CSS 422: Hardware and Computer Organization

Homework: Sequential Circuits

Instructor Rob Nash

## Student Name: Thuan Tran

A full-adder is a combinational circuit (memory-less) that forms the arithmetic sum of two input bits (say a and b) and a carry in (Cin, so three input bits total). The full-adder provides two outputs in the form of the (S)um and the carry out (Cout). The input bits a and b represent the terms to be added, but the full-adder needs to also consider the carry in bit, too. The following questions are each worth 1 point.

1. Construct a truth table for the Full-Adder

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| INPUT | | | OUTPUT | |
| A | B | Cin | Sum | Cout |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

1. Construct a K-Map for the sum output (S) and derive the most reduced Boolean equation from your map.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | !A!B | !AB | AB | A!B |
| !Cin |  | 1 |  | 1 |
| Cin | 1 |  | 1 |  |

Derived equation: !A!BC + !AB!C + ABC + A!B!C

1. Construct a K-Map for the carry out (Cout) and derive the most reduced Boolean equation from your map.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | !A!B | !AB | AB | A!B |
| !Cin |  |  | 1 |  |
| Cin |  | 1 | 1 | 1 |

Derived equation: AB + BC + AC

1. Using ***Boolean Algebra,*** prove that S can be expressed as the XOR of the three inputs. Said another way, show the steps in Boolean Algebra that prove:

S = a XOR b XOR c

= (A XOR B) XOR C

= (A!B + !AB) XOR C

= (A!B + !AB) !C + C !(A!B + !AB)

= A!B!C + !AB!C + C( (!A + B) ( A + !B) )

Now focus on this part: C( (!A + B) ( A + !B) )

= C ( !AA + BA + !A!B + !BB)

= ABC + !A!BC ( Get rid of !AA and !BB)

Now plug back the result we have

A!B!C + !AB!C + ABC + A!BC, which are the same as answer to question 2

1. Using algebraic manipulation (like in the previous problem), prove that:

Cout = ab + (a XOR b)\*Cin

= AB + (!AB + A!B)C

= (AB + (!AB + A!B) ) (AB + C)

= ( AB + AB + !AB + A!B) (AB + C)

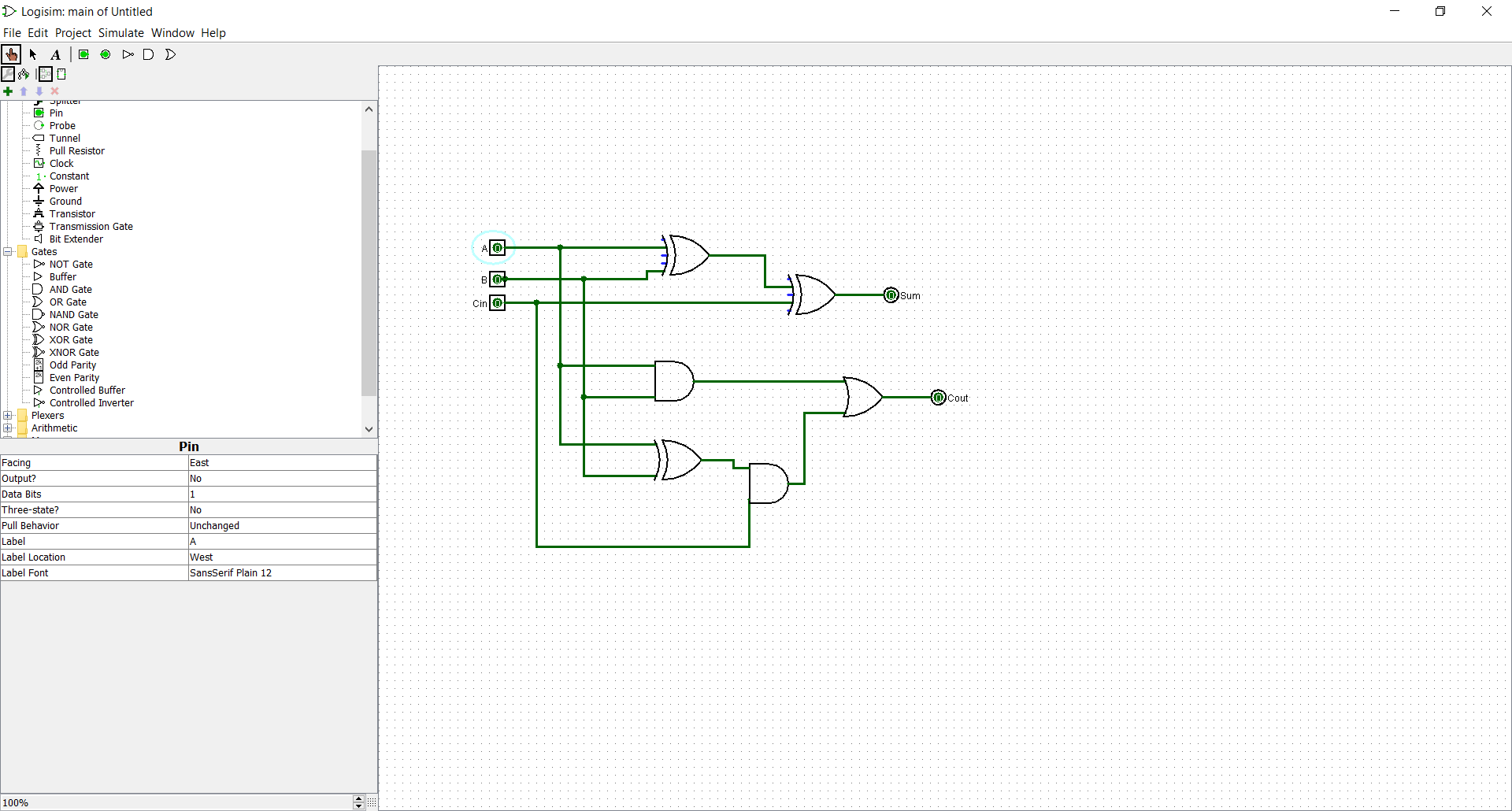
= ( A(B+!B) + B(A + !A) ) (AB + C)

