CS 221 – Computer Organization

Boolean Algebra

Dr. Charles W. Kann

Read Chapter 1 of Digital Circuits Projects
https://cupola.gettysburg.edu/oer/1/

Base 2 representation

- Base 2 representations
  - True/False
  - On/Off
  - High/Low (voltage)
  - 0/1
  - Open/Closed
- All of these different ways of specifying a base 2 value are valid. The situation will determine which will be used, and you need to know them all.

Operator Represenation

- These operators are universal in that any Boolean function can be represented using only these operators.
- We will use the engineering symbols for these operators (*, +, '), not the mathematical symbols (^, v, etc.)

Universal Boolean Operators

- Binary Boolean Operators
  - AND – (A*B) or (AB)
  - OR – (A+B)
- Unary Boolean Operators
  - NOT – A'

Other Common Boolean Operators

- Other common Boolean operators we will see include
  - NAND – (A*B)' or (AB)'
  - NOR – (A+B)'
  - XOR – (A⊕B)

Boolean Relationships

(1) x + 0 = x

(2) x * 0 = 0

(3) x + 1 = 1

(4) x * 1 = x

(5) x + x = x

(6) x * x = x

(7) x + x’ = 1

(8) x * x’ = 0

(9) x + y = y + x

(10) xy = yx

(11) x + (y + z) = (x + y) + z

(12) x(yz) = (xy)z

(13) x(y+z) = xy + xz

(14) x + yz = (x+y) (x+z)

(15) (x+y)’ = x’y’

(16) (xy)’ = x’ + y’

(17) (x’)’ = x

Simple Boolean Algebra

- We can use these Boolean relationships to specify and simplify Boolean functions.
- The following functions show reductions followed by the rule that reduces them.
- f(A,B,C) = AB + AC + AD = A (B + C + D) [13]
- AB + A'B = (A+A')B [13] = B [7]
- ABC + AB’C + AB’C’ =
- ABC + AB’C + AB’C + AB’C’ [5] =

  AC(B+B’) + AB’(C+C’) [13] = AC + AB’ [7]
- The only rules used in this class are 5, 7, and 13.

Why simplify equations?

- We will be dealing with hardware. Each Boolean operation will be reduced to a hardware gates.
- All gates use power and produce heat.
- All gates have a latency.
- So simpler circuits result in:
  - Less heat
  - Less power
  - Shorter clock cycle

Truth Tables

- All Boolean functions can be represented by a (possibly infinite) truth table.
- Any function can be reduced to a boolean function, so any function can be represented by a truth table.
- Format of a truth table:
  - Input variable(s)
  - Results for one or more functions

Simple Truth Table

Truth table for F(A,B) = A*B

A	B	f(A,B)
0	0	0
0	1	0
1	0	0
1	1	1

Truth Table with two outputs

A	B	f1(A,B)	f2(A,B)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Normal Forms

- A minterm is a row in the truth table where the function is 1(true).
- Disjunctive Normal Form – Sum of Products
  - AND all the values in the minterm
  - OR all the minterms together
- f1(A,B) from the previous slide can be created from AND, OR, and NOT
  - AB' + A'B (first term is A=1, B=0, second term is A’=0, B=1)

Consider the following Truth Table

A	B	C	f(A,B,C)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

DNF of the truth table

- Or all terms that are 1 in the truth table
  - A'B'C + A'BC' + A'BC + AB'C + ABC' + ABC
- Simplify the expression

  A'B'C + A'BC' + A'BC + AB'C + ABC' + ABC

  = (A+A')B'C + (A+A')BC' or (A+A')BC [13]

  = B'C + BC' + BC [7]

  = B'C + B(C+C') [13]

  = B'C + B [7]

  = (B+B') (B+C) [14] *** Yikes...

  = B+C [7]

How do we get the simplest equation?

