ESE 3700 Project 1 Report

Garrett Kirsch

3/30/2025

Unoptimized RCA Design

Introduction:

To begin with this project, I leveraged a RCA design that I learned while TAing for CIS 2400 and later when making a RCA in System Verilog for CIS 5710. This design leverages half-adders that take in 2 bits and calculates a cout and sum from them (logic described below). I then combine the half-adders into a full-adder so that I can account for the carry-in bit.

Half-Adder:

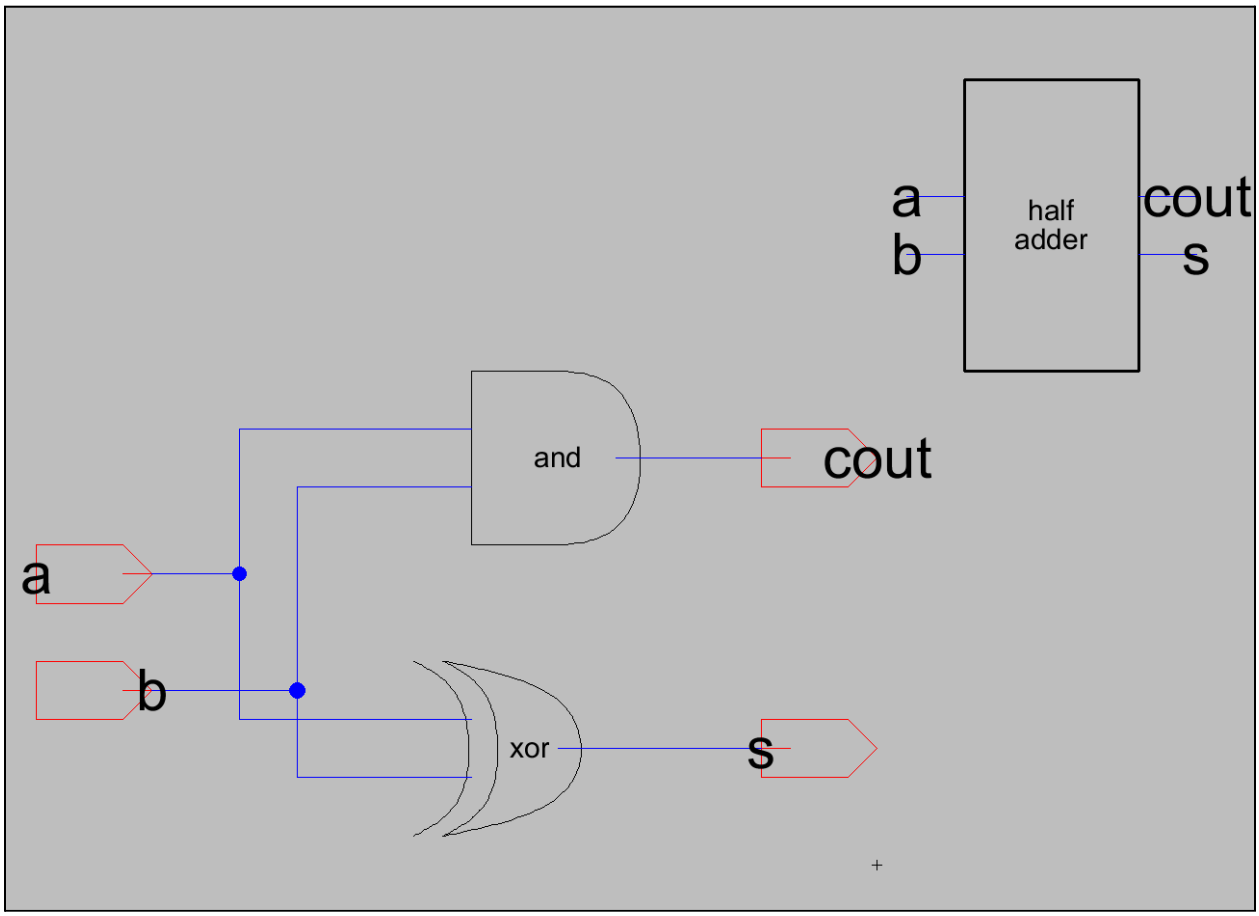

Figure 1.1: Half-Adder Schematic

The half-adder takes in two inputs, a and b, then outputs cout = a & b, sum = a ^ b. Let's look closer at the AND and XOR gates.

AND2:

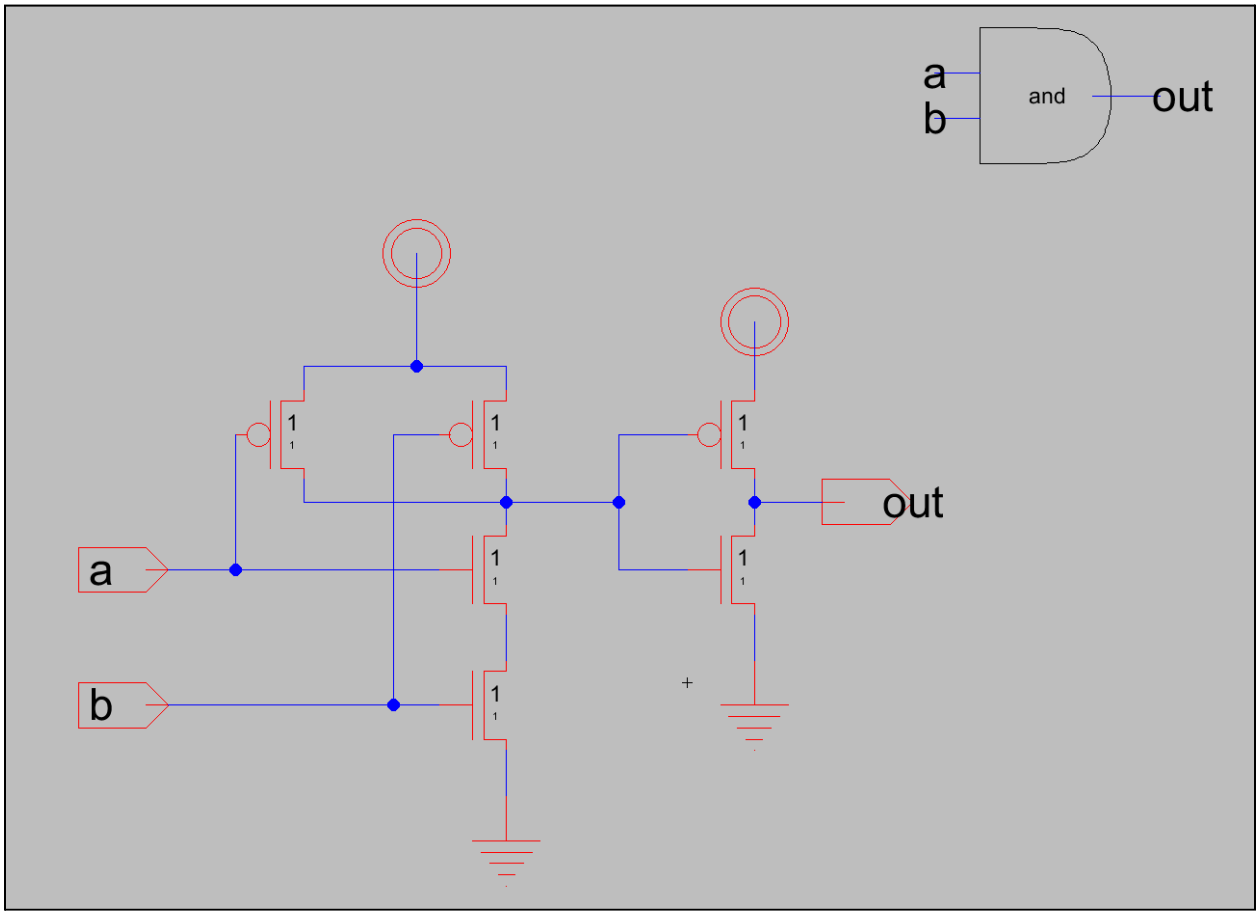

Figure 1.2: AND2 Gate Schematic

For the unoptimized design, the AND gate is made of a cascaded, min-sized NAND2 and NOT gate. We use this to calculate whether two bits would generate a carry-out.

XOR2:

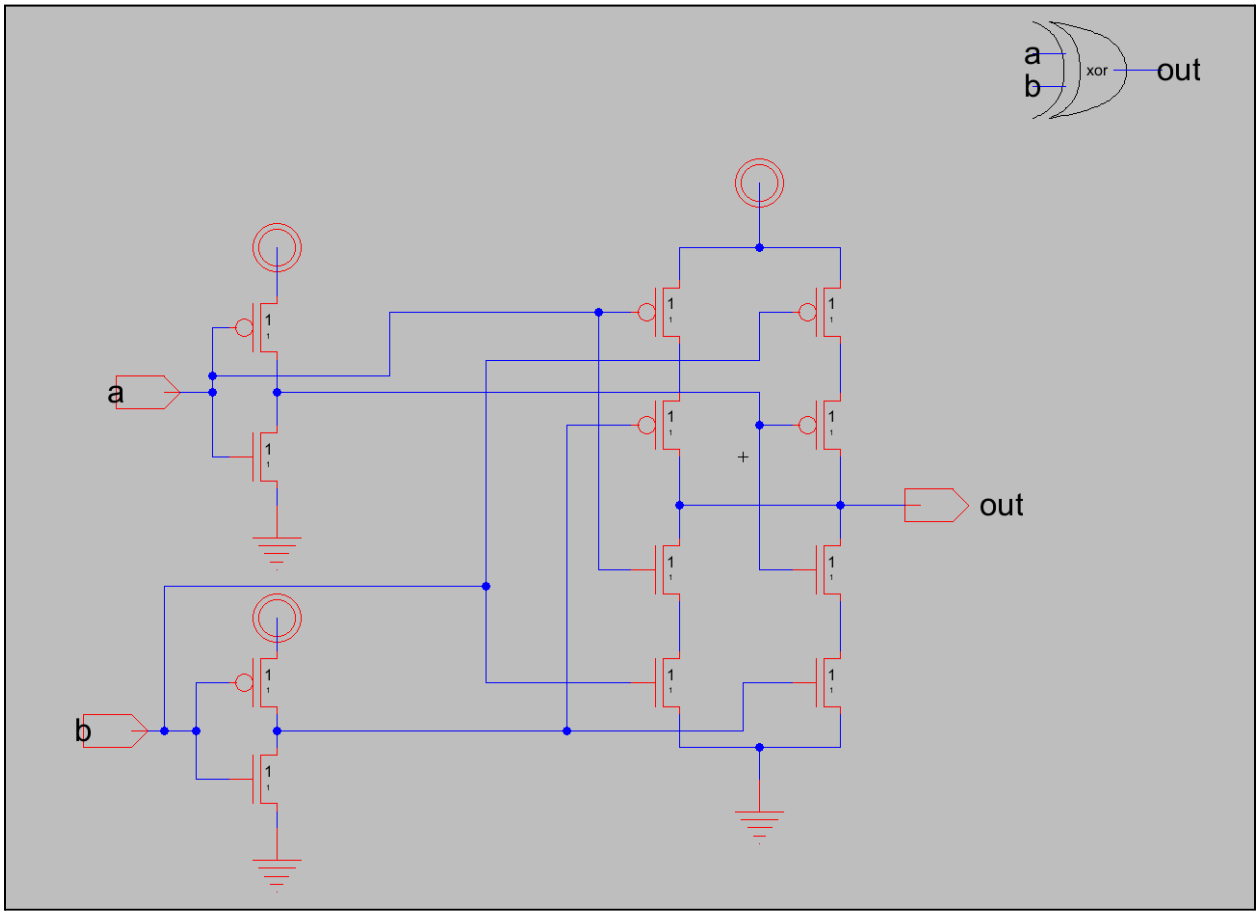

Figure 1.3: XOR2 Gate Schematic

For the XOR2 gate, we use the standard 12 transistor configuration as shown in class. Again, we are using minimum sized transistors for the unoptimized design.

Full-Adder:

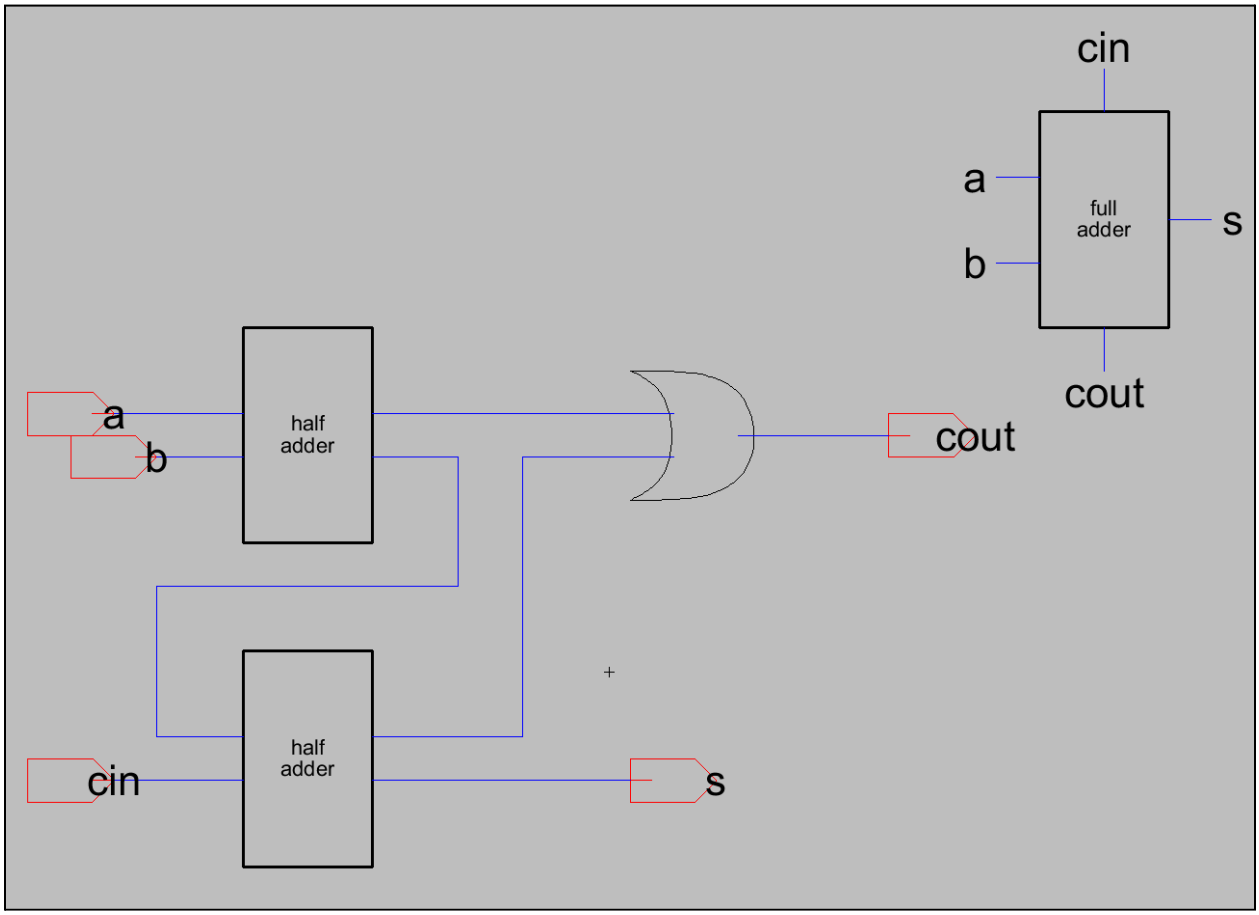

Figure 1.4: Full-Adder Schematic

Now here's the fun part. To make a full-adder that can add 2 bits, a and b, with a carry-in, we use two half-adders. The first takes in the two bits, a and b, and generates a temporary cout and sum. The sum then gets fed into the second half-adder, along with the carry-in. Finally, the output sum of the second half_adder is the final output sum and the cout of both adders are OR-ed together to generate the final cout. Here is the logic:

Sum_temp = a ^ b
Cout_1 = a * b
Final_Sum = Sum_Temp ^ cin = (a ^ b) ^ cin
Cout_2 = Sum_Temp * Cin
Final_Cout = Cout_1 + Cout_2 = (a * b) + (a ^ b) * Cin

OR2:

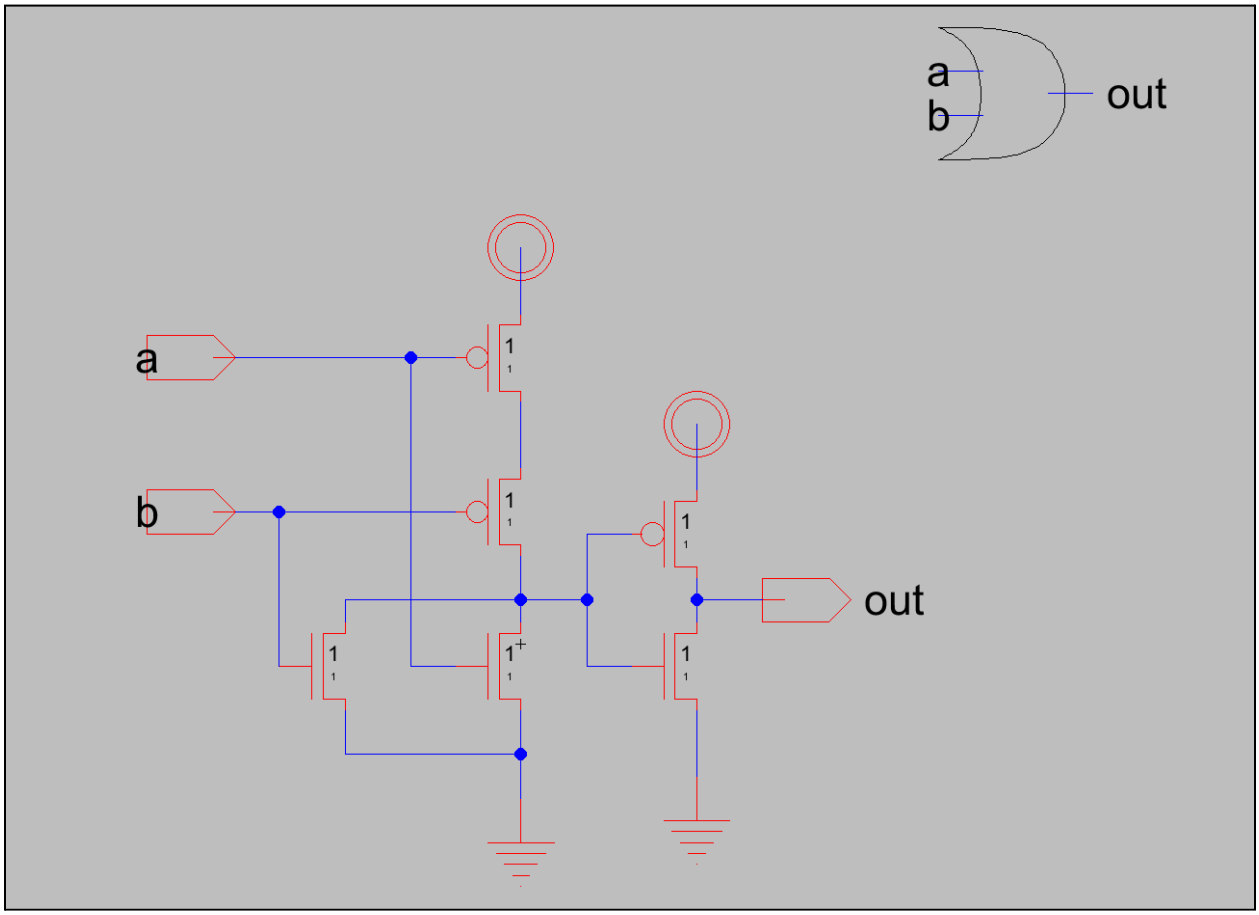

Figure 1.5: OR2 Gate Schematic

Here is the schematic of the OR gate used in the final step of the Full-Adder. It is a cascaded NOR2 and NOT gate using minimum-sized transistors.

8-Bit Adder:

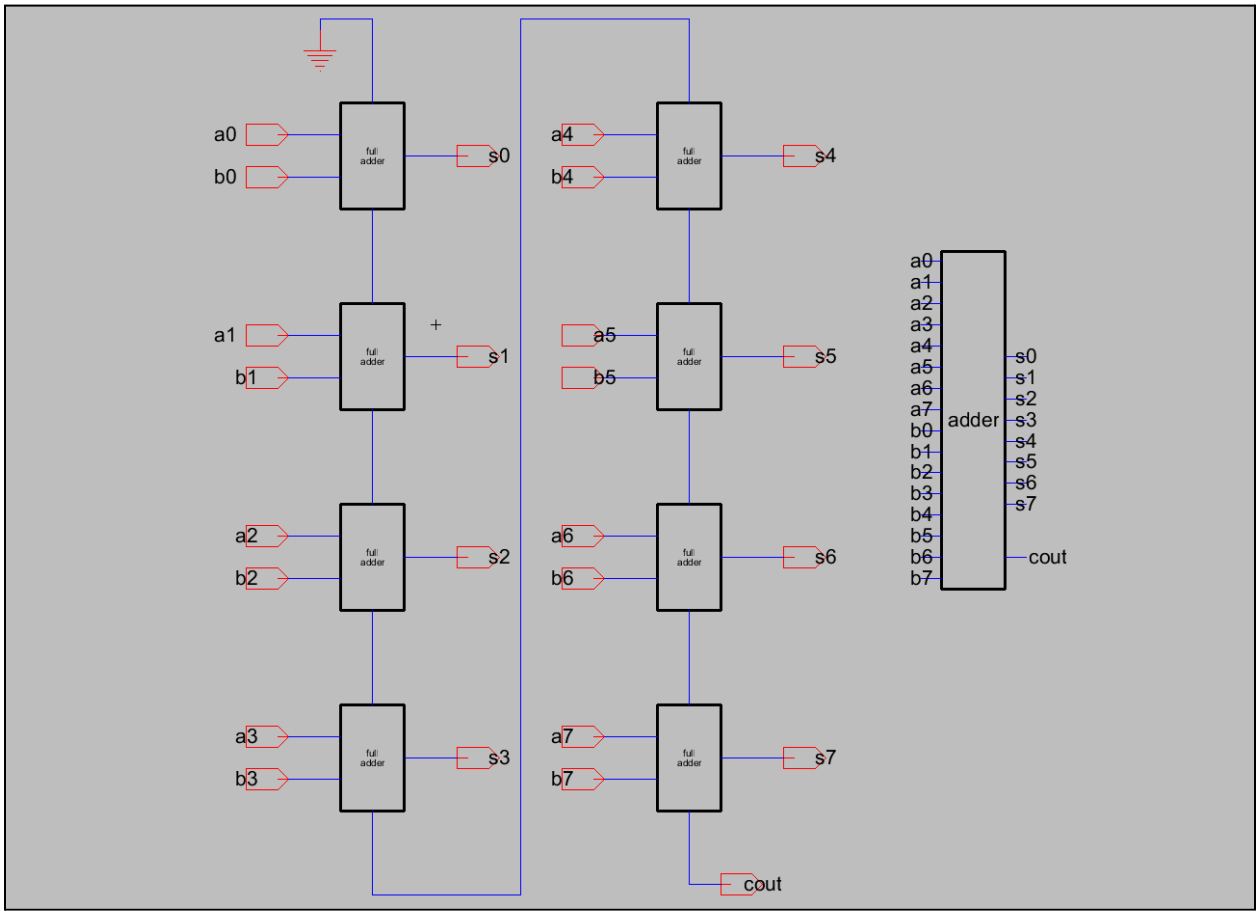

Figure 1.6: 8-bit Adder Schematic

The final 8-bit adder is composed of 8 Full-Adders. Each Full-Adder computes a sum and carry-out based on their inputs a[i], b[i], and cin[i]. In this way, we can add two 8-bit numbers, a[7:0] and b[7:0], and generate s[7:0] and a carry-out bit. In this design, I set cin to be ground since I am not trying to do subtraction or combine this circuit with other adders to make a larger one. This can easily be changed to accommodate different designs (so please don't take off points).

Design Notes:
Based on some insight from Professor Li, I will be using the same 8-bit adder schematic with cascading Full-Adders, since I want my design to be as modular as possible.

The critical path of the RCA design lies in the chain of carry-outs from one Full-Adder to the next. The last Full-Adder has to wait for all seven previous Full-Adders to compute their carry-outs before it can compute its sum and carry-out. This means that our most important area of optimization is the carry-out chain.

8-Bit Adder Logic Verification:

To verify the correct operation of the adder, I used 8 pulse generators at different frequencies to increment the adder by 1 every 0.5ns.

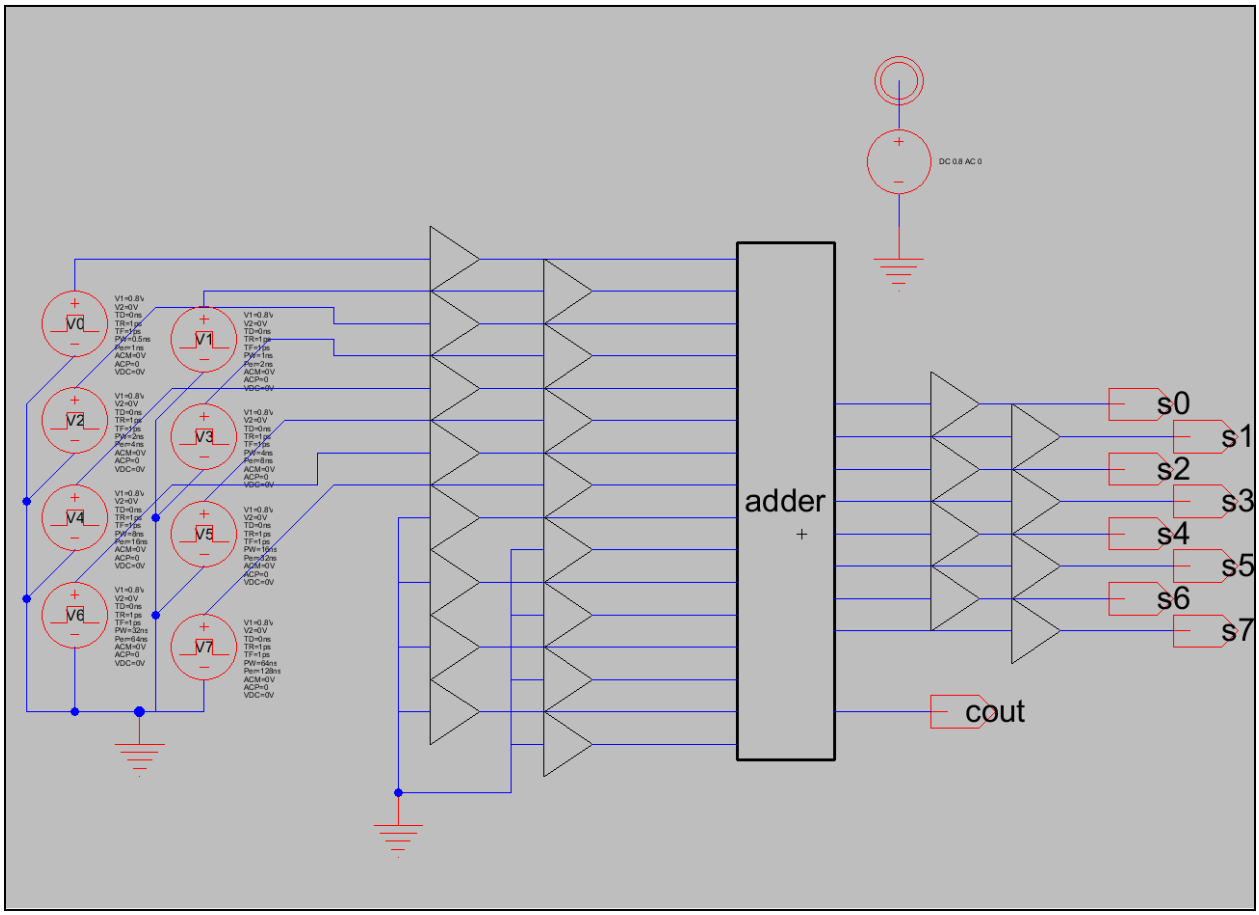

Figure 1.7: Logical Verification Test

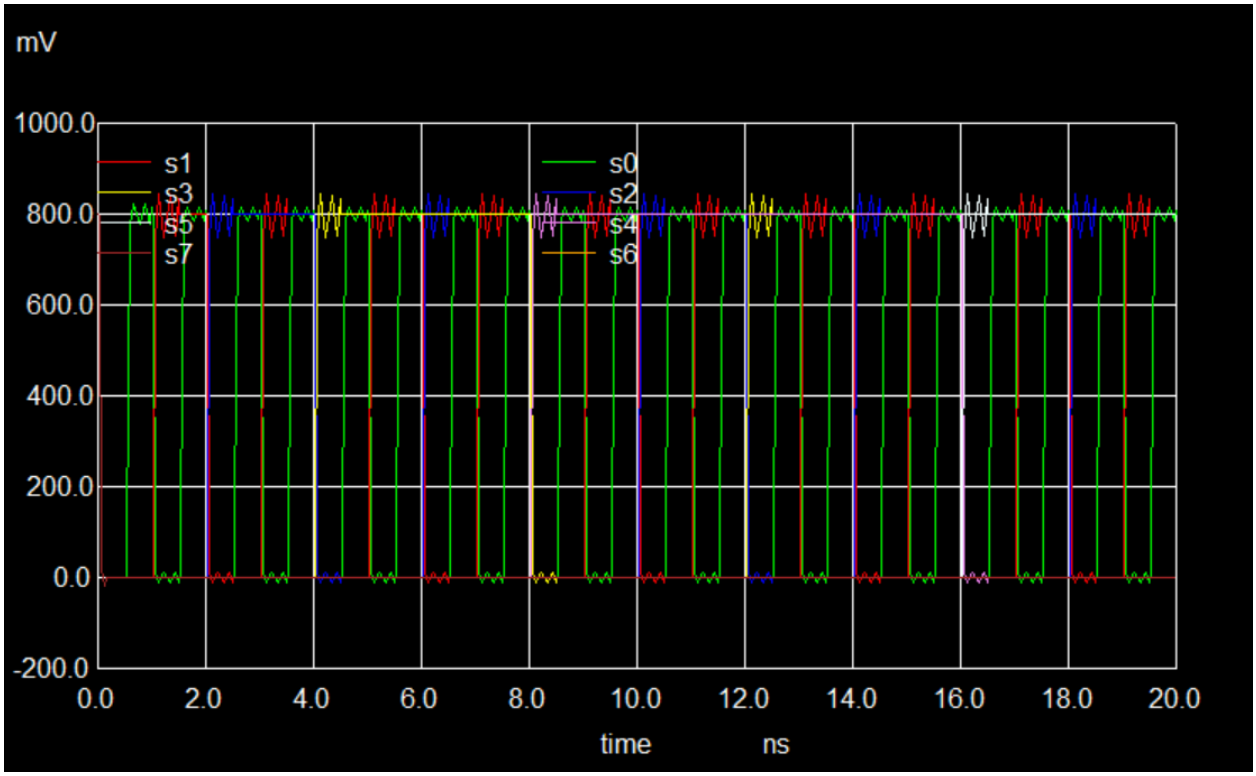

Figure 1.8: Logical Verification Test Result

The adder seems to be behaving normally across its inputs.

Unoptimized RCA Performance

Delay:

I assume that R_PMOS = 2 * R_NMOS = 2 * R_0 for all my τ calculations.

