EEL3705L, Digital Logic Design Lab, Spring 2011

**Pre-Lab #5**

For

Lab Assignment #5:

Design a Hex Adding Machine Using Mixed VHDL & Schematics

**Document Description:** This pre-laboratory report summarizes the work done in advance to the fifth lab of the EEL3705L class for the spring 2011 semester at the Florida State University. This report describes my primary design as well as the testing procedures to be done to verify that the design performs the problem given for the lab.

* **Introduction**

In this lab, students are required to make a sequential adding machine for unsigned bits. All of the number that must be displayed need to be in hexadecimal form. There will be a one-byte accumulator value that will be incremented or decremented by a one-byte input value. Students must design their own custom flip-flops and latches for this experiment. For the binary-to-hexadecimal display outputs, students must write out their own VHDL code to implement this, and the rest of the circuit needs to be designed in Quartus like previous labs.

* **Requirements**

1. The circuit must increment or decrement the stored value in the accumulator by the input value, with both values being one byte in size
2. The eight bit register that will store the accumulator value must be made from scratch, with no VHDL or megafunctions used. The basic SR Latch in this register must also be made from NOR gates cross-coupled together.
3. To translate the binary outputs to a display that will show their values in hexadecimal, VHDL code must be the only thing used to design it.

* **Theory and Design**

Display of calculated value hexadecimal number

Binary to HEX display hexadecimal representation

Selector for addition/subtraction

Eight-bit adder/subtractor

Eight-bit binary input number

Display of accumulator hexadecimal number

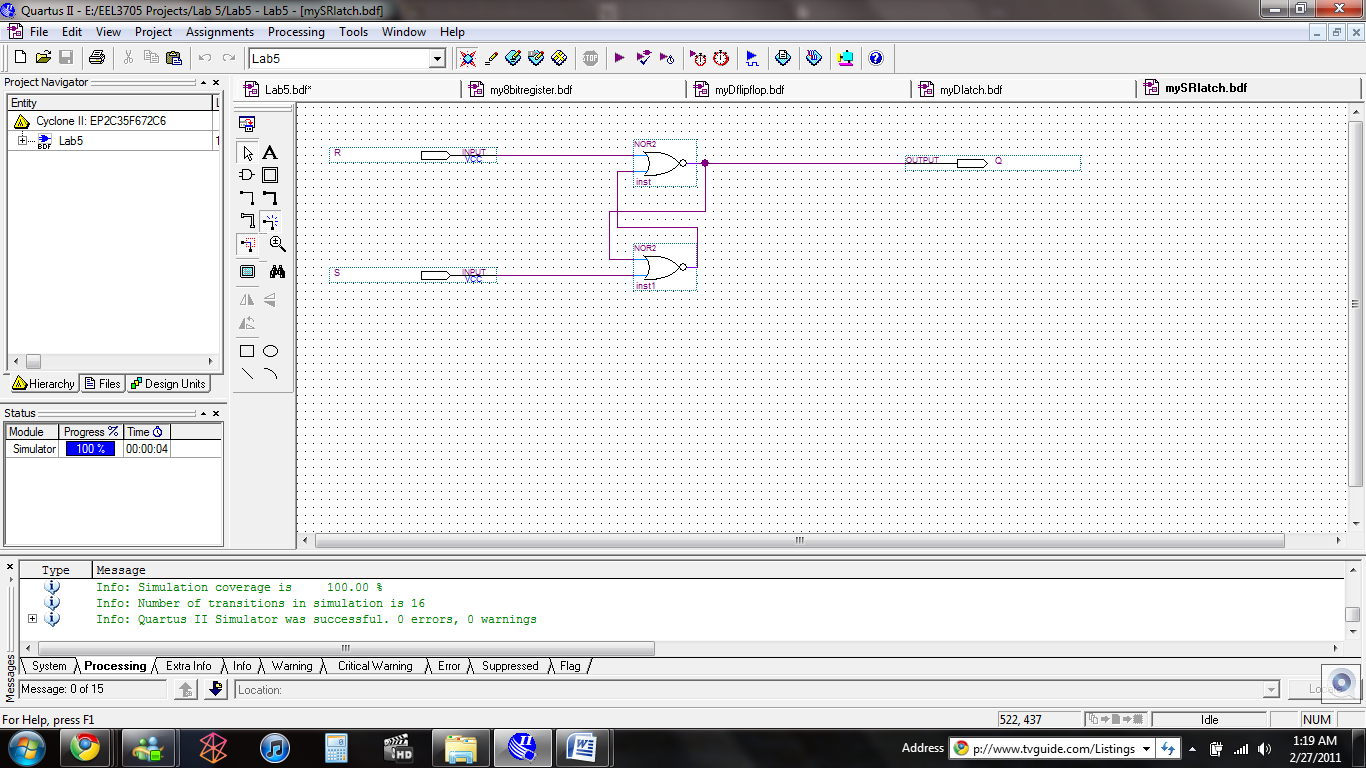
Binary to HEX display hexadecimal representation