- Using Boolean Algebra to get the simplest Equation is not straight forward, as the use of Rule 14 showed.
- An easy way to find the simplest expression is to use a Karaugh Map (k-map)
- Using a Karaugh Map (k-map) will also how to simply reduce an expression using rules 5, 7, and 13 only.

Grey Codes

- Grey Codes are an easy way to group binary numbers such that only one digit varies between the binary numbers.
- Grey Codes have many uses such as error detection and correction, rotary encoders, genetic algorithms, and many others
- Grey Codes will be used in k-maps to group common terms.

Grey Codes

- Grey Codes are created by reflecting the current code, and then adding a 0 to the top half and a 1 to the bottom half.
- Note that all numbers (including the wrap around between the first and the last) differ by one bit.

Grey Code Examples

0		0		00
1	→	1	→	01
		1		11
		0		10

00		00		000
01		01		001
11	→	11	→	011
10		10		010
		10		110
		11		111
		01		101
		00		100

k-maps

- A k-map uses the Grey Codes to organize the terms in a truth table such that 4 block squares in the map correspond to a single term.
- Overlaps between the 4 block square result in the union being a two block square corresponding to two terms.
- The following slides show how that is accomplished.

Terms A and A’

A\BC	00	01	11	10
0				
1				

Red = A’ (Part of table where A = 0)
Green = A (Part of table where A = 1)

Terms B and B’

A\BC	00	01	11	10
0				
1				

Red = B’ (Part of table where B = 0)
Green = B (Part of table where B = 1)

Terms C and C’

A\BC	00	01	11	10
0				
1				

Red = C’ (Part of table where C = 0)
Green = C (Part of table where C = 1)

Solving a problem

A	B	C	f(A,B,C)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

DNF

A’BC’ + A’BC + ABC

Solve k-map

Translate truth table to k-map

A\BC	00	01	11	10
0			1	1
1			1	

Find groups:

No groups of 4, so no single terms

Two groups of 2, so two groups of two terms

Grouping shown in k-map above are:

AC + A'B

Note that red circled term is used in both equations!

Use Boolean Algebra to simplify

DNF is A'BC' + A'BC + ABC

= A'BC' + A'BC + A'BC + ABC [5] This is why it was important to notice A'BC Is used in both kmap terms

= A'B(C'+C) + (A+A')BC [13]

