ECED2200 – Lab #4

STUDENT NAME(s): ____.

STUDENT NUMBER(s): B00 __________________________________.

Pre-Lab Information

It is recommended that you read this entire lab ahead of time. Doing so will save you considerable time during the lab.

There is also several videos showing the use of the software tools – you will save considerable time by watching those now. See http://colinoflynn.com/teaching/eced2200-intro-to-digital-circuits/ and look at the 'lab 4' information.

Overall Objective

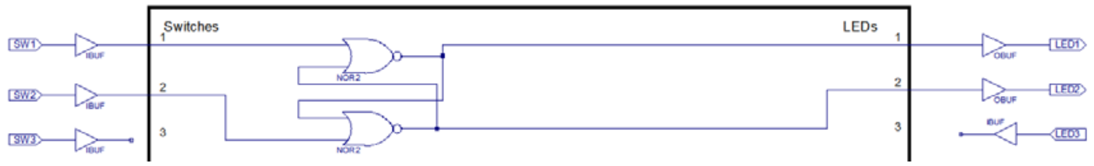
This lab has several objectives. In the first part you will learn about how flip-flops can be built from simple gates. In the second part you will use flip-flops as part of a larger counter circuit.

Deliverables

You may work in groups of 2 for this lab. Each group must deliver a lab report; the format for it is described separately.

Understanding State Transition Tables

This lab makes extensive use of 'State Transition Tables', which are described in this section. Generally, you will be implementing a circuit. For example in Part 1 you implement the following circuit:



The State Transition Table asks you to look at this circuit as a 'Black Box', which is to say you *do not* care what is inside the circuit. This circuit has two inputs (SW1, SW2), and two outputs (LED1, LED2).

For all of these circuits in Part #1 (1-A,1-B,1-C), LED1 is the 'Q' output. LED2 is the 'NOT Q' output. LED1 & LED2 should always be opposite states of each other if the circuit is functioning correctly.

Digital Circuits – Lab #4

Note that some of the circuits have 'illegal' or 'invalid' inputs, when you put those inputs into the circuit LED1 & LED2 will no longer be opposite. Depending on the circuit this might mean that LED1 & LED2 are both on, or LED1 & LED2 are both off. Such states are INVALID.

Here is what part of the state transition table you are filling out looks like:

SW1 (Final State)	SW2 (Final State)	Q (LED1 Initial State)	Q^+ (LED1 Final State)
		0	0
		0	1

<u>Procedure For Filling Out Each Row</u>

1. Start with Row #1
2. Look at the state of Q given, which in this example for Row #1 is initially '0'.
3. If your LED1 is not in the state of Q given (e.g. in this example OFF), toggle switches until LED1 is in the correct state (again OFF for Row #1). Remember LED1 & LED2 should be opposite, so LED2 should be ON if LED2 is off. If they are both OFF or both ON that *does not* count as a legitimate state of SW1 & SW2.
4. Your circuit is now in the 'Initial' state. You *do not* care about the current state of switches SW1 & SW2, they are at whatever state they may happen to be at.
5. Your objective is now to Make LED1 go to the state in the Q^+ column by changing a single switch at a time. Toggle switches SW1 or SW2, and see if LED1 is at the requested state in the Q^+ column, and LED2 is the opposite of LED1. If so you are done. Write down the state of SW1 & SW2 (e.g. 1 or 0). This state is what you record in SW1 & SW2 column.
6. If the Initial (Q) and Final (Q^+) state are the same (e.g. 0→0), the question is simply asking you to keep LED1 the same. If you toggle a switch and LED1 changes, you've got to get LED1 back to the correct 'Initial' state and try again.
7. Repeat for all rows of the table. You may have to toggle SW1 & SW2 to get LED1 into a different initial state.

Some of the rows have TWO entries for the same transition (e.g. Initial→Final = 0→0). This means there is TWO ways to go from the final state from the given initial state. One of these ways is actually to do nothing – that is toggle the switches until LED1 is in the requested state (e.g. OFF for Row #1). Then write down the 'final' state of SW1 & SW2.

Remember the question I'm asking you is "If LED1 is in the current state given by Q, what should I set the switches to such that LED1 is in the new state given by Q^+?". Sometimes the answer is that I don't need to modify the switches from their current setting.

We will use these tables in the design of logic circuits that can keep state. They are useful since they tell us that if we want to move an output from a certain state (e.g. ON/OFF) to a new state, what are possible ways of doing this based on the given inputs.

