Phillip Trent
ESE 370
Optimization of 16-bit Adder

1 The Objective

The goal is to optimize a single bit slice of a ripple carry adder in such a way as to minimize the delay of a 16b adder. The single bit slice waveform is shown in Figure 1 with the sum output S and carry out labeled $Cout$.

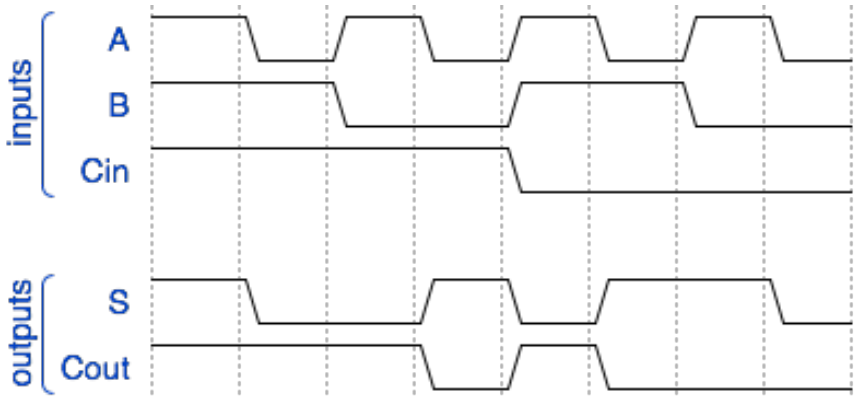


Figure 1: Target waveform for single bit slice

2 Baseline Design

For the baseline design, I used the following equations for sum and carry out logic:

$$S = A \oplus B \oplus Cin$$

$$Cout = AB + ACin + BCin$$

I created a 3-input CMOS XNOR gate. The schematic for which is shown in Figure 2 and then inverted the output for the sum as shown in Figure 3. The carry out was taken directly from the equation above and translated into CMOS logic. I did not use a gate structure because the logic is simple enough to create a custom CMOS design.

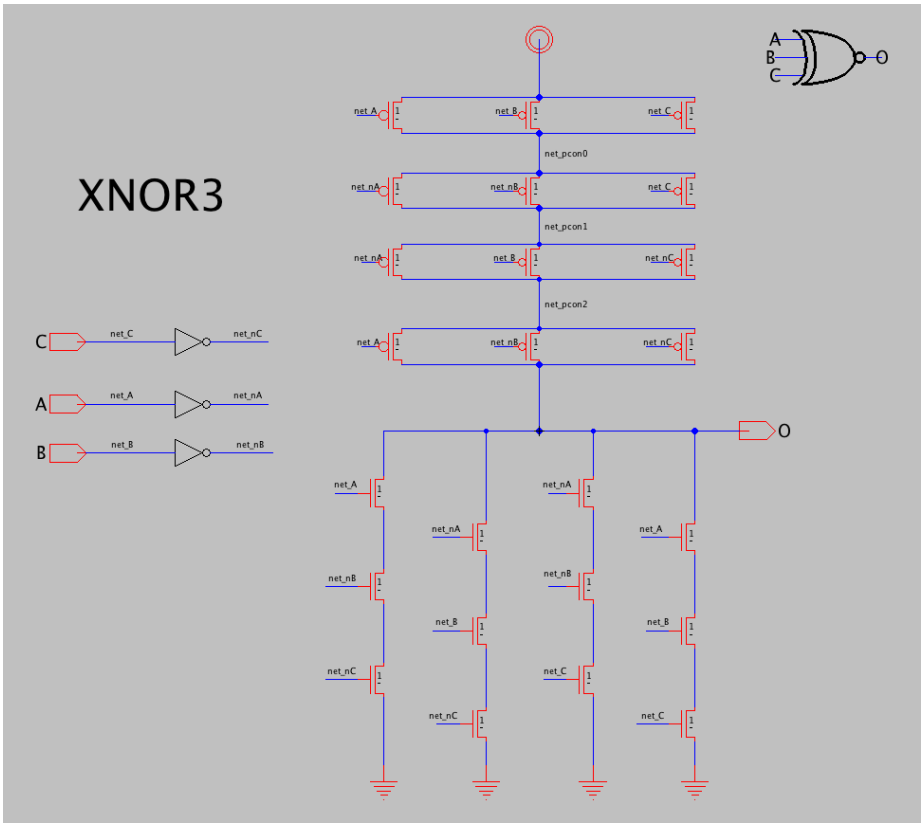


Figure 2: 3-input XNOR Gate

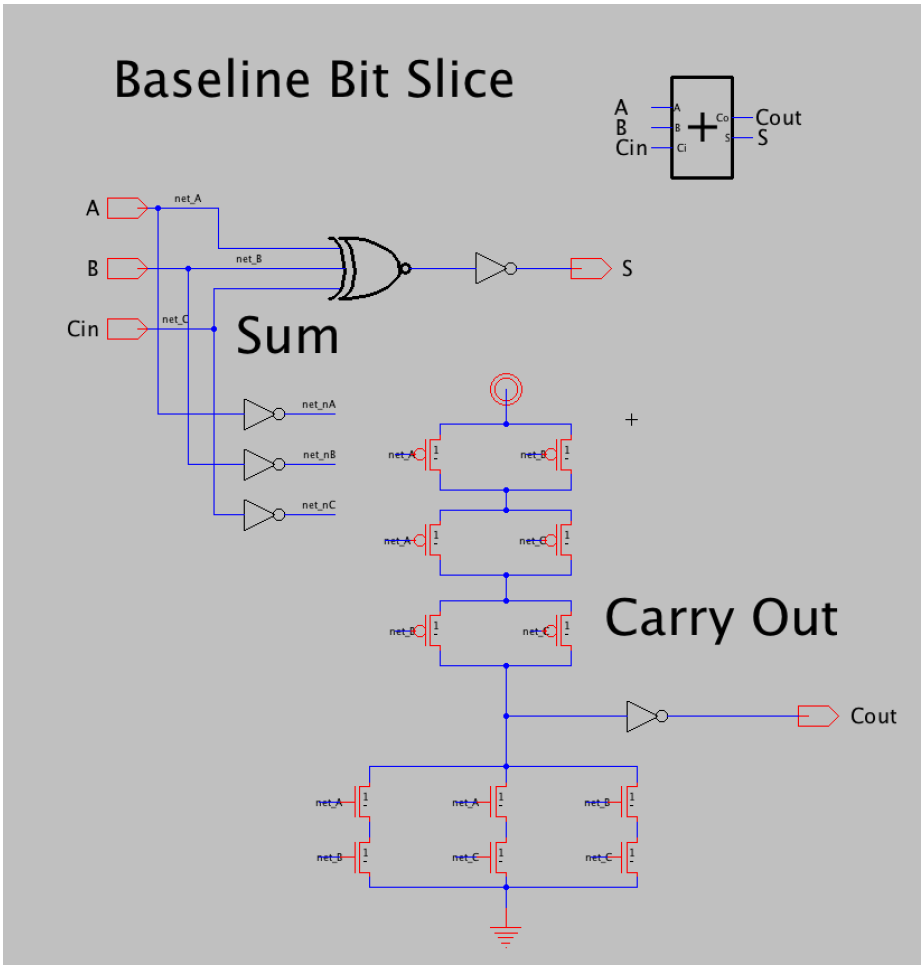


Figure 3: Baseline Bit Slice Cell Schematic

3 Analysis of Baseline Design

3.1 Logical Correctness

3.1.1 Bit Slice

First I validated the logical correctness of a single bit slice. The single bit slice should have the same input/output combinations as Figure 1. You can see the result in Figure 4 and the setup in Figure 5. You'll notice that the output waveform is equivalent to the one in Figure 1.

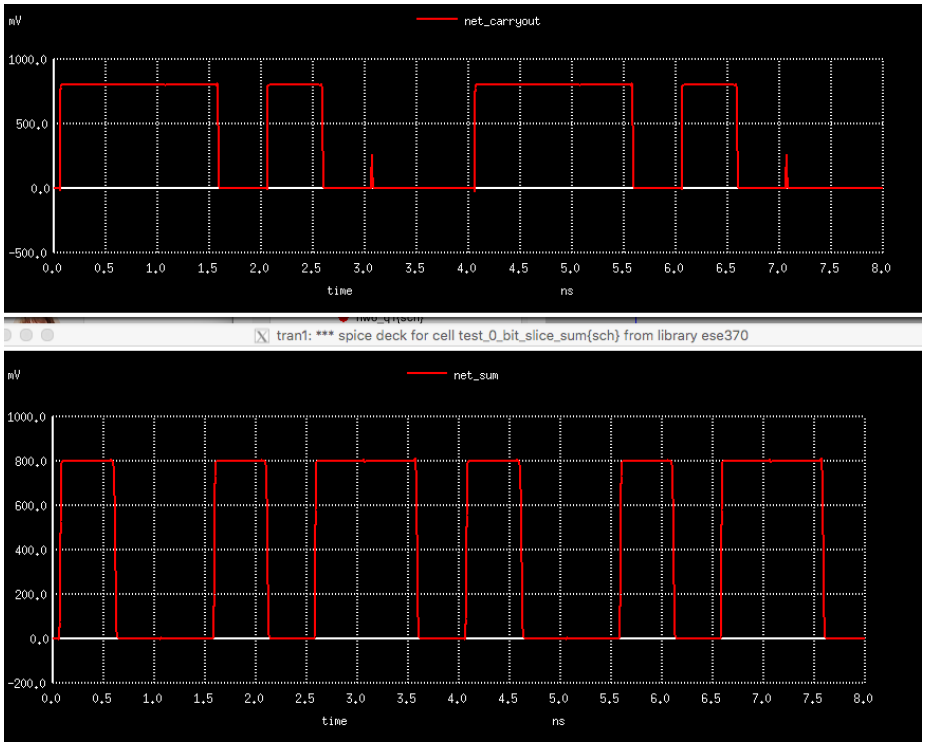


Figure 4: Output waveform for baseline logical test

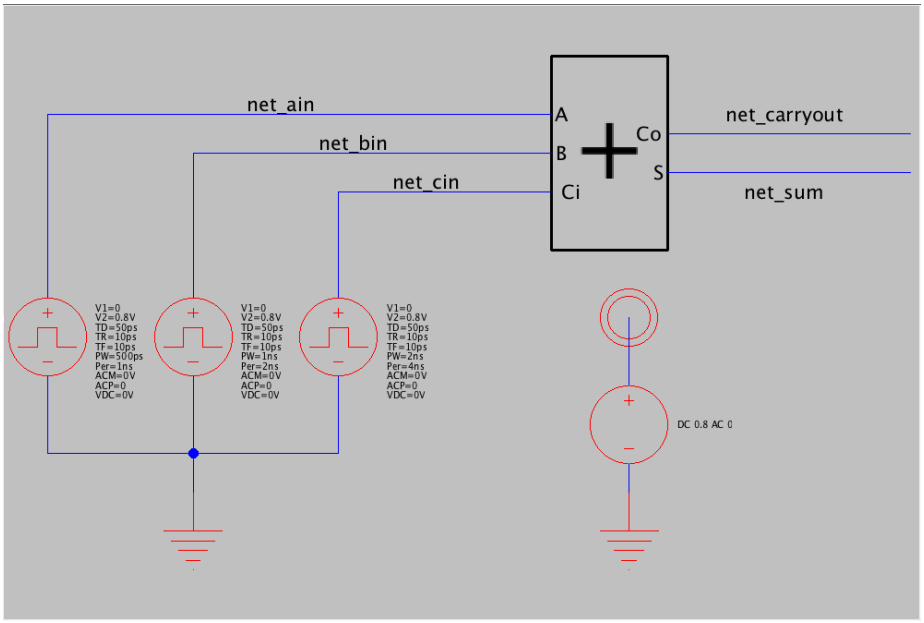


Figure 5: Schematic for testing logical correctness of the baseline

3.1.2 Entire Adder

I validated logical correctness for the entire adder in two ways. The first was by using the same input structure as in Figure 1. The second way was to check the 4 possible inputs for A and B. To do that, I used the inputs $(A, B) = (0xfffff, 0xfffff) \rightarrow (0x0000, 0xfffff) \rightarrow (0xfffff, 0x0000) \rightarrow (0x0000, 0x0000)$. This gives Figure 6.

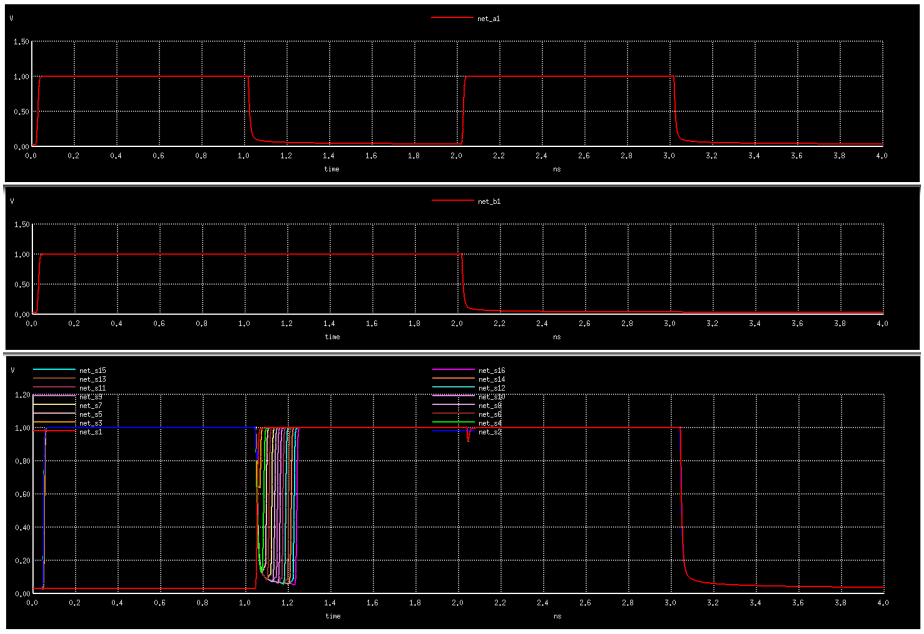


Figure 6: Waveform showing logical correctness of the 1-bit nAdder

3.2 Delay

The max delay of the baseline design is input $A = 0xfffff, B = 0x0000 \rightarrow 0x0001$ is approximately 358τ, which is caused by the rippling carry to the final output and is calculated below.

