CPEN 230L: Introduction to Digital Logic Laboratory

Lab # 8: Latches and Flip-Flops

Purpose

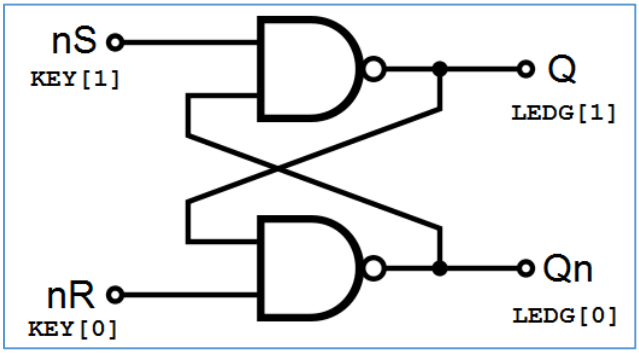
- Learn about latches, flip-flops and counters.
- Learn about the DE2-115 board push-button switches.
- Gain further competence with CAD tools and processes.

Lab Directory Structure

- Start with the provided Lab8 directory tree and files. When the lab is complete your directory structure and files should look like this:

```
Lab8
  SR_Latch                    << Part 1 directory
    src
      SR_Latch_top.v          << write it (short)
      SR_Latch.v              << write it (short)
    synth
      SR_Latch_top.tcl        << given
      SR_Latch_top.sdc        << given
      <other Quartus files>
  JK_FlipFlop                 << Part 2 directory
    src
      JK_FlipFlop_top.v       << write it (short)
      JK_FlipFlop.v           << given
    synth
      JK_FlipFlop_top.tcl     << given
      JK_FlipFlop_top.sdc     << given
      <other Quartus files>
  counter4bit                 << Part 3 directory (simulation and synthesis)
    sim
      counter4bit_tb.v        << write it
      <other ModelSim files>
    src
      counter4bit_top.v       << write it
      counter4bit.v           << write it
      JK_FlipFlop.v           << given in Part 2
      hex7seg.v               << write it
    synth
      counter4bit_top.sdc     << given
      counter4bit_top.tcl     << given
      <other Quartus files>
```

Part 1: S-R (Set-Reset) Latch



Pre-Lab:

- In the above diagram, the "n" preceding inputs S (Set) and R (Reset) indicates that they are active-low. For example, a 0 at nS means "do the Set function". The "n" in output Qn indicates Qnot or Q' or $\overline{Q}$ or "the complement of Q".
- Create a truth table for this latch with inputs nS, nR and outputs Q, Qn. For one of the four input states, Q can be either 0 or 1 with Qn being its complement. For this state just say "0 or 1" for Q and "1 or 0" for Qn.
- What combination of nS, nR inputs must be avoided when using this circuit as intended, to latch Q and Qn values?
- If this condition is allowed to occur, how can it introduce an error in the device this circuit is being used in?
- Create file **SR_Latch.v** to implement module SR_Latch with the inputs and outputs shows in the diagram.
- The 4 pushbuttons at the lower-right of the DE2-115 board are named KEY[3:0]. When not being pressed they provide a 1. When pressed they provide a 0.
- Create file **SR_Latch_top.v** to implement module SR_Latch_top that connects an instance of SR_Latch to the DE2-115 board pushbuttons and LEDs indicated in the diagram.
- Place the files in the Lab Directory Structure where indicated.
- Use Icarus Verilog or ModelSim to eliminate compile errors and warnings in the two Verilog files.

During Lab:

- Use Quartus and the DE2-115 board to build, debug and verify your pre-lab work.
- **When convinced your circuit is working properly, demonstrate it to your instructor.**
- Your report for this part should include the contents of SR_Latch.v and SR_Latch_top.v, a Quartus RTL Viewer image expanded to the gate level, discussion of any Quartus warnings, and anything else that will help your target reader.

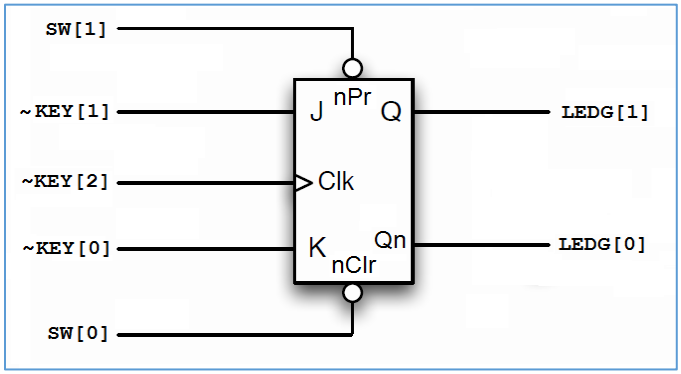
Part 2: J-K (Set-Reset-Toggle) Flip-Flop

Pre-Lab

- The following function table describes a positive edge triggered J-K Flip-Flop with active low asynchronous Preset and Clear. File **JK_FlipFlop.v** is its Verilog implementation.