= (A + B) (AB + C)

= AAB + AC + BAB + BC

=AB + AC + AB +BC = AB + AC +BC

1. ***Based on the previous formulas from (4) and (5)***, draw the circuit for the full-adder in Logisim; copy and paste a screen shot here and also submit the “.circ” file to the website. Note that just a screenshot is not sufficient.



1. Test your circuit and describe both the circuit and your test procedure.
   1. Does this adder look like the 1-bit adder you produced in the previous homework?
      1. If yes or no, why?

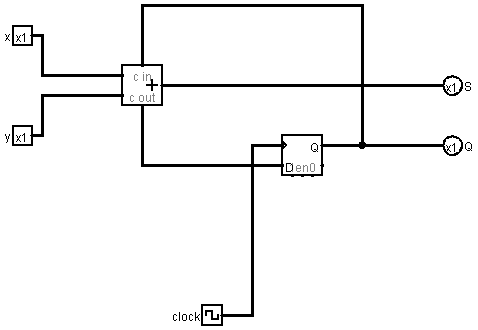
Yes, this looks like the 1bit adder. That is if we trace along the path to the Sum output, It can only be 1 if only 1 of the input to the second XOR gate is 1. Meaning only Cin can be and other must be 0 . Or Cin is 0 and at most 1 of A and B be 1. This correspond to the truth table

Test procedure: Manually adjust the input and record the output

Quick way: Analyze circuit and truth table was generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Xin | Yin | Cin | S | Cout |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

1. (3 points) The following sequential circuit includes state and a full-adder. Inputs are a,b,Cin and outputs are the next state of S and Cout Complete the following truth table to describe the sequential circuit presented here. (S is the output Sum, the D-ff stores the resultant carry-out Cout for use as the next carry-in, available on Q)



Hints: What is the relationship between the Carry-out before the clock and the Carry-in after the clock?

