

الجامعة الإسلامية العالمية ماليزيا  
INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA  
يُونِيسَيْتِي إِسْلَامُ، إِنْتَارَا يَغْسِيَا مِلْسِيَا

*Garden of Knowledge and Virtue*

**KULLIYAH OF ENGINEERING  
DEPARTMENT OF MECHATRONICS**

**(MCTE 2332)**

**DIGITAL SYSTEM AND MICROPROCESSOR  
SECTION 1**

**project  
(FIBONACCI NUMBERS GENERATOR)**

**Abubakar Ahmed Abdi Hassan 1726823**

# INTRODUCTION

Why do we learn mathematics?

Essentially, for three reasons: calculation, application, and inspiration.

Mathematics is the science of patterns, and we study it to learn how to think logically, critically, and creatively, but too much of the mathematics that we learn in school is not effectively motivated, and when the students ask, "Why are we learning this?" then they often hear that they'll need it in an upcoming math class or on a future test. But wouldn't it be great if we did mathematics simply because it was fun or beautiful or because it excited the mind? Now, let me give you a quick example with my favorite collection of numbers, the Fibonacci numbers. Fibonacci numbers can be appreciated in many different ways. From a calculation standpoint, they're very easy to understand 0 1 1 2 3 5 8 13 21 ... and so on. So each number is summed with the number prior to it to give the next number. Fibonacci was actually named the Leonardo of Pisa, and these numbers appear in his book "Liber Abaci," which taught the Western world the methods of arithmetic that we use today. In terms of applications, Fibonacci numbers appear in nature surprisingly often. The number of petals on a flower is typically a Fibonacci number, or the number of spirals on a sunflower or a pineapple tends to be a Fibonacci number as well.

Now let me present to you my logisim project .. and it is a "Fibonacci numbers generator"

## EQUATION:-

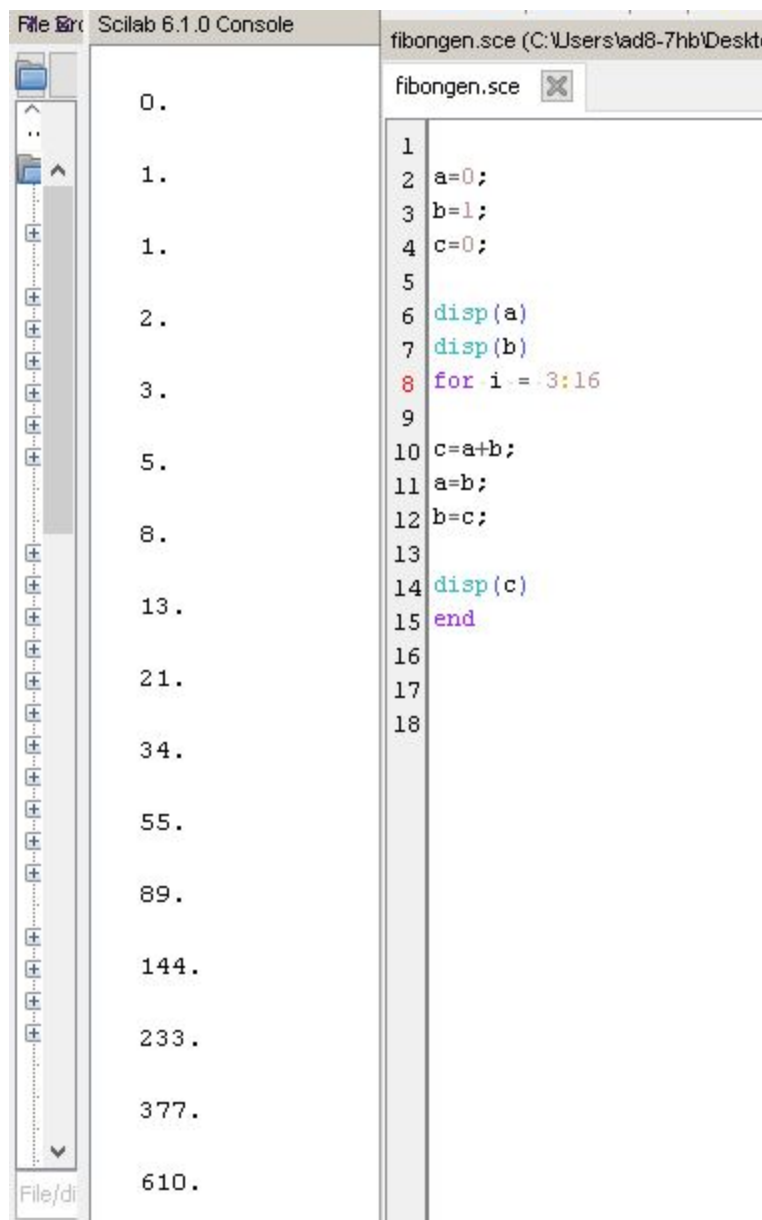
The equation that represents the Fibonacci series is:

$$F_{n+1} = F_n + F_{n-1}$$

Where (n) represents the number of term minus 1. So if we want the second term then  $n=2-1=1$ .

## THE CODE:-

To represent the same equation in Scilab .. the code will be:

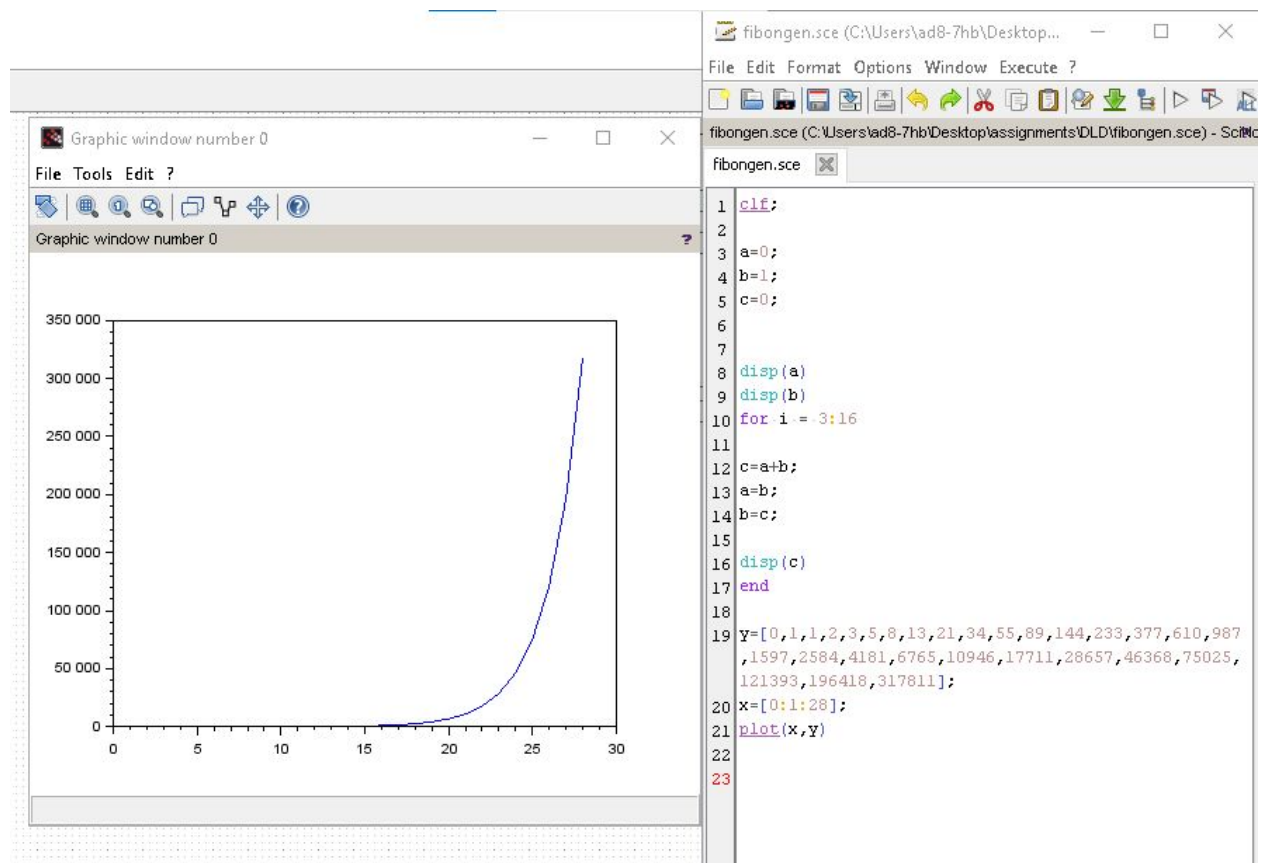


The image shows a screenshot of the Scilab 6.1.0 Console window. The console displays the output of a script named 'fibongen.sce'. The output is a list of Fibonacci numbers: 0., 1., 1., 2., 3., 5., 8., 13., 21., 34., 55., 89., 144., 233., 377., and 610. The script code is visible on the right side of the window, showing the initialization of variables a, b, and c, and a loop that calculates the Fibonacci sequence up to the 16th term.

```
File Edit Scilab 6.1.0 Console fibongen.sce (C:\Users\ad8-7hb\Desktop)\fibongen.sce
0.
1.
1.
2.
3.
5.
8.
13.
21.
34.
55.
89.
144.
233.
377.
610.

1
2 a=0;
3 b=1;
4 c=0;
5
6 disp(a)
7 disp(b)
8 for i = 3:16
9
10 c=a+b;
11 a=b;
12 b=c;
13
14 disp(c)
15 end
16
17
18
```

