# Multiplication and Accumulation (MAC) Datapath for Neural Networks
# Final Report

Dalton Reynolds (663275953), Lucas Bigos (671385506),
Tessa Urwin (661934267), Julian O'Hern (652167453)

# I. Introduction

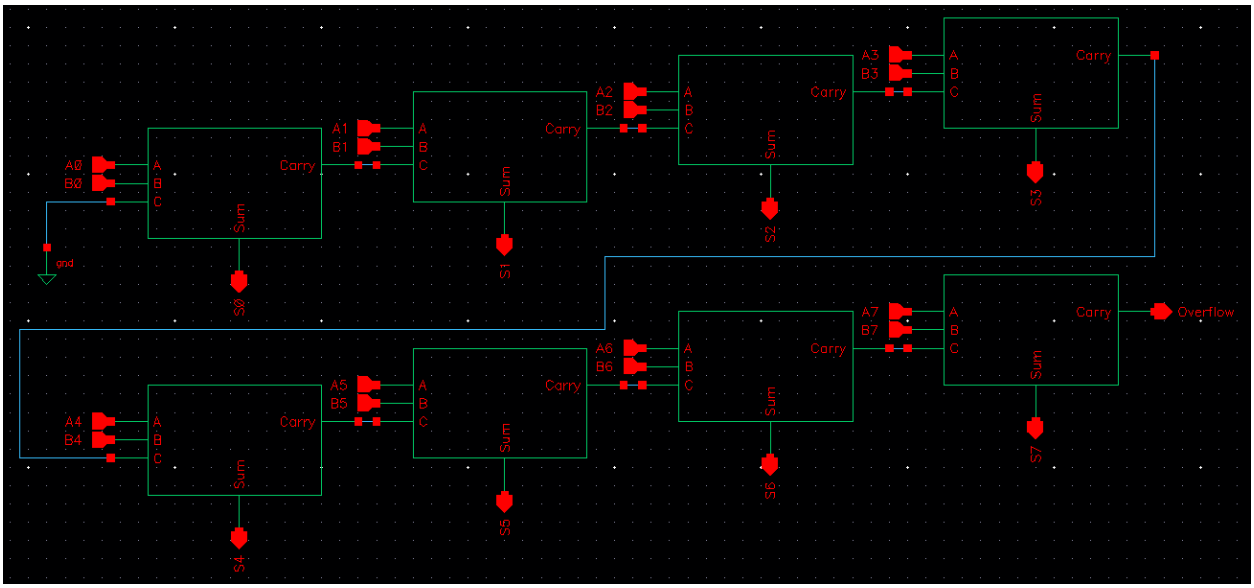
This project aimed to design and implement a Multiplication and Accumulation (MAC) Datapath as a demonstration of the digital design concepts covered in ECE 467. The MAC unit is an essential and highly utilized component that performs multiplication and addition operations efficiently. Although we were unable to fully integrate the complete datapath in the timeframe, each of the major sub-blocks, such as the inverter chain, logic gates library, and partial datapath components, was designed, simulated, and verified individually. Our key results showed expected functional behavior and confirmed that circuit performance improves at lower temperatures, with the inverter working best with about 5 steps, and optimal delay near a supply voltage of 1.22 V. A detailed summary of these results is presented in section IV.

# II. System Overview

The intended MAC datapath architecture consists of multiple submodules connected in a pipelined structure, including a multiplier, adder, registers, and control logic. The design focus was on achieving optimal performance through optimized transistor sizing and minimal propagation delay. Each building block was implemented and tested individually in Cadence Virtuoso. Due to time constraints, full system integration and testing were not completed. However, we were able to design the system blocks and test them individually to confirm correct functionality and compatibility for integration in a larger datapath, so we provide schematics for those blocks. With additional time, we would connect these blocks under a shared clock and verify the full functionality of the completed MAC.

