Digital Design Lab 6

Introduction to Sequential Logic

The Islamic University of Gaza
Engineering Faculty
Department of Computer Engineering

Author: Mohammed Nafiz ALMadhoun

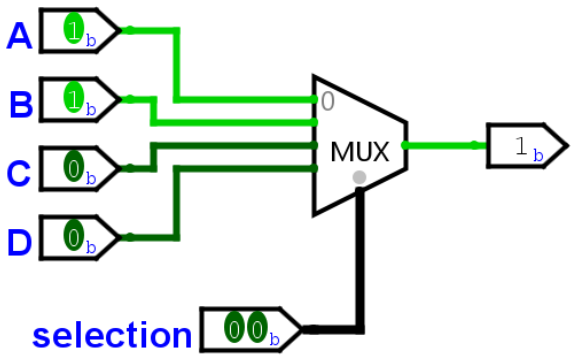
2021/11/15

Introduction

In this lab, we are going to talk about one combinational circuit, then we talk about sequential logic, and the meaning of memory elements in our designs, which is one of the most important concepts in logic design which will allow us to make our design remember.

Multiplexer

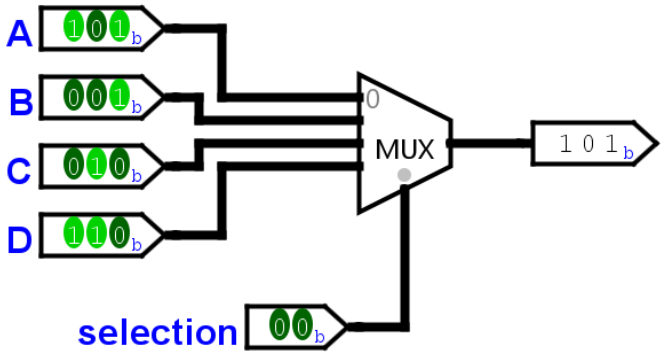
One of the most important combinational designs is the multiplexer, which will allow you to choose a single data line from many data lines, building a multiplexer is very similar to the encoder, the encoder will output a number that represents the selected line, the multiplexer will pass the data in that line to our output.



Multiplexer

So this design, we've selected A to be in our output by putting the selection bits to `00`.

The design will be more useful if we used more than one multiplexer together to multiplex a data bus



3-Bits data lines in Mutliplexer

Introduction to Sequential Logic

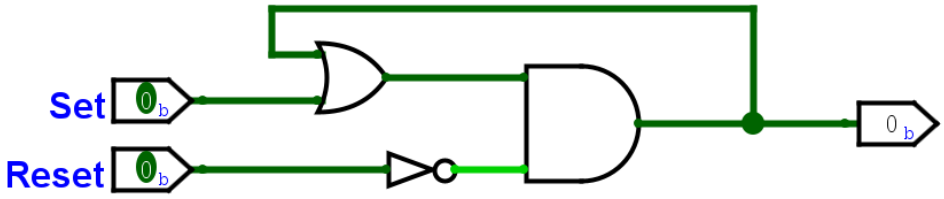
As you might notice when you use your calculator or the coffee vending machine, the machine will remember your choice even after your release the buttons, this is what a memory element will enable you to do.

A memory element is something that remembers its previous state, and we can create a simple memory element using gates of course.

SR Latch

The SR latch will enable us to set or reset it using two inputs called `S`, and `R`, and here is the truth table for it:

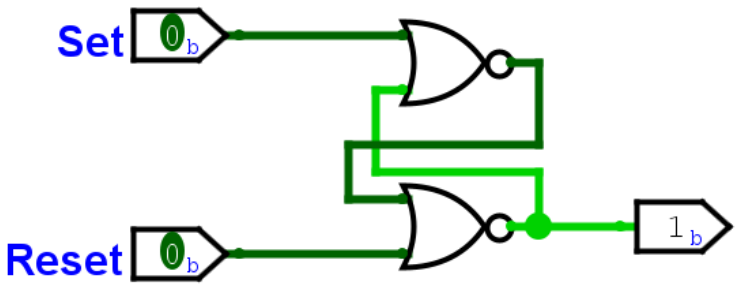
S	R	Q
0	0	Previous State
0	1	0
1	0	1
1	1	Invalid



Basic SR Latch

This design will output `1` if `S=1` and `R=0`, and it will save the value `1` even if `S=0` and `R=0`, and it will output `0` if `S=0` and `R=1`, and it will output the value `0` even after we set `S=0` and `R=0`.