Second Hint: Do the columns “Before Clock” first.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Xin | Yin | Carry-in  (or Q) Before Clock | S  Before Clock | Carry-out  Before Clock | S  After Clock | Carry-out  After Clock |
| 0 | **0** | **0** | 0 | 0 | 0 | 0 |
| 0 | **0** | **1** | 1 | 0 | 0 | 0 |
| 0 | **1** | **0** | 1 | 0 | 1 | 0 |
| 0 | **1** | **1** | 0 | 1 | 0 | 1 |
| 1 | **0** | **0** | 1 | 0 | 1 | 0 |
| 1 | **0** | **1** | 0 | 1 | 0 | 1 |
| 1 | **1** | **0** | 0 | 1 | 1 | 1 |
| 1 | **1** | **1** | 1 | 1 | 1 | 1 |

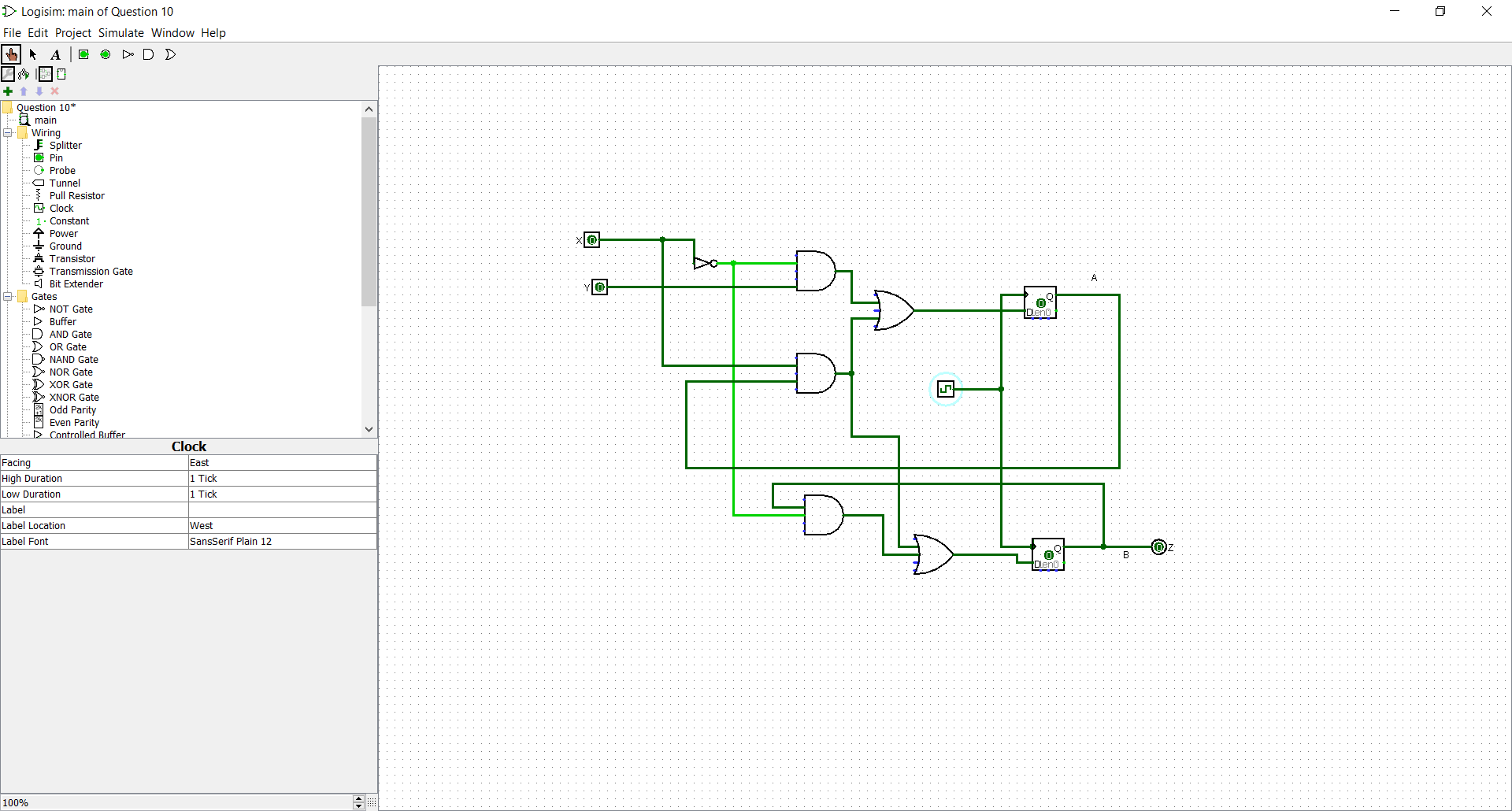
1. (5 pts) A sequential circuit has two D-FFs (A, B) and two inputs x and y, with one output z. The flip-flop input equations and circuit output is defined as:

Da = ~xy + xA

Db = ~xB + xA

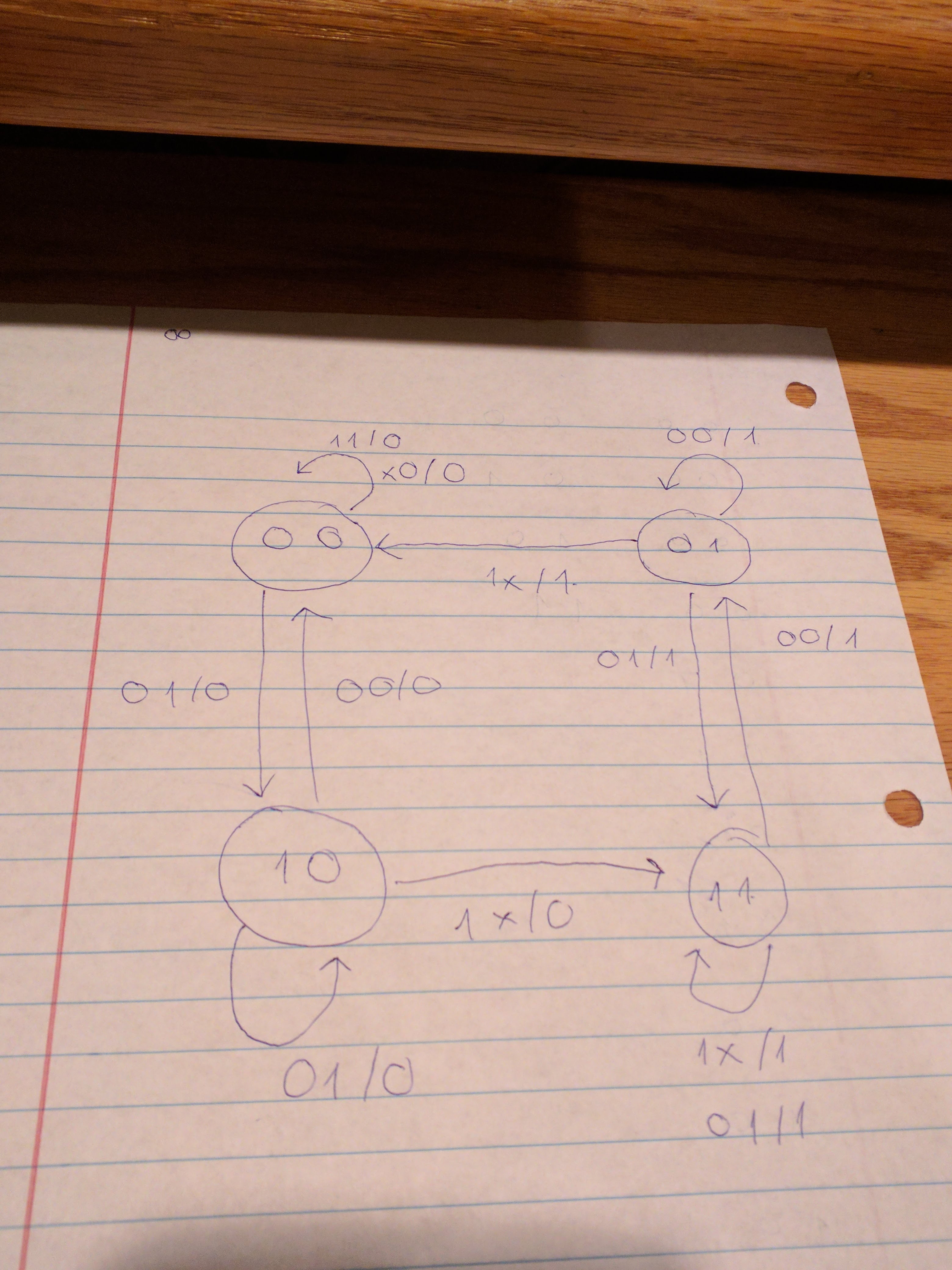
z = B

Draw the logic diagram of this circuit and test it in Logisim. Copy and paste a screenshot here and attach the “.circ” file separately.

