Justin VanWinkle
EEE 3342C
Experiment #8
28 July 2015

Objective

This experiment is intended to build and investigate the operation of asynchronous, clocked, and master slave SR flip-flops using the Xilinx ISE design tools. The designs will be created with XILINX ISE and simulated with the Xilinx ISIM tool and tested on a BASYS 2 FPGA development board.

Diagrams

The following diagrams give a good low level view of the devices being implemented in this project:

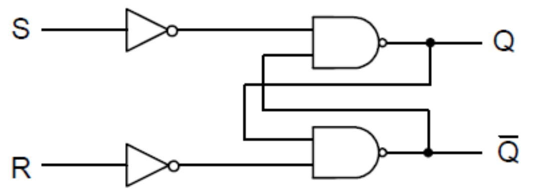


Figure 8-1: Asynchronous SR Flip-Flop

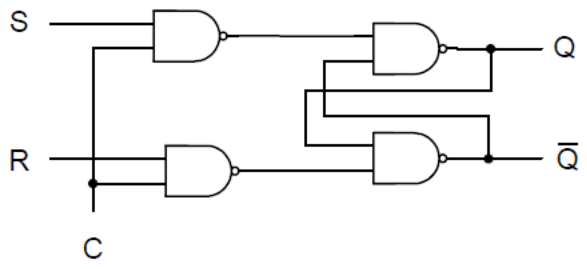


Figure 8-4: Clocked SR Flip-Flop

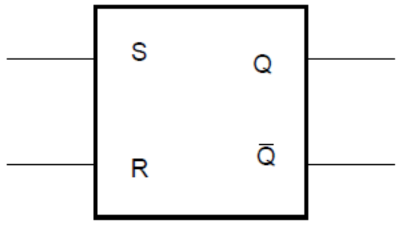


Figure 8-2: Asynchronous SR Flip-Flop Block Diagram

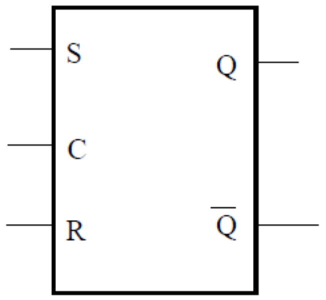


Figure 8-5: Clocked SR Flip-Flop Block Diagram

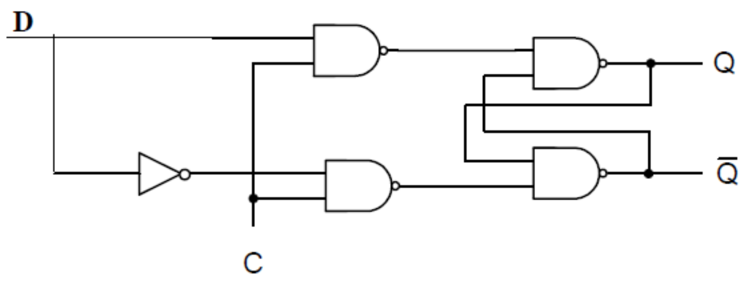


Figure 8-6: Clocked D Flip-Flop

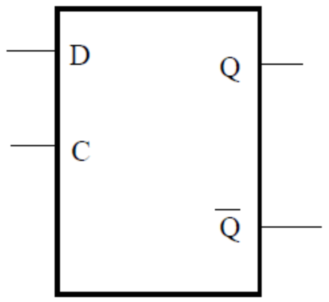


Figure 8-7: Clocked D Flip-Flop Block Diagram

As can be seen, each of the block diagrams can be easily represented by a minimum set of logic gates.

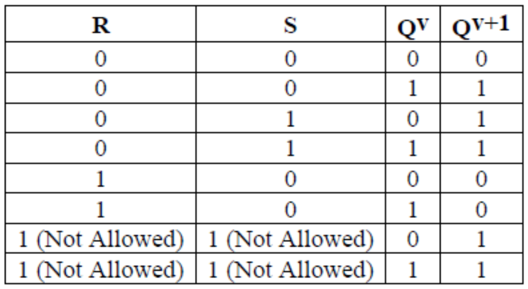
Equipment

The equipment used for this experiment included a computer with the Xilinx ISE design suite and Digilent ExPort installed as well as a Digilent BAYSYS 2 development board.