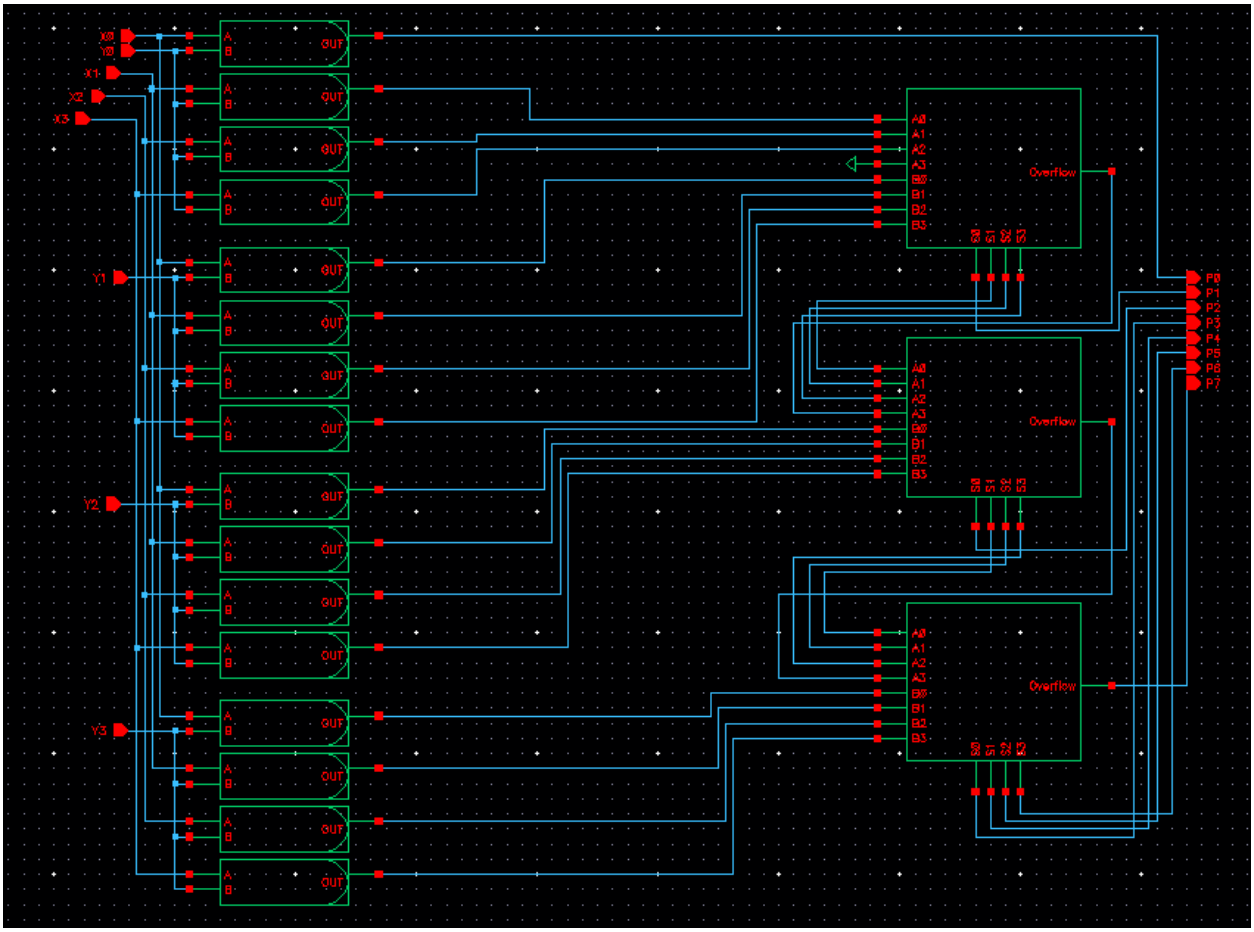
## 8-bit Adder

After creating a full bit adder, 8 of them were strung in series together to form a 8 bit full adder. This was a low performance piece handling two sets of 8 bit inputs and creating one in return.



## 4-bit Multiplier

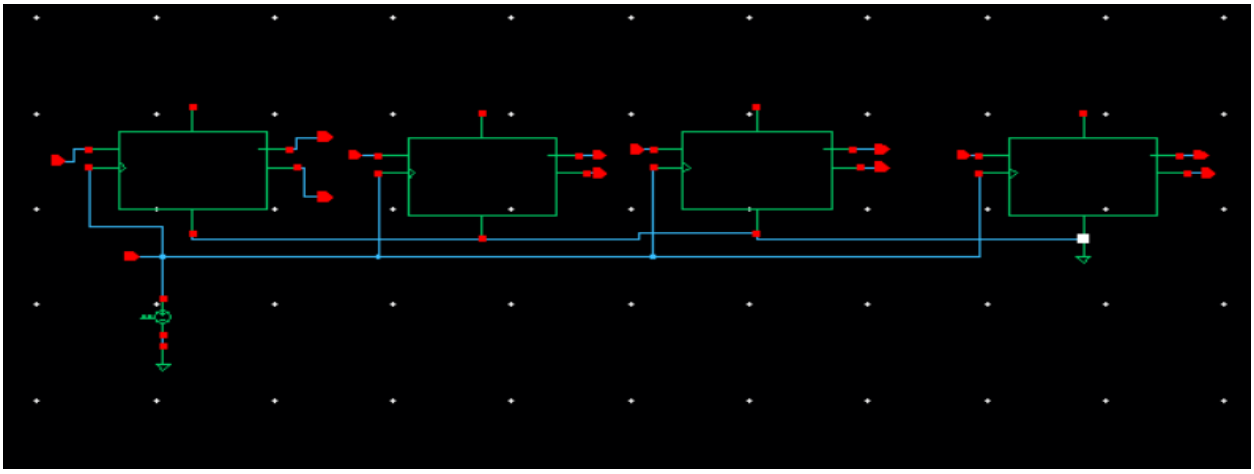
The 4-bit multiplier was built using three 4-bit adders, tied together with multiple AND gates used for most inputs.




In the overview A3 of the first 4 bit adder is tied to ground as it acts like the Carry-In of the whole system, since it operates independently and there wouldn't be any value for it, it's tied to ground.

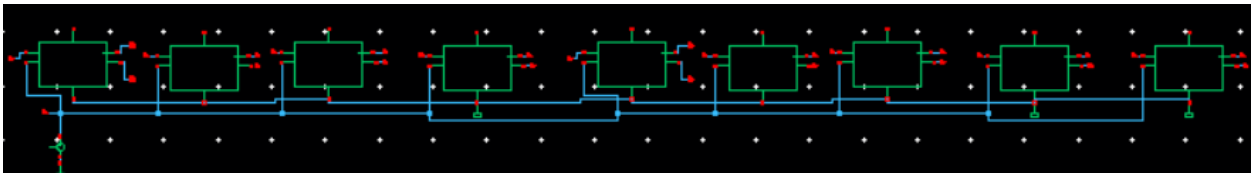
## 4-bit Registers

Four flip-flops chained together like the one demonstrated in class.