The critical path of this adder is a chain of AND - OR's that send the COUT signal through the adder, with the addition of one XOR at the beginning of the chain (from the first sum calculation in the first half-adder).

Rout of s[i] signals: 2R_0
Cin of a/b signals: 6C_0
Cin of Cin signals: 6C_0

Delay = $2R_0 * 6C_0 + 4R_0 * 6C_0 + 7 * (2R_0 * 6C_0 + 2R_0 * 2C_0 + 4R_0 * 2C_0) = 204\tau_0$

The input case for the worst case delay would be one where a carry-out is propagated from the first Full-Adder to the last.

Ex: a = b 1111 1111, b = b 0000 0001

We expect s[7:0] = b 0000 0000 and cout = 1. Since cout should be the last output to change, I will look at the propagation delay of cout to determine the worst case delay.

8-bit output load: 2R0
8-bit input capacitance: 6C0
Carry-Out output load: R0

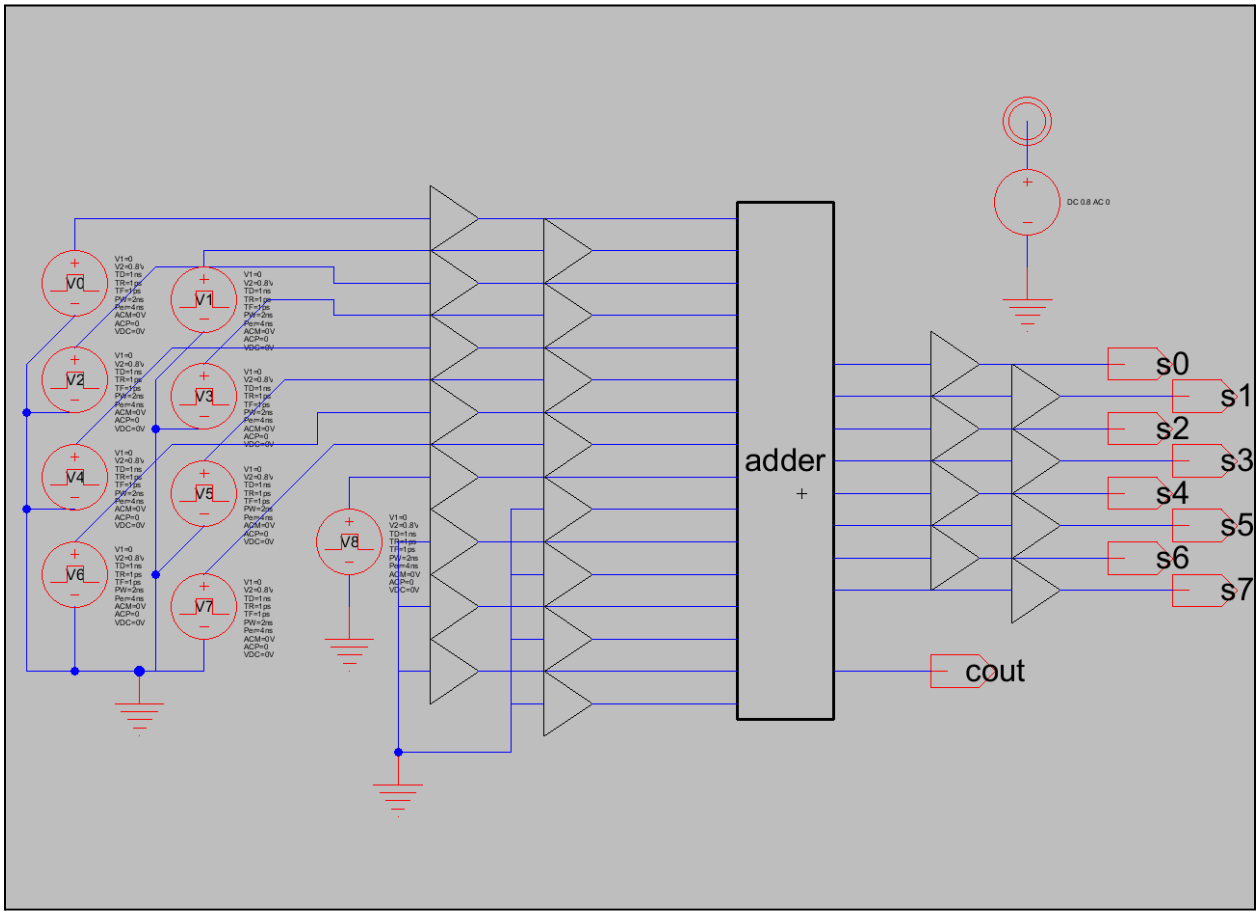

Figure 1.9: Delay Test Schematic

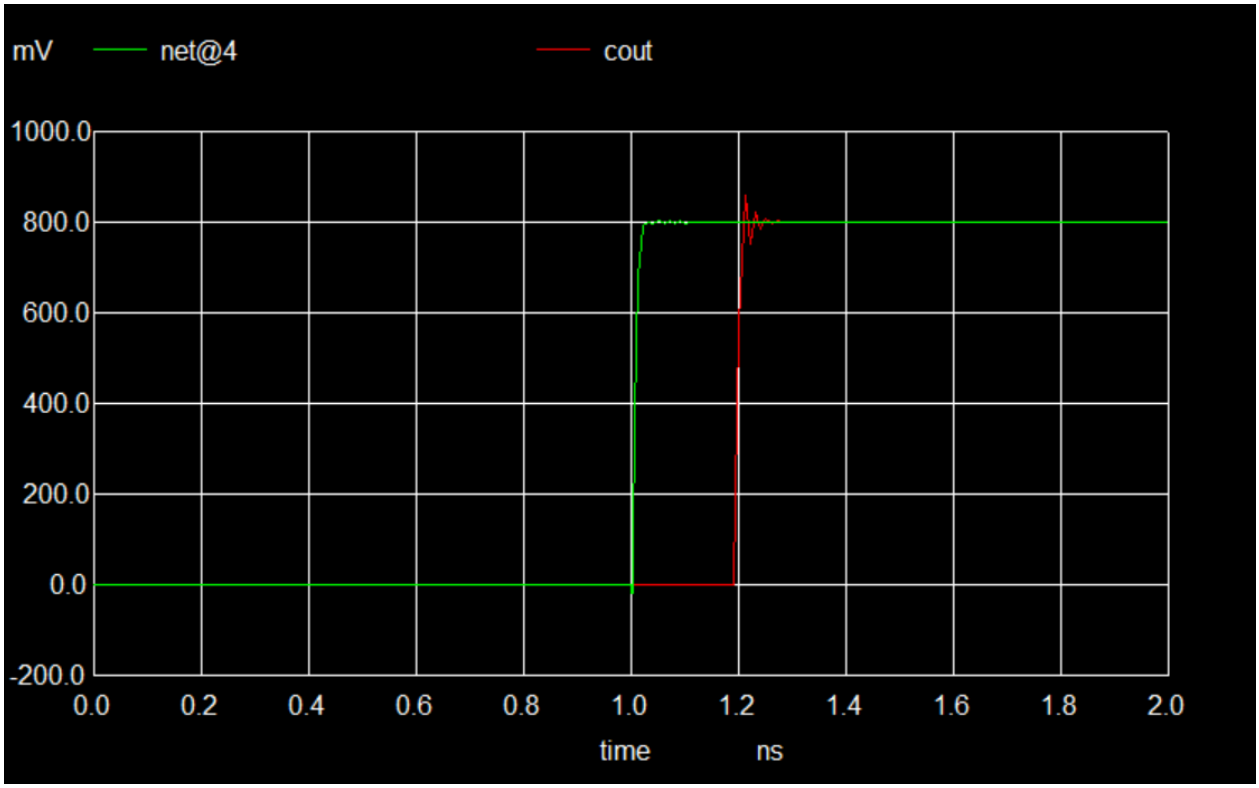

Figure 1.10: Delay Results (net@4 is the direct input to the adder)

From our test, we see that the worst-case delay of our unoptimized adder is 200ps.

Active Energy - Maximum:

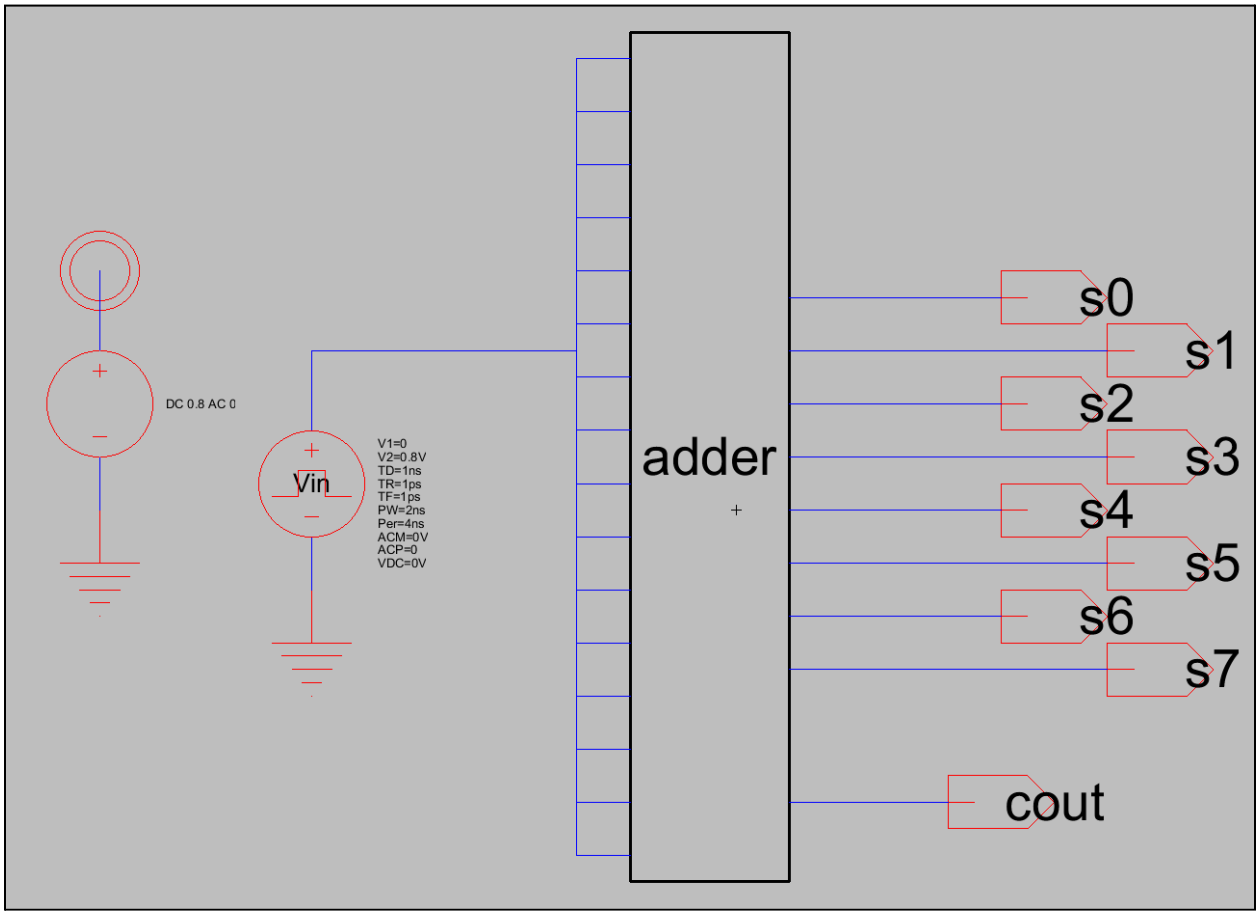

Figure 1.11: Max Active Energy Test Schematic

Integral of current through Vdd during switching: $1.99 * 10^{-15}$ Coulomb

Energy burned: E = Q * V = $1.99 * 10^{-15} C * 0.8 V = 1.592 * 10^{-15} Joules$

Active Energy - Average:

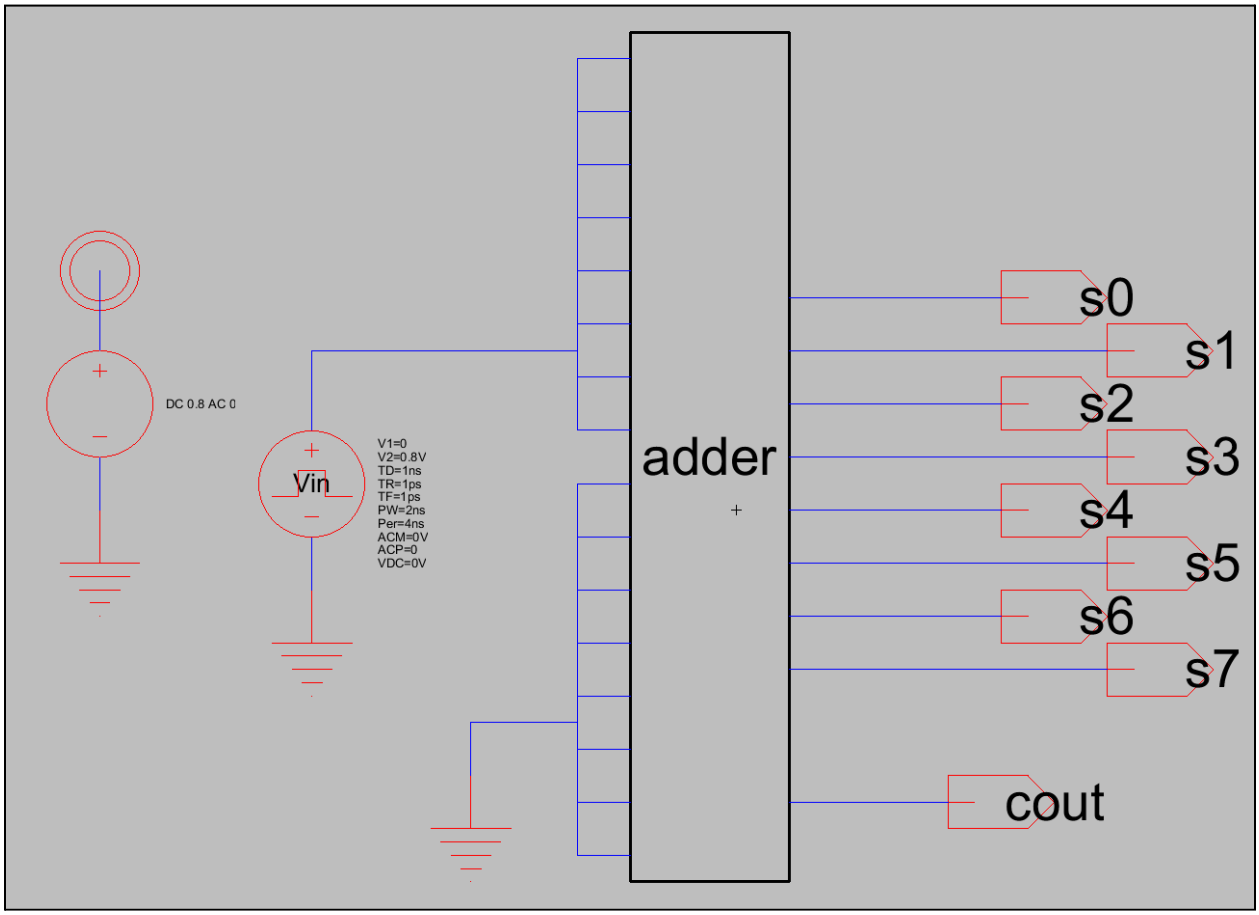

Figure 1.12: Average Active Energy Test Schematic

Integral of current through Vdd during switching: $1.334 * 10^{-15}$ Coulomb

Energy burned: E = Q * V = $1.334 * 10^{-15} C * 0.8 V = 1.0672 * 10^{-15} Joules$

Leakage Energy:

For the best precision, I will empirically determine the input combinations that produce the max and min leakage energy for the Full-Adder.