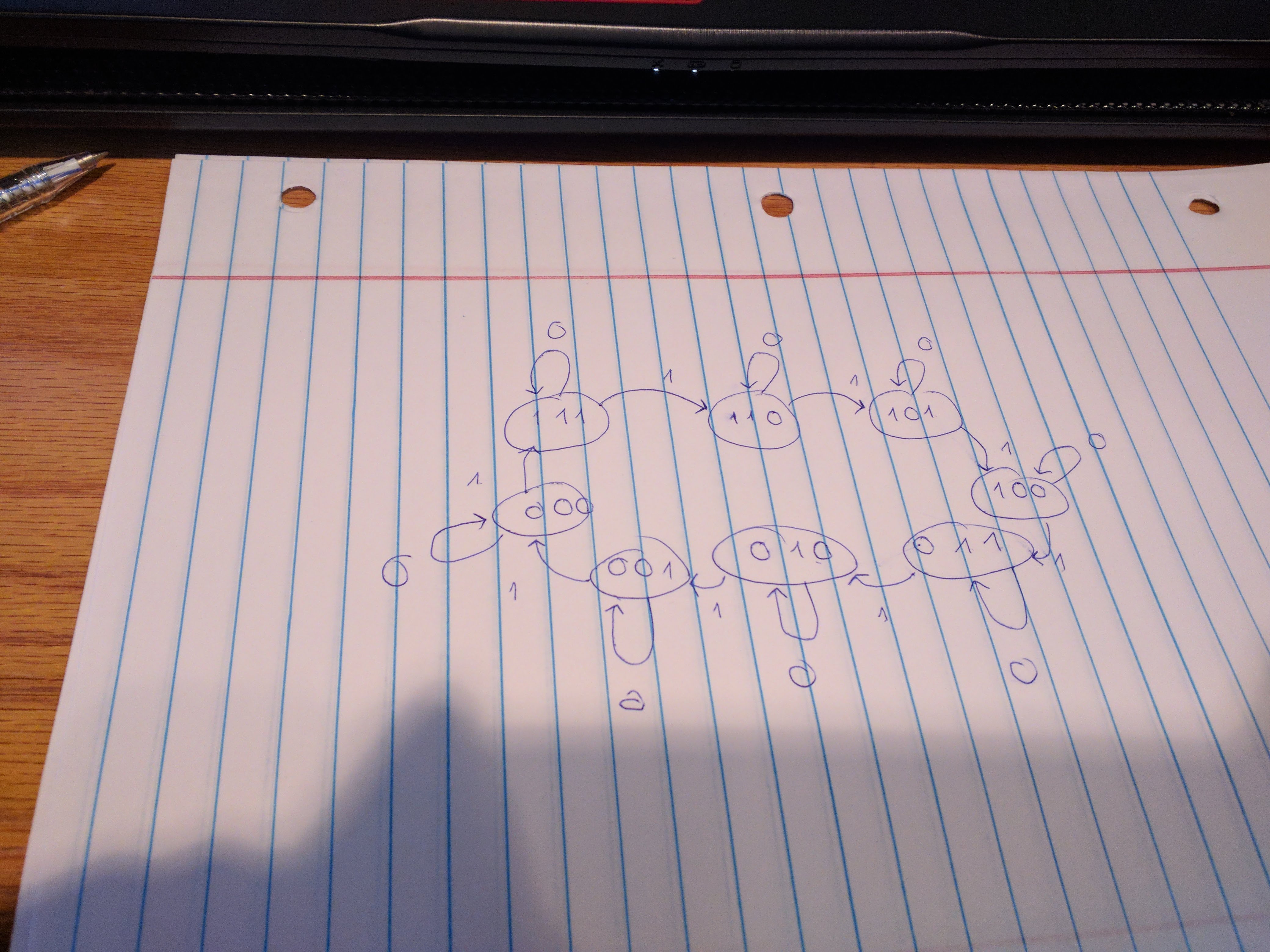


1. (2 pts) Construct a state diagram (a visual representation of an FSM) of this circuit.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Current | | Input | | Next | | Output |
| A | B | X | Y | A | B | Z |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |



1. (5 pts) Design a state-based circuit that uses 3 bits to count down from 111 to 000. This 3-bit count-down counter will use 3 JK flip-flops and one input x. If x is 0, don’t change state. If x==1, then progress the state sequence from 1112 to 0002. Start by drawing a state diagram (a visual representation of an FSM) here.