## 9-bit Registers

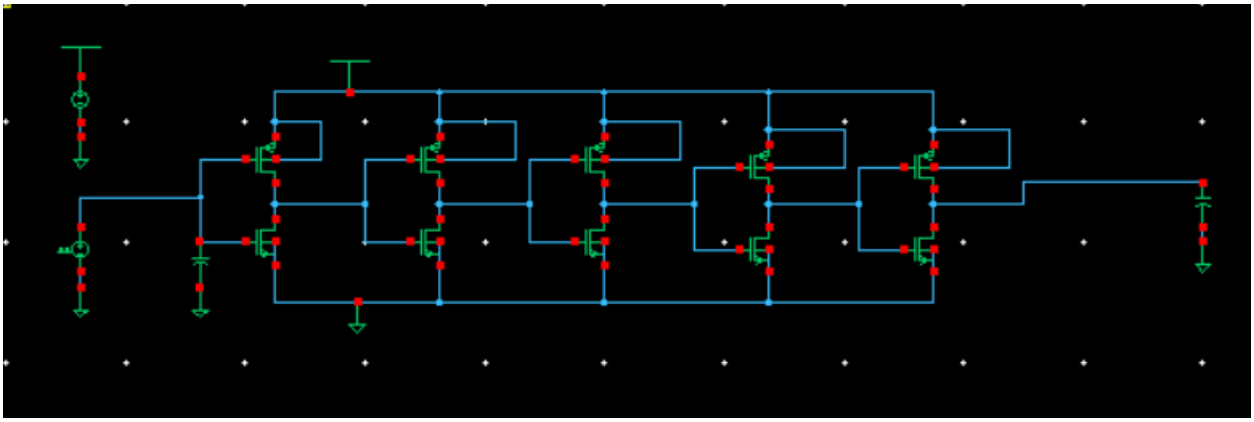
Nine flip-flops chained together like the one demonstrated in class.



# III. Component-Level Design Styles

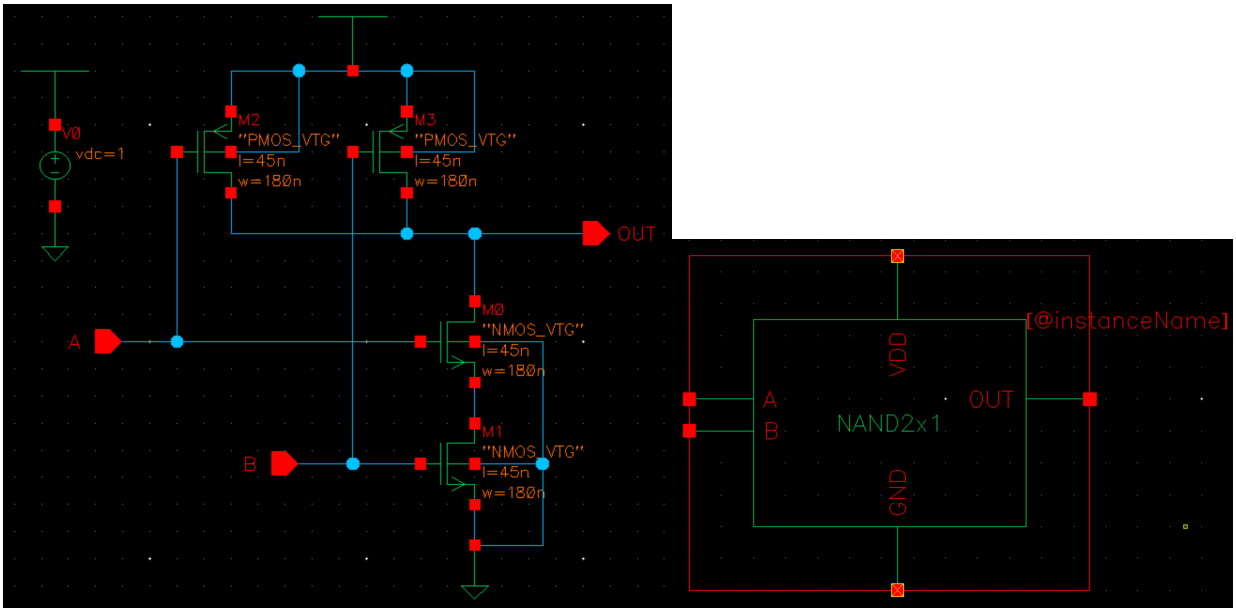
## Inverter Chain

A simple inverter like those demonstrated in class just chained together five times to produce an inverter chain with the optimal number of gates.



## 2-input NAND

The 2-input NAND gate was implemented using static complementary CMOS logic. The pull-up network consists of two PMOS transistors in parallel, and the pull-down network consists of two NMOS transistors in series. Inputs A and B each drive one PMOS and one NMOS device.