= A'B + BC [7] QED

Solving a problem

A	B	C	f(A,B,C)
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

DNF

A’BC’ + A’BC + ABC

Other simple k-maps

A\BC	00	01	11	10
0	1	1	1	
1	1	1	1	

Two groups of 4
B’ + C

Solving a problem

A	B	C	f(A,B,C)
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	1
1	1	1	0

DNF

A’BC’ + A’BC + ABC

Other simple k-maps

A\BC	00	01	11	10
0			1	
1	1			1

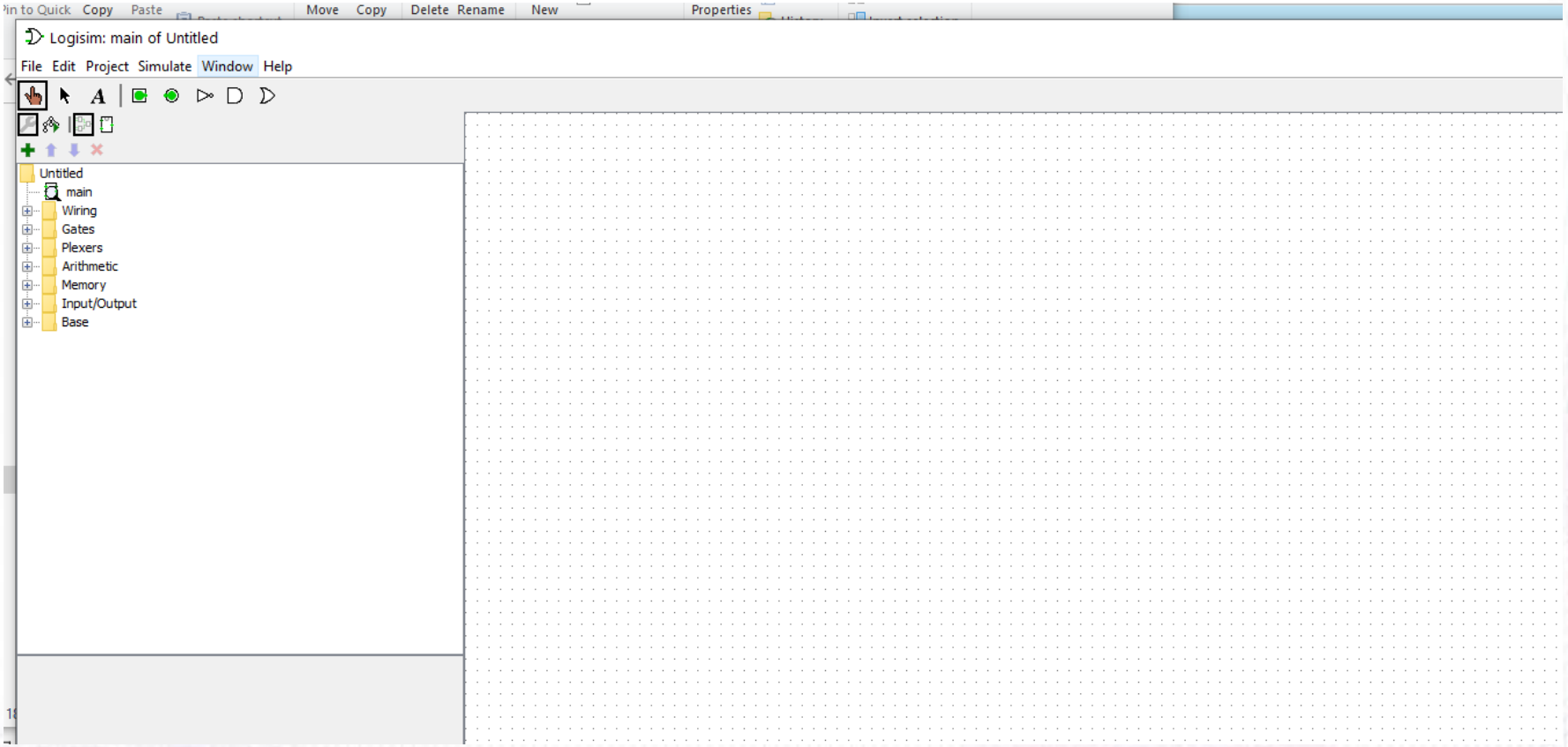
One group of two (wrap around on C') and one group of three
AC' and A'BC

Logisim

- Download Logisim
  - I use the older version, https://sourceforge.net/projects/circuit/
  - Those who want the latest, there is a Logisim Evolution at: https://sourceforge.net/projects/logisimevolution/
  - My programs later in the semester have not been ported to Logisim Evolution, so use it at your own risk.

Using Logisim

- After downloading and starting the jar file, you should see a screen like the following:



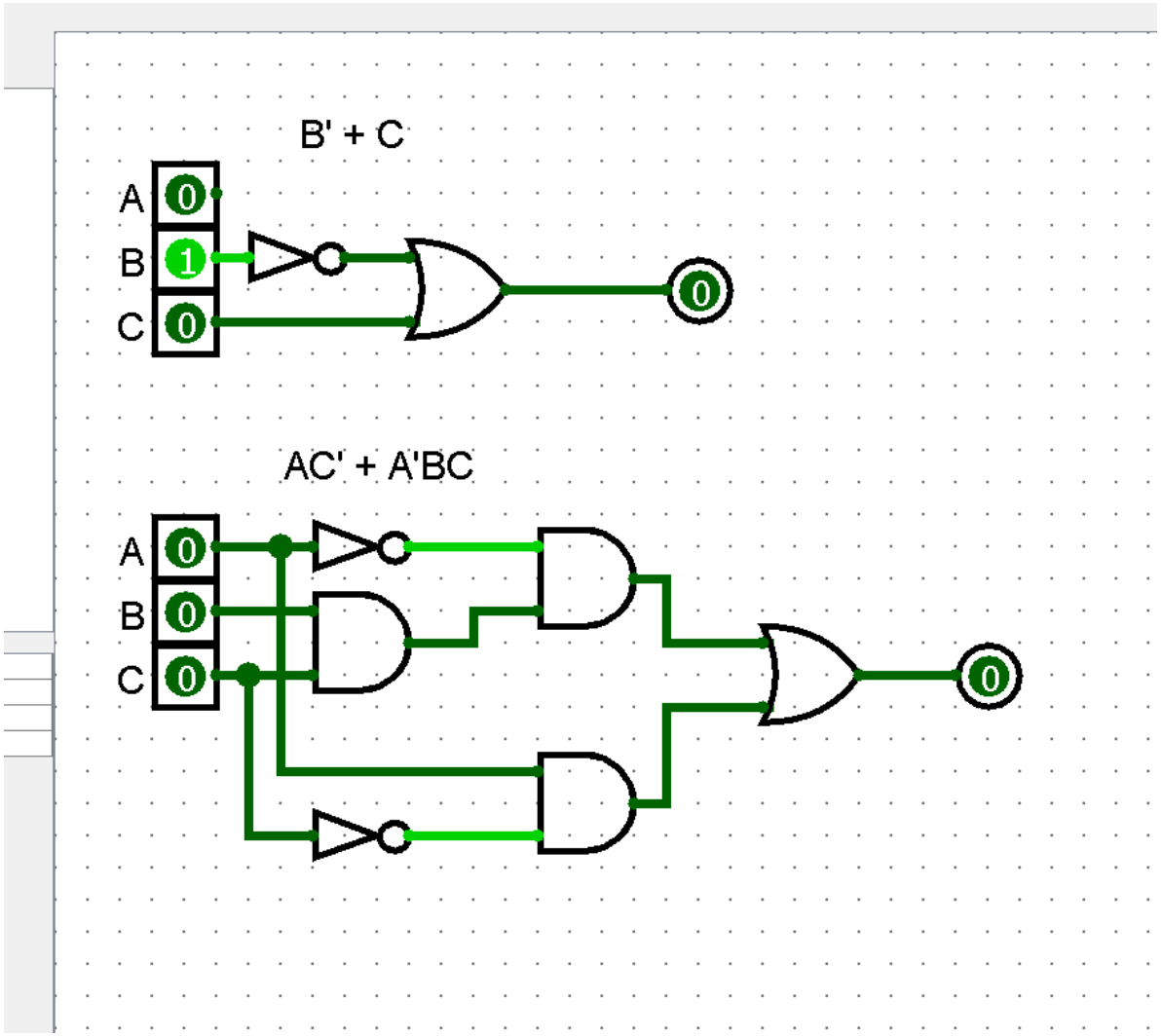
Implementing Circuits

- Let's implement the three circuits we solved with k-maps.
- Note that I selected the gate size to be narrow, and the number of inputs to be 2.
- After the circuits are implemented, click on the finger to run the circuit. Make sure it matches the truth table you used to derive the circuit.

Using Logisim

- Click on the gates button to see what gates you have available to you. Note that at the top of the interface you have the most common gates (AND, OR, and NOT) plus input and output pins. These are for convenience since these will be the ones you use most often.