3.2.1 Calculating Sum Delay: Bit Slice

For the delay of the sum, we calculate the delay from the input across the XNOR gate's internal inverter, and then across the XNOR gate and then across the

final inverter to the output.

$$
\begin{aligned}
d_s &= d_{input \to inv1} + d_{inv1 \to xnor} + d_{xnor \to inv2} + d_{inv2 \to output} \\
d_{input \to inv1} &= R_{output} C_{input} \\
C_{input} &= C_A = C_B = C_{Cin} \\
C_A &= C_{xnor_{in}} + C_{Cout_{in}} \\
&= 4C_0 + 4C_0 \\
&= 8C_0 \\
R_{drive} &= R_0 \text{ because of the single inverter on the output} \\
d_{input \to inv1} &= R_{output} C_{input} \\
&= (1R_0)(10C_0) \\
&= 10\tau \\
d_{inv \to xnor} &= R_{inv} C_{xor_{in}} \\
&= (1R_0)(4C_0) \\
&= 4\tau \\
d_{xnor \to inv2} &= R_{xor} C_{inv2} \\
&= (4R_0)(2C_0) \\
&= 8\tau \\
d_{inv2 \to output} &= R_{inv2} C_{input} \\
&= (1R_0)(8C_0) \\
&= 8\tau \\
d_s &= d_{input \to inv1} + d_{inv1 \to xnor} + d_{xnor \to inv2} + d_{inv2 \to output} \\
&= 8\tau + 4\tau + 8\tau + 8\tau \\
&= 28\tau
\end{aligned}
$$

3.2.2 Calculating Carry Out Delay: Bit Slice

Calculating the delay of the carry out is straightforward. First, we consider the input delay, which is the same as above. Next, we consider the gate and

inverter, and finally the output and its load.

$$
\begin{aligned}
d_{Co} &= d_{input \to gate} + d_{gate \to inv_{output}} + d_{inv_{output} \to output} \\
d_{input \to gate} &= 8\tau \\
d_{gate \to inv_{output}} &= R_{gate} C_{inv} \\
&= (3R_0)(2C_0) \\
&= 6\tau \\
d_{inv_{output} \to output} &= R_{inv} C_{load} \\
&= (1R_0)(8C_0) \\
&= 8\tau \\
d_s &= d_{input \to gate} + d_{gate \to inv_{output}} + d_{inv_{output} \to output} \\
&= 8\tau + 6\tau + 8\tau \\
&= 22\tau
\end{aligned}
$$

3.2.3 Measured Delay: Bit Slice

I graphed the delay and measured from $0.4V$ on the input to $0.4V$ on the output as we have done in previous assignments. The setup is shown in Figure 12. The delay for a single bit slice is measured in the switching case $(A, B, Cin) = (0 \to 1, 0, 0)$ since this only allows one path through the pull up network, and thus avoids a reduction in R_{on} by parallel PMOS transistors. These inputs need to be driven by an equivalent drive of the sum gate, so I created a driver as shown in Figure 10 to drive the inputs to all bit slices and load all sum outputs.
The total delay across the sum is: $63.7879ps$(Figure 7)
The total delay across the carry is: $37.3973ps$(Figure 8)
Calculations

For τ:

$$
\begin{aligned}
R_{on} &= 21454.6\Omega \\
Co &= 0.1694fF \\
2.2\tau &= 2.2R_{on} \times Co \\
&= 2.2 \times 21454.6\Omega \times 0.1694fF \\
&= 7.9957003 \times 10^{-12}s \\
&= 7.9957003ps
\end{aligned}
$$

For sum delay:

$$\begin{aligned} 28\tau &= 28 \times 7.9957003ps \\ &= 28 \times 7.9957003ps \qquad\qquad = 223.87961ps \end{aligned}$$

For carry out delay:

$$\begin{aligned} 22\tau &= 22 \times 7.9957003ps \\ &= 22 \times 7.9957003ps \qquad\qquad = 175.90541ps \end{aligned}$$

These values are much higher than the observed values most likely because of inaccuracies in R_{on} and Co observed in previous work.

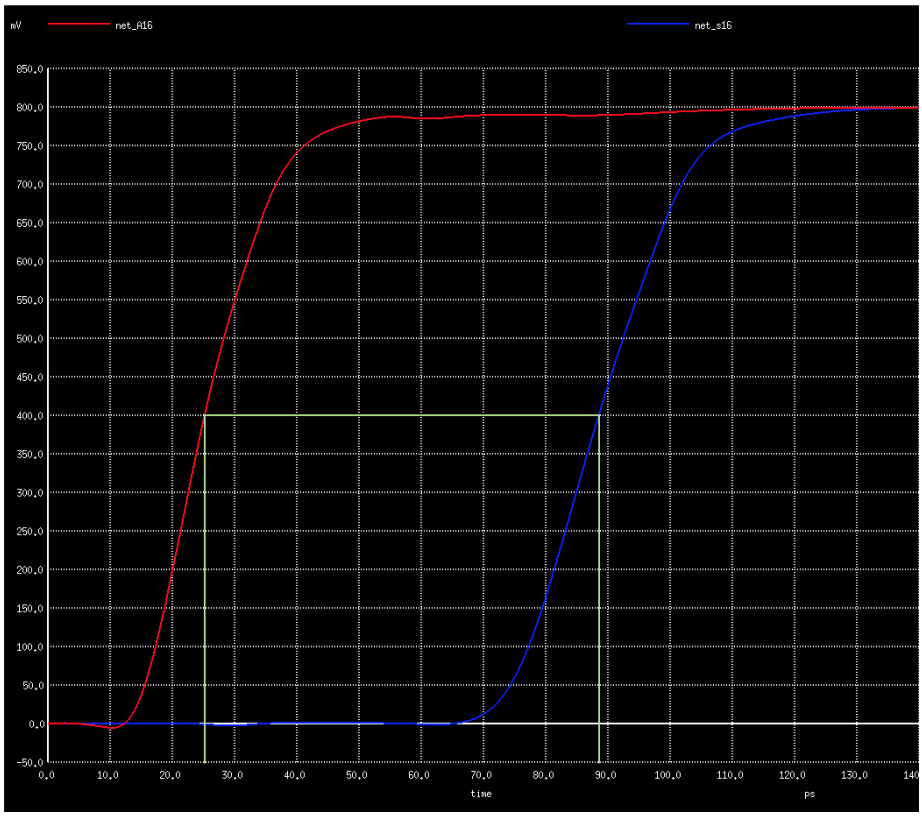


Figure 7: Waveform measuring input to output of baseline sum bit slice

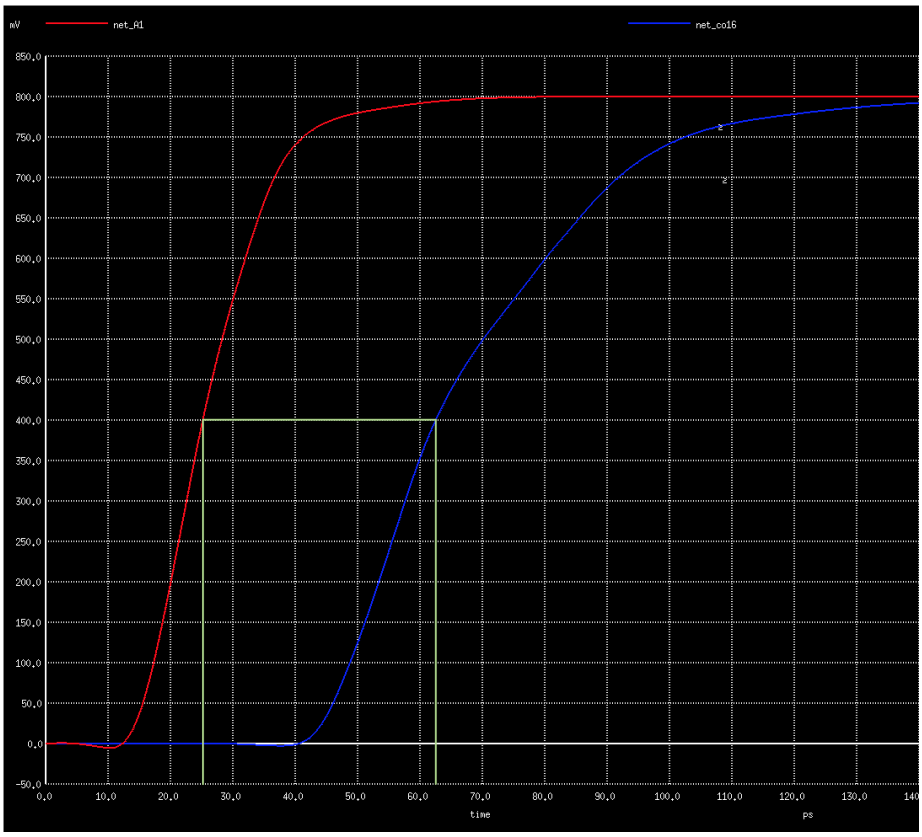


Figure 8: Waveform measuring input to output of carry out across a bit slice

3.2.4 Calculating Worst Case Delay: 16 Bit Adder

The worst case delay occurs when the carry is propagated through the entire adder changing every sum output value, and the sum outputs rise. That is, in the case $(A, B, Sum) = (0xffff, 0x0001 \rightarrow 0x0000, 0x0000 \rightarrow 0xffff)$. Intuition led me to believe that the last value to rise should be the carry out. However, because of the large fan-in of the baseline sum gate, the sum output must drive a large capacitance, slowing the rise time as seen above in the calculated sum τ value. Therefore, the worst case delay is the time it takes for the carry out to propagate through 15 bit slices plus the time it takes for sum 16 to rise.

For worst-case delay:

$$\begin{aligned} d &= 15 \times d_{carry} + d_{sum} \\ &= 15 \times 22\tau + 28\tau \\ &= 358\tau \end{aligned}$$

3.2.5 Measuring Worst Case Delay: 16 Bit Adder

The worst-case delay was measured on the waveform in Figure 9 from $0.4V$ to $0.4V$, and the resulting worst case delay is $667.123ps$. Comparing this to the calculated $358 \times 7.9957003ps = 2.8624607ns$ shows the disparity caused my a wrong C_0 and R_{on}.

