Lab 3: Latches, Flip-Flops and Registers

CSC 258, University of Toronto

January 28, 2014

About this Lab

Learning Objectives

1. To introduce you to latches, flip-flops, and registers.
2. To introduce the `always` construct in Verilog

Marking Scheme

As usual, all items in this document labeled **PRELAB TODO:** are to be completed before arriving to the lab, and are worth one mark out of the four for the labs. Items marked **INLAB TODO:** are to be completed in lab, with one mark for parts I to III, and one mark for parts IV to VI. Part VII is marked as *Advanced*, and it worth one mark.

1 Pre-lab Assignment

You are required to write the Verilog code for the Parts I to VII of the lab. Before the lab session, perform the prelab tasks specified in parts I to VI, and hand in your prepared Verilog code to the lab to be marked by the TAs.

1.1 Part I

Up until this week, all our labs have been with what we call *combinational* circuits. They all have logical formulae like $y = a + b$ – where the output value of the circuit can be determined from the input values alone.

With *sequential* circuits, the output depends not only on the input values, but also the previous state(s) of the circuit itself. It is similar to recurrence relations in mathematics – where formulae are expressed in forms such as $t_n = t_{n-1} + 1$ (where $t_0 = 0$). In this example, we take the previous value of t and use it to determine the next value. For example, $t_2 = t_1 + 1 = (t_0 + 1) + 1 = (0 + 1) + 1 = 2$.

Sequential circuits use the previous internal values of a circuit to store information, giving these circtuis the ability to have memory and maintain state information. A D latch is an example of a sequential circuit, which works like this:

- If `en`=0, the D latch's output (q) maintains its previous value (q_{prev})
- If `en`=1, the D latch's output (q) becomes the same as the input d

The D latch allows us to remember a value; it is a simple form of memory. One way of conceiving of it is shown below:

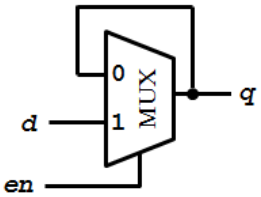


Figure 1: A latch, implemented with a multiplexer.

The truth table for this circuit is a bit different than the ones you've seen before, because now we have a variable, q_{prev}. It represents the q value that the D latch had at the previous instant.

en	d	q
0	0	q_{prev}
0	1	q_{prev}
1	0	0
1	1	1

PRELAB TODO: Answer the following questions:

1. Suppose we turned on the latch with en=1 and d=0. What would be the value of the latch's output q?

2. Now suppose we, after doing that step above, switched en=0. What value for q would the latch have?

3. And now, after that step, suppose we switched d=1. What value for q would the latch have?

4. Now after that step, suppose we switched en=1. What value for q would the latch have?

5. Lastly, after that, suppose we switched en=0. What value for q would the latch have?

Display your answer in a table in the following format when preparing your prelab report:

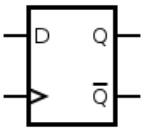
en	d	q_{prev}	q
1	0	X	
0	1		
...	...	...	

1.2 Part II

A flip-flop is another type of sequential circuit, and is very similar to a latch. It is different from a latch in when it takes in a new value. With the latch, we took in a new value whenever `en` was on – so if we left `en` on, and flipped d on and off, we would see q go on and off in response.

Flip-flops only take in a new value when `en` *becomes* on. Once `en` is on, nothing changes: we have to turn it off and then on again to load in a new value. This is why `en` is typically connected to a clock wave, so that we can regularly load in new values. Every time the clock wave becomes on, we load a new value; otherwise we hold onto the old one.

We represent D flip-flops with this symbol:



Like for the latch, d is our input, and q is the value we're remembering. The $>$ represents the clock input, replacing the `en` of the latch.

PRELAB TODO: What is the $\overline{Q}$ output? Look this up and write your answer in your pre-lab report. Make sure to cite your source[1].

