

Lab-1: NEXYS4 DDR Kit and VIVADO Tool

Introduction

Objectives:

- Study features of NEXYS4 DDR kit
- Review Verilog Hardware Description Language
- Understand the process of developing Verilog design using Xilinx VIVADO platform
- Simulate Verilog program
- Download one application design into NEXYS4 DDR FPGA

Materials:

- NEXYS4 DDR FPGA board
- PC with USB port
- USB cable
- Xilinx VIVADO tool

Verilog Operator:

Operator	Name
[]	Bit-select or part-select
()	Parenthesis
!, ~	Logical and bit-wise NOT
&, , ~&, ~ , ^, ~^, ^~	Reduction AND, OR, NAND, NOR, XOR, XNOR
+, -	Unary (sign) plus, unary (sign) minus
{ }	Concatenation
{ { } }	Replication
*, /, %	Multiply, divide, modulus
+, -	Binary addition, binary subtraction
<<, >>	Shift left, shift right
<, <=, >, >=	Comparisons
==, !=	Logical equality, logical inequality
===, !==	Case equality, case inequality
&	Bit-wise AND
^, ~^, ^~	Bit-wise XOR, bit-wise XNOR
	Bit-wise OR
&&,	Logical AND, logical OR
?:	Conditional

Demo 1- Implement Verilog Design on FPGA

In this experiment, a new project is created by using the Xilinx VIVADO tool. And a Verilog design is checked for syntax errors, downloaded on the FPGA board, and finally verified on the FPGA board.

Learn the following from the instructor:

- 1). Connect USB cable to PC and provide power supply to the board.
- 2). Create a new project and add a new Verilog file to this project.
- 3). Write the Verilog source code.
- 4). Check Syntax.
- 5). Download the design on FPGA.
- 6). Verify the results on FPGA.

Procedures:

1. Start VIVADO Design Tool
2. Quick Start -> Create Project -> Create a New Vivado Project
3. Set Project Location to be under directory of T:\
4. Type “demo” after “Project Name:” -> click on next, and then select “RTL Project”.
Do not specify sources at this time. Click on next.
5. Set up project settings:
Family: Artix-7
Device: xc7a100tcs324-1
Click on Next, and then click on Finish.
The results of steps 3, 4, and 5 are shown in Figure 1-1.

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:

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gb Transceivers	GTPE2 Transceivers
xc7a100tcsg324-2	324	210	63400	126800	135	0	240	0	0
xc7a100tcsg324-2L	324	210	63400	126800	135	0	240	0	0
xc7a100tcsg324-1	324	210	63400	126800	135	0	240	0	0
xc7a100tfgg484-3	484	285	63400	126800	135	0	240	4	4
xc7a100tfgg484-2	484	285	63400	126800	135	0	240	4	4
xc7a100tfgg484-2L	484	285	63400	126800	135	0	240	4	4
xc7a100tfgg484-1	484	285	63400	126800	135	0	240	4	4

?

Figure 1-1. Device properties

- Click on Simulator language “Mixed” in the Project Summary window, and then change the simulator language to “Verilog” inside the “Settings” popup window shown in Figure 1-2. Click on “OK” button.

