Dakota Lester

CSE 341 – Project 2 Write-up

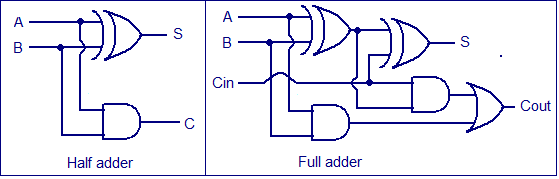
11/30/2015

Dr. Kris Schindler

**Table of Contents**

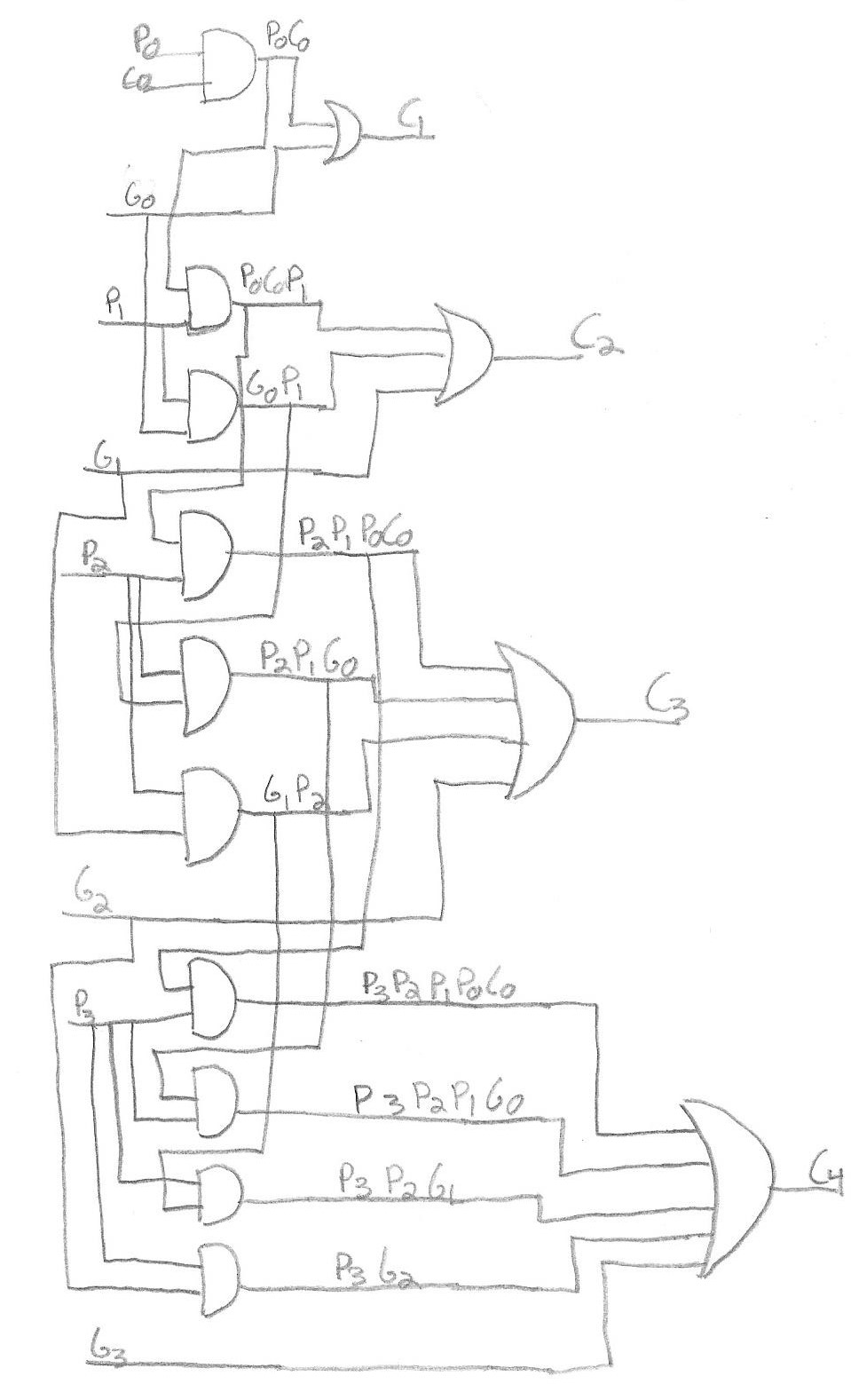
1. Circuit Diagram for Ripple Carry Adder
2. Circuit Diagram for Carry Look Ahead Adder
3. Critical Path of Ripple Carry Adder (Citation in Works Cited on last page)
4. Critical Path of Carry Look Ahead Adder (Citation in Works Cited on last page)
5. Critical Path Delay of Both Adders
6. Gate cost of the both adders
7. Simulation results under 10 data sets for Ripple Carry Adder and Carry Look Ahead Adder under zero gate delay model
8. Temporal delay under the Ripple Carry Adder with five examples showing how it is in the results
9. Average delay under 5,000 random inputs for Ripple Carry Adder and Carry Look Ahead adder
   1. How this was obtained?
   2. How does this compare to the critical path delay?

