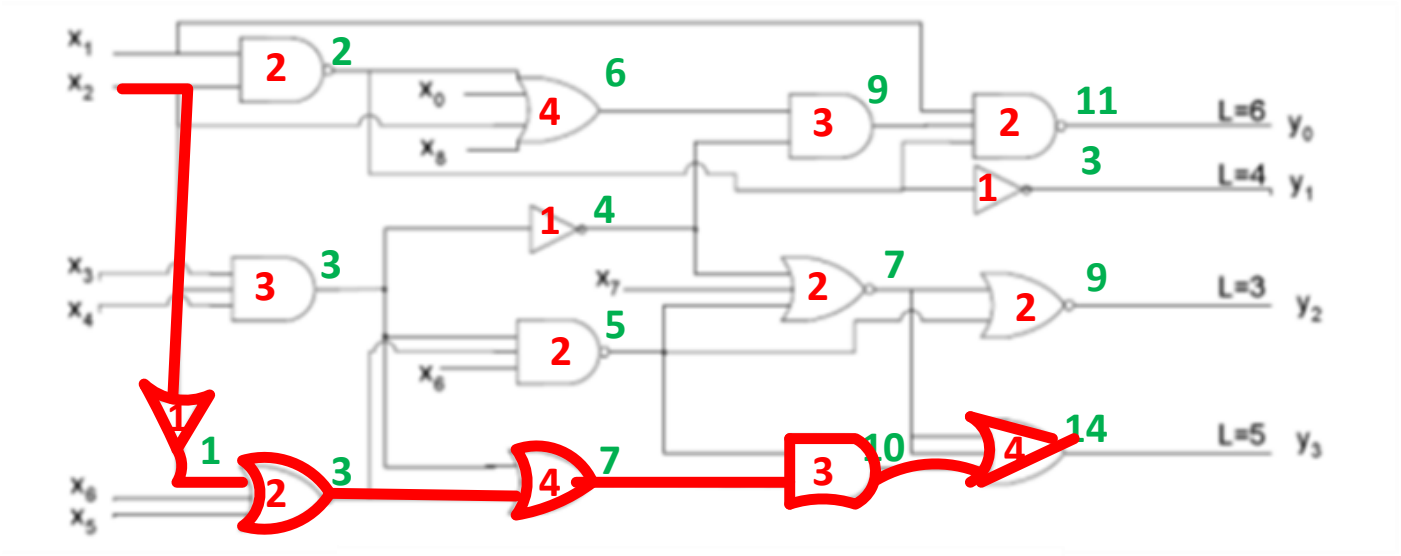
CSE320 Fall 2014

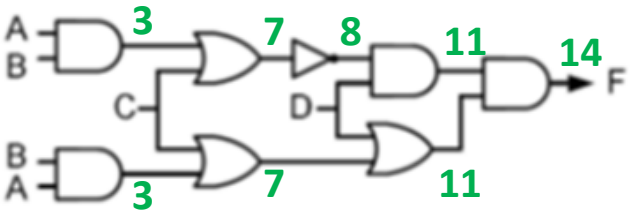
Homework #3
Quiz in Lecture on Mon 10/2/14 - No Make-up Quizzes!

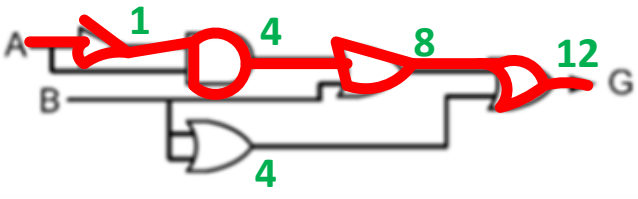
1. Calculate the Critical path through each of your gate networks for the previous problems and the ones below given the following delays.