Part #1: Implementing Latches

Objective

- Learn about different types of Flip-Flops

Required Materials

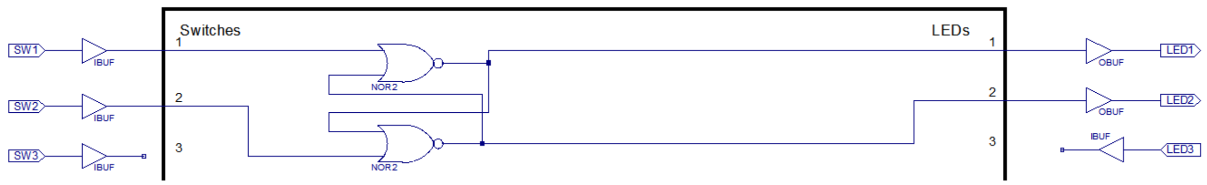
- Computer with Xilinx ISE 13.2 Webpack installed.
    - All computers in the lab have this installed.
    - This is free software so you can install on your own computer if you wish, you can download it from http://www.xilinx.com/support/download/index.htm - select '13.2' on the side. The file is very large so you may wish to download at school, and you are required to register to license it.
- Example project file DigitalTrainer_Simple.ZIP
    - This file contains an environment which is already setup for your lab.
- Digital Trainer Board

Background

Procedure

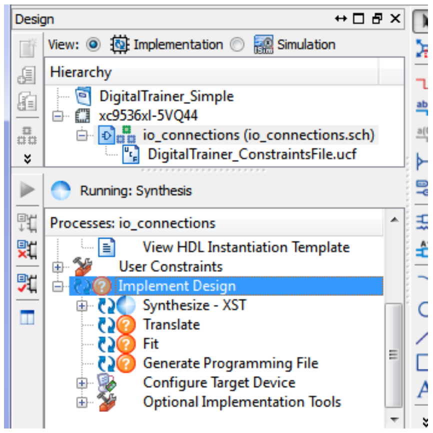
Part 1-A: RS-Latch

1. Download DigitalTrainer_Simple.zip from one of these locations:
    a. www.colinoflynn.com/teaching under the ECED2200 Lab #4 as 'ISE Project File'.
2. **Unzip** this somewhere. Do not simply open it, you must **copy all files out**.
3. Open the DigitalTrainer_Simple.xise
4. Open io_connections.sch
5. Build an RS Latch with NOR gates. An example implementation is shown below:

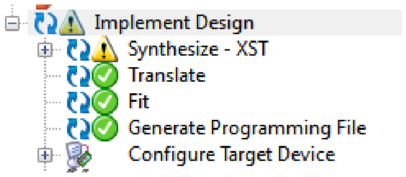


NOTE: LED1 is "Q" and LED2 is "NOT Q". Thus LED1 and LED2 should always be opposite of each other. You are ALWAYS looking at LED1 (Q) for filling out the observations.

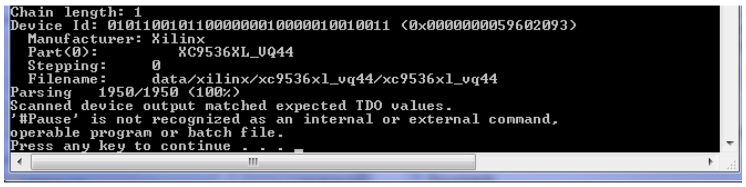
6. As before, save and close the schematic, then double-click the 'Implement Design' process:


Design
View: Implementation Simulation
Hierarchy
DigitalTrainer_Simple
xc9536xl-5VQ44
io_connections (io_connections.sch)
DigitalTrainer_ConstraintsFile.ucf
Running: Synthesis
Processes: io_connections
View HDL Instantiation Template
User Constraints
Implement Design
Synthesize - XST
Translate
Fit
Generate Programming File
Configure Target Device
Optional Implementation Tools

7. You should get green Check-marks beside ‘Generate Programming File’:



8. Plug in your Digital Explorer board.
9. Run ‘program.bat’ as before, and again you should see an indication it is successful:

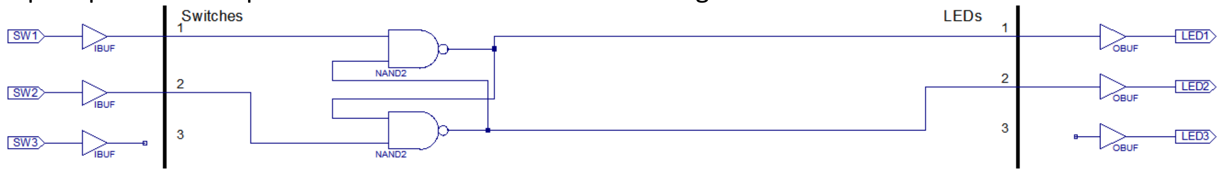


```
Chain length: 1
Device Id: 01011001011000000001000000010010011 (0x0000000059602093)
  Manufacturer: Xilinx
  Part(0):          XC9536XL_VQ44
  Stepping:     0
  Filename:     data/xilinx/xc9536xl_vq44/xc9536xl_vq44
Parsing   1950/1950 (100%)
Scanned device output matched expected TDO values.
'#Pause' is not recognized as an internal or external command,
operable program or batch file.
Press any key to continue . . . _
```

10. Using SW1, SW2, LED1, LED2 fill out the transition table and answer any questions given in the observation section.

Part 1-B:

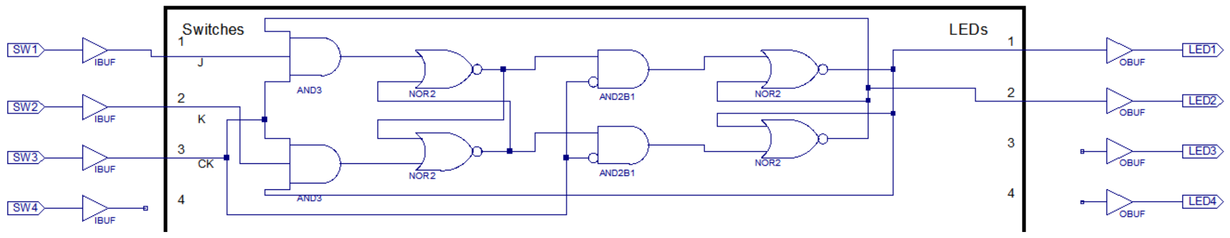
Repeat part 1-A except with the RS Latch built from NAND gates:



NOTE: LED1 is “Q” and LED2 is “NOT Q”. Thus LED1 and LED2 should always be opposite of each other. You are ALWAYS looking at LED1 (Q) for filling out the observations.

Part 1-C:

Repeat part 1-A, except with the JK Flip-Flop:



NOTE: LED1 is "Q" and LED2 is "NOT Q". Thus LED1 and LED2 should always be opposite of each other. You are ALWAYS looking at LED1 (Q) for filling out the observations.

Note that this part has a CLOCK input. SW1/SW2 are ONLY read when you toggle the clock low-high-low. Thus for each test of a value of SW1/SW2 you'll need to pulse the clock (SW3).

Observations

Part 1-A: NOR Based RS Latch

Q1) Complete the state transition table. It may take some experimentation to get from one state to the next. Where two lines are given for the same transition there may be more than one method of getting that transition to occur. The state transition table shows how we move from the state given by Q to the state given by Q^+.

SW1	SW2	Q (LED1 Initial State)	Q^+ (LED1 Final State)
		0	0
		0	0
		0	1
		1	0
		1	1
		1	1

Q2) What are the inputs SW1 and SW2 (which is Reset & Set). Are they active high or active low? What is the invalid state in this flip-flop (e.g.: SW1=?, SW=?) and in that invalid state what are the outputs?

Part 1-B: NAND Based RS Latch

Q1) Complete the state transition table. It may take some experimentation to get from one state to the next. Where two lines are given for the same transition there may be more than one method of getting that transition to occur. The state transition table shows how we move from the state given by Q to the state given by Q^+.

SW1	SW2	Q (LED1 Initial State)	Q^+ (LED1 Final State)
		0	0
		0	0
		0	1
		1	0
		1	1
		1	1

NOTE: LED1 is 'Q', and LED2 is 'NOT Q'. Remember from the description in the 'backgronud' section you are *always looking at the Q output (e.g. LED1). This Table DOES NOT consider the state of LED2. If LED2 and LED1 are NOT opposite (e.g. LED1 is ON & LED2 is ON, or LED1 is OFF & LED2 is OFF), the input is INVALID.*

Q2) What are the inputs SW1 and SW2 (which is Reset & Set). Are they active high or active low? What is the invalid state in this flip-flop (e.g.: SW1=?, SW=?) and in that invalid state what are the outputs.

