EXPERIMENT 4

4. Electrical Characteristics of Gates and Design of Register

4.1 Objectives

- To learn about propagation delays and ring oscillator.
- To learn about timing and functional simulation.
- To learn more about clock generation, control, and frequency.
- To learn the use of oscilloscopes.
- To learn how to use a register with *Asynchronous-Clear* and analyze its behavior.

4.2 Overview

This lab is divided into two parts. The first part requires you to develop a ring oscillator and study different properties of gates with the use of oscilloscope. In the second part, you will design a register with asynchronous clear capability and study its behavior.

You will also be introduced to the functions of oscilloscope, a device that displays the behavior of your signals.

To verify the correctness of your design, you will be performing functional and timing simulations of your design. Finally you will be downloading the design on the FPGA board.

4.3 Design Specifications

A ring oscillator is a chain of odd number of NOT-gates (inverters), where the output of the last inverter in the chain is feed backed and connected to the input of the first inverter as shown in Figure 4.1.

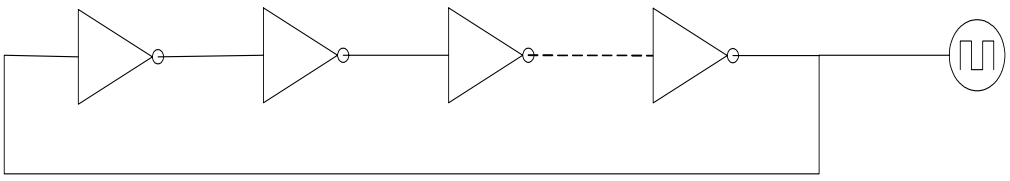

Figure 4.1: Ring Oscillator Diagram.

As the name suggests, a ring oscillator is a circuit that produces oscillations. It does so because of the ring-like nature in which odd number of inverters is connected.

To understand the oscillatory behavior of the ring-oscillator, note that any logic value at the start of the chain (input of first inverter) gets inverted at the end of the chain. For example, consider a high signal (logic-1) at the input of first inverter. This value produces a logic low (logic-0) at its output. This low causes the output of second inverter to be high. This alteration of the logic values continues till the last inverter in the chain. Since the number of inverters is odd, a high at the inputs of first is reflected as low at the last. Since the output of the last inverter is feed backed to the input of first, this toggling between logic high and logic low continues, producing the desired oscillatory behavior. The time it takes for logic high to be pulled to logic low and vice versa depends on propagation delay of the inverters connected in the chain.

The output of the ring oscillator is a signal that is similar to clock waveform as shown in Figure 4.2.

Figure 4.2: Ring Oscillator Output.

Since the ring oscillator can give you the delay in propagation of the signal from the input to the output, you can measure the delay of each gate, by dividing the total delay measured with the number of gates.

There are two types of digital circuits: combinational and sequential. The ‘register’ belongs to the sequential type of circuits. A register is a ‘memory element’. You can read from or write to a register. Many capabilities can be associated with registers, and based on these capabilities, the behavior and performance of registers can be evaluated. In your experiment, you can build a 4-bit register, using devices available in Spartan library. It should perform according to the following function table depending on the two function inputs (CLR and CE):

Table 4.1: Register Response to the Inputs CLR & CE

CLR	CE	Function
0	0	No operation – contents of register do not change
0	1	Register takes the values at the inputs (parallel load)
1	0	No operation – contents of register do not change
1	1	The contents of the register are “cleared”

4.4 Pre-Lab

- Read the portion on simulation in lab-manual
- Read about registers in Chapter-5 of the textbook
- Bring the schematic of the circuits covered in this lab.

4.5 In-Lab

4.5.1 Ring Oscillator

1. Start a new schematic, where you will design a schematic of ring oscillator of 11 gates by using NAND gates only and attach an AND gate with the first NAND gate in the chain (as shown in Figure 4.3), where one of inputs of AND gate is connected to a switch. Do timing simulation of this device (You will not be able to do functional simulation here. Try that and see what happens!). Watch the oscillator output as the switch is flipped between 0 & 1.
2. Measure the delay of the whole circuit using the timing simulation. Then, calculate the delay of one NAND gate. (see **Error! Reference source not found.** section of the lab guide)

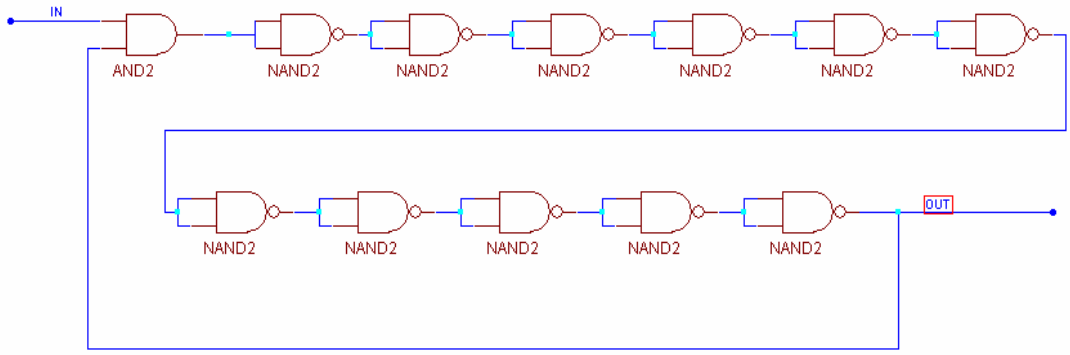

Figure 4.3: Ring Oscillator with Enable Input.