Eight-bit registry

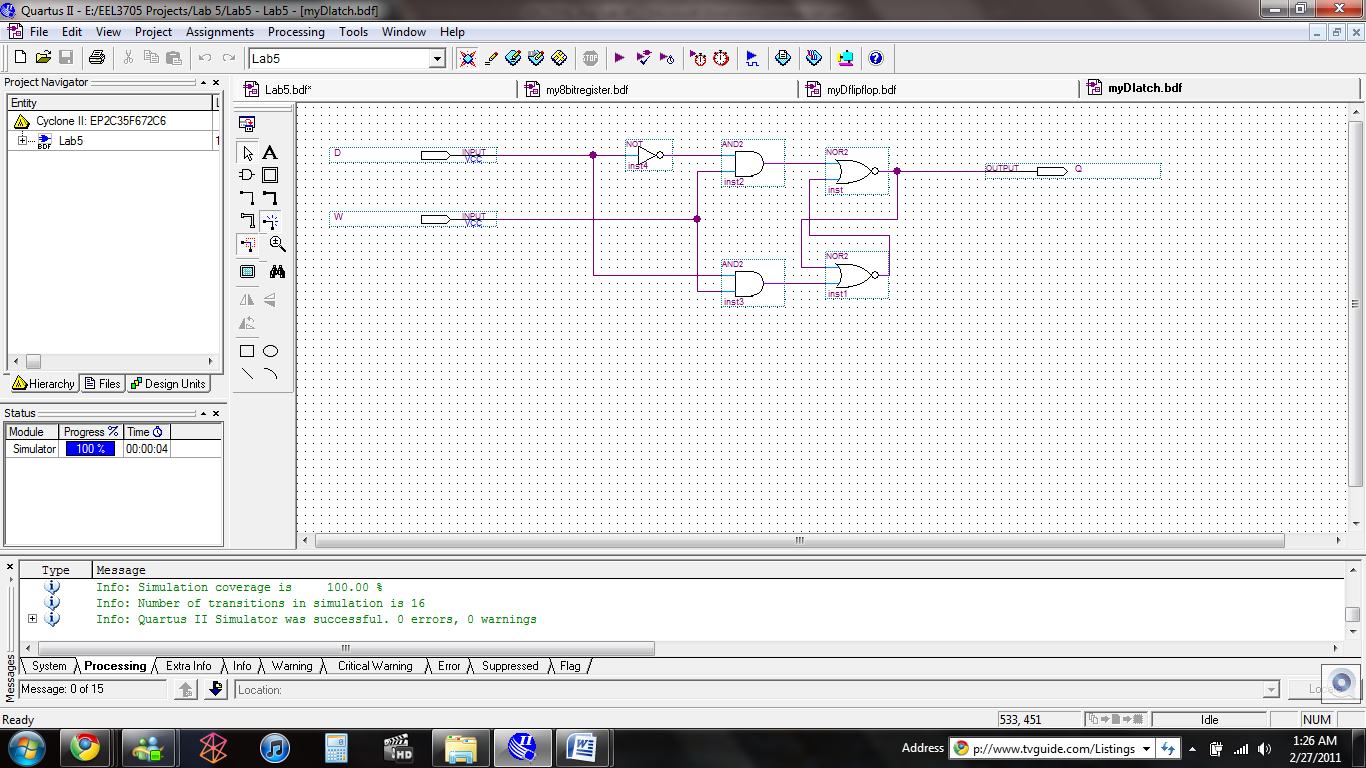
Display of input hexadecimal number

Binary to HEX display hexadecimal representation

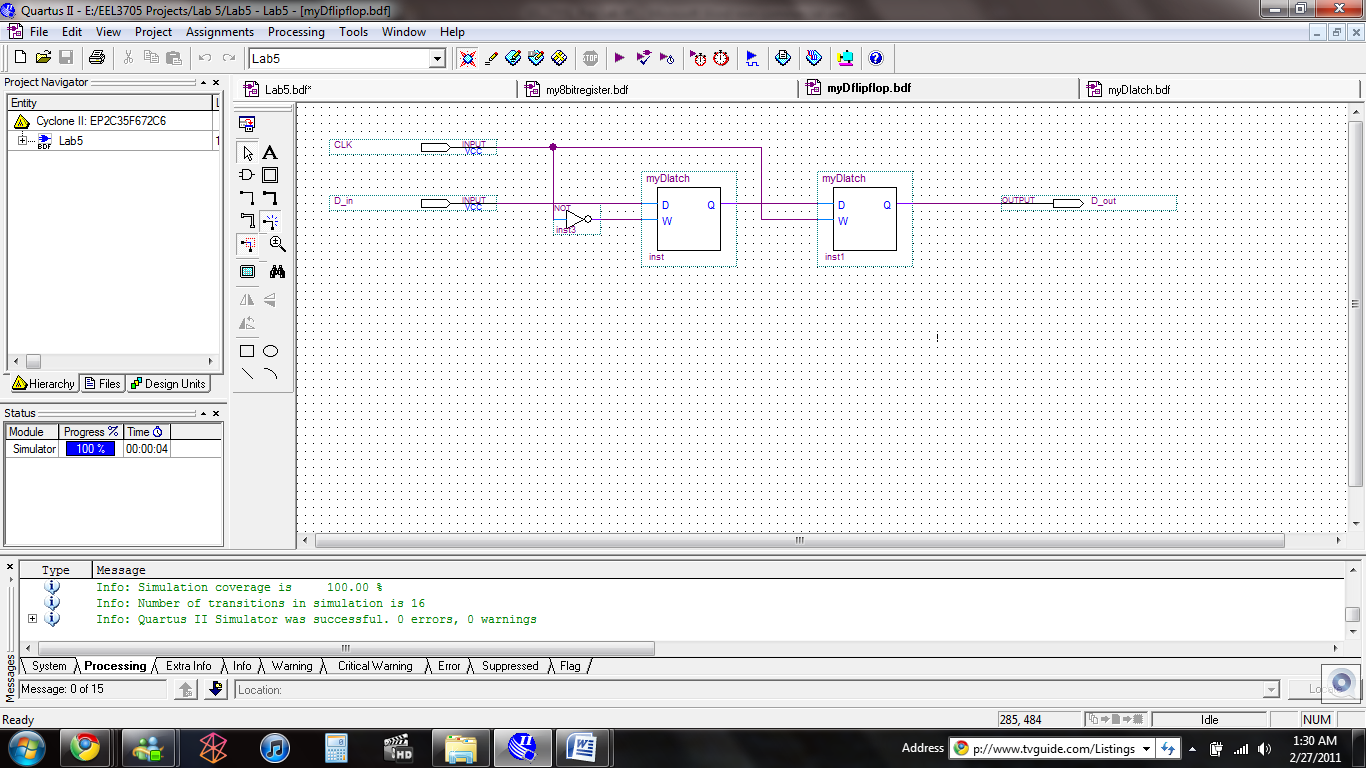
Because we had to use cross-coupled NOR gates for the basic SR Latch of our design, it was much easier to have the set and reset inputs function as active-high. The Quartus design for the SR latch is shown in the figure below.