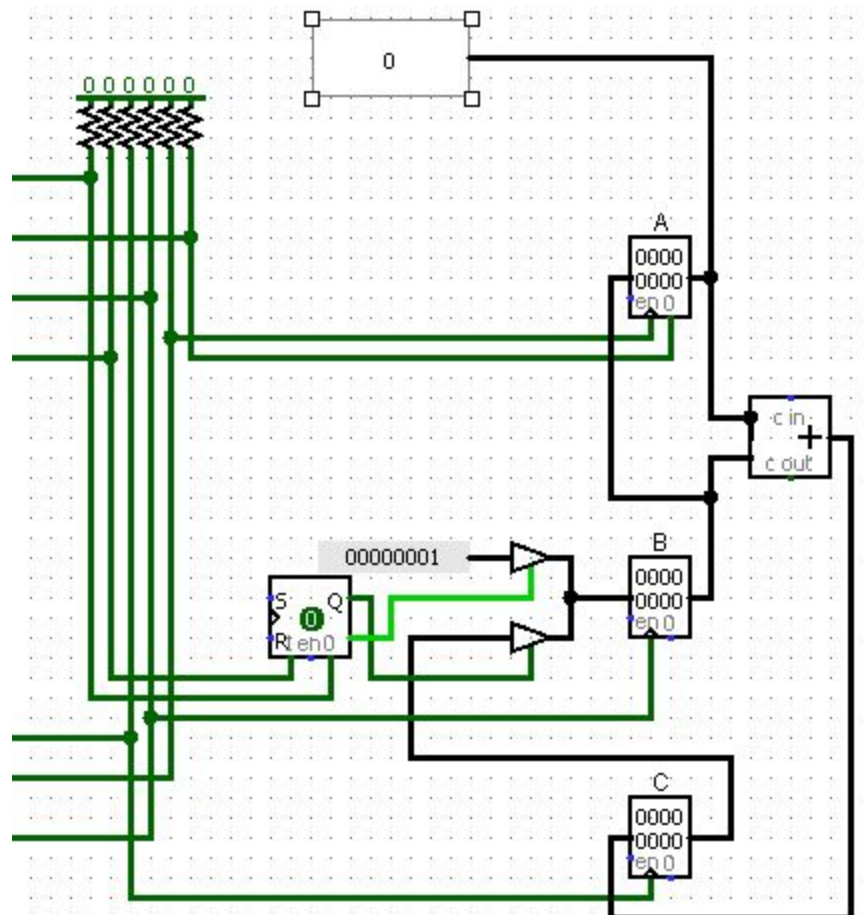
## THE GRAPH:-



### LOGISIM MODEL:-

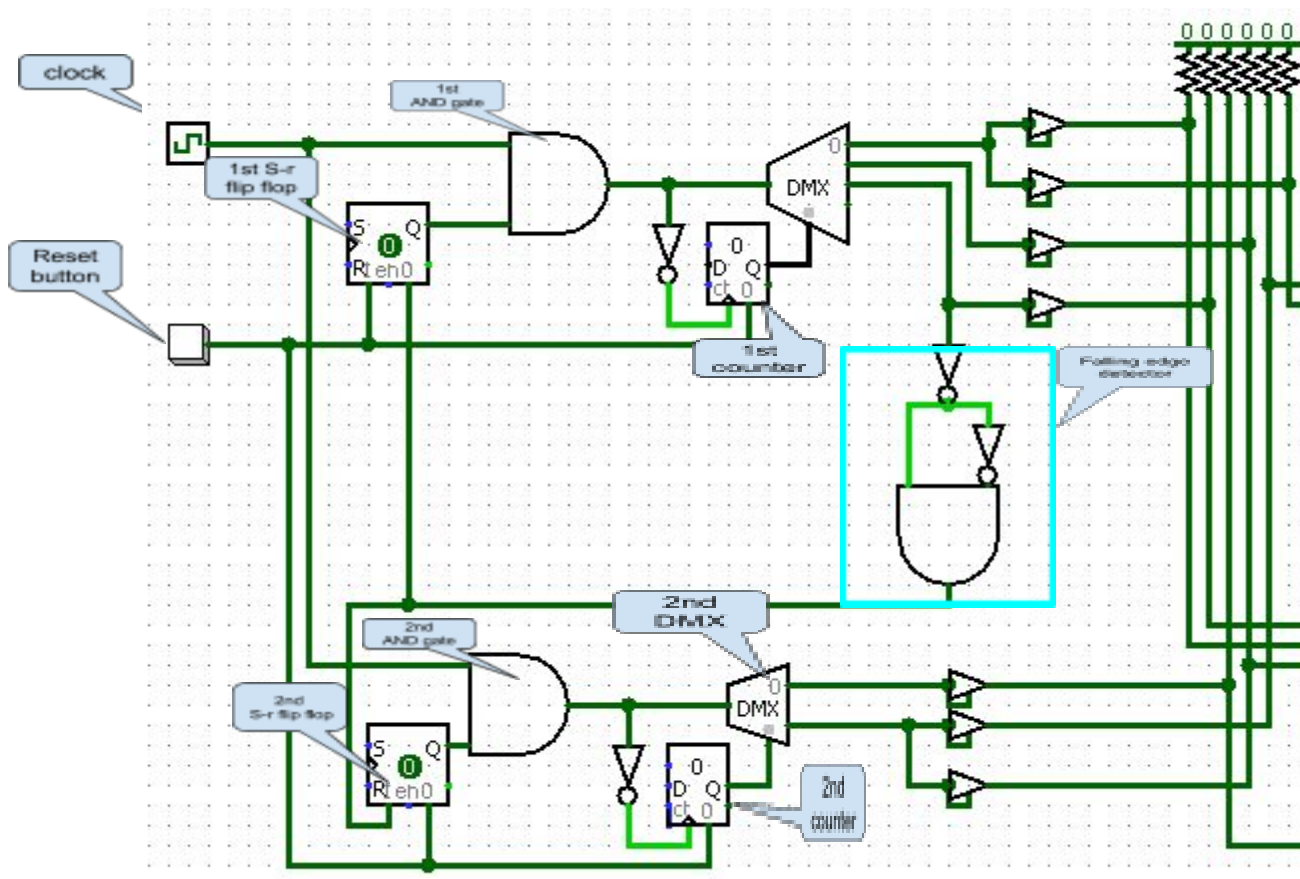
To implement the code in hardware. I used three registers. which basically are like variables, storing data. And I labeled them just like in my code A, B, and C .. to initialize the registers I'm simply starting by pulsing the reset line of register A. for register B, I used controlled buffers because firstly, I need to hard-wire a constant value to it, which is (the number 1), then in the next stage I need to put the output of the register C in register B, and if I did it directly, that will create a conflict on the input line for register B. Normally if we attempt to drive conflicting signals onto a common line we will get an error. However, with the controlled buffer it actually decides which signal to drive onto its output line. to control these buffers I used the memory circuit (SR flip-flop). when the circuit turns on it will be driving the number 1 on to register B, then after that, it will drive the output from C on to B's input line. Then I added an adder that will add the values of A and B then sum them and feed them directly into C.

