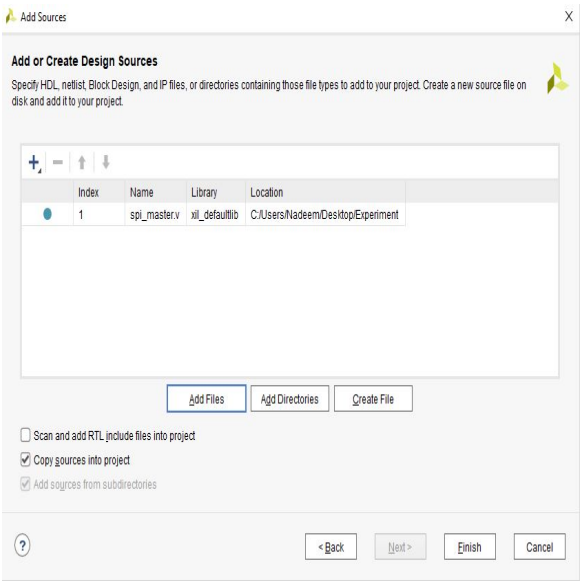

FIGURE 1.9: Adding Source File

1.2.2 Simulating SPI Master in verilog

1. Create a new source file with the name **SPI_master.v**.
2. Copy the code of file **SPI_master.v** (given with the manual) and synthesize it.
3. In the window that pops up, select *Add or create simulation sources*, then click *Next* to continue.
4. Create a new source file with the name **SPI_master_tb.v**.
5. Copy the code of file **SPI_master_tb.v** (given with the manual) and synthesize it.
6. Right Click on **SPI_master_tb.v** in Source section, and make it top module.
7. Go to simulation tap, and make **SPI_master_tb.v** top module using the step mentioned in 5.
8. Select *Run Behavioral Simulation* by clicking the *Run Simulation in Flow Navigator.*
9. Remember to Choose **Full view** to observe simulation.

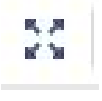

FIGURE 1.10: Full view

10. Observe the simulated waveform and compare it with the timing diagram of shown above.

1.2.3 Simulating SPI Slave in verilog

1. Create a new source file with the name **SPI_slave.v**.
2. Copy the code of file **SPI_slave.v** (given with the manual) and synthesize it.
3. Create a new source file with the name **SPI_slave_tb.v**.
4. Copy the code of file **SPI_slave_tb.v** (given with the manual) and synthesize it.
5. Right Click on **SPI_slave_tb.v** in Source section, and make it top module.
6. Go to simulation tap, and make **SPI_slave_tb.v** top module using the step mentioned in 5.
7. Observe the simulated waveform and compare it with the timing diagram of shown above.

1.2.4 Simulating SPI Master and SPI Slave

1. Create a new source file with the name **main_code.v**.
2. Copy the code of file **main_code_tb.v** (given with the manual) and synthesize it.
3. Right Click on **main_code_tb.v** in Source section, and make it top module.
4. Go to simulation tap, and make **main_code_tb.v** top module using the step mentioned in 5.
5. Observe the simulated waveform and compare it with the timing diagram of shown above.
6. Close the project and if you want also exit Vivado.

1.2.5 Final Implementation

1. Create a new project for hardware implementation as shown in section 1.2.1 *Creating a Project in Vivado 2019.2*.
2. Add the three source file with the name **main_code.v**, **SPI_master.v** and **SPI_slave.v** provided to you with the manual.
3. Right Click on **main_code.v** in Source section, and make it top module.

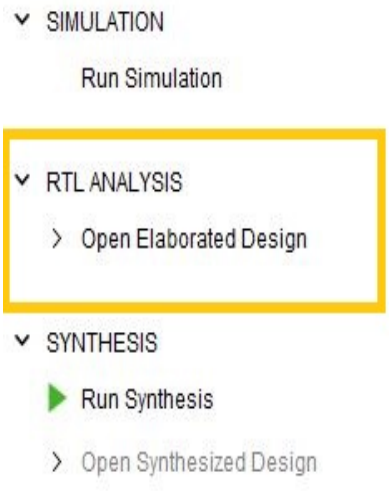

FIGURE 1.11: Opening Eleborate Design

4. In the Flow navigator, click *Open Elaborated Design* in the *RTL Analysis*. (Figure 1.11)

5. Open the***I/O ports*** tab and assign your input output pin connections under ***Package Pin*** according to the given table 1.1 or you may refer Nexys A7 Reference Manual (attached with the manual).

6. In the ***I/O Std*** choose your required voltage level (Figure 1.12) then press ctrl S and save the XDC file by name of your choice.

