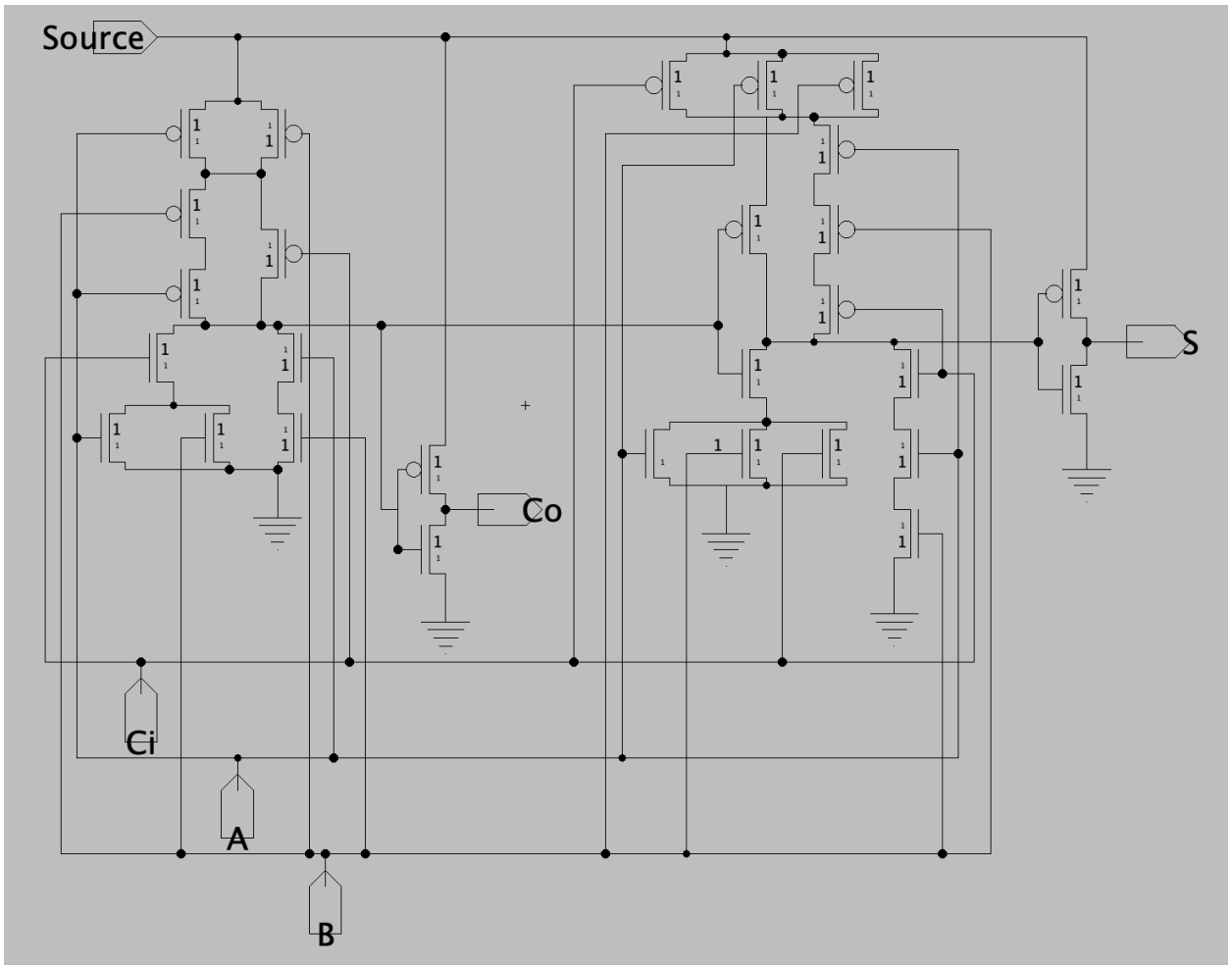
Ripple Carry Adder - Baseline Design

Circuit

Schematic



Note: Source is V_{dd}, C_{in} and C_{out} are carry input and output, respectively. S is adder output.

Design from: Jan M. Rabaey, Anantha Chandrakasan, and Borivoje Nikolic. *Digital Integrated Circuits, A Design Perspective*, 2nd edition, Prentice Hall, 2003.

Description of Logic and Operation

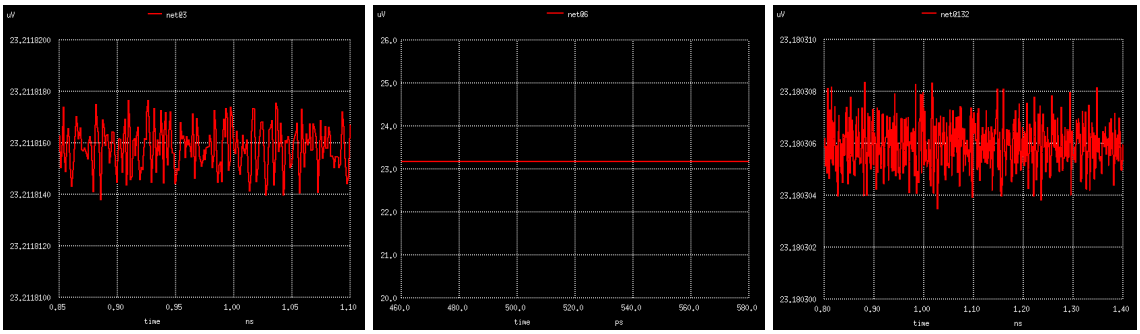
This is a static CMOS implementation of 1-bit adder slice. The function the adder slice implements, given inputs A, B, and C_i is: for S, $A \oplus B \oplus C_i$ and for C_o, $AB+BC_i+AC_i$. The design simplifies the boolean algebra to create a relatively simple CMOS design.

Validation of Baseline Design

Note: some of the inputs are noisy, but they are all at the correct voltage levels.

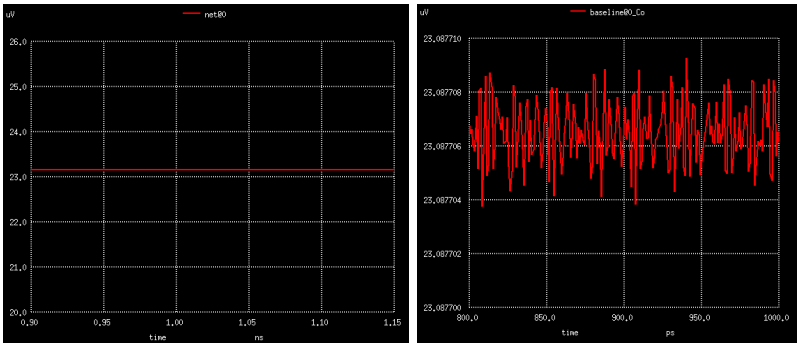
Case 1: A=0V, B=0V, C_{in}=0V

Inputs:



Note: net@132 is C_{in}, net@3 is A, and net@6 is B.

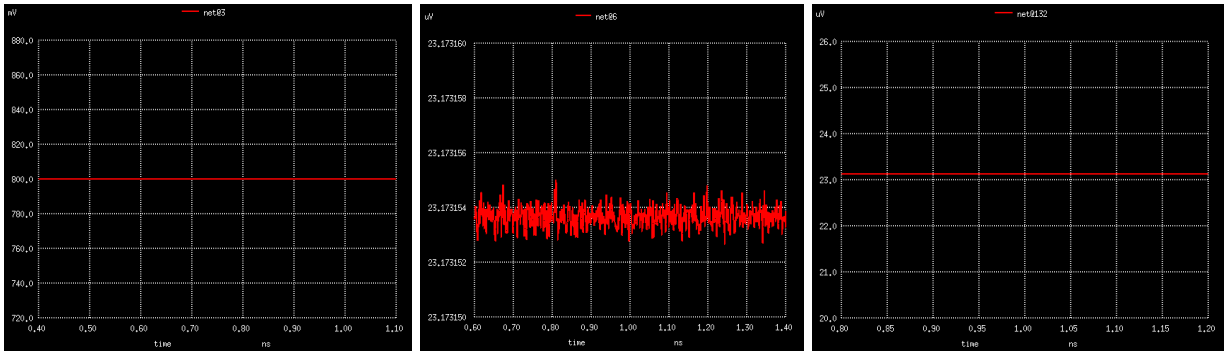
Outputs:



net@0 is S and baseline@0_Co is C_O

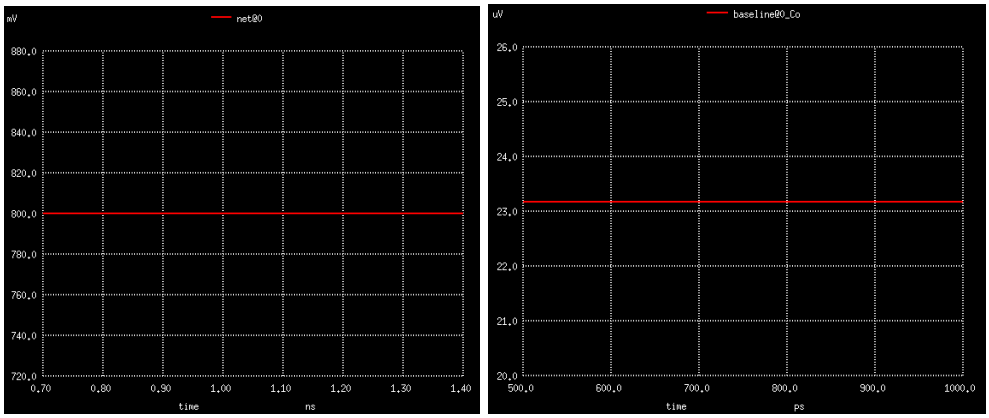
Case 2: A=0.8V, B=0V, C_{in}=0V

Inputs:



Note: net@132 is C_{in}, net@3 is A, and net@6 is B

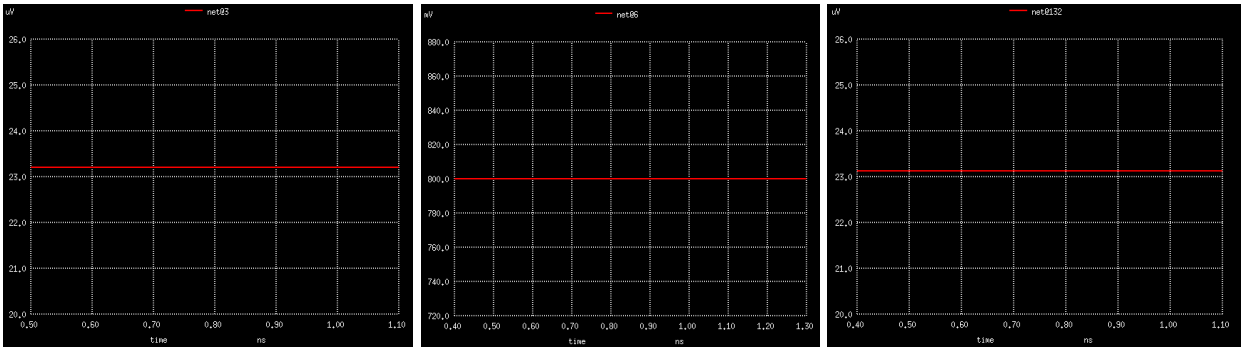
Outputs:



net@0 is S and baseline@0_Co is C_O

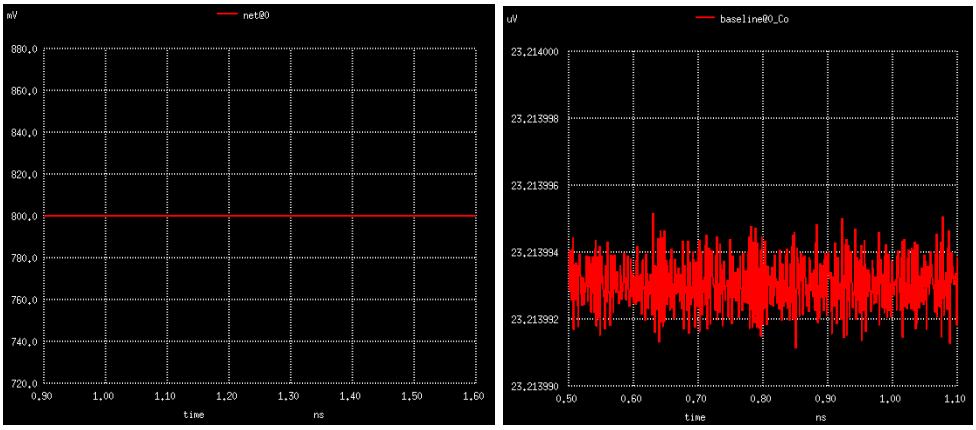
Case 3: A=0V, B=0.8V, C_{in}=0V

Inputs:



Note: net@132 is C_{in}, net@3 is A, and net@6 is B

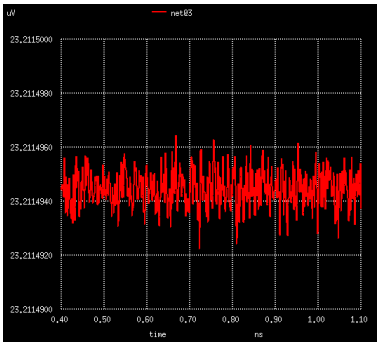
Outputs:

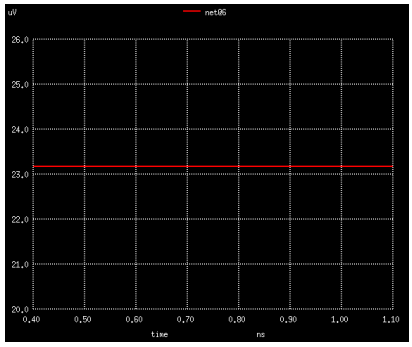


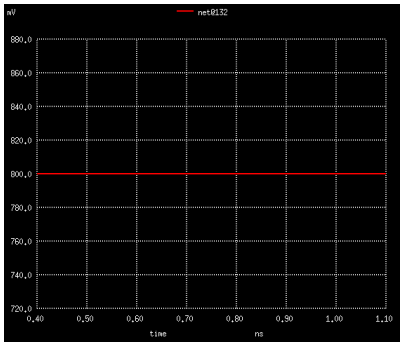
net@0 is S and baseline@0_Co is C_O

Case 4: A=0V, B=0V, C_{in}=0.8V

Inputs:

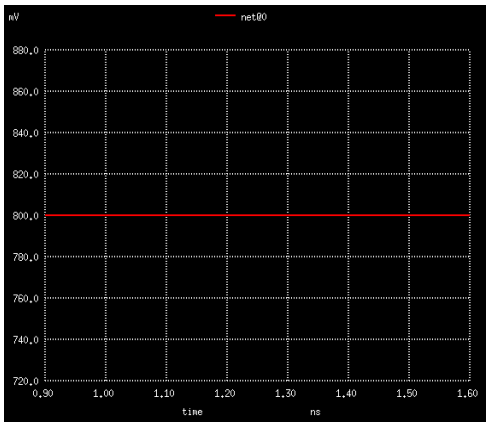


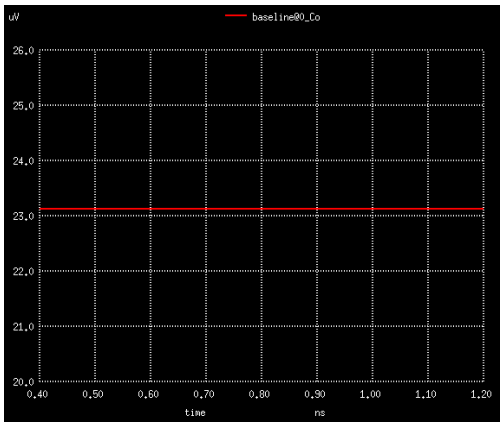




Note: net@132 is C_{in}, net@3 is A, and net@6 is B

Outputs:

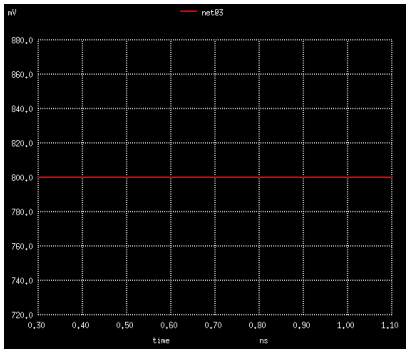


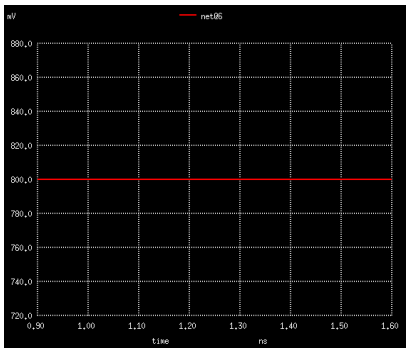


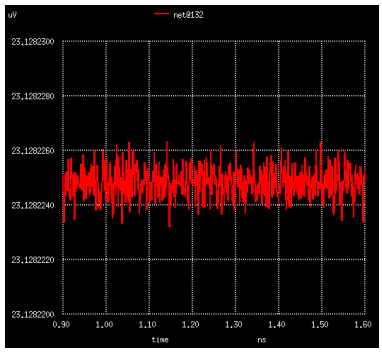
net@0 is S and baseline@0_Co is C_O

Case 5: A=0.8V, B=0.8V, C_{in}=0V

Inputs:

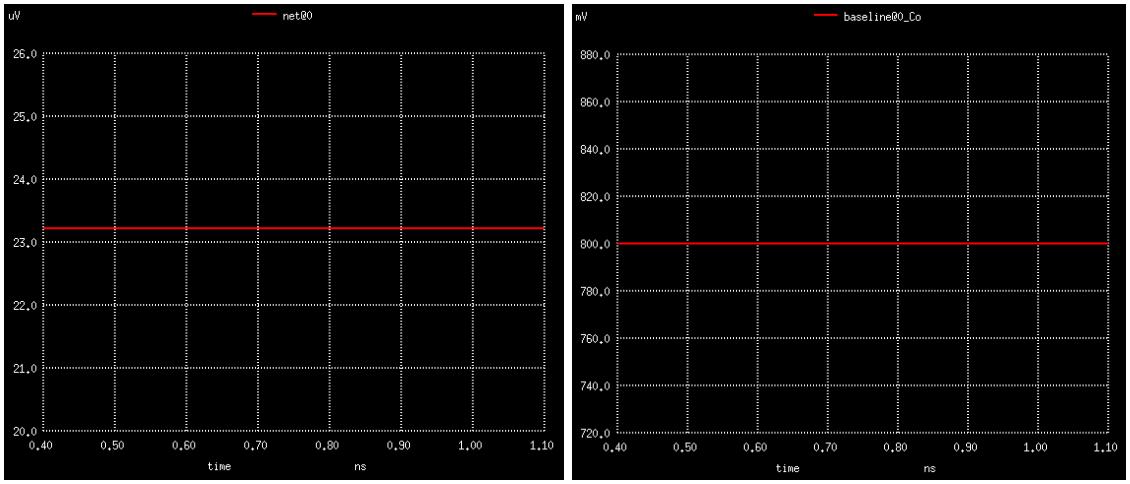






Note: net@132 is C_{in}, net@3 is A, and net@6 is B

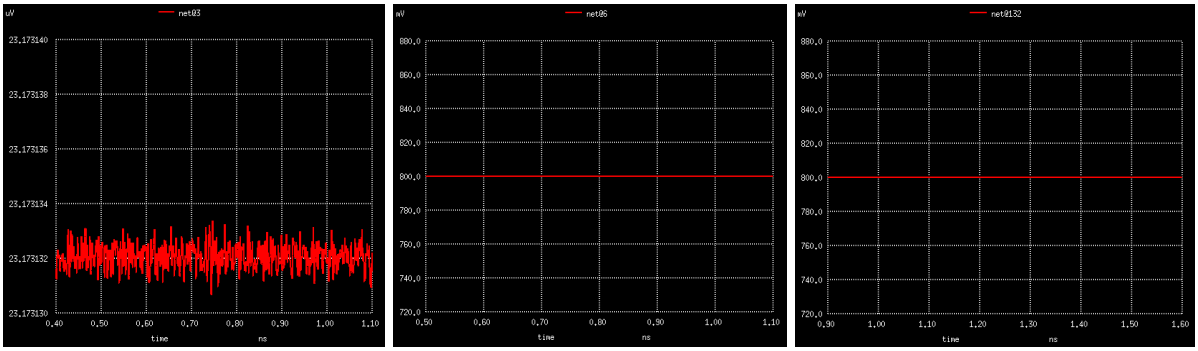
Outputs:



net@0 is S and baseline@0_Co is C_O

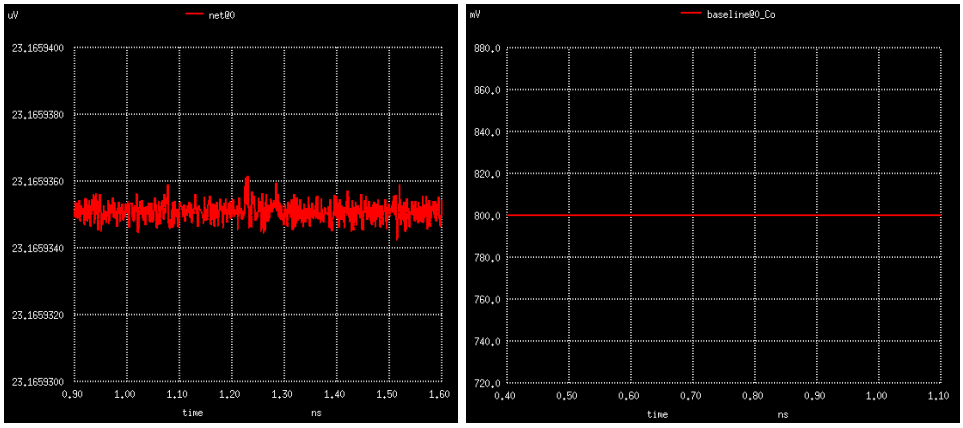
Case 6: A=0V, B=0.8V, C_{in}=0.8V

Inputs:



Note: net@132 is C_{in}, net@3 is A, and net@6 is B

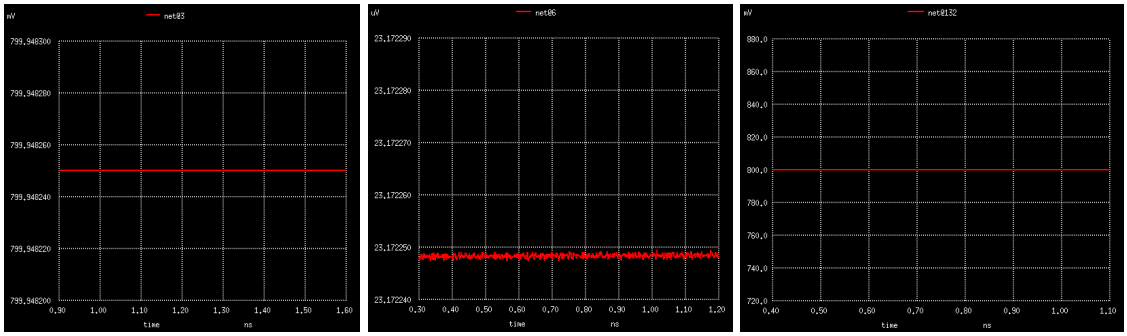
Outputs:



net@0 is S and baseline@0_Co is C_O

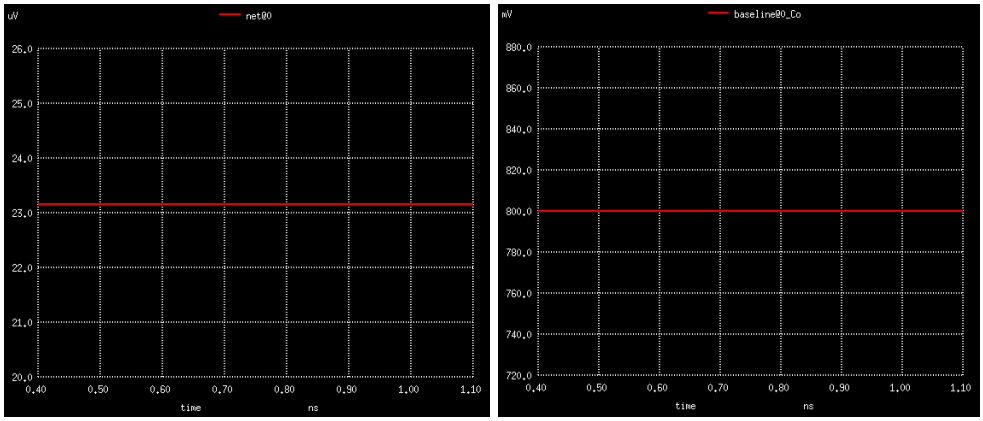
Case 7: A=0.8V, B=0V, C_{in}=0.8V

Inputs:



Note: net@132 is C_{in}, net@3 is A, and net@6 is B

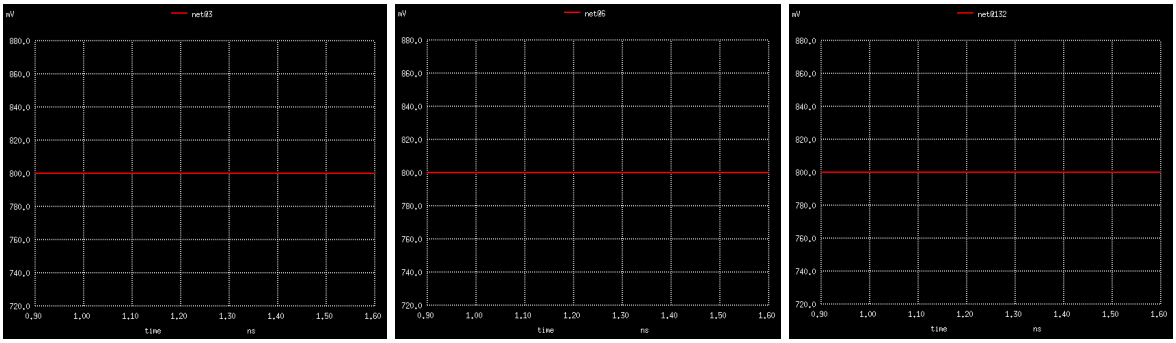
Outputs:



net@0 is S and baseline@0_Co is C_O

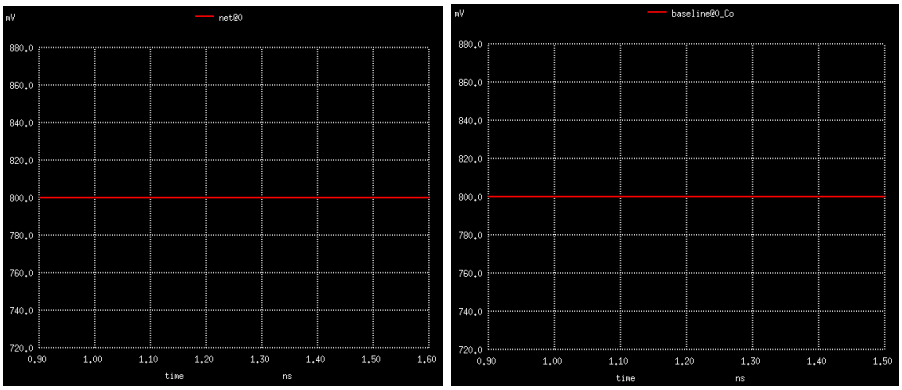
Case 8: A=0.8V, B=0.8V, C_{in}=0.8V

Inputs:



Note: net@132 is C_{in}, net@3 is A, and net@6 is B

Outputs:



net@0 is S and baseline@0_Co is C_O

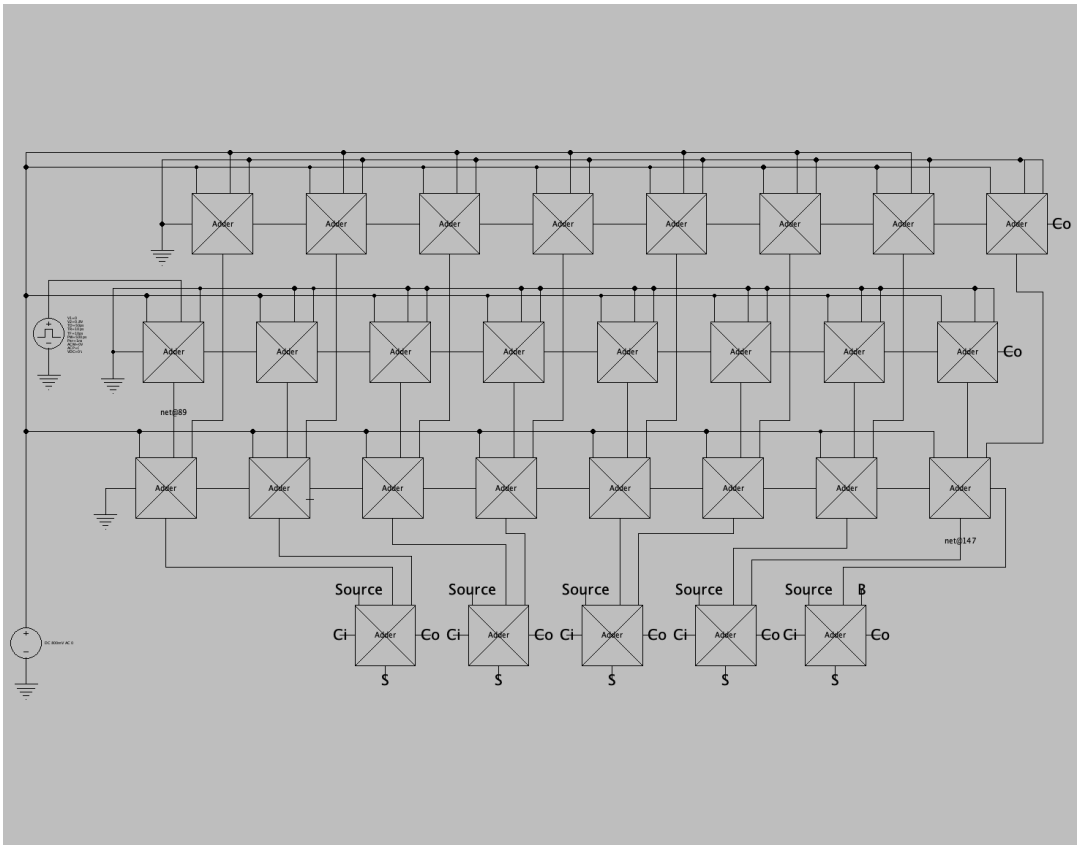
Delay

Switching Case
Input 1 (A): 00000000 → 00000001 → 00000000
Input 2 (B): 011111111

Justification
The switching case has the longest delay between the switching input (net@89 in the schematic below) and the output of the last adder slice in the 8-bit adder (net@147) because the carry out has to propagate through 7 1-bit adder slices before being outputted by the final 8-bit adder slice.

Calculations
Worst case fall time: $(R_0 * 8C_0) + 7((2R_0*4C_0) + (R_0 * 6C_0)) + ((5/2)R_0*4C_0) + (2R_0*2C_0) + (R_0*8C_0)$
= 128τ

Spice Test Schematic

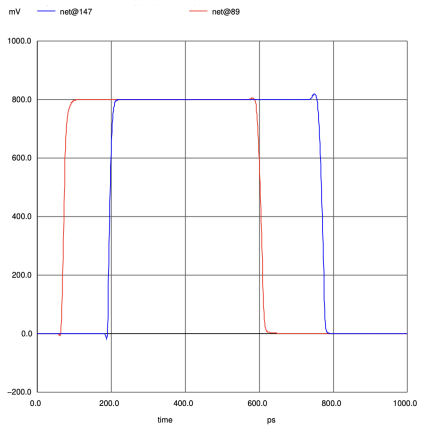


Note: net@89 and net@147 are labeled on the schematic

Simulation Results
Worst case fall time: **165.4213 ps**

Commands:
tran 1ps 2000ps
plot net@89 net@147
meas tran delay trig net@89 val=0.4 fall=1 targ net@147 val=0.4 fall=1
Vpulse Parameters:
V1=0, V2=0.8V, TD=50ps, TR=10ps, TF=10ps, PW=500ps, Per=1ns

Plot (rise time and fall time)



Note: net@147 is output of 8th adder and net@89 is input of 1st adder (see schematic)

Switching Energy

Switching Cases

Maximum Energy:

Input 1 (A): 00000000 → 111111111 → 00000000

Input 2 (B): 00000000 → 111111111 → 00000000

Average Energy

Input 1 (A): 00000000 → 00001111 → 00000000

Input 2 (B): 00000000 → 00001111 → 00000000

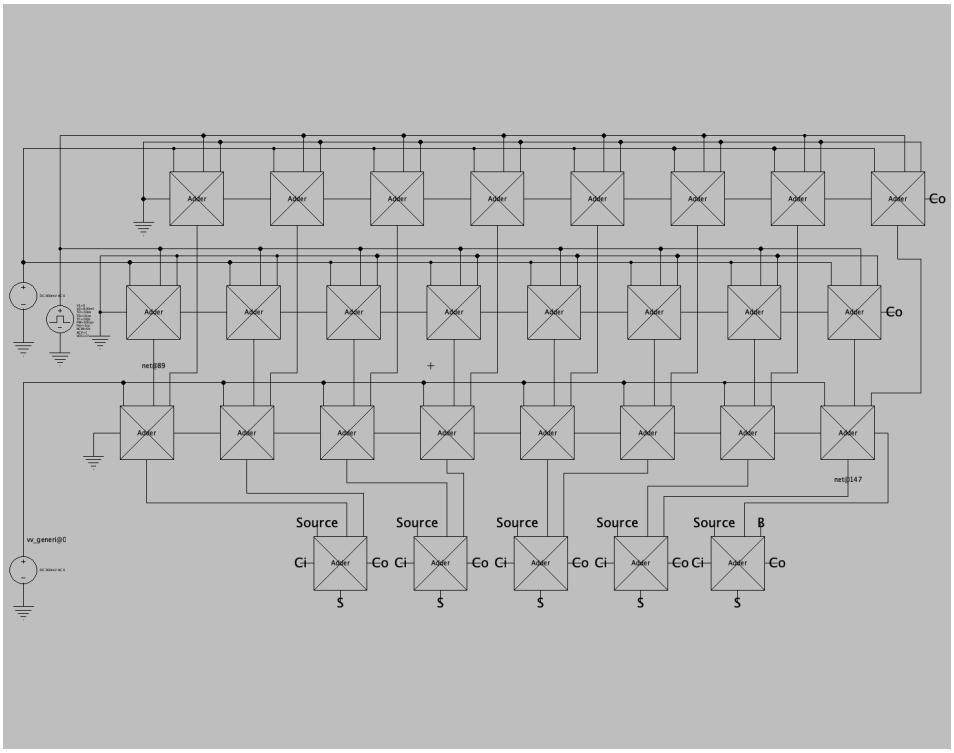
Justification

In the maximum energy switching case, all 8 of the 1-bit adder slices will have their outputs switched from 0→1 and 1→0.

For the average energy switching case, 4 (or half) of the 1-bit adder slices will have their outputs switched from 0→1 and 1→0.

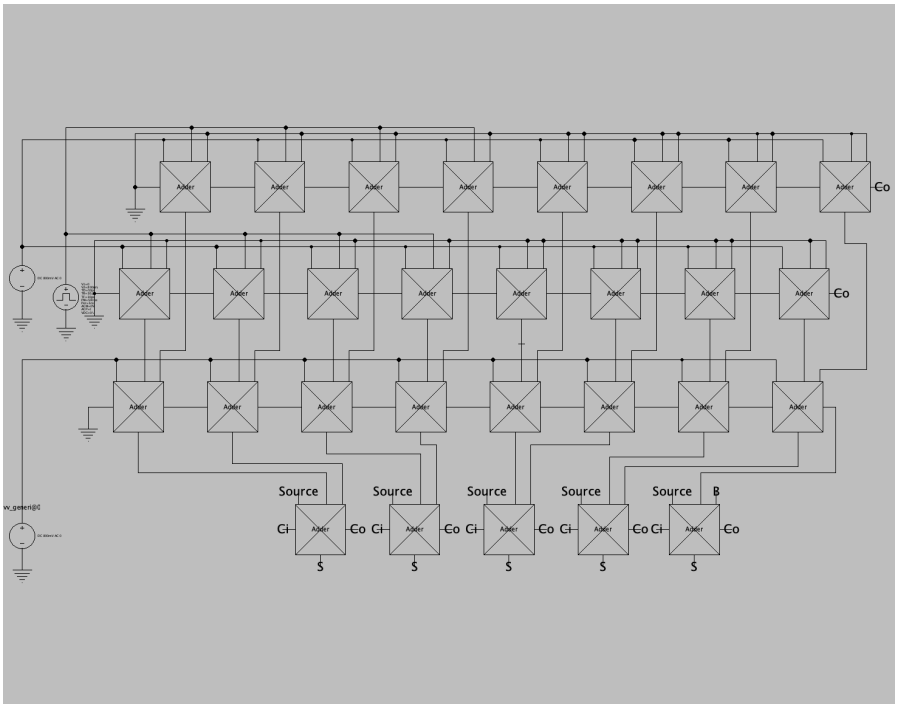
Spice Test Schematics

Max Energy Test Schematic



Note: vv_generi@0 is labeled on the schematic

Average Energy Test Schematic



Note: vv_generi@0 is labeled on the schematic

Simulation Results

Maximum Energy Consumption: **2.58549413 fJ**
Average Energy Consumption: **1.3788944 fJ**

NOTE: The SPICE current switching waveform had both positive area and negative area under the curve (see Plots section). To calculate the switching energy consumption, I integrated the positive and negative areas separately and summed their magnitudes.

Commands

Max Energy Commands:

tran 1ps 2000ps
plot I(vv_generi@0)
meas tran yint integ I(vv_generi@0) from=53.1195ps to=60.15ps (got -6.75013e-18)
meas tran yint integ I(vv_generi@0) from=60.15ps to=72.212ps (got 3.67094e-16)
meas tran yint integ I(vv_generi@0) from=72.212ps to=190ps (got -2.21165e-15)

Average Energy Commands:

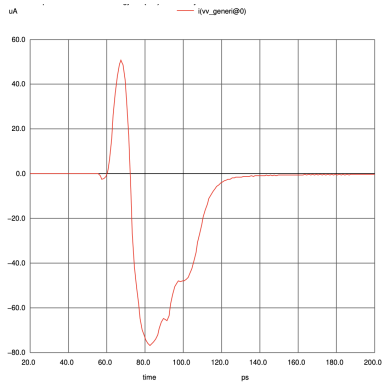
tran 1ps 2000ps
plot I(vv_generi@0)
meas tran y integ I(vv_generi@0) from=50.1ps to=60.168ps (got -3.55140e-18)
meas tran yint integ I(vv_generi@0) from=60.168ps to=72.215ps (got 1.83433e-16)
meas tran yint integ I(vv_generi@0) from=72.215ps to=180ps (got -1.19191e-15)

Vpulse Parameters:

V1=0, V2=0.8V, TD=50ps, TR=10ps, TF=10ps, PW=500ps, Per=1ns

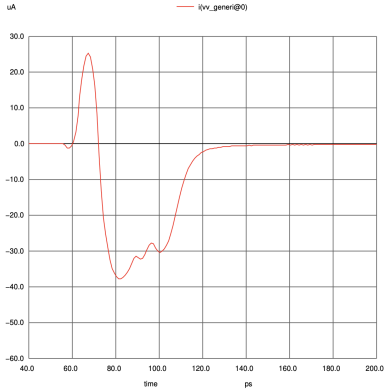
Plots

Max Energy Consumption. Current Entering Voltage Source.



Note: i(vv_generi@0) is current leaving the 8-bit adder under test. Both positive and negative areas under the curve were taken into account.

Average Energy Consumption. Current Entering Voltage Source.



Note: i(vv_generi@0) is current leaving the 8-bit adder under test. Both positive and negative areas under the curve were taken into account.

Leakage Energy

Cases (FOR 1-BIT SLICE)

There are 8 possible inputs into a 1-bit adder, so there are 8 cases:

Case 1
Input 1 (A): 0
Input 2 (B): 0
Input 3 (C_{in}): 0

Case 2
Input 1 (A): 1
Input 2 (B): 0
Input 3 (C_{in}): 0

Case 3
Input 1 (A): 0
Input 2 (B): 1
Input 3 (C_{in}): 0

Case 4
Input 1 (A): 0
Input 2 (B): 0
Input 3 (C_{in}): 1

Case 5
Input 1 (A): 1
Input 2 (B): 1
Input 3 (C_{in}): 0

Case 6
Input 1 (A): 0
Input 2 (B): 1
Input 3 (C_{in}): 1

Case 7
Input 1 (A): 1
Input 2 (B): 0
Input 3 (C_{in}): 1

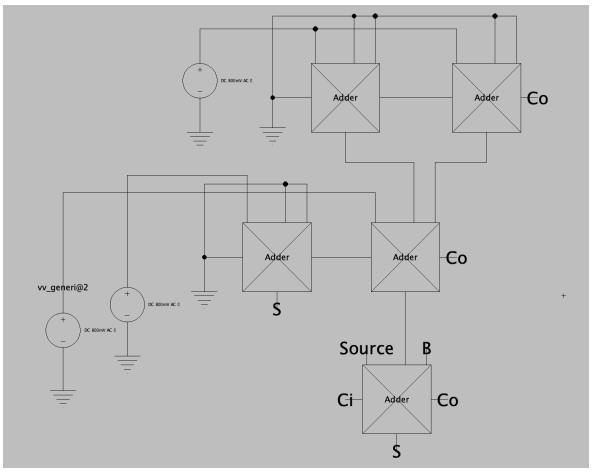
Case 8
Input 1 (A): 1
Input 2 (B): 1
Input 3 (C_{in}): 1

Justification

To calculate the maximum leakage energy, there are 8 possible static inputs into a 1-bit adder slice (2^3), as provided above. We can measure the leakage energy for each of these input cases and from this, the maximum and minimum leakage energies can be found.