Drawing circuits in Logisim



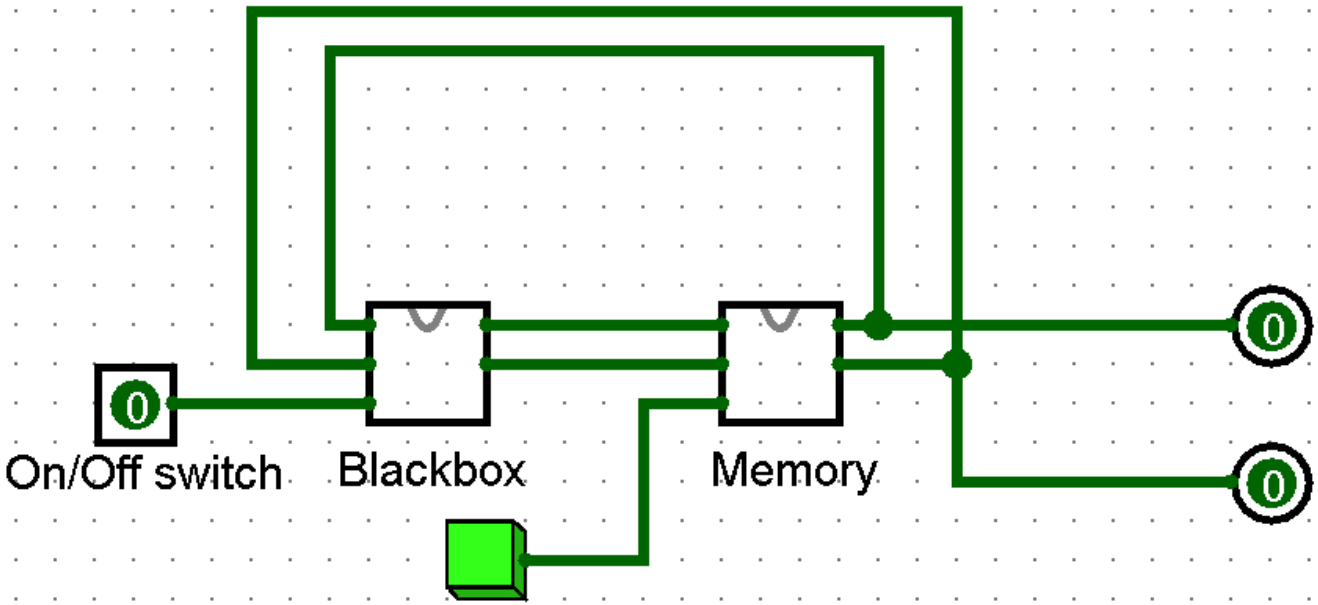
A real world problem

- A two-light problem, where the lights go “00, 01, 10, 11”, as shown in problem.

Overview of Logisim

- Let's start from a blackbox project, and create a working circuit solution.
- This is a mod-4 counter. It will count 0-1-2-3 by flashing lights. 0 is no lights on, 1 is the top light is on, 2 is the bottom light is on, and 3 is both light (1 and 2) are on.
- Repeats when it gets to the end.
- Button makes the circuit change to next state.
- This circuit is called "Mod4Counter.circ" in the notes. Open it in Logisim and run it.
  - If the switch is on (1) a button press advances the counter
  - If the switch is off (0) a button press does not advance the counter