Gate	Timing
AND	3ns
OR	4ns
NOT	1ns
NAND	2ns
NOR	2ns

Gate	Timing
XOR	2ns
Multiplexor (any size)	3ns
Decoder	5ns
Encoder	4ns







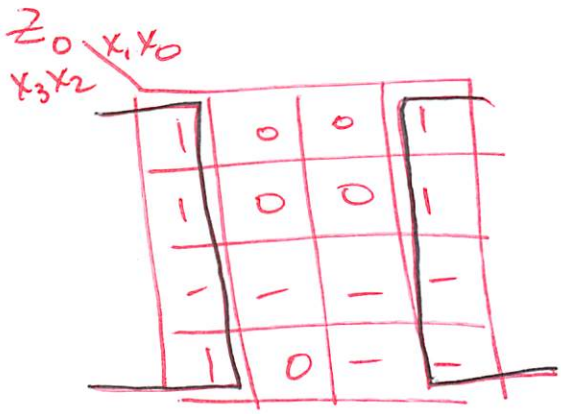
The whole network is on the critical path

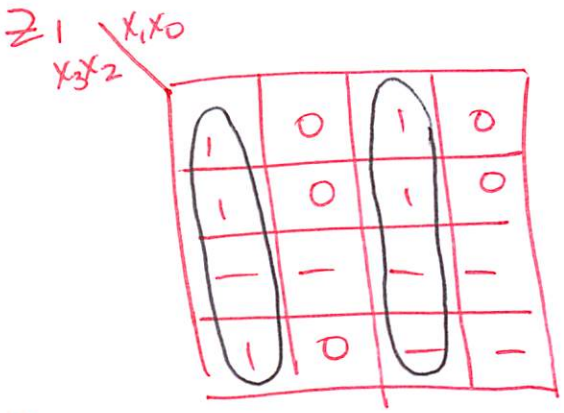
2. Design a black box that converts a decimal digit (0-9) in binary as (X_3,X_2,X_1,X_0) to its 4-bit Excess-3 encoding (Z_3,Z_2,Z_1,Z_0) (http://en.wikipedia.org/wiki/Excess-3). [From Wikipedia] In Excess-3, numbers are represented as decimal digits, and each digit is represented by four bits as the digit value + 3 (the "excess" amount). Eg. 7 becomes 10, 9 becomes 12.
   a. Complete the truth table: input (X_3,X_2,X_1,X_0), output (Z_3,Z_2,Z_1,Z_0))
   b. Write the minterm expressions for each output.
   c. Implement all Z outputs with a single 4-bit Decoder and OR gates.

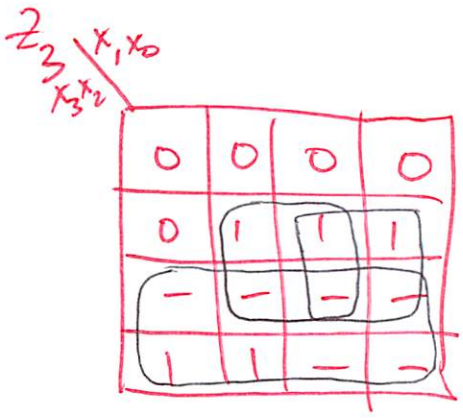
X3	X2	X1	X0	Z3	Z2	Z1	Z0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	-	-	-	-
1	0	1	1	-	-	-	-
1	1	0	0	-	-	-	-
1	1	0	1	-	-	-	-
1	1	1	0	-	-	-	-
1	1	1	1	-	-	-	-

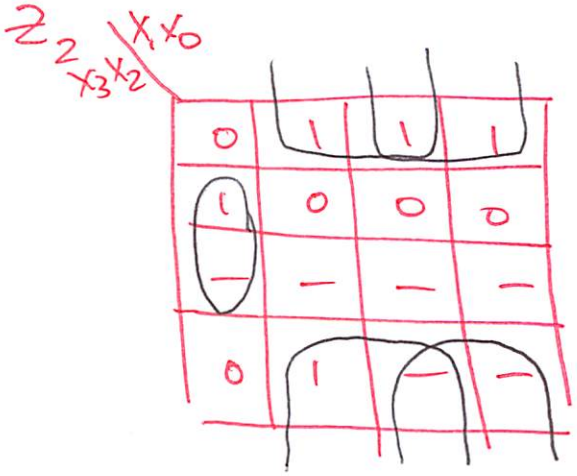
Using k-maps the expressions for each output is:

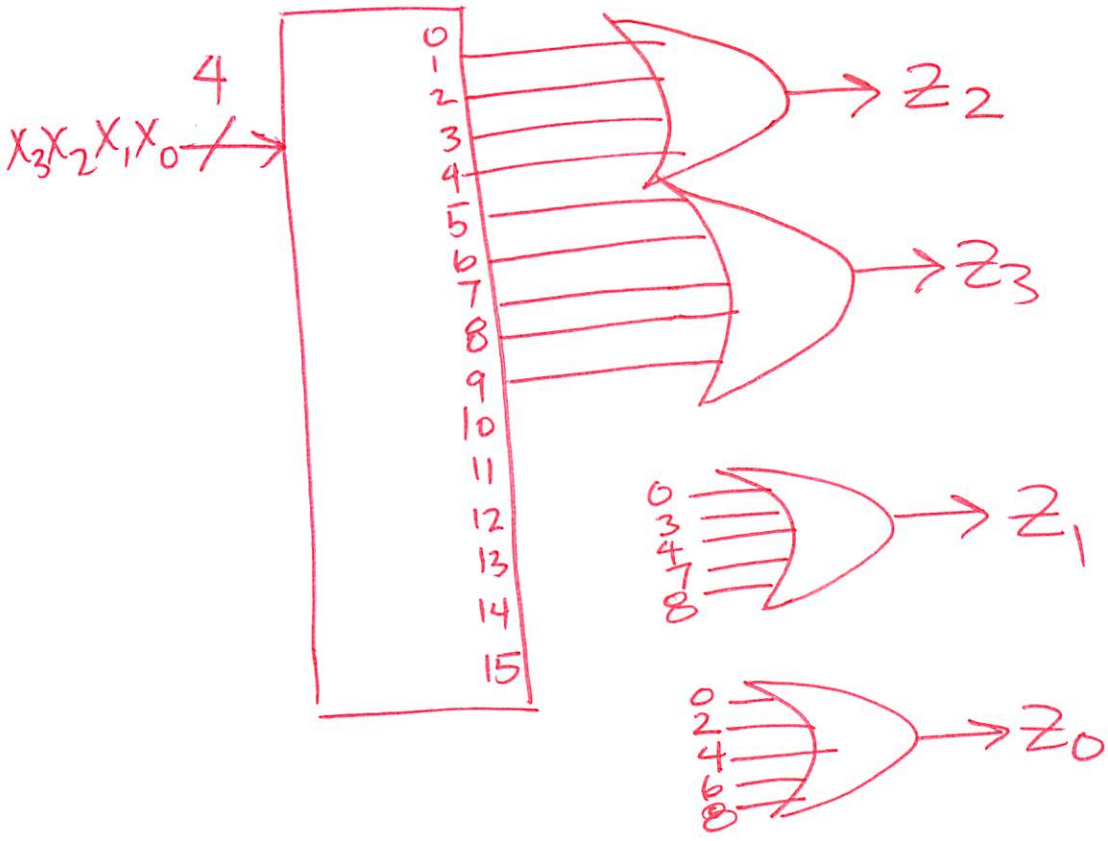
Z0 = X1`
Z1 = X1'X0' + X1X0
Z2 = X2X1'X0' + X2'X0 + X2'X1
Z3 = X2X0 + X2X1 + X3