Altera FPGAs include flip-flops that are available for implementing your circuits. We will show how to make use of these flip-flops in Part IV of this exercise. Before we do that though, we will show how storage elements can be created in an FPGA with gates, instead of using its dedicated flip-flops.

Figure 2 depicts a gated RS latch circuit. Two styles of Verilog code that can be used to describe this circuit are given in Figure 2a and 2b. Part a of the figure specifies the latch by instantiating logic gates, and part b uses logic expressions to create the same circuit. When this latch is implemented on an FPGA, the specification software will represent the resulting circuit as a 4-input lookup tables (LUTs), which is similar to a dictionary in Python. When you display the LUT that represents these circuits, only one lookup table is needed to represent the behaviour, as shown in Figure 3a.

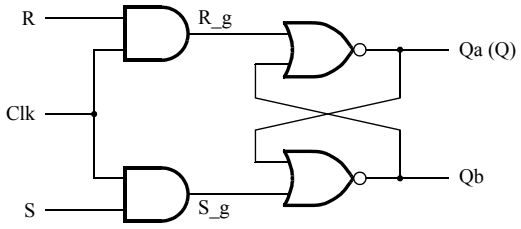


Figure 2: A gated RS latch circuit.

[1] If you are searching on the internet, it will help to look for a "D Flip-Flop"

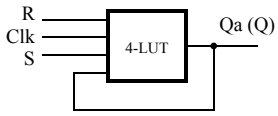
```
// A gated RS latch
module part1 (Clk, R, S, Q);
   input Clk, R, S;
   output Q;

   wire R_g, S_g, Qa, Qb /* synthesis keep */ ;

   and (R_g, R, Clk);
   and (S_g, S, Clk);
   nor (Qa, R_g, Qb);
   nor (Qb, S_g, Qa);

   assign Q = Qa;

endmodule
```

Figure 2*a*. Instantiating logic gates for the RS latch.

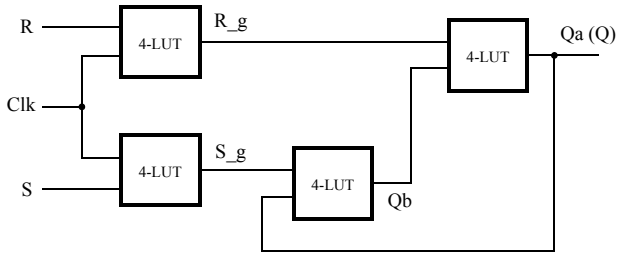
```
// A gated RS latch
module part1 (Clk, R, S, Q);
   input Clk, R, S;
   output Q;

   wire R_g, S_g, Qa, Qb /* synthesis keep */ ;

   assign R_g = R & Clk;
   assign S_g = S & Clk;
   assign Qa = ~(R_g | Qb);
   assign Qb = ~(S_g | Qa);

   assign Q = Qa;

endmodule
```

Figure 2*b*. Specifying the RS latch by using logic expressions.

Although the latch can be correctly realized using one 4-input LUT, this implementation does not allow us to observe its internal signals, such as R_g and S_g, because the LUT only preserves the input and output signals for the entire circuit. To preserve these internal signals in the implemented circuit, it is necessary to include a *compiler directive* in the code to instruct the complier on what signals to hold on to. In Figure 2*a* and 2*b* the directive **/* synthesis keep */** is included to instruct the Quartus II compiler to use separate logic elements that produce each of the signals R_g, S_g, Qa, and Qb. Compiling the code produces the circuit with four 4-LUTs depicted in Figure 3*b*.



(a) Using one 4-input lookup table for the RS latch.



(b) Using four 4-input lookup tables for the RS latch.

Figure 3: Design of the RS latch from Figure 2.

PRELAB TODO: Create a Quartus II project for the RS latch circuit as follows:

1. Create a new project for the RS latch. Select as the target chip the Cyclone II EP2C35F672C6, which is the FPGA chip on the Altera DE2 board.

2. Generate a Verilog file with the code in either part *a* or *b* of Figure 2 (both versions of the code should produce the same circuit) and include it in the project.

