Fibonacci Sequence Calculator Circuit

CP220 Project Phase 1

Ahmad Wali

#169036947

DESCRIPTION:

The Fibonacci sequence, often called the “golden ratio” is the sequence of numbers starting from 0 and 1, where the next number is the sum of the previous two numbers. This can be given as the equation: F(n) = F(n-2) + F(n-1) where n is the index iteration.

An example of the Fibonacci sequence is as follows:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144…

In order to find the 3rd index of the sequence, we need to first find the 1st and 2nd values of the series recursively, and to find those values we will need to start from F(0) = 1 and F(1) = 1.

F(0) = 0

F(1) = 1

F(2) = F(1) + F(0) = 1+ 0 = 1

F(3) = F(1) + F(2) = 1 +1 = 2

Therefore, 2 is the 3rd index in the Fibonacci sequence.

INPUTS

The Fibonacci sequence calculator will calculate the n’th term of the sequence. It will receive the term provided by intaking a 3 bit binary value and calculating the next term iteratively. This means that the input value into the system can be labelled as some constant value called “X”.

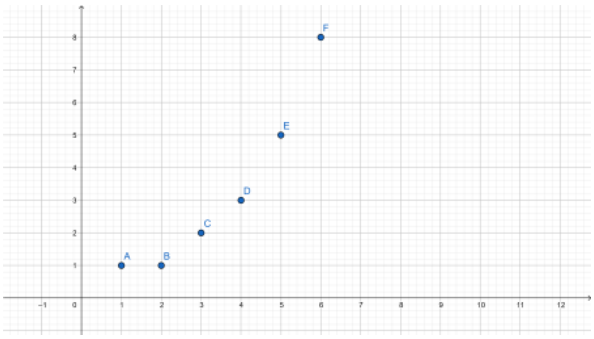
INPUT RESTRICTIONS:

There are restriction that the inputs MUST follow:

1. Must be an integer
2. Must be a number greater than 0

This is because the Fibonacci sequence is not defined for terms such as the “3.5” term, as the sequence is a discrete summation of a series of points. The Fibonacci sequence must also deal

with positive numbers as the sequence deals with iterations, and a “negative” iteration of something does not exist. Thankfully, working with binary allows us to ensure we are working with Z+ (whole integers) that follow these restrictions. As long as these conditions are met, there are no other error conditions nor ambiguous possibilities that exist.



This image helps to better illustrate the discontinuity of the series, aswell as how the series grows only in the positive domain.

OUTPUTS

The Fibonacci sequence will be returned to in the form of 4 binary values, that show what nth iterations value is (up to the 7^{th} iteration). The circuit will calculate the output recursively as it approaches the iteration of the sequence. Meaning, in order to get to value A(7), the computer must first calculate A(2), A(3) and so on till A(n-1) and A(n-2) are found and summed.

Binary input	Iteration A(n)	Sequence Value	Binary Output
000	A(0)	O	0000
001	A(1)	1	0001
010	A(2)	1	0001
011	A(3)	2	0010
100	A(4)	3	0011
101	A(5)	5	0101
110	A(6)	8	1000
111	A(7)	13	1101

END OF PART 1

Fibonacci Sequence Calculator Circuit

CP220 Project Phase 2

Ahmad Wali

#169036947

Truth Table for Inputs/Outputs:

The following table highlights the number fibonnaci sequence and which numbers it will be scanning through, the inputs and their outputs.

INDEX	A2,A1,A0	Fibonnaci Numbers	L3,L2,L1,L0
0	000	0	0000
1	001	1	0001
2	010	1	0001
3	011	2	0010
4	100	3	0011
5	101	5	0101
6	110	8	1000
7	111	13	1101

Truth Table for binary quantities:

To find the logical equations for the circuit, the binary we will begin by examining the binary inputs/outputs. The inputs are given by A, and the outputs by B. Each output bit will be considered separately inorder to form the equations for the circuit, as each output contains its own logical equation. Since this function has 4 possible outputs, there will be 4 equations to assess.