Min Leakage Energy:

- A = 1
- CIN = 0 = B

Max Leakage Energy:

- A = B = 0

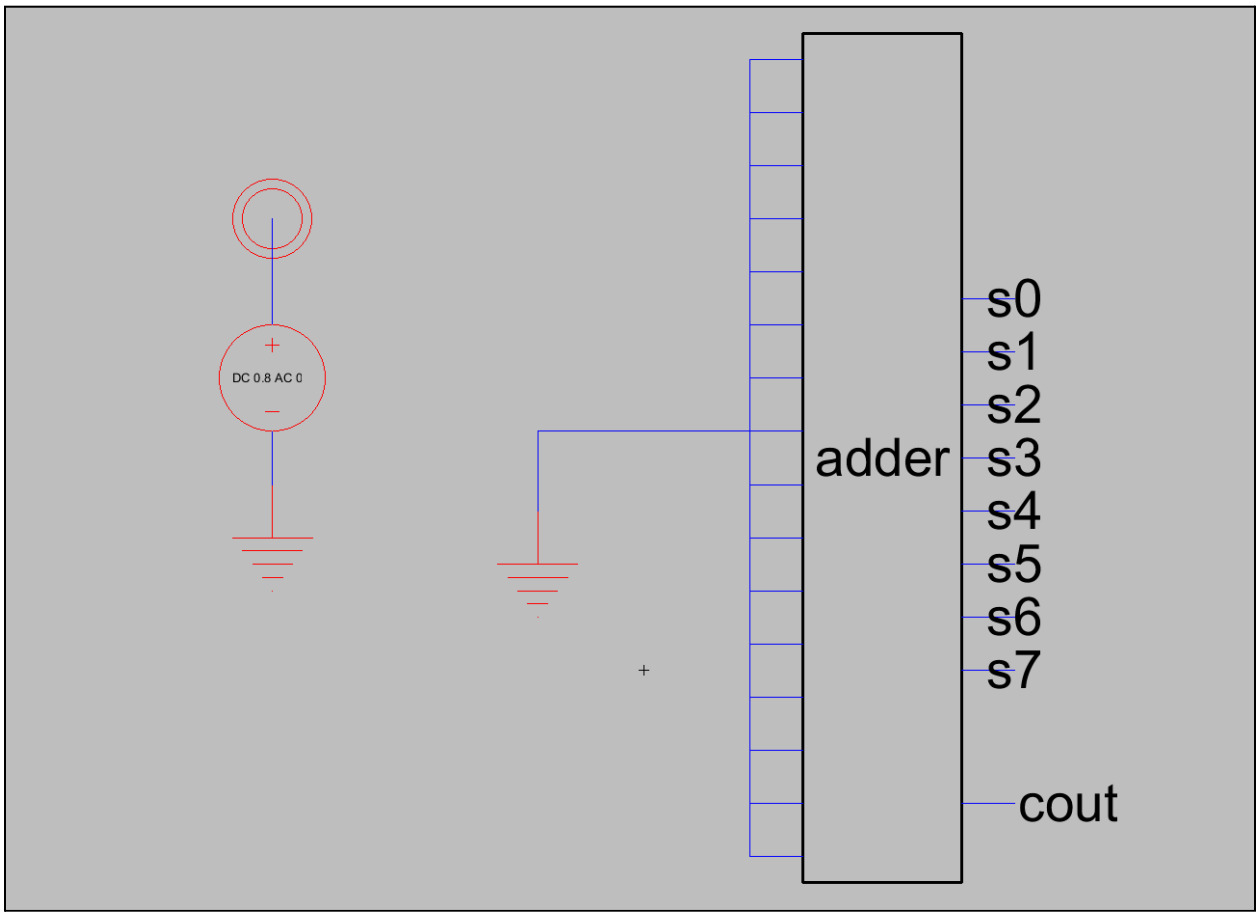

Figure 1.13: Max Leakage Energy Test Schematic

Integral of current through Vdd over delay period (200ps): $4.06 * 10^{-17}$ Coulomb

Energy burned: E = Q * V = $4.06 * 10^{-17} C * 0.8 V = 3.248 * 10^{-17} Joules$

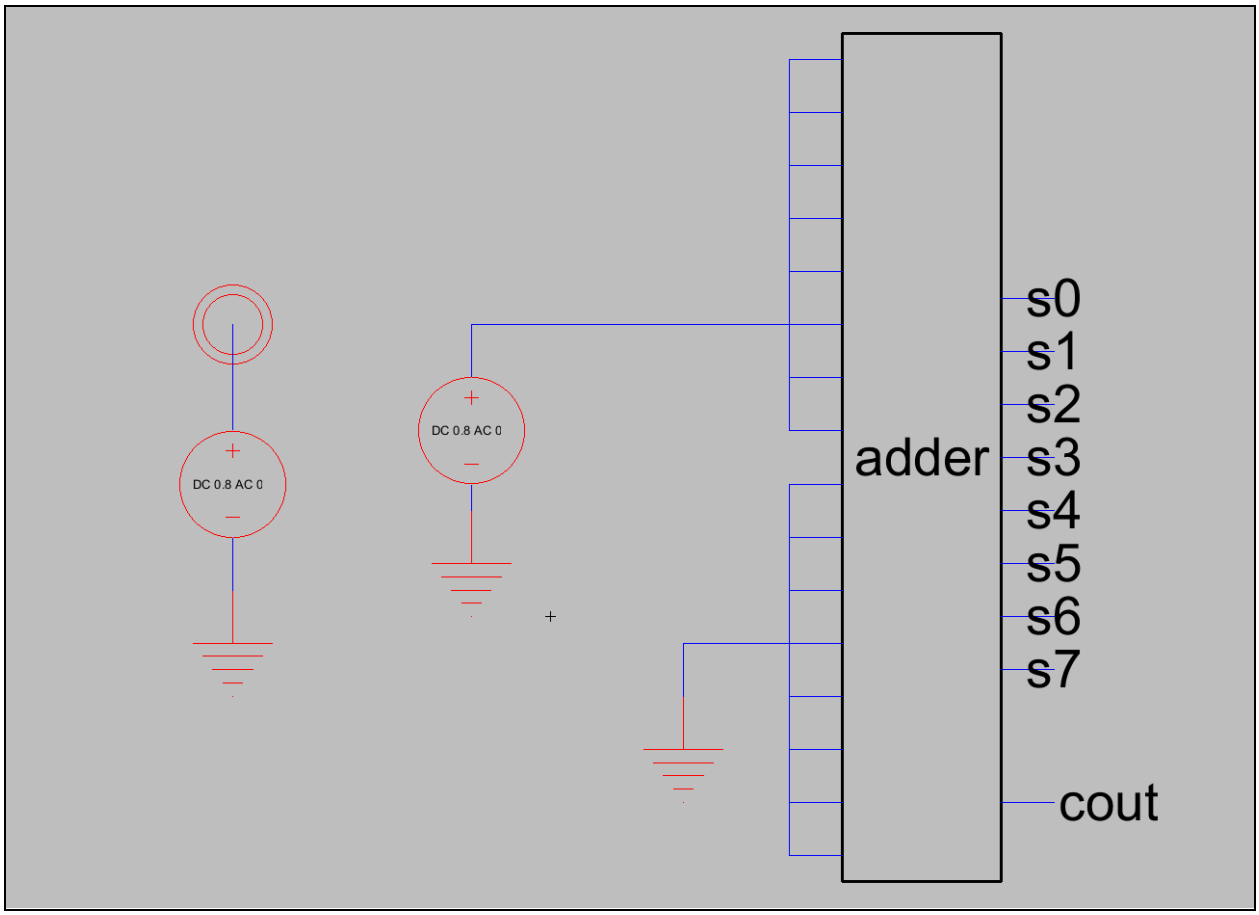

Figure 1.14: Min Leakage Energy Test Schematic

Integral of current through Vdd over delay period (200ps): $3.72 * 10^{-17}$ Coulomb

Energy burned: E = Q * V = $3.72 * 10^{-17} C * 0.8 V = 2.98 * 10^{-17} Joules$

Area: (note: all transistors are minimum sized)

AND2 Gate: 6 transistors
XOR2 Gate: 12 transistors
OR2 Gate: 6 transistors

Half-Adder: 1 AND2 gate and 1 XOR2 gate
Full-Adder: 2 Half-Adders and 1 OR2 gate
8-Bit Adder: 8 Full Adders

$6 * 1 * 2 * 8 = 96$ transistors from AND2 gates.

$12 * 1 * 2 * 8 = 192$ transistors from XOR2 gates.

$6 * 1 * 8 = 48$ transistors from OR2 gates

Final Area: 96 + 192 + 48 = 336

Optimized RCA Design

List of Considerations:

- The major optimization needs to occur on the Carry Out chain.
- The performance of the sum gates does not matter by comparison - I can probably leave them as a minimal design or make a large, custom sum gate.
- I may be able to improve delay by creating a custom carry-out gate. My current design has 10 transistors along with 1 NOT gates, which will probably perform better than the current chaining.
- I don't care very much about energy or area, so I won't be treating them as first class citizens.
- Since we are mostly concerned with the path for the carries, we want to minimize the fanout of all carry signals if possible (minimize input capacitance for non-critical gates).

COUT Gate - Function:

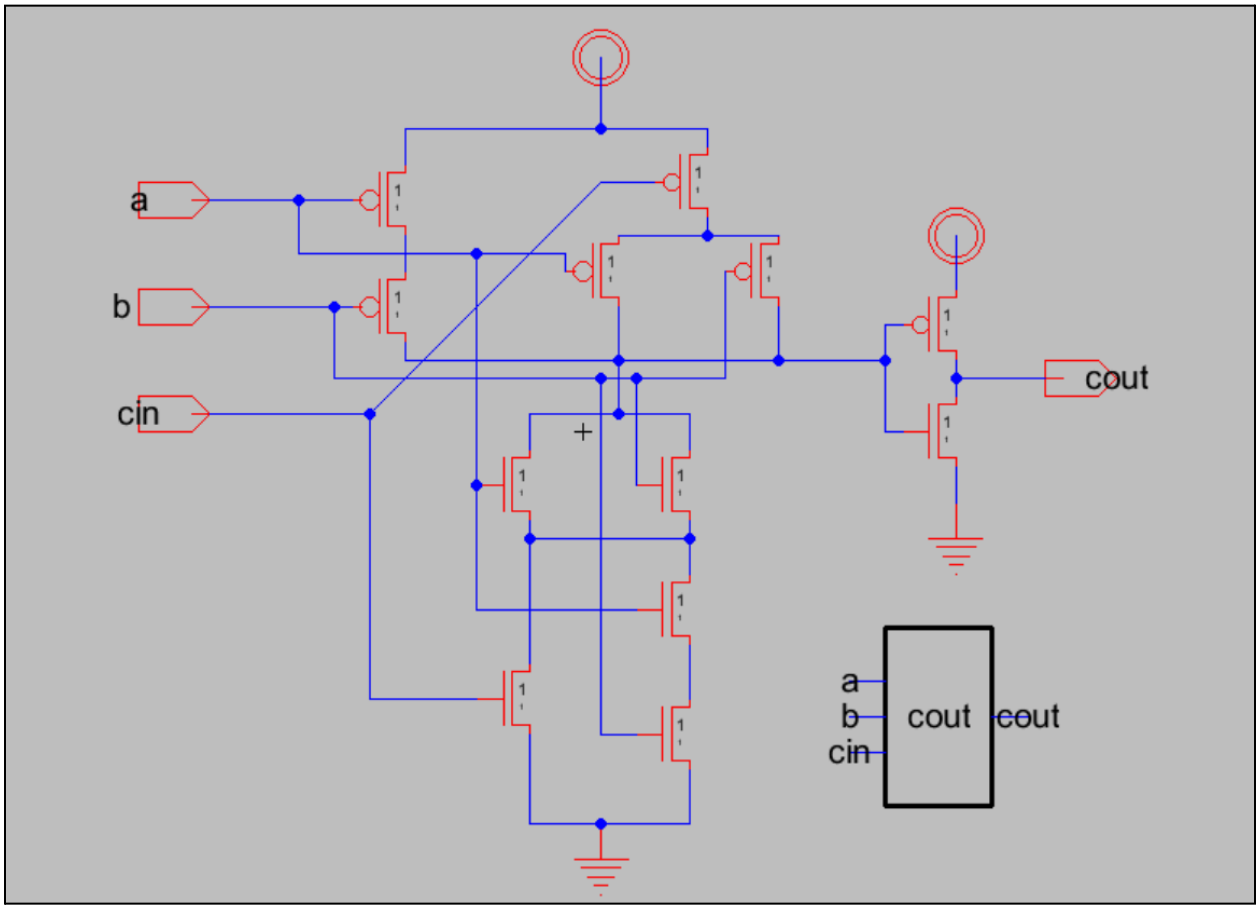

Figure 2.1 First Attempt at COUT Gate

Here is my first attempt at a COUT Gate. I have tried to prioritize minimizing the fanout of the CIN signal - you can see that it only drives two gates. The gate is actually a cascaded ~COUT

and NOT gate. I did it this way because I did not want to have to invert my inputs and thereby use more transistors unnecessarily.

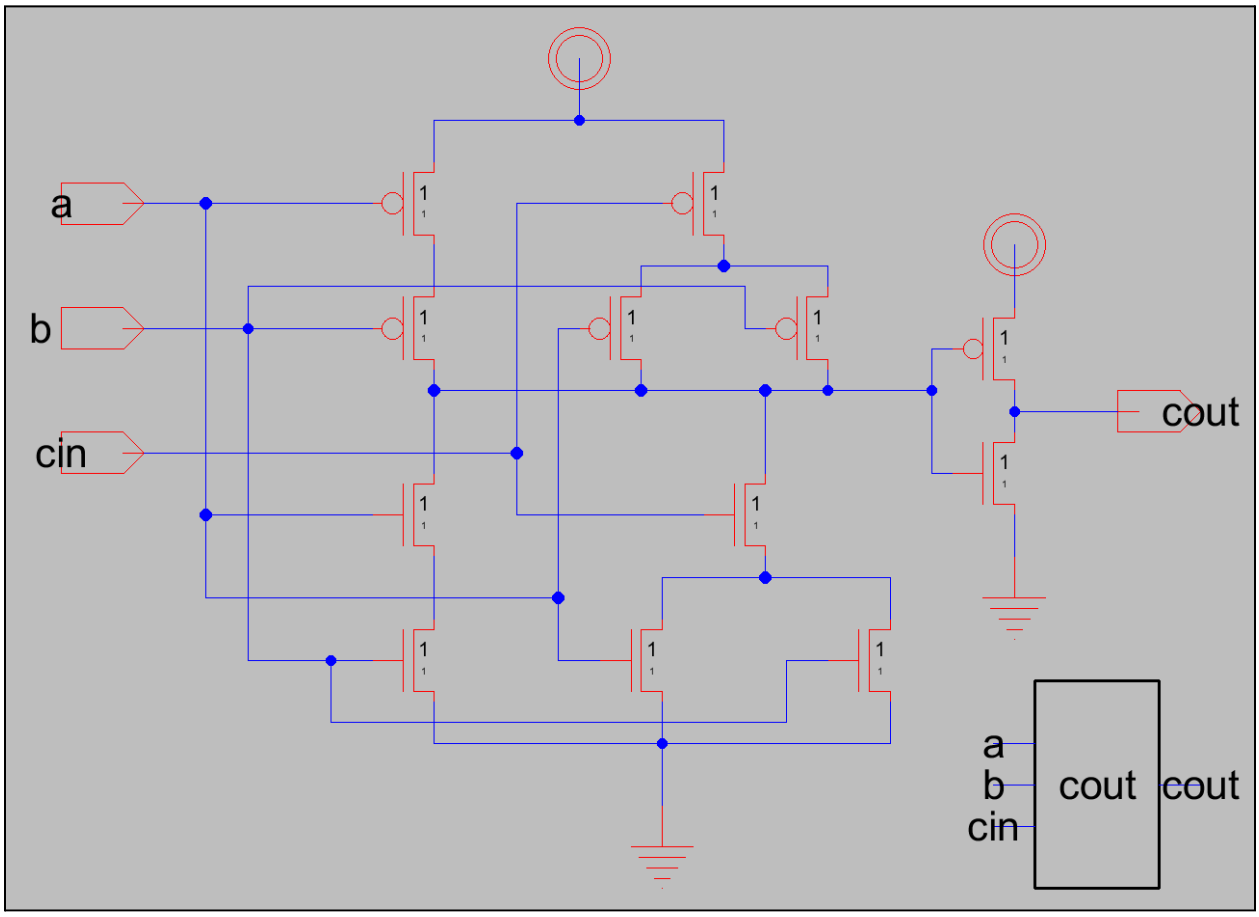

Figure 2.2: Second Attempt at COUT Gate

I preferred this design for the sole reason that it decreases the worst case resistance (2 NMOS in series vs 3 NMOS in series) of the pull-down network for the ~COUT Gate. This should be a general improvement to the previous design.