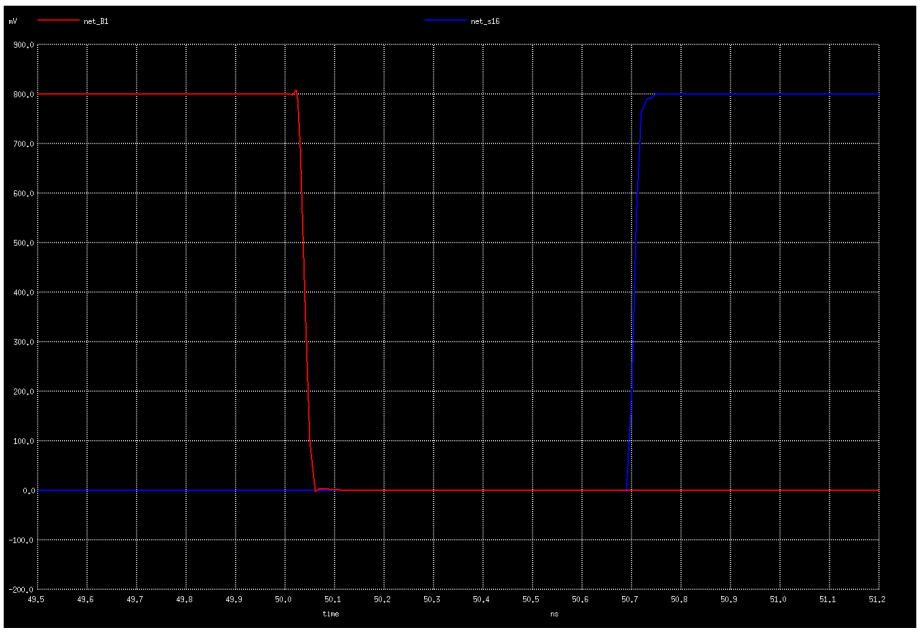


Figure 9: Switching input compared to 16^{th} sum output for worst-case delay

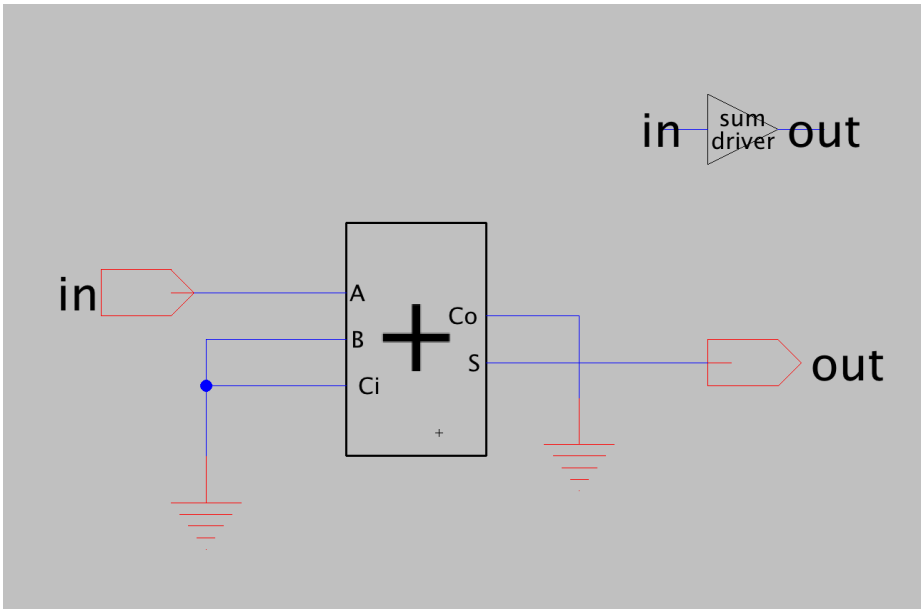


Figure 10: Schematic for a driver and load based off of the sum cell

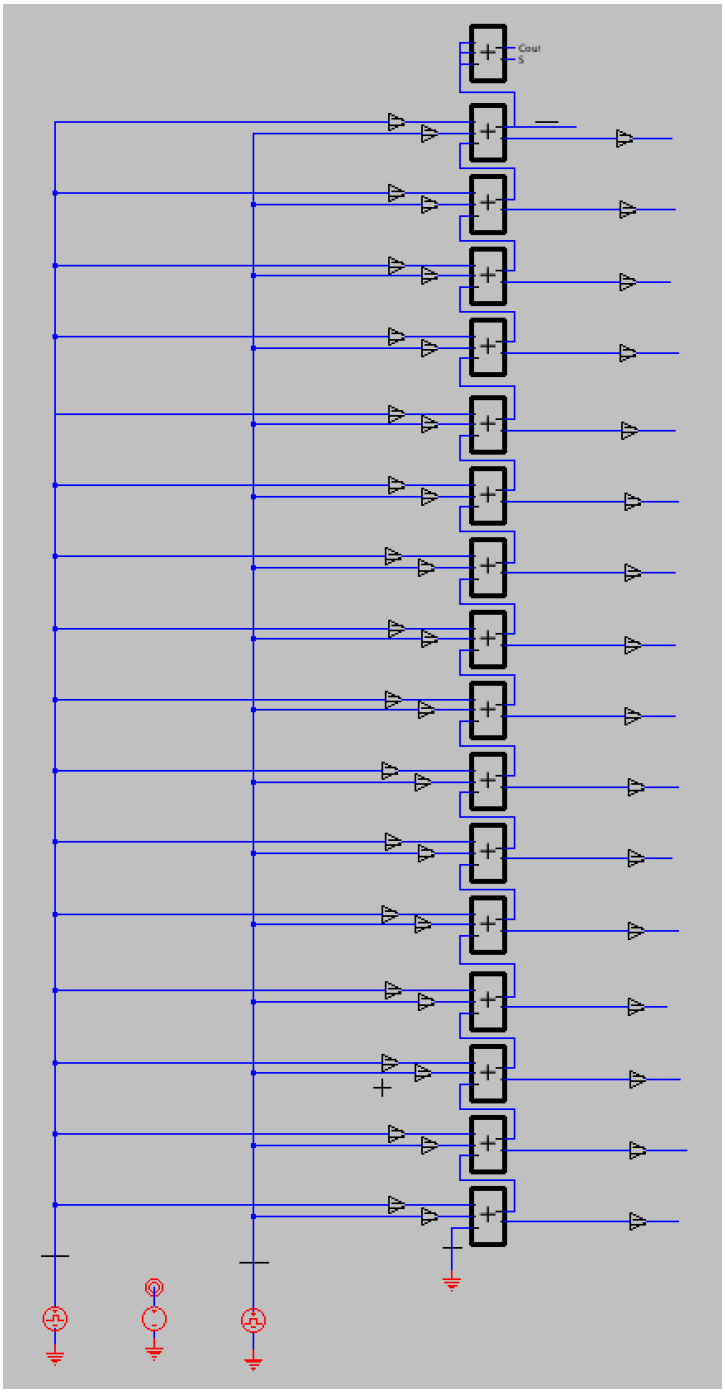


Figure 11: 16bit Adder

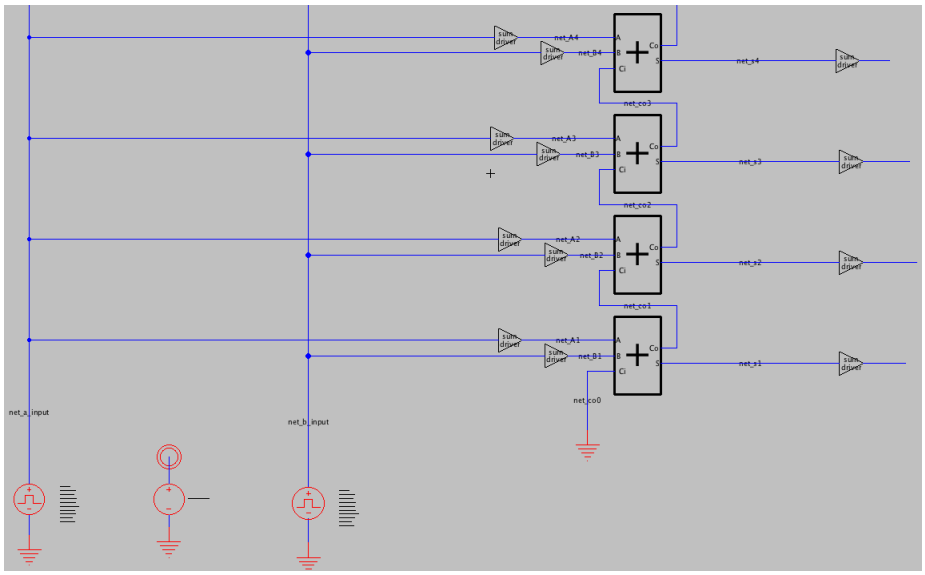


Figure 12: Zoom of 16bit Adder

3.3 Design Metrics

3.3.1 Area

The XNOR gate has 30 transistors. The baseline bit slice has 2 inverters which each have 2 transistors and the logic for the carry out, which has 12 transistors. In total, since every transistor is width 1 minimum unit, this gives an area of $30 + 2 + 2 + 12 = 46$ units for each bit slice. There are 16 of these, and thus we have the following:

$$Area = 16 \times 46 = 736 units$$

3.3.2 Delay

See above.

3.3.3 Active Energy

Maximum switching energy $= (0.582845 fW) \times 16 = 9.32552 fW.$
Average switching energy $= \frac{3.90943 fW}{8 inputs} \times 16 = 7.81886 fW$

3.3.4 Leakage Energy

1. Minimum leakage energy $= 0.507343 aW \times 16 = 8.117488 aW$
2. Maximum leakage energy $= 23.1757 aW \times 16 = 370.8112 aW$

3.4 Metric Evaluation

Area
Area was calculated crudely based on the suggested method in the handout. This is noted as crude, but less important for our analysis.
Delay
Delay was calculated with the τ approximation and measured in ngspice. The delay was only measured from the start of the input to the final rise of the last carry out since that is the longest path through the adder.
Active Energy
Maximum switching energy was evaluated by the schematic in Figure 13 and was found to be the case when the inputs switched from $(A, B, C) = (0 \rightarrow 1, 0, 0)$. To do this, I evaluated the 8 possible cases and multiplied by 16 because of the 16 gates dissipating energy.
Average switching energy was evaluated by the schematic in Figure 13. For this evaluation, I integrated over the entire input sequence from all 0 inputs to all 1 inputs and divided by the number of inputs. This gives the average energy dissipation for a single bit slice given a random selection of inputs. I also multiplied this by 16 to model the entire adder's energy dissipation.
Leakage Energy
Both maximum and minimum leakage energy were found by plotting the current into a single-bit adder (Figure 14 and 15). As one can see, the minimum occurs when all inputs are high and the maximum occurs around $0.8ns$, which is the input case $(A, B, C) = (0, 1, 1)$.

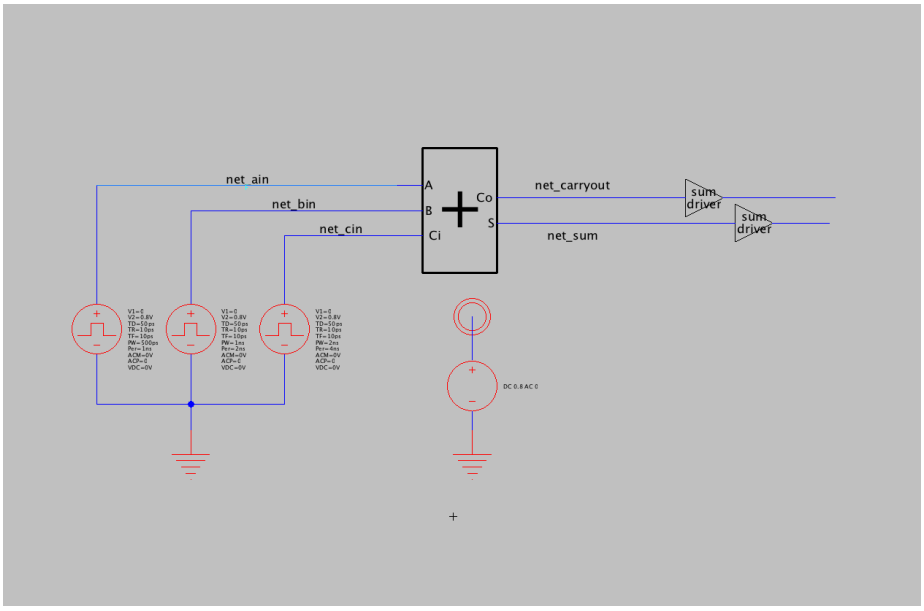


Figure 13: Schematic for evaluating energy requirement

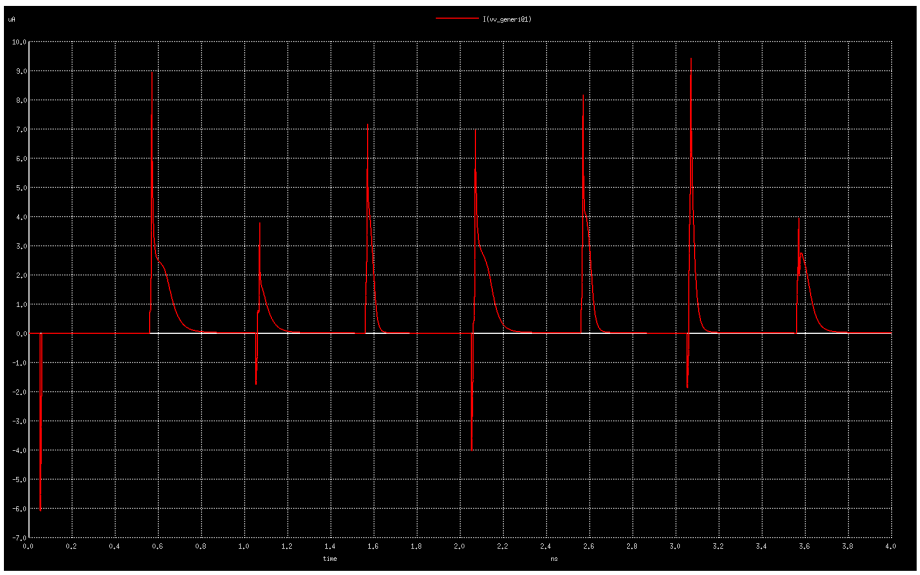


Figure 14: Simulation of current through adder

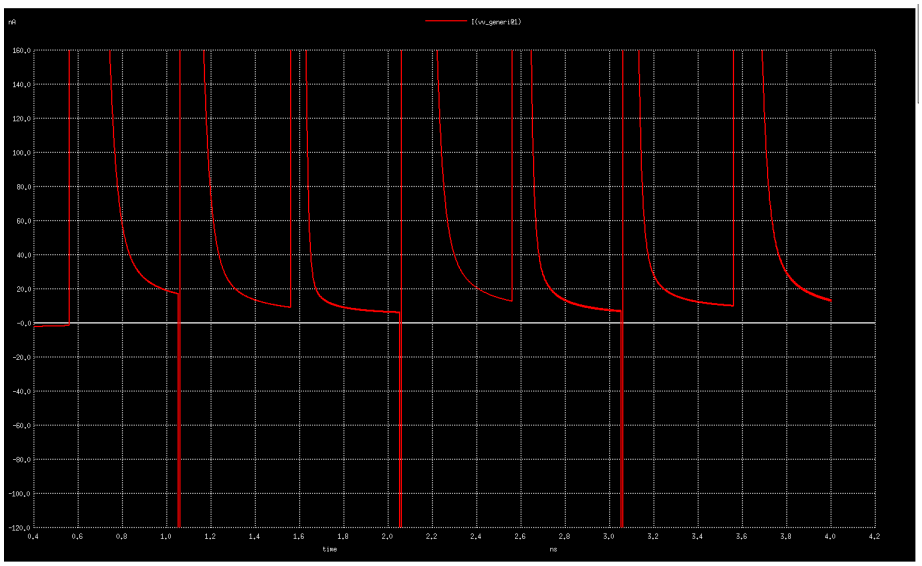


Figure 15: Zoomed simulation of current through adder

4 Optimized Design

The optimized design has four components, which are built into two different 1-bit adders. They will be referenced as Carry (Figure 16), nCarry (Figure 17), Sum (Figure 18), and nSum (Figure 19), which combine to build a 1-bit Adder (Figure 20) and a 1-bit nAdder (Figure 21). Those adders are cascaded to complete the full 16-bit Adder (Figure 22, Figure 23, Figure 24)The prefix n indicates that the module takes as an input $\overline{C}$ as opposed to C. The equations for the outputs are as follows:
Sum:

$$S = \overline{A \oplus \overline{(B \oplus C_{in})}}$$

nSum:

$$S = \overline{A \oplus (B \oplus \overline{C_{in}})}$$

Carry: (consumes the carry in and generates the compliment of the carry out)

$$\overline{C_{out}} = AB + ACin + BCin$$

nCarry: (consumes the compliment of the carry in and generates the carry out)

$$C_{out} = \overline{A}\ \overline{B} + \overline{A}\ \overline{C_{in}} + \overline{B}\ \overline{C_{in}}$$