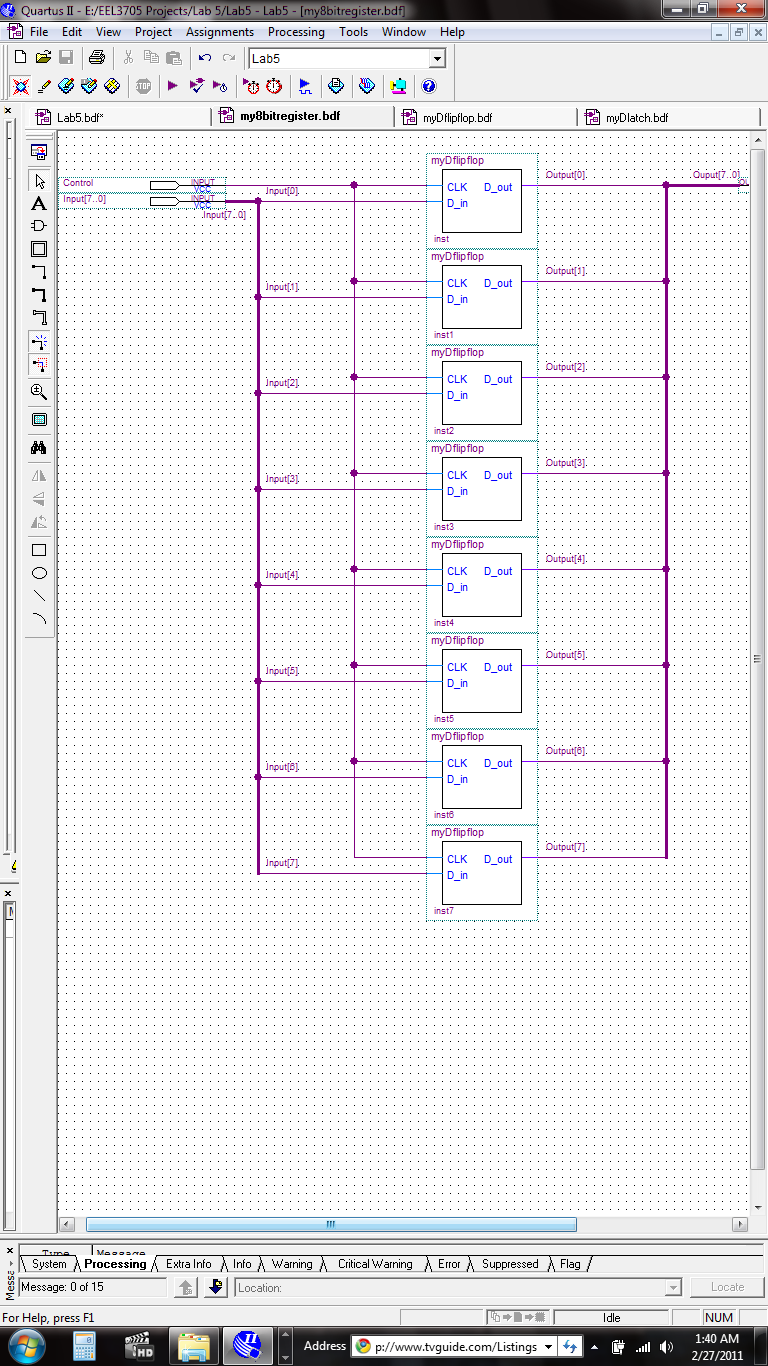
Now that we have the SR latch designed, we can expand on its design to create a D Latch, which is shown below as represented in Quartus.



This circuit was made into a block named “myDlatch” to be used in a larger circuit. With two of these D latches, we can now make a D flip-flop. The Quartus representation for the one that we used is shown in the figure below.