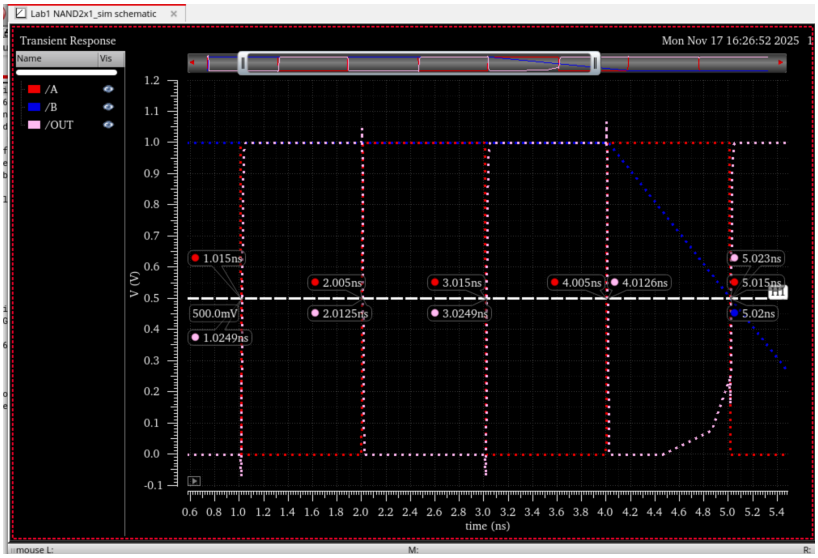


Width sizing for NAND2X1, NAND2X2, NAND2X3:

The pull-down uses two NMOS in series, so each NMOS is sized to 180 nm (90 nm x 2) to keep the overall pull-down strength equal to the reference inverter. The pull-up has two PMOS in parallel, so each PMOS uses the same PMOS width as the inverter (180 nm).

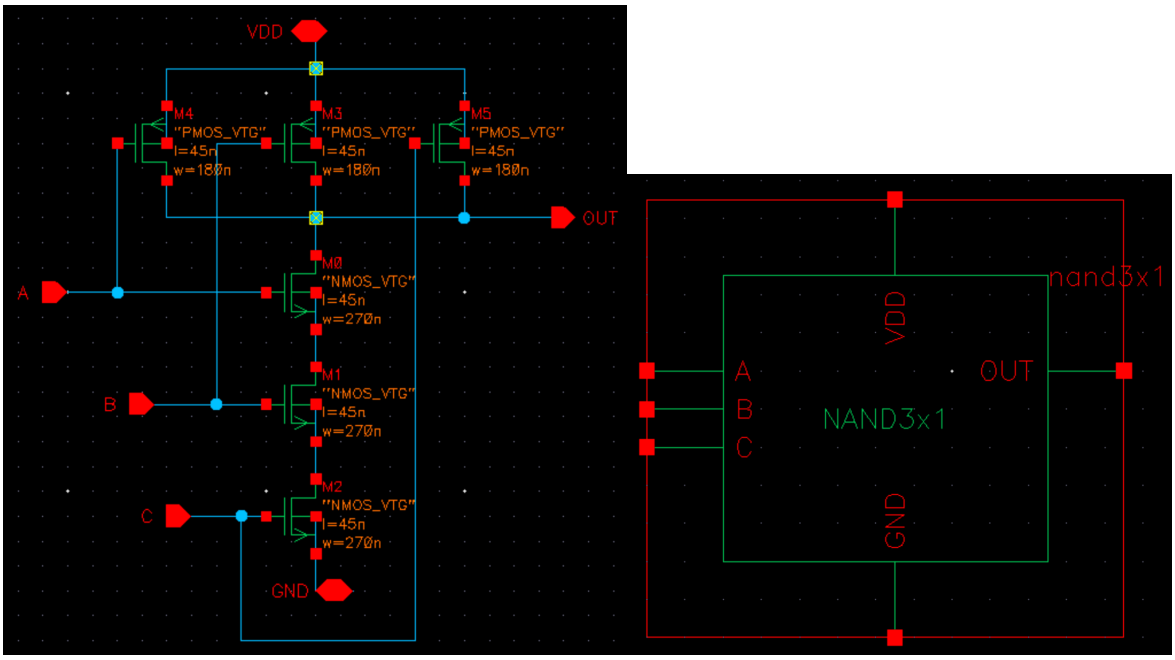
Drive strength iterations all scale widths uniformly:

- NAND2x1: base widths = 180 nm
- NAND2x2: widths scaled by 2 = 360 nm
- NAND2x3: widths scaled by 3 = 540 nm




## 3-input NAND

The 3-input NAND gate was implemented using static complementary CMOS logic. The pull-up network consists of three PMOS transistors in parallel, and the pull-down network consists of three NMOS transistors in series. Inputs A, B, and C each drive one PMOS and one NMOS device.

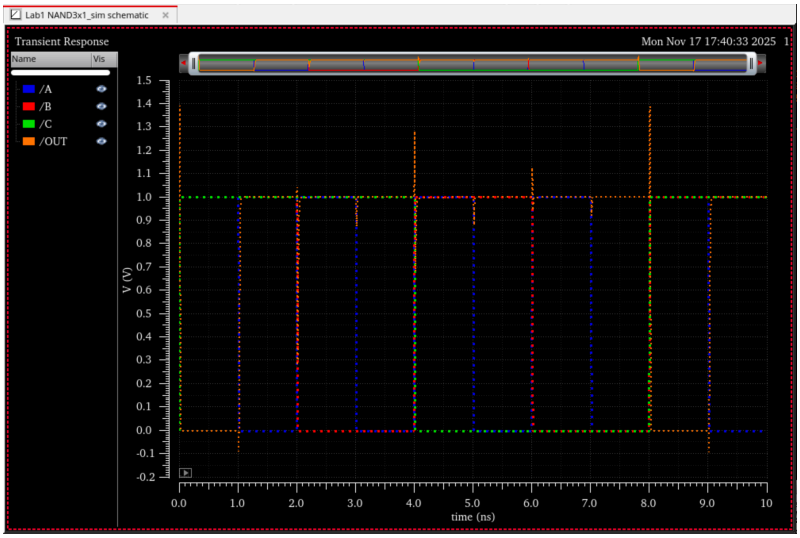



Width sizing for NAND3X1, NAND3X2, NAND3X3:

The pull-down uses three NMOS in series which increases resistance by a factor of three, so each NMOS is sized to 270 nm (90 nm x 3) to match the inverter strength. The pull-up has three PMOS in parallel, so each PMOS uses the inverter PMOS width (180 nm).