1. Complete the following excitation table to determine the inputs Ja, Ka, Jb, Kb, Jc, and Kc respectively for your previous count-down counter. Do note the unusual counting order is intentional.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | x | A t+1 | B  t+1 | C  t+1 | Ja | Ka | Jb | Kb | Jc | Kc |
| 0 | **0** | **0** | **0** | **0** | **0** | **0** | 0 | x | 0 | x | 0 | X |
| 0 | **0** | **0** | **1** | **0** | **1** | 0 | 0 | x | 1 | X | **0** | **X** |
| 0 | **0** | **1** | **0** | **0** | **0** | **1** | 0 | x | 0 | X | **x** | **0** |
| 0 | **0** | **1** | **1** | **0** | **1** | 1 | 0 | X | 1 | x | **x** | **0** |
| 0 | **1** | **0** | **0** | **0** | **1** | **0** | 0 | X | **X** | **0** | **0** | **X** |
| 0 | **1** | **0** | **1** | **1** | **0** | **0** | 1 | X | **X** | **1** | **0** | **X** |
| 0 | **1** | **1** | **0** | **0** | **1** | **1** | 0 | X | **X** | **0** | **X** | **0** |
| 0 | **1** | **1** | **1** | **1** | **0** | **1** | 1 | X | **X** | **1** | **X** | **0** |
| 1 | **0** | **0** | **0** | **1** | **0** | **0** | **X** | **0** | 0 | X | **0** | **X** |
| 1 | **0** | **0** | **1** | **1** | **1** | **0** | **X** | **0** | 1 | X | **0** | **X** |
| 1 | **0** | **1** | **0** | **1** | **0** | **1** | **X** | **0** | **0** | **X** | **X** | **0** |
| 1 | **0** | **1** | **1** | **1** | **1** | **1** | **X** | **0** | **1** | **X** | **X** | **0** |
| 1 | **1** | **0** | **0** | **1** | **1** | **0** | **X** | **0** | **X** | **0** | **0** | **X** |
| 1 | **1** | **0** | **1** | **0** | **0** | **1** | **X** | **1** | **X** | **1** | **1** | **X** |
| 1 | **1** | **1** | **0** | **1** | **1** | **1** | **X** | **0** | **X** | **0** | **X** | **0** |
| 1 | **1** | **1** | **1** | **0** | **0** | **0** | **X** | **1** | **X** | **1** | **X** | **1** |

1. Draw K-maps and then derive Boolean equations using those K-maps. Reduce the equation to its simplest form.

Ja = 1 big 8 loop and a big 4 loop

= A + Bx

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | !A!B | !AB | AB | A!B |
| !C!X |  |  | X | X |
| !CX |  | 1 | X | X |
| CX |  | 1 | X | X |
| C!X |  |  | X | X |

Ka = !A + Bx

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | !A!B | !AB | AB | A!B |
| !C!X | X | X |  |  |
| !CX | X | X | 1 |  |
| CX | X | X | 1 |  |
| C!X | X | X |  |  |

Jb = B + X

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | !A!B | !AB | AB | A!B |
| !C!X |  | X | X |  |
| !CX | 1 | X | X | 1 |
| CX | 1 | X | X | 1 |
| C!X |  | X | X |  |

Kb = !B + X

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | !A!B | !AB | AB | A!B |
| !C!X | X |  |  | X |
| !CX | X | 1 | 1 | X |
| CX | X | 1 | 1 | X |
| C!X | X |  |  | X |