3. Compile the code. Use the Quartus II RTL Viewer tool to examine the gate-level circuit produced from the code, and use the Technology Viewer tool to verify that the latch is implemented as shown in Figure 3*b*.

4. Using either QSim or ModelSim, create waveforms for the R and S inputs and use the Quartus II Simulator to produce the corresponding waveforms for R_g, S_g, Qa, and Qb. Verify that the latch works as expected using both functional and timing simulation.

1.3 Part III

Figure 4 shows the circuit for a gated D latch.

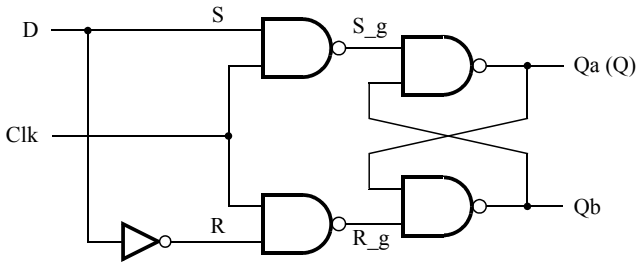


Figure 4: Circuit for a gated D latch.

This section will be performed with TTL chips and wires on the breadboard.
PRELAB TODO: Perform the following steps:

1. Decide what chips you'll need to recreate the circuit shown above (you should only need two).
2. For each of the gates in this circuit, label the pin numbers that your design will use.

The remainder of the work for this section will be done in the lab.

1.4 Part IV

Consider the circuit for a gated D latch, shown in Figure 4 above.
PRELAB TODO: Perform the following steps:

1. Create a new Quartus II project. Generate a Verilog file using the style of code in Figure 2*b* for the gated D latch. Use the /* synthesis keep */ directive to ensure that separate logic elements are used to implement the signals R, S_g, R_g, Qa, and Qb.
2. Select as the target chip the Cyclone II EP2C35F672C6 and compile the code. Use the Technology Viewer tool to examine the implemented circuit.
3. Verify that the latch works properly for all input conditions by using functional simulation. Examine the timing characteristics of the circuit by using timing simulation.
4. Create a new Quartus II project which will be used for implementation of the gated D latch on the DE2 board. This project should consist of a top-level module that contains the appropriate input and output ports (pins) for the DE2 board. Instantiate your latch in this top-level module. Use switch SW_0 to drive the D input of the latch, and use SW_1 as the *Clk* input. Connect the Q output to $LEDR_0$.

1.5 Part V

Figure 5 shows the circuit for a master-slave D flip-flop.

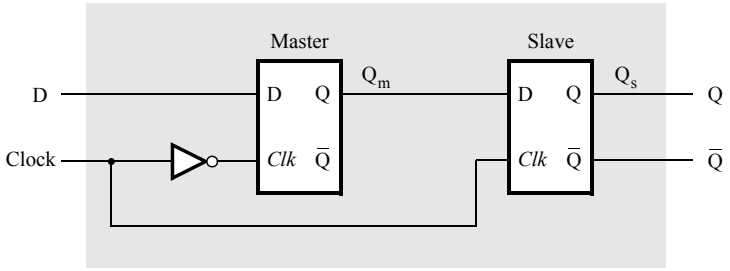


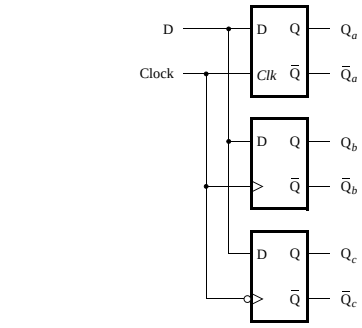
Figure 5: Circuit for a master-slave D flip-flop.