A2	A1	A0	L3	L2	L1	L0
0	0	0	0	0	0	0
0	0	1	0	0	0	1
0	1	0	0	0	0	1
0	1	1	0	0	1	0
1	0	0	0	0	1	1
1	0	1	0	1	0	1
1	1	0	1	0	0	0
1	1	1	1	1	0	1

Truth table for each binary inputs and outputs:

Since all outputs have their own logic equation, we can represent each output with a K-Map to determine the simplified Sum-Of-Products logical equation for the outputs. After we have found the logical equation, we can then assess them.

L0 TRUTH TABLE:

A2	A1	A0	L3
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

L1 TRUTH TABLE:

A2	A1	A0	L1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

L2 TRUTH TABLE:

A2	A1	A0	L2
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

L3 TRUTH TABLE:

A2	A1	A0	L3
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

K-Maps for all binary outputs:

Looking at the truth tables above, we can simplify these into K-Maps and return what their simplified Sum-Of-Products is. (note, I use ~ to simplify showing a negation)

L0 K-Map:

L0		A1,A0			
		00	**01**	**11**	**10**
A2	**0**	0	1	0	1
	1	1	1	1	0

Looking at the K-Map for the binary output above, we can see that we get 1's in the cases where:

L0 = (A2)(~A1) OR L0=(~A2)(A0)(~A1) OR L0 = (A2)(A1)(A0) OR L0=(~A2)(~A0)(A1)

L0 = (A2)(~A1) + (A2)(A1)(A0) + (~A2)(A0)(~A1) + (~A2)(~A0)(A1)

L1 K-Map:

L1		A1,A0			
		00	**01**	**11**	**10**
A2	**0**	0	0	1	0
	1	1	0	0	0

Looking at the K-Map for the binary output above, we can see that we get a 1 in the cases where:

L1 = (A2)(~A1)(~A0) OR when L2 = (~A2)(A1)(A0). Therefore, we can combine these statements and say

L1 = (A2)(~A1)(~A0) + (~A2)(A1)(A0)

L2 K-Map:

L2		A1,A0			
		00	**01**	**11**	**10**
A2	**0**	0	0	0	0
	1	0	1	1	0

Looking at the K-Map for the binary output above, we can see that we get a 1 in the cases where

L2 = (A2)(A0)(~A1) OR when L2=(A2)(A1)(A0). Therefore we can use Boolean algebra to combine this and say:

L2 = (A2)(A0)(~A1) + (A2)(A0)(A1)

L3 K-Map:

L3		A1,A0			
		00	**01**	**11**	**10**
A2	**0**	0	0	0	0
	1	0	0	1	1

Looking at the K-Map for the table above, we can see that we are able to to return a 1 in the cases where we we have A2 and A1 (A0 being irrelevant). In this case, we can say L3 = (A1)(A2).

Equation Testing:

Following the given equations, by pluggin them into maxima we get the following:

Inputs:

```
t1: a2 and a1;
t2: (a2 and a0 and (not a1)) or (a2 and a0 and a1);
t3: (a2 and (not a0) and (not a1)) or (a2 and a1 and a0) or ((not a2) and a1 and (not a0));
t4: (a2 and (not a1)) or (a2 and a1 and a0) or ((not a2) and a0 and (not a1)) or ((not a2) and (not a0) and a1);
```

Testing L0:

```
(%i6) fibonacci, a0=false, a1=false, a2=false;
(%o6) false                                                    0

(%i7) fibonacci, a0=true, a1=false, a2=false;
(%o7) true                                                     1

(%i8) fibonacci, a0=false, a1=true, a2=false;
(%o8) true                                                     1

(%i9) fibonacci, a0=true, a1=true, a2=false;
(%o9) false                                                    0

(%i10) fibonacci, a0=false, a1=false, a2=true;
(%o10) true                                                    1

(%i11) fibonacci, a0=true, a1=false, a2=true;
(%o11) true                                                    1

(%i12) fibonacci, a0=false, a1=true, a2=true;
(%o12) false                                                   0

(%i13) fibonacci, a0=true, a1=true, a2=true;
(%o13) true                                                    1
```

TESTING L1:

Output L_1

```
(%i6)  fibonacci, a0=false, a1=false, a2=false;
(%o6)  false                                                    0

(%i7)  fibonacci, a0=true, a1=false, a2=false;
(%o7)  false                                                    0

(%i8)  fibonacci, a0=false, a1=true, a2=false;
(%o8)  true                                                     1

(%i9)  fibonacci, a0=true, a1=true, a2=false;
(%o9)  false                                                    0

(%i10) fibonacci, a0=false, a1=false, a2=true;
(%o10) true                                                     1

(%i11) fibonacci, a0=true, a1=false, a2=true;
(%o11) false                                                    0

(%i12) fibonacci, a0=false, a1=true, a2=true;
(%o12) false                                                    0

(%i13) fibonacci, a0=true, a1=true, a2=true;
(%o13) true                                                     1
```

TESTING L2:

Output L_2

```
(%i6)  fibonacci, a0=false, a1=false, a2=false;
(%o6)  false                                                    0

(%i7)  fibonacci, a0=true, a1=false, a2=false;
(%o7)  false                                                    0

(%i8)  fibonacci, a0=false, a1=true, a2=false;
(%o8)  false                                                    0

(%i9)  fibonacci, a0=true, a1=true, a2=false;
(%o9)  false                                                    0

(%i10) fibonacci, a0=false, a1=false, a2=true;
(%o10) false                                                    0

(%i11) fibonacci, a0=true, a1=false, a2=true;
(%o11) true                                                     1

(%i12) fibonacci, a0=false, a1=true, a2=true;
(%o12) false                                                    0

(%i13) fibonacci, a0=true, a1=true, a2=true;
(%o13) true                                                     1
```

Teting L3:

```
(%i6)  fibonacci, a0=false, a1=false, a2=false;
(%o6)  false                                                    0

(%i7)  fibonacci, a0=true, a1=false, a2=false;
(%o7)  false                                                    0

(%i8)  fibonacci, a0=false, a1=true, a2=false;
(%o8)  false                                                    0

(%i9)  fibonacci, a0=true, a1=true, a2=false;
(%o9)  false                                                    0

(%i10) fibonacci, a0=false, a1=false, a2=true;
(%o10) false                                                    0

(%i11) fibonacci, a0=true, a1=false, a2=true;
(%o11) false                                                    0

(%i12) fibonacci, a0=false, a1=true, a2=true;
(%o12) true                                                     1

(%i13) fibonacci, a0=true, a1=true, a2=true;
(%o13) true                                                     1
```

Summary:

Given the above statements and the equations found, we can confirm that our equations are indeed correct by comparing them to the output of what Maxima shows us. In conclusion, the equations for the circuit are:

L3 = (A1)(A2).

L2 = (A2)(A0)(~A1) + (A2)(A0)(A1)

L1 = (A2)(~A1)(~A0) + (~A2)(A1)(A0)

L0 = (A2)(~A1) + (A2)(A1)(A0) + (~A2)(A0)(~A1) + (~A2)(~A0)(A1)

Fibonacci Sequence Calculator Circuit

CP220 Project Phase 3

Ahmad Wali

#169036947

Recalling what we learned in Project Phase 2, we learned what the equations for each digit (n) of the output (L) of a Fibonacci number were. As in, by looking at the K-Map of when each digit (n) for the output (L) would activate, we know under which conditions they light up, giving us a SOP (sum of product) expansion for each digit. It is important to acknowledge mistakes made in part 2. Upon going back and re-analyzing my work, I can see that I did NOT simplify by grouping with my K-Map, and had inaccurate outputs. The correct outputs would be:

L3 = (A1)(A2)

L2 = (A2)(A0)

