Justin Ngo
Digital Logic Design
Barry Johnson
16 November 2020

Laboratory Assignment 3

Problem Statement

The task for this laboratory assignment is to design a 4-bit Carry-Lookahead Adder (CLA) and use two 4-bit CLAs to design an 8-bit Group-Ripple Adder. In the design of the 4-bit CLA circuit it is assumed that the carry generation logic should be designed using AND gates and OR gates, exclusive-OR gates can be used to form each sum bit once the carry bits are generated, and each AND gate and OR gate must have a maximum of 8 inputs.

Design Solution

Derivation of the Boolean equations for each carry bit in my 4-bit CLA:

$$c_{i+1} = g_i + g_i p_i$$

Truth Table for a Propagate, Generate, and Carry-out

c_i	a_i	b_i	g_i	p_i	c_{i+1}
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	0	1	1
1	1	1	1	1	1

The equation and truth table above show a carry-out if generate one, carry in if propagate one.

Equations for carry bits 1-4:

$$g_i = a_i\, b_i \qquad p_i = a_i + b_i \qquad c_{i+1} = g_i + c_i\, p_i$$

$$c_1 = g_0 + c_0p_0$$

$$c_2 = g_1 + c_1p_1 = g_1 + (g_0 + c_0p_0)p_1 = g_1 + g_0p_1 + c_0p_0p_1$$

$$c_3 = g_2 + c_2p_2 = g_2 + (g_1 + g_0p_1 + c_0p_0p_1)p_2 = g_2 + g_1p_2 + g_0p_1p_2 + c_0p_0p_1p_2$$

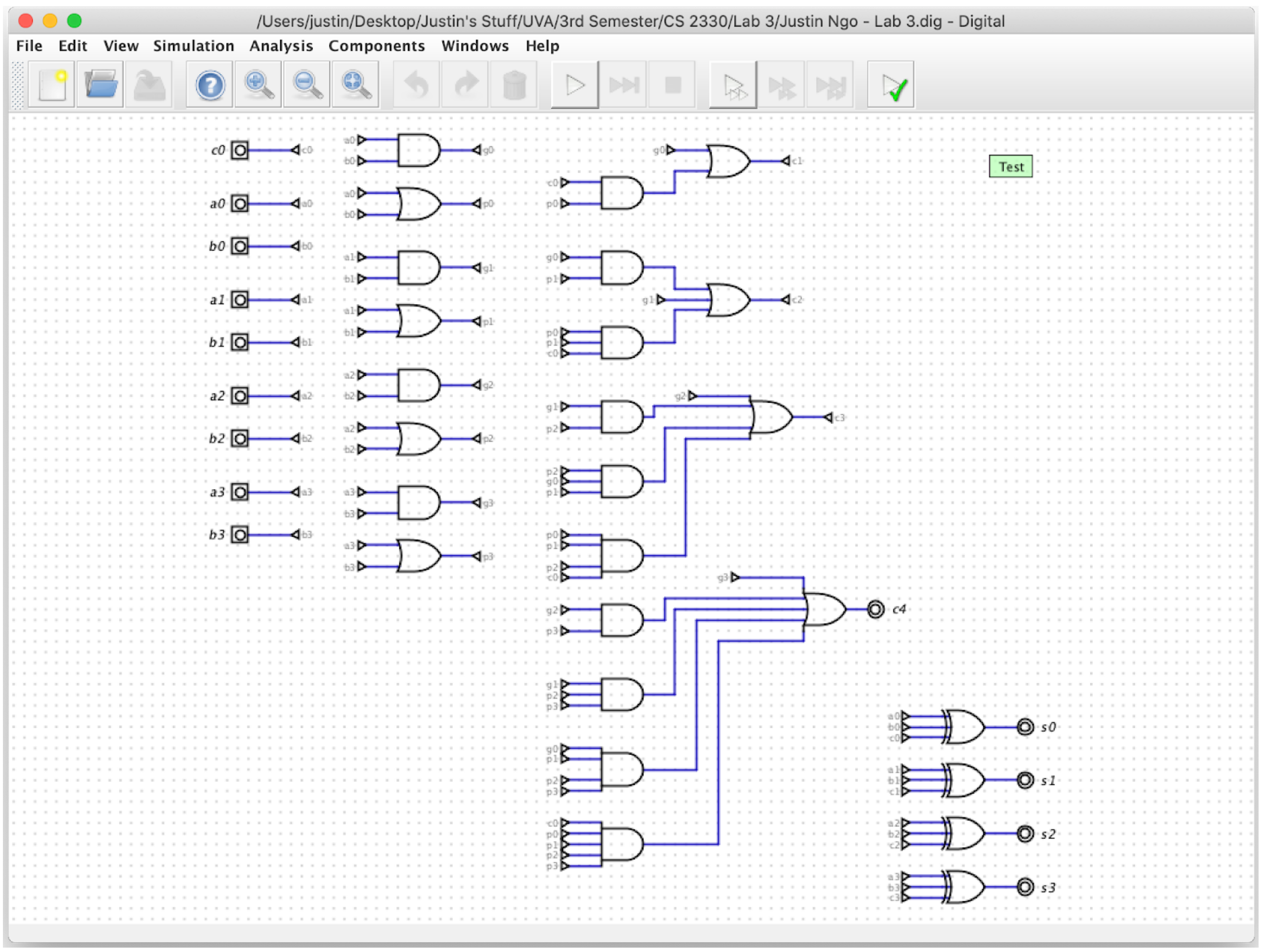
$$c_4 = g_3 + c_3p_3 = g_3 + (g_2 + g_1p_2 + g_0p_1p_2 + c_0p_0p_1p_2)p_3 = g_3 + g_2p_3 + g_1p_2p_3 + g_0p_1p_2p_3 + c_0p_0p_1p_2p_3$$