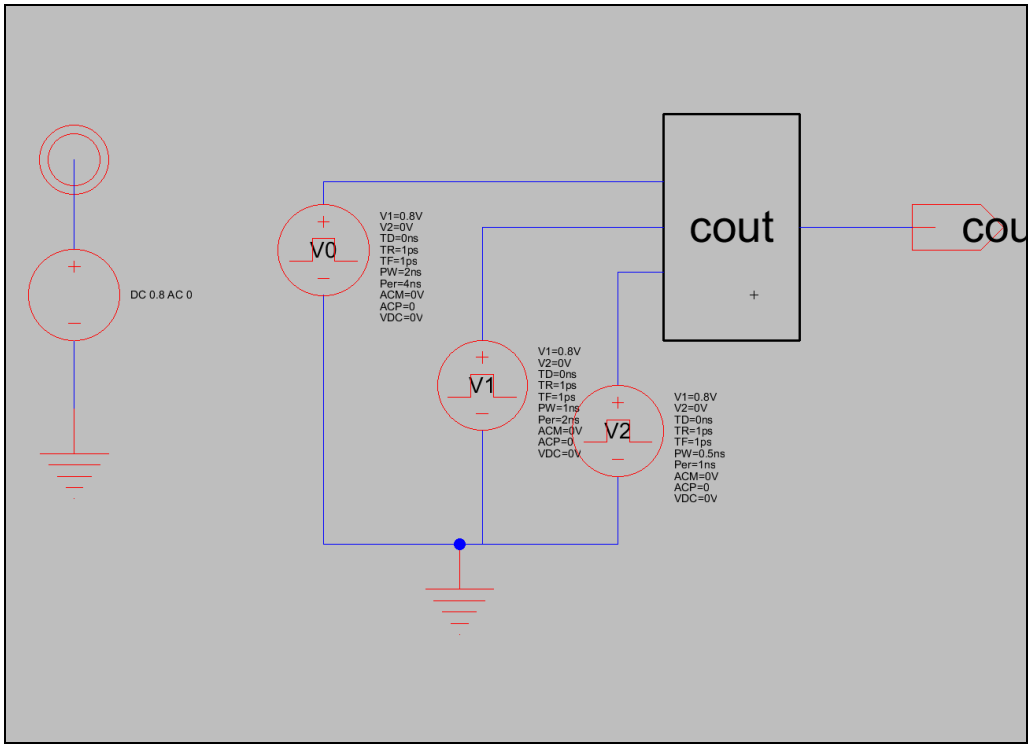

Figure 2.3 Cout Logic Test Schematic

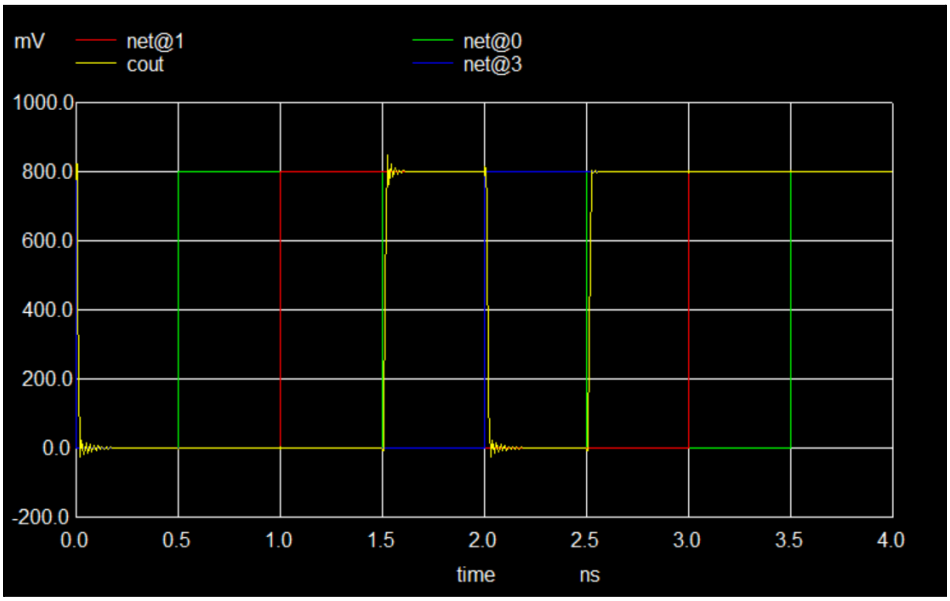

Figure 2.4 Cout Gate Output (Yellow) for all possible input combination)

I have verified that my COUT Gate functions correctly for all possible inputs, so I will now work on discovering its worst case delay and how I can improve it.

COUT Logic Explanation:

COUT: at least two of A, B, and CIN are 1

~COUT: less that two of A, B, and CIN are 1

~COUT = ~A * ~B + ~A * ~CIN + ~B * ~CIN

Since we want to minimize the number of gates CIN is driving, we can rewrite this logic to be:

~COUT = ~A*~B + ~CIN * (~A + ~B)

SUM Gate - Function:

To start, we do not care about the performance of the SUM gates. What we do care about is minimizing the capacitive load on incoming CIN signals, since it affects the output load of the COUT gates.

To this end, I have opted for a simple cascaded XOR2 x 2 design for the SUM calculations. The final boolean expression would be s = a ^ b ^ cin. As a simple optimization, we input the CIN signal to the last XOR2 gate.

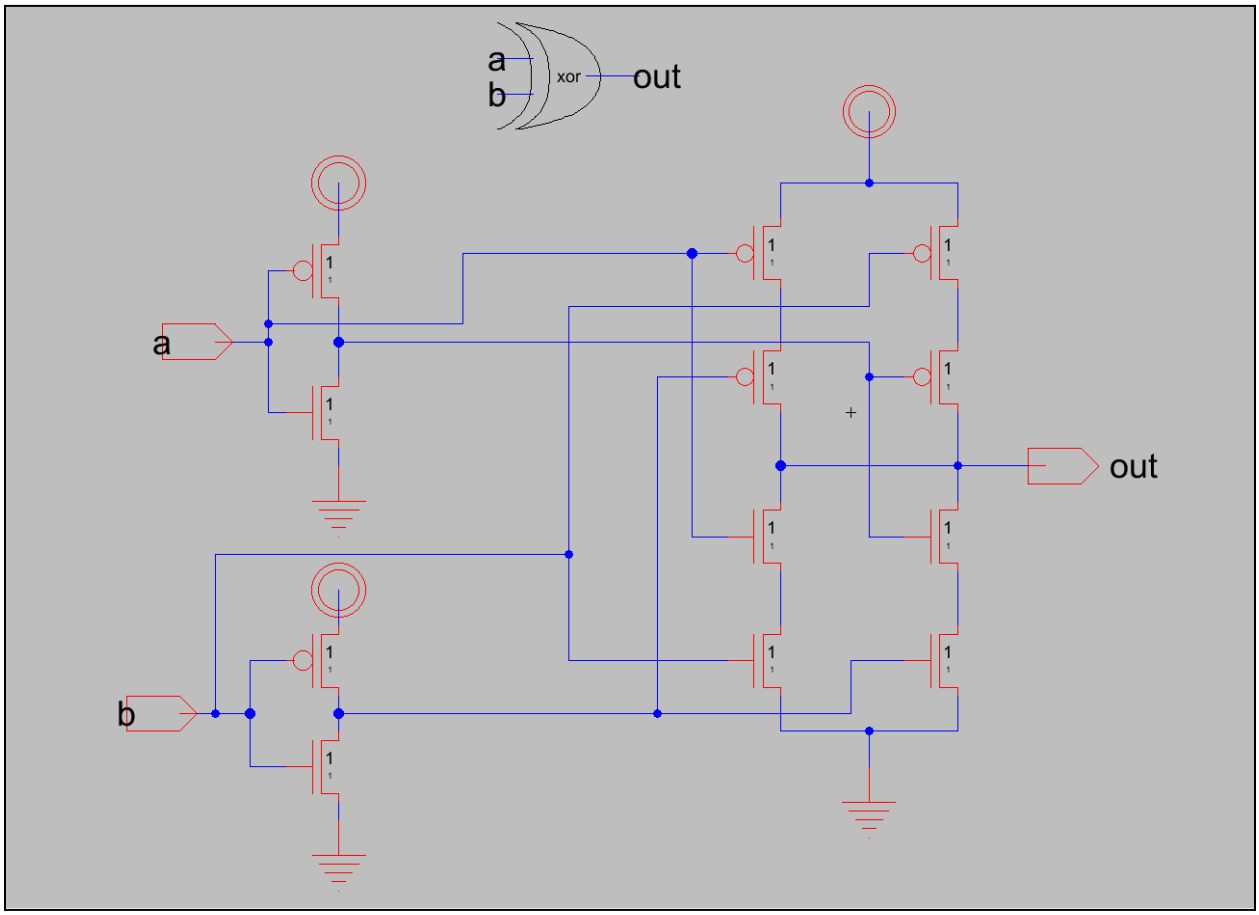

Figure 2.5 XOR Gate

The capacitive load on any input signal is 4C_0 since each input drives 4 minimum sized gates. Since we do not want to increase this, we will not be sizing up any of these transistors.

Here is the Full-Adder schematic with the COUT Gate and cascaded XOR2s:

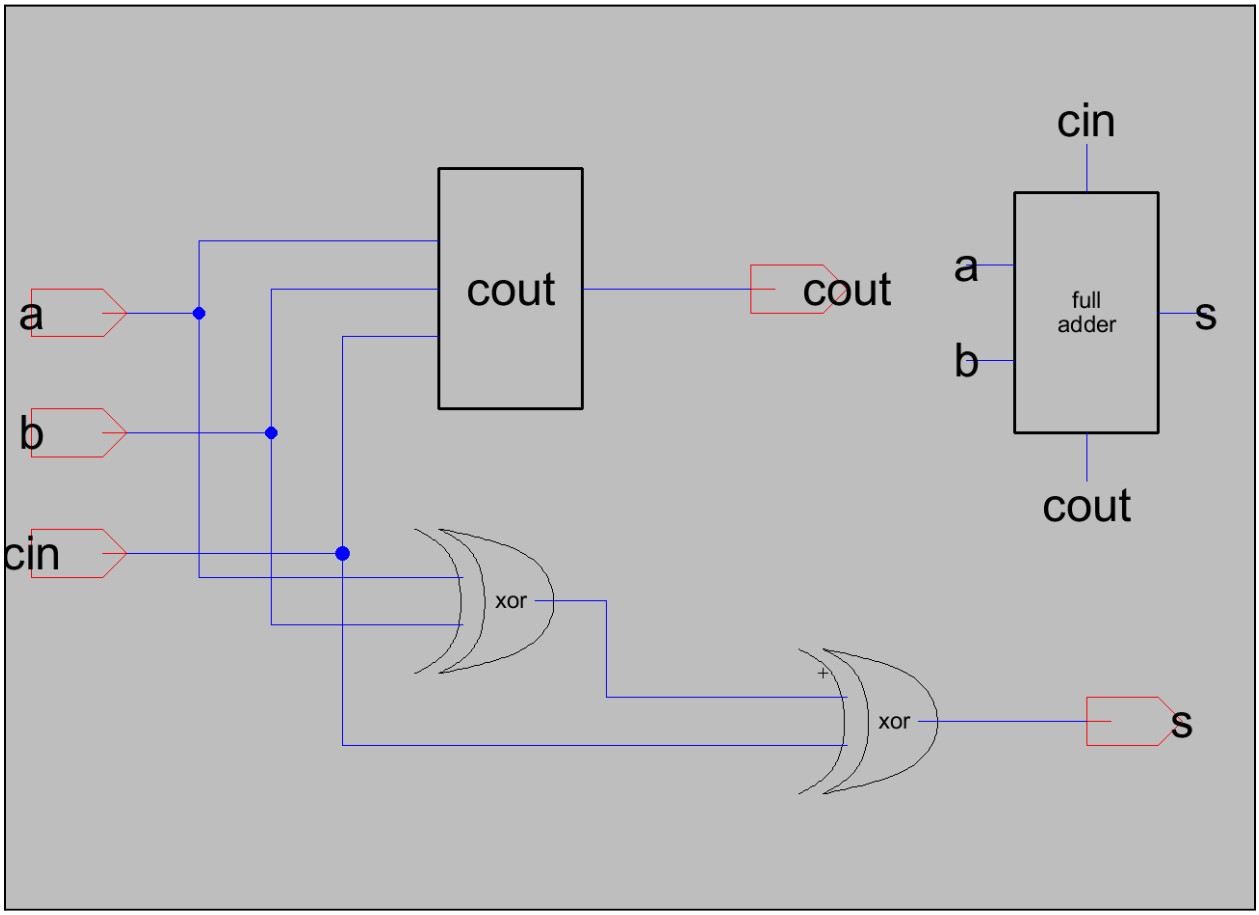

Figure 2.6: Optimized Full-Adder

COUT Gate - Delay:

To minimize delay due to junction capacitance, and with the assumption that the CIN signal will always arrive last due to its nature as the critical path, I will make the following adjustment to the Pull Up Network of the COUT Gate:

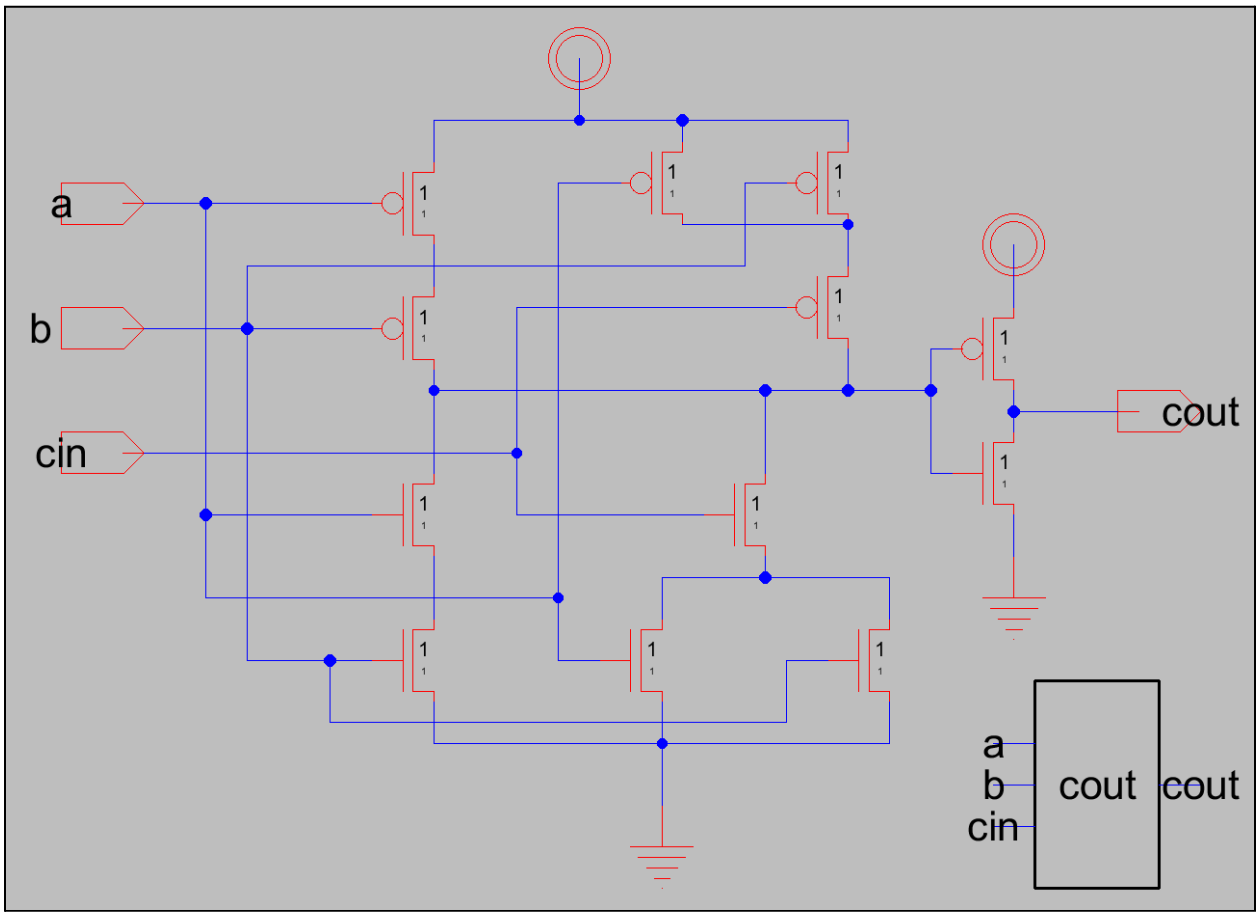

Figure 2.7: Third Attempt at COUT Gate

Assumptions:

- Every COUT gate is driving another COUT gate and an XOR2 Gate.
- Signals A and B are already populated (CIN is always coming after A and B)
- Capacitive load of driving an XOR2 gate is 4C_0

Worst Case Delay Calculation 1: $t = 2R_0 * (4C_0 + 2C_0) + 4R_0 * 2C_0 = 20\tau_0$

We can improve this band-of-the-hand calculation by sizing the PUN of the inverter at the end of the COUT gate to have equal output resistance to the PDN (R_0).