I then placed pull resistors on each of the clock inputs to make sure that the signal rises from 0 to 1 .. and not from below zero. I attached a probe to the output of register A to display the generated numbers into their normal integer form.



To automatically control this circuit I used two demultiplexers.

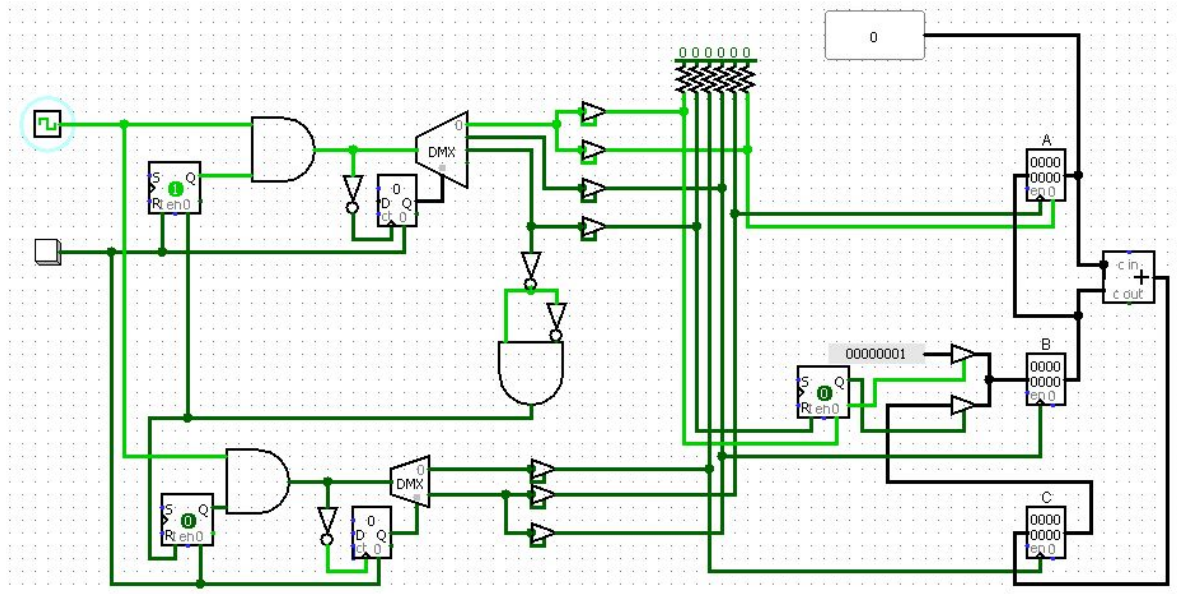
The first one will act as the starting spark .. it will make sure to reset the values and initiate the generator then it will stop. letting The second demultiplexer work as a non-stop motor and keep generating the numbers automatically. The Attached clock is the main signal driver in the circuit .. the AND gate and the S-R flip-flop are there to make sure that the signal won't go through after the mission of the first demultiplexer is over. the counter is there to update to the next output line when the clock signal passes through the AND gate. The signal is offset by the NOT gate. so the signal that is passed to the DMX goes through first, then the counter will get the signal then update to the next output line afterward. This is a (falling edge detector), and it is there to deactivate the first DMX, it's job is to deactivate the S-R flip-flop after the DMX has sent the signal to the 3rd output .. and that will prevent the signal from passing from the AND gate to the DMX. which will then deactivate that whole part of the circuit. As the (falling edge detector) deactivates the first DMX .. it also activates the second DMX at the same time. By turning on the second S-R flip flop. This is the reset button and it resets register A, the first and second counters and the first and second S-R flip flops