These equations for g_i, p_i, and c_{i+1} allowed me to develop an expression for the carry-in at each adder cell in terms of the inputs a_i and b_i (for all i and in terms of the propagate and generate signals) as well as the initial carry-in c_0.

An example step by step expansion of the c_i equation for c_3 :

$$c_{i+1} = g_i + c_i\, p_i$$
$$c_3 = g_2 + p_2\, c_2$$
$$c_3 = g_2 + p_2(g_1 + p_1\, c_1)$$
$$c_3 = g_2 + p_2\, g_1 + p_2\, p_1 g_0 + p_2\, p_1\, p_0\, c_0$$

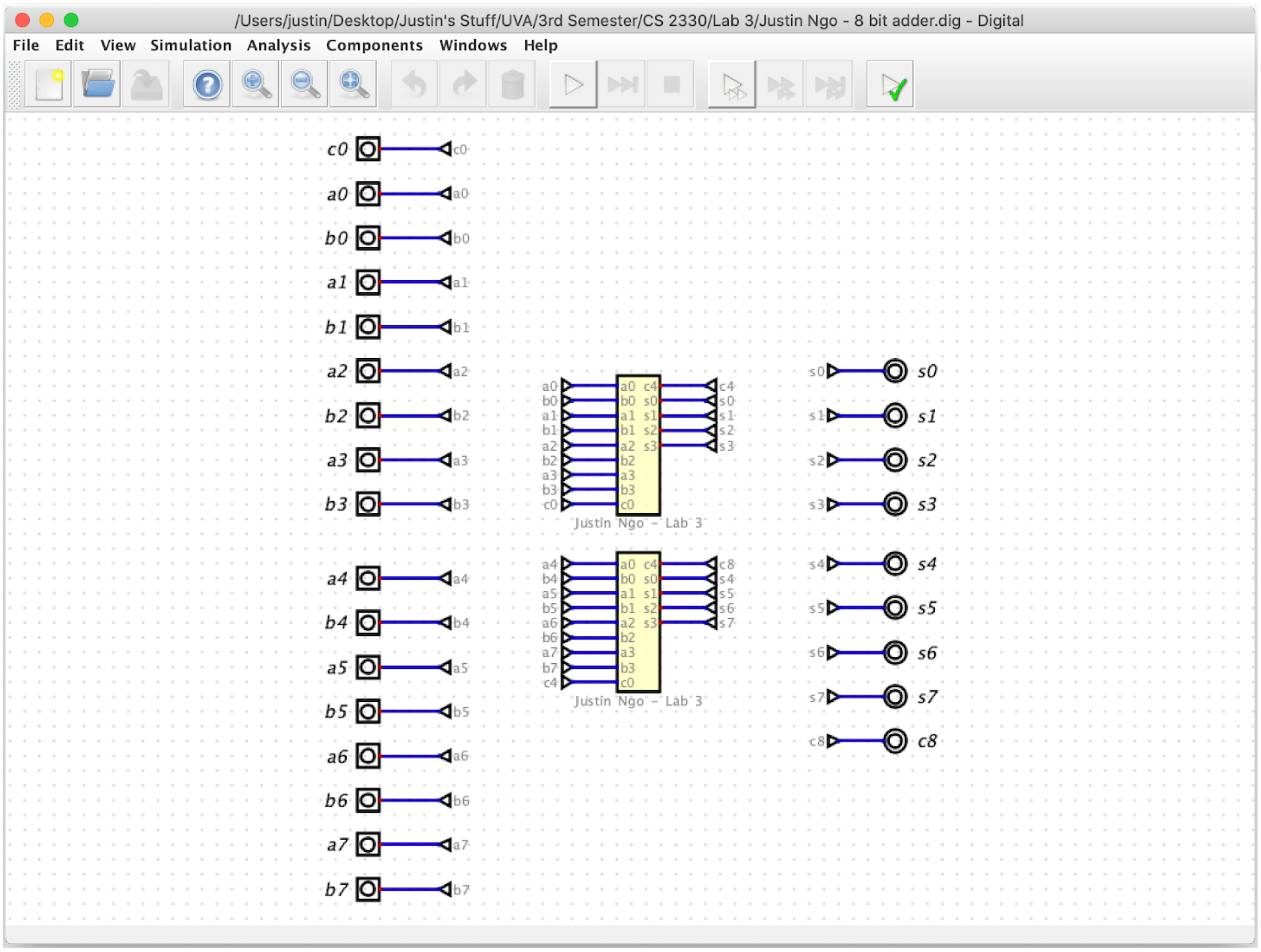
Gate-level Schematic of my 4-bit CLA Design:



/Users/justin/Desktop/Justin's Stuff/UVA/3rd Semester/CS 2330/Lab 3/Justin Ngo - Lab 3.dig - Digital
File Edit View Simulation Analysis Components Windows Help
Test
c0
a0
b0
a1
b1
a2
b2
a3
b3
c4
s0
s1
s2
s3

In my 4-bit CLA design, carry bits c1, c2, c3, and c4 are calculated in parallel. I designed my circuit following the equations I derived in the previous section. The carry generation logic is designed using AND gates and OR gates starting with 4 input bits for a and b and then according to the equations derived for c1, c2, c3, and c4. Then, I used exclusive-OR gates to form each sum bit s0-s3 as well as c4 once the carry bits were generated.