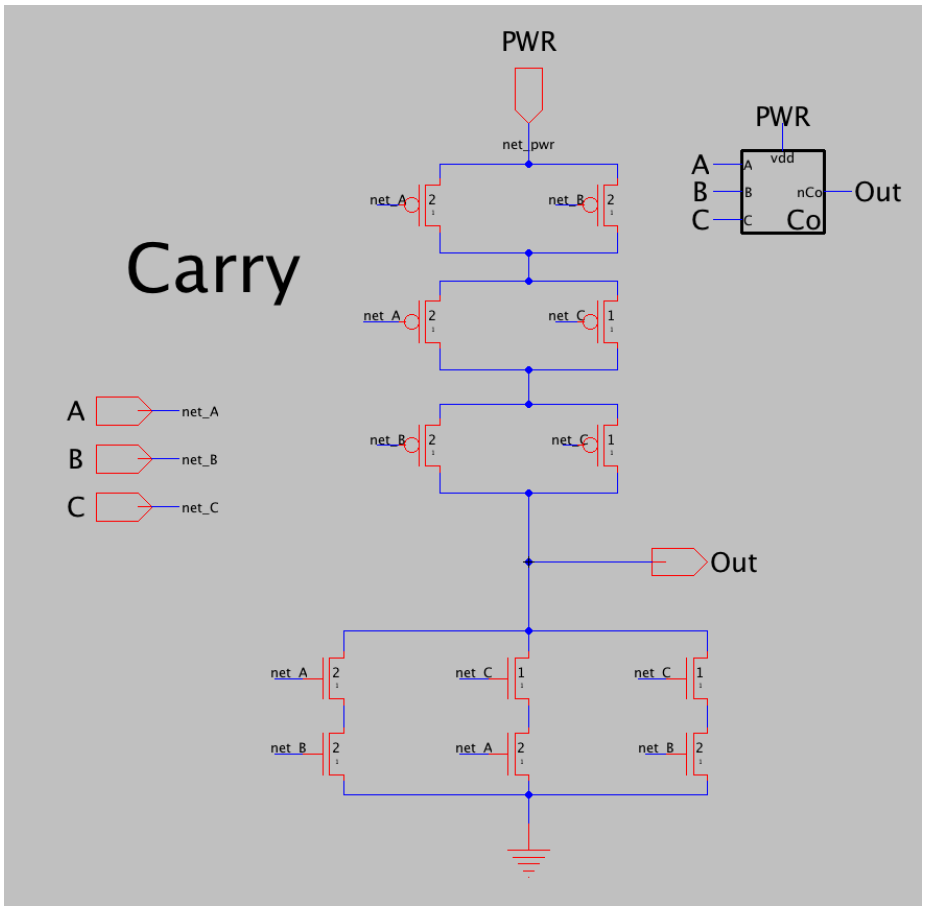


Figure 16: Schematic for Optimized Carry

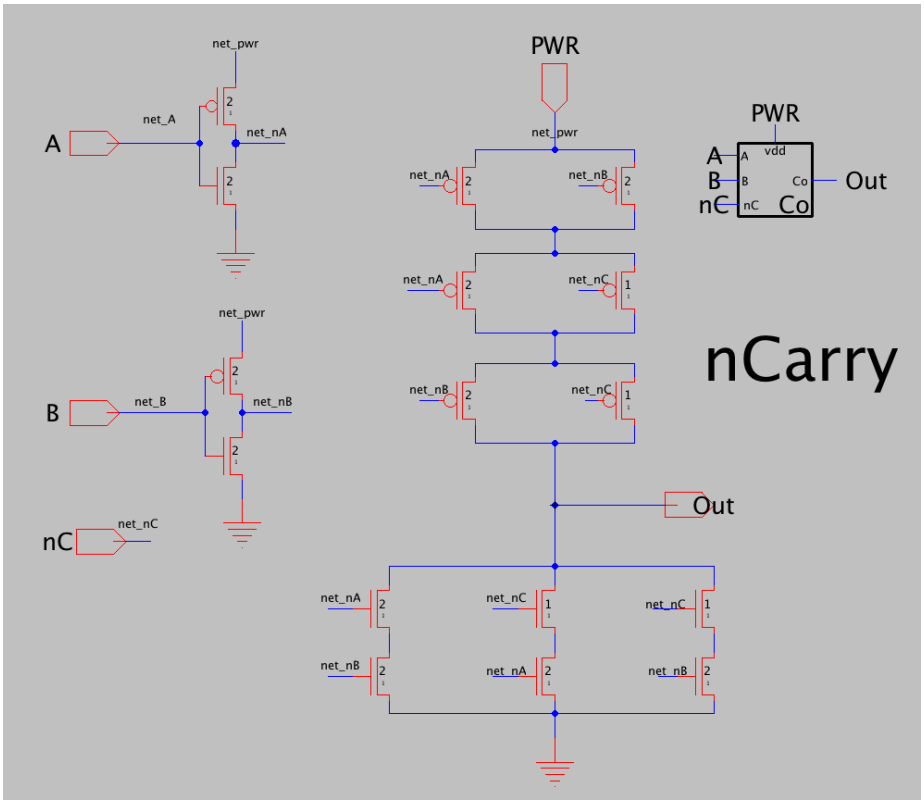


Figure 17: Schematic for Optimized nCarry

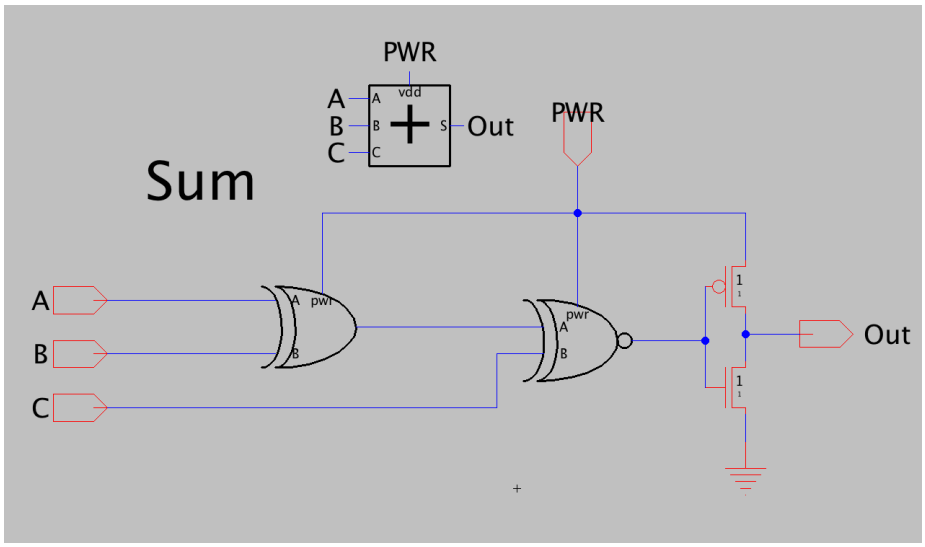


Figure 18: Schematic for Optimized Sum

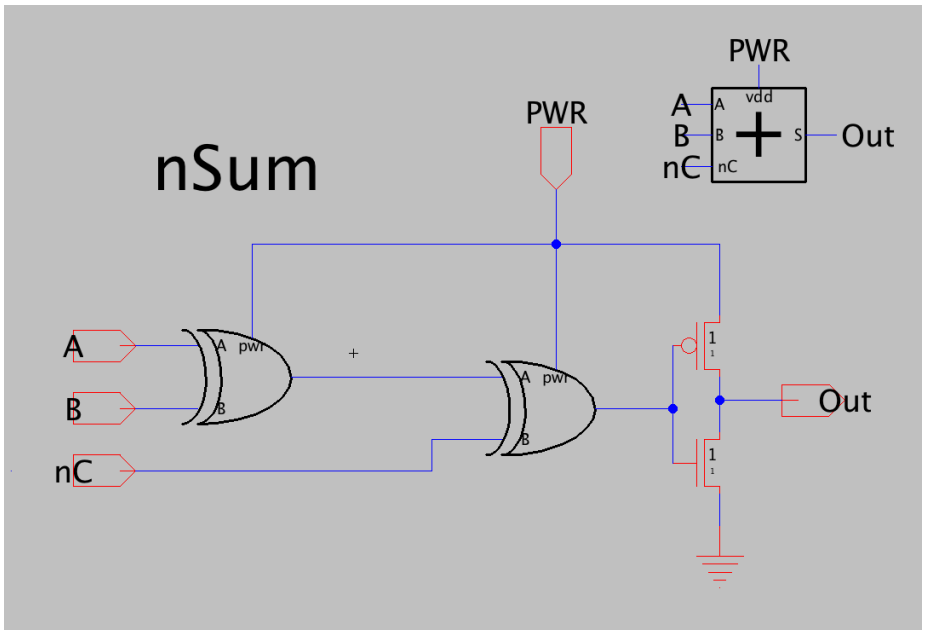


Figure 19: Schematic for Optimized nSum

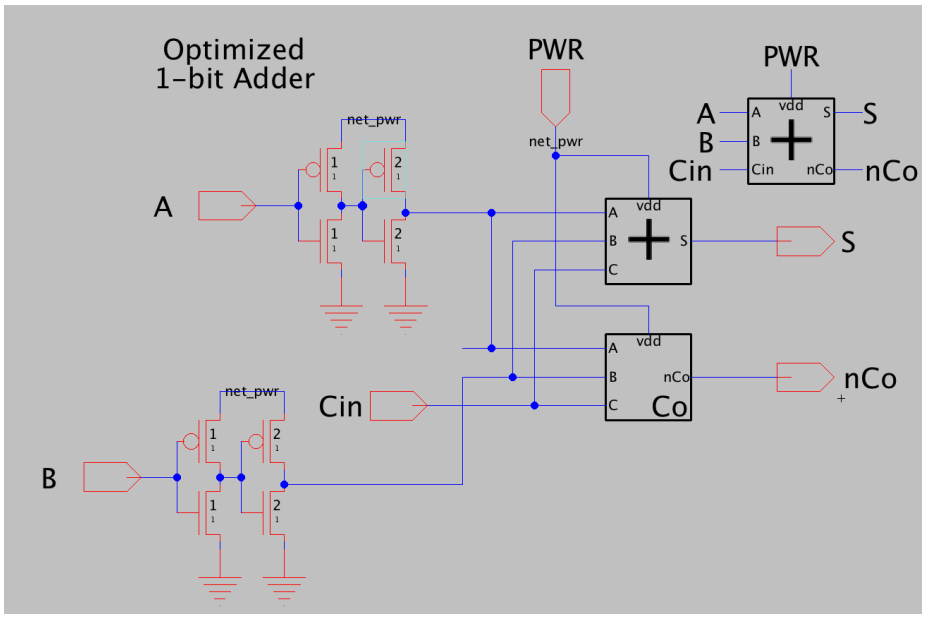


Figure 20: Schematic for 1-b Adder

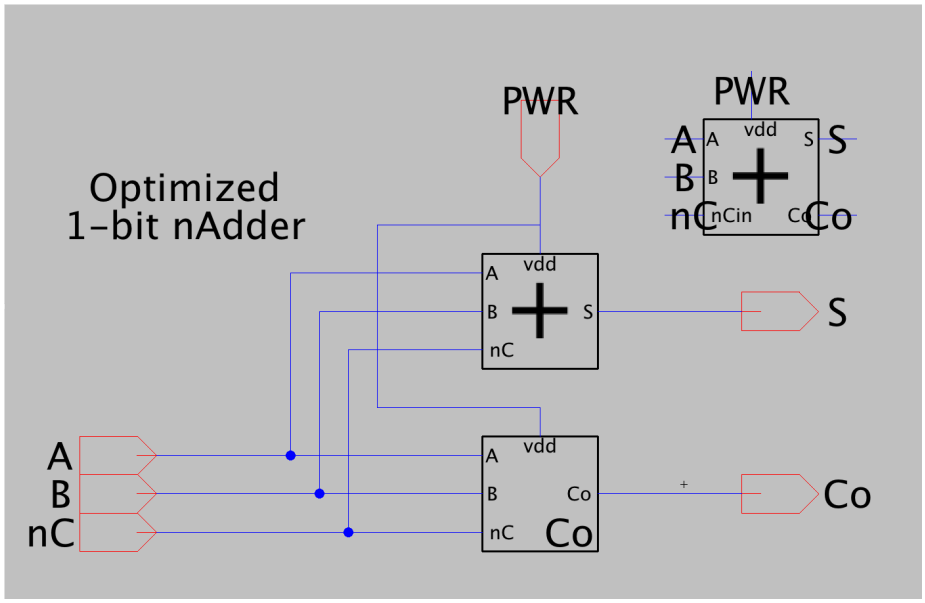
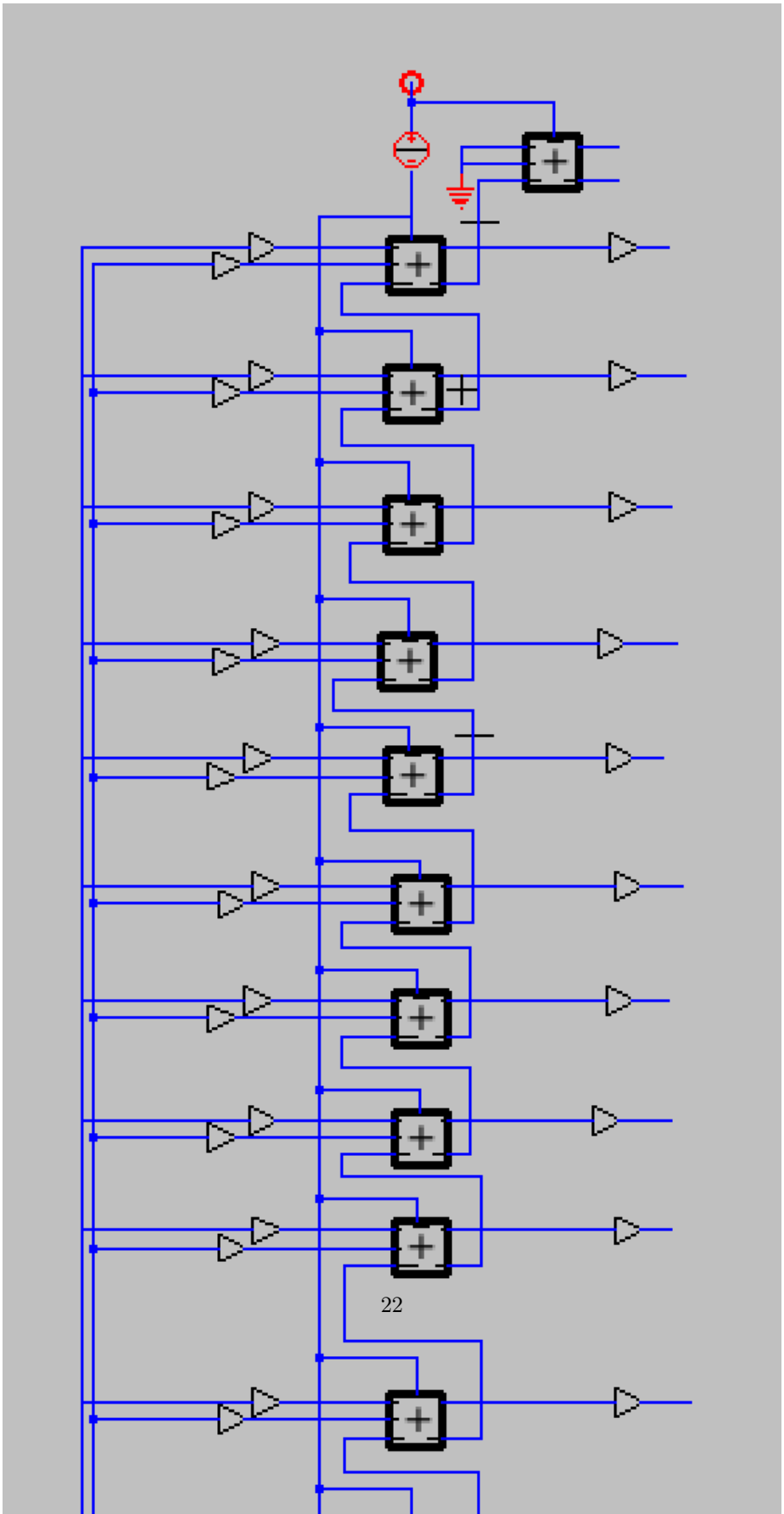


Figure 21: Schematic for 1-b nAdder



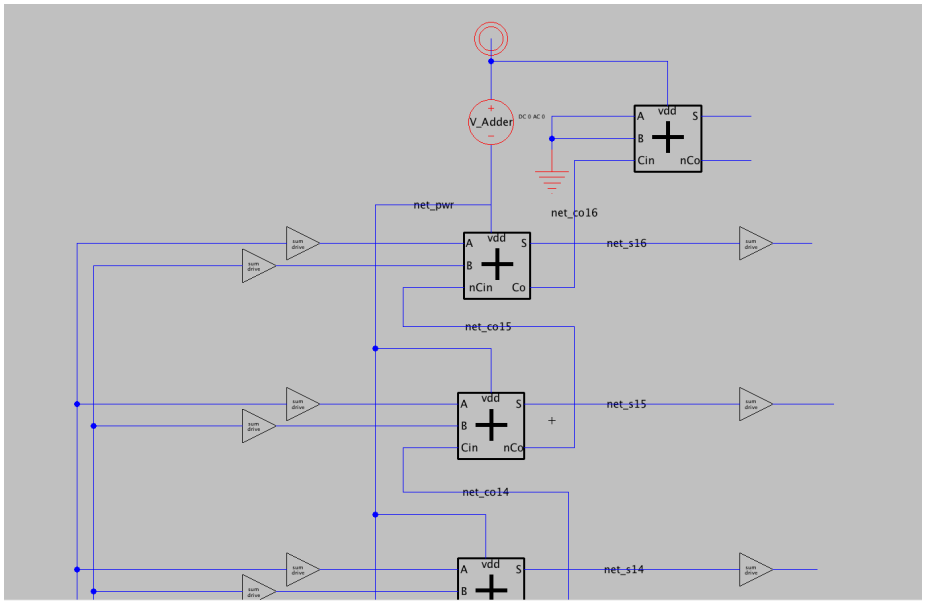


Figure 23: Portion of schematic for entire 16-bit adder

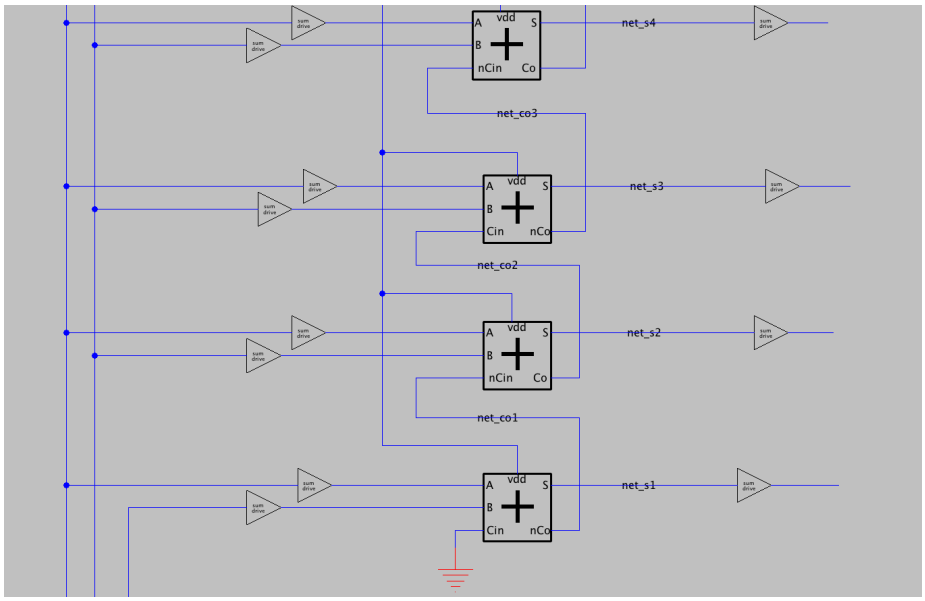


Figure 24: Portion of schematic for entire 16-bit adder

5 Analysis of Optimized Design

5.1 Optimizing

5.1.1 The Optimization Process

I went through a series of optimizations before arriving at the final design, checking the decrease in delay at each stage. Below is an enumeration of each optimization and the resulting measured delay across the entire adder.