## HOW IT WORKS:-

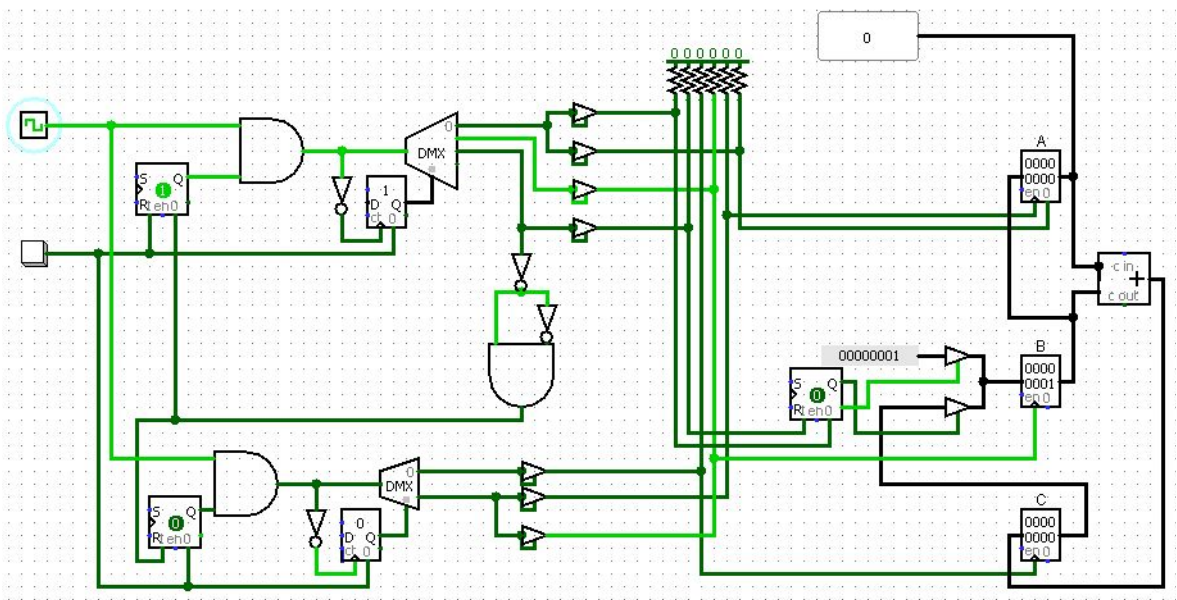
1st signal from the clock:-

- will set register A to zero, and the S-R flip flop will get ready to pass (the number 1) to register B



