ISTANBUL TECHNICAL UNIVERSITY

Digital System Design Applications

Experiment V
MEMORY ELEMENTS

Preliminary

Students should cover Section III of LogiCORE IP Block Memory Generator v7.3 Product Guide.

Objectives

- Become familiar with flip-flop types
- To define memory elements with Verilog

Requirements

Students are expected to be able to

- define hardwares with Verilog
- create project on ISE
- synthesize, simulate, implement designs, generate bitstreams and configure FPGA

References:

1. Nexys2 Reference Manual
2. Spartan-3E Libraries Guide for HDL Designs
3. Constraints Guide
4. Brown&Vrasenic, "Fundamentals of Digital Logic with Verilog Design", McGraw-Hill, p.349-369
5. M. Mano, "Digital Design", Chapters 6.2, 6.3, p.203-219
6. LogiCORE IP Block Memory Generator v7.3 Product Guide

Creating ISE Project

Create a new ISE project with the settings below:

- Family: **Spartan 3E**
- Device: **XC3S500E**
- Package: **FG320**
- Speed: **-4**
- Synthesis Tool: **XST (VHDL\Verilog)**
- Simulator: **ISim (VHDL\Verilog)**
- Preferred Language: **Verilog**

Simple Memory Element (20 Minutes)

1. From Brown&Vrasenic Fig. 7.2, realize the simple memory element which consists of two **NOT gates** using your **SSI Library**.
2. Assign one of the outputs of NOT gates to a **LED**.
3. Since your module does not have any inputs, use **Save Net Flag** constraint to prevent trimming unconnected signals.
4. **Synthesize** your design.
5. Add the following to your report:
   - Synthesize warning
   - RTL and Technology Schematic
6. Generate Programming File, and download generated bitstream to FPGA. If the LED is connected 0, change your LED connection to the output of other NOT gate. Add the observed results to your report.

SR Latch with NOR Gate (20 Minutes)

1. From M. Mano, "Digital Design" Fig. 5.3, realize the SR Latch with **NOR gates** using your **SSI Library**. This module is going to have 1-b,t inputs **S**, **R**, and 1-bit outputs **Q**, **Qn**. Connect these ports, **S**, **R**, **Q**, **Qn**, to **SW0**, **SW1**, **LD0**, **LD1**, respectively.
2. Add the followings to your report:
   - Truth and excitation table of SR Latch
   - Characteristic and inverse characteristic function of SR Latch
   - RTL and Technology Schematics
   - PAD with the longest delay and its delay time
3. Generate Programming File, and download generated bitstream to FPGA. Show that your design works correctly.

SR Latch with NAND Gate (20 Minutes)

1. From M. Mano, "Digital Design" Fig. 5.4, realize the SR Latch with **NAND gates** using your **SSI Library**.
2. Repeat the same steps from SR Latch with NOR Gate.

SR Latch with Control Input (20 Minutes)

1. From M. Mano, "Digital Design" Fig. 5.5, realize the **SR Latch with Control Input** using your **SSI Library**.
2. Add the followings to your report:
   - Truth and excitation table of SR Latch
   - Characteristic and inverse characteristic function of SR Latch
   - RTL and Technology Schematics
3. Generate Programming File, and download generated bitstream to FPGA. Show that your design works correctly.

D Latch (20 Minutes)

1. From M. Mano, "Digital Design" Fig. 5.6, realize the **D Latch** using your **SSI Library**.
2. Add the followings to your report:
   - Truth and excitation table of D Latch
   - Characteristic and inverse characteristic function of D Latch
   - RTL and Technology Schematics
3. Generate Programming File, and download generated bitstream to FPGA. Show that your design works correctly.

Master-Slave D Flip-Flop (20 Minutes)

1. From Brown&Vrasenic Fig. 7.10, realize the **Master-Slave D Flip-Flop** using previously designed **D Latch**.
2. Add the followings to your report:
   - Explanation of how Master-Slave D Flip-Flop works
   - Which edge of the clock signal effects the output?
   - RTL and Technology Schematics
3. Draw **Qm** and **Q** signals on the given time diagram in Fig. 1, and add it to your report.
4. Generate Programming File, and download generated bitstream to FPGA. Show that your design works correctly.