PRELAB TODO: Perform the following:

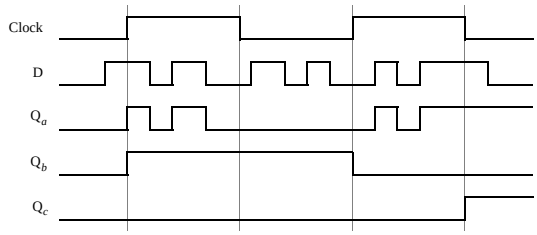
1. Create a new Quartus II project. Generate a Verilog file that instantiates two copies of your gated D latch module from Part II to implement the master-slave flip-flop.
2. Include in your project the appropriate input and output ports for the Altera DE2 board. Use switch SW_0 to drive the D input of the flip-flop, and use SW_1 as the *Clock* input. Connect the Q output to $LEDR_0$.
3. Compile your project.
4. Use the Technology Viewer to examine the D flip-flop circuit, and use simulation to verify its correct operation. Take a snapshot with a clock signal and different values of D, print it out and attach it to your prelab submission.

1.6 Part VI

Figure 6 shows a circuit with three different storage elements: a gated D latch, a positive-edge triggered D flip-flop, and a negative-edge triggered D flip-flop.



(a) Circuit



(b) Timing diagram

Figure 6: Circuit and waveforms for Part VI.

PRELAB TODO: Implement and simulate this circuit using Quartus II software as follows:

1. Create a new Quartus II project.
2. Write a Verilog file that instantiates the three storage elements. For this part you should no longer use the /* synthesis keep */ directive from Parts I to III. Figure 7 gives a behavioral style of Verilog code that specifies the gated D latch in Figure 4. This latch can be implemented in one 4-input lookup table. Use a similar style of code to specify the flip-flops in Figure 5.
3. Compile your code and use the Technology Viewer to examine the implemented circuit. Verify that the latch uses one lookup table and that the flip-flops are implemented using the flip-flops provided in the target FPGA.
4. Simulate this in either QSim or ModelSim. Specify input signals for *D* and *Clock* as indicated in Figure 6. Use functional simulation to obtain the three output signals. Observe the different behavior of the three storage elements, and print out a snapshot of the behaviour.

```
module D_latch  (D, Clk, Q);
   input D, Clk;
   output reg Q;

   always @ (D, Clk)
      if (Clk)
         Q = D;
endmodule
```

Figure 7. A behavioral style of Verilog code that specifies a gated D latch.

1.7 Part VII – Advanced

PRELAB TODO: Write code for the following circuit. We wish to display the hexadecimal value of a 16-bit number A on the four 7-segment displays, $HEX7-4$. We also wish to display the hex value of a 16-bit number B on the four 7-segment displays, $HEX3-0$. The values of A and B will be provided by the user, through inputs to the circuit in the form of switches SW_{15-0}. This is to be done by first setting the 16 switches to the value of A, sending a clock pulse to the circuit, and then setting the switches to the value of B; therefore, the value of A must be stored in the circuit after the clock pulse.

1. Create a new Quartus II project which will be used to implement the desired circuit on the Altera DE2 board.
2. Write a Verilog file that provides the necessary functionality. Use KEY_0 as an active-low asynchronous reset, and use KEY_1 as a clock input.
3. Include the Verilog file in your project and compile the circuit.
4. Assign the pins on the FPGA to connect to the switches and 7-segment displays, as indicated in the User Manual for the DE2 board.

2 In lab work

For each part of the pre-lab, compile and download your code to the FPGA. Test the functionality of your design by toggling the switches and observing the displays.

1. **INLAB TODO:** Show your answers to your TA for Part I
2. **INLAB TODO:** Show your board to your TA for Part II
3. **INLAB TODO:** For Part III, implement your latch design on the protoboard, using the chips specified in your design. Use switches on the switch board as your D and Clk input and the lights as output. Once you've got it working, demonstrate it to one of the teaching assistants.
4. **INLAB TODO:** Show your board to your TA for Part IV
5. **INLAB TODO:** Show your board to your TA for Part V
6. **INLAB TODO:** Show your board to your TA for Part VI
7. **INLAB TODO:** (Advanced) Show your board to your TA for Part VII

Clean up your station once you are done the lab.