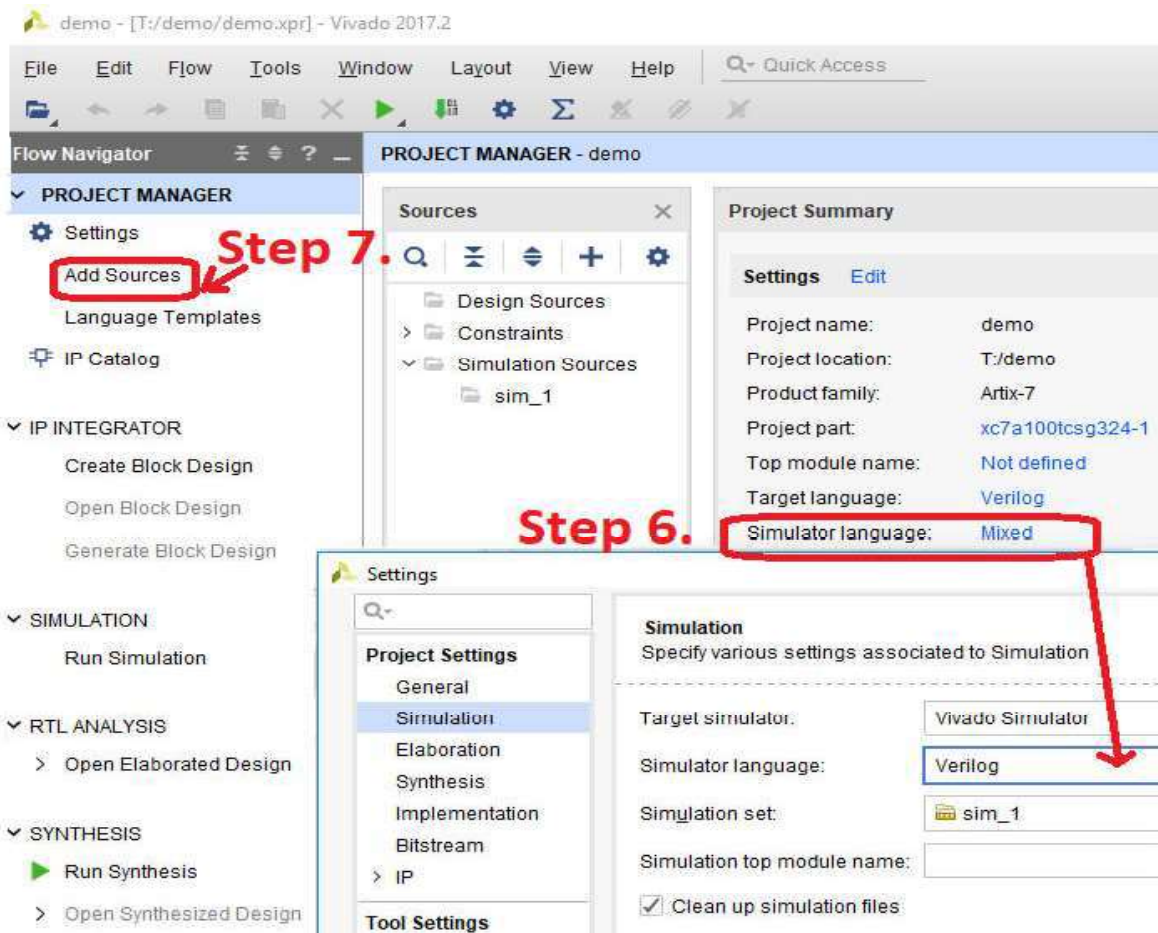


Figure 1-2. Project settings

7. Click on “Add Sources” shown in Figure 1-2, choose “Add or create design sources” -> Next.
8. In the “Add Sources” popup window shown in Figure 1-3, click on “Create File” button, and then enter “myand” as Verilog file name. Next, click on “OK” -> “Finish”.

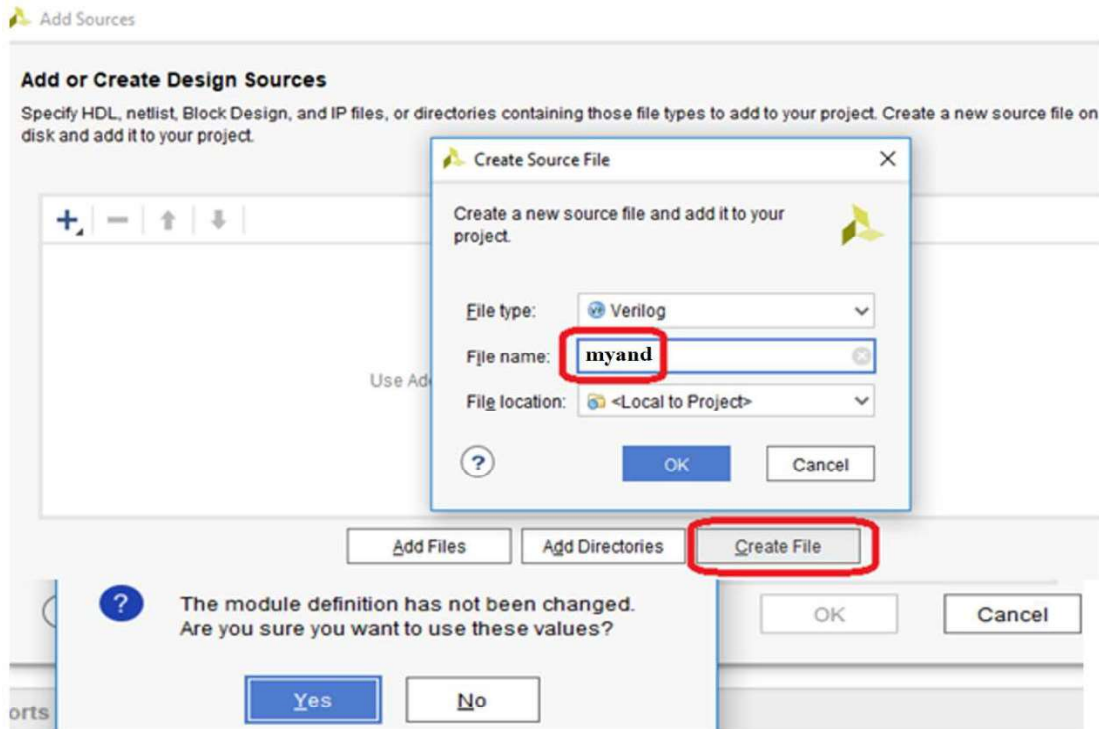


Figure 1-3. Add Sources

9.

