HOCHSCHULE RHEIN-WAAL
Rhine-Waal University of Applied Sciences

LABORATORY 4

Model-based Hardware Design

Latches

Lab notes prepared by Mr. Gligor Duchev

Lab notes revised by: Prof. Andy Stamm

Contents

1. Safety instructions

ESD Warning

Caution Although this product has been designed to be as robust as possible, ESD (Electrostatic Discharge) can damage or upset this product. This product must be protected at all times from ESD. Static charges may easily produce potentials of several kilovolts on the human body or equipment, which can discharge without detection. Industry-standard ESD precautions must be employed at all times.

The NI Digital Electronics FPGA Board is designed and intended for use as a development platform for hardware or software in an educational/professional laboratory environment. To facilitate usage, the board is manufactured with its components and connecting traces openly exposed to the operator and the environment. As a result, ESD sensitive (ESDS) components on the board, such as the semiconductor integrated circuits, can be damaged when exposed to an ESD event. To indicate the ESD sensitivity of the NI Digital Electronics FPGA Board, it carries the symbol shown below.

Handling the NI Digital Electronics FPGA Board can damage the board components if ESD prevention measures are not applied. **Before handling or setup, equalize your potential with the board by touching one of the integrated ESD discharge pads.** During all handling and setup, ESD prevention measures must be applied. In addition, the NI Digital Electronics FPGA Board should be handled by the edges. Touching exposed circuits, components or connectors could result in an ESD event.

2. UCF File Constraints

Slide Switches

The UCF file constraints for the eight slide switches, SW0 to SW7, are listed as follows. SWx refers to the slide switch line, LOC indicates the FPGA line location, and IOSTANDARD is the I/O standard used.

```
Net "SW0" LOC="J11" | IOSTANDARD = LVCMOS33;
Net "SW1" LOC="J12" | IOSTANDARD = LVCMOS33;
Net "SW2" LOC="H16" | IOSTANDARD = LVCMOS33;
Net "SW3" LOC="H13" | IOSTANDARD = LVCMOS33;
Net "SW4" LOC="G12" | IOSTANDARD = LVCMOS33;
Net "SW5" LOC="E14" | IOSTANDARD = LVCMOS33;
Net "SW6" LOC="D16" | IOSTANDARD = LVCMOS33;
Net "SW7" LOC="B16" | IOSTANDARD = LVCMOS33;
```

Push Buttons

The UCF file constraints for the four push buttons, BTN0 to BTN3, are listed as follows. BTNx refers to the push button line, LOC indicates the FPGA line location, and IOSTANDARD is the I/O standard used.

```
Net "BTN0" LOC="C13" | IOSTANDARD = LVCMOS33;
Net "BTN1" LOC="D12" | IOSTANDARD = LVCMOS33;
Net "BTN2" LOC="C12" | IOSTANDARD = LVCMOS33;
Net "BTN3" LOC="C10" | IOSTANDARD = LVCMOS33;
```

LEDs

The UCF file constraints for the eight LEDs, LED0 to LED7, are listed as follows. LEDx refers to the LED line, LOC indicates the FPGA line location, IOSTANDARD is the I/O standard used, SLEW refers to the slew rate, the maximum rate of change of a signal, and DRIVE indicates the current drive strength on the FPGA in milliamps.

```
Net "LED0" LOC="C11" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8;
Net "LED1" LOC="D11" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8;
Net "LED2" LOC="B11" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8;
Net "LED3" LOC="A12" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8;
Net "LED4" LOC="A13" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8;
Net "LED5" LOC="B13" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8;
Net "LED6" LOC="A14" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8;
Net "LED7" LOC="B14" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8;
```

3. Basics of Memory

In digital electronics memory plays a great role. In principle there are two types of memory volatile and non-volatile memory. Examples for non-volatile memory are ROM, PROM, EPROM and EEPROM which are used for storing firmware; these memories are often referred as secondary. The second type of memory is the volatile or primary memory. This memory is used to provide a fast response.

Memory is created by using logic gates. In combination, logic gates can create circuits that store bits of information. These circuits can be replicated to provide multiple bit storage. Amplifying the number of the bits leads to the large storage capacity in today's memory chips.

The most basic circuit that stores a bit is the SR (set/reset) latch, composed of two NOR gates. This circuit will be used to output only one state, and it will be either 1 or 0.