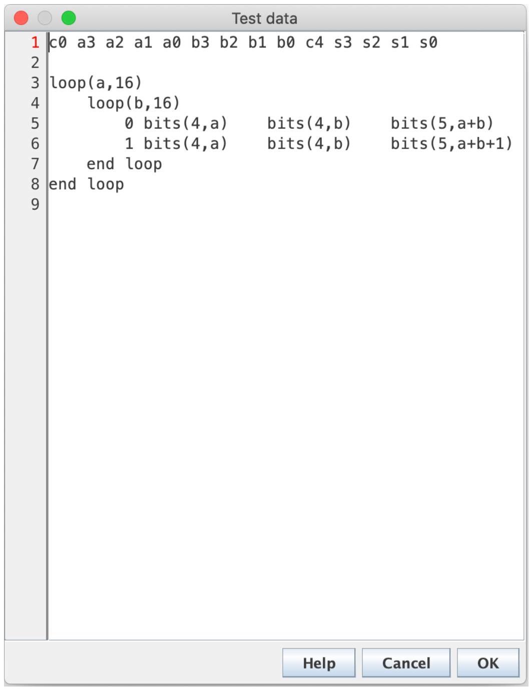
Block Diagram of my 8-bit Group-Ripple Adder:


/Users/justin/Desktop/Justin's Stuff/UVA/3rd Semester/CS 2330/Lab 3/Justin Ngo - 8 bit adder.dig - Digital
File Edit View Simulation Analysis Components Windows Help
c0
a0
b0
a1
b1
a2
b2
a3
b3
a4
b4
a5
b5
a6
b6
a7
b7
Justin Ngo - Lab 3
Justin Ngo - Lab 3
s0
s1
s2
s3
s4
s5
s6
s7
c8

In my 8-bit group-ripple adder I use two of the 4-bit CLAs I designed to build an 8-bit group-ripple adder. The group ripple adder takes in 8 bits a0-a7 and b0-b7 as well as a carry bit c0. c4 is the carry bit which is generated when adding in the first 8-bit ripple adder, and c4 from the first 4-bit CLA goes into c0 of the second 4-bit CLA in order to create the 8-bit group ripple adder. There are 9 output sum bits s0-s7 as well as the output carry bit c8.

Test Results

Test Feature:

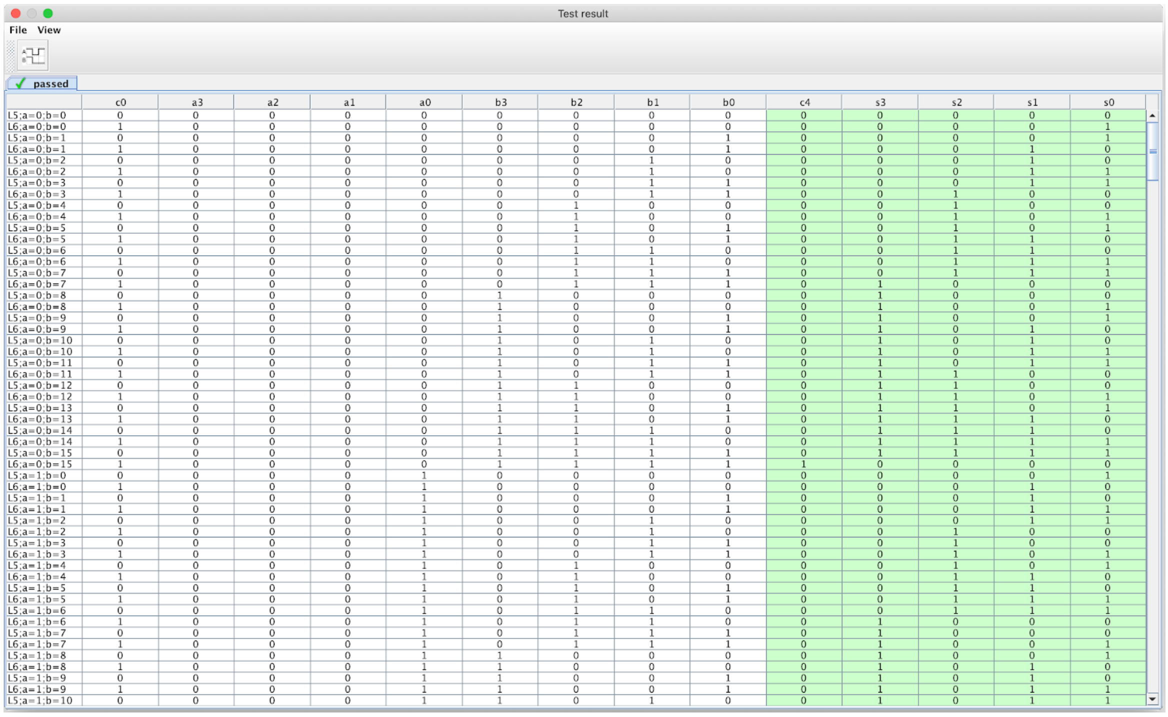


```
c0 a3 a2 a1 a0 b3 b2 b1 b0 c4 s3 s2 s1 s0

loop(a,16)
    loop(b,16)
        0 bits(4,a)    bits(4,b)    bits(5,a+b)
        1 bits(4,a)    bits(4,b)    bits(5,a+b+1)
    end loop
end loop