Procedure

Part 1

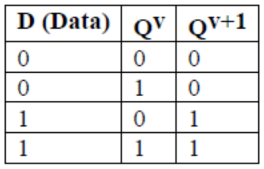
The first step in this experiment should be analyzing and understanding what our expected output should be based on any given set of inputs. This will give us a better understanding of how to build the schematic of the circuit. The below tables are a comprehensive view of the logic possibilities in each of the circuits at any given moment, matched with their current state and next state:



R	S	Q^v	Q^{v+1}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1 (Not Allowed)	1 (Not Allowed)	0	1
1 (Not Allowed)	1 (Not Allowed)	1	1

The clocked SR flip-flop performs identical to the chart on the left, however, it will only change states upon detection of a clock signal.

Figure 8-3: Asynchronous SR Flip-Flop State Transition Table



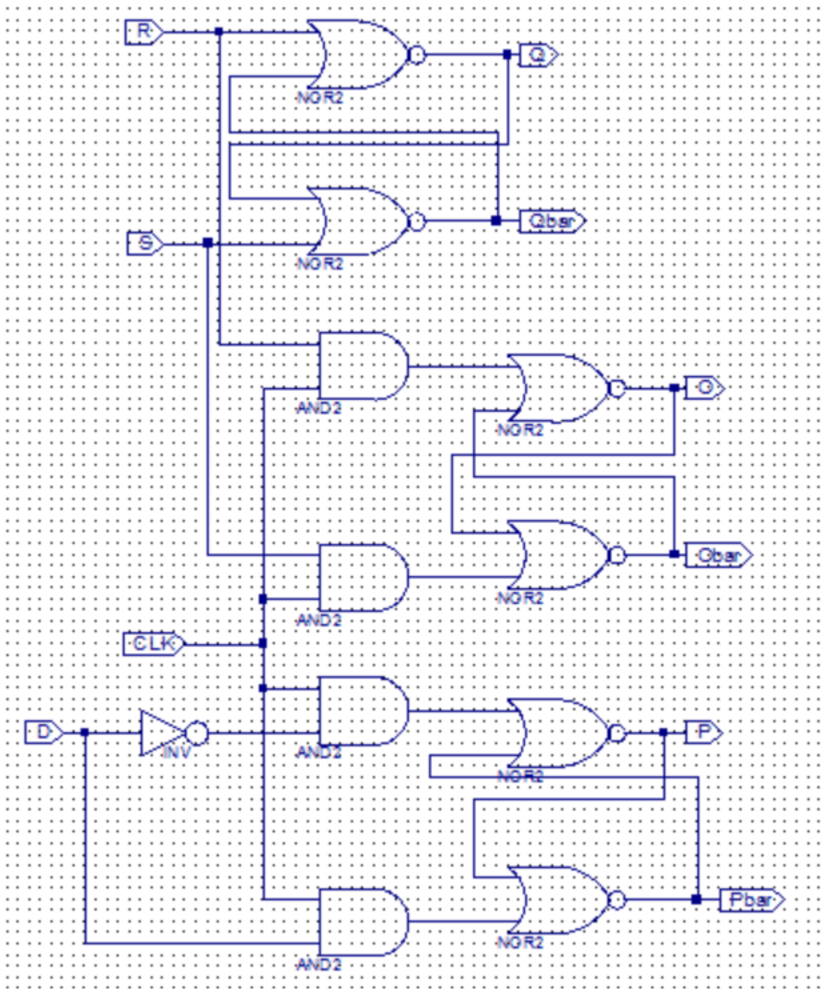
D (Data)	Q^v	Q^{v+1}
0	0	0
0	1	0
1	0	1
1	1	1