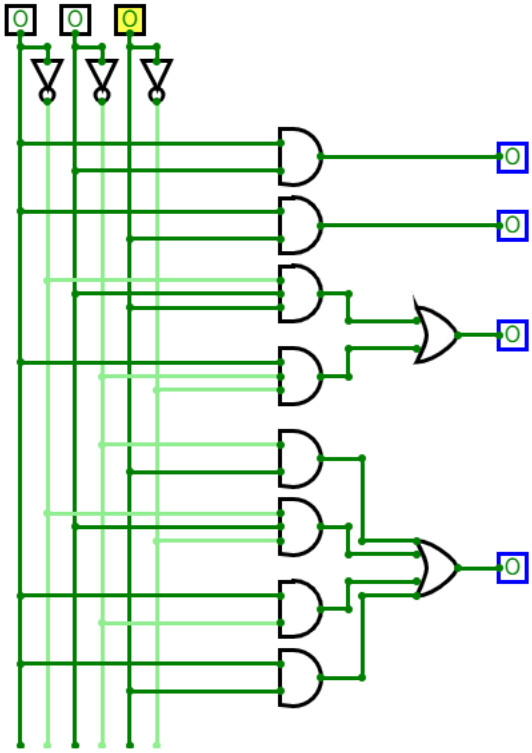
L1 = (A2)(~A1)(A0) + (~A2)(A1)(A0)

L0=(A0)(~A1)+(~A2)(A1)(~A0)+(A2)(~A1)+(A2)(A0)

CIRCUIT COMPONENT CHOICE EXPLAINED:

Given this information, we can take the POS expansion for each spot and create a diagram. It's important to remember that a ~ implies a NOT gate, + implies an OR gate and a multiplying two terms implies an AND gate. By applying Boolean logic, this means that for outputs of L3, we only need to use an AND gate (as its only true when we multiply A1 AND A2). However, for outputs of L0 we will need to use an OR gate to combine all the possibilities for when it should turn on (where each possibility will be comprised of AND gates and if needed a NOT gate). Each of those possibilities is represented as a term in the POS expansion for L0. This same logic applies for L2 and L1 as well. The OR gates symbolise situations where our circuit will yield a HIGH output, or times when a digit should fire to be a 1, which covers multiple cases of when a digit can fire to high when counting.

CIRCUIT FOR DESIGING THE FIBONNACI SEQUENCE

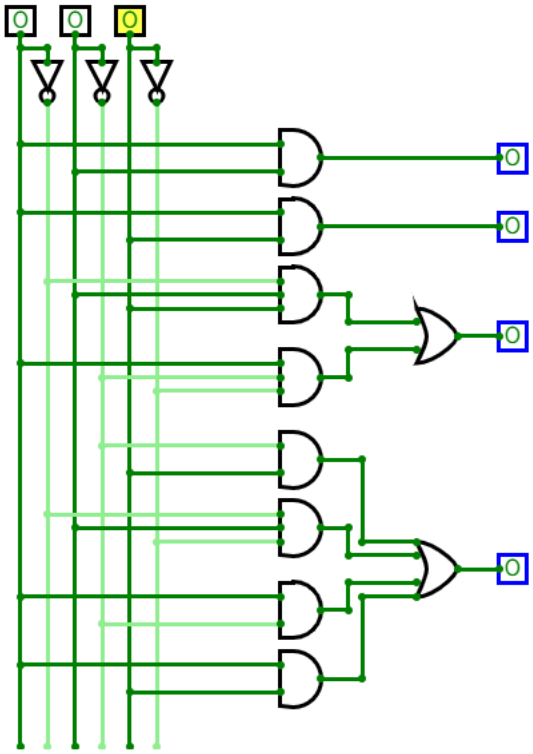


The circuit is constructed in such a way, where following the K map simplification for which conditions lead to each digit firing to HIGH, we construct AND cases and set them high according to the minimum POS case for each digit.

By referring back to the truth tables and the K-Maps showing when each component should fire high, we can create the following test cases:

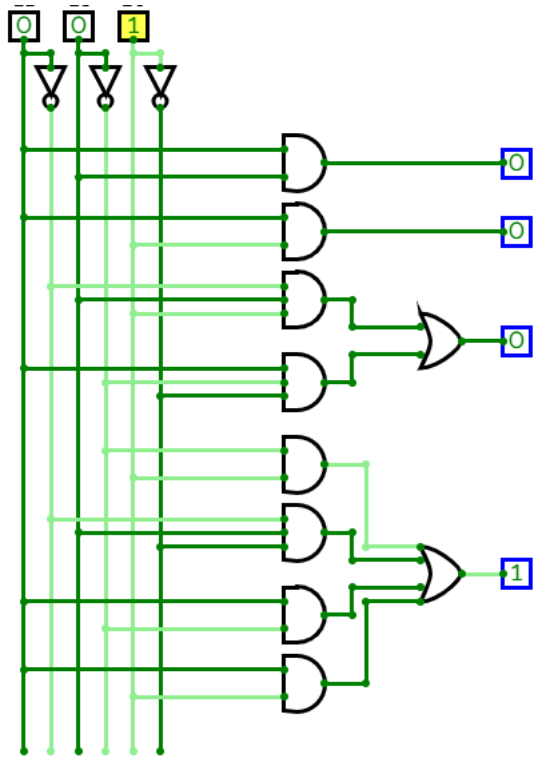
CASE 1:

In this case we compute the FIRST index of the simulation, starting at 0. A1,A2,A3 are at 000 and the output for L3,L2,L1,L1 should be 0000. This means that none of the lights should turn on as none of the truth tables for any of L are ~((A2)(A1)(A0)).



CASE 2:

In this case, the circuit is on its SECOND index of the simulation, which is at 1. A1,A2,A3 are at 001 and the output for L3,L2,L1,L1 should be 0001. This means that only L0 should light up as 1 is where the Fibbonnaci sequence starts.

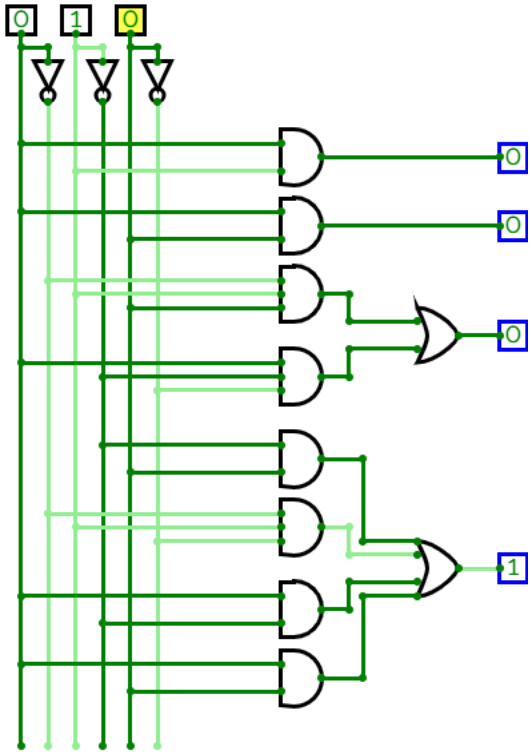


This matches the circuit on the case:

L0=(A0)(~A1)

CASE 3:

In this case, the circuit is on its THRD index of the simulation, which is at 2. A1,A2,A3 are at 010 and the output for L3,L2,L1,L1 should be 0001. This means that only L0 should light up as 1 is where the Fibonacci sequence has to computer 1+0=1 for the third case.

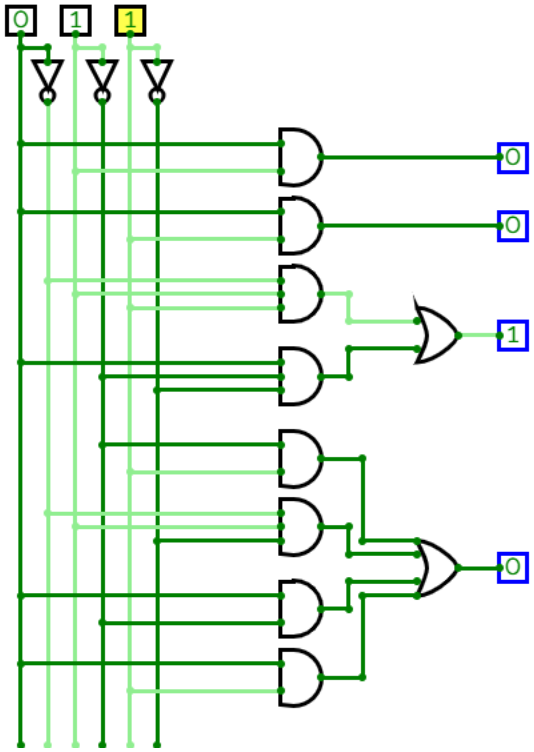


This matches the circuit on the case:

L0=(A0)(~A1)

CASE 4:

In this case, the circuit is on its FOURTH index of the simulation, which is at 3. A1,A2,A3 are at 011 and the output for L3,L2,L1,L0 should be 0010 (binary representation of 2). This means that only L1 should light up as 1 because it activates the case where is where the Fibonacci sequence has to computer 1+1=2 for the fourth case.

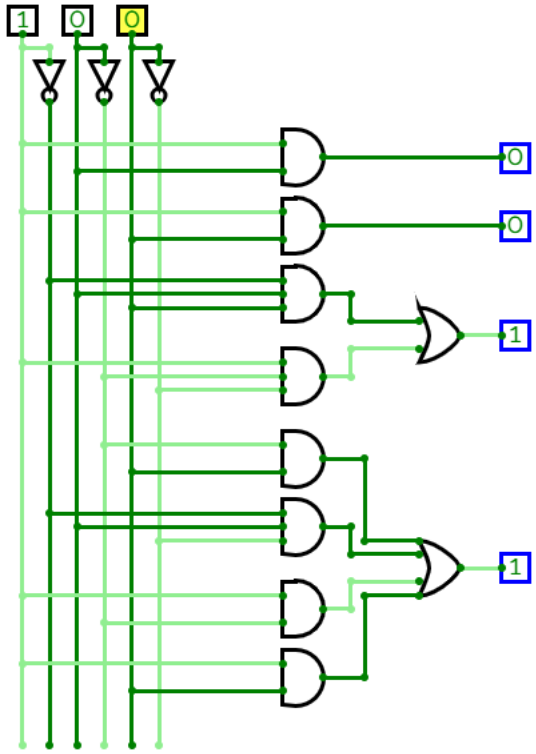


This matches the circuit for the case:

L1= (A1)(A2)

CASE 5:

In this case, the circuit is on its FIFTH index of the simulation, which is at 4. A1,A2,A3 are at 100 and the output for L3,L2,L1,L0 should be 0011 (binary representation of 3). This means that only L1 AND L0 should light up as 1 because it activates the case where is where the Fibonacci sequence has to computer 1+2=3 for the FIFTH case.



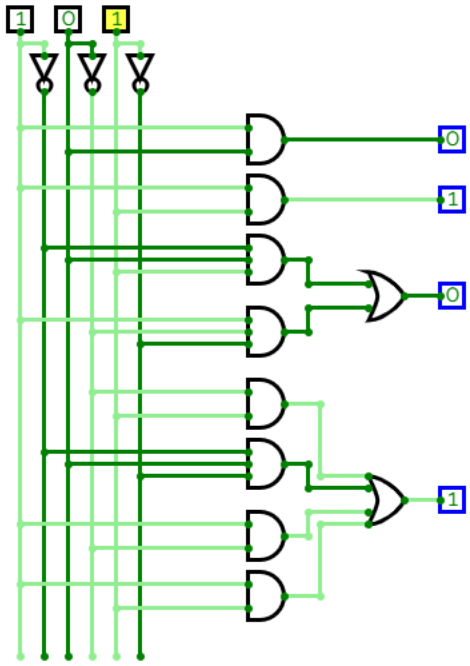
This matches the circuit for the case:

L(0) = (~A1)(A2)

L(1) = (A2)(~A1)(~A0)

CASE 6:

In this case, the circuit is on its SIXTH index of the simulation, which is at 5. A1,A2,A3 are at 101 and the output for L3,L2,L1,L0 should be 0101 (binary representation of 5). This means that only L2 AND L0 should light up as 1 because it activates the case where is where the Fibonacci sequence has to computer 2+3=5 for the SIXTH case.