```

Test Results:



	c0	a3	a2	a1	a0	b3	b2	b1	b0	c4	s3	s2	s1	s0
L5;a=0;b=0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L6;a=0;b=0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
L5;a=0;b=1	0	0	0	0	0	0	0	0	1	0	0	0	0	1
L6;a=0;b=1	1	0	0	0	0	0	0	0	1	0	0	0	1	0
L5;a=0;b=2	0	0	0	0	0	0	0	1	0	0	0	0	1	0
L6;a=0;b=2	1	0	0	0	0	0	0	1	0	0	0	0	1	1
L5;a=0;b=3	0	0	0	0	0	0	0	1	1	0	0	0	1	1
L6;a=0;b=3	1	0	0	0	0	0	0	1	1	0	0	1	0	0
L5;a=0;b=4	0	0	0	0	0	0	1	0	0	0	0	1	0	0
L6;a=0;b=4	1	0	0	0	0	0	1	0	0	0	0	1	0	1
L5;a=0;b=5	0	0	0	0	0	0	1	0	1	0	0	1	0	1
L6;a=0;b=5	1	0	0	0	0	0	1	0	1	0	0	1	1	0
L5;a=0;b=6	0	0	0	0	0	0	1	1	0	0	0	1	1	0
L6;a=0;b=6	1	0	0	0	0	0	1	1	0	0	0	1	1	1
L5;a=0;b=7	0	0	0	0	0	0	1	1	1	0	0	1	1	1
L6;a=0;b=7	1	0	0	0	0	0	1	1	1	0	1	0	0	0
L5;a=0;b=8	0	0	0	0	0	1	0	0	0	0	1	0	0	0
L6;a=0;b=8	1	0	0	0	0	1	0	0	0	0	1	0	0	1
L5;a=0;b=9	0	0	0	0	0	1	0	0	1	0	1	0	0	1
L6;a=0;b=9	1	0	0	0	0	1	0	0	1	0	1	0	1	0
L5;a=0;b=10	0	0	0	0	0	1	0	1	0	0	1	0	1	0
L6;a=0;b=10	1	0	0	0	0	1	0	1	0	0	1	0	1	1
L5;a=0;b=11	0	0	0	0	0	1	0	1	1	0	1	0	1	1
L6;a=0;b=11	1	0	0	0	0	1	0	1	1	0	1	1	0	0
L5;a=0;b=12	0	0	0	0	0	1	1	0	0	0	1	1	0	0
L6;a=0;b=12	1	0	0	0	0	1	1	0	0	0	1	1	0	1
L5;a=0;b=13	0	0	0	0	0	1	1	0	1	0	1	1	0	1
L6;a=0;b=13	1	0	0	0	0	1	1	0	1	0	1	1	1	0
L5;a=0;b=14	0	0	0	0	0	1	1	1	0	0	1	1	1	0
L6;a=0;b=14	1	0	0	0	0	1	1	1	0	0	1	1	1	1
L5;a=0;b=15	0	0	0	0	0	1	1	1	1	0	1	1	1	1
L6;a=0;b=15	1	0	0	0	0	1	1	1	1	1	0	0	0	0
L5;a=1;b=0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
L6;a=1;b=0	1	0	0	0	1	0	0	0	0	0	0	0	1	0
L5;a=1;b=1	0	0	0	0	1	0	0	0	1	0	0	0	1	0
L6;a=1;b=1	1	0	0	0	1	0	0	0	1	0	0	0	1	1
L5;a=1;b=2	0	0	0	0	1	0	0	1	0	0	0	0	1	1
L6;a=1;b=2	1	0	0	0	1	0	0	1	0	0	0	1	0	0
L5;a=1;b=3	0	0	0	0	1	0	0	1	1	0	0	1	0	0
L6;a=1;b=3	1	0	0	0	1	0	0	1	1	0	0	1	0	1
L5;a=1;b=4	0	0	0	0	1	0	1	0	0	0	0	1	0	1
L6;a=1;b=4	1	0	0	0	1	0	1	0	0	0	0	1	1	0
L5;a=1;b=5	0	0	0	0	1	0	1	0	1	0	0	1	1	0
L6;a=1;b=5	1	0	0	0	1	0	1	0	1	0	0	1	1	1
L5;a=1;b=6	0	0	0	0	1	0	1	1	0	0	0	1	1	1
L6;a=1;b=6	1	0	0	0	1	0	1	1	0	0	1	0	0	0
L5;a=1;b=7	0	0	0	0	1	0	1	1	1	0	1	0	0	0
L6;a=1;b=7	1	0	0	0	1	0	1	1	1	0	1	0	0	1
L5;a=1;b=8	0	0	0	0	1	1	0	0	0	0	1	0	0	1
L6;a=1;b=8	1	0	0	0	1	1	0	0	0	0	1	0	1	0
L5;a=1;b=9	0	0	0	0	1	1	0	0	1	0	1	0	1	0
L6;a=1;b=9	1	0	0	0	1	1	0	0	1	0	1	0	1	1
L5;a=1;b=10	0	0	0	0	1	1	0	1	0	0	1	0	1	1