Part 1-C: JK Flip-Flop

Q1) Complete the state transition table. It may take some experimentation to get from one state to the next. Where two lines are given for the same transition there may be more than one method of getting that transition to occur. The state transition table shows how we move from the state given by Q to the state given by Q^+. Note for the JK flip-flop you must use the CLOCK line to cause the state transition to occur. So the procedure is:

1) The current state of LED1 is the value of Q, thus the current state.
2) Set SW1/SW2 (J/K) to the desired value. Note unlike the Part 1-A or Part 1-B the circuit will do nothing until you pulse SW3, thus it doesn't actually notice changes in the input switches until you pulse SW3 described below.
3) Pulse SW3 (CLK) to 1 then back to 0, and observe the transition that occurred.

SW1 (J)	SW2 (K)	Q (LED1 Initial State)	Q^+ (LED1 Final State)
		0	0
		0	0
		0	1
		0	1
		1	0
		1	0
		1	1
		1	1

Q2) On which clock edge is this flip-flop changing state? E.g. does the output changes when you move the CLK (SW3) from 0→1 (Falling Edge) or 1→0 (Rising Edge)?

Part #2 – Using Registers for a Simple Counter

Objective

- Implement a simple counter.

Required Materials

- Computer with Xilinx ISE 13.2 Webpack installed.
  - All computers in the lab have this installed.
  - This is free software so you can install on your own computer if you wish, you can download it from http://www.xilinx.com/support/download/index.htm - select '13.2' on the side. The file is very large so you may wish to download at school, and you are required to register to license it.
- Example project file DigitalTrainer_Simple.ZIP
  - This file contains an environment which is already setup for your lab.
- Digital Trainer Board

Background

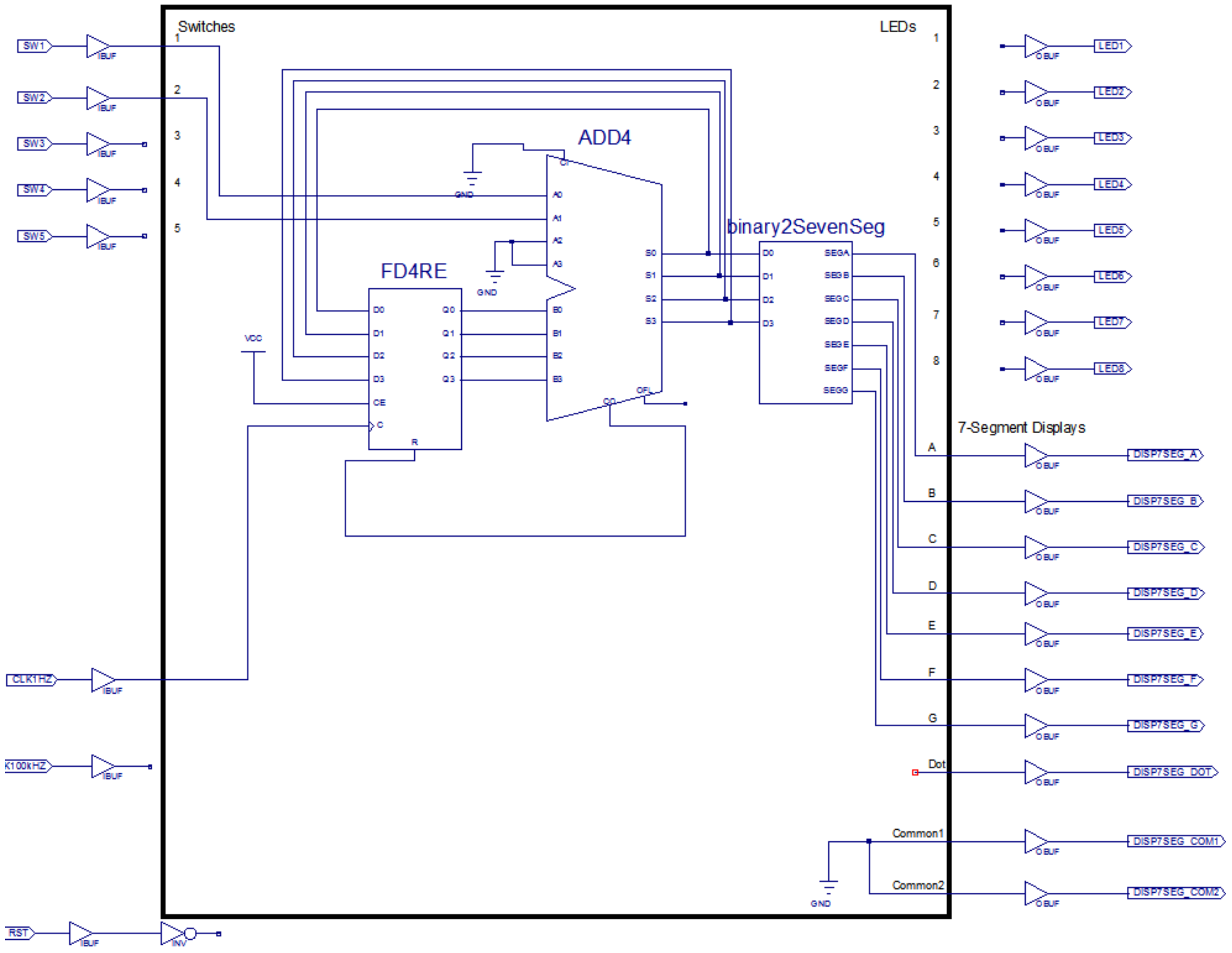
We will be later learning about structures of simple counters. For now though let's use what we have already become familiar with: a simple D flip-flop, and an adder circuit from a previous lab. With this we can create a simple counter.