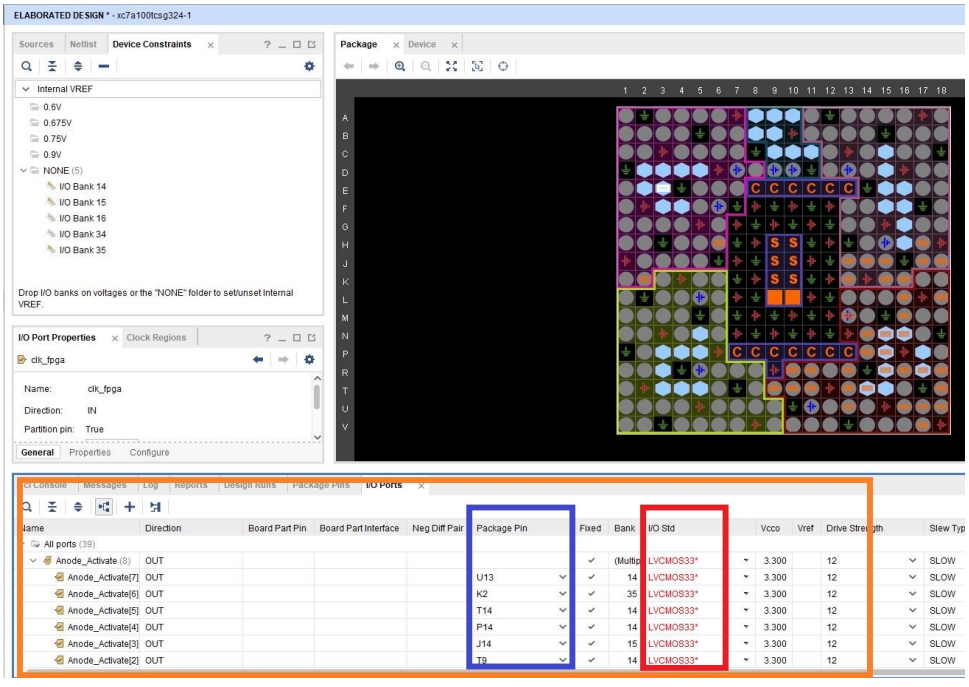

FIGURE 1.12: Assigning Input Output

7. Run synthesis and click OK in the pop up appeared.
8. Click *Run Implementation* under *Implementation* in the Flow Navigator.
9. Click *Generate BitStream* under *Program and Debug* in the Flow Navigator.

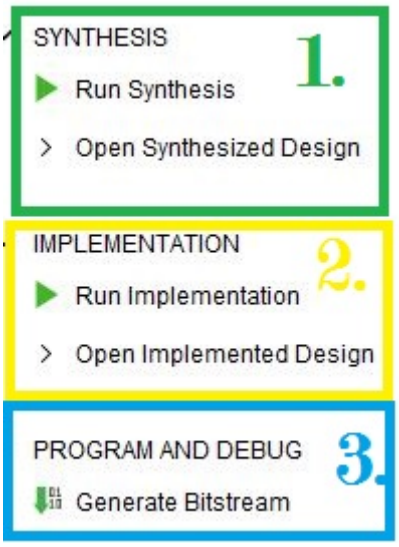

FIGURE 1.13: Synthesis, Implementation, Generate BitStream

10. You can view your Schematic Design under *Synthesis*. (Figure 1.11)

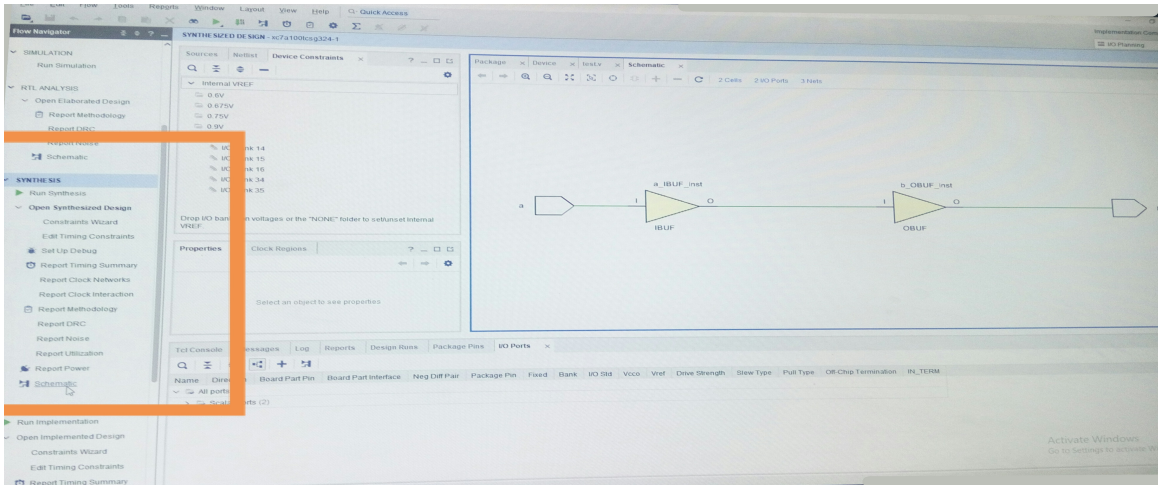

FIGURE 1.14: Schematic Design

11. Connect your FPGA Board to your computer using USB port.
12. Open the *Hardware Manager* under *Program and Debug* in the Flow Navigator or from the pop up that appeared at the finish of generating BitStream.
13. From the dropdown that opens, select *Open Target*.
14. Choose *AutoConnect*.