1. Replace 3-input XOR with pass transistor logic XOR cascade ($667.123ps \rightarrow 528.022ps$)
2. Change $V_{dd} = 0.8V \rightarrow V_{dd} = 1V$ ($528.022ps \rightarrow 309.259ps$)
3. Re-order inputs to Carry and nCarry to reduce the C_{diff} for the later input ($309.259ps \rightarrow 295.593ps$)
4. Increase width of transistors with A and B inputs for Carry and nCarry ($295.593ps \rightarrow 274.925ps$)
5. Add step-up stages for A and B inputs for Carry and nCarry ($274.925ps \rightarrow 280.286ps$)*
6. Add step-up stages for A and B inputs for 1-bit Adder ($274.925ps \rightarrow 271.282ps$)
7. Add step-up stages for A and B inputs for 1-bit nAdder ($271.282ps \rightarrow 271.864ps$)*

Above are listed all the design optimizations that I considered, implemented, and ephemerally tested. Not all of these resulted in a decrease in delay (namely, the ones marked with *), and thus only the optimizations that decreased delay were used. Below are the schematics for the new XOR (Figure 25) and XNOR (Figure 26) gates. Notice that these gates are NOT regenerative. I knew how I would be cascading them in such a way that the B input is never driven by another pass transistor, and the A input is restored by the gate. Furthermore, as you can see in Figure 18 and Figure 19 above, the output of the final XOR or XNOR is regenerated by a final inverter.

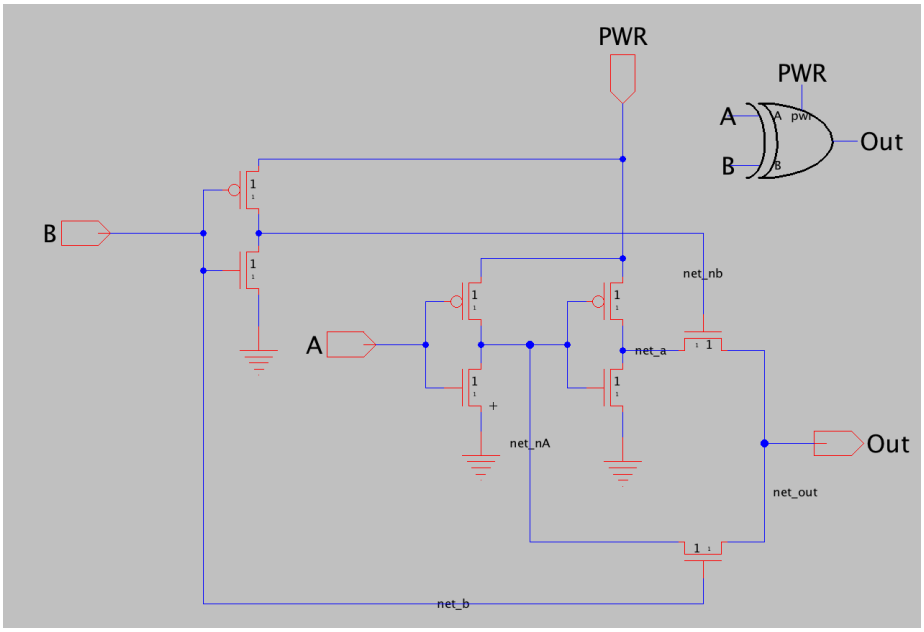


Figure 25: Schematic for XOR with pass transistor logic

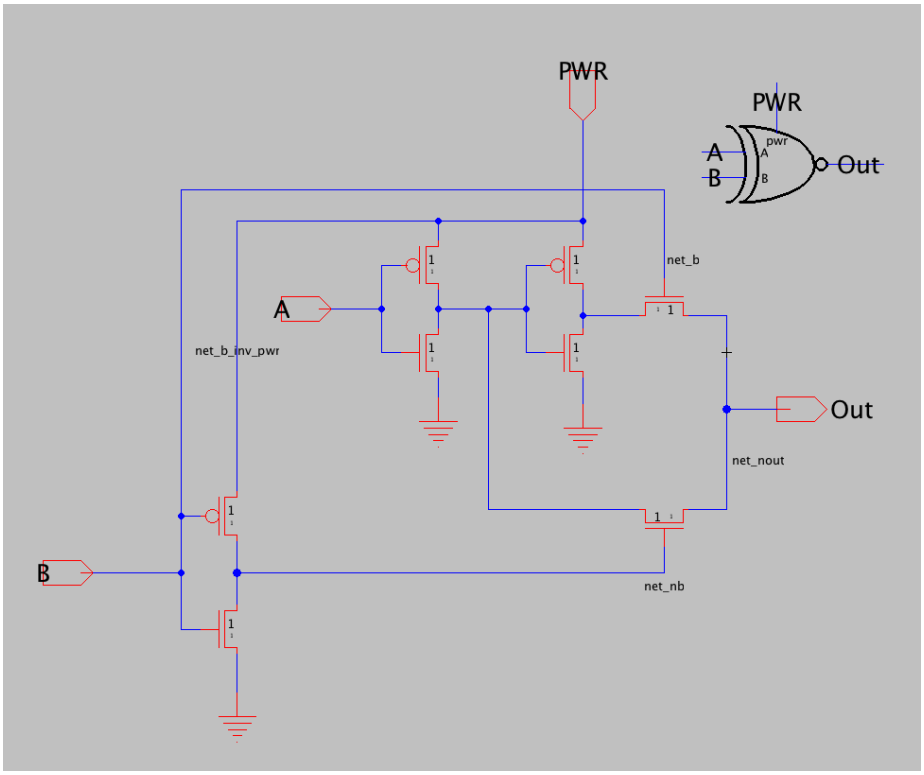


Figure 26: Schematic for XNOR with pass transistor logic

5.2 Logical Correctness

For all but the 16-bit adder, I checked every possible permutation of inputs. For the Sum, the output should be as shown in Figure 27. For carry out, the target waveform is shown in Figure 28.

