Digital Logic
CSC 244L
Laboratory 4
Intro to Sequential Circuits: Latches and Flip-Flops

1 Objectives

1. implement hierarchical modules for Latches and Flip-Flops in System Verilog;
2. Clean noisy inputs using a debouncer module;
3. Practice SV, Quartus, and FPGA design, implementation, and testing.

2 Pre-Laboratory Procedure

Illegible work and files that do not open in the D2L dropbox will result in reduced grades. It is your responsibility to ensure that what you turn in is readable by the TAs. Please read the submission instructions and follow them to ensure you receive all points. The circuit diagrams for pre-lab may be neatly hand drawn. To be completed the midnight before your lab meets (individually):

2.1 Read the entire lab procedure, and Chapters 3.1–3.2 in your book;

2.2 Complete these items (in this lab manual) for your pre-lab:

- 3.1.1.1, 3.1.1.2
- 3.1.2.1, 3.1.2.2
- 3.2.1.1
- 3.2.2.1
- 3.2.3.1

2.3 Bring your circuit diagrams, block diagrams and bring soft copies of all your SV modules to lab to be compiled and loaded onto the FPGA for testing and demonstration.

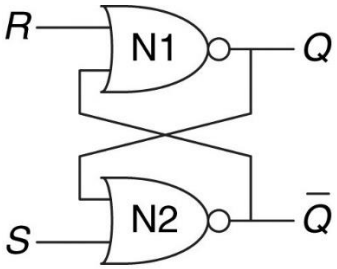
You can come to lab with SystemVerilog modules that compile the very first time. You will ensure that your files compile correctly by compiling them at home with Quartus. A proactive student would also test that the compiled SV works on their DE10-Lite board (allowing the student to leave early!).

3 Laboratory Procedure

We will investigate the sequential circuits discussed in class using the Intel DE10-Lite FPGA boards. Recall that latches are level-sensitive memory elements, and flip-flops are edge-triggered.

3.1 Latches

3.1.1 SR-Latch



Case	S	R	Q	$\bar{Q}$
IV	0	0	Q_{prev}	$\bar{Q}_{prev}$
I	0	1	0	1
II	1	0	1	0
III	1	1	0	0

Figure 1: A 1-bit SR-Latch using cross-coupled NOR gates. The characteristic table shows the four possible cases.

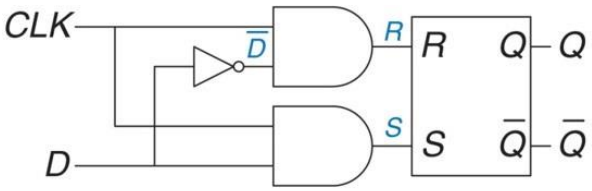
3.1.1.1 Read Section 3.2.1 of your textbook.
3.1.1.2 Create a SV file named "sr_latch.sv" that contains one SV module named sr_latch. Write behavioral SV to describe the operation of a SR-latch.

3.1.1.3 Compile the SR-Latch SV using Quartus Prime Light. Assign switches SW[1:0] to the S and R inputs, and LEDR[1:0] to monitor the outputs Q and ¯Q.

3.1.1.4 Load the SR-Latch to your FPGA. Verify the operation of your SV module by checking every possible input combination. Record these values in a truth table and compare to the expected values.
3.1.1.5 Show your working FPGA module, and truth table to your TA and have them sign off on your lab sheet.

3.1.2 Gated D-Latch

3.1.2.1 Read Section 3.2.2 of your textbook.



(a)

CLK	D	$\bar{D}$	S	R	Q	$\bar{Q}$
0	X	$\bar{X}$	0	0	Q_{prev}	$\bar{Q}_{prev}$
1	0	1	0	1	0	1
1	1	0	1	0	1	0

(b)

Figure 2: (a) A 1-bit Gated D-Latch using a clock signal and an SR-Latch. (b) The characteristic table shows the possible cases.

3.1.2.2 Create a SV file named "d_latch.sv" that contains one SV module named d_latch. Write behavioral/structural SV to describe the operation of a D-latch using your SR-Latch module.
3.1.2.3 Compile the D-Latch SV using Quartus Prime Light. Assign switches SW[1:0] to the D and CLK inputs, and LEDR[1:0] to monitor the outputs Q and ¯Q'.
3.1.2.4 Load the D-Latch to your FPGA. Verify the operation of your SV module by checking every possible input combination. Record these values in a truth table and compare to expected values. Show your working FPGA module and truth table to your TA and have them sign off on your lab sheet.