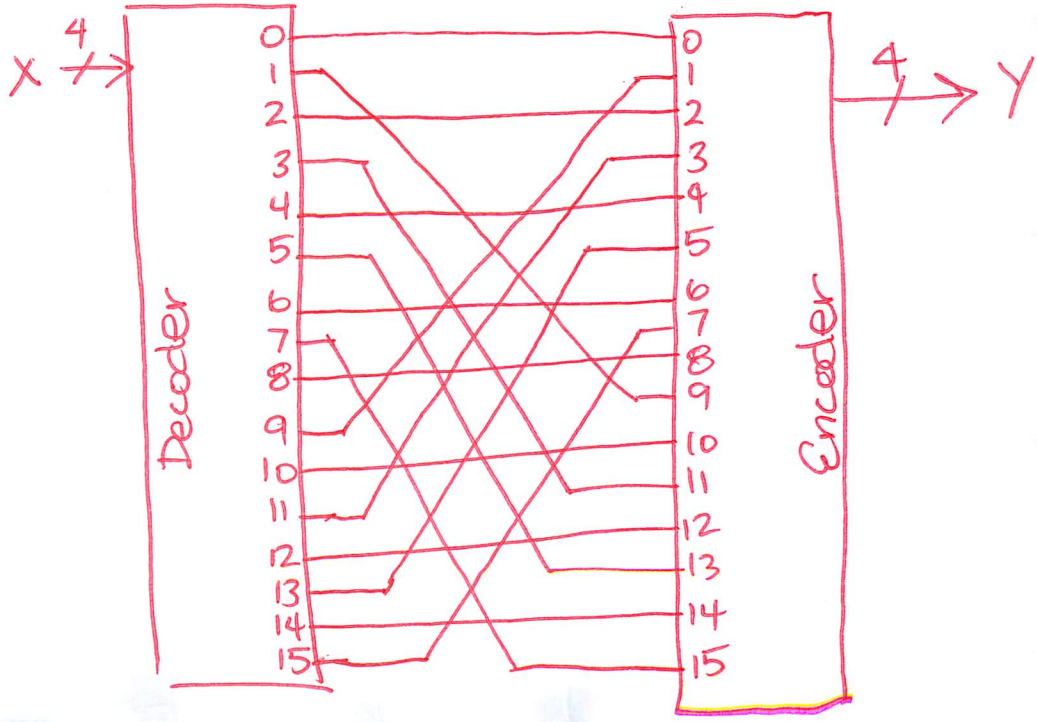





4
$X_3X_2X_1X_0$
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Z_2
Z_3
0
3
4
7
8
Z_1
0
2
4
6
8
Z_0

3. Design a black box that has a 4-bit input (X) and a 4-bit output (Y) which computes Y = (9x) % 16. Use only a decoder and an encoder.
   a. Create the truth table with decimal values.
   b. Draw a Decoder. (How many inputs/outputs does it have? How does it work?)
   c. Draw an Encoder. (How many inputs/outputs does it have? How does it work?)
   d. Connect the Decoder to the Encoder according to the truth table in (a). Mare sure to label all input and output values.

X	Y=(9X) %16
0	0
1	9
2	2
3	11
4	4
5	13
6	6
7	15
8	8
9	1
10	10
11	3
12	12
13	5
14	14
15	7