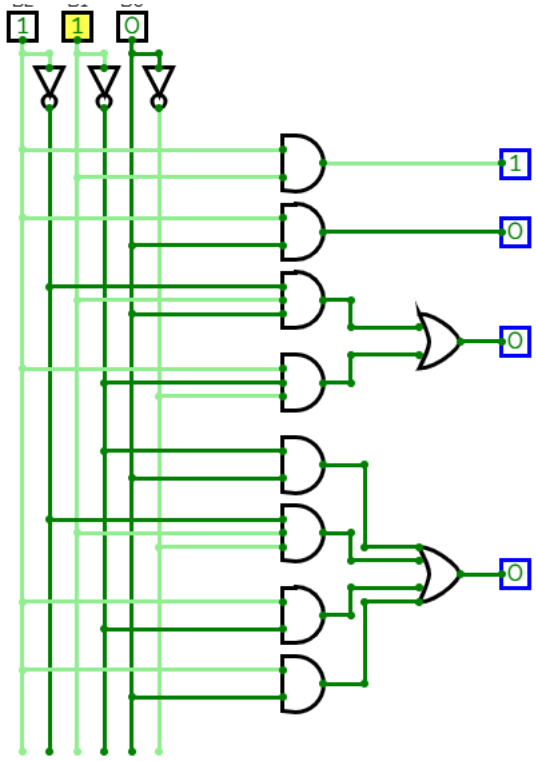
This matches for the cases:

L2 = (A0)(A1)

L0 = (~A1)(A0)

CASE 7:

In this case, the circuit is on its SEVENT index of the simulation, which is at 6. A1,A2,A3 are at 110 and the output for L3,L2,L1,L0 should be 1000 (binary representation of 8). This means that only L3 should light up as 1 because it activates the case where is where the Fibonacci sequence has to computer 5+3 = 8 for the FIFTH case.

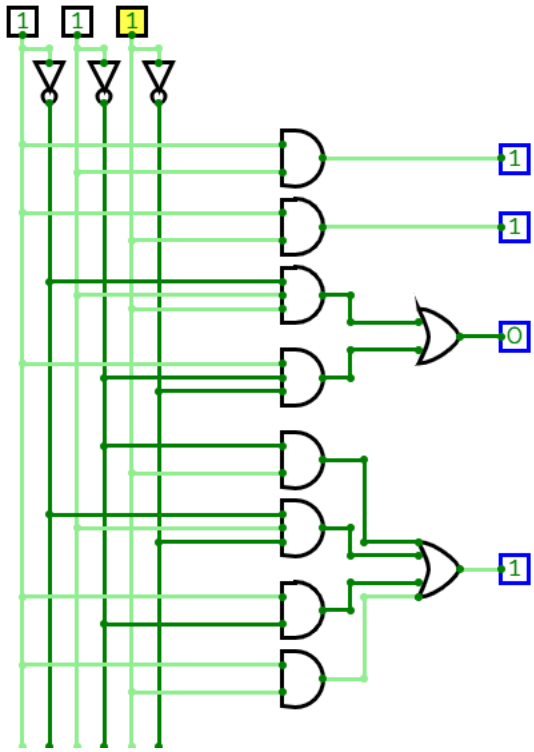


This matches for the cases:

L3 = (A2)(A1)

CASE 8:

In this case, the circuit is on its EIGTH (and final) index of the simulation, which is at 7. This is the final iteration because the circuit can only compute 2^n cases for n digits (3 digits, 263 = 8) and by starting at 0, our eight case is the 7th iteration. A1,A2,A3 are at 110 and the output for L3,L2,L1,L0 should be 1101(binary representation of 13). This means that L3,L2 and L0 should light up as 1 because it activates the case where is where the Fibonacci sequence has to computer 8+3 = 13 for the FIFTH case.



This works for the cases:

L3 = (A2)(A1)

L2 = (A2)(A0)

L0 =(A2)(A0)

SUMMARY:

In concnlsuion we can see that by creating K maps and finding when the minimum POS for when they fire high, we can create a circuit which accurately shows the binary output for the first 8 iterations of the fibonnaci sequence, starting at 0.