3.2 Flip-Flops

For all SV portions of this section, you will need to *debounce* the noisy clock signal that will come from your KEY[0] button. Because the key is mechanical, when pressed the electrical contact *bounces* multiple times before settling to the correct value. You can see an o-scope capture of the noisy key inputs from your DE10-Lite Board in Fig. 3.

If this signal is directly input to a circuit that relies on a clock (CLK), that circuit will see multiple active CLK edges and malfunction. To overcome this, a *debouncer* circuit or SV module can be used so that only one clean CLK edge is seen by your circuit. Mr. Galipeau has provided a SV module called **debouncer** for you to accomplish this for this lab and Project 1 ("debouncer.sv"). The module takes two inputs (A_noisy and CLK50M) and outputs the clean signal A. The DE10-Lite inputs for the module are provided in Table 3.2. The output should be connected to your internal CLK **logic**.

Table 1: Inputs for the debouncer module.

Module Input	DE10-Lite Input	Pin
A-noisy	KEH[0]	PIN_B8
CLK50M	50 MHz Clock	PIN_P11

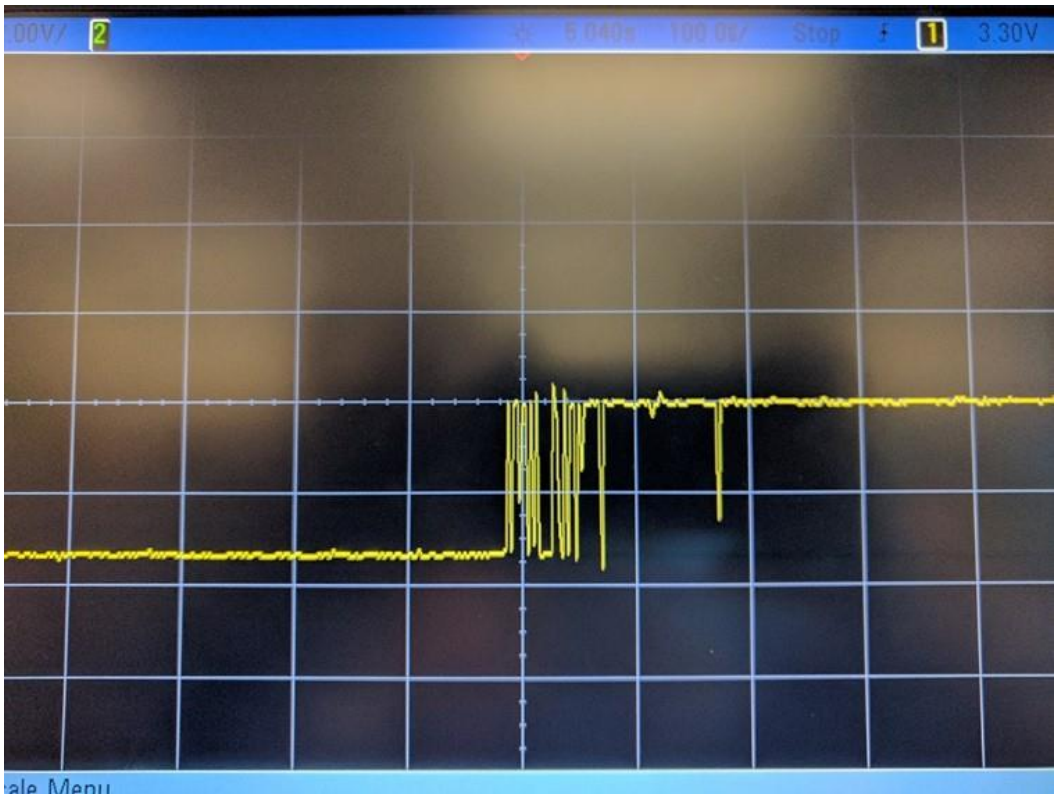


Figure 3: The noisy key input of the DE10-Lite board captured on the oscilloscope.