4.5.2 Register:

1. Start the Xilinx project manager, configure the device and name the project as lab4b.
2. Start a new schematic, where you will use a device FD4CE in the Spartan library that implements a 4-bit register with the above mentioned capabilities.
3. FD4CE has an asynchronous Clear (CLR) input. Connect it to an input pad through an input buffer, and constraint it to a push button (BTN2) and similarly clock (C) input to another push button (BTN1) as shown in Figure 4.4.
4. Connect Chip Enable (CE) to SW5.
5. Constraint the four inputs (D0, D1, D2, and D3) to four level switches (SW1, SW2, SW3, and SW4).

6. Constraint the four outputs (Q0, Q1, Q2, and Q3) to the four LEDs (LD1, LD2, LD3, and LD4).
7. Connect the other inputs of multiplexers as shown in Figure 4.4.
8. Perform an integrity test. If there is any error, remove it.
9. Verify functionality of the design by simulating it.
10. Implement the design by downloading the design on Digilab board.
11. Verify its functionality on the board.
12. How to verify the circuit:
13. Make CE = 0 and CLR = 0. Now change inputs at D0 to D3. Nothing will happen to the output LEDs.
14. Load parallel 4-bit data into the register through four input switches, by making CLR= 0 and CE = 1 and, and by pressing the push button (BTN1) for clock. Observe the four output LEDs. These should show same data that you have loaded in parallel.
15. Now make CLR = 1 and CE = 0, and apply the clock pulse by pressing BTN1. Nothing should happen to the output.
16. To clear the contents (data) of the register *asynchronously*, make CLR = 1 by pressing (BTN2) and CE = 1 by making SW5 equal to logic 1. Observe the four output LEDs. These should show logic 0 at the output. Notice that there will be no effect due to the clock here.

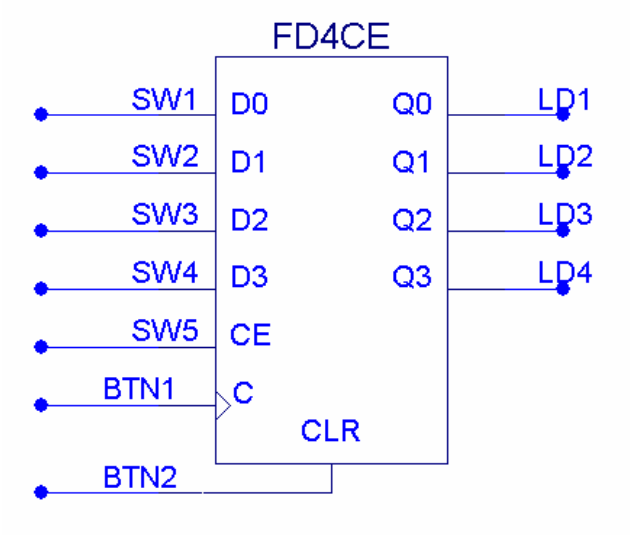

Figure 4.4: Four-Bit Register.

4.5.3 Debounce Problem:

While shifting the data right or left, you might face a problem that with a push of clock button, the data may shift more than one position. It should shift only single position to the right or left with a single push. This happens due to a debounce problem in the push buttons.

What is it? Push-button switches have metal contacts that make and break the circuit and since at least one of the contacts is on a movable strip of metal, it has

springiness. Because the moving contacts have mass and springiness with low damping they will be bouncy as they make and break. That is, when a normally open pair of contacts is closed, the contacts will come together and bounce off each other several times before finally coming to rest in a closed position as shown in Figure 4.5. In logic circuits, it causes several pulses instead of one.

Figure 4.5: Debounce Problem.

Solution*:* This problem can be solved by using an SR NAND latch with switch contacts connected to +5 volts. Bounce is ignored since that condition results in inputs of S =1, R = 1, which is a no change condition. This is shown on the Figure 4.6 below.

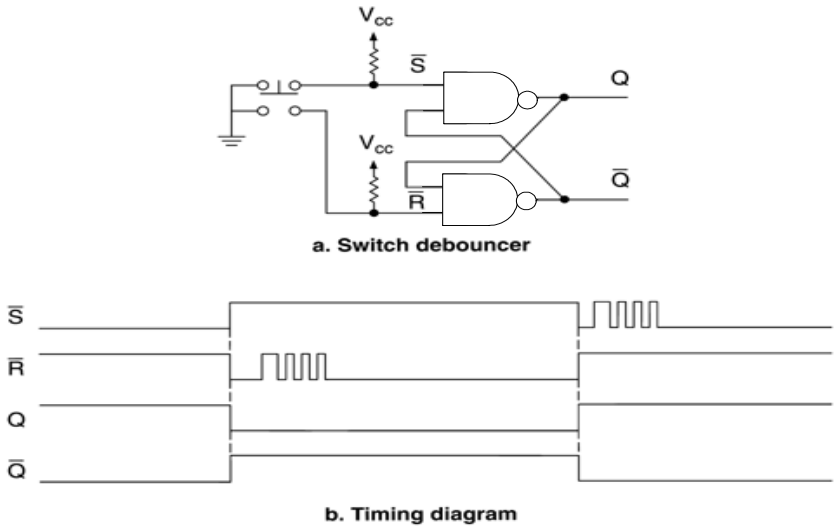

Figure 4.6: Debounce Free Switch.

4.6 Post-Lab

Document your designs and provide a lab report.

You can use blank boxes under each input/output unit in the diagram below to remember its usage.