2nd signal from the clock:-

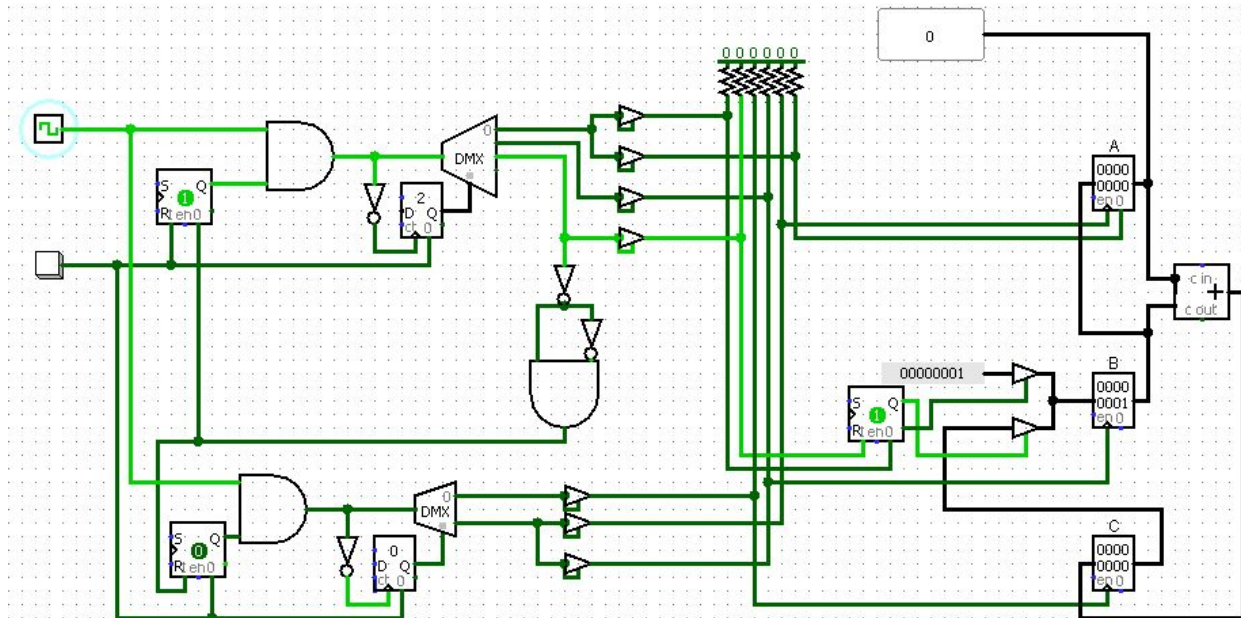
- will set register B to (00000001)
- The values of A&B are passed to the adder to be summed.