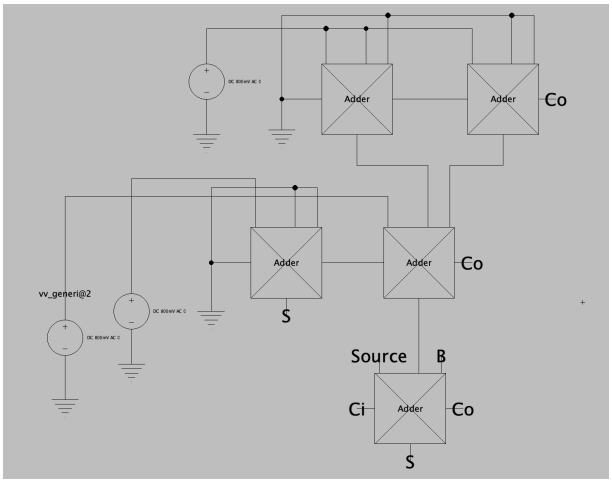
Spice Test Schematics (FOR 1-BIT SLICE)

Case 1



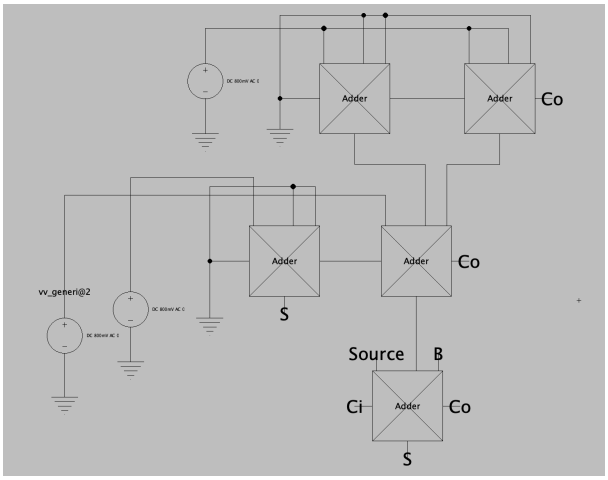
Note that the adder in the middle row is under test. Also, vv_generi@2 is labeled.

Case 2



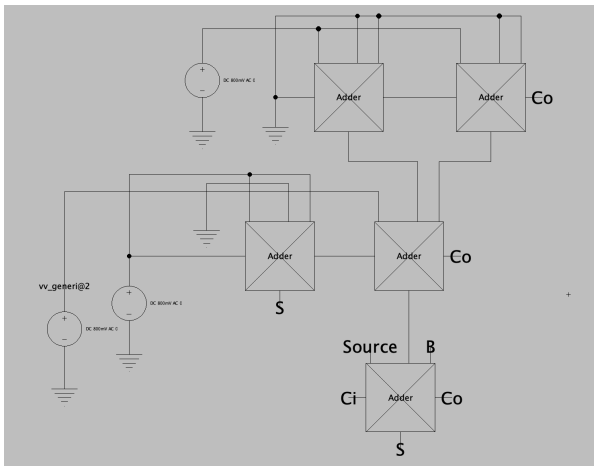
Note that the adder in the middle row is under test. Also, vv_generi@2 is labeled.

Case 3



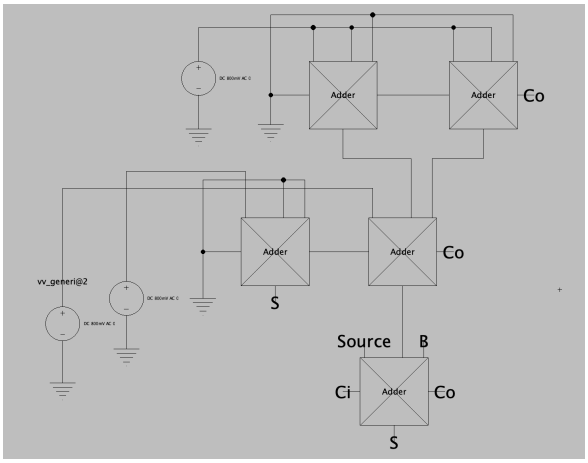
Note that the adder in the middle row is under test. Also, vv_generi@2 is labeled.

Case 4



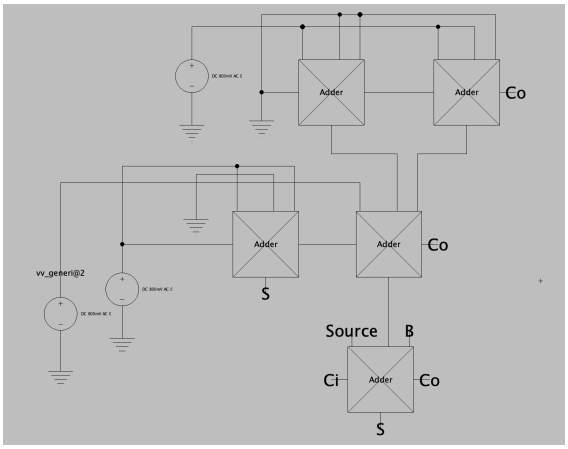
Note that the adder in the middle row is under test. Also, vv_generi@2 is labeled.

Case 5



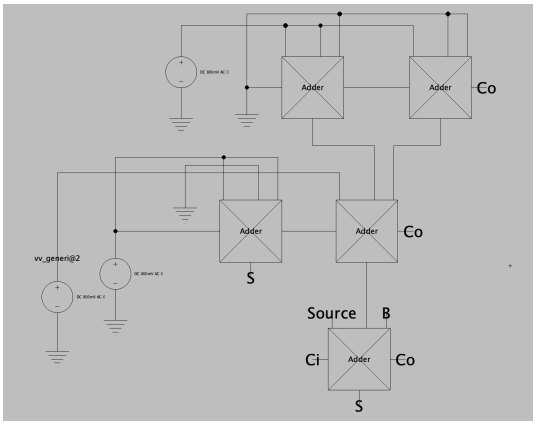
Note that the adder in the middle row is under test. Also, vv_generi@2 is labeled.

Case 6



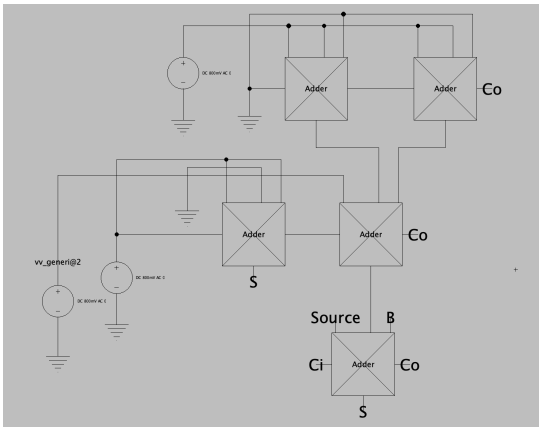
Note that the adder in the middle row is under test. Also, vv_generi@2 is labeled.

Case 7



Note that the adder in the middle row is under test. Also, vv_generi@2 is labeled.

Case 8



Note that the adder in the middle row is under test. Also, vv_generi@2 is labeled.

<u>Simulation Results (FOR 1-BIT SLICE)</u>

Case 1 leakage energy: **0.000875745 fJ**

Case 2 leakage energy: **0.00141079 fJ**
Case 3 leakage energy: **0.00166005 fJ**
Case 4 leakage energy: **0.00145392 fJ**
Case 5 leakage energy: **0.00136211 fJ**
Case 6 leakage energy: **0.00141662 fJ**
Case 7 leakage energy: **0.00167836 fJ**
Case 8 leakage energy: **0.000761954 fJ**

The Max Leakage Energy Case is A=1, B=0, C_{in}=1: 0.00167836 fJ
The Min Leakage Energy Case is A=1, B=1, C_{in}=1: 0.000761954 fJ

Because an 8-bit adder consists of 8 1-bit adder slices, we can multiply the maximum and minimum leakage energies for the 1-bit adder by 8 to get the maximum and minimum leakage energies for the 8-bit adder.

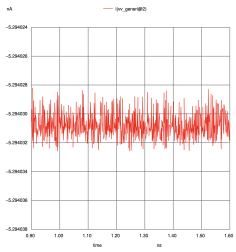
Maximum Leakage Energy for 8-Bit Adder: (8 * 0.00167836) = **0.01342688 fJ**
Minimum Leakage Energy for 8-Bit Adder: (8 * 0.000761954) = **0.006095632 fJ**

Commands (FOR 1-BIT SLICE)

tran 1ps 2000ps
plot I(vv_generi@2)
meas tran yint integ I(vv_generi@2) from=1000ps to=1165.4213ps

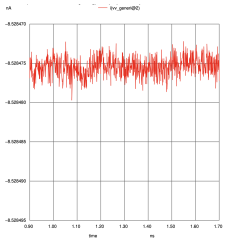
Plots for Each Case (FOR 1-BIT SLICE)

Case 1



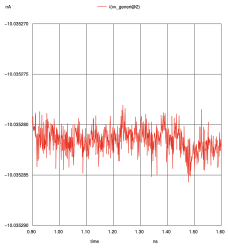
Note: vv_generi@2 is the current leaving the 1-bit adder.

Case 2



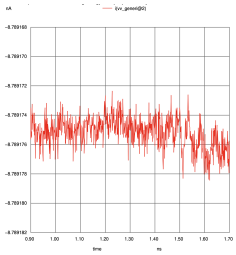
Note: vv_generi@2 is the current leaving the 1-bit adder.

Case 3



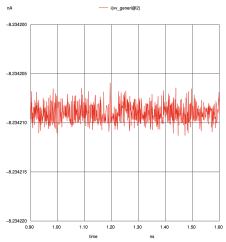
Note: vv_generi@2 is the current leaving the 1-bit adder.

Case 4



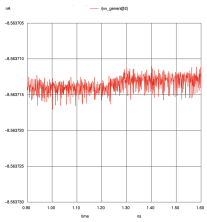
Note: vv_generi@2 is the current leaving the 1-bit adder.

Case 5



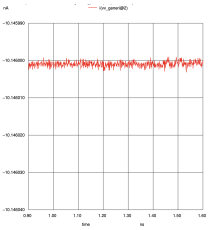
Note: vv_generi@2 is the current leaving the 1-bit adder.

Case 6



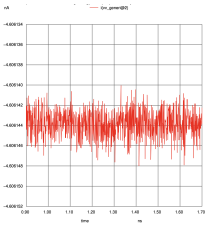
Note: vv_generi@2 is the current leaving the 1-bit adder.

Case 7



Note: vv_generi@2 is the current leaving the 1-bit adder.

Case 8



Note: vv_generi@2 is the current leaving the 1-bit adder.

Area

Calculations
Sum the transistor widths (there are 28 min-size transistors):
(28 * 1) = **28**

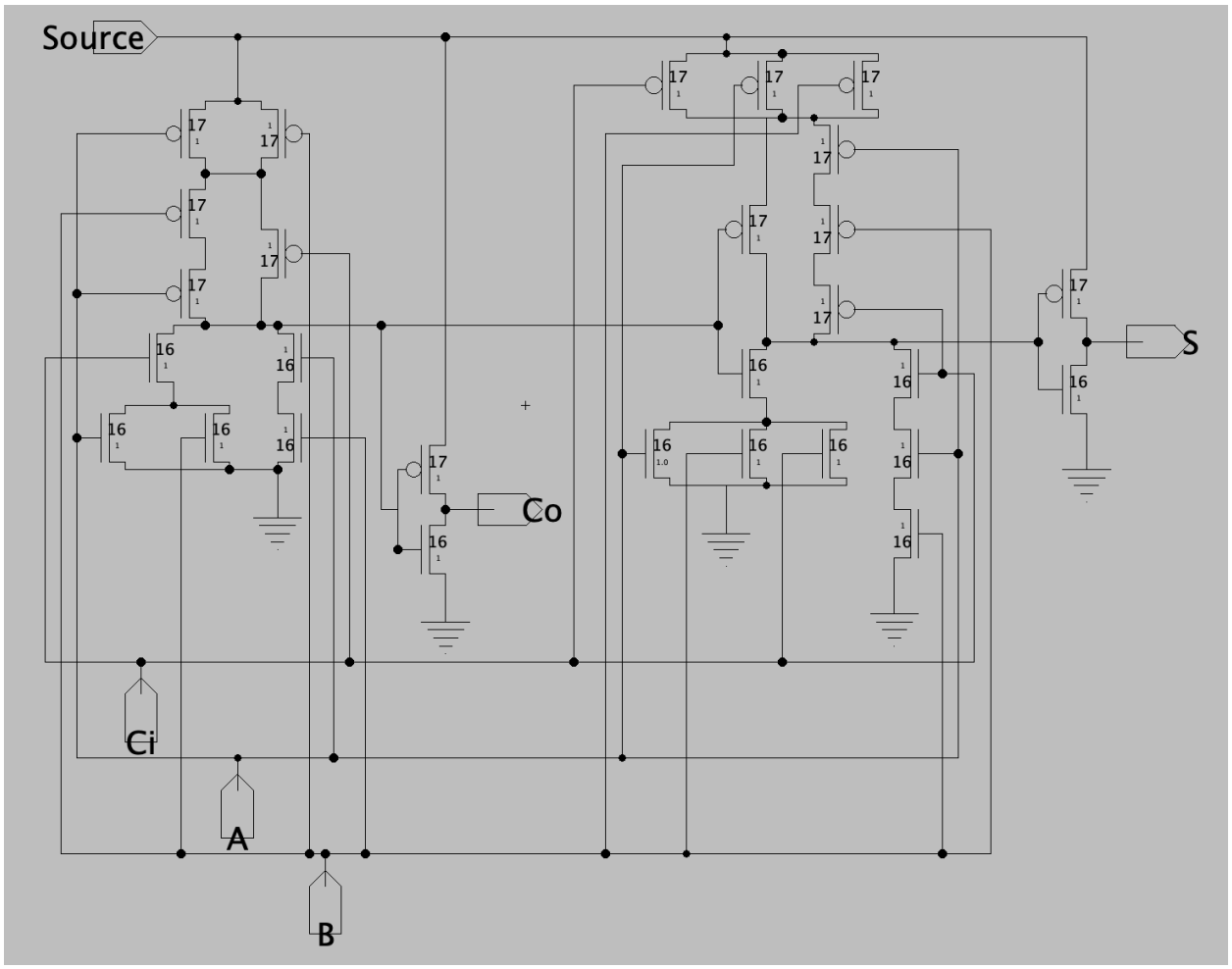
Summary Table

	Baseline Design
Worst Case Delay	**165.4213 ps**
Worst Case Switching	**2.58549413 fJ**
Average Case Switching	**1.3788944 fJ**
Max Leakage	**0.01342688 fJ**
Min Leakage	**0.006095632 fJ**

Optimized Design

Circuit

Schematic



Note: Source is V_{dd}, C_{in} and C_{out} are carry input and output, respectively. S is adder output.