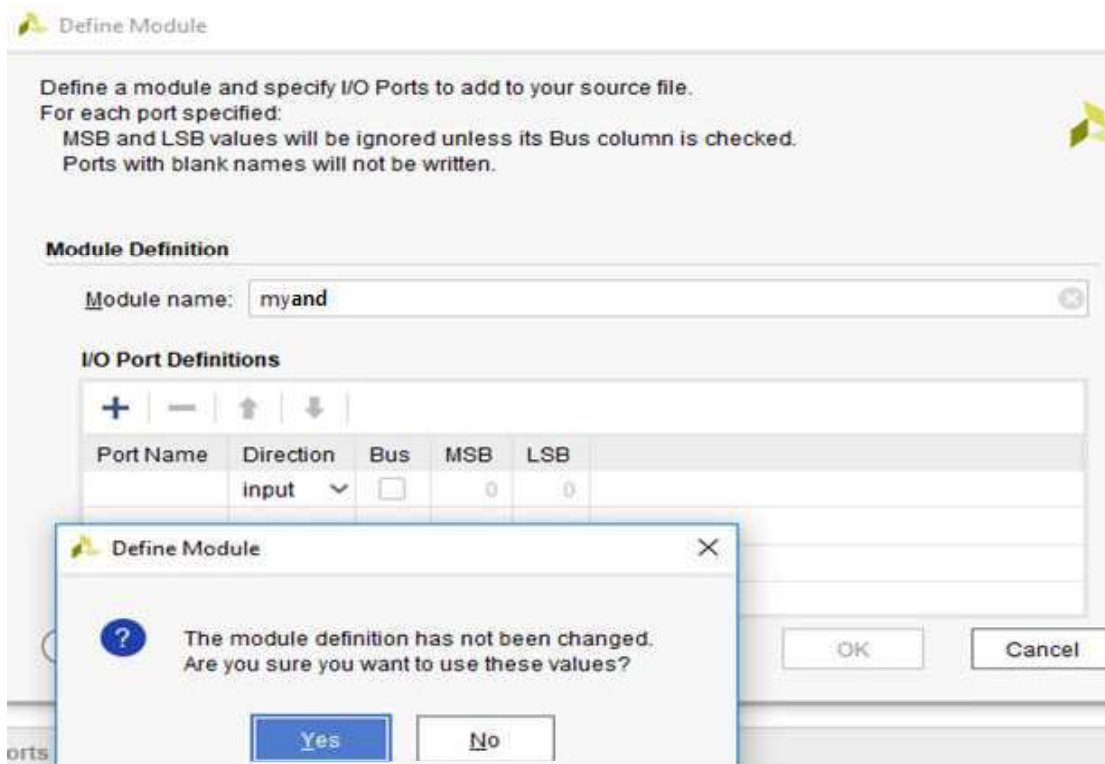


Figure 1-4. Device properties

10. Enter the following Verilog program and save it as myand.v

```
module myand (a, b, c);  
  input  a, b;  
  output c;  
  
  and g1(c, a, b);  
  
endmodule
```

11. In the Project Manager window, right click on “Add Sources” button, and choose “Add or create constraints” button, then click on next -> “Create File”. Enter “myinv-pin” as the constraint file name. Edit the constraint file as shown below:

```
set-property -dict { PACKAGE-PIN J15  IOSTANDARD LVCMOS33 } [ get-ports { a } ];  
set-property -dict { PACKAGE-PIN L16  IOSTANDARD LVCMOS33 } [ get-ports { b } ];  
set-property -dict { PACKAGE-PIN H17  IOSTANDARD LVCMOS33 } [ get-ports { f } ];
```

In this design, one input signal “a” is connected with switch “J15”, “b” is connected with switch “L16”, and one output signal “f” is connected with LED “H17” shown in Figure 1-5.