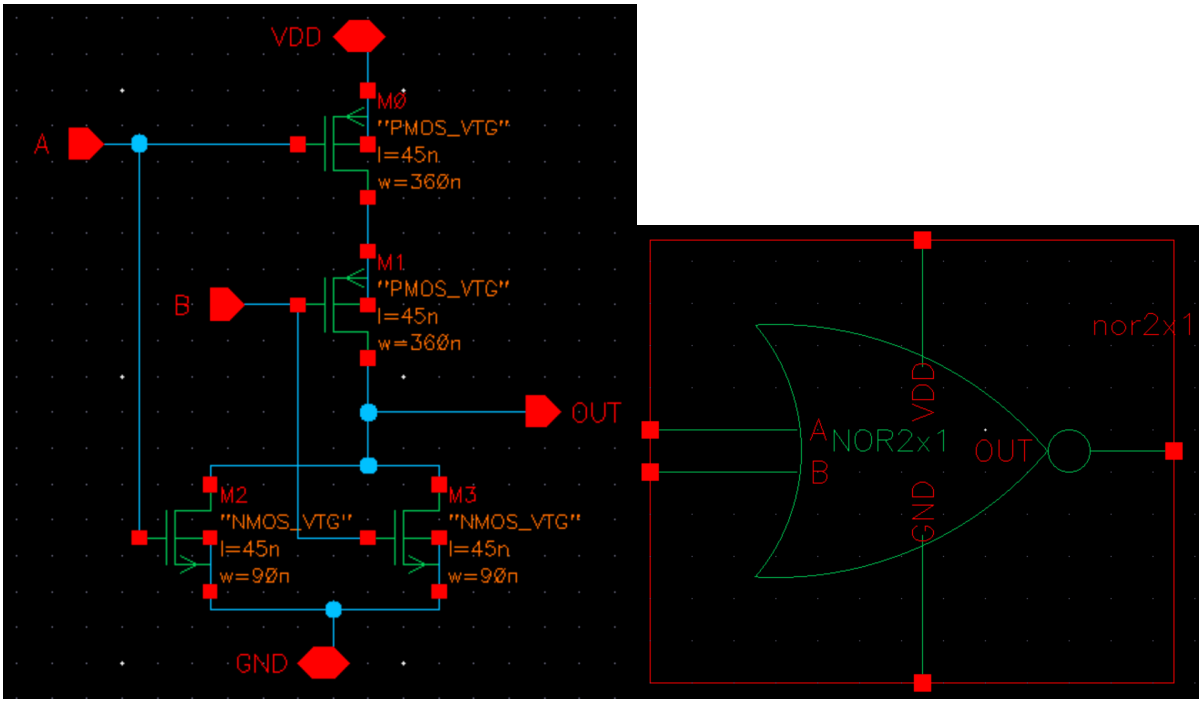
Drive strength iterations all scale widths uniformly:

- NAND3x1: base = NMOS 270 nm / PMOS 180 nm
- NAND3x2: widths scaled by 2 = NMOS 540 nm / PMOS 360 nm
- NAND3x3: widths scaled by 3 = NMOS 810 nm / PMOS 540 nm



## 2-input NOR

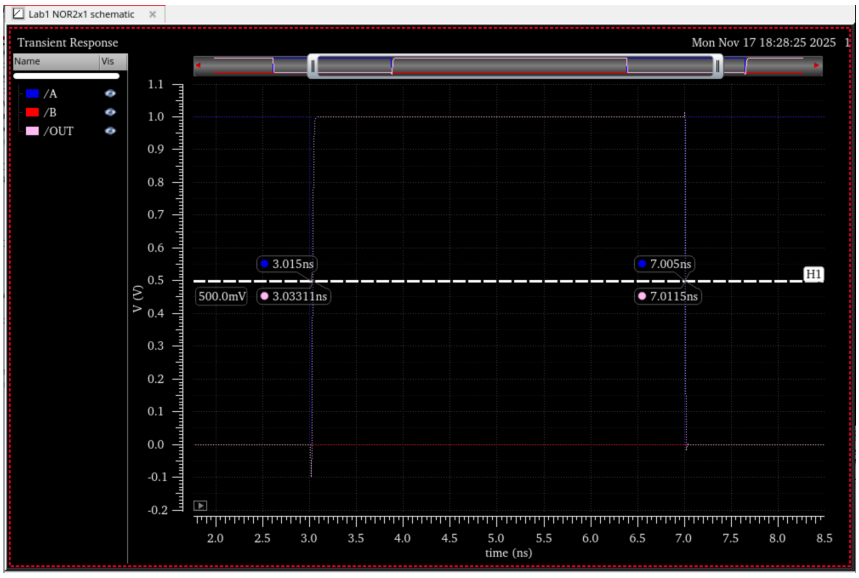
The 2-input NOR gate was implemented using static complementary CMOS logic. The pull-up network consists of two PMOS transistors in series, and the pull-down network consists of two NMOS transistors in parallel. Inputs A and B each drive one PMOS and one NMOS device.