Design from: Jan M. Rabaey, Anantha Chandrakasan, and Borivoje Nikolic. *Digital Integrated Circuits, A Design Perspective*, 2nd edition, Prentice Hall, 2003.

Description of Logic and Operation

The design above is the same design used in the baseline design, except the NMOS and PMOS transistors are sized to minimize the delay and V_{dd} is increased to 1.

Justification of Optimization Choices:

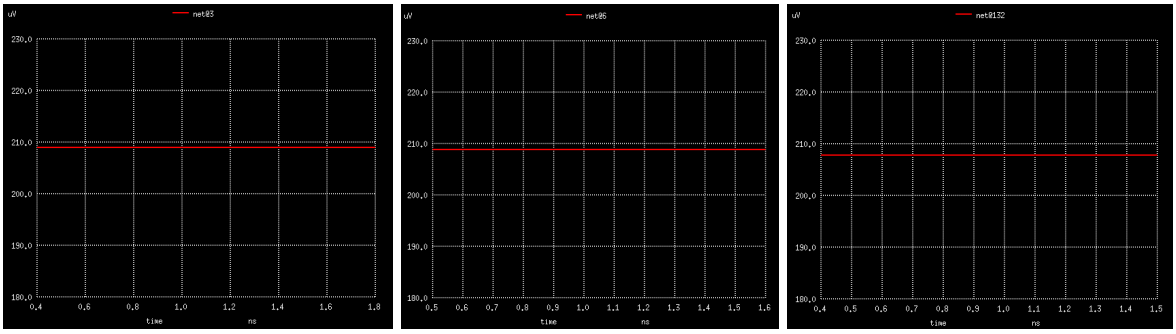
For the sizing of the NMOS and PMOS transistors, I used the delay calculations from the baseline calculations, but I created two new variables: W_p (width of PMOS) and W_n (width of NMOS). Plugging these in for $R_0 = R_0 / W$ and $C_0 = C_0W$, where the W is PMOS or NMOS

depending on switching case, I had a delay function based on W_p and W_n. Thus, I could find the W_p and W_n that minimized the delay function.
By making V_{dd} higher, there will be a higher voltage difference whenever a transistor gate is switching from 0→1 or 1→0. Thus, more current will be delivered to the transistor's gate, which allows more electrons/holes to contribute to the n-channel/p-channel path, thereby allowing the transistor to pull up/down its output more quickly.

Validation of Optimized Design

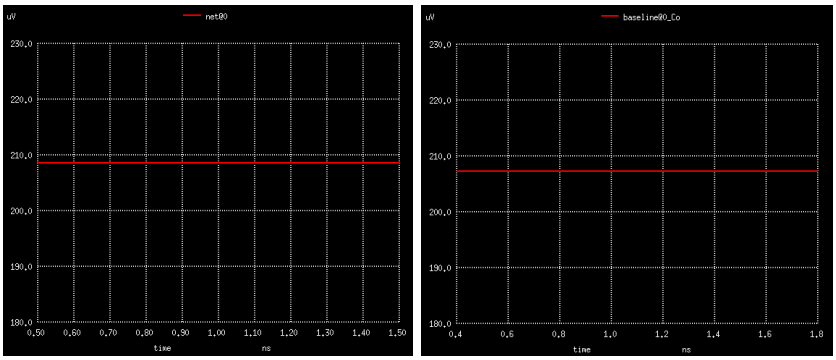
Case 1: A=0V, B=0V, C_{in}=0V

Inputs:



Note: net@132 is C_{in}, net@3 is A, and net@6 is B.

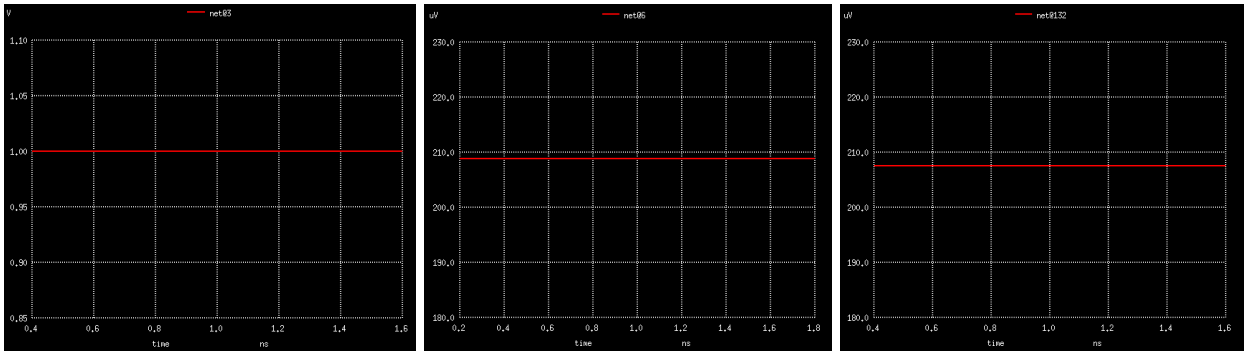
Outputs:



net@0 is S and baseline@0_Co is C_O

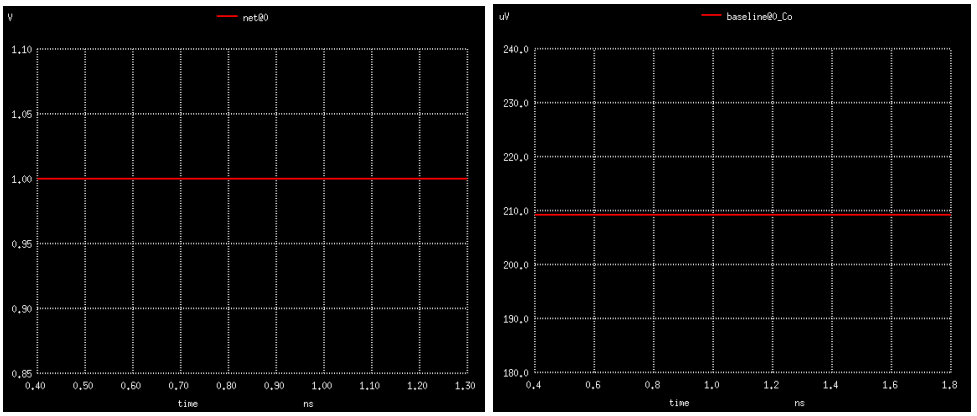
Case 2: A=0.8V, B=0V, C_{in}=0V

Inputs:



Note: net@132 is C_{in}, net@3 is A, and net@6 is B

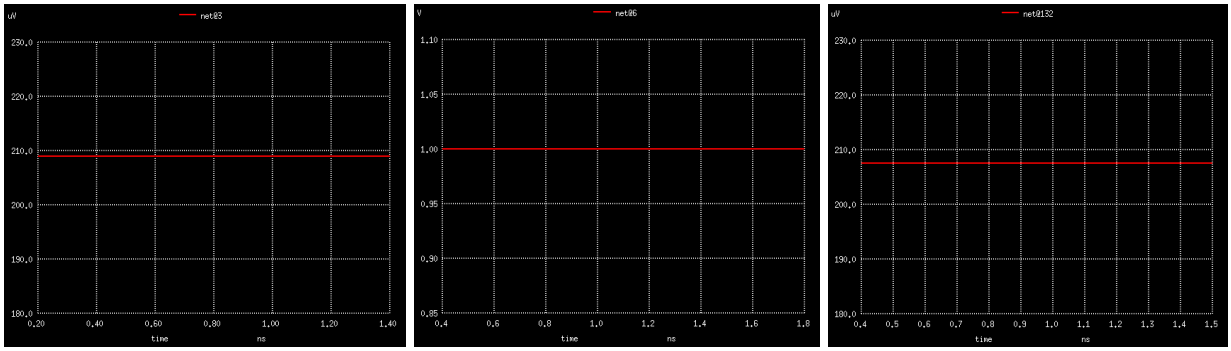
Outputs:



net@0 is S and baseline@0_Co is C_O

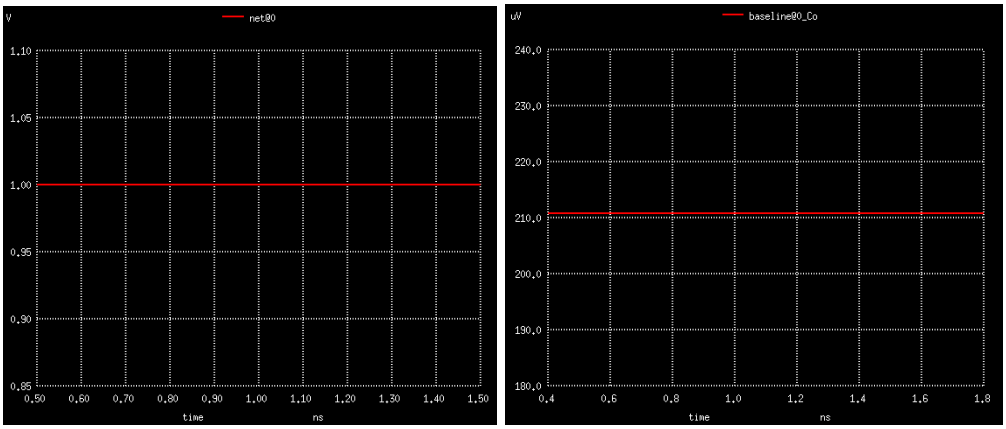
Case 3: A=0V, B=0.8V, C_{in}=0V

Inputs:



Note: net@132 is C_{in}, net@3 is A, and net@6 is B

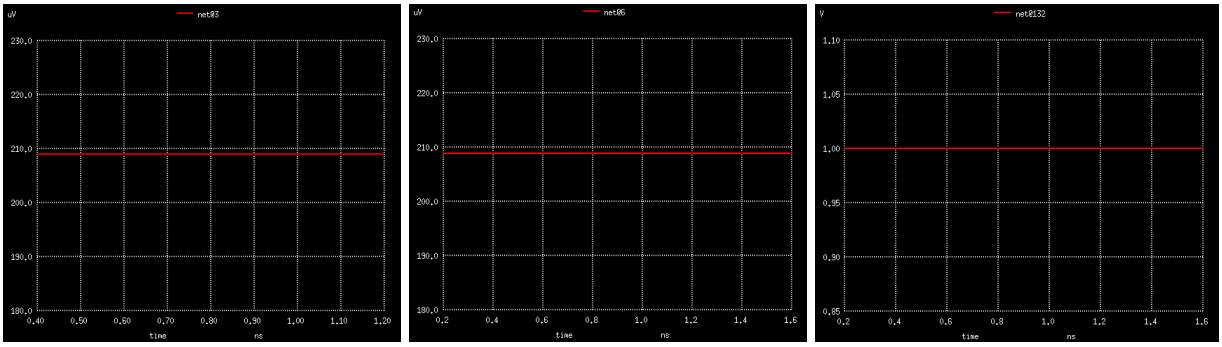
Outputs:



net@0 is S and baseline@0_Co is C_O

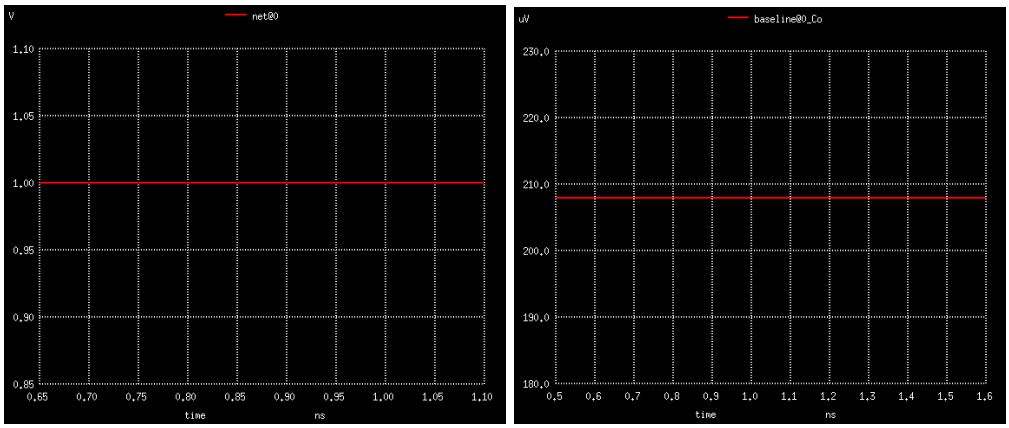
Case 4: A=0V, B=0V, C_{in}=0.8V

Inputs:



Note: net@132 is C_{in}, net@3 is A, and net@6 is B

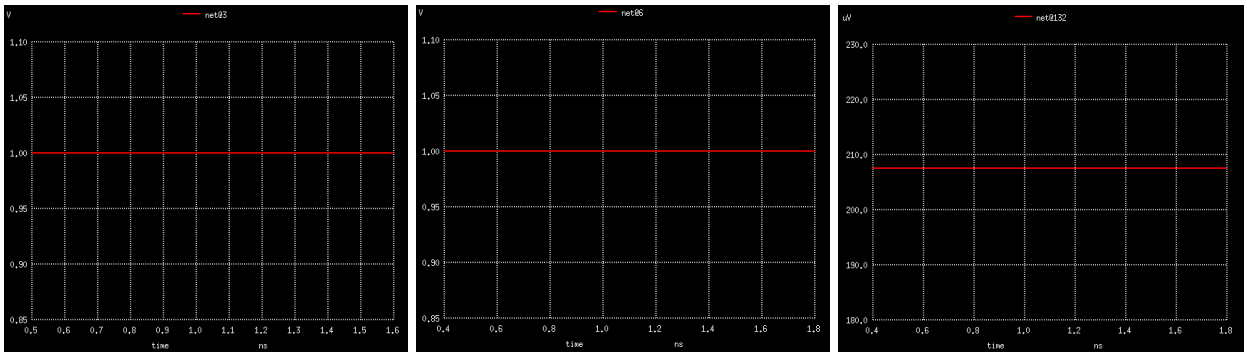
Outputs:



net@0 is S and baseline@0_Co is C_O

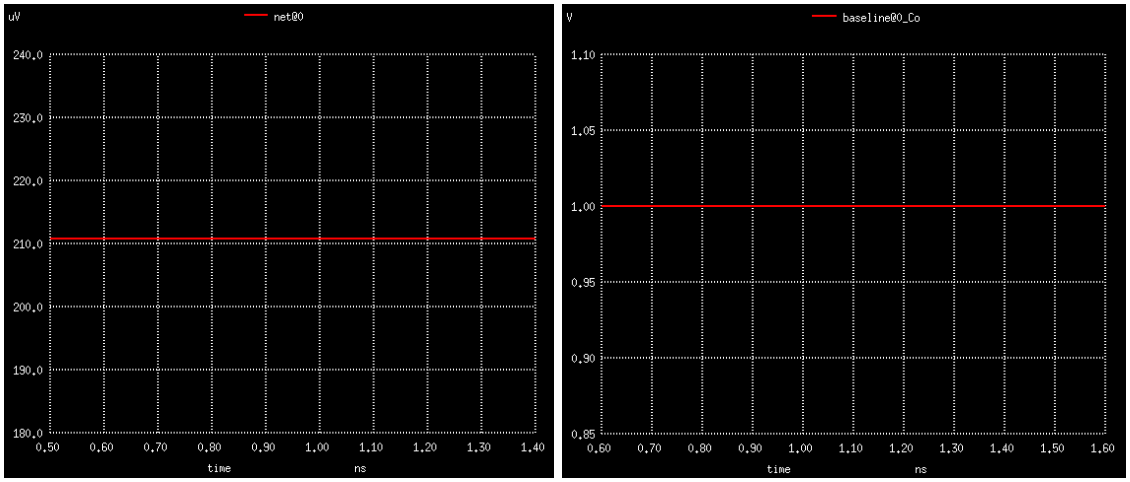
Case 5: A=0.8V, B=0.8V, C_{in}=0V

Inputs:



Note: net@132 is C_{in}, net@3 is A, and net@6 is B

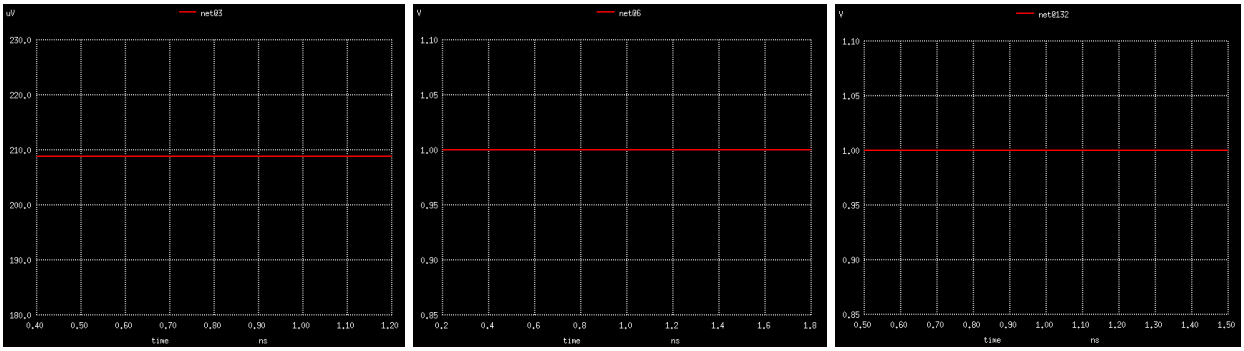
Outputs:



net@0 is S and baseline@0_Co is C_O

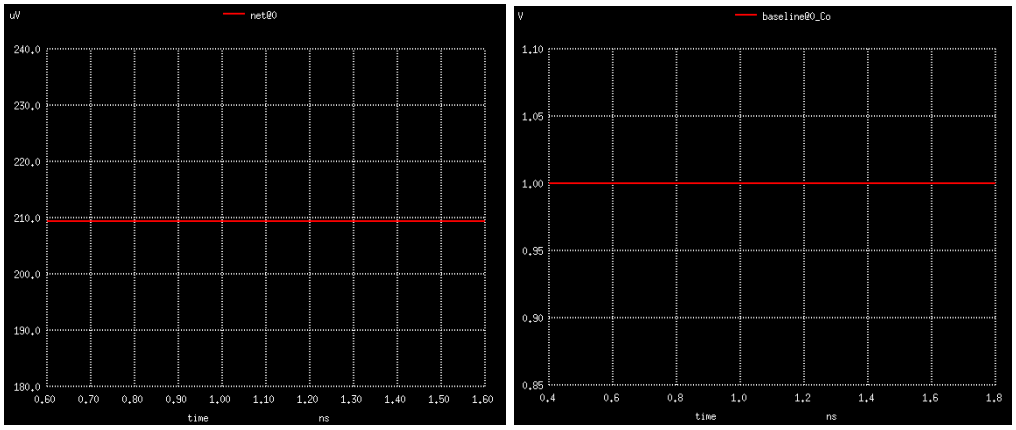
Case 6: A=0V, B=0.8V, C_{in}=0.8V

Inputs:



Note: net@132 is C_{in}, net@3 is A, and net@6 is B

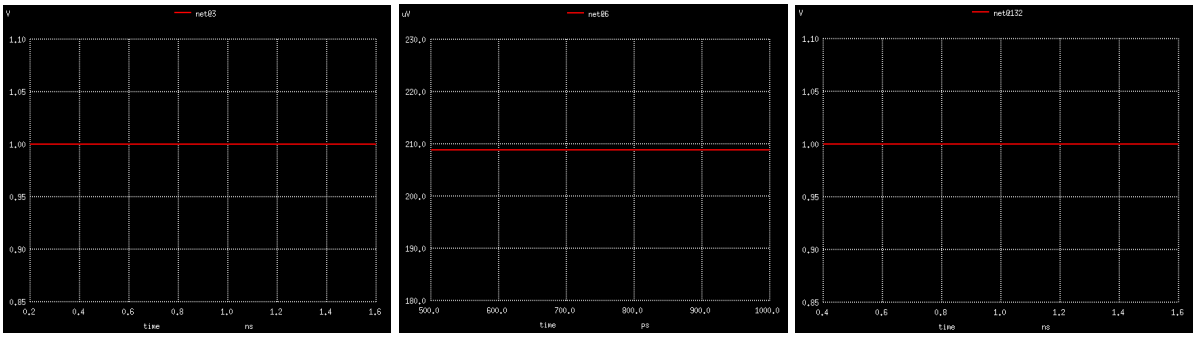
Outputs:



net@0 is S and baseline@0_Co is C_O

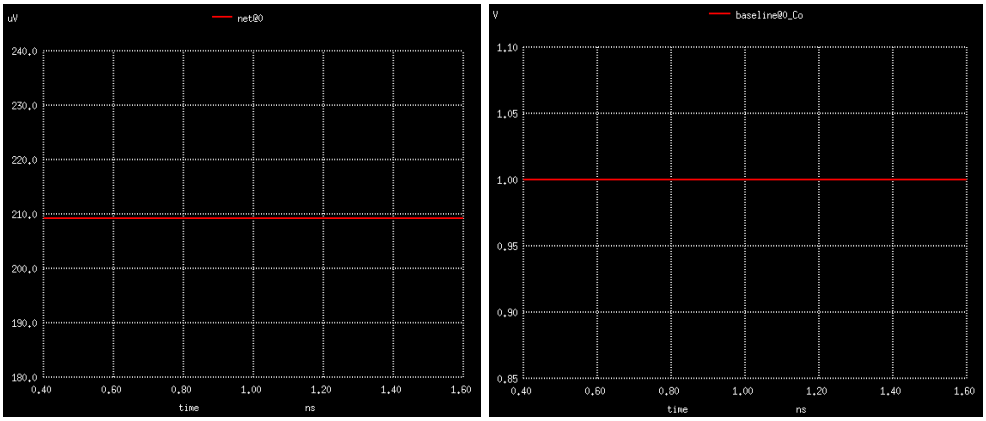
Case 7: A=0.8V, B=0V, C_{in}=0.8V

Inputs:



Note: net@132 is C_{in}, net@3 is A, and net@6 is B

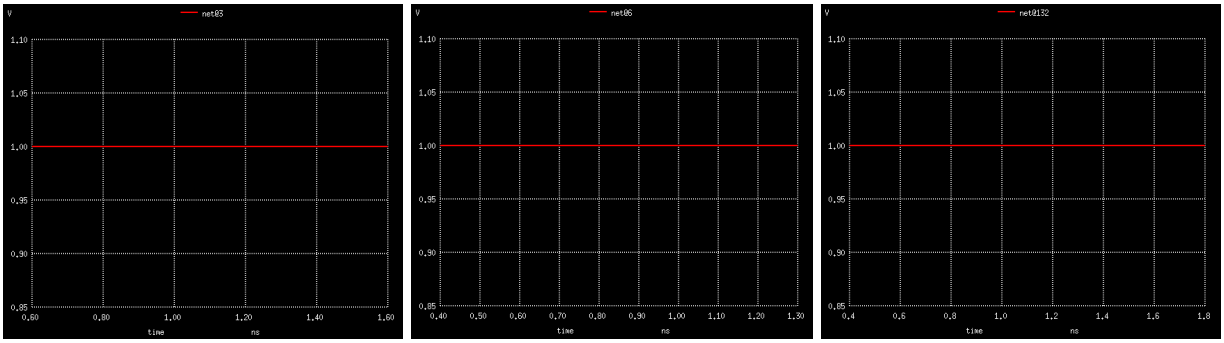
Outputs:



net@0 is S and baseline@0_Co is C_O

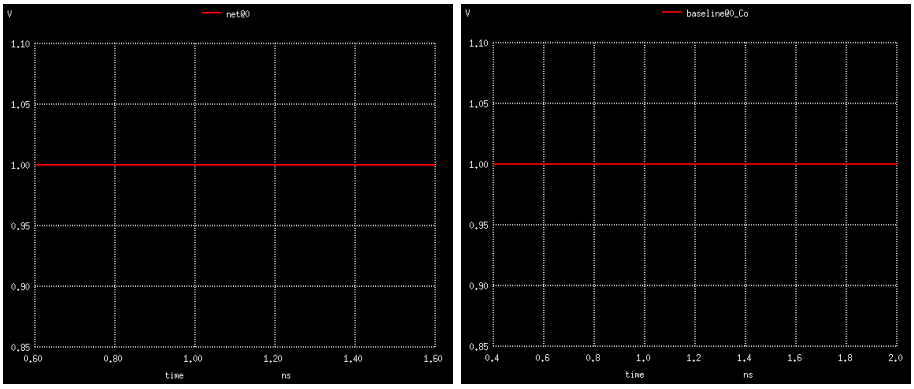
Case 8: A=0.8V, B=0.8V, C_{in}=0.8V

Inputs:



Note: net@132 is C_{in}, net@3 is A, and net@6 is B

Outputs:



net@0 is S and baseline@0_Co is C_O

Delay

Switching Case
Input 1 (A): 00000000 → 00000001 → 00000000
Input 2 (B): 011111111

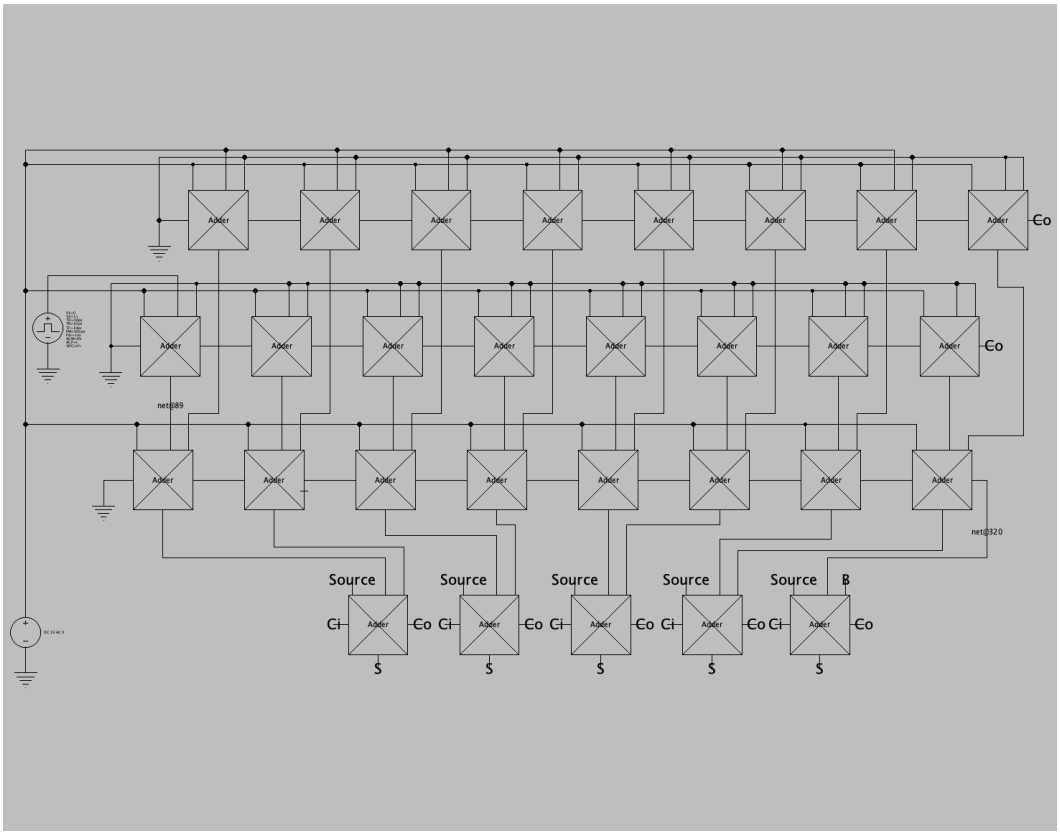
Optimization Width Calculations
Size PMOS as W_P and NMOS as W_N:
$((R_0 / W_P) * (4W_P + 4W_N)C_0) + 7(((2R_0/W_N)*(2W_N + 2W_P)C_0) + ((R_0/W_P) * (3W_P + 3W_N)C_0)) + ((5/2)(R_0/W_P)*(2W_N + 2W_P)C_0) + (2(R_0/W_N)*(W_N + W_P)C_0) + ((R_0/W_P)*(4W_P + 4W_N)C_0)$
Simplifying:
$4((W_P + W_N) / W_P) + 7(4((W_N + W_P) / W_N) + 3((W_P + W_N) / W_P)) + 5((W_N + W_P) / W_P) + 2((W_N + W_P) / W_N) +4((W_N + W_P) / W_P)$

Optimizing, let $W_P = x$ and $W_N = y$:
$4((x + y) / x) + 7(4((x + y) / y) + 3((x + y) / x)) + 5((x + y) / x) + 2((x + y) / y) + 4((x + y) / x)$
$F(x, y) = 34((x + y) / x) + 30((x + y) / y) = 30(x/y) + 34(y/x) + 64$
Optimizing, we get a local minima where $x \approx 1.03$ and $y \approx 0.97$
We can scale this down to $W_P = 17$ and $W_N = 16$
Also, increase V_{dd} to 1 (justification provided in Description of Logic and Operation section above).