The test feature shows the test case I made which performs an 100% test for my 4-bit CLA. All possible 512 input combinations are applied to the circuit. The test case checks that the output of the circuit is correct. The first line in the test data lists the input and output signals. Below this, the input values applied and the output values checked are specified in the loops in a row, as in a truth table. Variables a and b in my test data loops are each traversed from 0 to 15. The respective values of a and b are then assigned to inputs 'a[n]' and 'b[n]'. Then it is checked whether the circuit outputs the value a+b. Furthermore, it is checked again with the carry bit set, in which case a+b+1 must result. The test results of the test I created on my circuit show that my circuit works and passes for all of the test cases, producing the correct sum bits s0-s3 and c4.

Summary and Conclusions

I began the laboratory assignment by reading and studying about CLAs in the textbook and lectures for Modules #6-3 and #6-4. I researched the concepts and details of CLAs, as well as the use of multiple n-bit CLAs to create an overall m-bit adder using the group-ripple adder concept. I learned how to derive the equations for a CLA and implement my own 4-bit CLA circuit based on the derived equations. I then learned how to make an 8-bit group ripple adder using the 4-bit CLA I designed by connecting the output of the fourth carry bit from the first CLA to the first input of the second CLA. In addition, I learned how to use the testing capability included in the digital simulator to efficiently test all the possible inputs and outputs of my circuit design. A problem I encountered in the laboratory assignment was figuring out how to use the test capability included in the digital simulator. I did not know how to write the loops for the test cases. I overcame this problem by reading through the documentation of the digital simulator to figure out how to format and write the test cases for my circuit properly.