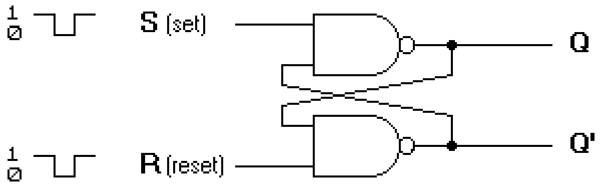

Figure 1: SR latch

Whenever we input something in S or R that value is going to show up in the output as either Q or Q`. Both Q and Q` can be on at the same time but this should be avoided, other than that they are suitable for representing/storing a bit.

To refresh your memory, a NOR gate truth table:

Table 1: NOR gate truth table

A	B	X
0	0	1
1	0	0
0	1	0
1	1	0

Now take a look at what happens when we input something in the SR latch before and after we input a value:

Table 2: SR latch input loops

Step	R	S	Q	Q`	Q(next)	Q`(next)
1	0	0	?	?	Q	Q`
2	0	1	Q	Q`	Q	0
3	0	1	Q	0	1	0
4	0	1	1	0	1	0
5	0	0	1	0	1	0

In row 1 we have no input, and we can clearly see that the circuit is looping whatever state it had before. In row 2 we set S to 1 and continue to go through the logics. In this step we get that Q will be Q again but Q` will change to 0. We feed these values back and we get that Q becomes 1. Running the logics again we see that our inputted values are stored even though our inputs are zeros. Remember that the two outputs should never be 1 at the same time. This logic circuit takes some time to run; the values are not stored instantly.

The SR latch is a useful circuit, but most of the time we want to store an input value whatever it is.

For this purpose we want to use a Data line (D) and a clock (CP), the logic of this circuit should map the value of D to the value of Q. In order to do this we need to add D and CP. This as you know will be done with an AND gate. To avoid both inputs to the SR latch being 1 or 0 at the same time we use a NOT gate. Our schematic now looks like this:

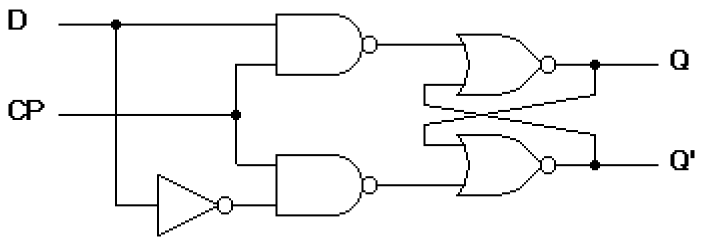

Figure 2: D latch

This circuit is called a D latch, since the output follows whatever data is put into it. To avoid complexity this latch is usually drawn as following:

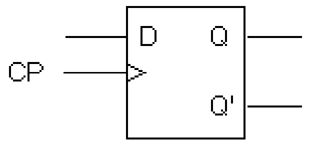

Figure 3: D latch simplified

4. Task 1: Implementing the SR-latch

In order to successfully complete this task you need to generate your programing file, download and test it on your FPGA board using the methods provided in the previous laboratory.

You have to implement the SR-latch in VHDL and test it on the Xilinx board. Use BTN0 and BTN1 as inputs and LED0 as output to your SR-latch.

5. Task 2: Implementing the D-latch

You have to implement the D-latch in VHDL and test it on the Xilinx board. Use BTN0 (as data) and BTN3 (as clk) as inputs and LED0 as output to your D-latch.

6. Task 3: Saving the switch state

For this task please use SW0 and SW1 as inputs to turn on/off two LEDs (LD0 & LD1), BTN0 to save and BTN3 to display the state of the LEDs .

- The LEDs are switched on and off when operating the switches.
- When button 0 is pushed, the current state of the LEDs is saved.
- When button 3 is pushed, the saved state is presented by the LEDs (switches should be turned off before displaying).

7. Task 4: Extend your code to support 6 LEDs

- Add the other LEDs to your circuit

8. Bonus Task 5: Second state

- Add another pair of buttons so that you can save another combination of LEDs.

9. Bonus Task 6: Saving the state of a seven segment Display

- Use the switches to input a binary number, decode the number into decimal.
- Save the state of this number with one button and display it with another.