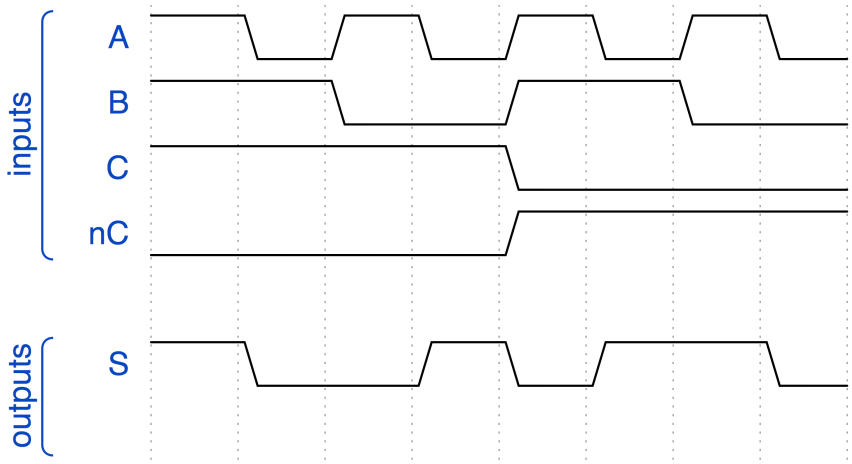


Figure 27: Target waveform for sum logic

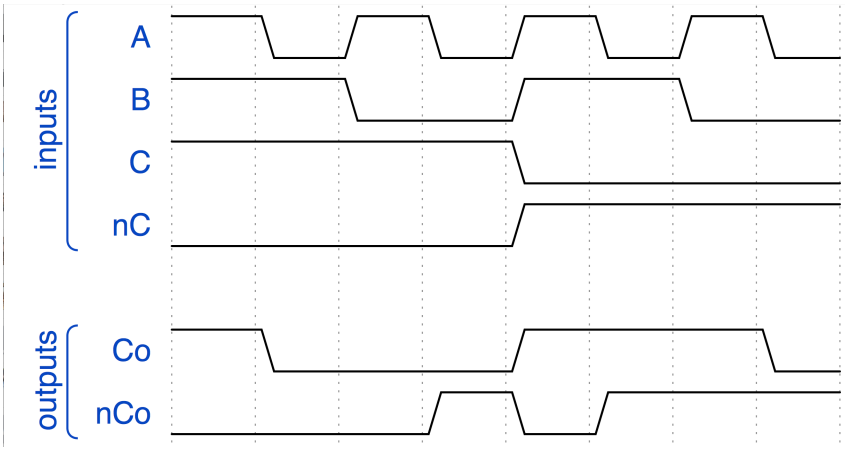


Figure 28: Target waveform for carry out logic

5.2.1 Sum

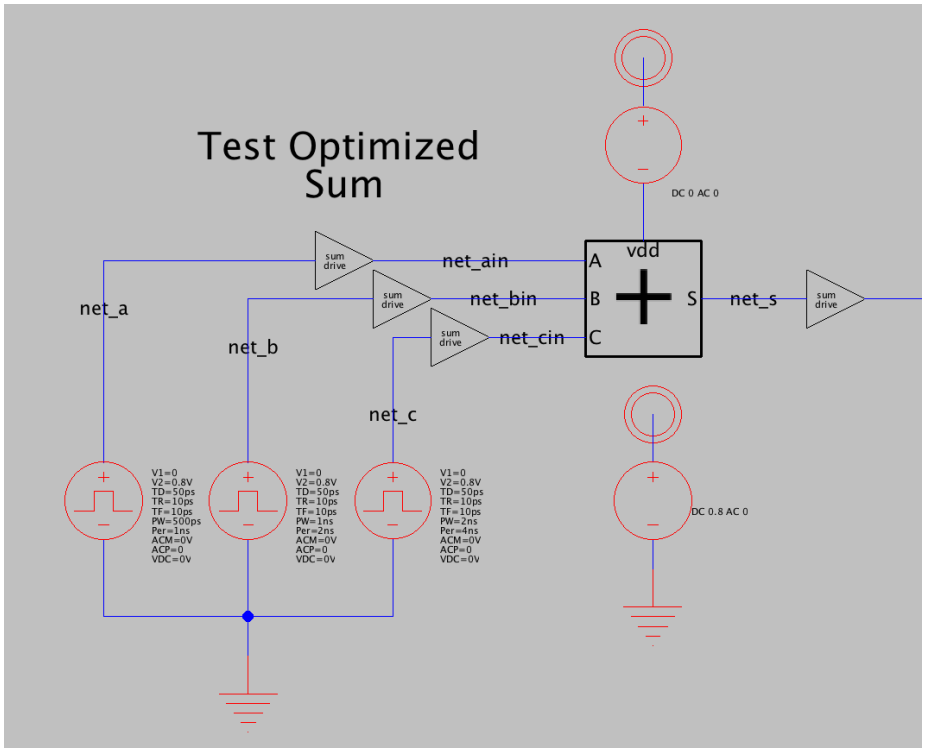


Figure 29: Schematic used to test logical correctness of Sum

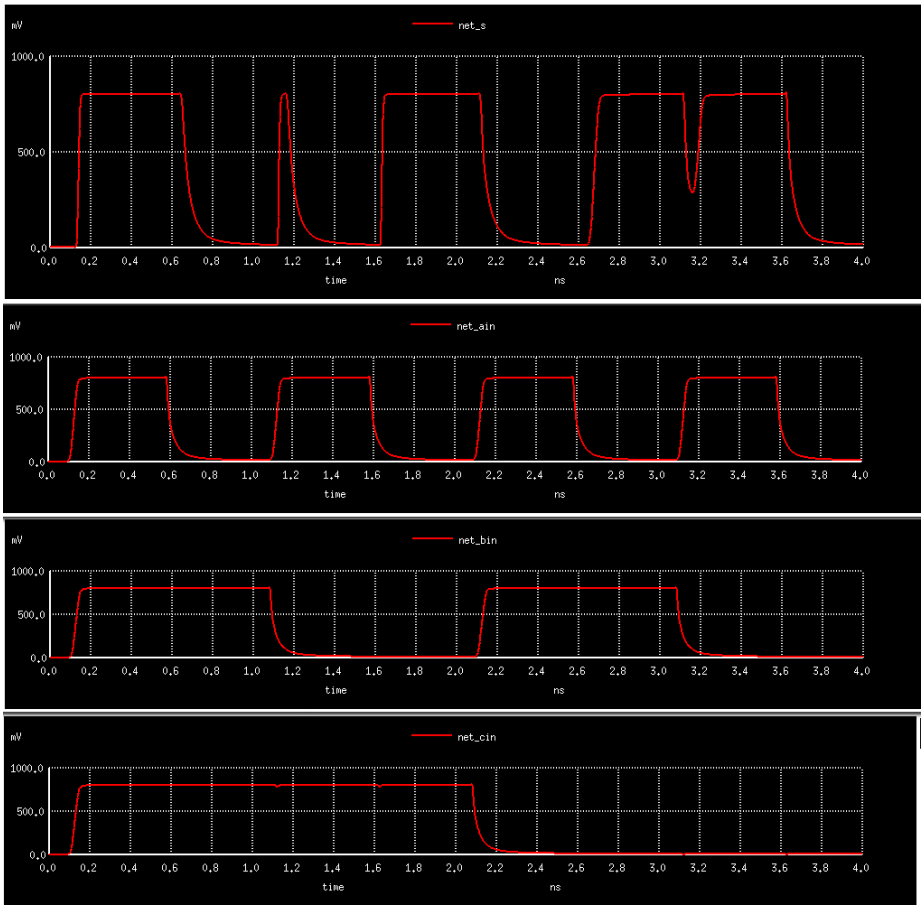


Figure 30: Waveform showing logical correctness of Sum

5.2.2 nSum

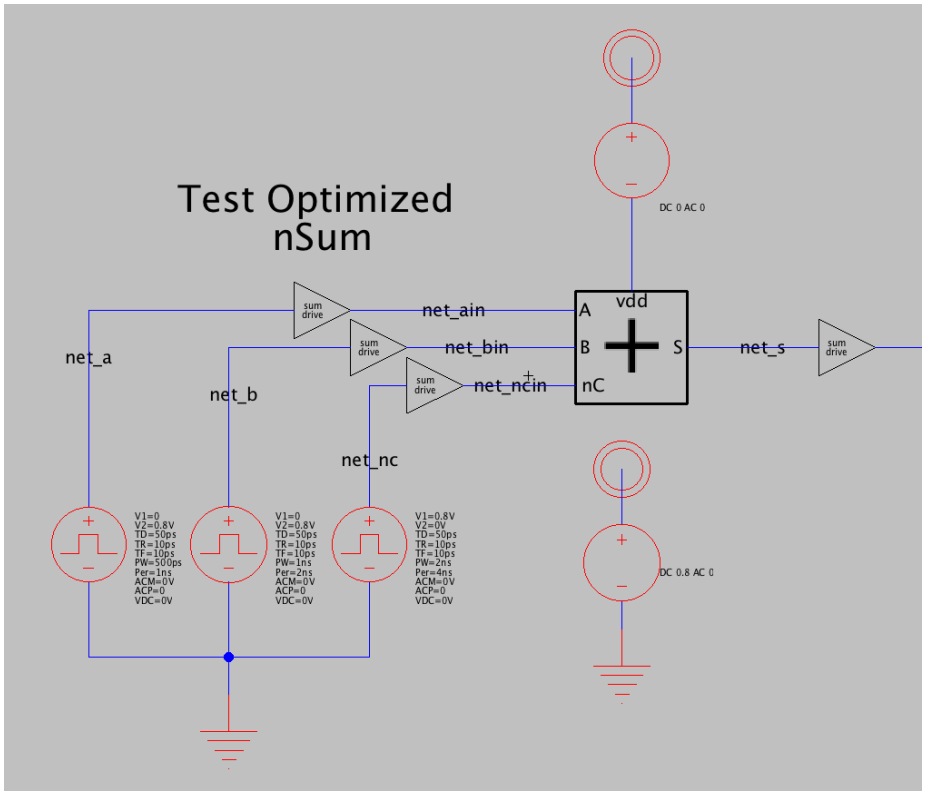


Figure 31: Schematic used to test logical correctness of nSum

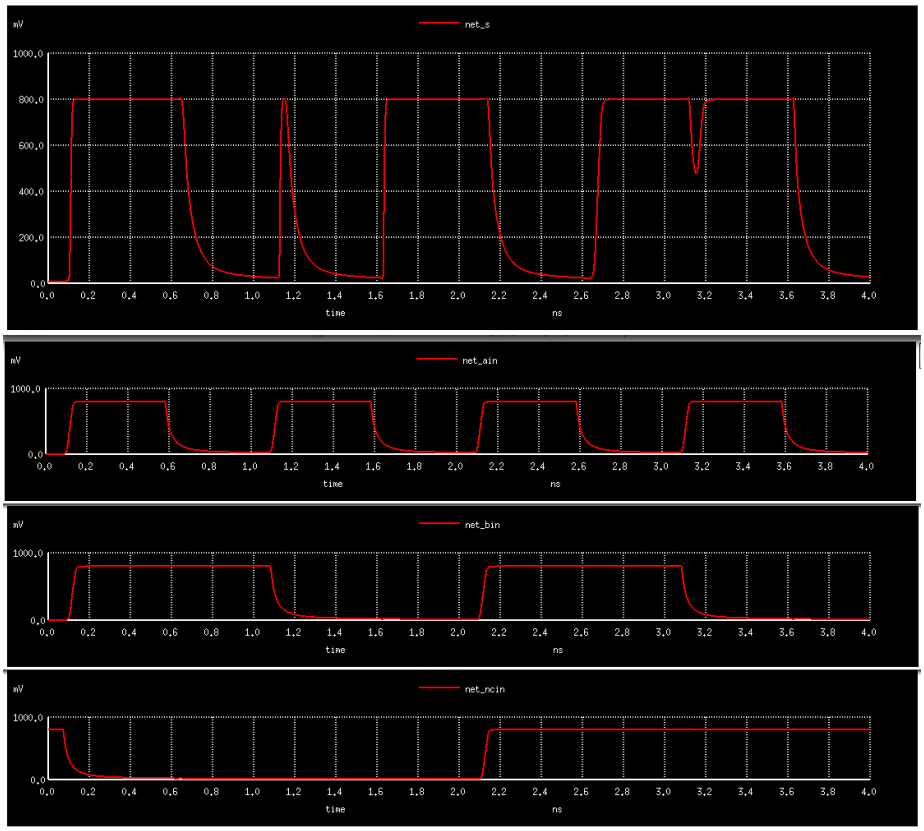


Figure 32: Waveform showing logical correctness of nSum

5.2.3 Carry

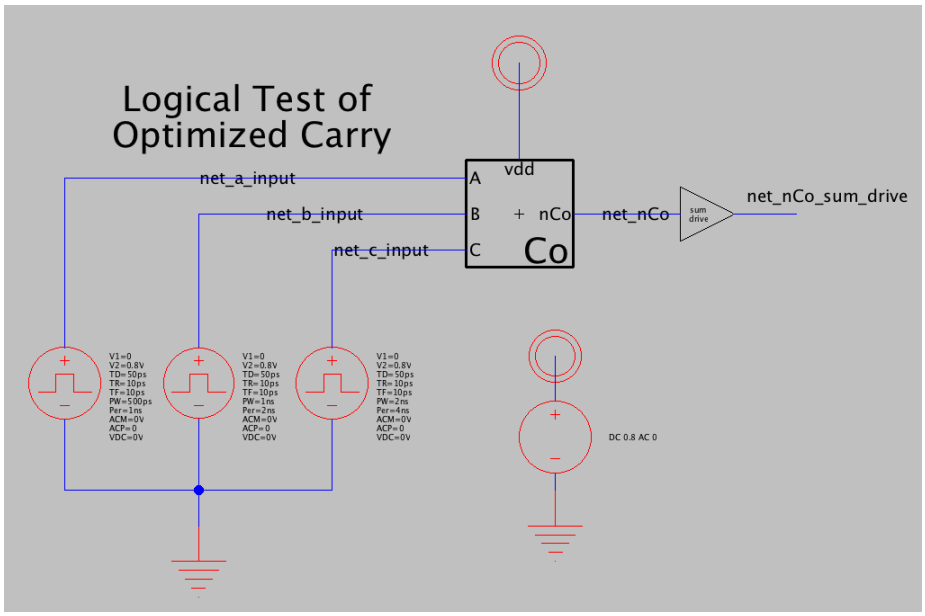


Figure 33: Schematic used to test logical correctness of Carry

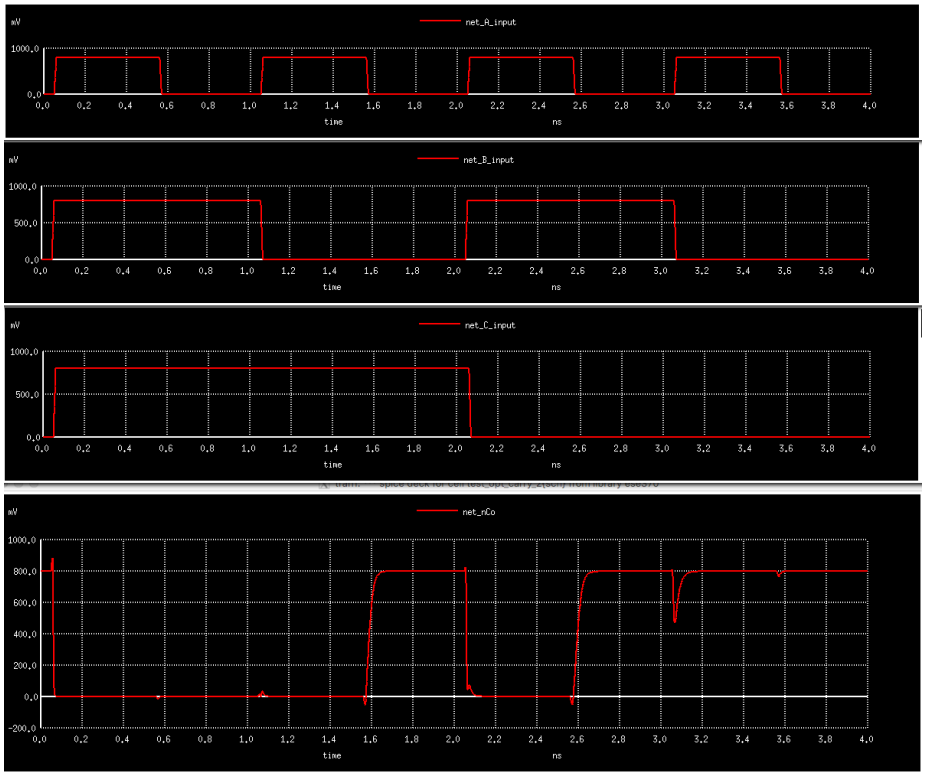


Figure 34: Waveform showing logical correctness of Carry

5.2.4 nCarry

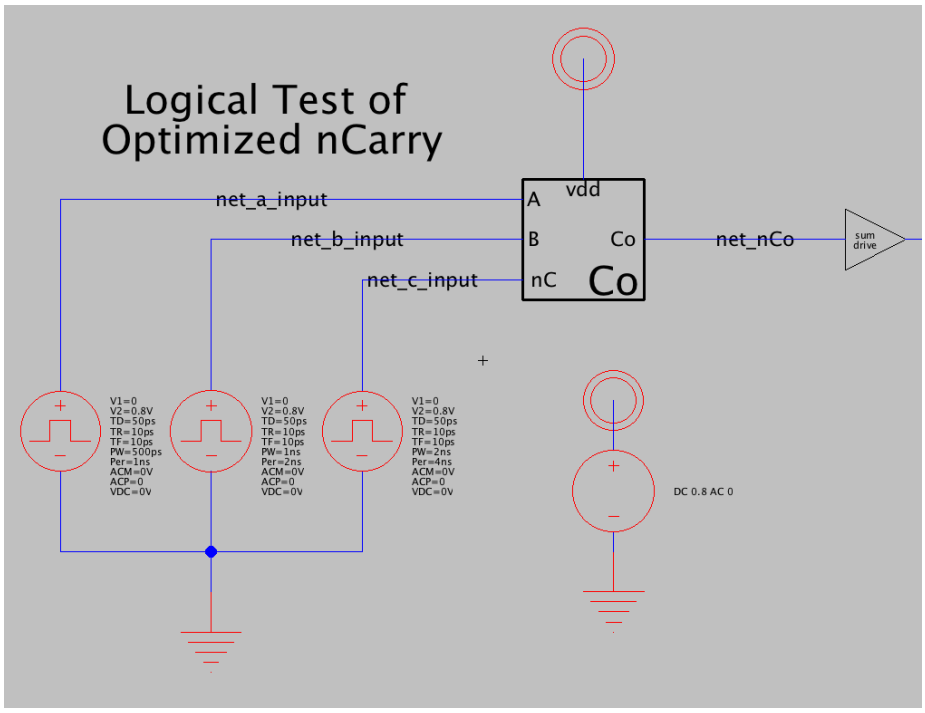


Figure 35: Schematic used to test logical correctness of nCarry

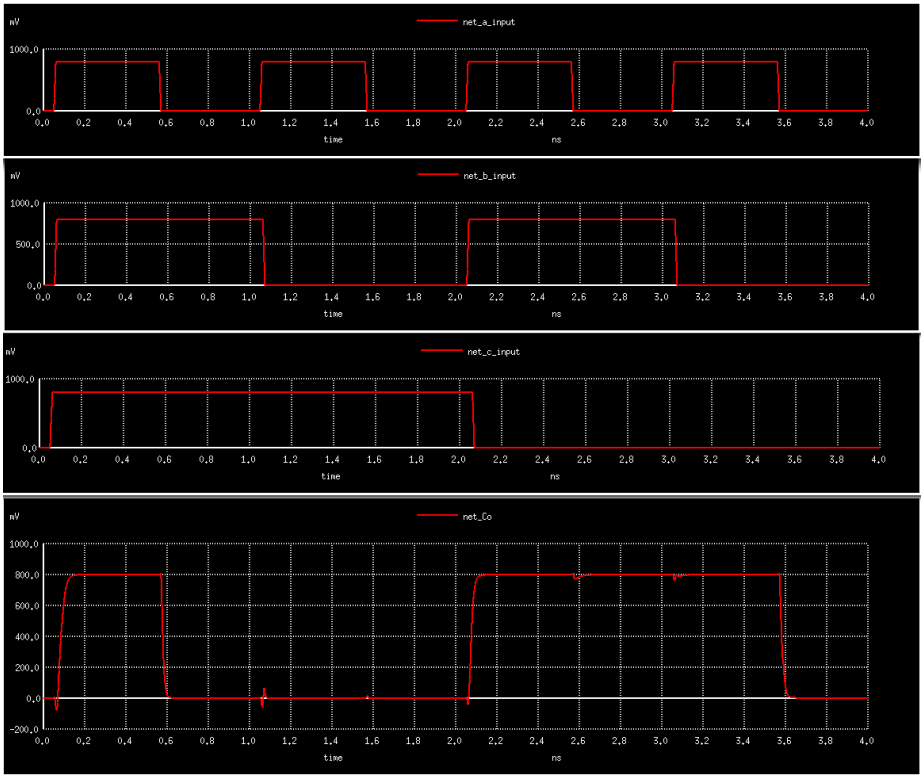


Figure 36: Waveform showing logical correctness of nCarry

5.2.5 1-bit Adder

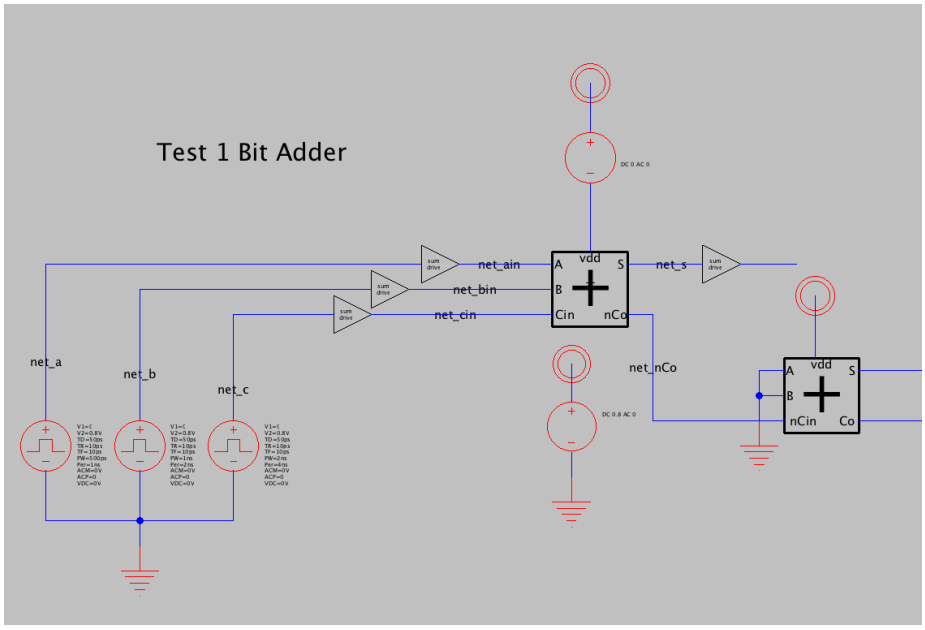


Figure 37: Schematic used to test logical correctness of the 1-bit Adder

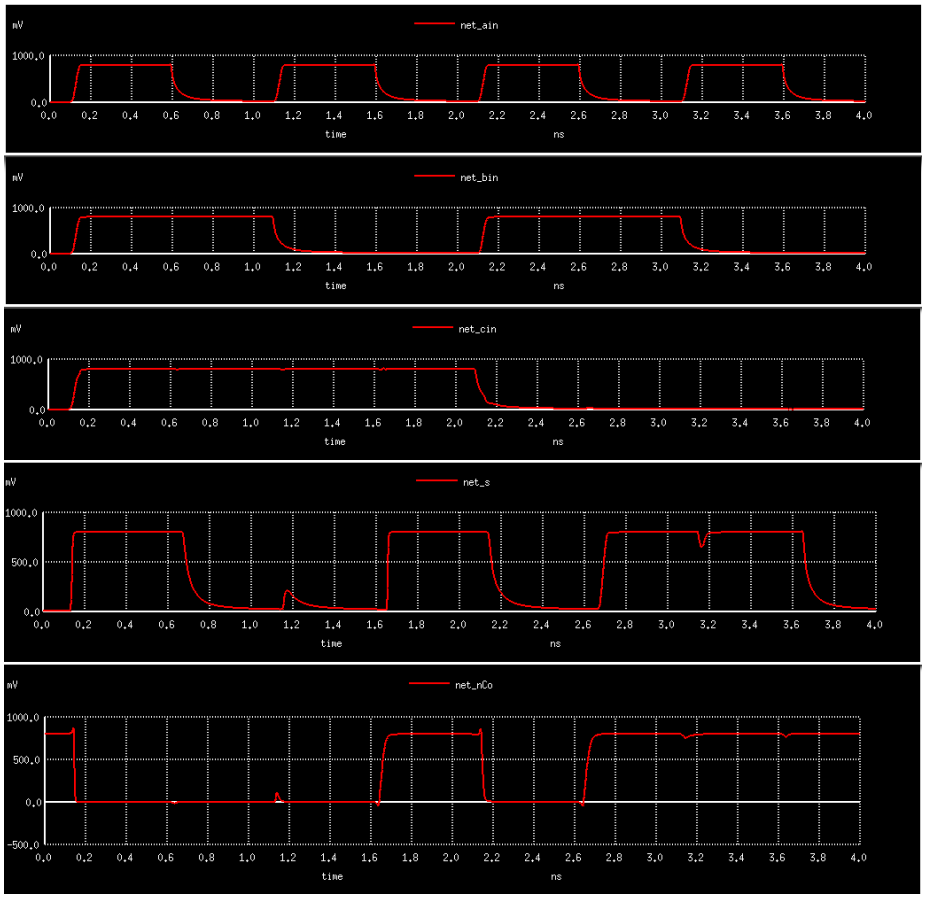


Figure 38: Waveform showing logical correctness of the 1-bit Adder

5.2.6 1-bit nAdder

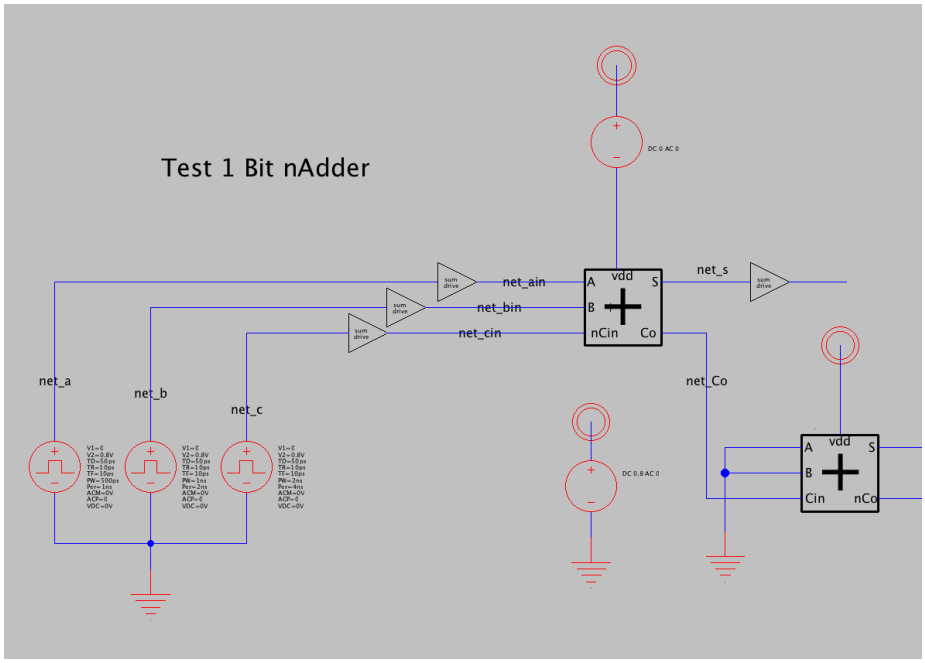


Figure 39: Schematic used to test logical correctness of the 1-bit nAdder

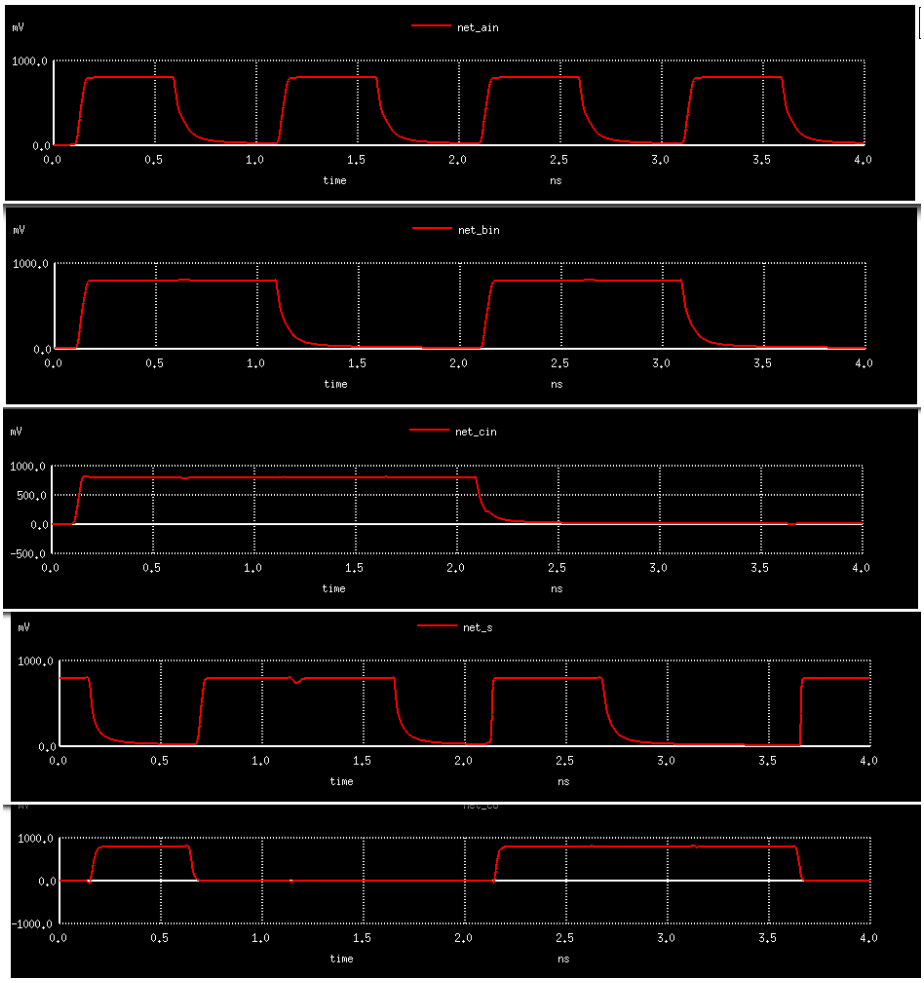


Figure 40: Waveform showing logical correctness of the 1-bit nAdder

5.2.7 16-bit Adder

To test the logical correctness of the 16-bit adder, I used the inputs $(A, B) = (0xffff, 0xffff) \rightarrow (0x0000, 0xffff) \rightarrow (0xffff, 0x0000) \rightarrow (0x0000, 0x0000)$. The important cases for correctness are the last three, which should have all sums as output V_{dd}, V_{dd}, and 0 respectively. Those can be clearly seen from the simulation below in Figure 41.

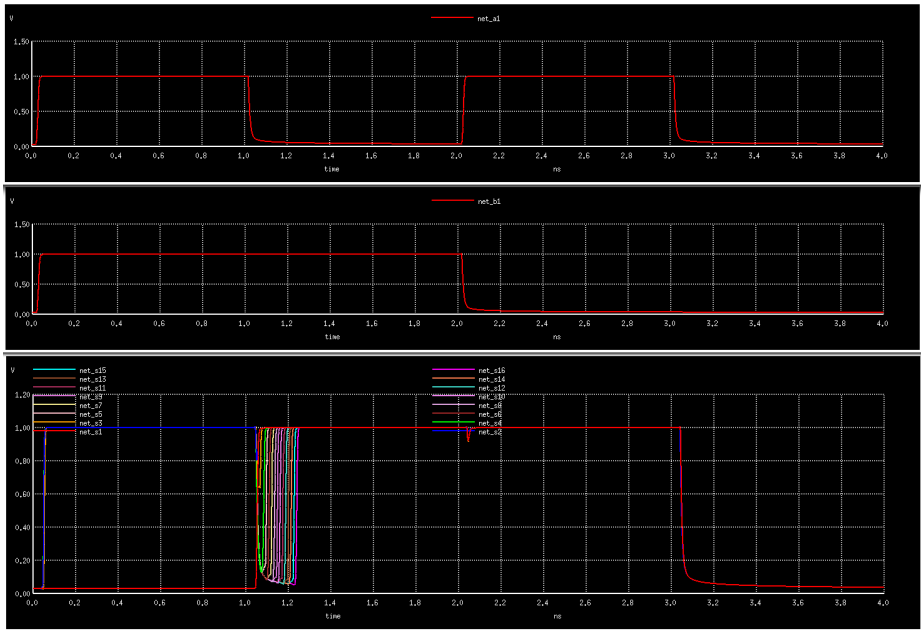


Figure 41: Waveform showing logical correctness of the 1-bit nAdder

5.3 Delay

5.3.1 τ Approximation

Sum

$$
\begin{aligned}
d_{Sum} &= d_{XOR} + d_{XNOR} + d_{inv} \\
d_{XOR} &= max(d_{XOR(A,B)=(0,1)}, d_{XOR(A,B)=(1,0)}) \\
d_{XNOR} &= max(d_{XNOR(A,B)=(1,1)}, d_{XNOR(A,B)=(0,0)})
\end{aligned}
$$

for $XOR_{(A,B)=(1,0)}$

$$
\begin{aligned}
d_{XOR(A,B)=(1,0)} &= R_{op}(C_{diff} + C_g) + R_{op}(C_{diff}) + (R_{op} + R_{on})(C_{diff} + 2C_g) \\
&\text{Assume } R_{op} = R_{on} = R_0 \\
&= R_0(C_{diff} + C_g) + R_0(C_{diff}) + (R_0 + R_0)(C_{diff} + 2C_g) \\