Figure 8-9: Clocked D Flip-Flop State Transition Table

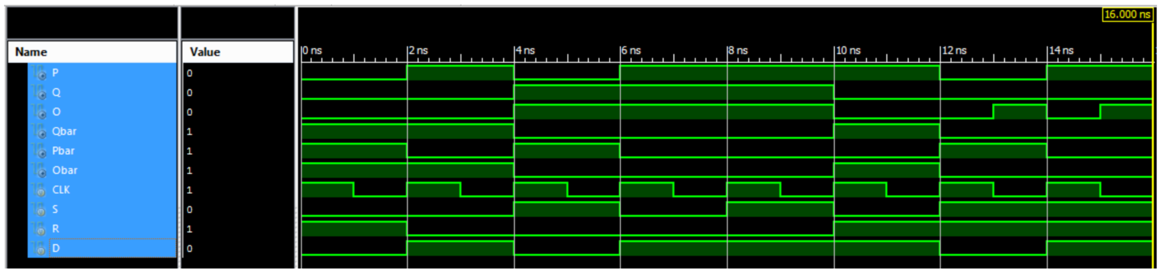
The above logic diagram shows the expected next states of the flip-flops of this experiment based on all possible combinations of inputs given a current state.

Next, the following schematic was created using the data from the logic tables:

Justin VanWinkle
28 June 2015
Experiment #8

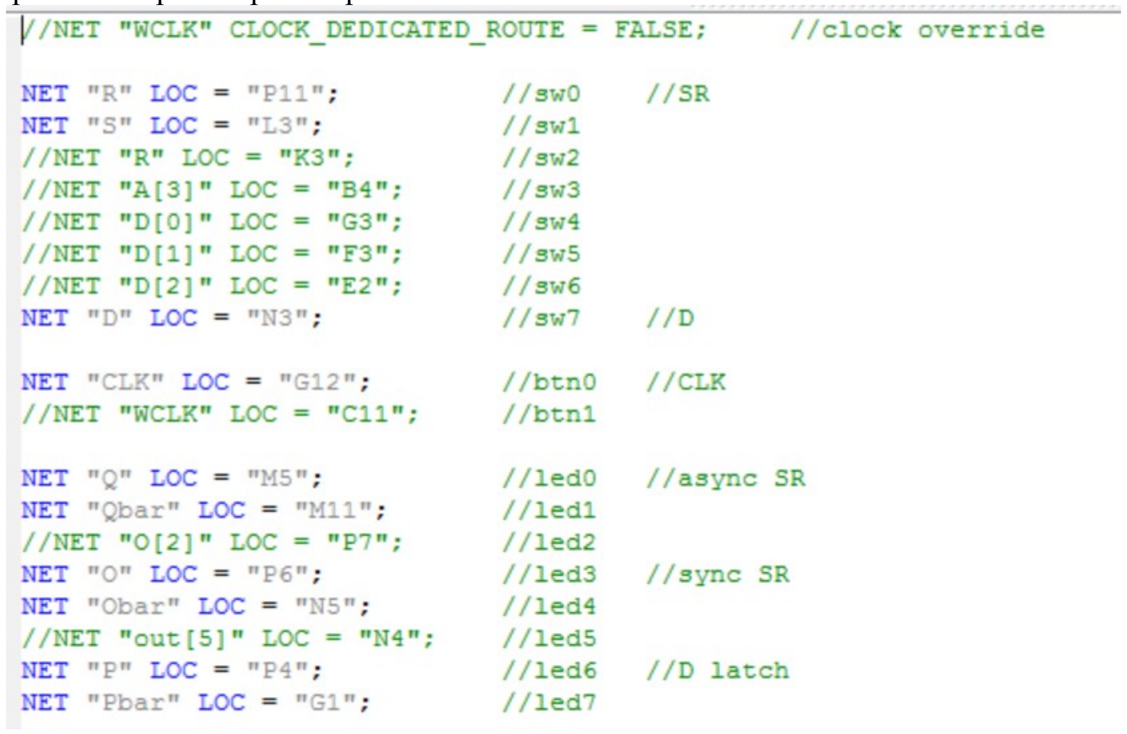


After completion of the above schematic capture, the operator then synthesized and implemented the design. Using the isim interface, the operator next simulated and tested the schematic design for its accuracy based on all possible combinations of input and function selects. The result of this simulation is as follows:



Name	Value
P	0
Q	0
O	0
Qbar	1
Pbar	1
Obar	1
CLK	1
S	0
R	1
D	0

With a proper simulation result, the operator next created constraints that mapped each input and output to specific pins of the CP132. The constraints were:



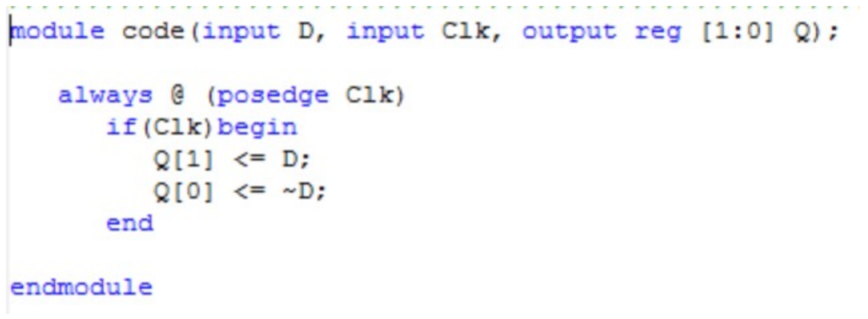
```
//NET "WCLK" CLOCK_DEDICATED_ROUTE = FALSE;     //clock override

NET "R" LOC = "P11";            //sw0    //SR
NET "S" LOC = "L3";             //sw1
//NET "R" LOC = "K3";           //sw2
//NET "A[3]" LOC = "B4";        //sw3
//NET "D[0]" LOC = "G3";        //sw4
//NET "D[1]" LOC = "F3";        //sw5
//NET "D[2]" LOC = "E2";        //sw6
NET "D" LOC = "N3";             //sw7    //D

NET "CLK" LOC = "G12";          //btn0   //CLK
//NET "WCLK" LOC = "C11";       //btn1

NET "Q" LOC = "M5";             //led0   //async SR
NET "Qbar" LOC = "M11";         //led1
//NET "O[2]" LOC = "P7";        //led2
NET "O" LOC = "P6";             //led3   //sync SR
NET "Obar" LOC = "N5";          //led4
//NET "out[5]" LOC = "N4";      //led5
NET "P" LOC = "P4";             //led6   //D latch
NET "Pbar" LOC = "G1";          //led7
```

Now, the operator was ready to implement the design a final time and create the .bit file used to program the BASYS 2 board. To transfer the file to the BASYS 2 board, Digilent Adept was used. Once in Adept, with the BASYS 2 board powered on, the operator selected the initialize chain button to establish communication between the computer and the BASYS 2 board. Next, the operator used the browse button next to the "FPGA" text to select the newly created *.bit file for programming the BASYS 2 board. Finally, the "program chain" button was used to push the program to the BASYS 2 board and the operator could test the program on the board for proper output.

Part 2

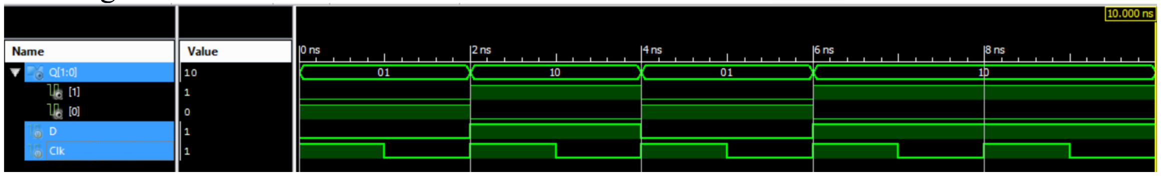
For the second part of the experiment, the operator used verilog to implement clocked D flip-flop. This very slightly changed the process up to the point of creating a simulation, but beyond that, the process is identical.