Width sizing for NOR2x1, NOR2X2, NOR2X3:
The pull-up uses two PMOS in series, so each PMOS is sized to 360 nm (180 nm x 2) to match the inverter strength. The pull-down has two NMOS in parallel, so each NMOS uses the inverter NMOS width (180 nm).

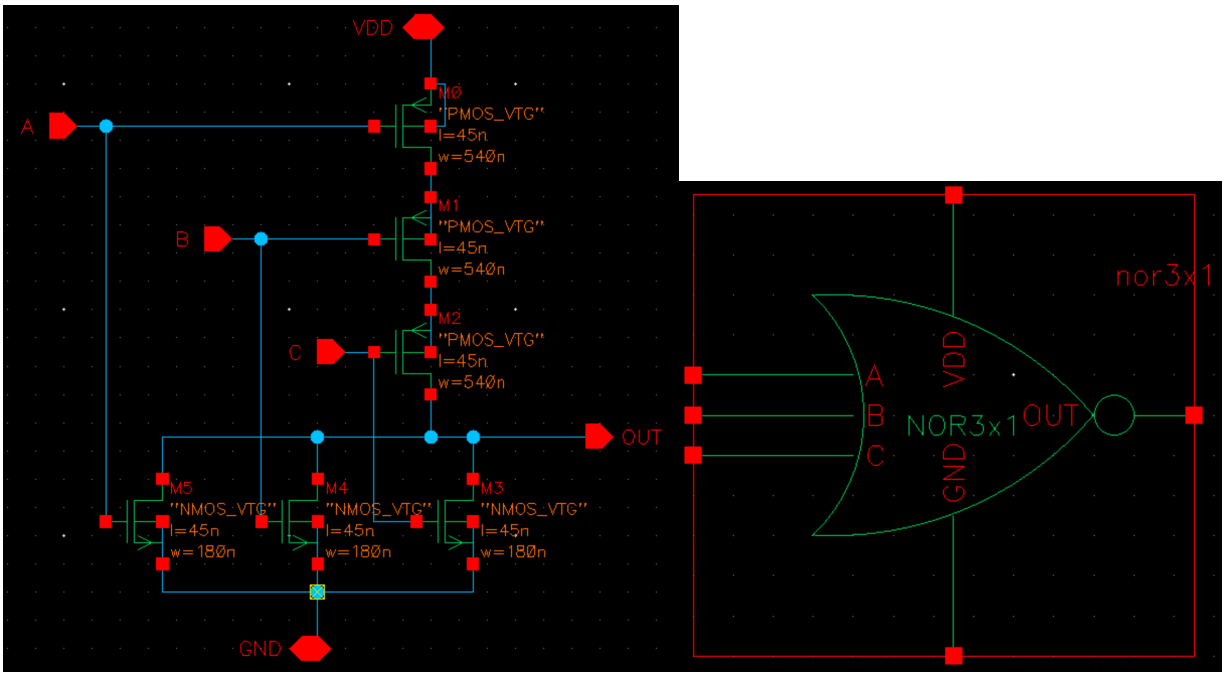
Drive strength iterations all scale widths uniformly:

- NOR2x1: base widths = PMOS 360 nm / NMOS 180 nm
- NOR2x2: widths scaled by 2 = PMOS nm 720 nm / NMOS 360 nm
- NOR2x3: widths scaled by 3 = PMOS 1080 nm / NMOS 540 nm




## 3-input NOR

The 3-input NOR gate was implemented using static complementary CMOS logic. The pull-up network consists of three PMOS transistors in series, and the pull-down network consists of three NMOS transistors in parallel. Inputs A, B, and C each drive one PMOS and one NMOS device.

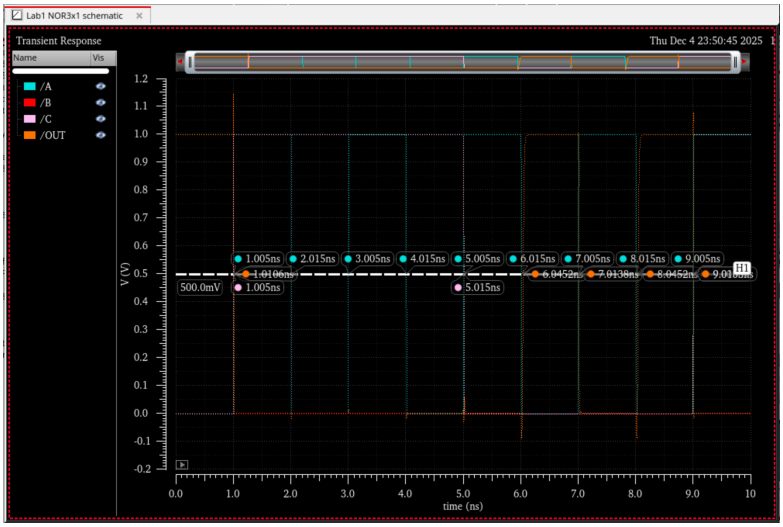



Width sizing for NOR3X1, NOR3X2, NOR3X3:

The pull-up uses three PMOS in series, so each PMOS is sized to 540 nm (180 nm x 3). The pull-down has three NMOS in parallel, so each NMOS uses the inverter NMOS width (180 nm).

