|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 🔑 **Essential** 🔑 | | | |  | 🏆 **Enhanced** 🏆 | | | |
|  |  |  |  |  |  |  |  |  |
| \_\_\_\_\_ | \_\_\_\_\_ | \_\_\_\_\_ | \_\_\_\_\_ |  | \_\_\_\_\_ | \_\_\_\_\_ | \_\_\_\_\_ | \_\_\_\_\_ |

**Score: \_\_\_\_\_**

**Lab04 – Sequential Circuits and Static Memory (SRAM)**

COMP256 – Computing Abstractions

Dickinson College

Spring 2022

Prof. Grant Braught

**Name(s):**

**Introduction:**

In the first three labs we worked our way up the hardware abstraction hierarchy from electrical signals, to 0’s and 1’s to transistors to logic gates and circuits and ultimately to a small “programmable” machine. That machine could be programmed using two-bit instructions to perform one of four different operations on its inputs. What is still missing is a sense of how the data and the program instruction are stored. For example, we know that in the Knob and Switch computer that we have been working with, the instructions and data are stored in the main memory cells and the registers. So, the natural question at this point is… how are those built from electronic components? In this lab you will learn about the behavior of memory cells and how one can be constructed using the logic gates with which we are already familiar.

**An Abstract Memory Cell:**

A storage box provides an analogy that is often helpful in thinking about memory cells as shown below. The value of E controls whether the box is open or closed. The box will be open when the input E = 1 (left) and will be closed when the input E = 0 (right). When the box is open (E=1) the value of the input D is placed into the box. But when the box is closed (E=0), the value of D is blocked from going into the box. The output Q will always show us what is currently stored in the box.



The following sequence of images illustrates how this box stores one bit of information:



The above sequence goes as follows:

(i) E=1 so the box is open. The input D=0, so a 0 is stored into the box and appears on the output Q.

(ii) E=0 so the box is now closed. Because the box is closed the value of the output Q is “frozen” or we can say that the value 0 is stored in the box.

(iii) E=0 so the box is still closed. So, even though the input D has now become a 1, the value of Q remains 0, because the box is closed and thus D cannot be put into it.

(iv) E=1 so the box is reopened. The input D=1, so a 1 is now stored into the box and also appears on the output Q.

(v) E=0 so once again the box is closed. The value of Q remains a 1 because that’s what we put in the box.

(vi) E=0 so the box is still closed. The value of Q remains a 1 reflecting the value that was stored in the box even though the input D at this point has changed to a 0.

The above sequence also illustrates that we will need a fundamentally different type of circuit to implement a memory cell**. With the combinational logic circuits we have been using the inputs uniquely determine the output. That is, if we put in the same combination of inputs, we will always get the same output.** That relationship is exactly what is expressed by a truth table. Each row of the truth table corresponds to one combination of the inputs and that row has a specific output. The circuit will always generate the given output for the given combination of inputs.

🔑 1. Now, consider the sequence of images above again.

a. Treating D and E as inputs and Q as the output, identify a pair of images (give their Roman numerals) that show that a memory cell cannot be constructed using a combinational logic circuit. Briefly explain why this pair shows that the memory cell cannot be built using a combinational logic circuit. (Hint: Read the paragraphs above and use what it says about sequential circuits to identify something that they cannot do in the images.).

b. Identify a second pair of images (give their Roman numerals) that also shows that the memory cell cannot be built using a combinational logic circuit.

**Sequential Circuits:**

In this section you will learn about what are called *sequential circuits*. They provide us with one way to create a memory cell. Ultimately, you will build a circuit that implements a 1-bit memory cell that behaves like the box in our analogy above.

The unique aspect of a sequential circuit that allows us to build a memory cell is that the outputs of the circuit *“feedback”* and become additional inputs to the circuit. For example, consider the circuit shown below:

A close up of a logo

Description automatically generated 

Notice in this circuit how the values of P and Q both “feedback” as inputs to the NOR gates. This creates a cycle, where the output Q must be defined in terms of itself. While that may seem problematic, we’ll see that with well-constructed sequential circuits it is not. And in fact, if we think about it, it makes sense that this should be a necessary condition for building a memory cell. If the output Q is supposed to remain the same regardless of the input D (i.e. when E=0), then somehow the circuit must know (or remember) the value of Q. Having Q feed back into the circuit is what makes this possible.