4. Design a 2-bit adder with Carry_in and Carry_out signals from logic gates.
    a. Create the truth table and Boolean logic expressions for 2-bit SUM and Carry_out signals.
    b. Implement (a) using basic logic gates (AND, OR NOT, NAND, NOR, or MUX).
    c. Implement (a) using 1-bit full adders.

The truth table would have input X1, X0, Y1, Y0 for the 2-bit adder inputs and the Carry_in bit. For output Z1, Z0, Carry_out.

There will be 32 combinations of the inputs. Each output signal when then be a Boolean expression which is a combination of the inputs.

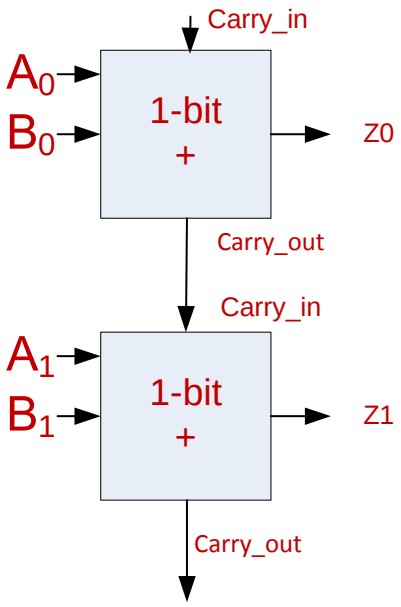
Z1 (X1, X0, Y1, Y0, Carry_in) = …

Z0 (X1, X0, Y1, Y0, Carry_in) = …

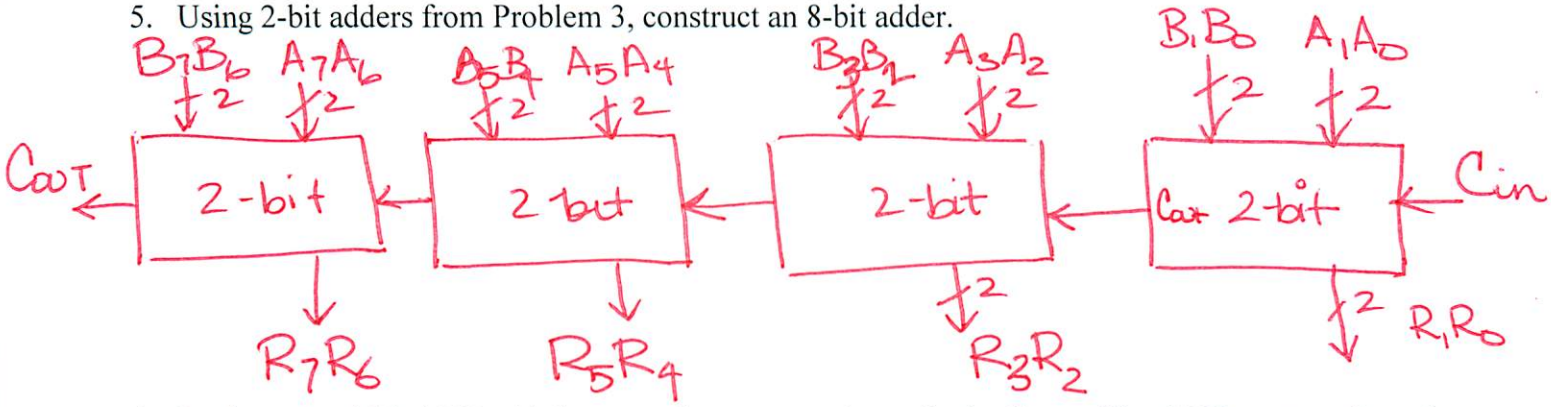
Carry_out (X1, X0, Y1, Y0, Carry_in) = …