Note the 7-segment display will be used again. This time it will output a **hexadecimal** code, so valid values are 0-9,A-F.

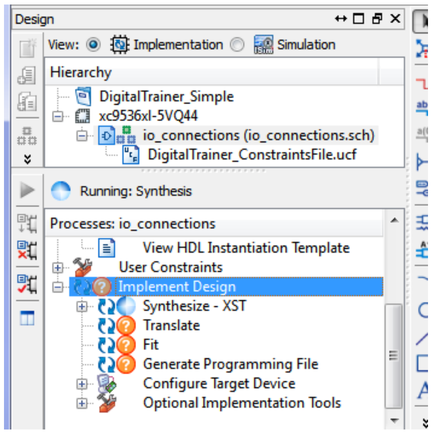
Procedure

Part 2-A

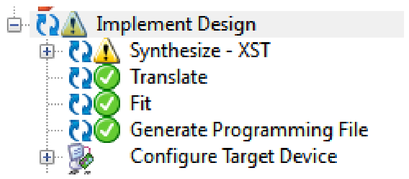
1. Download DigitalTrainer_Simple.zip from one of these locations:
    a. www.colinoflynn.com/teaching under the ECED2200 Lab #4 as 'ISE Project File'.
2. **Unzip** this somewhere. Do not simply open it, you must **copy all files out**.
3. Open the DigitalTrainer_Simple.xise
4. Open io_connections.sch
5. The schematic shown below requires five components. Draw the schematic as given, provided that:
    a. FD4RE is found in the 'Flip_Flop' category
    b. ADD4 is found in the 'Arithmetic' category
    c. Binary2SevenSeg is found in the first category. Do not use BCD2SevenSeg, be sure to use Binary2SevenSeg this time.
    d. VCC is found in 'General'
    e. GND is found in 'General'



6. As before, save and close the schematic, then double-click the ‘Implement Design’ process:


Design
View: Implementation Simulation
Hierarchy
DigitalTrainer_Simple
xc9536xl-5VQ44
io_connections (io_connections.sch)
DigitalTrainer_ConstraintsFile.ucf
Running: Synthesis
Processes: io_connections
View HDL Instantiation Template
User Constraints
Implement Design
Synthesize - XST
Translate
Fit
Generate Programming File
Configure Target Device
Optional Implementation Tools