&\text{Assume } \frac{C_{diff}}{\gamma} = C_g = C_0 \\
&= R_0(\gamma C_0 + C_0) + R_0(\gamma C_0) + (R_0 + R_0)(\gamma C_0 + 2C_g) \\
&= R_0\gamma C_0 + R_0 C_0 + R_0\gamma C_0 + 2R_0\gamma C_0 + 2R_0 2C_g \\
&= (6 + 3\gamma)\tau
\end{aligned}
$$

for $XOR_{(A,B)=(0,1)}$

$$
\begin{aligned}
d_{XOR(A,B)=(0,1)} &= R_{op}(C_{diff} + 2C_g) + (R_{op} + R_{on})(C_{diff} + 2C_g) \\
&\text{Assume } R_{op} = R_{on} = R_0 \\
&= R_0(C_{diff} + 2C_g) + (2R_0)(C_{diff} + 2C_g) \\
&\text{Assume } \frac{C_{diff}}{\gamma} = C_g = C_0 \\
&= R_0(\gamma C_0 + 2C_0) + (2R_0)(\gamma C_0 + 2C_0) \\
&= R_0\gamma C_0 + R_0 2C_0 + 2R_0\gamma C_0 + 2R_0 2C_0 \\
&= \gamma\tau + 2\tau + 2\gamma\tau + 4\tau \\
&= 3\gamma\tau + 6\tau \\
&= (6 + 3\gamma)\tau \\
d_{XOR} &= max(d_{XOR(A,B)=(0,1)}, d_{XOR(A,B)=(1,0)}) \\
&= (6 + 3\gamma)\tau
\end{aligned}
$$

$$
\begin{aligned}
d_{XOR} &= max(d_{XOR(A,B)=(0,1)}, d_{XOR(A,B)=(1,0)}) \\
&= max((6 + 3\gamma)\tau, (6 + 3\gamma)\tau) \\
&= (6 + 3\gamma)\tau
\end{aligned}
$$

for $XNOR_{(A,B)=(1,1)}$

$$
\begin{aligned}
d_{XOR(A,B)=(1,1)} &= R_{op}(C_{diff}) + (R_{op} + R_{on})(C_{diff} + 2C_g) \\
&\text{Assume } R_{op} = R_{on} = R_0 \\
&= R_0(C_{diff}) + (2R_0)(C_{diff} + 2C_g) \\
&\text{Assume } \frac{C_{diff}}{\gamma} = C_g = C_0 \\
&= R_0(\gamma C_0) + (2R_0)(\gamma C_0 + 2C_0) \\
&= \gamma\tau + 2\gamma\tau + 2\tau
\end{aligned}
$$

nSum

$$\begin{aligned} d_{nSum} &= d_{XOR} + d_{XOR} + d_{inv} \\ &= (6 + 3\gamma)\tau + (6 + 3\gamma)\tau + 3\tau \\ &= (15 + 6\gamma)\tau \end{aligned}$$

Carry

$$\begin{aligned} d_{Carry} &= \frac{R_{op}}{2} + R_{op}\frac{R_{op}}{3})(4C_g) \\ &= \frac{22\tau}{3} \\ &\approx 7.33\tau \end{aligned}$$

nCarry

$$\begin{aligned} d_{nCarry} &= \frac{3R_{op}}{6} + R_{op}\frac{2R_{op}}{6})(4C_g) \\ &= \frac{22\tau}{3} \\ &\approx 7.33\tau \end{aligned}$$

1-bit Adder

$$\begin{aligned} d_{1bAdder} &= d_{buffer} + max(d_{Sum}, d_{Carry}) \\ &= 6\tau max((16 + 7\gamma)\tau, 7.33\tau) \\ &= (22 + 7\gamma)\tau \end{aligned}$$

1-bit nAdder

$$\begin{aligned} d_{1bnAdder} &= max(d_{nSum}, d_{nCarry}) \\ &= max((15 + 6\gamma)\tau, 7.33\tau) \end{aligned}$$

16-bit Adder

$$\begin{aligned} d_{16bAdder} &= 15d_{Carry} + d_{Sum} \\ &= 15(\frac{22\tau}{3}) + (16 + 7\gamma)\tau \\ &= (126 + 7\gamma)\tau \end{aligned}$$

5.3.2 Measuring τ

To measure τ, I used the FO4 topology (Figure 42) with a minimum-sized inverter. This topology should give delay of $R_0 \times 8C_0 = 8\tau$. After simulation (Figure 43), I found $8\tau = 4.21505 \times 10^{-12} = 526.881 fs$.