Then each of these can be implemented with logic gates. This problem is meant to be long a tedious. The point is to illustrate that creating all operations from "scratch" can often be more expensive than building components and connecting them together.

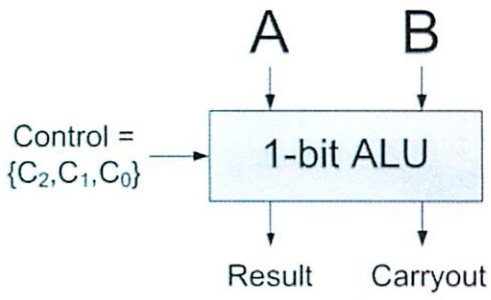
To implement with full adders:



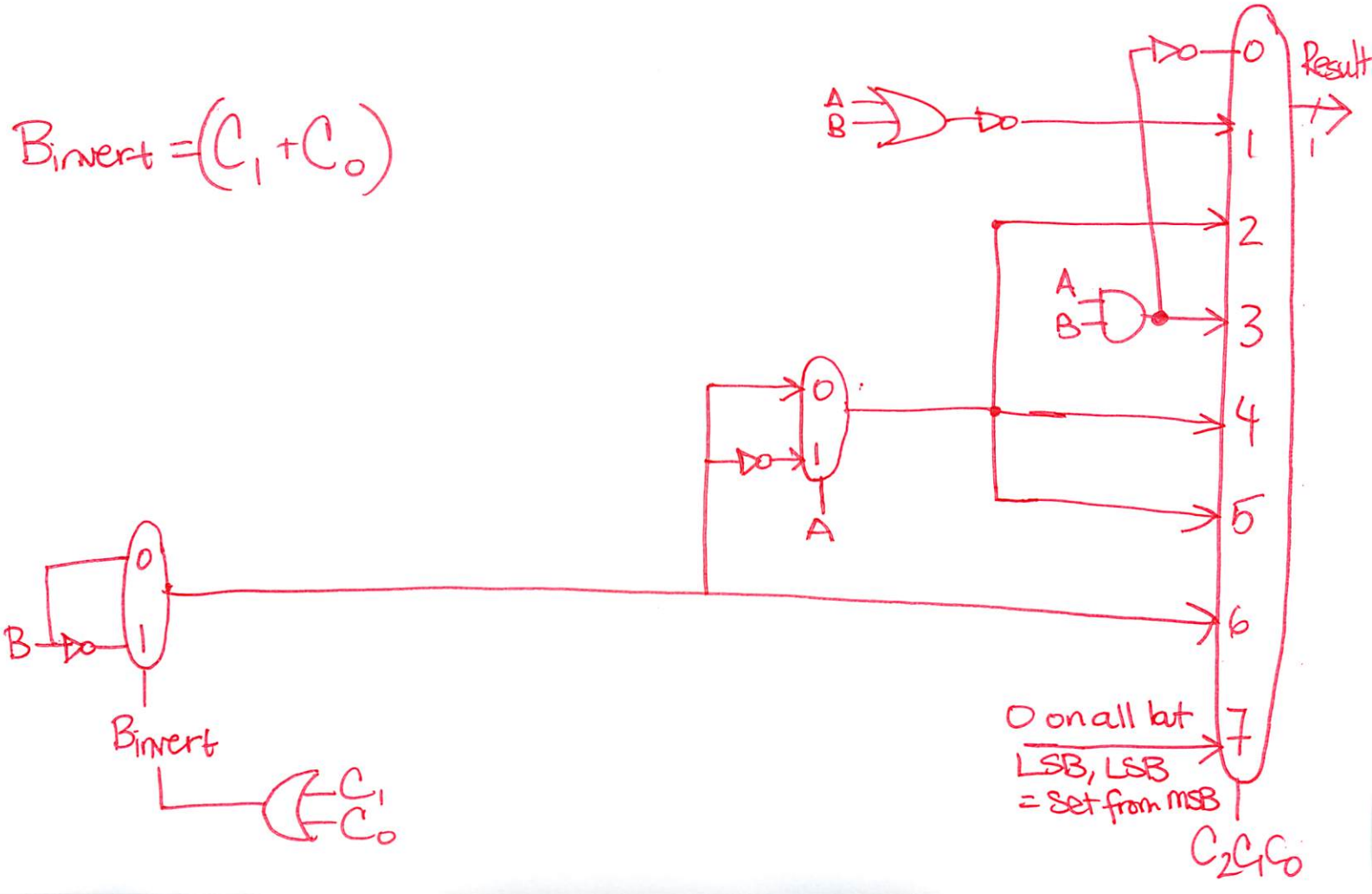
5. Using 2-bit adders from Problem 3, construct an 8-bit adder.