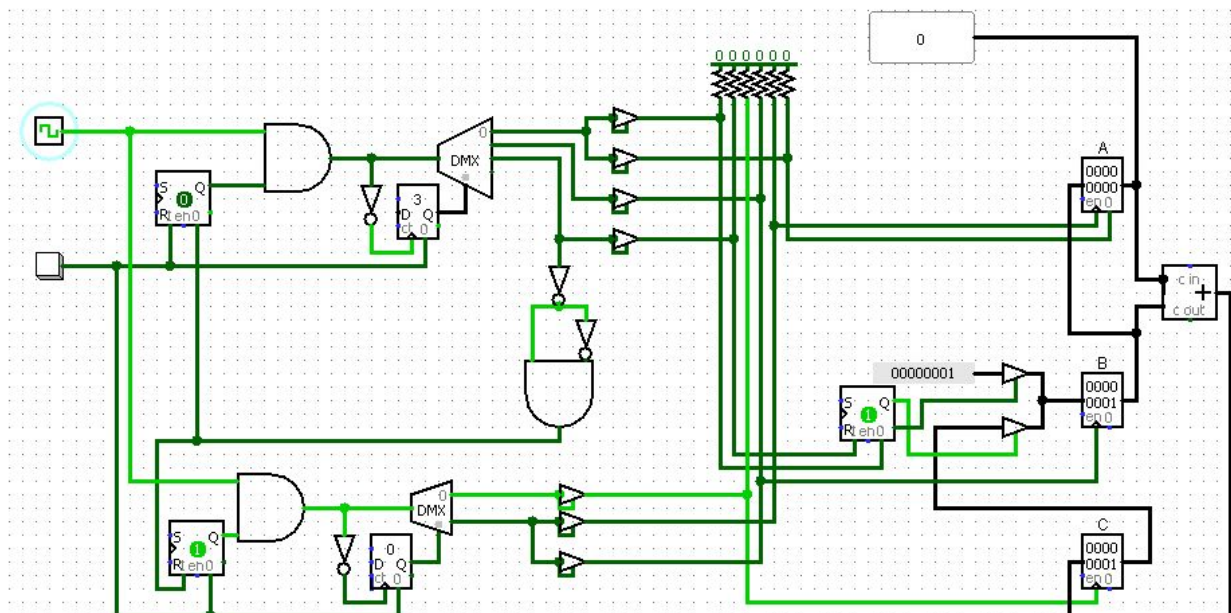
3rd signal from the clock:-

- the S-R flip flop will get ready to pass (the value of C) to register B.
- the (falling edge detector) will deactivate the first DMX .. and activate the second DMX at the same time.



4th signal from the clock:-

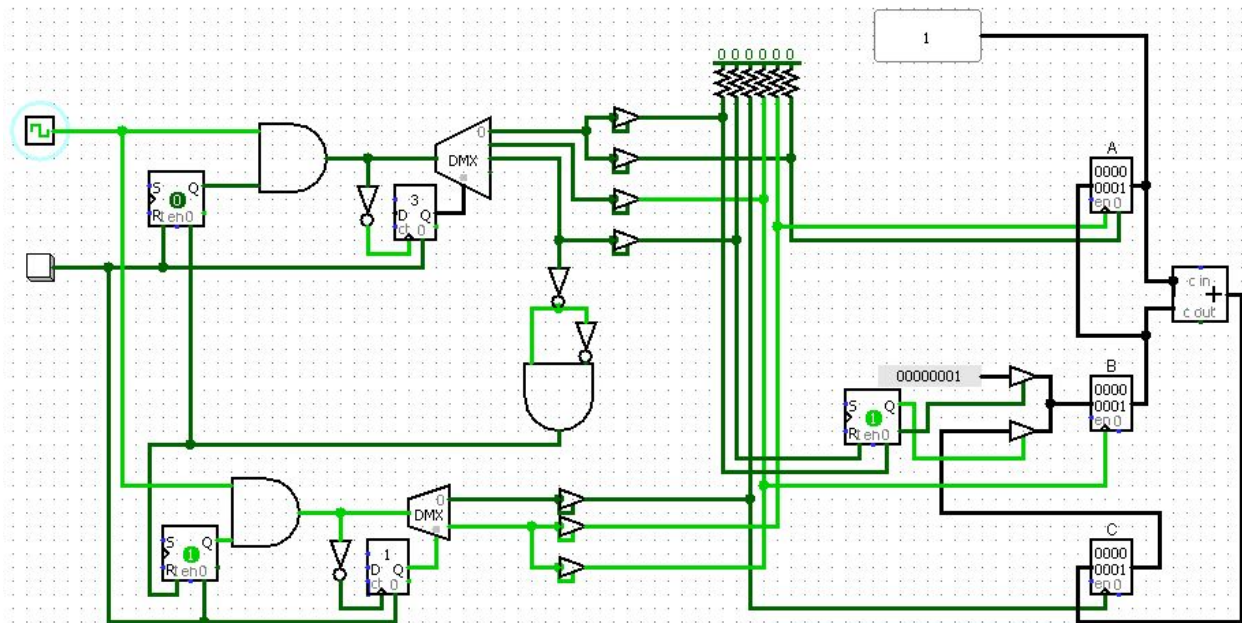
- will set register C to the sum of A+B from the adder.





5th signal from the clock:-

- will reset the value of A and make it equal to B, and reset the value of B and make it equal to C.
- The values are passed to the adder to be summed.



The 6th signal will do the same as the 4th and the 7th do the same as the 5th .. and the loop continues.

## TINKERCAD:-

This is an example of the project on Tinkercad .. it is not quite the same as my logisim design. mine does not have a 7 segment decoder .. but it is just for demonstration purposes.