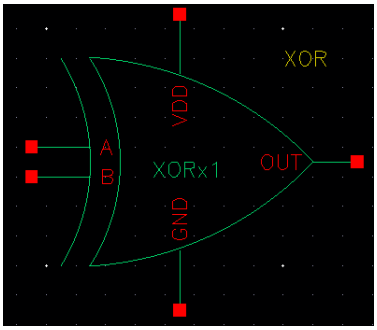
Drive strength iterations all scale widths uniformly:

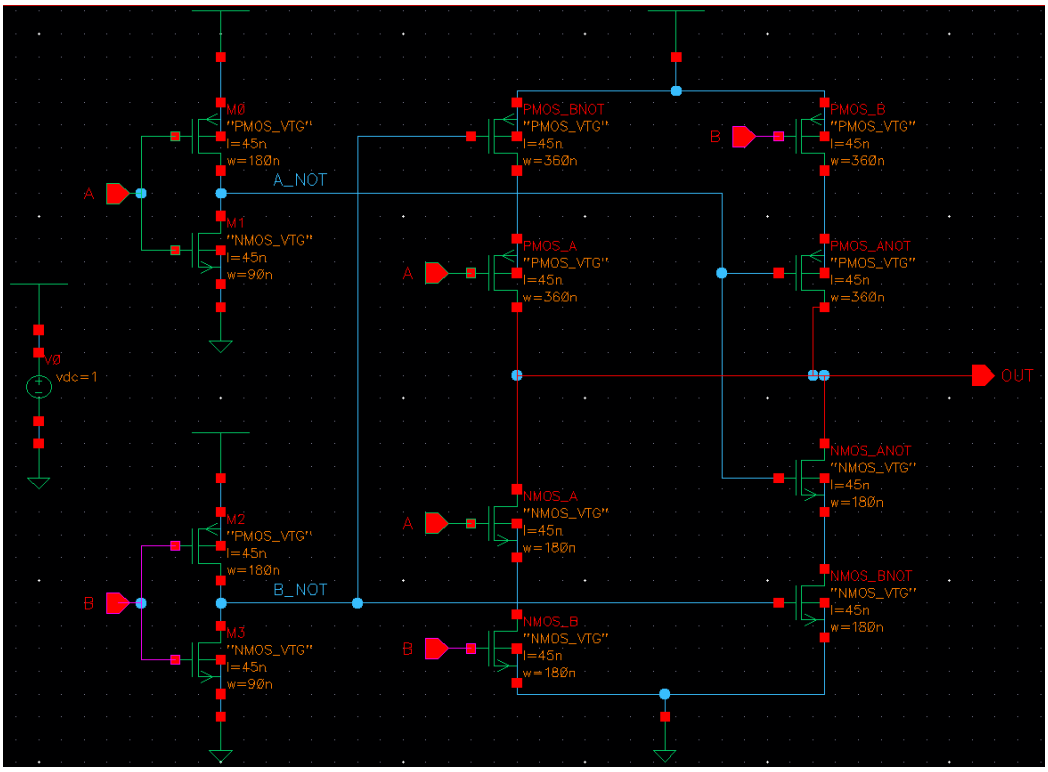
- NOR3x1: base widths = PMOS 540 nm / NMOS 180 nm
- NOR3x2: widths scaled by 2 = PMOS 1080 nm / NMOS 360 nm
- NOR3x3: widths scaled by 3 = PMOS 1620 nm / PMOS 540 nm




## XOR

The XOR gate was implemented with a pair of PMOS transistors put in series, in parallel with another pair of transistors. The same was done for the pull-down network with NMOS transistors. For the pull-up network, one of the transistors is driven by input A, and the transistor it's in series with is driven by B', which is the inverted signal of the input B. The other pair of pull-up transistors is the same, but with input A inverted instead. The pull-down network has a series pair of transistors driven by inputs B and A, respectively, while the other pair is driven by the inverted signals, A' and B'.



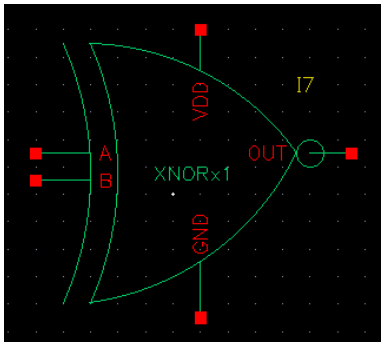



Sizing for XORx1, x2, x3:
Because the pull-down and pull-up networks are two pairs of transistors in series, the sizing for each transistor is twice that of the reference inverter. The PMOS transistors are 360nm (180nm x 2), and the size of the NMOS transistors is 180nm (90nm x 2).