**I. Circuit Diagram for Ripple Carry Adder**



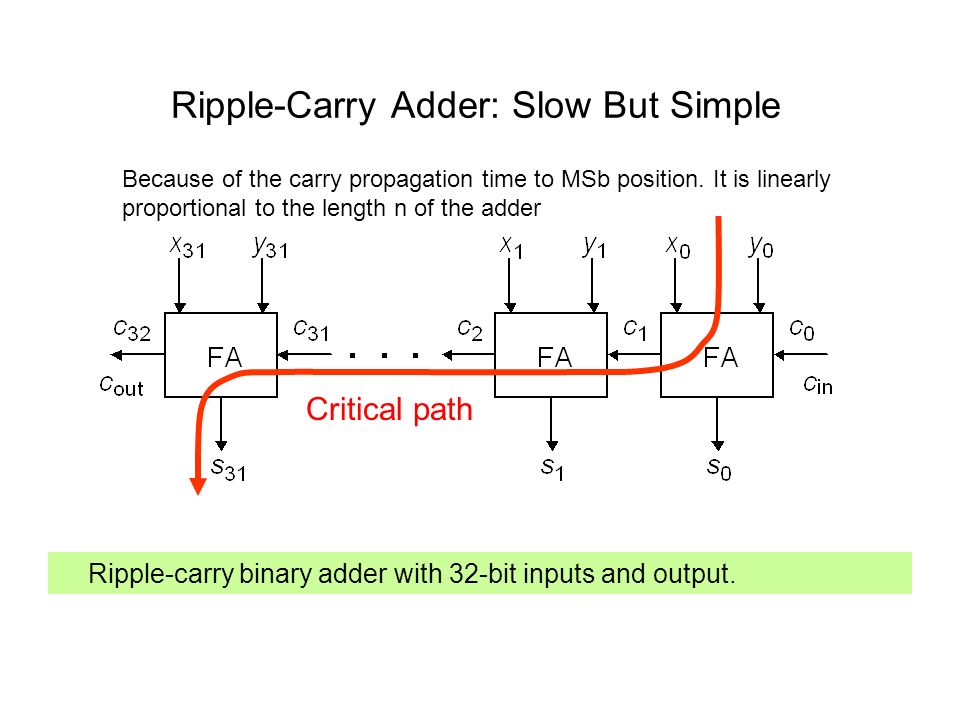
**II. Circuit Diagram for Carry Look Ahead Adder**

This diagram shown below illustrates a four-bit Carry Look Ahead adder which multiply this same picture by four and the same can be used to illustrate a sixteen-bit Carry Look Ahead adder.