**The SR-Latch:**

Let’s now work on understanding the behavior of the above circuit, which is known as an *SR-Latch*. Note that an SR-Latch will not quite behave like the 1-bit memory cell we are looking for, but once we understand it, we will use it as a building block for the circuit that does.

To understand how this circuit works we will need to *“chase values around the circuit.”* For example, consider the situation where S=1, R=0 and Q=0 as shown below:



In this situation we know that the inputs to the top NOR gate will be 1 0 (S Q) so the output P will be P=0. This change means that the input to bottom NOR gate will be 0 0 (P R) which makes its output Q change from Q=0 to Q=1. This change then has a ripple effect changing the inputs to the top NOR gate to 1 1 (S Q). With these inputs the output of the top NOR gate remains P=0. At this point, all of the inputs and outputs have *“settled down”* (i.e. further “chasing” will not change anything).

One good way to “chase the values” is to draw out the circuit on some scrap paper. Then write the starting values next to S, R and Q. Then “chase the values” around the circuit, crossing out old values and filling in new ones until they settle down. The result of “chasing the values” for S=1, R=0, Q=0, as described above, is illustrated below:



🔑 2. As just described, the result of “chasing the values S=1, R=0, Q=0” results in a new output of Q=1. In the truth table below, this “settled down” or next value of Q is labeled Q’. The example with inputs S=1, R=0, and Q=1 described above is shown in yellow highlight in the table. Once you are sure you understand the highlighted line, complete the rest of the table below by “chasing” the specified input values around the circuit until things “settle down”.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  | **S** | **R** | **Q** | **Q’** |  |
|  | 0 | 1 | 0 |  |  |
|  | 0 | 1 | 1 |  |  |
|  | 1 | 0 | 0 | 1 |  |
|  | 1 | 0 | 1 |  |  |
|  |  |  |  |  |  |

🔑 3. Looking at the table above we can see why this circuit is called an SR-Latch. **When S=1 the value of Q is *Set* to 1 and when R=1 the value of Q is Reset to 0.** If your table does not show this behavior revisit questions #2 and resolve any issues before going on.

🔑 4. As mentioned, this circuit is known as an SR-Latch. We now understand that the S and R part of the name come from Set and the Reset. But why Latch? The term *Latch* is used to describe a circuit that exhibits memory (i.e. that a 0 or 1 can be “*latched into*” or saved by the circuit). Complete the truth table below by “chasing” the given input patterns around the circuit.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  | **S** | **R** | **Q** | **Q’** |  |
|  | 0 | 0 | 0 |  |  |
|  | 0 | 0 | 1 |  |  |
|  |  |  |  |  |  |

🔑 5. Looking at the truth table in #4 you should see that when S=0 and R=0 the value of Q’ will be the same as Q. That is, Q did not change when you “chased” the values. If your table does not show this behavior revisit questions #4 and resolve any issues.

**One way to interpret the table in question #4 is to think of it as showing that Q *remembers* what it had been before S and R were changed to 0 0.** For example, imagine that we make S=0, R=1 to Reset Q=0. If we now change to S=0, R=0 then Q will *remember* that it had been Reset to 0. Similarly imagine we make S=1, R=0 to S(et) Q=1. If we now change S=0, R=0 then Q will *remember* that it had been Set to 1. Summarizing, if we make S=0, R=0 then Q remembers if it has most recently been Set to 1 or Reset to 0. Note again that this behavior cannot be accomplished with a combinational logic circuit because we are seeing that for the same inputs (S=0, R=0) there can be two different outputs (Q can be either 0 or 1).

