İSTANBUL TEKNİK ÜNİVERSİTESİ 1773

ISTANBUL TECHNICAL UNIVERSITY

Digital System Design Applications

Experiment I
SSI COMPONENTS

2021

Preliminary

Students should know Vivado Design Suite Tutorial(Logic Simulation-Synthesis-Implementation-Using Constraints). These tutorials is going to teach you how to create an Vivado project, simulate and implement your HDL-based design, and configure your FPGA. Make sure that you understand and learn every concept given in Vivado Design Suite Tutorial.

Objectives

- Become familiar with Xilinx Vivado Software
- Create a library which contains SSI Components

Requirements

Students are expected to be able to

- define hardwares with Verilog
- create project on Vivado
- synthesize, simulate, implement designs

Experiment Report Checklist

Each student is going to prepare his/her own report. Reports should include:

1. **OR Gate**
   - Verilog code,testbench code and behavioral simulation wave screenshots.
   - RTL Schematic.
   - Technology Schematic,Truth Table of LUT
   - Add **Synthesis Report**
     - Low Level Synthesis - area consumption
     - Final Report - design statistics, cell usage, device utilization summary, maximum combinational path delay
     - Timing Detail - discuss each path delay
   - Create testbench for your design then make behavioral and **Post Synthesis Timing simulations**.
   - Add **Post-synthesis simulation model**
   - Investigate the generated module after **Simulation**, add the following informations to your report:
     - Primitives and their functions
     - Initial value of LUT primitive, and the meaning of it
   - Investigate the generated module after **Implementation**, and compare it with the one generated in the simulation step. Add your findings to your report.
   - Investigate **Implementation Reports** , and add the following informations to your report:
     - Number of LUTs
     - Number of bonded IOBs
     - IOB properties
   - Generate **Timing Report Summary**. Investigate it, and add the following to your report:
     - Delays from source pads to destination pad.
2. **NOT Gate**
   - Technology Schematic,behavioral simulation wave screenshot.
3. **NAND Gate**
   - Verilog code,testbench code and behavioral simulation wave screenshots.

  - Technology Schematic, Truth table of LUT

4. **NOR Gate**
   - Technology Schematic, Truth table of LUT,behavioral simulation screenshots.
5. **EXOR Gate**
   - Verilog code,testbench code and behavioral simulation screenshots.
   - Technology Schematic, Truth table of LUT
6. **EXNOR Gate**
   - RTL Schematic, Technology Schematic, Truth table of LUT,behavioral simulation screenshots.
7. **TRI Component**
   - Technology Schematic, Truth table of LUT,behavioral simulation screenshots.
8. **Top Module**
   - Verilog code, testbench code
   - Technology Schematic, Truth table of LUT,behavioral simulation screenshots.
   - Implementation Report Timing Summary
9. **Your comments on results**
10. Definition of **fan-out**

- **Projects and reports are to be done INDIVIDUALLY. High amounts of points will be deducted from similar works.**
- **Reports must be written in a proper manner. Divide your text to sections and subsections if needed, label your figures and connect your sections with proper explanations of your works. Reports filled with imprecisely placed tables and figures, with no verbal explanations in workflow, will not fare well.**
- **Check homeworks section in Ninova for submission dates. There will be two different submissions for project archive folder and a PDF report file.**

Creating SSI Library

1. Create a new Vivado project with the settings below:
   - Download and extract this Zip file from the link below https://reference.digilentinc.com/reference/software/vivado/board-files
   - Copy the files that in this directory "vivado-board-master>new>boards-files"
   - Paste the files to C:Xilinx>Vivado>2018.2>data>boards>board-files
   - Restart Vivado
   - You are now ready to use a Vivado project for the Digilent Nexys 4, Nexys 4 DDR, Zybo, Zedboard and Basys 3 FPGA Boards.
   - You can select a FPGA board which want to study on it.

Add a new Verilog module named **SSI_ Library** to your project. Clear all lines in your module.