6. Implement a 1-bit ALU with inputs and outputs as shown in the figure. The ALU must perform the operations shown in the table according to the specified control signals. Use AND, OR, NOT and MUX gates to implement the unit. Assume only uncomplemented forms of the variables are given.

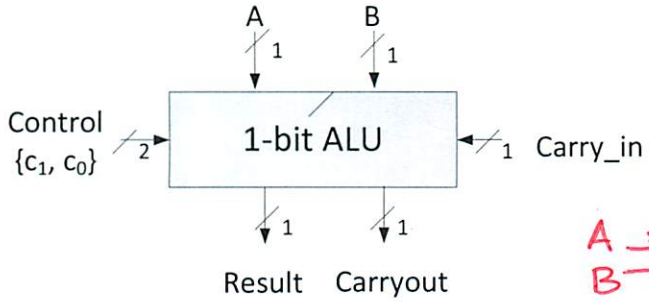


C_2	C_1	C_0	Operation	Binv
0	0	0	A NAND B	–
0	0	1	A NOR B	–
0	1	0	A XOR B	–
0	1	1	A AND B	–
1	0	0	A+B (carryout)	0
1	0	1	A – B	1
1	1	0	B'	1
1	1	1	SLT (if B < A, result = 1, else result = 0)	–

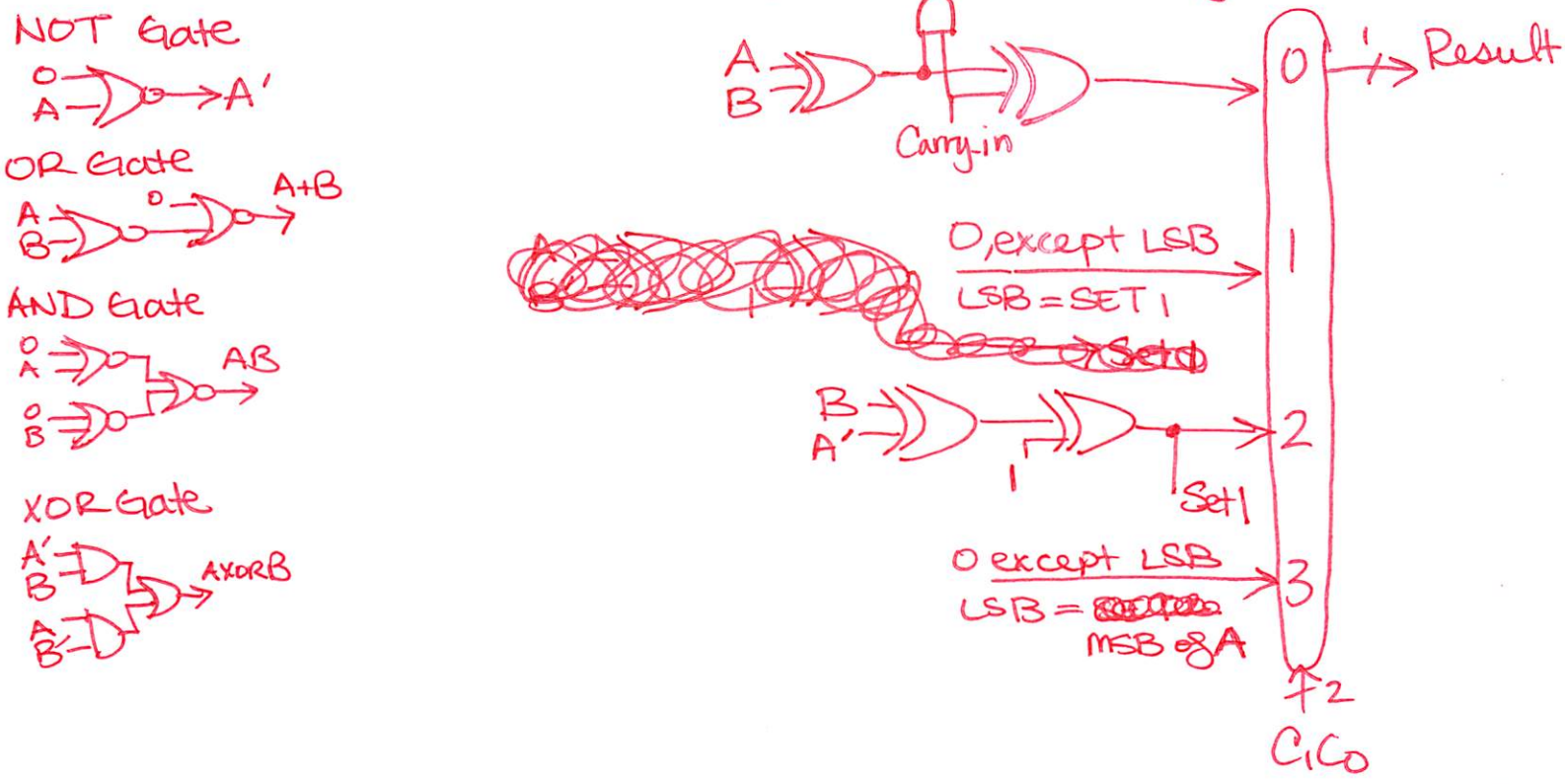


7. Implement a 1-bit ALU with inputs and outputs as shown in the figure. The ALU must perform the operations shown in the table according to the specified control signals. Use <u>one</u> MUX gate and NOR gates to implement the unit. Assume only uncomplemented forms of the variables are given.

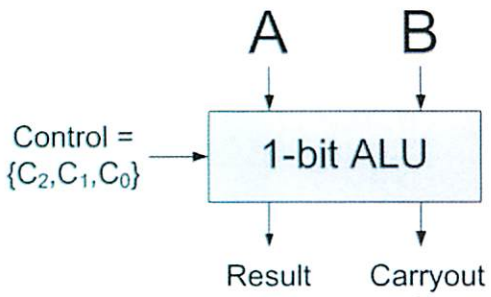
   SLTZ stands for "set on less than zero". If A is < 0, then output 1, otherwise output 0.