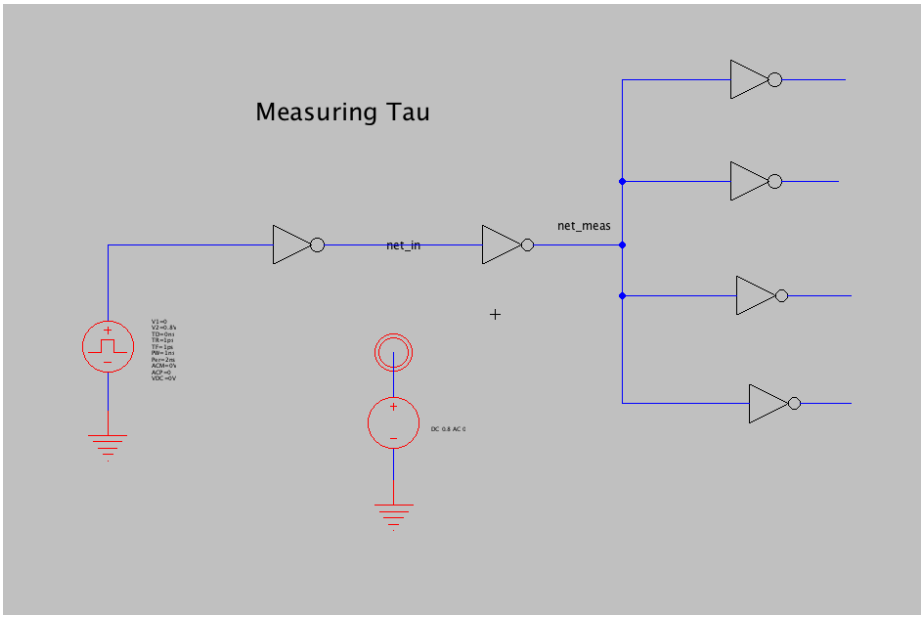


Figure 42: FO4 inverter topology for measuring τ

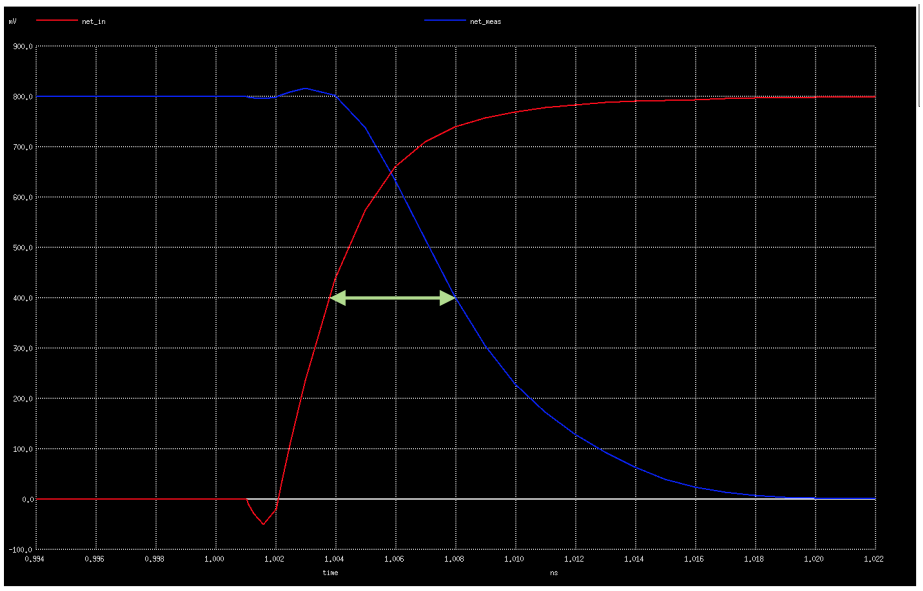


Figure 43: Simulation of FO4 inverter output

5.3.3 Measuring 16-bit Adder Delay

The delay from the 16-bit adder was measured with inputs $(A, B) = (0xffff, 0x0000) \rightarrow (0xffff, 0x0001)$. This is because the propagation delay for the carry through the entire 16-bit adder takes longest when the carry must be propagated through all 16 adders. The last node to charge on the output is the sum output of the 16th adder because the input capacitance of the sum gate. Each set of 1-bit adder inputs is driven with a buffer made from the 1-bit adder, and they are loaded with the same 1-bit adder buffer (Figure 44). The final two outputs are the 16th 1-bit adder sum and 16th 1-bit adder carry out. Those simulated waveforms are below in Figure 45 and Figure 45. The delay is $211.463ps$.

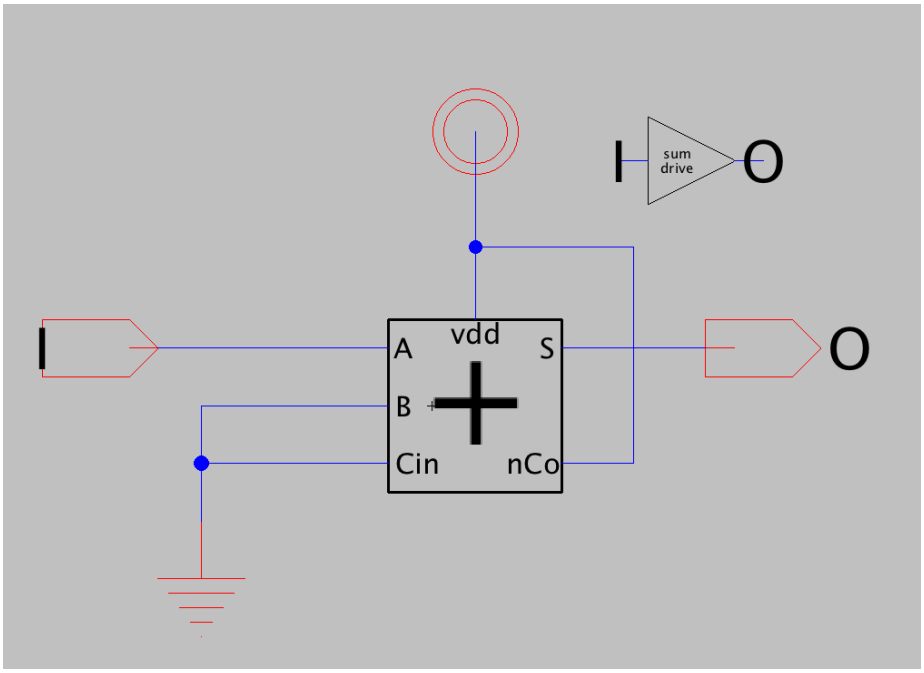


Figure 44: Buffer for Adder

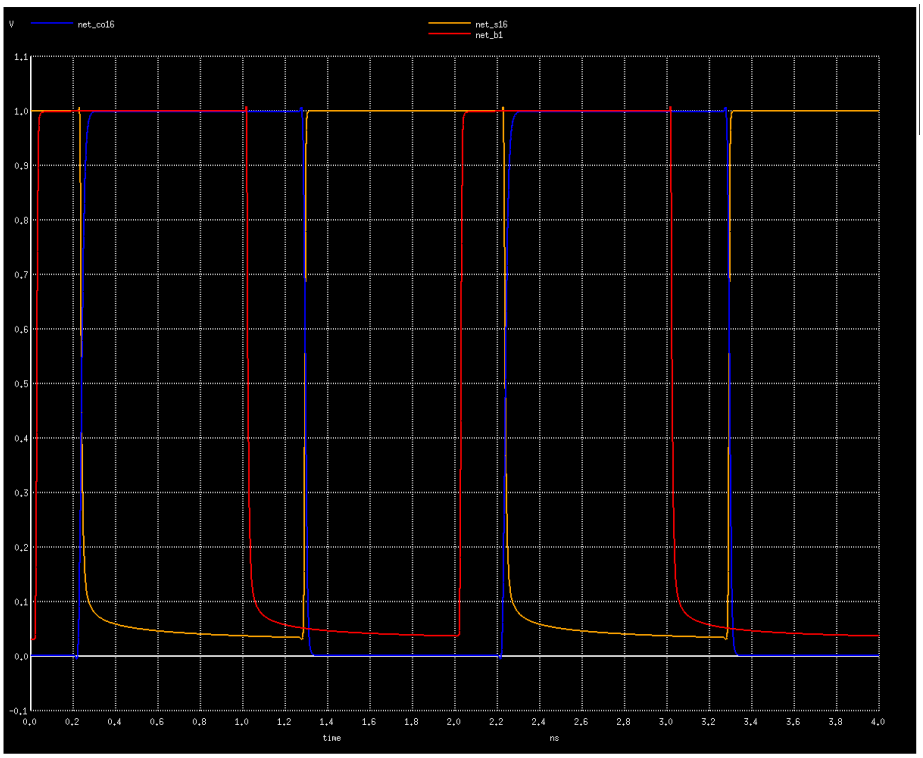


Figure 45: Simulation of 16-Bit adder

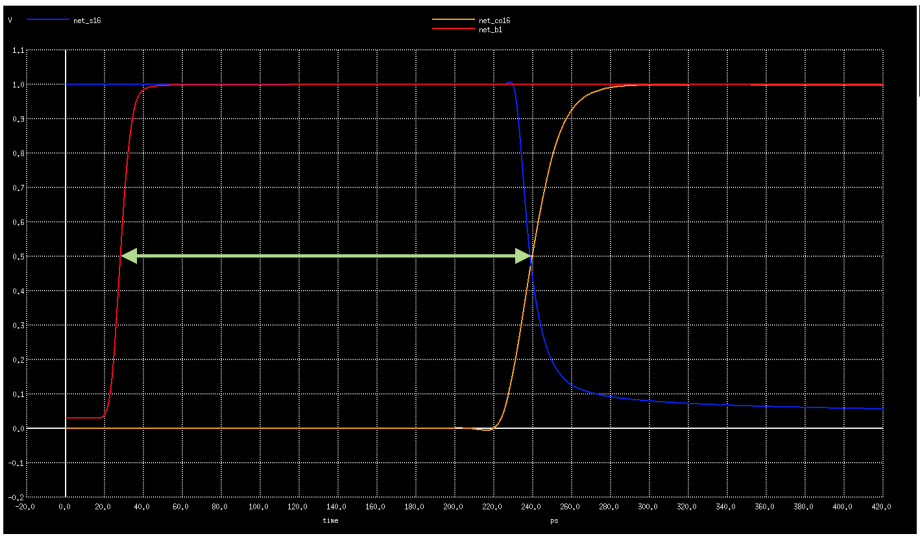


Figure 46: Simulation of 16-Bit adder showing delay

5.3.4 Comparing Calculations with Simulation

The calculated delay is $(126 + 7\gamma)\tau$. Let's make the assumption that $\gamma \approx 1.5$. therefore, we have a delay of $136.5\tau = 300.3 \times 0.5269 = 158.2ps$, which is less than the simulated delay of $211.463ps$, but is close considering I did not take into consideration the diffusion region capacitances for the carry modules. It is evident that models more complex than the τ model must be used to estimate the delay of this circuit.

5.4 Area

The area is calculated with a minimum-sized transistor as 1 unit and a transistor of width 2 and length 1 is 2 units:

$$\begin{aligned}
Area &= 8 \times A_{Adder} + 8 \times A_{nAdder} \\
A_{Adder} &= 12 + A_{Sum} + A_{Carry} \\
&= 12 + A_{XOR} + A_{XNOR} + 2 + 20 \\
&= 12 + 8 + 8 + 2 + 20 \\
&= 50 units \\
A_{nAdder} &= A_{nSum} + A_{nCarry} \\
&= A_{XOR} + A_{XOR} + 2 + 28 \\
&= 8 + 8 + 2 + 28 \\
&= 46 units
\end{aligned}$$

$$\begin{aligned}
Area &= 8 \times A_{Adder} + 8 \times A_{nAdder} \\
&= 8 \times 50 + 8 \times 46 \\
&= 768 units
\end{aligned}$$

5.5 Energy

5.5.1 Leakage Energy

To find maximum leakage energy input permutation, I used the same logic as for 1-bit Adder and nAdder since all inputs are accounted for. The graph of the current consumed by the Adder is in Figure 47

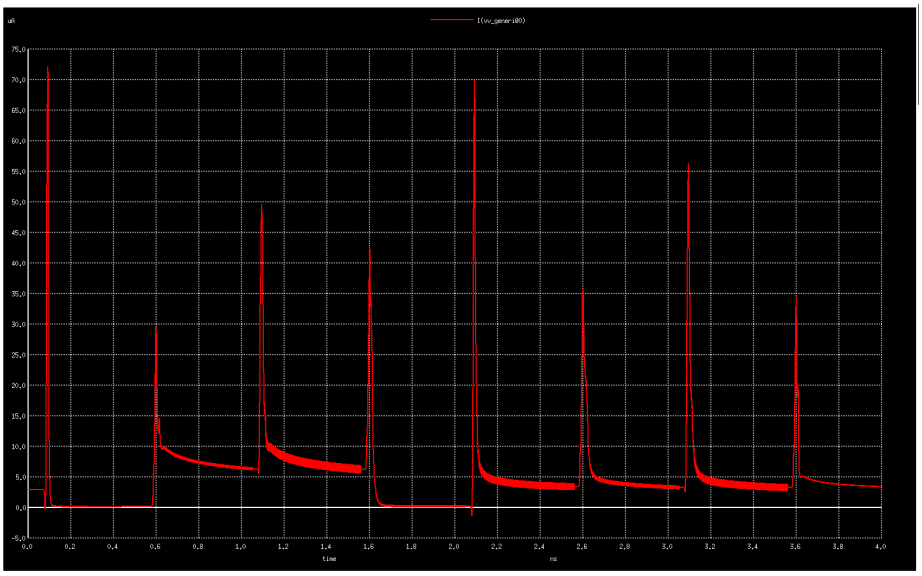


Figure 47: Simulation of the energy consumption of the 1-bit Adder

Maximum Case

The maximum case occurs between $t = 1.2 \rightarrow 1.4ns$, which is the input case $(A, B, C) = (1, 0, 1)$ and using
`meas tran yint integ I(vv_generi@0) from=1200ps to=1411ps`
we get $1.54217fW$. Each adder consumes this much energy, so we have $16 \times 1.54217fW = 24.67472fW$

Minimum Case

The maximum case occurs in the interval $t = 0.2 \rightarrow 0.4ns$, which is the input case $(A, B, C) = (1, 1, 1)$ and using
`meas tran yint integ I(vv_generi@0) from=200ps to=411ps`
we get $0.0388214fW$. Each adder also consumes this much minimum case leakage energy, so we have $16 \times 0.0388214fW = 0.6211424fW$.

5.5.2 Active Energy

Maximum Case

The maximum energy case is when all outputs must be charged by the minimum number of parallel transistors. This occurs when only 2 of the 3 inputs for the sum is on. That is $(A, B) = (0xffff, 0x0000 \rightarrow 0x0001)$ so the delay propagates. I used the following spice command:
`meas tran yint integ I(vv_adder) from=0ps to=4ns`
The result is $268.364fW$.

Average Case

To calculate this, I use the four cases $(A, B) = (0xffff, 0xffff) \rightarrow (0x0000, 0xffff) \rightarrow (0xffff, 0x0000) \rightarrow (0x0000, 0x0000) \rightarrow (0xffff, 0xffff)$ and divided the total energy consumed by 4, since we have 4 switching input cases. I used the following command:
`meas tran yint integ I(vv_adder) from=0ps to=5ns`
The result is $\frac{236.575fW}{4} = 59.14375fW$.

6 Alternate Designs and Considerations

As stated above, the following is the order in which I tried optimizations.

1. Replace 3-input XOR with pass transistor logic XOR cascade ($667.123ps \rightarrow 528.022ps$)
2. Change $V_{dd} = 0.8V \rightarrow V_{dd} = 1V$ ($528.022ps \rightarrow 309.259ps$)
3. Re-order inputs to Carry and nCarry to reduce the C_{diff} for the later input ($309.259ps \rightarrow 295.593ps$)
4. Increase width of transistors with A and B inputs for Carry and nCarry ($295.593ps \rightarrow 274.925ps$)
5. Add step-up stages for A and B inputs for Carry and nCarry ($274.925ps \rightarrow 280.286ps$)*
6. Add step-up stages for A and B inputs for 1-bit Adder ($274.925ps \rightarrow 271.282ps$)
7. Add step-up stages for A and B inputs for 1-bit nAdder ($271.282ps \rightarrow 271.864ps$)*

The two alternate designs denoted with the * failed to improve timing. In both cases, the added delay created by the buffers outweighed the reduction in delay from reducing the input capacitance to the gate.
As part of the pass-transistor logic, I first attempted to create a non-cascaded version, making a single pass-transistor XOR3 gate. This, however created a large Ron, since there were 2 pass transistors in series, and it also had a non-minimal input capacitance.
The increase in V_{dd} gave a sizable decrease in delay, much more than expected. The gain from input reordering was also more sizable than I originally expected.

7 Continuing Optimizations

List of proposed further optimizations (other than those above):

1. explore cascaded CMOS topology for sum instead of pass transistor logic

2. ratioed logic
3. cascade gates to calculate carry out
4. manipulate transistor widths to equalize wost-case Ron in all stages
5. optimize number of stages for sum
6. equalize input capacitance on all inputs

8 Code of Academic Integrity

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