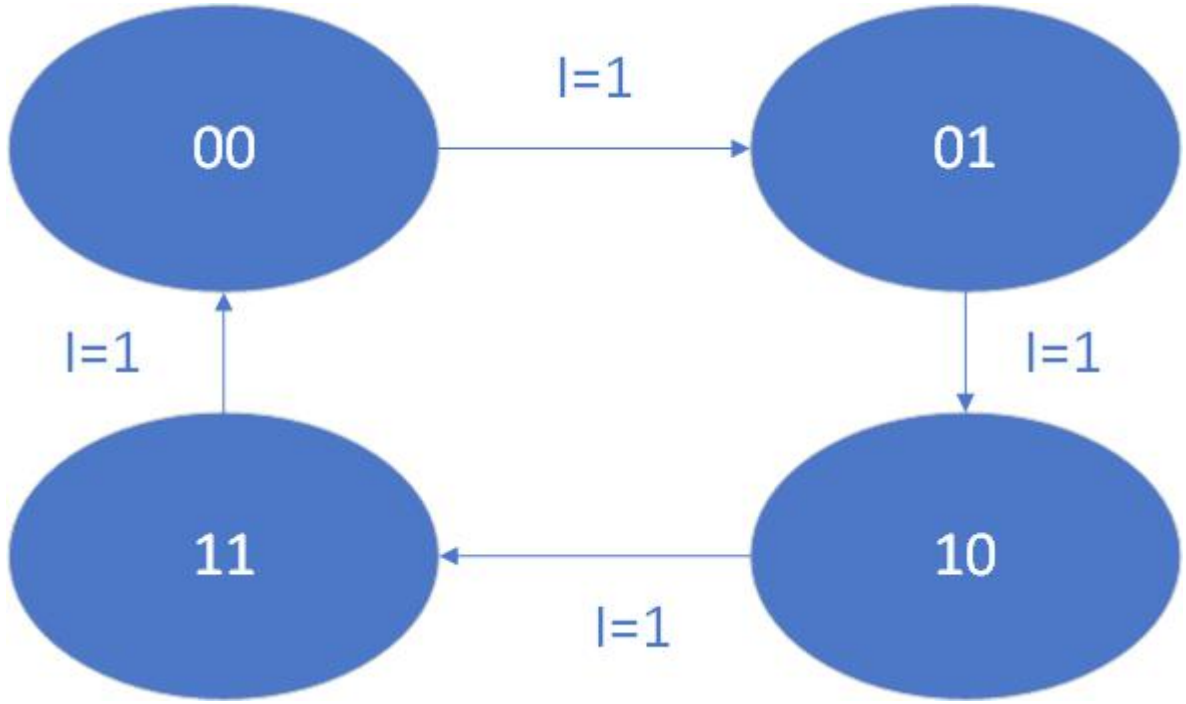
Solving a hardware problem



Representing the problem as a state diagram

- This problem can be represented as a state diagram.
  - The state is represented by two binary digits, having the values of 00, 01, 02, and 03.
  - If the switch is off, the lights do not change even if the button is pushed.
  - If the switch is on, the lights count from 0, 1, 2, 3, 0, 1, 2, 3, etc. each time the button is pressed.
- The state diagram is represented on the next slide.

Representing the problem in a state table



Translating the state diagram into state table.

- We represent the state as state_old (before the button is pressed), and state_new (after the button is pressed).
  - These are Q0 and Q0_new for the first bit in the two bit value, and Q1 and Q1_new for the second bit.
  - This can then be translated into a state table represented in the next slide.

State Diagram

I	Q0	Q1	Q0new	Q1new
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	1
1	0	0	0	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	0

Notice

- The state diagram looks just like a truth table, and we can use it as a truth table. Apply the steps in the first part of this lecture to solve it.