🔑 6. Okay, that all may seem a little strange, but let’s build it and see that it does actually work. Create a new circuit in TINKERCAD named Lab04-SRLatch and setup a full-size breadboard as shown below. Notice that I have added labels to the slide switches for the inputs and the LED for the output. These will correspond to the inputs and outputs of the SR latch. **Be sure to put the 74HC02 (NOR) chip to the right edge of the board and the switches to the left. This will leave room for things that you’ll be adding to this circuit later.**



🔑 7. Wire up the 74HC02 chip to create an SR latch with the inputs S and R provided by the slide switches and the output Q displayed by the LED (off = 0, on = 1**). The pin-outs for all of the chips are again included at the end of the lab for reference.**

🔑 8. Experiment with your circuit to complete the following truth table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | **S** | **R** | **Q** |  |
|  | 0 | 1 |  |  |
|  | 1 | 0 |  |  |
|  |  |  |  |  |

🔑 9. The table above you should agree with what you predicted in question #3. That is, **S=1 *Sets* the value of Q to 1 and R=1 *Resets* the value of Q to 0**. If your circuit does not exhibit this behavior revisit questions #6-#8 and resolve any issues.

🔑 10. Now let’s check the latching feature of the SR-latch as well. Use your circuit to complete each row in the truth table. Use S or R to set the value of Q needed for the row. Then set S=0 and R=0 to find the value of Q’ (the next value of Q). If what you find here does not agree with what you predicted in question #4, revisit your circuit to be sure it is wired correctly.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  | **S** | **R** | **Q** | **Q’** |  |
|  | 0 | 0 | 0 |  |  |
|  | 0 | 0 | 1 |  |  |
|  |  |  |  |  |  |

**The D-Latch:**

You may have noticed that thus far we have looked at every input pattern for our SR-Latch except those when S=1 and R=1. This should make some sense, because it is unclear what it would mean to both Set and Reset the value of Q at the same time! For this reason, the SR-Latch is often augmented with some additional circuitry to ensure that S and R cannot both be 1 at the same time. This configuration is shown below and is called a *D-Latch*. Notice that the D-Latch contains an SR-Latch as a part of its circuit (highlighted in yellow).

A picture containing clock

Description automatically generated

Let’s explore the behavior of this D-Latch.

🔑 11. One way to understand the behavior of a D-Latch is to consider how the inputs D and E set the values of S and R and then use our abstract understanding of the SR-Latch to understand the D-Latch. To facilitate that, complete the following truth table that shows the values of S and R for each pattern of the inputs D and E.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  | **D** | **E** | **S** | **R** |  |
|  | 0 | 0 |  |  |  |
|  | 0 | 1 |  |  |  |
|  | 1 | 0 |  |  |  |
|  | 1 | 1 |  |  |  |
|  |  |  |  |  |  |

🔑 12. The above table shows all possible inputs D and E to this circuit. Your table should show that the situation where the SR-Latch is being both Set (S=1) and Reset (R=1) at the same time never occurs. If it does, revisit your work for question #11 to resolve any issues.

🔑 13. Now, consider just the rows from the table in question #11 where E=1. Use the table from question #11 and your understanding of how the SR-Latch is Set and Reset to complete the following table showing the output (Q) of the D-Latch when E=1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  | **D** | **E** | **S** | **R** | **Q** |  |
|  | 0 | 1 |  |  |  |  |
|  | 1 | 1 |  |  |  |  |
|  |  |  |  |  |  |  |

🔑 14. From the table in question #13 you should see that when E=1, the next value of Q will be whatever D is. For this reason, the E input is called the ***Enable*** input. It enables the input D to be stored. If your table does not show this behavior revisit questions #11-13 and resolve any issues.

🔑 15. Compare the behavior of the D-Latch expressed by the truth table in question #13 to the images in the box analogy at the start of the lab. Which images (give their Roman numerals) confirm that the D-Latch behaves like the desired memory circuit when E=1?

🔑 16. Now let’s consider the case where E=0. The table below shows the rows from the table in question #11 where E=0, where each row has been duplicated to account for the case where Q=0 and the case where Q=1. Use the table from question #11 and your understanding of how the SR-Latch is Set and Reset to complete this table showing the next output (Q’) of the D-Latch when E=0.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  | **D** | **E** | **S** | **R** | **Q** | **Q'** |  |
|  | 0 | 0 |  |  | 0 |  |  |
|  | 0 | 0 |  |  | 1 |  |  |
|  | 1 | 0 |  |  | 0 |  |  |
|  | 1 | 0 |  |  | 1 |  |  |
|  |  |  |  |  |  |  |  |

🔑 17. Compare the behavior of the D-Latch observed in question #16 to the images in the box analogy at the start of the lab. Briefly explain how you can tell that the D-Latch behaves like the desired memory circuit when E=0.