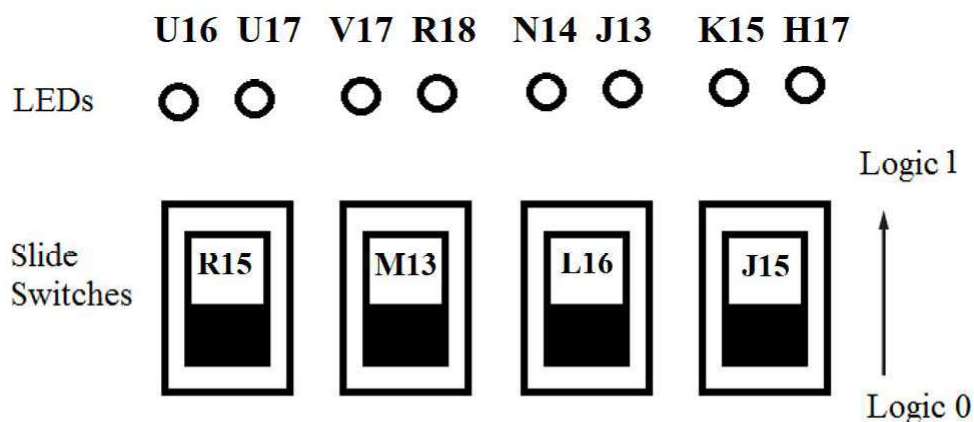
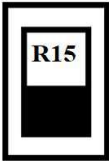
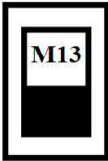
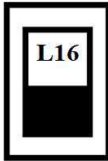
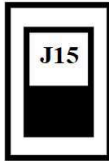


Figure 1-5. Slide switches and LEDs with FPGA pin numbers

12. In the “Project Manager” window, click on “Run Synthesis” button.
13. After synthesis is successfully completed, click on “Run Implementation” button.
14. After implementation is successfully completed, click on “Generate Bitstream” button.
15. After bitstream generation is successfully completed, select the “Open Hardware Manager” button.
16. After Hardware Manager window is opened, click on “Open target” -> “Auto Connect”.
17. Click on “Program device”, next select “xc7a100t_0”. And then you’ll see Bitstream File name of “demo.bit” with the correct directory name listed. Next, click on “Program”.
18. Test “AND” gate functionality on FPGA board according to Table 1-1.

Table 1-1. AND gate functionality table, LED and slide switch associated FPGA pin numbers

Input Switches		Output LED
a	b	f
0	0	0
0	1	0
1	0	0
1	1	1

	U16	U17	V17	R18	N14	J13	K15	H17
LEDs	○	○	○	○	○	○	○	○
								f
Slide Switches								
					b		a	

Logic 1 ↑
Logic 0

Demo 2- Multiplexer Design & Simulation

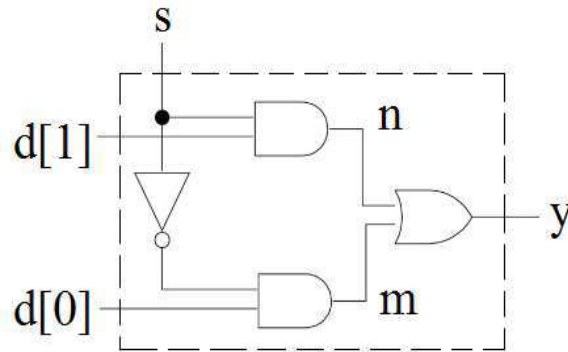


Figure 1-6. 2-to-1 Multiplexer

Table 1-2. Verilog design and testbench for 2-to-1 multiplexer

Verilog Design of 2-to-1 Multiplexer	Multiplexer Testbench
<pre> module mux2to1(d, s, y); input [1:0] d; input s; output y; wire m, n; assign m = (~s) & d[0]; assign n = s & d[1]; assign y = m n; endmodule </pre>	<pre> module mymux_tb; reg [1:0] d; reg s; wire y; mux2to1 u1(.d(d), .s(s), .y(y)); initial begin //d[0]=0; d[1]=1; s=0; d=2'b10; s=0; #10; //d[0]=1; d[1]=0; s=0; d=2'b01; s=0; #10; //d[0]=0; d[1]=1; s=1; d=2'b10; s=1; #10; //d[0]=1; d[1]=0; s=1; d=2'b01; s=1; #20 \$stop; end endmodule </pre>



Figure 1-7. 2-to-1 multiplexer simulation waveform

Part 1. 2-to-4 Decoder Design

In this lab, you will build a 2-to-4 decoder circuit shown in the diagram below

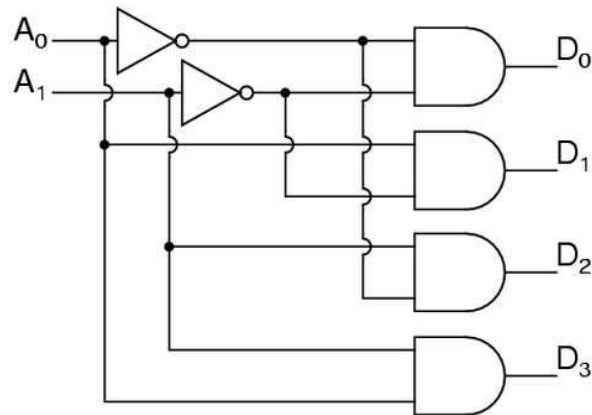


Figure 1-8. 2-to-4 decoder circuit

Implement the decoder circuit in Verilog based on the figure above, download your decoder design to the NEXYS4 DDR FPGA board, and show demo to lab instructor.