Worst Case Delay Calculation 2: $t = R_0 * (4C_0 + 2C_0) + 4R_0 * 3C_0 = 18\tau_0$

Let's test for the worst case delay! The input combination to test worst case delay would be:

- A = 1
- B = 0
- Cin: 1 -> 0 -> 1

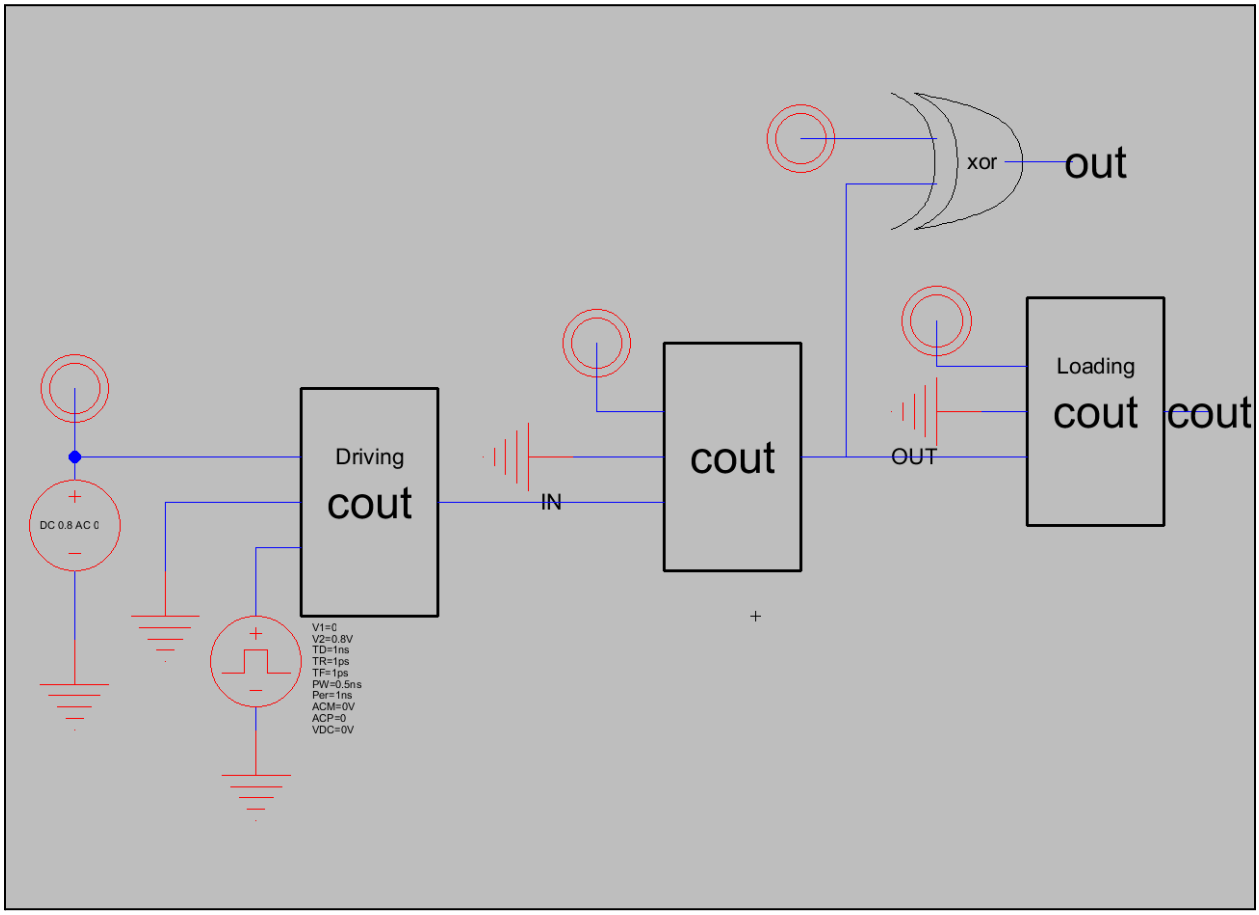

Figure 2.8: COUT Delay Test Schematic

We drive the COUT gate in the middle with another COUT gate and load it with a COUT gate and XOR gate to mimic real operation.

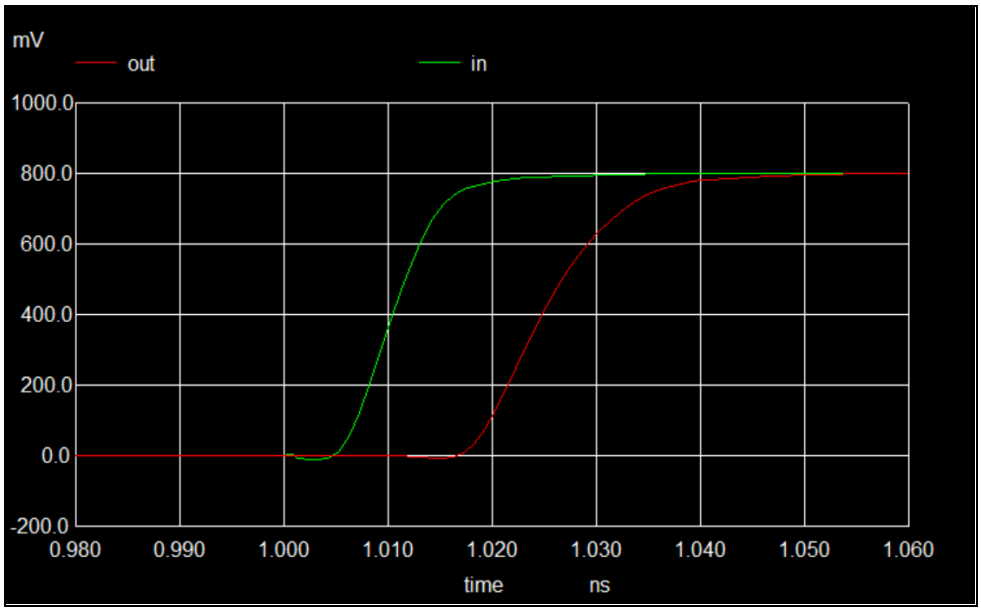

Figure 2.9: 0 -> 1 Delay for Min Sized COUT Gate

Delay = 14.2ps

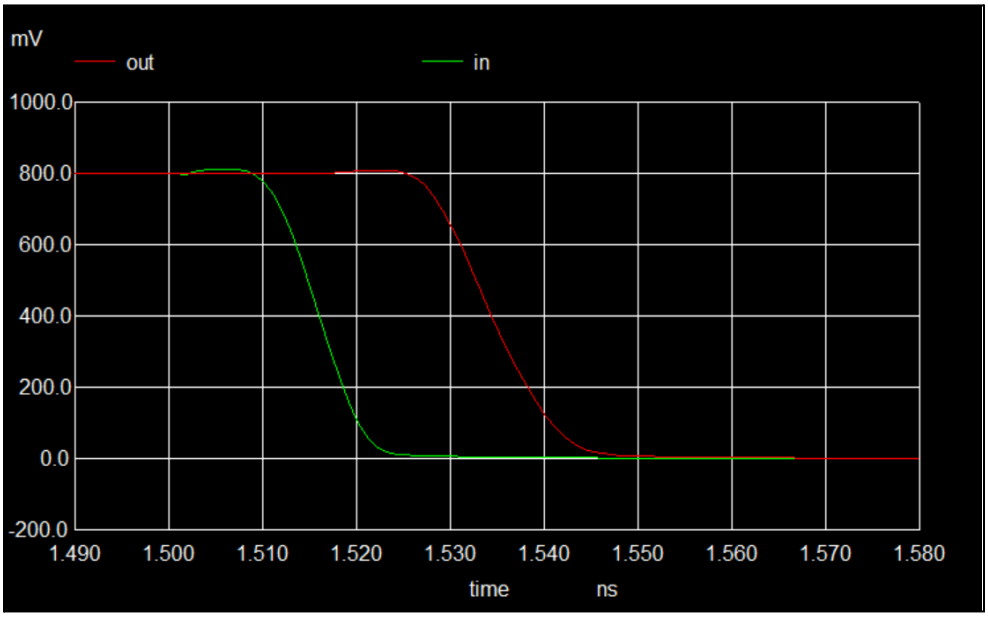

Figure 2.10: 1 -> 0 Delay for Min Sized COUT Gate

Delay = 18.3ps

The delay simulations show that our COUT design has a delay of 18.3ps on the falling edge. This can likely be improved by sizing the PUNs better.

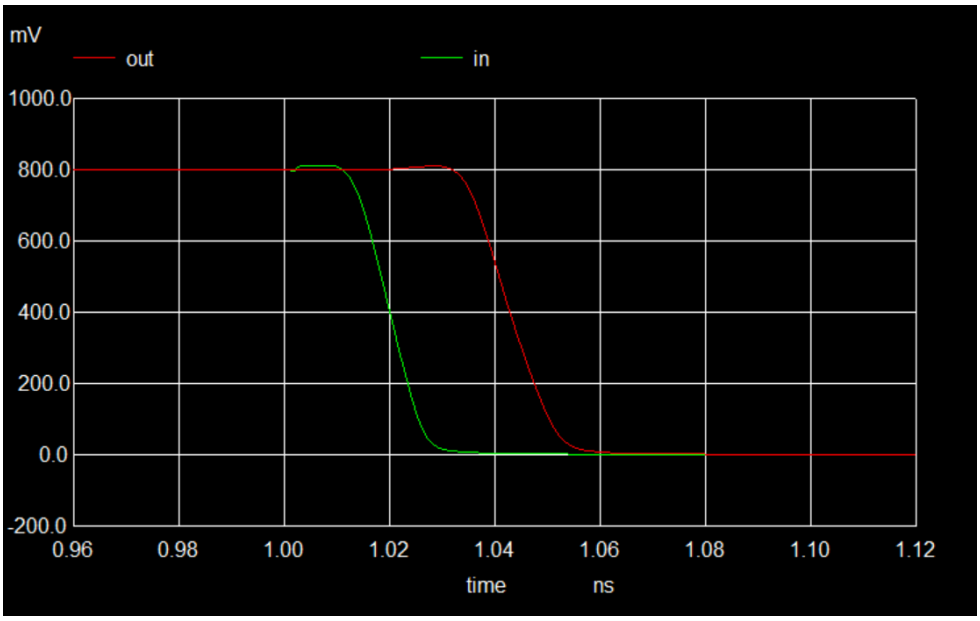

Figure 2.11: 1 -> 0 Delay for COUT Gate with Inverter PUN sized 2x

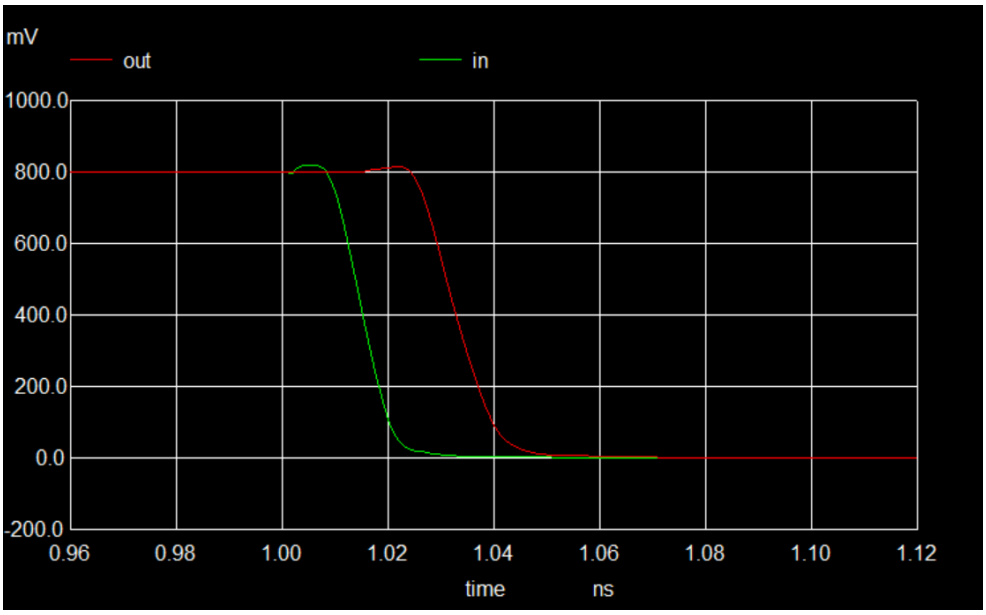

Figure 2.12: 1 -> 0 Delay for COUT Gate with both PUNs sized 2x

After some SPICE experimentation, I can conclude that sizing up the transistors of the COUT gate hurts its performance. It is likely that my tau model assumption for resistance was not accurate. The additional capacitance seems to dominate the circuit, likely due to the small number of output signals.