🔑 18. Make a copy of your Lab04-SRLatch in TINKERCAD and name it Lab04-DLatch. Add to this circuit the necessary components to create a D-Latch. The two slider switches should now be used as the D and E inputs to the D-Latch.

🏆 19. Use your circuit to complete the table below. For each row, use D and E to set the value of Q to the proper value. Then set D and E to their values for the row to find the next value of Q (i.e. Q’). Ignore the final column without a header for now.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  | **D** | **E** | **Q** | **Q'** |  |  |
|  | 0 | 0 | 0 |  |  |  |
|  | 0 | 0 | 1 |  |  |  |
|  | 0 | 1 | 0 |  |  |  |
|  | 0 | 1 | 1 |  |  |  |
|  | 1 | 0 | 0 |  |  |  |
|  | 1 | 0 | 1 |  |  |  |
|  | 1 | 1 | 0 |  |  |  |
|  | 1 | 1 | 1 |  |  |  |
|  |  |  |  |  |  |  |

🏆 20. By this point, it should hopefully be clear that the D-Latch is acting like our 1-bit memory. But just to reinforce this, fill in the final column in the table above with the letter (A-F) of the picture from the start of the lab that corresponds to that row. Note that the two cells that are shaded will not get a letter\*\*.

\*\* So why the two shaded cells? All of the states to which you assigned a letter are called *stable states*. That is, for the values of D, E and Q in that row Q’ is the same as Q. So, nothing is changing in the circuit – it is stable and will say that way until an input is changed. For the two shaded cells however Q’ is different than Q. So, things are changing in the circuit (i.e. the value “chasing” that you did in the SR-Latch part of the circuit would not be finished). These states where the output is changing are called *transient states*. If you look closely at these transient states though, you will see that once Q changes to be Q’ the circuit will change into one of the stable states.

**Making a Register:**

How does this D-Latch relate to a real computer? Well, consider the registers in the Knob & Switch computer. Each register has to store 16 bits. To do that the hardware could simply contain 16 D-Latches, one for each bit. To store a value into a register, the Enable input for all 16 of those D-Latches would be set to 1 and the D input of each one would be set to the bit to be stored. Then the Enable input would be set back to 0 so save (i.e. latch in) the value. This is very close to what happens in the registers in a real computer.

🏆 21. TINKERCAD includes a 74HC75 chip which contains four D-Latches (they call it a 4-bit latch). Use your favorite search engine to find a pin-out diagram for the 74HC75 chip and paste it below along with the URL where you found it.

🏆 22. Create a new TINKERCAD circuit named LAB04-74HC75. Connect two slide switches (one for D and one for E), an LED (for Q) and one of the D-Latches on the 74HC75 chip. Experiment with your circuit to confirm that the D-Latch on the 74HC75 chip behaves just like the D-Latch that you built using logic gates earlier. Note: Your pin outs for the 74HC75 chip may use G instead of E in the diagram. It will also contain a output, this will always be the logical opposite of Q, you can ignore and just use Q.

*Your TINKERCAD circuit is your answer for this question.*

🏆 🏆 23. Create a new TINKERCAD circuit named LAB04-4Bit-Register. Build a circuit that uses all four of the D-Latches on the 74HC75 as a 3-bit register. Your circuit should have:

* 4 slide switches to specify the 4 bits to be saved in the register.
* 4 LEDs to show the contents of the register.
* A slide switch for E that causes a new value to be saved into the register.

*Your TINKERCAD circuit is your answer for this question.*

🏆 24. Use your circuit from #23 to store the value 1110 into the register using unsigned binary. Paste a screen shot of your circuit with this values saved in the register as your answer to this question.

**Chip Pin-Out Reference:**

A picture containing clock

Description automatically generated

Optional: To help me improve and scope these activities for future semesters please consider providing the following feedback.

a. Approximately how much time did you spend on this Lab (include the 2-hour lab period in your total)?

b. Please comment on any particular challenges you faced in completing this Lab.

**Acknowledgements:**

Box images from Open Clip Art and used under a CC0 license.