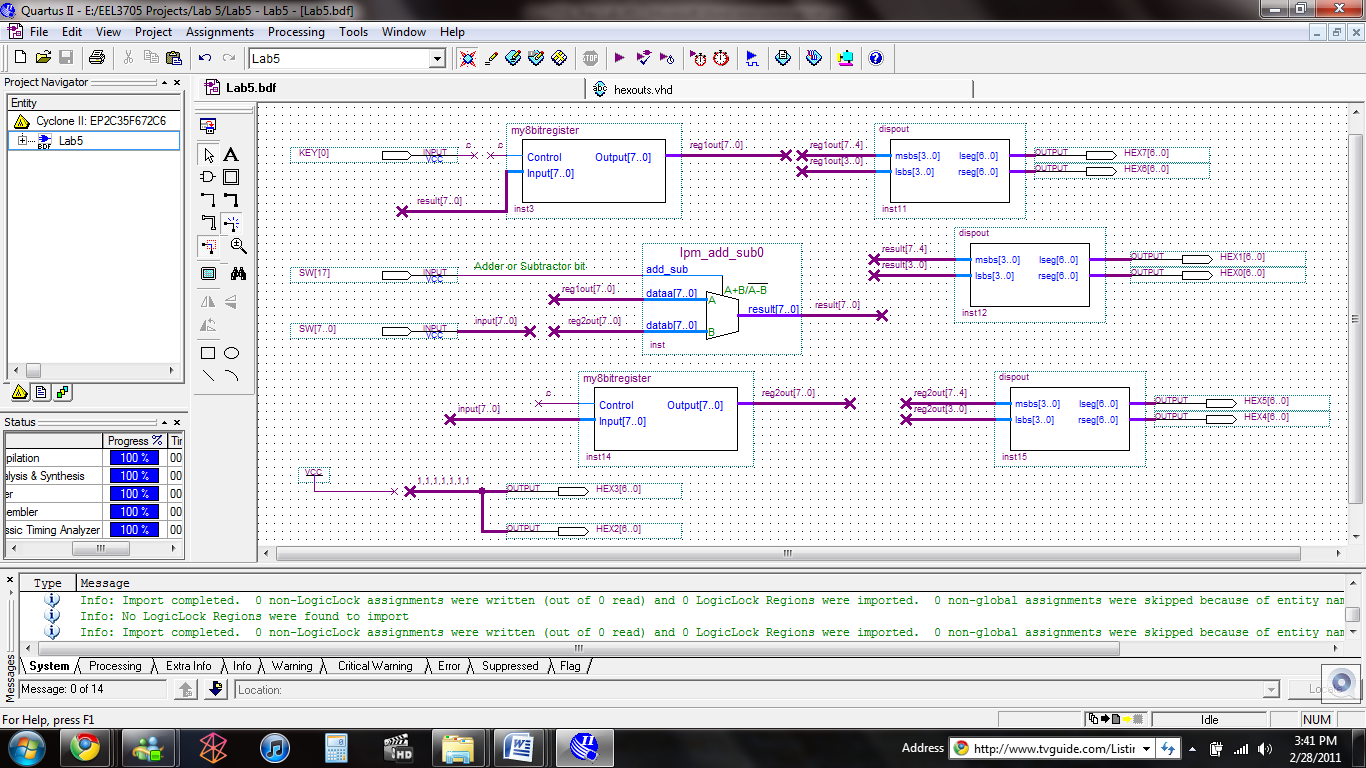


From this circuit, we can see that the D flip-flop designed is a rising edge triggered flip-flop because the first D latch will only be written to when there is a low signal from the clock. Once the clock signal goes high, the second D latch will be written to. Therefore, the data will only be output when the input clock goes from low to high, and therefore is a rising edge signal. A block was made for this circuit named “myDflipflop”, so that it can be easily implemented in a larger design. To make an eight-bit register, we simply have eight of the D flip-flops in parallel, one for each bit of the accumulator value, each with the same input clock. The eight-bit register represented in Quartus is shown below.