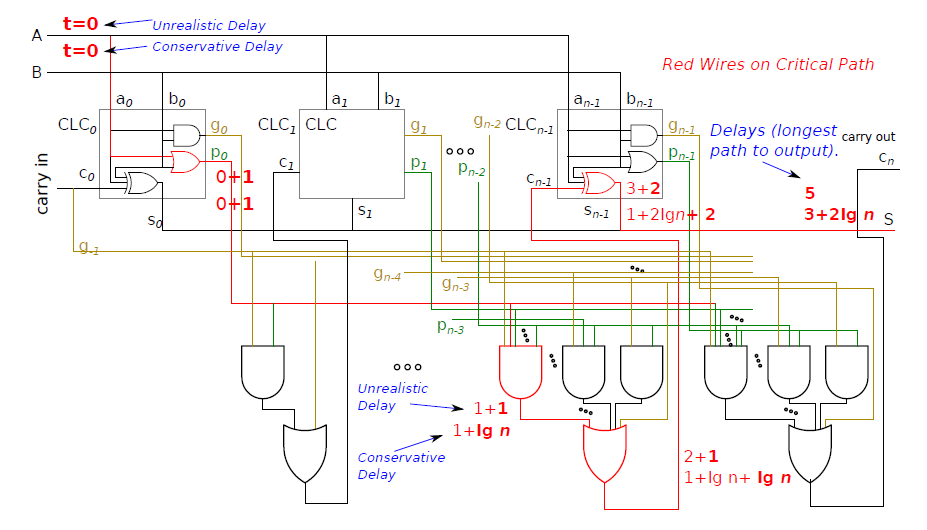


**III. Critical Path for Ripple Carry Adder**

Take the picture shown below which illustrates a thirty-two-bit ripple carry adder and divide by two and the same result is shown for a sixteen-bit adder.



**IV. Critical Path for Carry Look Ahead adder**



**V. Critical Path delay for both adders**

In order to obtain the critical path delay for the ripple carry adder the general formula used to obtain this is the amount of levels of logic times the number of bits the adder is. In this case sense this is a sixteen-bit ripple carry adder the critical path delay would be (3n = 3(16) = 48 nanoseconds) delay (3). To obtain the critical path delay for a carry look ahead adder sense there is two levels of logic in the partial full adders and two levels of logic in the carry look ahead unit, the time would be (“1 + 2 + 2 + 2 + 1 = 8”,) nanosecond delay (3).

**VI. Gate cost of both adders**

The gate cost for the ripple and carry look ahead adders are found by counting the amount of gates used for the entire circuit. For the ripple carry adder that is one full adder of gates which is five times the amount of full adders (sixteen) which is (5\*16 = 90 gates) and for the ripple carry adder. For the carry look ahead that is the amount of gates for one partial full adder, which is fourteen and multiply that by four to get the gate cost for a sixteen-bit adder and you get (14\*4 = 56) as your gate cost for the carry look ahead adder.

**VII. Simulation results under both adders with zero gate delay model**

All of these inputs were obtained through a random number generator created in C++ that I will included with the files required.

Here are the results for the ripple carry adder under a zero gate delay model:

0 ,a= x, b= x, c=x, s= x, cout=x

100 ,a=46987, b=40720, c=1, s=22172, cout=1

200 ,a=23853, b=58119, c=0, s=16436, cout=1

300 ,a=36605, b=58019, c=0, s=29088, cout=1

400 ,a=15283, b= 7262, c=1, s=22546, cout=0

500 ,a=36143, b=46195, c=0, s=16802, cout=1

600 ,a=48199, b=21648, c=1, s= 4312, cout=1

700 ,a=15171, b=44976, c=1, s=60148, cout=0

800 ,a=46613, b=39006, c=0, s=20083, cout=1

900 ,a=38971, b=20060, c=1, s=59032, cout=0

1000 ,a=49163, b= 1185, c=0, s=50348, cout=0

1100 ,a=52465, b=43708, c=0, s=30637, cout=1

1200 ,a=21799, b= 2025, c=1, s=23825, cout=0

1300 ,a=42871, b=34167, c=1, s=11503, cout=1

1400 ,a=27323, b=25802, c=1, s=53126, cout=0

1500 ,a=57242, b=61946, c=0, s=53652, cout=1

1600 ,a=31990, b=16497, c=0, s=48487, cout=0

1700 ,a= 5657, b=31669, c=0, s=37326, cout=0

1800 ,a=39843, b=54634, c=0, s=28941, cout=1

1900 ,a=59443, b=23606, c=0, s=17513, cout=1

2000 ,a= 5094, b=49121, c=0, s=54215, cout=0

2100 ,a=35212, b=31586, c=0, s= 1262, cout=1

2200 ,a=36050, b=53385, c=0, s=23899, cout=1

2300 ,a= 1360, b= 2608, c=1, s= 3969, cout=0

2400 ,a=24689, b= 6283, c=1, s=30973, cout=0

2500 ,a=55280, b=63526, c=1, s=53271, cout=1

2600 ,a=38418, b= 1868, c=1, s=40287, cout=0

2700 ,a=19557, b=53878, c=0, s= 7899, cout=1

2800 ,a=45671, b= 73, c=0, s=45744, cout=0

2900 ,a=17144, b=59516, c=0, s=11124, cout=1

3000 ,a=32324, b=40962, c=1, s= 7751, cout=1

3100 ,a=20578, b= 6175, c=0, s=26753, cout=0

3200 ,a=52541, b=42225, c=1, s=29231, cout=1

3300 ,a=58417, b=19937, c=0, s=12818, cout=1

3400 ,a= 4812, b=44626, c=1, s=49439, cout=0

3500 ,a=47009, b= 6258, c=1, s=53268, cout=0

3600 ,a=57505, b=21029, c=0, s=12998, cout=1

3700 ,a=14498, b=40586, c=0, s=55084, cout=0

3800 ,a=49512, b=62609, c=1, s=46586, cout=1

3900 ,a=45512, b= 9753, c=1, s=55266, cout=0

4000 ,a=41471, b=18430, c=1, s=59902, cout=0

4100 ,a=16550, b=39008, c=0, s=55558, cout=0

4200 ,a= 4032, b= 2365, c=0, s= 6397, cout=0

4300 ,a=12324, b=32670, c=1, s=44995, cout=0

4400 ,a=27688, b=13834, c=0, s=41522, cout=0

4500 ,a=25687, b=60844, c=0, s=20995, cout=1

4600 ,a=63565, b=24701, c=1, s=22731, cout=1

4700 ,a=33311, b=39199, c=1, s= 6975, cout=1

4800 ,a=56935, b=65063, c=1, s=56463, cout=1

4900 ,a=10632, b=16927, c=0, s=27559, cout=0

5000 ,a=23845, b=58398, c=0, s=16707, cout=1

5100 ,a=54372, b= 4949, c=1, s=59322, cout=0