Delay Calculations:
$(30(17/16) + 34(16/17) + 64)\boldsymbol{\tau}$
$= \mathbf{127.875\tau}$

Note: while the delay improvement may seem small, the delay is further decreased by increasing V_{dd} to 1V. The simulations show more than 25% in delay improvement.

Spice Test Schematic



Note: net@89 and net@147 are labeled on the schematic

Simulation Results

Worst case fall time: **121.5104 ps**

Commands:

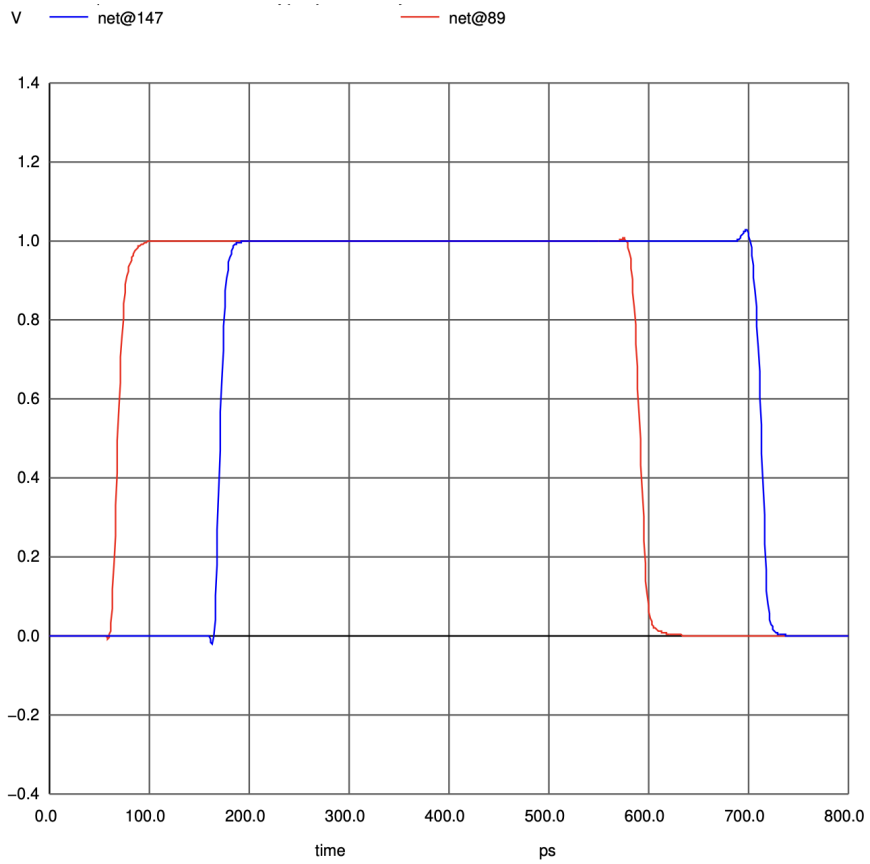
tran 1ps 2000ps
plot net@89 net@147
meas tran delay trig net@89 val=0.5 fall=1 targ net@147 val=0.5 fall=1
meas tran delay trig net@89 val=0.5 rise=1 targ net@147 val=0.5 rise=1

Vpulse Parameters:

V1=0, V2=1V, TD=50ps, TR=10ps, TF=10ps, PW=500ps, Per=1ns

Therefore, delay for optimized design improved by 26.5%

Plot (rise time and fall time)



Note: net@147 is output of 8th adder and net@89 is input of 1st adder (switching case)

Switching Energy

Switching Cases

Maximum Energy:

Input 1 (A): 00000000 → 111111111 → 00000000

Input 2 (B): 00000000 → 111111111 → 00000000

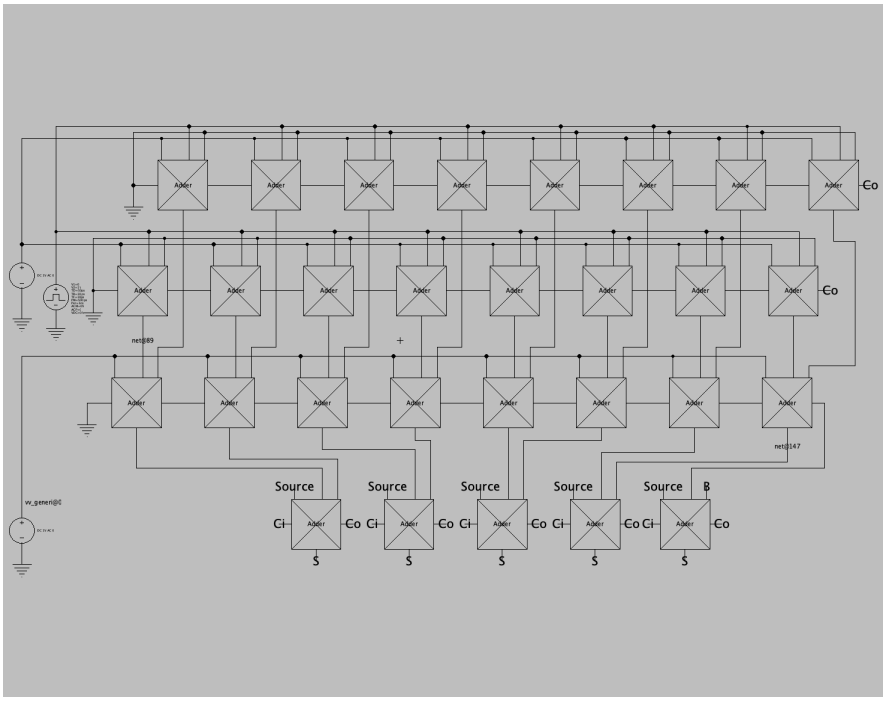
Average Energy

Input 1 (A): 00000000 → 00001111 → 00000000

Input 2 (B): 00000000 → 00001111 → 00000000

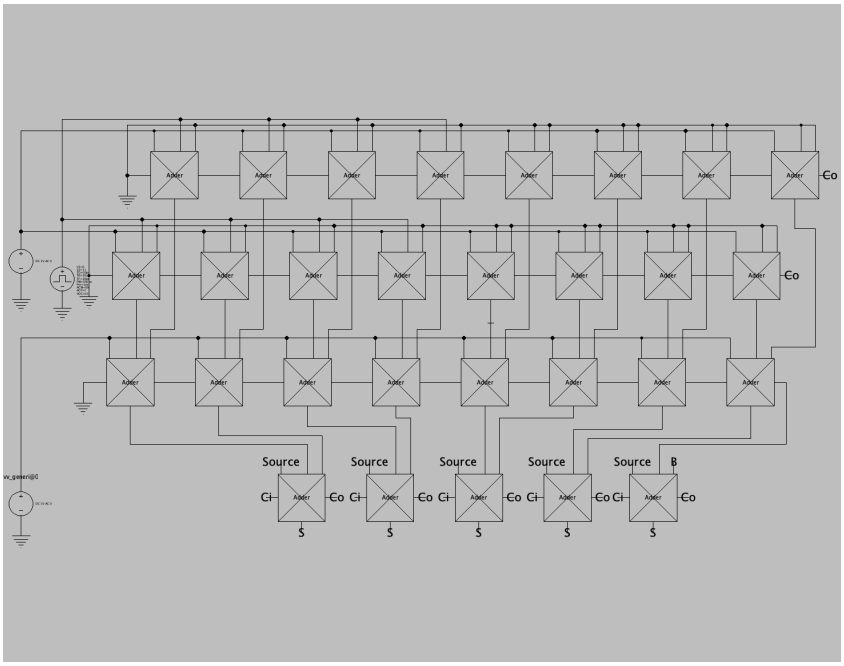
Spice Test Schematics

Max Energy Test Schematic



Note: vv_generi@0 is labeled on the schematic

Average Energy Test Schematic



Note: vv_generi@0 is labeled on the schematic

Simulation Results

Maximum Energy Consumption: **86.046638 fJ**
Average Energy Consumption: **46.314662 fJ**

NOTE: The SPICE current switching waveform had both positive area and negative area under the curve (see Plots section). To calculate the switching energy consumption, I integrated the positive and negative areas separately and summed their magnitudes.

Commands

Max Energy Commands:

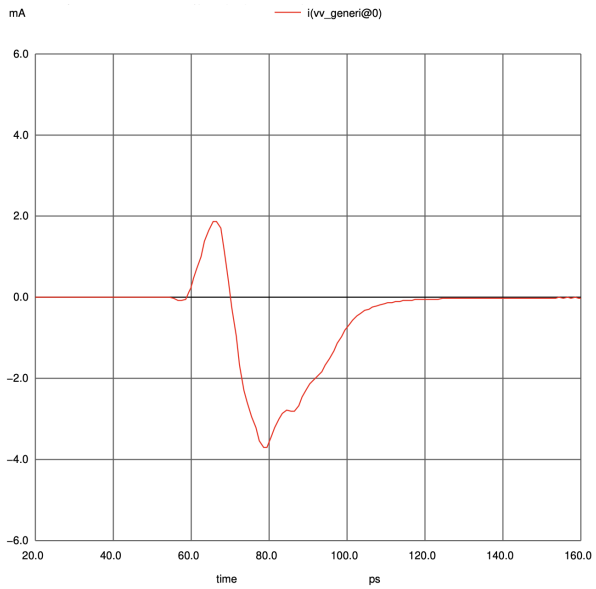
tran 1ps 2000ps
plot I(vv_generi@0)
meas tran yint integ I(vv_generi@0) from=54.5ps to=58.580ps (got -2.04138e-16)
meas tran yint integ I(vv_generi@0) from=58.580ps to=70.085ps (got 1.22061e-14)
meas tran yint integ I(vv_generi@0) from=70.085ps to=140ps (got -7.36364e-14)

Average Energy Commands:

tran 1ps 2000ps
plot I(vv_generi@0)
meas tran yint integ I(vv_generi@0) from=50.1ps to=58.815ps (got -1.39432e-16)
meas tran yint integ I(vv_generi@0) from=58.815ps to=70.90ps (got 5.95563e-15)
meas tran yint integ I(vv_generi@0) from=70.90ps to=150ps (got -4.02196e-14)

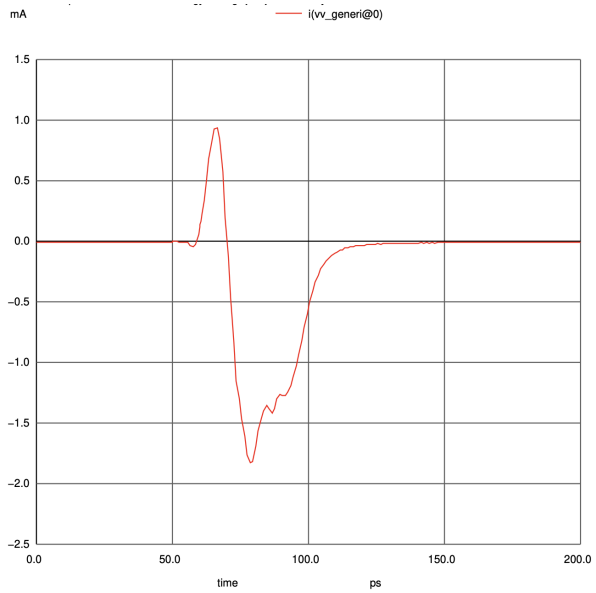
Plots

Max Energy Consumption. Current Entering Voltage Source.



Note: i(vv_generi@0) is current leaving the 8-bit adder under test. Both positive and negative areas under the curve were taken into account.

Average Energy Consumption. Current Entering Voltage Source.



Note: i(vv_generi@0) is current leaving the 8-bit adder under test. Both positive and negative areas under the curve were taken into account.

Leakage Energy

Cases (FOR 1-BIT SLICE)

There are 4 possible inputs into a 1-bit adder, so there are 4 cases:

Case 1
Input 1 (A): 0
Input 2 (B): 0
Input 3 (C_{in}): 0

Case 2
Input 1 (A): 1
Input 2 (B): 0
Input 3 (C_{in}): 0

Case 3
Input 1 (A): 0
Input 2 (B): 1
Input 3 (C_{in}): 0

Case 4
Input 1 (A): 0
Input 2 (B): 0
Input 3 (C_{in}): 1

Case 5
Input 1 (A): 1
Input 2 (B): 1
Input 3 (C_{in}): 0

Case 6
Input 1 (A): 0
Input 2 (B): 1
Input 3 (C_{in}): 1

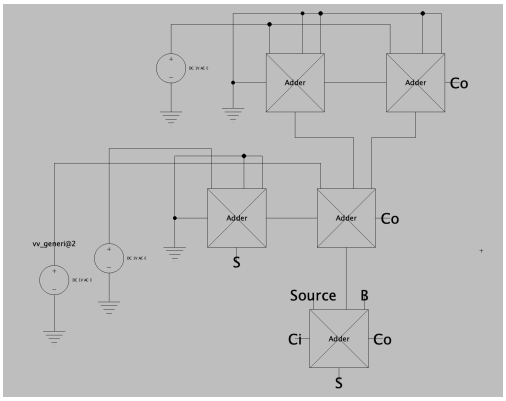
Case 7
Input 1 (A): 1
Input 2 (B): 0
Input 3 (C_{in}): 1

Case 8
Input 1 (A): 1
Input 2 (B): 1
Input 3 (C_{in}): 1

From this, the maximum and minimum leakage energies can be calculated.