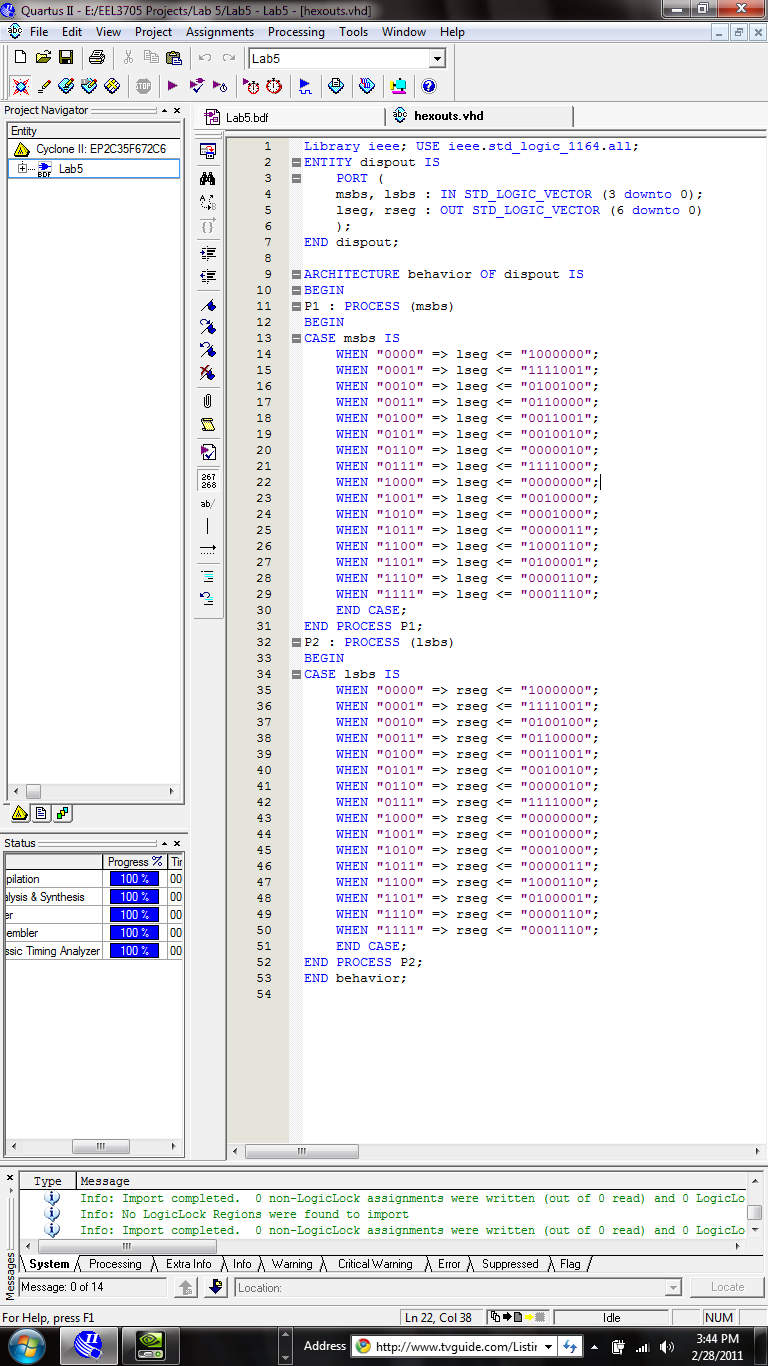


This too was made into a block, which was named “my8bitregister”.