C_1	C_0	Operation
0	0	A + B + Carry_in
0	1	SLT
1	0	B – A
1	1	SLTZ

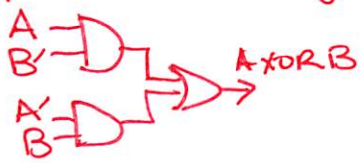


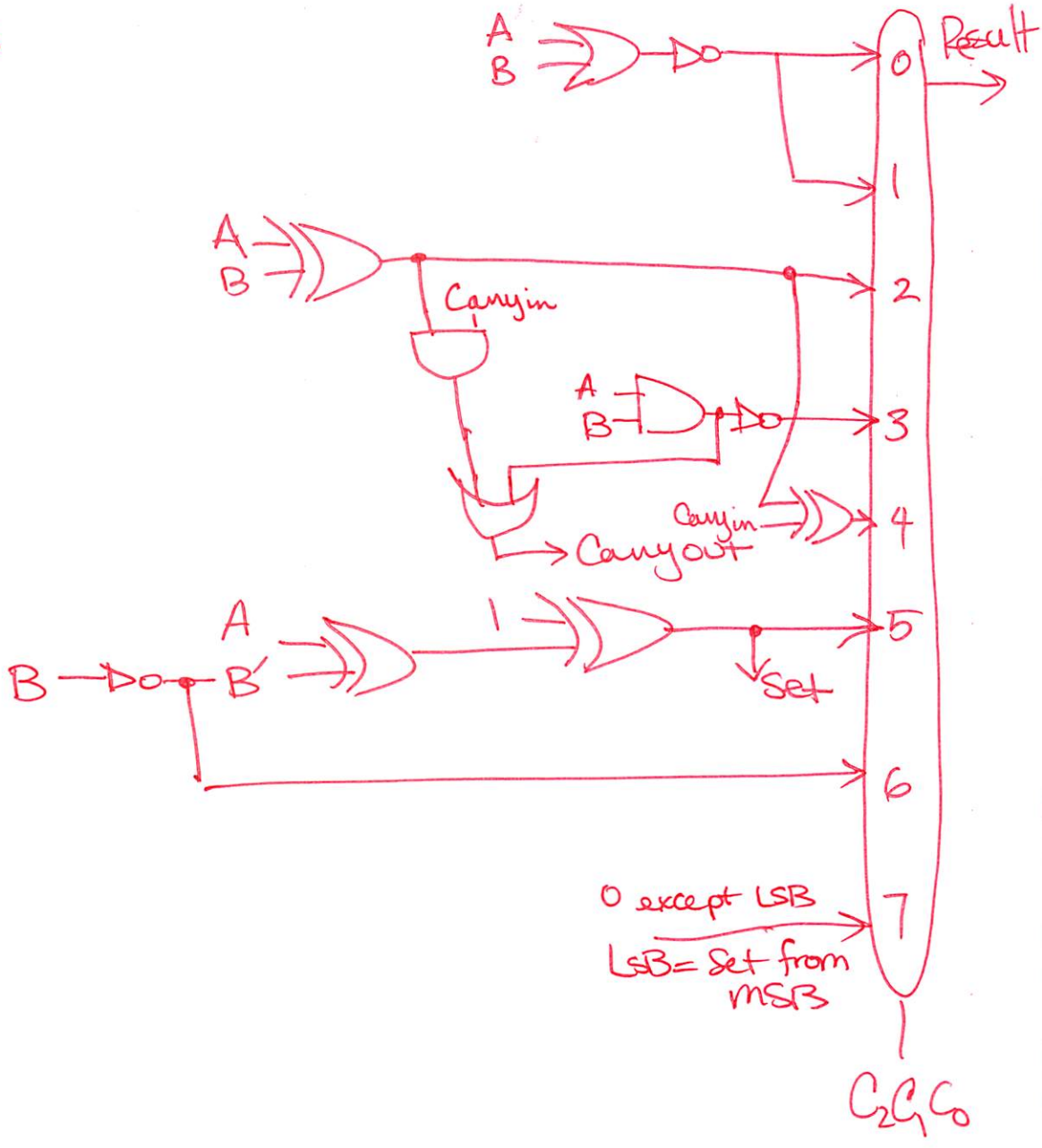
8. Implement a 1-bit ALU with inputs and outputs as shown in the figure. The ALU must perform the operations shown in the table according to the specified control signals. Use AND, OR, NOT and MUX gates to implement the unit. Assume only uncomplemented forms of the variables are given.



C_2	C_1	C_0	Operation
0	0	0	A' AND B'
0	0	1	A NOR B
0	1	0	A XOR B
0	1	1	A NAND B
1	0	0	A+B (carryout)
1	0	1	A – B
1	1	0	B'
1	1	1	SLT (if B < A, result = 1, else result = 0)

Implement XOR gate





9. Consider two new gates □ and ☺ described by the following truth table. Write the minimal Boolean expression for the gate. Show that each gate is a universal gate by building another universal set of gates. You many use the constants 0 and 1 if needed.

x	y	x□y	x☺y
0	0	0	1
0	1	1	0
1	0	0	1
1	1	0	1

The minterm expression for x□y = x'y

NOT GATE: If we set y = 1, then x□1 = x'(1) = x'
AND GATE: If we make x = x' the x□y = (x')'y = xy

The maxterm expression for x☺y = (x + y')

NOT GATE: If we set x = 0, then 0☺y = (0 + y') = y'
OR GATE: If we make y = y', then x☺(y') = (x + (y')') = (x + y)