5200 ,a=21075, b= 4517, c=0, s=25592, cout=0

5300 ,a=30003, b=63193, c=1, s=27661, cout=1

5400 ,a=35220, b=20881, c=1, s=56102, cout=0

5500 ,a=61416, b=22921, c=1, s=18802, cout=1

5600 ,a=47519, b=16486, c=0, s=64005, cout=0

5700 ,a=31002, b=26149, c=1, s=57152, cout=0

5800 ,a=31424, b=13084, c=0, s=44508, cout=0

5900 ,a=24455, b= 68, c=0, s=24523, cout=0

6000 ,a=27239, b= 265, c=0, s=27504, cout=0

6100 ,a=24806, b=54637, c=1, s=13908, cout=1

6200 ,a=61381, b= 5712, c=1, s= 1558, cout=1

6300 ,a= 2209, b=12067, c=0, s=14276, cout=0

6400 ,a=22060, b=23640, c=0, s=45700, cout=0

6500 ,a=55746, b=15056, c=1, s= 5267, cout=1

6600 ,a=10275, b=62575, c=1, s= 7315, cout=1

6700 ,a=35858, b= 4394, c=0, s=40252, cout=0

6800 ,a=30639, b=31354, c=0, s=61993, cout=0

6900 ,a= 1300, b=55809, c=0, s=57109, cout=0

7000 ,a=49960, b=59400, c=1, s=43825, cout=1

7100 ,a=21482, b=60558, c=0, s=16504, cout=1

7200 ,a=44306, b=28291, c=1, s= 7062, cout=1

7300 ,a= 1647, b= 6853, c=0, s= 8500, cout=0

7400 ,a=22181, b=28913, c=0, s=51094, cout=0

7500 ,a=39949, b=14659, c=1, s=54609, cout=0

7600 ,a=10638, b= 1287, c=0, s=11925, cout=0

7700 ,a=20517, b=37145, c=0, s=57662, cout=0

7800 ,a=10193, b= 2248, c=0, s=12441, cout=0

7900 ,a=35657, b=40972, c=1, s=11094, cout=1

8000 ,a=61189, b=20932, c=0, s=16585, cout=1

8100 ,a=63338, b=18767, c=0, s=16569, cout=1

8200 ,a=50125, b=63073, c=0, s=47662, cout=1

8300 ,a=65328, b=41072, c=1, s=40865, cout=1

8400 ,a=46246, b=63253, c=0, s=43963, cout=1

8500 ,a=27157, b= 9555, c=0, s=36712, cout=0

8600 ,a= 3940, b= 1009, c=1, s= 4950, cout=0

8700 ,a=34370, b=17063, c=0, s=51433, cout=0

8800 ,a=45547, b= 3608, c=0, s=49155, cout=0

8900 ,a=18077, b=39266, c=1, s=57344, cout=0

9000 ,a=22769, b= 6807, c=1, s=29577, cout=0

9100 ,a=63213, b= 146, c=0, s=63359, cout=0

9200 ,a=24215, b=26623, c=1, s=50839, cout=0

9300 ,a=13509, b=21951, c=1, s=35461, cout=0

9400 ,a=56008, b=44549, c=1, s=35022, cout=1

9500 ,a=57421, b=48058, c=0, s=39943, cout=1

9600 ,a=49845, b=47534, c=1, s=31844, cout=1

9700 ,a= 1668, b=58257, c=0, s=59925, cout=0

9800 ,a=28060, b=33804, c=1, s=61865, cout=0

9900 ,a=33052, b=51881, c=1, s=19398, cout=1

10000 ,a=32665, b= 4650, c=0, s=37315, cout=0

Here are the results for the simulation under the carry look ahead zero gate delay model:

0 a = x, b = x, cin = x, s = x, cout = x

100 a = 1105, b = 62469, cin = 0, s = 63574, cout = 0

200 a = 60603, b = 41672, cin = 1, s = 36740, cout = 1

300 a = 47244, b = 59187, cin = 1, s = 40896, cout = 1

400 a = 2270, b = 47545, cin = 1, s = 49816, cout = 0

500 a = 51755, b = 15278, cin = 1, s = 1498, cout = 1

600 a = 27389, b = 24505, cin = 0, s = 51894, cout = 0

700 a = 54913, b = 23059, cin = 1, s = 12437, cout = 1

800 a = 389, b = 51540, cin = 0, s = 51929, cout = 0

900 a = 32368, b = 45389, cin = 1, s = 12222, cout = 1

1000 a = 29056, b = 47060, cin = 0, s = 10580, cout = 1

1100 a = 23494, b = 24517, cin = 0, s = 48011, cout = 0

1200 a = 10418, b = 15121, cin = 0, s = 25539, cout = 0

1300 a = 12005, b = 62365, cin = 0, s = 8834, cout = 1

1400 a = 15722, b = 60171, cin = 1, s = 10358, cout = 1

1500 a = 59733, b = 18278, cin = 0, s = 12475, cout = 1

1600 a = 3343, b = 22020, cin = 1, s = 25364, cout = 0

1700 a = 30015, b = 6933, cin = 0, s = 36948, cout = 0

1800 a = 57855, b = 2858, cin = 0, s = 60713, cout = 0

1900 a = 6907, b = 11578, cin = 1, s = 18486, cout = 0

2000 a = 14726, b = 40635, cin = 1, s = 55362, cout = 0

2100 a = 33468, b = 40481, cin = 1, s = 8414, cout = 1

2200 a = 21032, b = 50899, cin = 0, s = 6395, cout = 1

2300 a = 50269, b = 39256, cin = 1, s = 23990, cout = 1

2400 a = 45045, b = 31330, cin = 1, s = 10840, cout = 1

2500 a = 63718, b = 1880, cin = 1, s = 63, cout = 1

2600 a = 4022, b = 47111, cin = 1, s = 51134, cout = 0

2700 a = 38832, b = 7126, cin = 0, s = 45958, cout = 0

2800 a = 3052, b = 41333, cin = 1, s = 44386, cout = 0

2900 a = 65165, b = 24592, cin = 1, s = 24222, cout = 1

3000 a = 56314, b = 39318, cin = 0, s = 30096, cout = 1

3100 a = 47228, b = 49139, cin = 1, s = 30832, cout = 1

3200 a = 62659, b = 171, cin = 0, s = 62830, cout = 0

3300 a = 23211, b = 26792, cin = 0, s = 50003, cout = 0

3400 a = 22480, b = 1837, cin = 1, s = 24318, cout = 0

3500 a = 58273, b = 41908, cin = 1, s = 34646, cout = 1

3600 a = 42345, b = 45930, cin = 0, s = 22739, cout = 1

3700 a = 24788, b = 14762, cin = 0, s = 39550, cout = 0

3800 a = 60517, b = 64167, cin = 0, s = 59148, cout = 1

3900 a = 29104, b = 59332, cin = 0, s = 22900, cout = 1

4000 a = 55621, b = 21998, cin = 1, s = 12084, cout = 1

4100 a = 29126, b = 3690, cin = 0, s = 32816, cout = 0

4200 a = 1223, b = 38237, cin = 1, s = 39461, cout = 0

4300 a = 37996, b = 37800, cin = 0, s = 10260, cout = 1

4400 a = 24840, b = 36633, cin = 1, s = 61474, cout = 0

4500 a = 43015, b = 24906, cin = 1, s = 2386, cout = 1

4600 a = 34958, b = 43603, cin = 0, s = 13025, cout = 1

4700 a = 27660, b = 44744, cin = 0, s = 6868, cout = 1

4800 a = 50193, b = 35261, cin = 1, s = 19919, cout = 1

4900 a = 44059, b = 64365, cin = 1, s = 42889, cout = 1

5000 a = 62517, b = 26338, cin = 0, s = 23319, cout = 1

5100 a = 26646, b = 55464, cin = 0, s = 16574, cout = 1

5200 a = 39914, b = 28576, cin = 0, s = 2954, cout = 1

5300 a = 29706, b = 42924, cin = 0, s = 7094, cout = 1

5400 a = 22642, b = 2228, cin = 1, s = 24871, cout = 0

5500 a = 17415, b = 40780, cin = 0, s = 58195, cout = 0

5600 a = 54097, b = 5738, cin = 1, s = 59836, cout = 0

5700 a = 43061, b = 9751, cin = 1, s = 52813, cout = 0

5800 a = 46787, b = 59944, cin = 0, s = 41195, cout = 1

5900 a = 21033, b = 10355, cin = 1, s = 31389, cout = 0

6000 a = 30962, b = 2872, cin = 0, s = 33834, cout = 0

6100 a = 62890, b = 29519, cin = 0, s = 26873, cout = 1

6200 a = 24045, b = 45785, cin = 0, s = 4294, cout = 1

6300 a = 46141, b = 51843, cin = 0, s = 32448, cout = 1

6400 a = 59685, b = 4485, cin = 1, s = 64171, cout = 0

6500 a = 16215, b = 21901, cin = 1, s = 38117, cout = 0

6600 a = 44003, b = 5998, cin = 1, s = 50002, cout = 0

6700 a = 45394, b = 25411, cin = 0, s = 5269, cout = 1

6800 a = 45178, b = 2198, cin = 0, s = 47376, cout = 0

6900 a = 32926, b = 4047, cin = 1, s = 36974, cout = 0