Because in this lab we are not using a clock to control the intervals in time to increment or decrement, but an “enter” key, we decided to use a pushbutton on the DE2 board to serve this function, KEY[0], to be exact. Because the pushbuttons on the DE2 are debounced and output a low signal when depressed, the releasing of these buttons will produce a rising edge signal, and therefore it is this action that will determine when the value of the result is to be changed. In order to easily switch between incrementing or decrementing, we used a single slider switch, SW[17] on the board, to serve as the indicator to do whichever function. In our circuit, a high signal represents incrementing, while a low signal represents decrementing. As for the eight bit input binary number, we used the same method to input binary as in other labs, which is using the appropriate amount of the DE2 slider switches. In our circuit, we used switches SW[7] through SW[0] for the input number, with SW[7] representing the most significant bit, and SW[0] representing the least significant bit. The most top-level design of our circuit represented in Quartus is shown below.



To increment or decrement, we just use the Quartus megafunction lpm\_add\_sub. The blocks labeled “dispout” represent the circuits that were made via VHDL code that take an eight-bit binary input number and translate that to a hexadecimal representation on the DE2 board’s HEX displays. The entire VHDL code for these block is shown below.

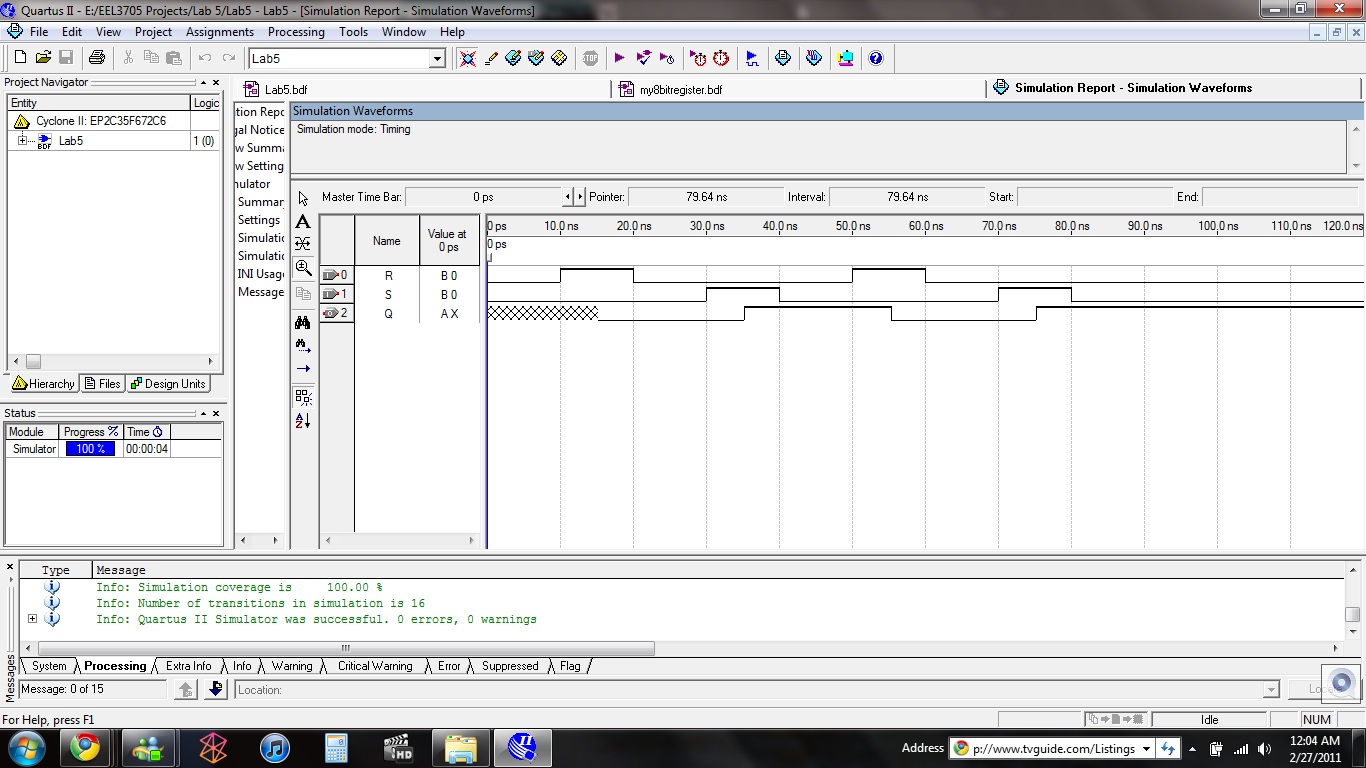


In our circuit, we decided that we would use HEX[0] and HEX[1] to display the hexadecimal number that is the incremented or decremented calculated vale. We also used HEX[7] and HEX[6] to show the hexadecimal representation of the accumulator value. Finally, we used HEX[5] and HEX[4] to display the input binary number in its hexadecimal form.