This is our basic **SR Latch** that enables us to store a value in it, note that no one design SR Latch using `AND`, `OR`, and `NOT` gates, we could use two gates of `NOR`, or `NAND` to achieve it.



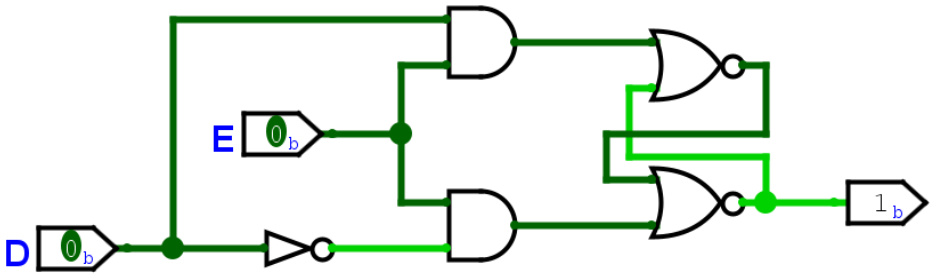
SR Latch using NOR

Note: The output of the latch will be on the Reset side, the inverted output will be on the Set side.

D-Latch

As you may notice, the design of the SR latch is kinda useless to store data, as you will have to drive the `S`, and `R` singles, but SR Latch is the basic building block for memory elements.

The D Latch has two inputs Data and Enable, the `D` input will be the data we want to store, and the `E` input will be the enabling single to store that data.



D-Latch

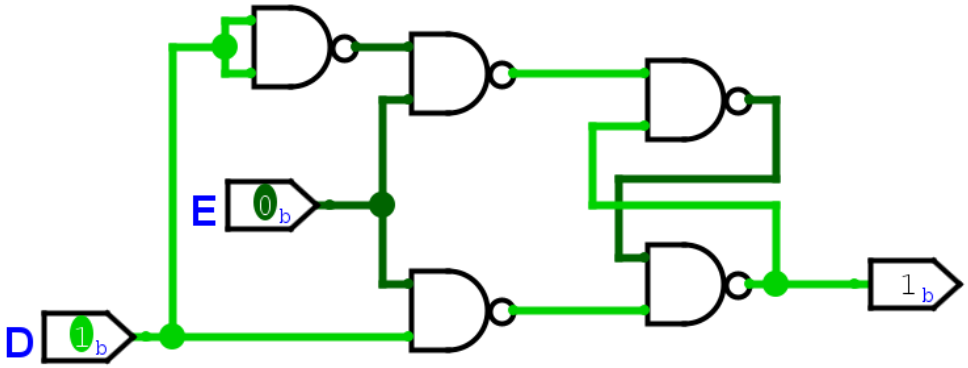
The truth table for the **D-Latch**, which will have two inputs `D`, and `E`, and output two singles `S`, and `R` will be:

E	D	S	R
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	0

which will result in a latch with this truth table:

E	D	Q
0	0	Previous State
0	1	Previous State
1	0	0
1	1	1

We could design a **D-Latch** using only `NAND` gates using the following design.



D-Latch Using NANDs

Lab Tasks

Task 1: Build an SR Latch

In this task, you should design an SR Latch using `NOR` gates, you should simulate it using Logisim first.

Task 2: Convert your previous SR Latch to D-Latch

Using two `AND` gates, and an inverter build a D-Latch, you should simulate your design before building it.

Datasheets

- 74/02 datasheet: https://www.futurlec.com/Datasheet/74ls/74LS02.pdf (https://www.futurlec.com/Datasheet/74ls/74LS02.pdf)
- 74/04 datasheet: https://www.futurlec.com/Datasheet/74ls/74LS04.pdf (https://www.futurlec.com/Datasheet/74ls/74LS04.pdf)
- 74/08 datasheet: https://www.futurlec.com/Datasheet/74ls/74LS08.pdf (https://www.futurlec.com/Datasheet/74ls/74LS08.pdf)

tags: `Digital Design` `Digital` `IUG` `Computer Engineering`

End Of Lab 6