Inputs					Outputs		Function
nPr	**nClr**	**Clk**	**J**	**K**	**Q**	**Qn**	
X	0	X	X	X	0	1	Clear
0	1	X	X	X	1	0	Preset
1	1	+edge	1	0	1	0	Set
1	1	+edge	0	1	0	1	Reset
1	1	+edge	1	1	~Q_0	~Qn_0	Toggle
1	1	+edge	0	0	Q_0	Qn_0	No Change

- Which inputs have higher precedence, J and K, or nPr and nClr?
- In order to use the J & K inputs what level should be applied to nPr & nClr?
- How can this flip-flop be operated as an S-R Latch?
- Read file **JK_FlipFlop.v** to help with your understanding of Verilog.
- Create file **JK_FlipFlop_top.v** that connects an instance of JK_FlipFlop to the DE2-115 board input/output devices as shown below. Connecting an input to ~KEY[n] means to drive the input with the complement of KEY[n], so pressing the push-button results in a 1 at the flip-flop input.



- Place the files in the Lab Directory Structure where indicated.
- Use Icarus Verilog or ModelSim to eliminate compile errors and warnings in the two Verilog files.

During Lab

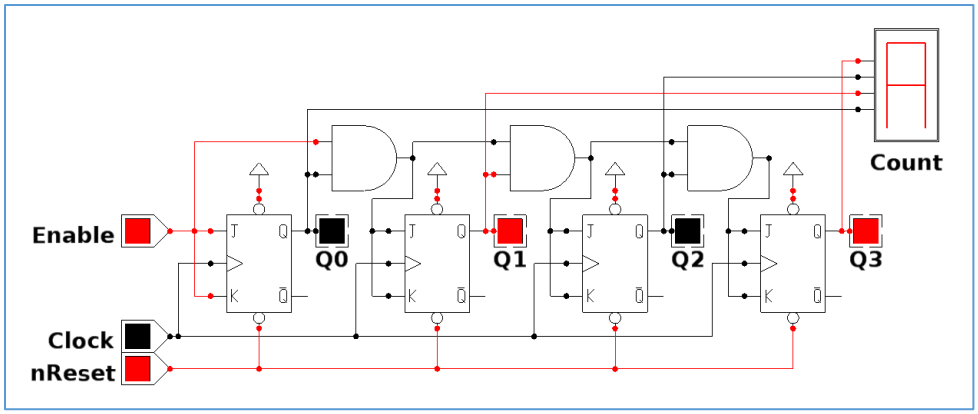
- Use Quartus and the DE2-115 board to build, debug and verify your pre-lab work. You may observe that pressing a push button once registers multiple presses. This is known as "switch bounce" and will be explored further in the future. Gently pressing and releasing the buttons helps work around the problem.
- Verify operation by testing all rows of the JK Flip-Flop function table.
- **When convinced your circuit is working properly, demonstrate it to your instructor.**

- Your report for this part should include the contents of JK_FlipFlop_top.v, a Quartus RTL Viewer image expanded to the gate level, discussion of any Quartus warnings, and anything else that will help your target reader.

Part 3: A Simple 4-bit Synchronous Counter

Pre-Lab

- The 4-bit binary counter shown below is built from JK Flip-Flops operating in toggle mode (J = K). Active-low input nReset forces the output to 0. Active-high input Enable causes counting to stop when low. When Enable = 1 and nReset = 1, positive edges at input Clock cause the counter to increment by 1, from hex 0 to F and then repeat. The flip-flop active low nPr inputs are all inactive (1 or high or Vcc).



- Create file **hex7seg.v** to convert a 4-bit input (hex 0 through F) to a 7-bit output that will display "0" through "F" on a DE2-115 board 7-segment display.
- Create file **counter4bit.v** to implement the 4-bit ripple counter shown above. It has inputs Clock, nReset, and Enable. Its output is 4-bit Count. Tips: This module has nothing to do with DE2-115 board switches and displays. It instantiates 4 instances of JK_FlipFlop and connects them as shown above.
- Create file **counter4bit_top.v** that instantiates counter4bit and hex7seg. Its inputs will be DE2-115 board pushbuttons KEY[2:0] (~Clock, Enable, nReset respectively). Its outputs will be DE2-115 board 7-segment display HEX0 and red LEDs LEDR[3:0]. For example, as shown in the diagram, when the counter is at binary 1010 (decimal 10) LEDR[3:0] will display 1010 and HEX0 will display "A". Pressing KEY[2] will increment the count to 1011 ("b").
- Complete provided file **counter4bit_tb.v** as a test bench for counter4bit.
- **Use Icarus Verilog and GTKWave, or ModelSim, to verify your counter4bit functions correctly.**

During Lab

- Use Quartus and the DE2-115 board to debug and verify your counter.
- **When convinced your circuit is working properly, demonstrate it to your instructor.**

- Your report for this part should include the contents of **counter4bit_tb.v**, **counter4bit_top.v**, **counter4bit.v**, **hex7seg.v**, s ModelSim text output, GTKWave or ModelSim waveforms (with an explanation of what is being demonstrated), a Quartus RTL Viewer image (whatever you think will be most helpful to your target reader), discussion of any Quartus warnings, and anything else that will help your target reader.