To make sure that the output calculated value would only change at one instant, when the pushbutton was let go of, we made another registry that would give the adder both of the values it needs to add or subtract dependant on the rising edge signal of the pushbutton.

* **Simulation Testing**

For simulations, we were only recommended to simulate the NOR gate SR latch that we had designed, since we were not given how this would be designed. The Quartus simulation results are shown in the figure below.



We can see from the simulation that the circuit behaves as it should, with the output Q becoming a low signal when the reset bit, R, goes high, and Q goes high when the set bit, S, goes high.

* **Design of Prototype Testing Experiments**

I think that the best approach to testing out this circuit is starting very small. Incrementing by small values, like 1, should be used first. After a verification that that works, higher values should then be incremented to the number. The accumulator and input values should be added by hand before the circuit, to make sure that it behaves properly. Because the output of the adder/subtractor is not signed, negative and overflow values will most likely be off from the real result.

* **Answers to Questions**

1. An appropriate, and easy, way for the user to input an eight-bit binary number into the circuit would be the slider switches on the DE2 board. We will use the first eight for our application
2. Our “enter” button will be the first pushbutton on the DE2 board. Because we were required to make a rising edge D flip-flop, releasing the pushbutton will create a rising edge and increment or decrement the accumulator number.
3. Because we can use the lpm\_add\_sub megafunction which has an option to have a 1 or 0 inputted to determine addition or subtraction, we decided to use switch SW[17] to distinguish between these two arithmetic operations.
4. Because the values that need to be displayed are all unsigned eight bit binary numbers in their hexadecimal notation, we will need two hexadecimal digits for each number. So, we can use two of the DE2 board’s HEX displays for these numbers.
5. A good device on the DE2 board to display the raw binary of the numbers would be the green or red, or perhaps both, LEDs on the board.
6. In order to make sure that the calculated value only change when the pushbutton was released, we needed to use another registry that would hold the input numbers.
7. We know that for an eight-bit binary number, the first four bits solely control the least significant hexadecimal bit equivalent, and the last four solely control the most significant hexadecimal bit equivalent. So, we can write code to describe what the outputs will be on the least significant HEX display based on the four least significant binary bits 0-F in hexadecimal. We can then use the same patter for the most significant HEX display based on the four most significant binary bits.
8. The first picture in this document shows the schematic of the SR latch that we will be using, and the simulation results for it given above verify that it will do the function it needs to do.
9. The second picture in this document shows our basic SR latch with two extra AND gates and an inverter to make it into a D latch.
10. The third picture in this document shows the D flip-flop that we will use, as implemented by two D latches and an inverter. As explained above, we see that it is rising-edge triggered.
11. The fourth picture in this document then shows eight of the D flip-flops designed arranged in a parallel manner to make out eight-bit registry.

* **Conclusion**

This lab really did introduce some very new topic to me. Even though I had used temporary memory devices like JK flip-flops before, I have never built a registry to store values of a circuit. I found this problem to be quite confusing initially, but I was able to figure out what was needed to be done after taking the whole process step-by-step. Designing latches and flip=flops from scratch with just basic logic gates was also a completely new experience for me.

VHDL was another thing that was a new experience. I had never used, or even heard of this type of programming language before this class. I found that it was very different from any other programming languages that I had learned like C++, Visual Basic, or HTML. But, like any programming language, it had its own stylistic patterns and syntax that become easier to identify and implement as it is used and studied more and more.