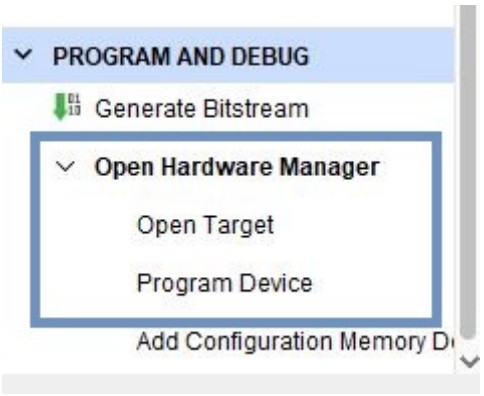

FIGURE 1.15: Connect Hardware

15. Now its time to program the bit file into your hardware. In the dropdown of the*Hardware Manager*, click *Program Device* and choose the device (xc..) to program.

16. The Bitstream File field should be automatically filled in with the bit file generated earlier. If not, click the button at the right end of the field and navigate to $< ProjectDirectory > / < ProjectName > .runs/impl\_1/$ and select the bit file and click *Program*.

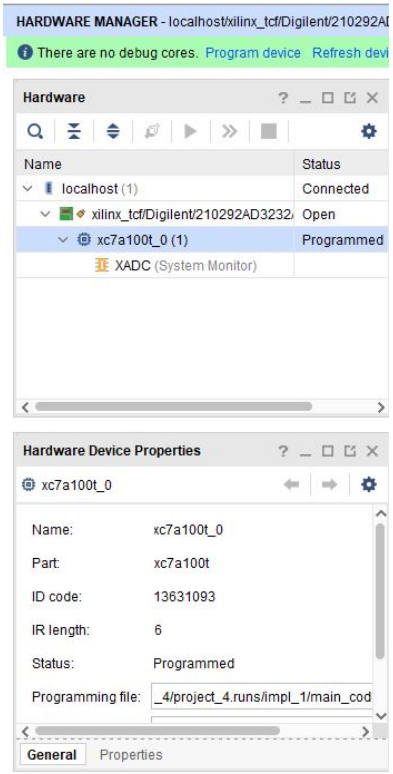

FIGURE 1.16: Hardware Manager

17. Your code is loaded to your FPGA. Pressing buttons or as your code chose it should have your output displayed on your FPGA.

Table 1.1: Pin Configuration

Input/Output Name	Pin Assigned	Comments
Anode_Activate[7]	U13	Output to be connected to AN7
Anode_Activate[6]	K2	Output to be connected to AN6
Anode_Activate[5]	T14	Output to be connected to AN5
Anode_Activate[4]	P14	Output to be connected to AN4
Anode_Activate[3]	J14	Output to be connected to AN3
Anode_Activate[2]	T9	Output to be connected to AN2
Anode_Activate[1]	J18	Output to be connected to AN1
Anode_Activate[0]	J17	Output to be connected to AN0
LED_out[6]	T10	Output to Segment A
LED_out[5]	R10	Output to Segment B
LED_out[4]	K16	Output to Segment C
LED_out[3]	K13	Output to Segment D
LED_out[2]	P15	Output to Segment E
LED_out[1]	T11	Output to Segment F
LED_out[0]	L18	Output to Segment G
data_led[7]	H17	Output to LED 7
data_led[6]	K15	Output to LED 6
data_led[5]	J13	Output to LED 5
data_led[4]	N14	Output to LED 4
data_led[3]	R18	Output to LED 3
data_led[2]	V17	Output to LED 2
data_led[1]	U17	Output to LED 1
data_led[0]	U16	Output to LED 0
indicator[3]	M16	RGB-GREEN Led Status Output
indicator[2]	R11	RGB-GREEN Led Status Output
indicator[1]	N15	RGB-RED Led Status Output
indicator[0]	N16	RGB-RED Led Status Output
data[7]	R15	Switch to get Input
data[6]	R17	Switch to get Input
data[5]	T18	Switch to get Input
data[4]	U18	Switch to get Input
data[3]	R13	Switch to get Input
data[2]	H6	Switch to get Input
data[1]	T13	Switch to get Input
data[0]	R16	Switch to get Input
clk_fpga	E3	System Clock
write_signal	L16	Switch to control write operation
read_signal	M13	Switch to control read operation
slave_sel	J15	Switch to select slave device