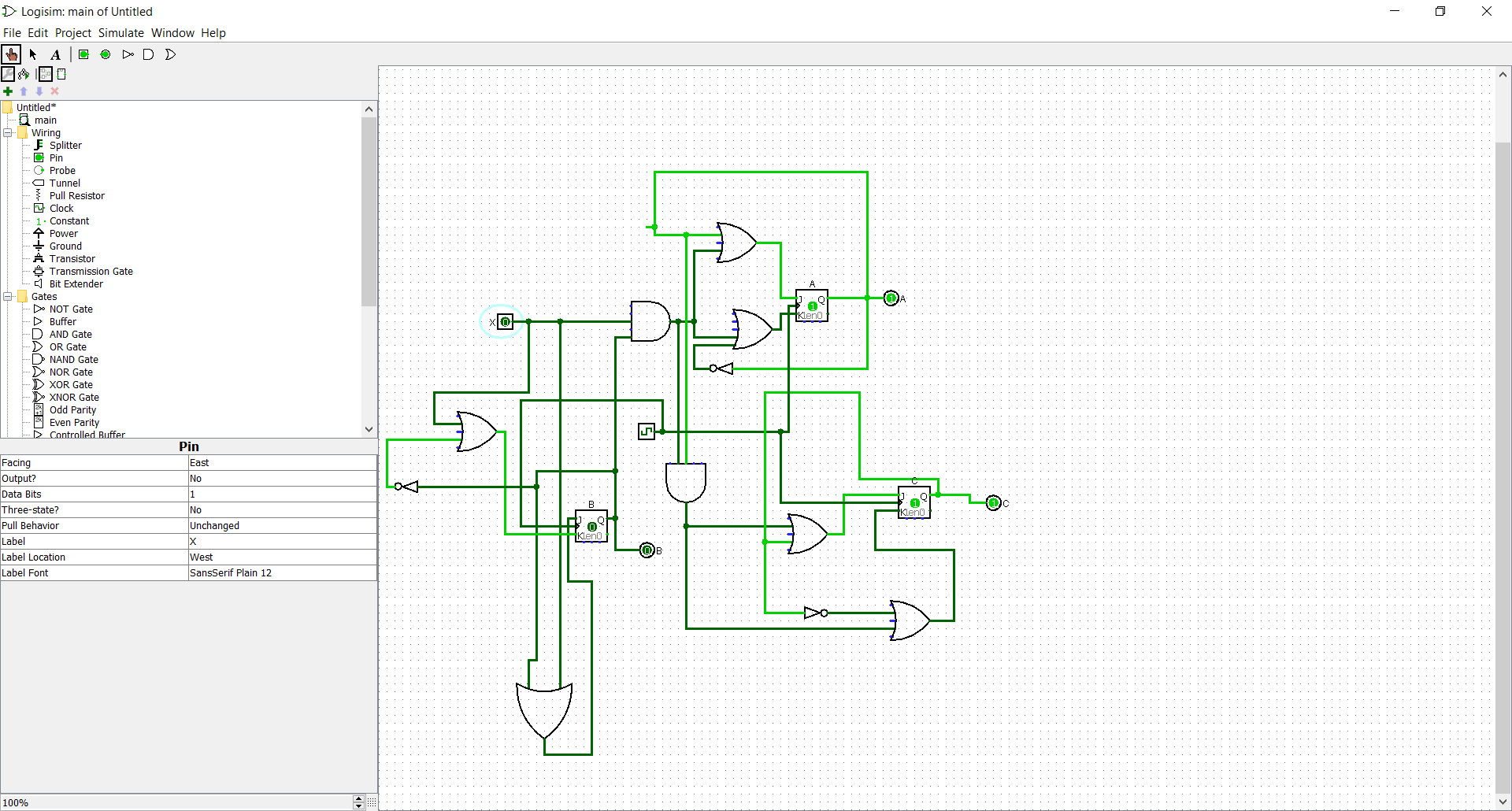
Jc = C + ABx

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | !A!B | !AB | AB | A!B |
| !C!X |  |  |  |  |
| !CX |  |  | 1 |  |
| CX | X | X | X | X |
| C!X | X | X | X | X |

Kc = !C + ABx

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | !A!B | !AB | AB | A!B |
| !C!X | X | X | X | X |
| !CX | X | X | X | X |
| CX |  |  | 1 |  |
| C!X |  |  |  |  |

1. Draw this circuit in Logisim and include a screen shot of the circuit here, as well as attaching the “.circ” file for analysis.



1. Test your system and report on your testing.
   1. What type of testing did you do?
      1. What did it reveal?

I set the clock to auto tick and log the value. And also manually “poke” the input to compare the result

I notice that this acts like a counter up from 000 to 111. I only need to switch the position of the table from ABC to CAB