When creating a VERILOG module, the user will select inputs and outputs based on the need of the problem and will also select a naming convention for each. The code for flip-flop is as follows:

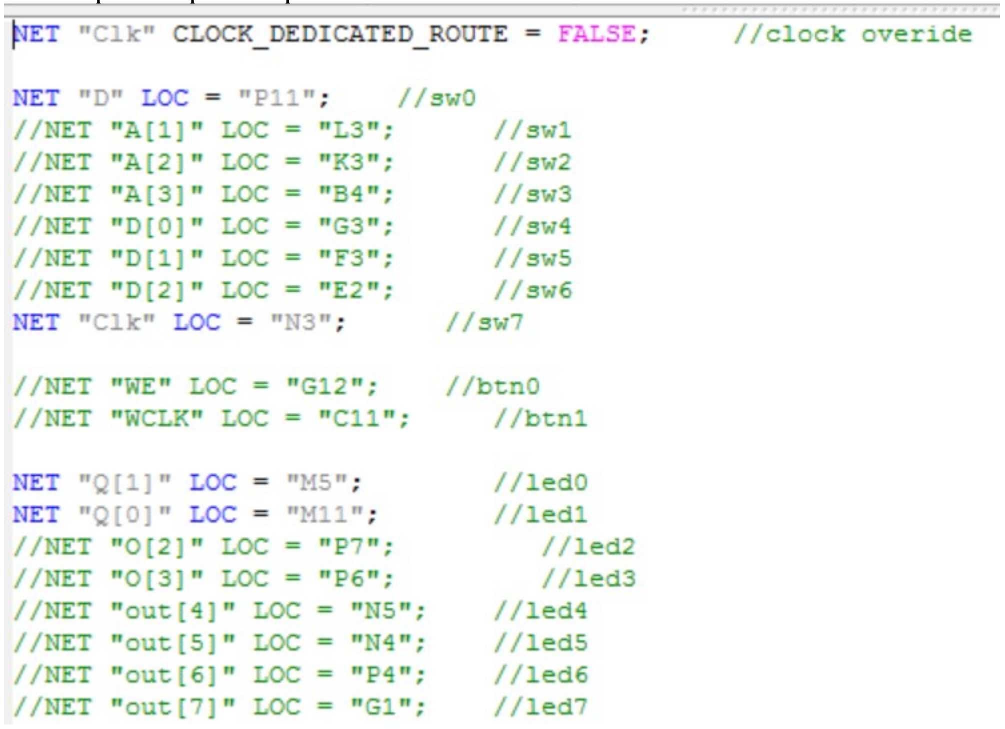


```
module code(input D, input Clk, output reg [1:0] Q);

   always @ (posedge Clk)
      if(Clk)begin
         Q[1] <= D;
         Q[0] <= ~D;
      end

endmodule
```

After completion of each of the above verilog code, the operator then synthesized and implemented the design. Next, the operator used the newly written code to simulate the intended circuit for its accuracy based on all possible combinations of inputs. The resulting simulation is as follows:



With a proper simulation result, the operator next created constraints that mapped each input and output to specific pins of the CP132. The constraints were:



```
NET "Clk" CLOCK_DEDICATED_ROUTE = FALSE;      //clock overide

NET "D" LOC = "P11";     //sw0
//NET "A[1]" LOC = "L3";       //sw1
//NET "A[2]" LOC = "K3";       //sw2
//NET "A[3]" LOC = "B4";       //sw3
//NET "D[0]" LOC = "G3";       //sw4
//NET "D[1]" LOC = "F3";       //sw5
//NET "D[2]" LOC = "E2";       //sw6
NET "Clk" LOC = "N3";       //sw7

//NET "WE" LOC = "G12";     //btn0
//NET "WCLK" LOC = "C11";      //btn1

NET "Q[1]" LOC = "M5";         //led0
NET "Q[0]" LOC = "M11";        //led1
//NET "O[2]" LOC = "P7";          //led2
//NET "O[3]" LOC = "P6";          //led3
//NET "out[4]" LOC = "N5";     //led4
//NET "out[5]" LOC = "N4";     //led5
//NET "out[6]" LOC = "P4";     //led6
//NET "out[7]" LOC = "G1";     //led7
```

Now, the operator was ready to implement the design a final time and create the .bit file used to program the BASYS 2 board. To transfer the file to the BASYS 2 board, Digilent Adept was used. Once in Adept, with the BASYS 2 board powered on, the operator selected the initialize chain button to establish communication between the computer and the BASYS 2 board. Next, the operator used the browse button next to the “FPGA” text to select the newly created *.bit file for programming the BASYS 2 baord. Finally, the “program chain” button was used to push the program to the BASYS 2 board and the operator could test the program on the board for proper output.