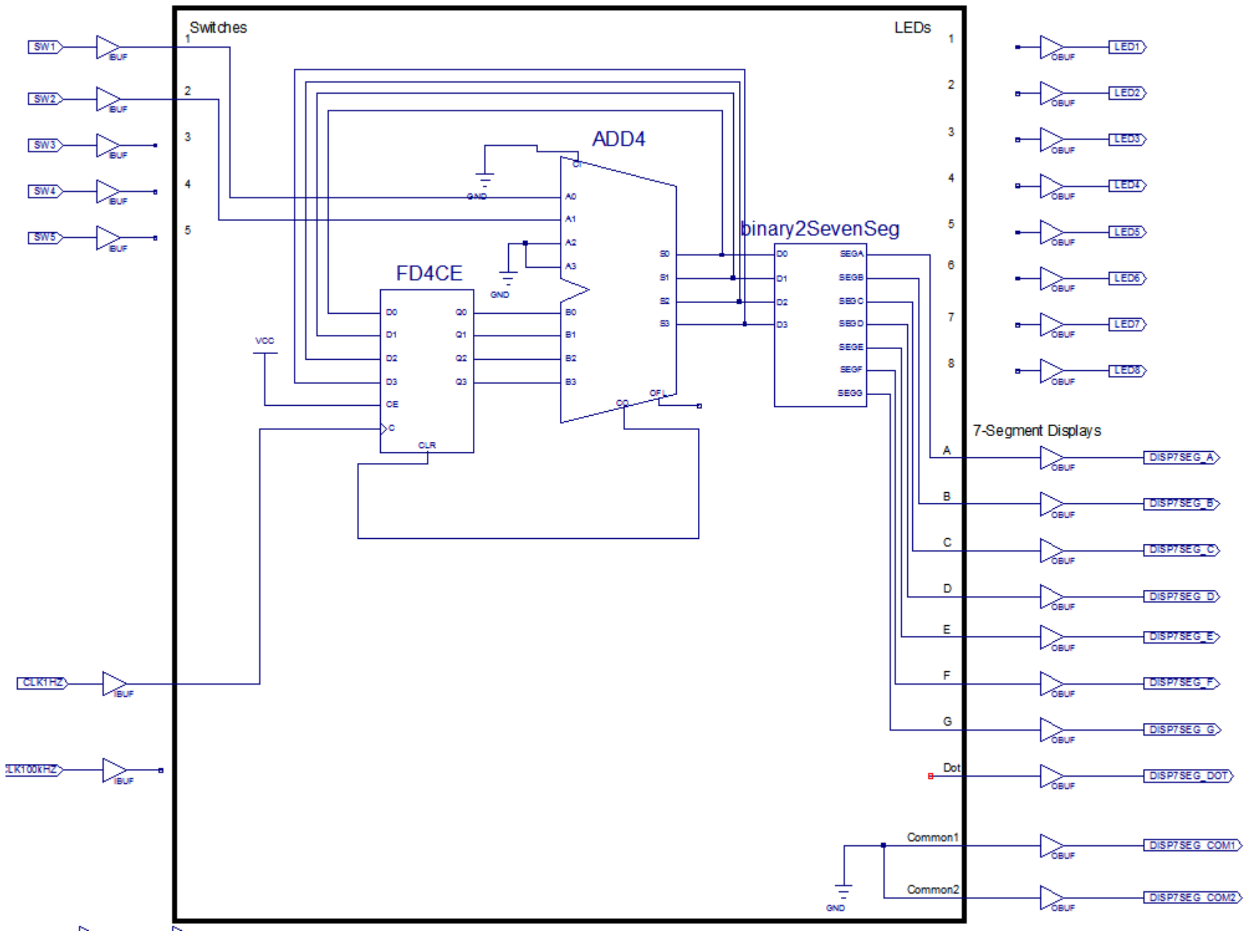
7. You should get green Check-marks beside ‘Generate Programming File’:



8. Plug in your Digital Explorer board.
9. Run 'program.bat' as before, and again you should see an indication it is successful.
10. Answer the questions in Observations.

Part 2-B

11. Replace the FD4RE part with a FD4CE part. In which case your schematic looks as such:



12. Answer the questions in the Observations.

Observations

Part 2-A

Q1)Play with the switches SW1 & SW2. What does changing the setting of the switches do to the count sequence you observe?

Q2) Fill out the following table:

Switch Settings		Count Sequence															
SW1	SW2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0																
0	1																

<u>Part 2-B</u>

Q1) Set SW1 = 1, SW2 = 0. This should result a count sequence of 0,1,2,3,…,d,e,f. What is the sequence now?

Switch Settings		Count Sequence															
SW1	SW2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0																

Q2) FD4RE has a synchronous reset, that is to say it is only valid on the rising clock edge. FD4CE has an asynchronous clear; that is to say as soon as it goes high, even for an instant, the output is cleared. Why do you think the asynchronous clear is giving us problems in this implementation?

Conclusion

This lab has introduced some flip-flops built from simple gates, and also shows some potential problems with using flip-flops incorrectly.