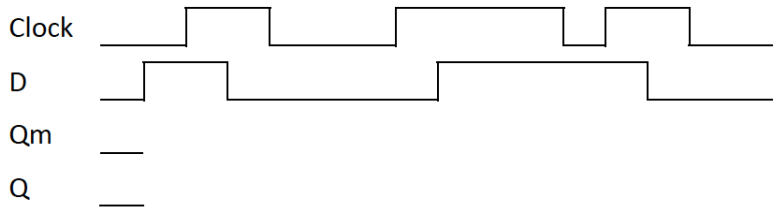

Figure 1: Time diagram for Master-Slave Flip-Flop.

Edge-Triggered D Flip-Flop (20 Minutes)

1. From Brown&Vrasenic Fig. 7.11, realize the **Edge-Triggered D Flip-Flop** using **SSI Library**.
2. Add the followings to your report:
   - Explanation of how Edge-Triggered D Flip-Flop works
   - Which edge of the clock signal effects the output?
   - Explanation of Clear and Preset operations in Brown&Vrasenic Fig. 7.15.
   - RTL and Technology Schematics
3. Generate Programming File, and download generated bitstream to FPGA. Show that your design works correctly.

D Flip-Flop in Logic Slice (20 Minutes)

1. Write down another module which has same ports with **Edge-Triggered D Flip-Flop** . This module is going to have 1-bit **reg** named **FF**. Using **always** block, assign **D input** to **FF** on positive-edge of clock signal. Assign **FF** to **Q** output, and complement of **FF** to **Qn** output.
2. Add the followings to your report:
   - RTL and Technology Schematics
3. Generate Programming File, and download generated bitstream to FPGA. Show that your design works correctly.

8-bit Register (40 Minutes)

1. Write down another module which has the following ports:
   - 8-bit input **SW**
   - 1-bit input **BTN**
   - 8-bit output **LED**
2. This module is going to assign values of SW to LED output on the positive-edge BTN input.
3. Generate Programming File, and download generated bitstream to FPGA. Show that your design works correctly.

4. Now add another input named **CLEAR** to your module. When CLEAR input equals to 1, LED output must be 0. This operation must be independent from positive-edge of BTN input.

5. Add the followings to your report:
   - RTL and Technology Schematics of both designs
   - Investigate how you define 4x8-bit of register array in Verilog.

6. Generate Programming File, and download generated bitstream to FPGA. Show that your design works correctly.

Block RAM (40 Minutes)

1. You are going to design a Block RAM using **Core Generator**. Create a block RAM with specifications below:
   - Memory Type: **Single Port RAM**,
   - Algorithm: **Minimum Area**,
   - Write Width: **8**,
   - Operating Mode: **Write First**,
   - Enable: **Always Enabled**,
   - Load Init File: **memory.coe** (it is provided in Ninova),
   - Leave other options unchanged.

2. Write down a top module for generated block RAM which has the following ports:
   - 1-bit input **clka**
   - 1-bit input **wea**
   - 4-bit input **addra**
   - 8-bit output **douta**

3. Connect **clka** to **50Mhz clock**, **wea** to **BTN0**, **addra** to **SW0-3**, and **douta** to **LED0-7**. Connect **dina** input of your generated block RAM to a 8-bit wire in the top module.

4. Generate Programming File, and download generated bitstream to FPGA. Examine values on LEDs.

5. Examine **memory.coe**, and change it so that each address in memory has the value of address **mod** group number.

6. Generate Programming File, and download generated bitstream to FPGA. Show that your design works correctly.

7. Add the followings to your report:
   - Functionalities of Block RAM ports,
   - Structure of coe file

Experiment Report

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

- Screenshots and results asked above
- Your comments on results
- Investigate **Mealy** and **Moore** machines, and comment on their similarities and differences.

Reports are to be submitted before next experiment. Submission includes reports and project files.