Authors:
Sam Triest - striest@u.rochester.edu
William Wilson - wwils11@u.rochester.edu
Ryan Green - rgreen14@u.rochester.edu

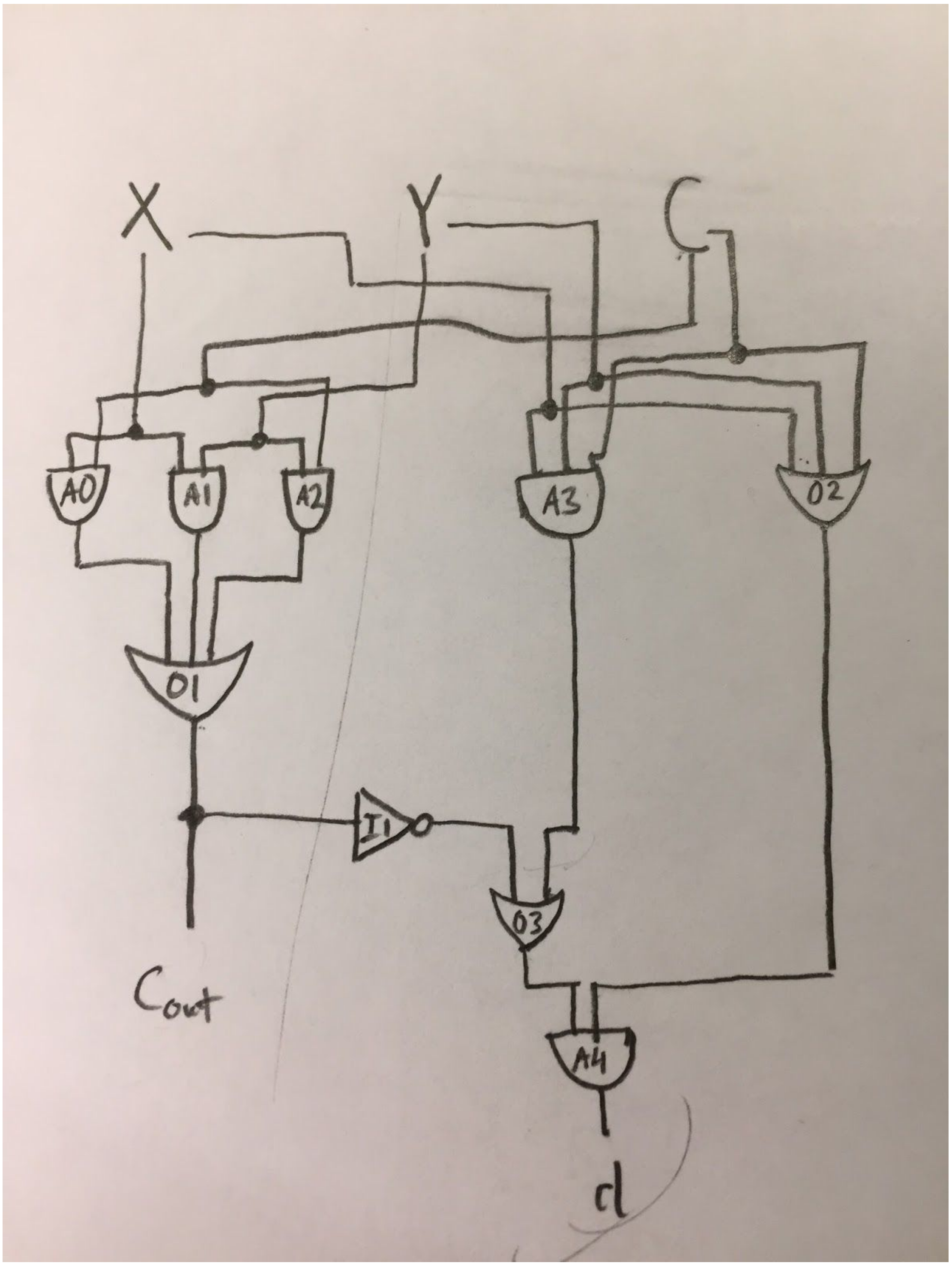
To Run Project:
Build with
make
Once you have built the project, run with
./boosim

Project Description:
This project focused on the representation of boolean circuits in C. Given the C code from the project description, we were able to represent the three circuits in the project description.

To implement NOR and NAND gates, we just attached inverters to the output of OR/AND gates. (Equivalence of the two gates shown below)

A	B	$\neg(A\ AND\ B)$	A NAND B	$\neg(A\ OR\ B)$	A NOR B
1	1	0	0	0	0
1	0	1	1	0	0
0	1	1	1	0	0
0	0	1	1	1	1

For the one bit adder circuit, we did not use the implementation in the FOCS book. Since the program can only update one gate at a time, depth becomes an irrelevant measure of circuit complexity, since the time that the circuit will fully update in is a function of the size of the circuit. As such, our one bit adder reduces the amount of gates compares to the FOCS book's version.



Equivalent boolean expressions for circuits a), b), and c).

a) $(x \wedge \neg y) \vee (y \wedge z)$

b) $\neg(\neg(x \wedge \neg y) \vee \neg(y \wedge z))$

$\equiv \neg((\neg x \vee y) \vee (\neg y \vee \neg z))$

$\equiv \neg(\neg x \vee y) \wedge \neg(\neg y \vee \neg z)$

$\equiv (x \wedge \neg y) \wedge (y \wedge z)$

$\equiv (x \wedge z) \wedge (y \wedge \neg y)$

$\equiv x \wedge z \wedge F$

$\equiv F$

c) $(x \wedge y) \vee (\neg x \wedge \neg y) \equiv x \leftrightarrow y$