2. Another method for adding FPGA board **(Nexys4DDR)** to your project
   - Family: **Artix-7**
   - Device: **XC7a100t**
   - Package: **CGS324**
   - Speed: **-1**
   - Synthesis Tool: **XST (Verilog\Verilog)**
   - Simulator: **ISim (VHDL\Verilog)**
   - Preferred Language: **Verilog**

OR Gate

1. Write down a module which is called **OR** into your **SSI_ Library.v** file. This module is going to have 1-bit inputs **I1**, **I2** and 1-bit output **O**. This module should behave like an **or gate**. Obtain this behaviour using only **logic operators**.
2. Check syntax.
3. Synthesize your design if there is no problem with syntax.
4. Investigate **Synthesis Report**, and add the following informations to your report:
   - Low Level Synthesis - area consumption (generate Utilization Report)
   - Final Report - design statistics, cell usage, device utilization summary, maximum combinational path delay
   - Timing Detail - discuss each path delay (generate Timing Report).
5. View both **RTL** (obtain from RTL Analysis part) and **Technology** schematic (obtain from Synthesis part) of your module.
6. Verify that schematic of LUT2 in Technology schematic and RTL schematic are same, and they are both or gates.
7. Create testbench for your design then make **behavioral simulation** and **Post Synthesis Timing simulations**. Find post-synthesis simulation model from the sim directory of your project.
8. Add constraint file to your design. **Implement** your design.
9. Generate **Post-Implementation Simulation Model** by running **Post-Implementation Timing simulation.**

NOT Gate

1. Write down another module which is called **NOT** into your **SSI_ Library.v** file. This module is going to have 1-bit input **I**, and 1-bit output **O**. This module should behave like an **not gate**. Obtain this behaviour using only **logic operators**.
2. View Schematic and Create testbench of your design and check the **behavioral simulation**.

NAND Gate

1. Write down another module which is called **NAND** into your **SSI_ Library.v** file. This module is going to have 1-bit inputs **I1**, **I2** and 1-bit output **O**. This module should behave like an **nand gate**. Obtain this behaviour using **always** block.
2. Check Technology Schematic
3. Create testbench of your design and check the **behavioral simulation**.

NOR Gate

1. Write down another module which is called **NOR** into your **SSI_ Library.v** file. This module is going to have 1-bit inputs **I1**, **I2** and 1-bit output **O**. This module should behave like an **nor gate**. Obtain this behaviour using **always** block. Repeat the steps from NAND gate.

EXOR Gate

1. Write down another module which is called **EXOR** into your **SSI_ Library.v** file. This module is going to have 1-bit inputs **I1**, **I2** and 1-bit output **O**. This module should behave like an **exor gate**. Obtain this behaviour using **LUT2 primitive**. Repeat the steps from NAND gate.

EXNOR Gate

1. Write down another module which is called **EXNOR** into your **SSI_ Library.v** file. This module is going to have 1-bit inputs **I1**, **I2** and 1-bit output **O**. This module should behave like an **exnor gate**. Obtain this behaviour using **LUT2 primitive**. Repeat the steps from NAND gate.

TRI Component

1. Write down another module which is called **TRI** into your **SSI_ Library.v** file. This module is going to have 1-bit inputs **I**, **E** and 1-bit output **O**. When E input is equal to 1, O output is gong to have the same value as input I. When E input is equal to 0, O output will be high impedance Obtain this behaviour using **?** operator. Repeat the steps from NAND gate.

Top Module

1. Add a new Verilog module named **Top_ Module** to your project.
2. Add all logic gates you designed to **Top_ Module**.Make appropriate connections to the outputs to analyze the output of each logic circuit on the simulation and board leds.
3. Synthesize **Top_ Module** and check RTL and Technology Schematics
4. Create Constraint file.

5. In Constraint file, led[0] *(H17)* will be output of **OR Gate**,led[1] *(K15)* will be output of **NOT Gate** and so that each logic gates output will be shown in leds.
6. Generate programming file and program your FPGA.Observe your design on FPGA.

References:

1. Nexys4DDR Reference Manual
2. Artix-7 Libraries Guide for HDL Designs
3. Constraints Guide
4. Brown&Vrasenic, "Fundamentals of Digital Logic with Verilog Design", McGraw-Hill, p.17-47, 87-107