Expected Results of Circuit

The expected results of this circuit can be seen in the state tables previously created. During the simulation, the developer created assumptions that checked for proper response based on the state charts at the beginning of this report. A message was shown to report any unexpected outputs. This serves s a fail-safe way of testing.

Justin VanWinkle
28 June 2015
Experiment #8

Design Specification Plan

The results of this experiment draw a sound conclusion that the objective has been met by the execution of the procedure. This is given by the fact that simulations and test results match verbatim to that of the expected results based on the calculations performed beforehand. The methodology chosen for this experiment followed a calculation-first route such that the operator would know what to expect. This also provided simplicity for the operator when designing the schematic as well as the VERILOG module. Given that a successful attempt was made to design, simulate, and implement a function using a multiplexer, which was the objective of this experiment, this experiment has met all of its requirements.

Test Plan

This experiment should be tested and verified by the following means. The tester should obtain a BASYS 2 development board with this program loaded on it. Once the tester has set up the board and powered it on, he should test for output based on the provided state tables. Additionally, the tester can run a simulation at any time and watch for console output informing him of wrong outputs.

Results

Please see the section entitled procedures for simulation results.
The outcome of this experiment accurately matched that of the calculations performed and presented in the expected results section of this report. This means that a parallel adder was successfully designed and implemented, because when it was given two numbers, the parallel adder was able to produce and return the sum of the given values.

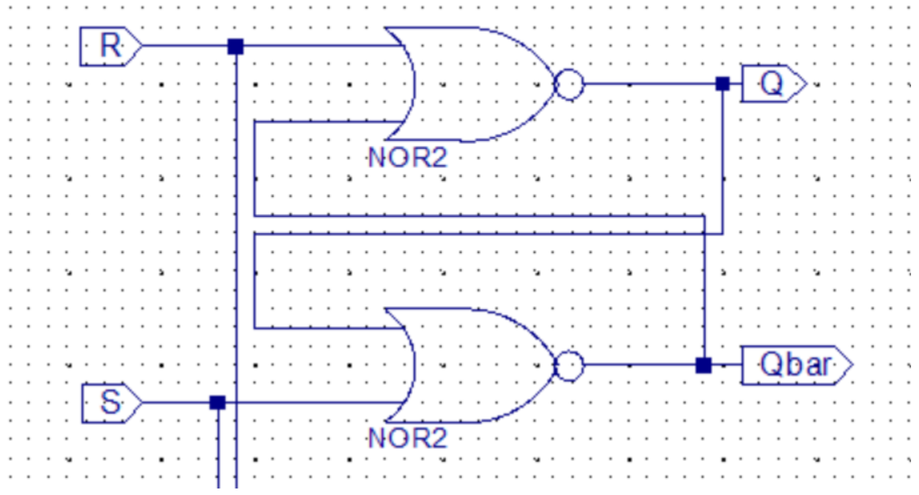
Conclusion

In conclusion, this flip-flops clearly serve a large role in logic design. While small and simple to design, these devices serve in a much larger capacity.

Questions

1. An async flip-flop doesn't have to wait for a clock signal before changing to its next state while a sync flip-flop can provide more predictable behavior in terms of timing. A D flip-flop provides a very straight forward interface where the SR is slightly more complex and can serve in a slightly more complex capacity.

2. I used NOR gates to complete the lab…

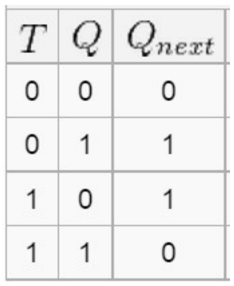


Justin VanWinkle
28 June 2015
Experiment #8

3. A level-triggered flip-flop will have its CE line active as long as the clock is at a designated level whereas an edge-triggered flip-flop will only have its CE line active at the moment that the designated edge occurs.

4. Bit-binary registers, shift registers, counters, frequency division, etc.

5. If the toggle line is active, then the output will change. Otherwise, the output will remain the same.



T	Q	Q_{next}
0	0	0
0	1	1
1	0	1
1	1	0

6. always @ (posedge clk) will be a positive edge trigger
   always @ (negedge clk) will be a negative edge trigger