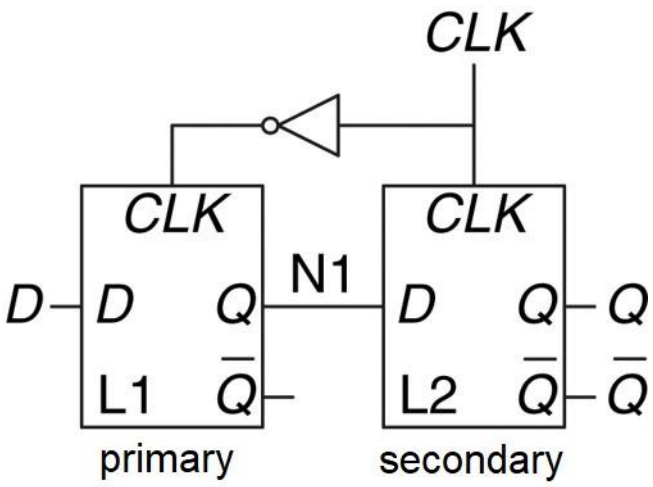


Figure 4: A 1-bit positive-edge triggered D Flip-Flop using a master-slave configuration of two gated D-latches.

3.2.1 D Flip-Flop

3.2.1.1 Read Section 3.2.3 of your textbook.

3.2.1.2 Create a SV file named "d_ff.sv" that contains one SV module named d_ff. Write behavioral/structural SV to describe the operation of a D-FF using two of your gated D-Latch modules.

3.2.1.3 Create a top-level file called "ff_toplevel.sv" that connects SW[0] to the D input of your D-FF and LEDR[1:0] to monitor the outputs Q and $\bar{Q}$. Connect the debouncer module between KEY[0] and the CLK input of your D-FF.

3.2.1.4 Compile and load the top-level file to your FPGA. Verify the operation of your SV module by checking every possible input combination. Record these values in a truth table and compare to expected values. Show your working FPGA module, and truth table to your TA and have them sign off on your lab sheet.

3.2.2 T Flip-Flop

A T Flip-Flop, or toggle Flip-Flop, holds the prior value on the active clock edge if T = 0, otherwise it toggles its output if T = 1. This component is useful in counting circuits, clock circuits, and more. There is no discussion in your book, but you can read about them here: https://www.electronicshub.org/t-flip-flop/.

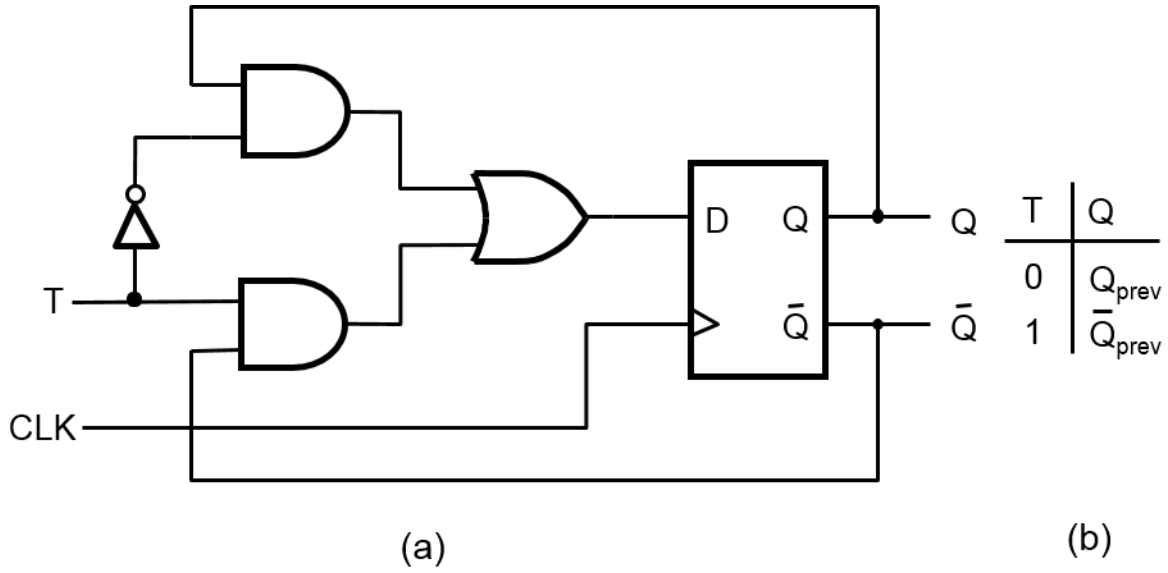


Figure 5: A 1-bit positive-edge triggered T Flip-Flop.