Final delay for COUT Gate: 18.3ps

8-Bit Adder:

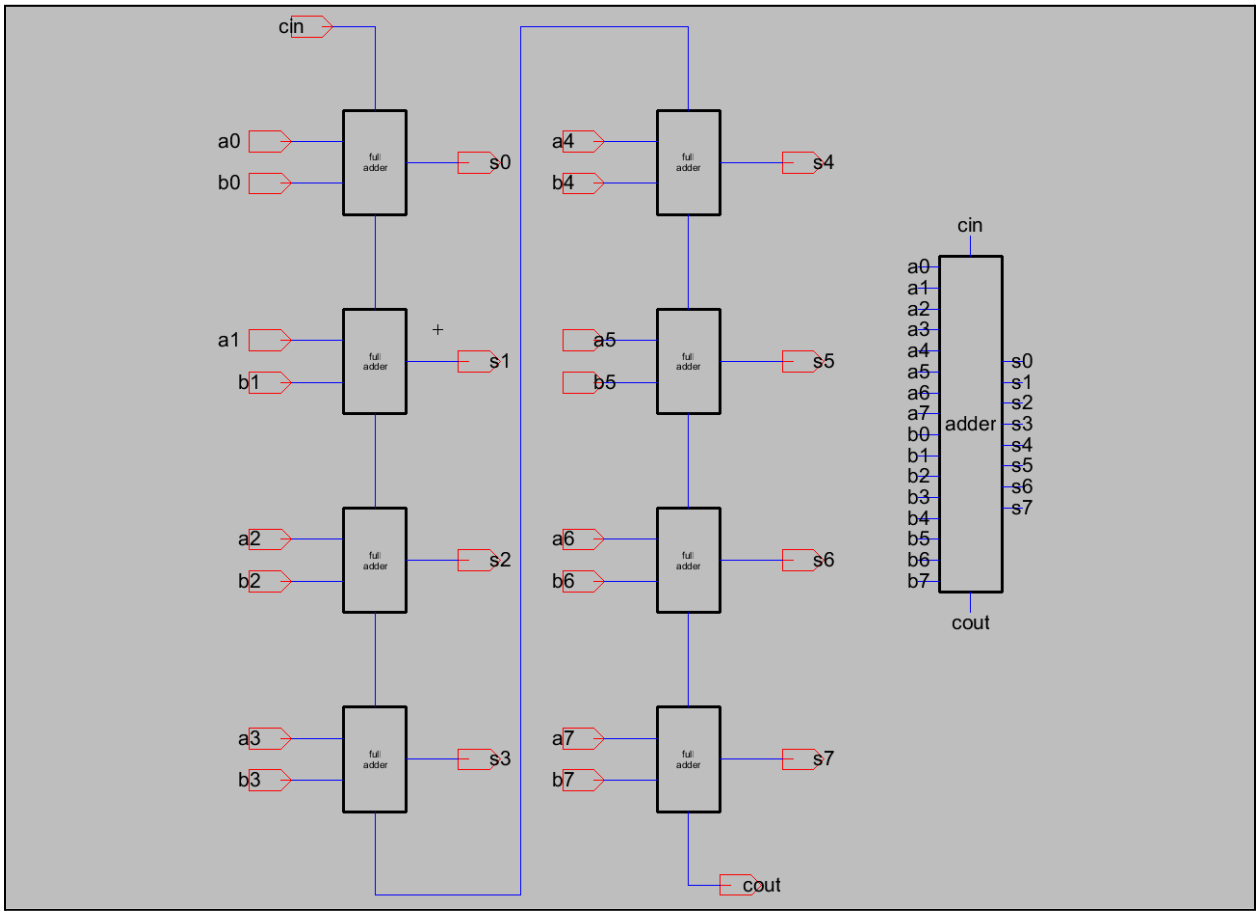

Figure 2.13: 8-Bit Adder Design

The final adder design is nearly the same as the baseline. The only two things that have changed are the composition of the full adders as shown in Figure 2.6 and the addition of a Carry-In input that was implicit in the previous version but not implemented in Electric as it was not necessary. Here it is needed for proper loading of the COUT output for testing purposes, as I am more concerned about the validity of my metrics for this design.

Optimized RCA Performance

Delay:

Assuming that the delay of the adder is equivalent to the delay of the critical path of the COUT gates (which it is - notice that COUT in the resulting graph is updated after the final sum bit), the delay of the adder equals the delay of the COUT gate times 8.

Tau model delay: $20\tau_0 * 8 = 160\tau_0$

We will conduct the same worst case scenario for delay as done for the unoptimized design as it remains the longest possible path through the circuit.

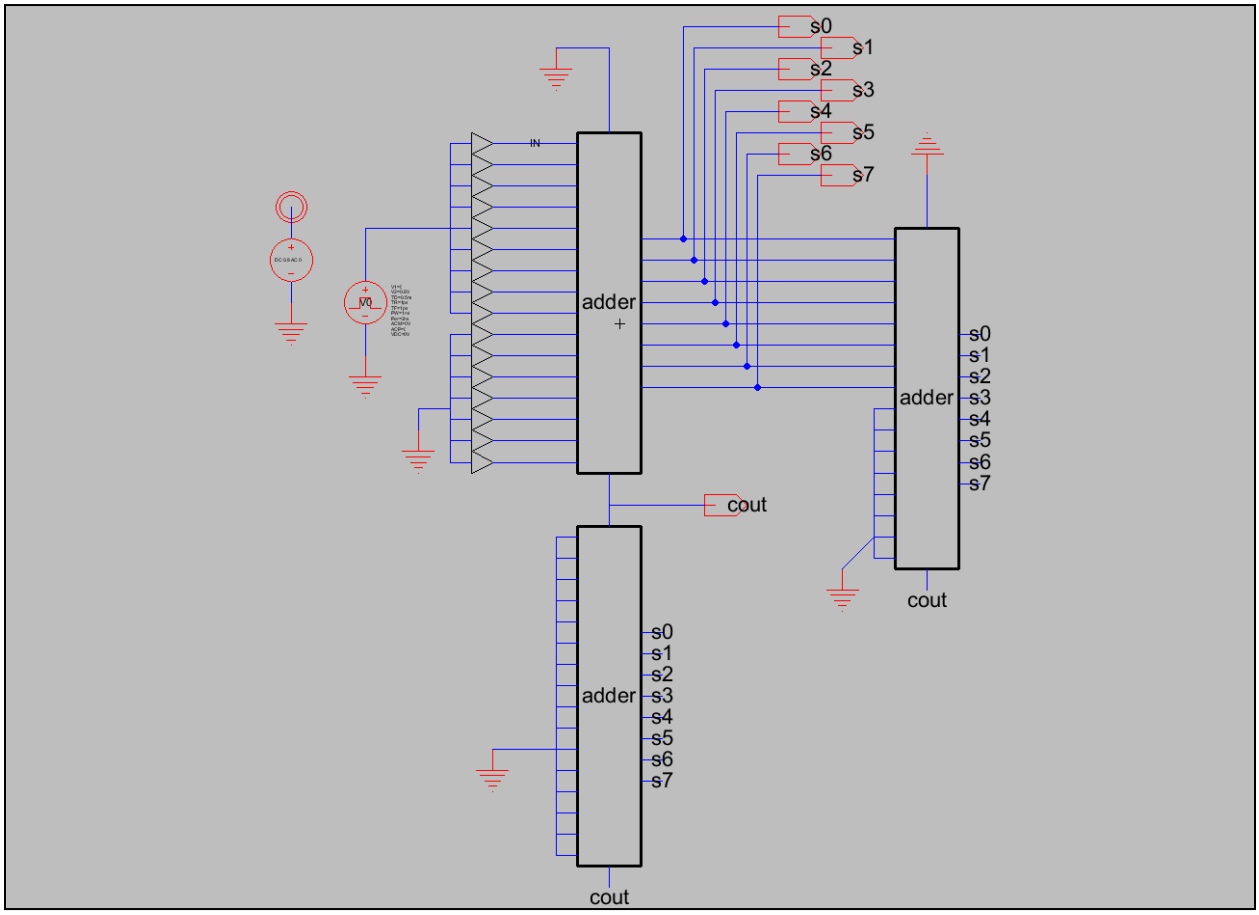

Figure 2.14: Delay Testing Schematic

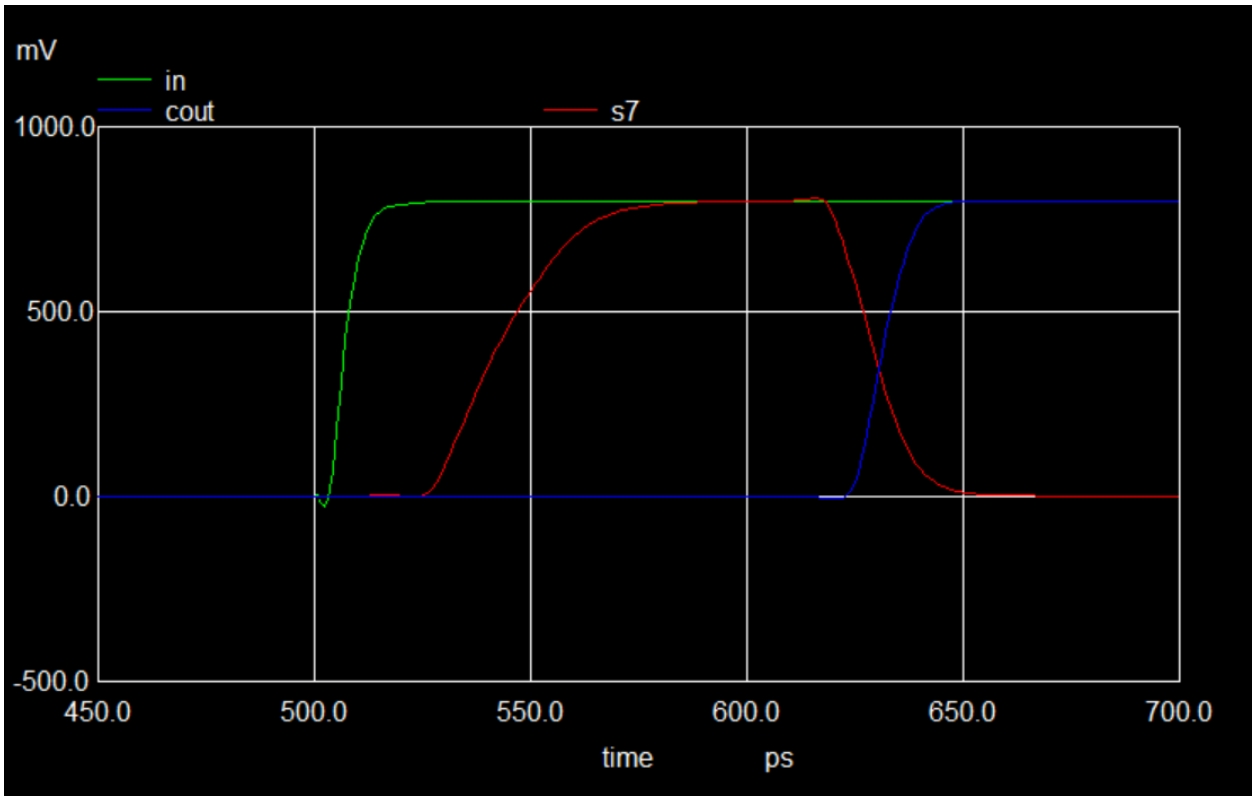

Figure 2.15: Worst Case Delay for Optimized Adder

Final Delay: 124 ps

The delay that I measured for the improved design is 62% of the delay of the unoptimized design. This is a 1.61x speed-up, which was achieved while maintaining the bitwise modularity of the design and not massively increasing the size of the schematic.

Active Energy - Max:

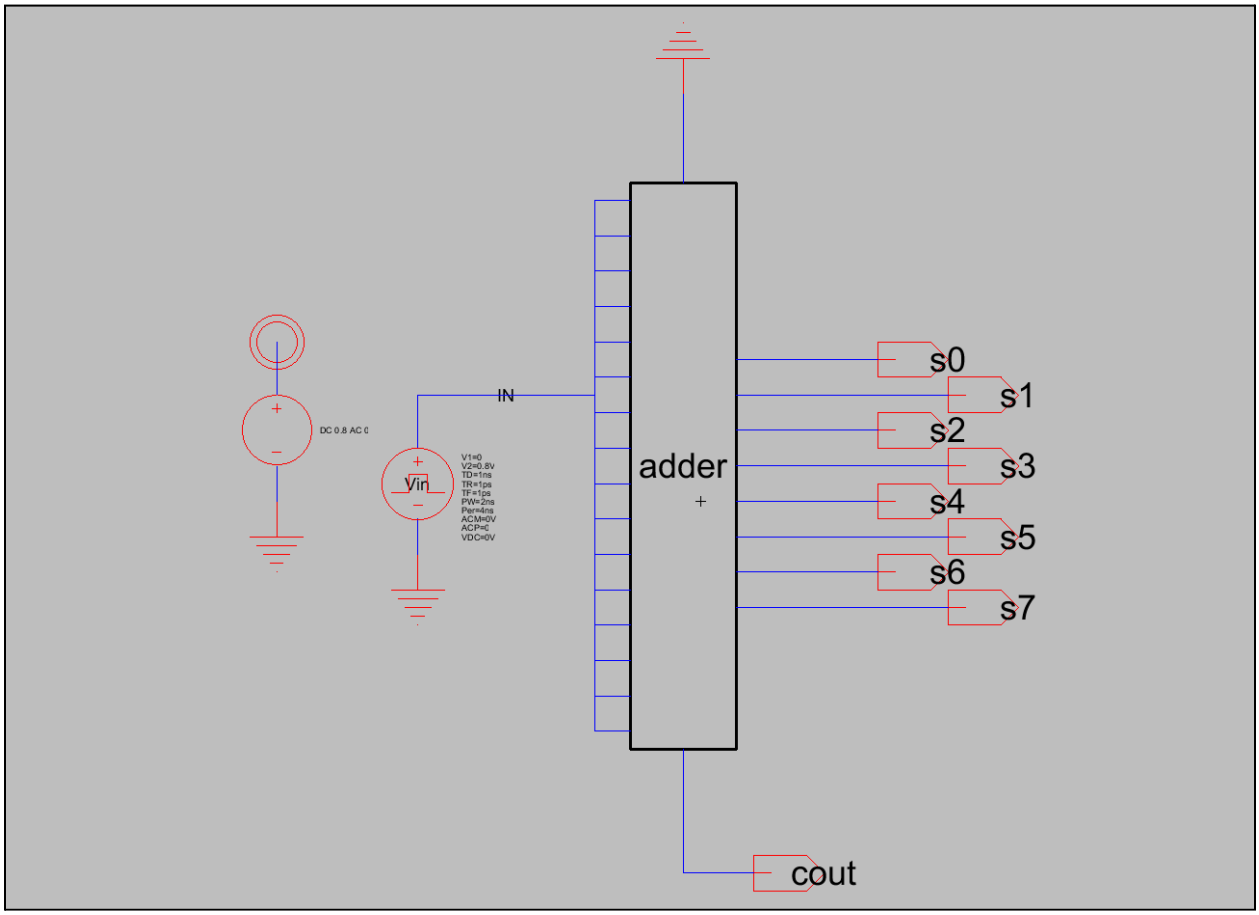

Figure 2.16: Max Active Energy Test Schematic

Integral of current through Vdd during switching: $1.43 * 10^{-15}$ Coulomb

Energy burned: E = Q * V = $1.43 * 10^{-15} C * 0.8 V = 1.144 * 10^{-15} Joules$

Active Energy - Average:

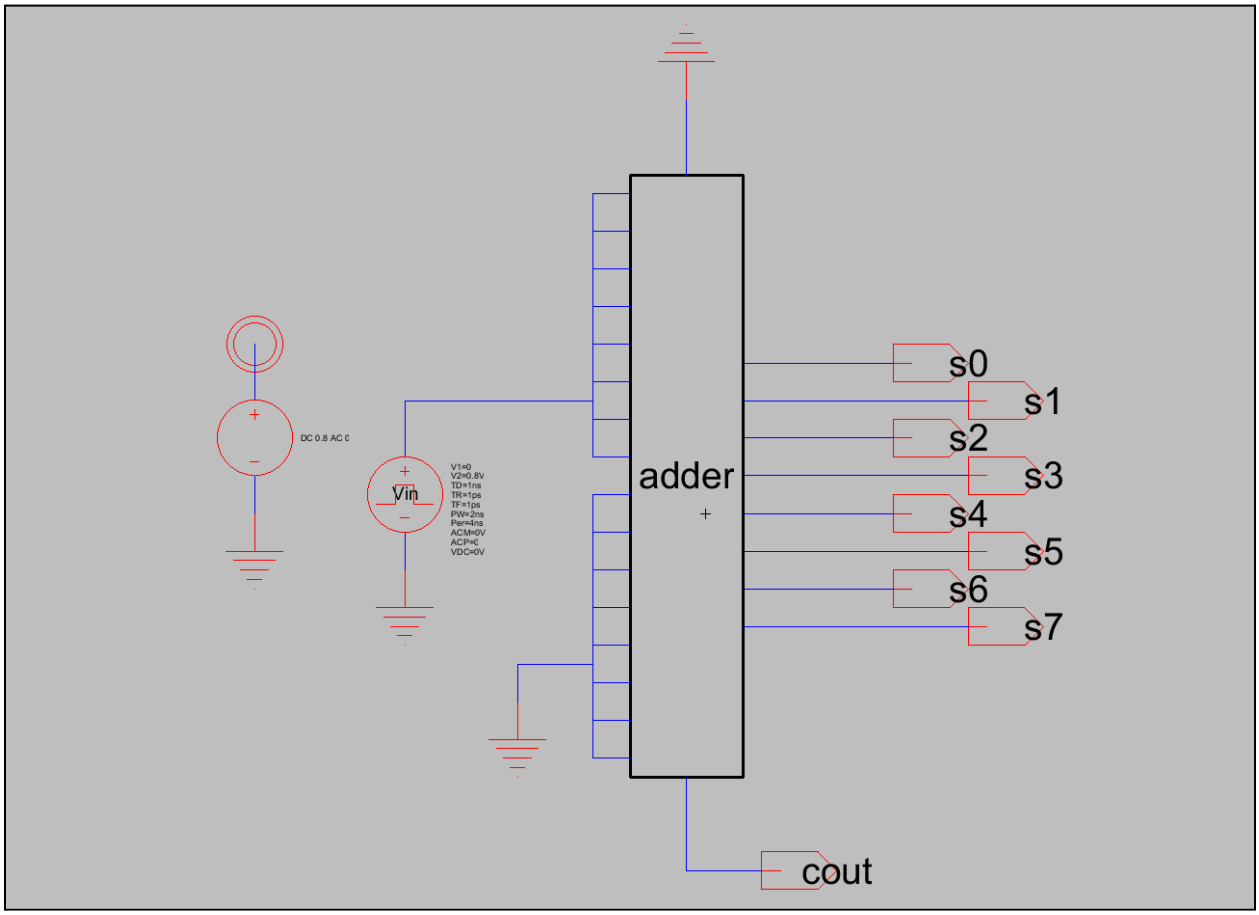

Figure 2.17: Average Active Energy Test Schematic

Integral of current through Vdd during switching: $1.02 * 10^{-15}$ Coulomb

Energy burned: E = Q * V = $1.02 * 10^{-15} C * 0.8 V = 8.16 * 10^{-16} Joules$

Leakage Energy:

Pre-Testing Hypotheses:

COUT Gate Max Leakage:
- A = B = 0
- CIN = 1

COUT Gate Min Leakage:
- CIN = A = 1
- B = 0

XOR2 Gate Max Leakage:
- A = 1
- B = 0

XOR2 Gate Min Leakage:
- A = B = 1

Because I want accurate results, I will empirically determine the max and min leakage current input cases using tests.

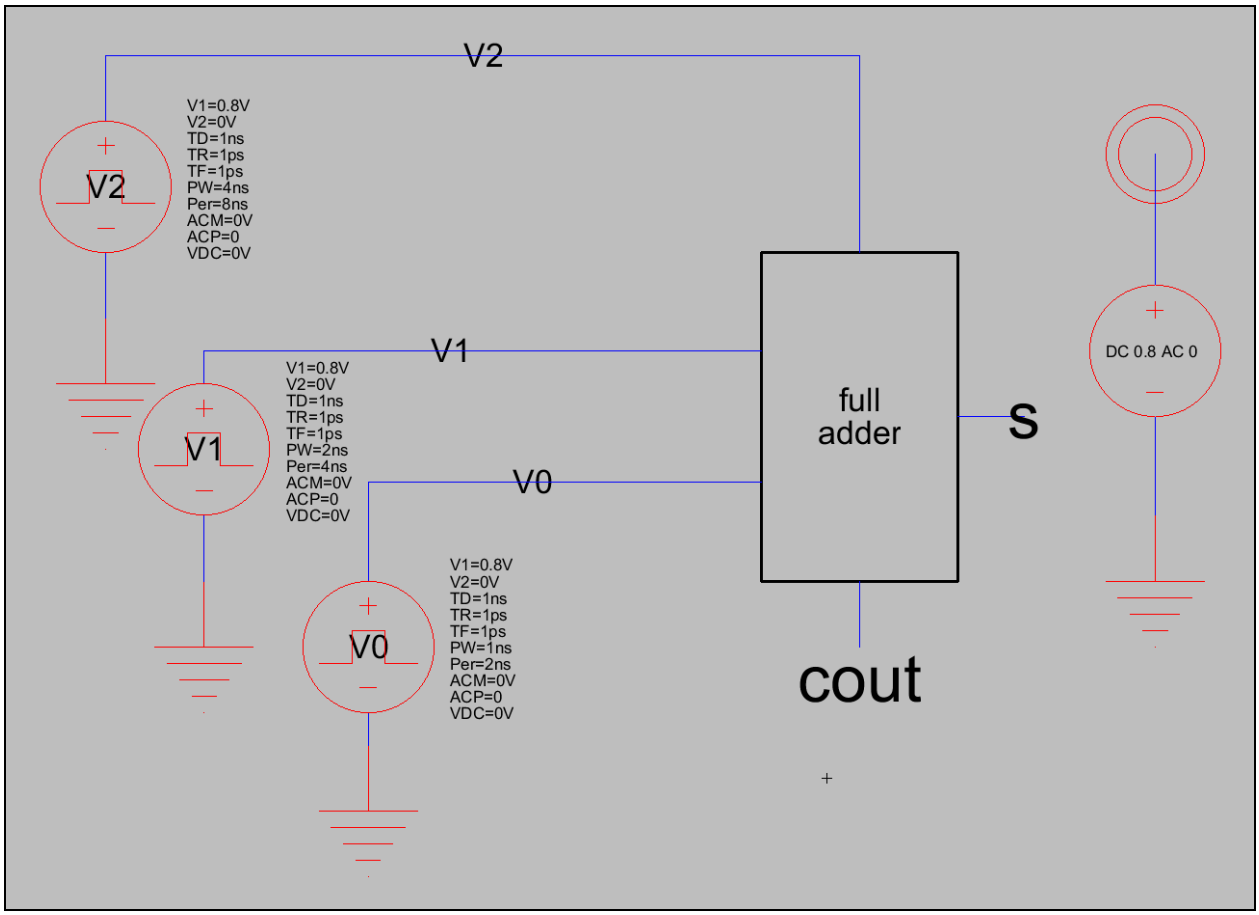

Figure 2.18: Leakage Max/Min Test Circuit

Results:

Min leakage occurs:

- CIN = A = B = 0

Max Leakage occurs:

- CIN = A = 1
- B = 0

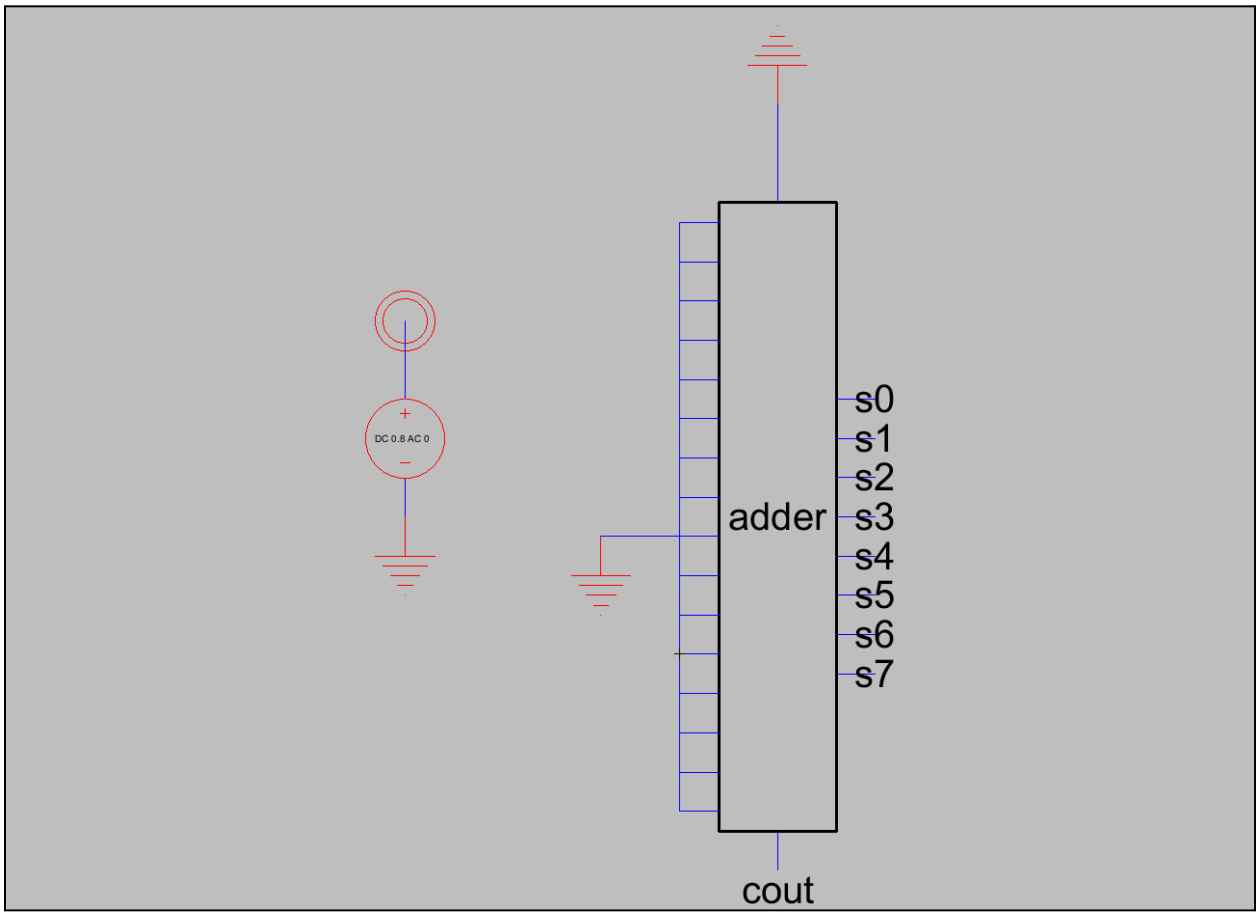

Figure 2.19: Min Leakage Test Schematic

Leakage charge over 1 delay period (124ps): $1.75 * 10^{-17} Coulomb$

Energy burned: E = Q * V = $1.75 * 10^{-17} C * 0.8 V = 1.4 * 10^{-17} Joules$

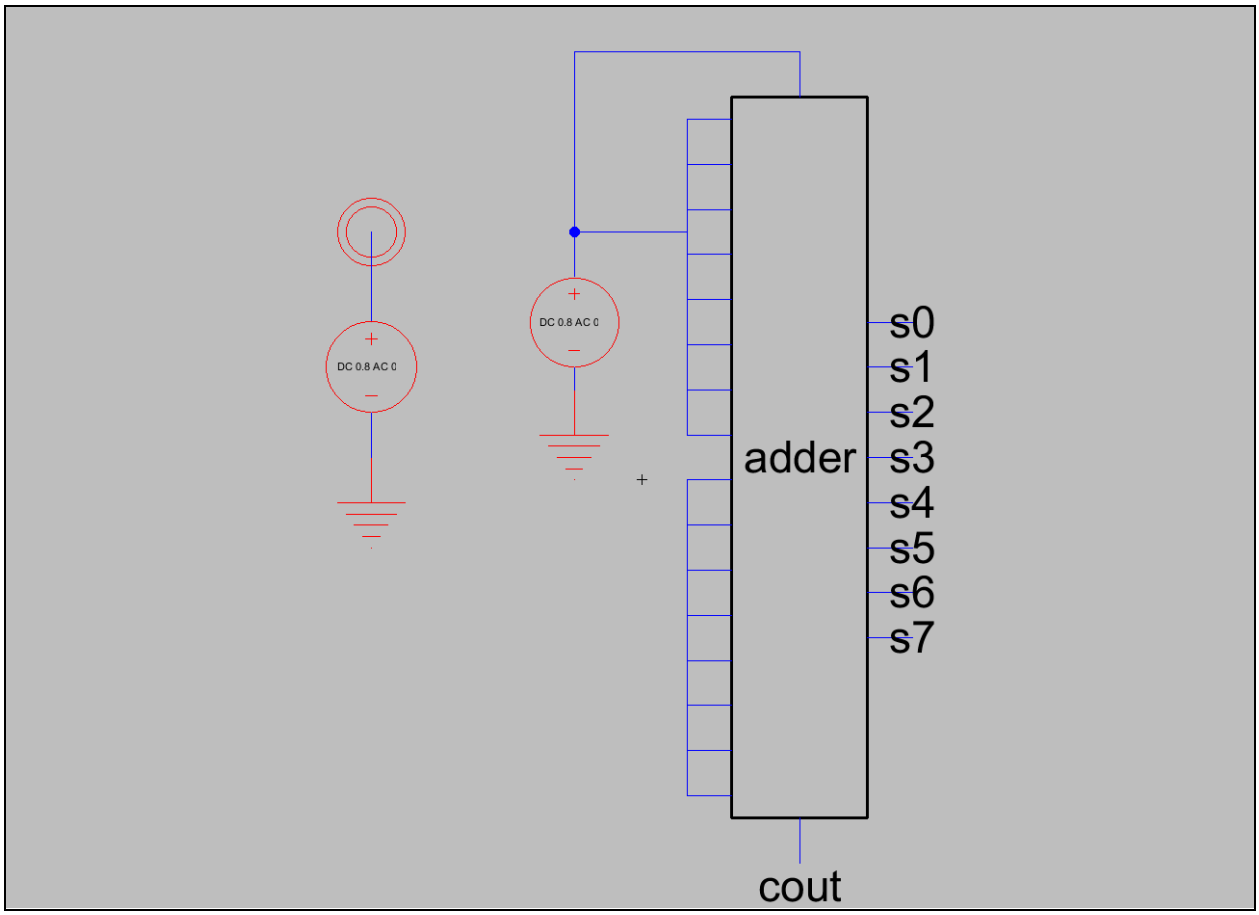

Figure 2.20 Max Leakage Test Circuit Schematic

Leakage charge over 1 delay period (124ps): $1.14 * 10^{-15} Coulomb$

Energy burned: E = Q * V = $1.14 * 10^{-15} C * 0.8 V = 9.12 * 10^{-16} Joules$

Area: (note: all transistors are minimum sized)

XOR2 Gate: 12 transistors
COUT Gate: 14 transistors

There are 2 XOR2 gates and 1 COUT gate per Full Adder.

There are 8 Full Adders in the 8-Bit Adder.

$12 * 2 * 8 = 192$ transistors from XOR2 gates.

$14 * 1 * 8 = 112$ transistors from COUT gates.

Final Area: 112 + 192 = 304

Summary Table:

	Delay (ps)	Max Active Energy (10^{-18}J)	Avg Active Energy (10^{-18}J)	Max Leakage Energy (10^{-18}J)	Min Leakage Energy (10^{-18}J)	Area (Width Units)
Unoptimized	200	1592	1067	32.48	29.8	336
Optimized	124	1144	816	912	14	304

Note: My optimized design is extremely modular and can be broken up into any discrete number of bits - I am very proud of the high speed-up I was able to achieve and I hope you agree that I did a good job, because it took a lot of time.

I, Garrett Kirsch, certify that I have complied with the University of Pennsylvania's Code of Academic Integrity in completing this project.