7000 a = 15040, b = 30546, cin = 0, s = 45586, cout = 0

7100 a = 4308, b = 23436, cin = 1, s = 27745, cout = 0

7200 a = 40112, b = 23833, cin = 0, s = 63945, cout = 0

7300 a = 15309, b = 4438, cin = 0, s = 19747, cout = 0

7400 a = 57077, b = 36012, cin = 1, s = 27554, cout = 1

7500 a = 2962, b = 28579, cin = 0, s = 31541, cout = 0

7600 a = 35979, b = 2582, cin = 0, s = 38561, cout = 0

7700 a = 4735, b = 47976, cin = 1, s = 52712, cout = 0

7800 a = 53241, b = 3970, cin = 0, s = 57211, cout = 0

7900 a = 44396, b = 32433, cin = 1, s = 11294, cout = 1

8000 a = 4017, b = 23825, cin = 1, s = 27843, cout = 0

8100 a = 3935, b = 23669, cin = 0, s = 27604, cout = 0

8200 a = 14188, b = 63781, cin = 1, s = 12434, cout = 1

8300 a = 227, b = 9090, cin = 0, s = 9317, cout = 0

8400 a = 38109, b = 42520, cin = 0, s = 15093, cout = 1

8500 a = 10476, b = 21834, cin = 1, s = 32311, cout = 0

8600 a = 27449, b = 57814, cin = 1, s = 19728, cout = 1

8700 a = 50419, b = 38901, cin = 1, s = 23785, cout = 1

8800 a = 28801, b = 2958, cin = 0, s = 31759, cout = 0

8900 a = 55733, b = 42890, cin = 1, s = 33088, cout = 1

9000 a = 12248, b = 18796, cin = 0, s = 31044, cout = 0

9100 a = 3774, b = 3547, cin = 1, s = 7322, cout = 0

9200 a = 56160, b = 13272, cin = 0, s = 3896, cout = 1

9300 a = 1307, b = 13499, cin = 1, s = 14807, cout = 0

9400 a = 7501, b = 51608, cin = 1, s = 59110, cout = 0

9500 a = 30523, b = 38436, cin = 1, s = 3424, cout = 1

9600 a = 12124, b = 349, cin = 1, s = 12474, cout = 0

9700 a = 42659, b = 22657, cin = 0, s = 65316, cout = 0

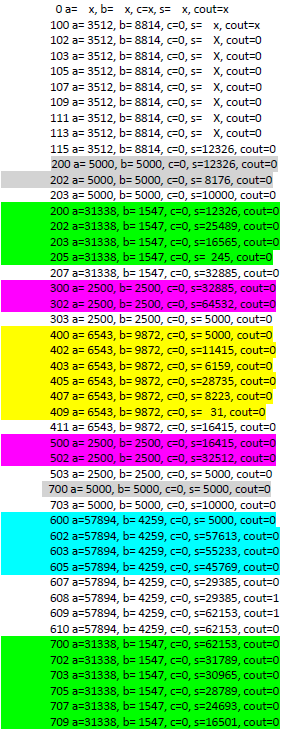
9800 a = 51171, b = 51458, cin = 1, s = 37094, cout = 1

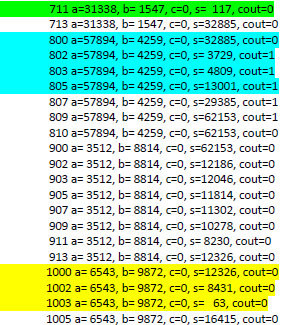
9900 a = 35541, b = 37191, cin = 1, s = 7197, cout = 1

10000 a = 52344, b = 49439, cin = 1, s = 36248, cout = 1

**VIII. Temporal Delay**

The way that I showed temporal delay is by setting the same input under the ripple carry adder unit gate delay model at different times to show that the resulting sum would change based on the delay, as shown below. The highlighted portions indicate the matching inputs to show temporal delay.





**IX. Average Delay**

In order to obtain the random five thousand inputs to calculate the average delay under both the ripple carry adder and carry look ahead adder under the unit gate delay model I created a C++ program that in a ‘for’ loop, looped 5000 times and randomized ‘a’, ‘b’, and ‘cin’ where ‘a’ and ‘b’ were randomized from zero to seventy thousand while ‘cin’ was just randomized to either a one or zero sense that is the only case carry in can be. This will be submitted with the required files. To find the average delay I created another C++ program that searched the output file for the first instance of a delay and the last instance of the delay, found the difference and added it to a counter. Once the file has been completely traversed I divided the counter by five thousand and my results for ripple carry adder and carry look ahead adder are as follows. The average delay for the ripple carry adder was 6.04 nanoseconds. The average delay for the carry look ahead adder was 5.1304 or approximately 5.13 nanoseconds. In relation to my critical path delay the average delay for my ripple carry is far less then what the delay should be which is thirty-two and for the carry look ahead adder my delay is close to what the approximate delay is which is eight.

Works Cited

1. Critical Path for Ripple Carry Adder - http://images.slideplayer.com/1/252895/slides/slide\_33.jpg

2. Critical Path for Carry Look Ahead Adder - <http://www.ece.lsu.edu/ee3755/2013f/cla.pdf> (slide 11)

3. Obtain critical path delay for both adders - <http://www.cse.buffalo.edu/~kds/cse-341/Lectures/mult_div.pdf>

4. Circuit diagram for ripple carry adder - http://www.circuitstoday.com/wp-content/uploads/2012/03/half-adder-full-adder-circuit.png