3.2.2.1 Create a SV file named "t_ff.sv" that contains one SV module named t_ff. Write behavioral/structural SV to describe the operation of a T-FF using your D Flip-Flop module.

3.2.2.2 Replace the D-FF in "ff_toplevel.sv" with your T-FF module.

3.2.2.3 Compile and load the T-FF to your FPGA. Verify the operation of your SV module by checking every possible input combination. Record these values in a truth table and compare to expected vales. Show your working FPGA module, and truth table to your TA and have them sign off on your lab sheet.

3.2.3 JK Flip-Flop

A JK Flip-Flop is known as a universal flip-flop because you can use them to create a D or T flip-flop. When building finite state machines, JK flip-flops can be used to greatly simplify your state transition logic. As most digital design is now performed on programmable logic devices, JK flip-flops have fallen out of favor for the simpler D flip-flop.

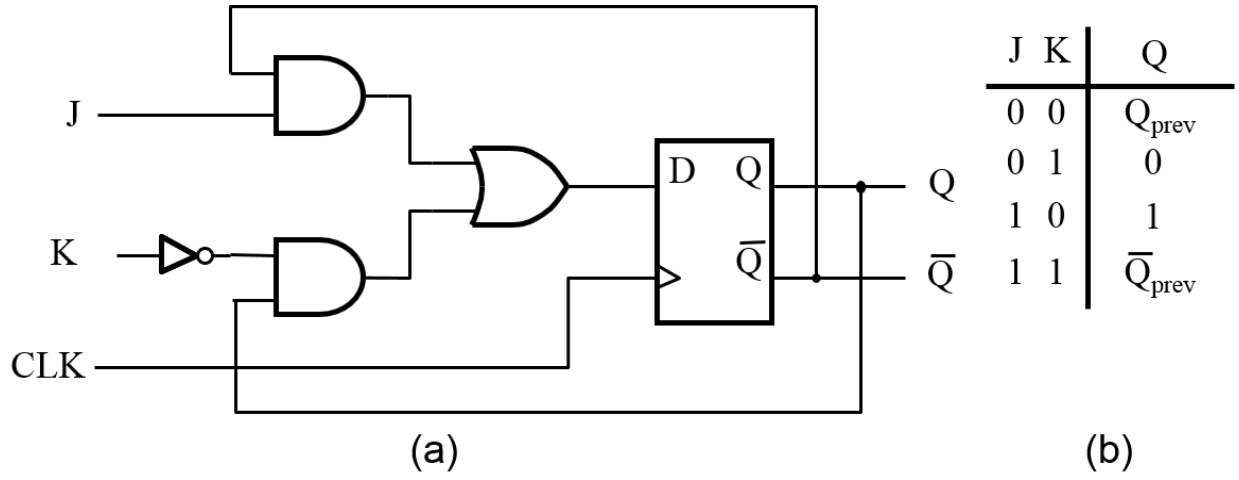


Figure 6: A 1-bit positive-edge triggered JK Flip-Flop.

3.2.3.1 Create a SV file named “jk_ff.sv” that contains one SV module named jk_ff. Write behavioral/structural SV to describe the operation of a JK-Flip-Flop using your D Flip-Flop module.
3.2.3.2 Replace the T-FF in “ff_toplevel.sv” with your JK-FF module. Assign switches SW[1:0] to the J and K inputs.
3.2.3.3 Compile and load the JK-FF to your FPGA. Verify the operation of your SV module by checking every possible input combination. Record these values in a truth table and compare to expected values. Show your working FPGA module, and truth table to your TA and have them sign off on your lab sheet. (good work!)

Lab Check off

Pre Lab		Points
SR Latch		5
D Latch		5
D Flipflop		5
T Flipflop		5
JK Flipflop		5
Lab Demos		
SR Latch		10
D Latch		10
D Flipflop		10
T Flipflop		10
JK Flipflop		10
total		75