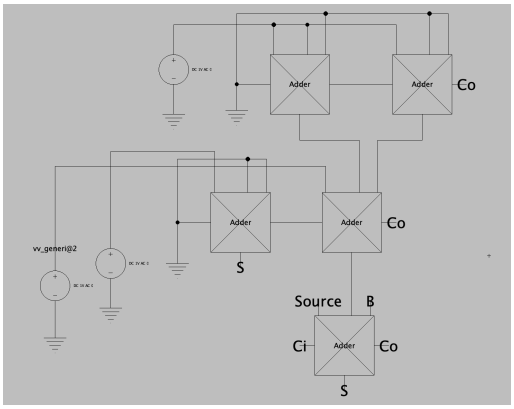
Spice Test Schematics (FOR 1-BIT SLICE)

Case 1



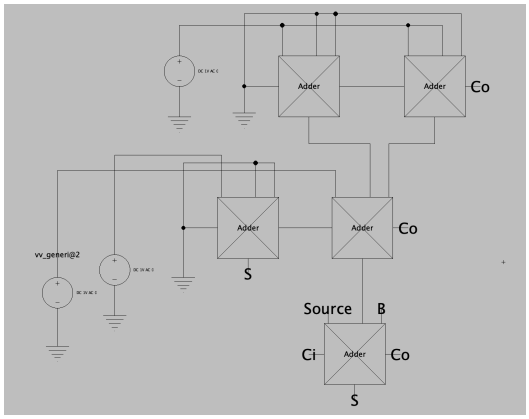
Note that the adder in the middle row is under test. Also, vv_generi@2 is labeled.

Case 2



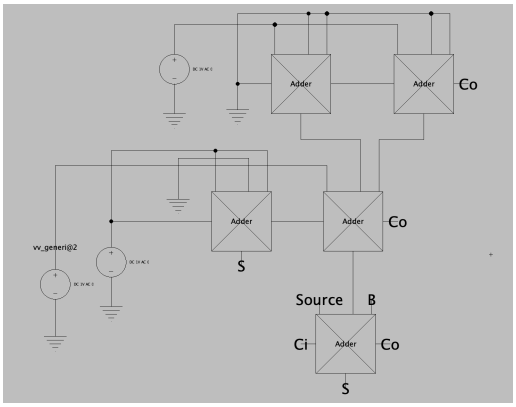
Note that the adder in the middle row is under test. Also, vv_generi@2 is labeled.

Case 3



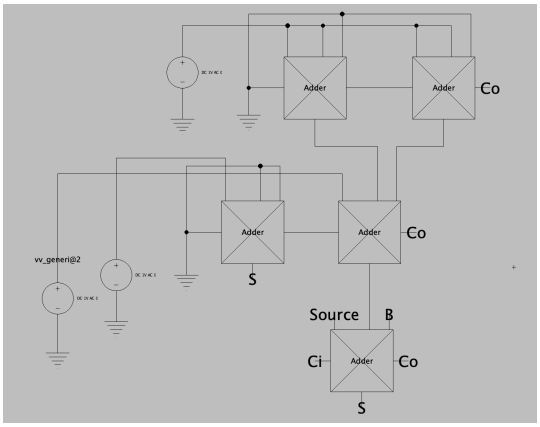
Note that the adder in the middle row is under test. Also, vv_generi@2 is labeled.

Case 4



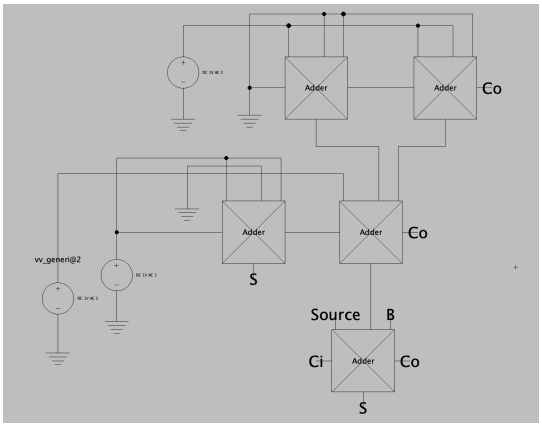
Note that the adder in the middle row is under test. Also, vv_generi@2 is labeled.

Case 5



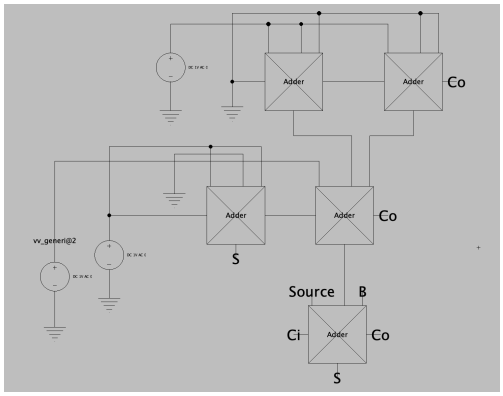
Note that the adder in the middle row is under test. Also, vv_generi@2 is labeled.

Case 6



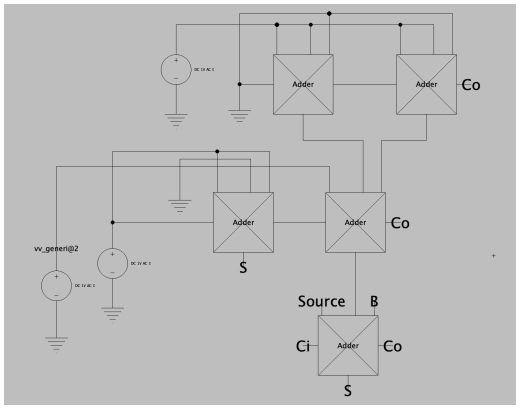
Note that the adder in the middle row is under test. Also, vv_generi@2 is labeled.

Case 7



Note that the adder in the middle row is under test. Also, vv_generi@2 is labeled.

Case 8



Note that the adder in the middle row is under test. Also, vv_generi@2 is labeled.

<u>Simulation Results (FOR 1-BIT SLICE)</u>

Case 1 leakage energy: **0.0969752 fJ**
Case 2 leakage energy: **0.134074 fJ**
Case 3 leakage energy: **0.152013 fJ**
Case 4 leakage energy: **0.156178 fJ**
Case 5 leakage energy: **0.109096 fJ**
Case 6 leakage energy: **0.133749 fJ**
Case 7 leakage energy: **0.171470 fJ**
Case 8 leakage energy: **0.0479209 fJ**

FOR 1-bit OPTIMIZED ADDER:
The Max Leakage Energy Case is A=1, B=0, C_{in}=1: **0.171470 fJ**
The Min Leakage Energy Case is A=1, B=1, C_{in}=1: **0.0479209 fJ**

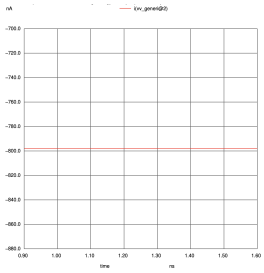
Because an 8-bit adder consists of 8 1-bit adder slices, we can multiply the maximum and minimum leakage energies for the 1-bit adder by 8 to get the maximum and minimum leakage energies for the 8-bit adder.

Maximum Leakage Energy for 8-Bit Optimized Adder: (8 * 0.171470) = **1.37176 fJ**
Minimum Leakage Energy for 8-Bit Optimized Adder: (8 * 0.0479209) = **0.3833672 fJ**

Commands (FOR 1-BIT SLICE)
tran 1ps 2000ps
plot I(vv_generi@2)
meas tran yint integ I(vv_generi@2) from=1000ps to=1121.5104ps

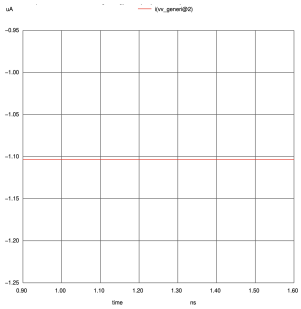
Plots for Each Case (FOR 1-BIT SLICE)

Case 1



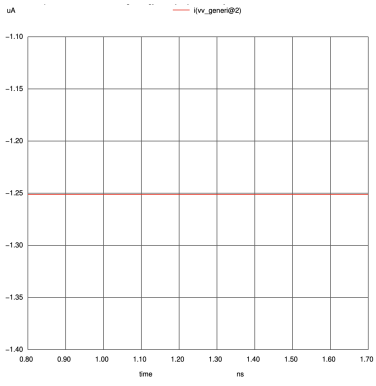
Note: vv_generi@2 is the current leaving the 1-bit adder.

Case 2



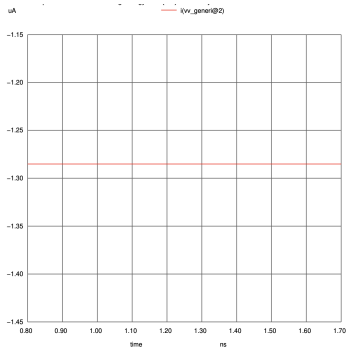
Note: vv_generi@2 is the current leaving the 1-bit adder.

Case 3



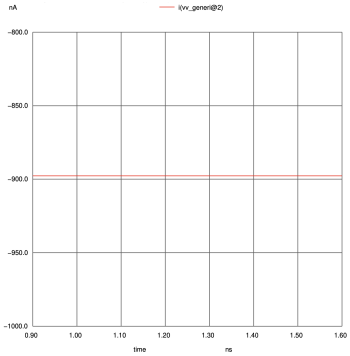
Note: vv_generi@2 is the current leaving the 1-bit adder.

Case 4



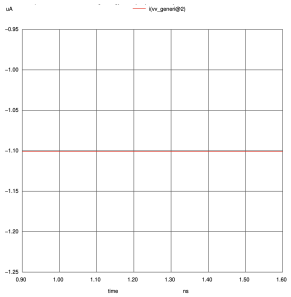
Note: vv_generi@2 is the current leaving the 1-bit adder.

Case 5



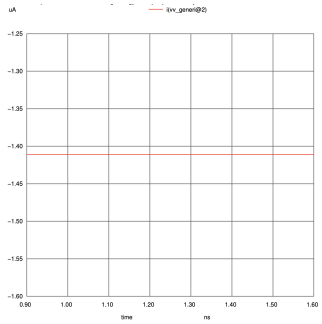
Note: vv_generi@2 is the current leaving the 1-bit adder.

Case 6



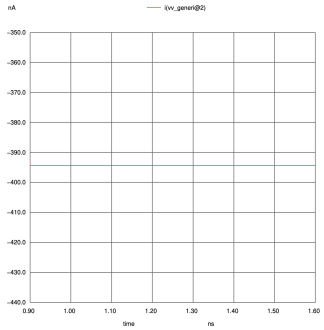
Note: vv_generi@2 is the current leaving the 1-bit adder.

Case 7



Note: vv_generi@2 is the current leaving the 1-bit adder.

Case 8



Note: vv_generi@2 is the current leaving the 1-bit adder.

Area

Calculations

Sum the transistor widths (there are 14 PMOS transistors and 14 NMOS transistors):
((14*16) + (14*17)) = **462**

Summary Table

	Optimized Design
Worst Case Delay	**121.5104 ps**
Worst Case Switching	**86.046638 fJ**
Average Case Switching	**46.314662 fJ**
Max Leakage	**1.37176 fJ**
Min Leakage	**0.3833672 fJ**

Questions

1. I explored 6 ideas for optimizing the 1-bit adder slice.
   The first two ideas were increasing V_{dd} and sizing the NMOS and PMOS transistors, which would be relatively easy (the CMOS design would not have to change) and effective in lowering delay.
   Another idea was using pass transistor logic. Pass transistor logic would make the circuit much simpler and decrease the footprint, especially considering an XOR could be implemented using just one NMOS and an inverter. However, the issue with pass transistor logic was that it didn't pass 1s or 0s very well, which would mean I would need to add an inverter to achieve regeneration. Therefore, this would further complicate the design and would not decrease the delay by much.
   Another idea was inverting all the inputs of all the odd adder slices, which would eliminate the need for inverters in the carry chain. This idea is elaborated on in the textbook. While this would slightly decrease the delay, it would require me to design two different adder slices, which would lead to a lot of debugging issues.
   I also considered implementing my own XOR gates that I could then integrate into an 8-bit adder slice. However, when further considering this, an XOR gate implemented in CMOS logic would require a lot of transistors, and using multiple XORs would require even more. This would probably lead to a worse delay calculation and a lot of debugging issues.
   The final idea I explored was using the Manchester Carry Adder design provided in the textbook. However, this design used pass transistor logic, which has difficulty effectively passing 1s and 0s effectively, so a regenerative element would be needed. Also, the design is ideal for 4-bit adders, but for 8-bit adders, the propagation delay would be very high because the signal would have to propagate through the entire adder.
2. From considering the pass transistor logic and the Manchester Carry Adder designs, I learned that while some designs would yield less transistors and thus a smaller area, that didn't always imply a smaller delay and better functionality. In the case of using pass transistors, more problems would arise about regeneration. For inverting odd adder slice inputs ideas (detailed in textbook) and the idea for creating my own XOR gates, I learned that it is important to have a relatively single design for an adder slice that I could debug, rather than multiple designs that would all require debugging and validation.
3. The area and delay of the adder would decrease linearly as you reduce the number of bits the adder can "add". For area, this is because you are decreasing the number of adder slices, and since each adder slice contains some amount of transistors, you will be decreasing the area by some constant amount for each bit you remove from the adder. Next, for delay, the worst case delay will always be propagating from the LSB adder slice to the MSB adder slice. The delay for each adder will remain constant, but as you decrease the number of adders the signal has to propagate through, the worst case delay will decrease by some constant amount for each adder slice you remove.

Statement of Academic Integrity

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