Kmaps

Q0new

I\Q0Q1	00	01	11	10
0	0	0	1	1
1	0	1	0	1

Q1new

I\Q0Q1	00	01	11	10
0	0	1	0	1
1	0	1	0	1

Solving for Q0_new and Q1_new

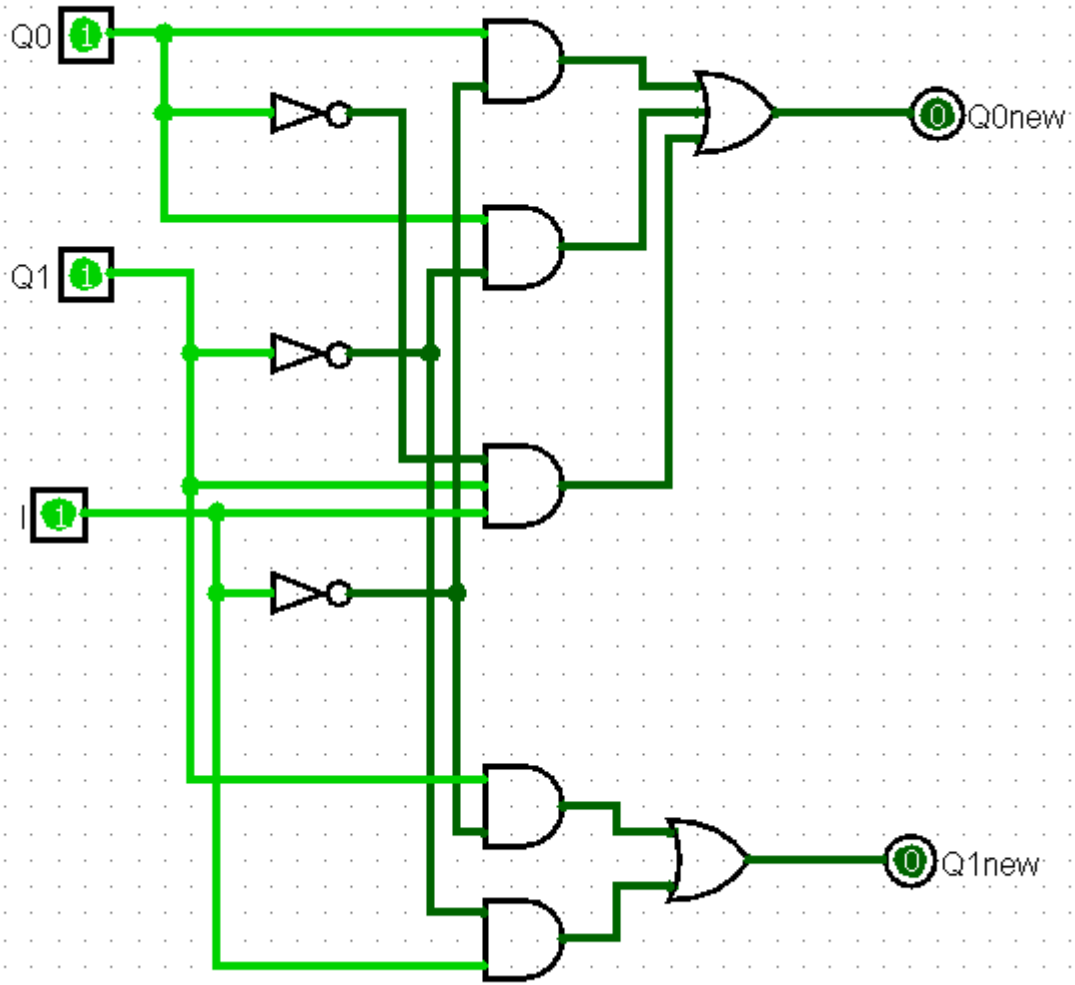
Q0new = Q0I’ + Q0Q1’ + Q0’Q1I

Q1new = Q1I’ + Q1’I

Draw the Circuit in the Black Box

- The circuit in the black box can now be drawn.
- In Logisim, if you double click on the black box you will see this is the circuit that was implemented.

Final “blackbox” circuit



Note that this type of programming is still used today

- Creating circuits this way to implement simple systems is still used in machines. It goes by the name of PLA, PAL, or PLD programming. Since CPU and FPGA chips are becoming cheaper, it is less common, but still around.
- PLD - https://ewh.ieee.org/r6/scv/sps/DecNickReconf.ppt
- PLA - https://www.tutorialspoint.com/digital_circuits/digital_circuits_programmable_logic_devices.htm

Assignments

Problem 1

Show the DNF form of the following equation. Simplify the equation to its simplest form. Implement the simple form of the equation in Logisim.

A	B	C	f(A,B,C)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Problem 2

Show the DNF form of the following equation. Simplify the equation to its simplest form. Implement the simple form of the equation in Logisim.

A	B	C	f(A,B,C)
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Problem 3

Show the DNF form of the following equation.
Simplify the equation to its simplest form.
Implement the simple form of the equation in Logisim.

A	B	C	F(A, B, C)
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1