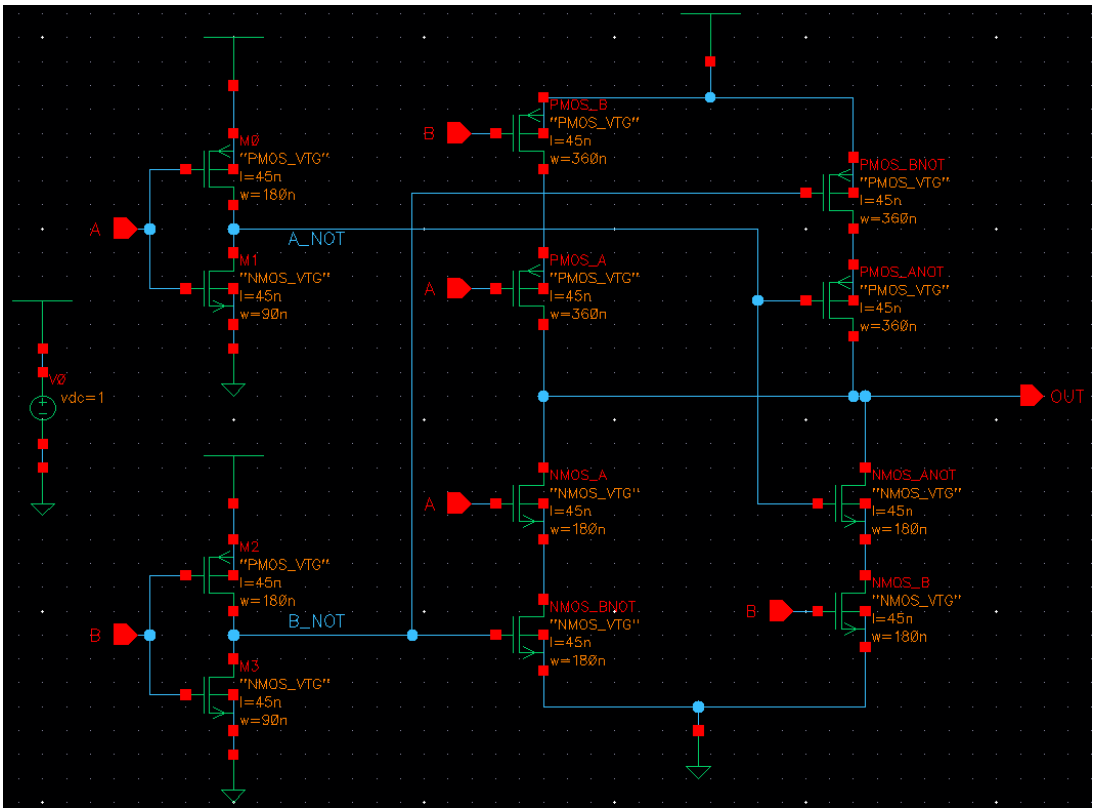
- XORx1: PMOS width = 360nm / NMOS width = 180nm
- XORx2: PMOS width = 720nm / NMOS width = 360nm
- XORx3: PMOS width = 1080nm / NMOS width = 540nm

## XNOR

The XNOR gate is implemented similarly to the XOR gate, but with the pull-up network swapped with the pull-down network



.

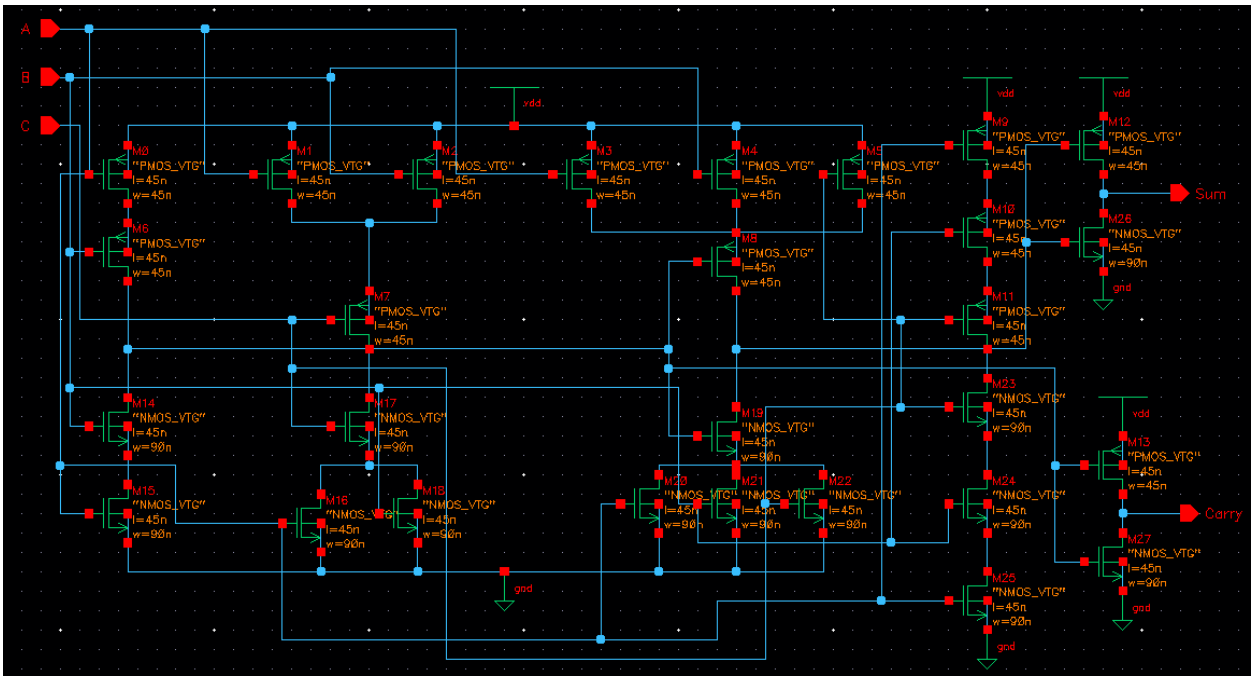



Sizing for XNORx1, x2, x3:
Because the pull-down and pull-up networks are two pairs of transistors in series, the sizing for each transistor is twice that of the reference inverter. The PMOS transistors are 360nm (180nm x 2), and the size of the NMOS transistors is 180nm (90nm x 2).

- XNORx1: PMOS width = 360nm / NMOS width = 180nm
- XNORx2: PMOS width = 720nm / NMOS width = 360nm
- XNORx3: PMOS width = 1080nm / NMOS width = 540nm

## 1-bit adder cell

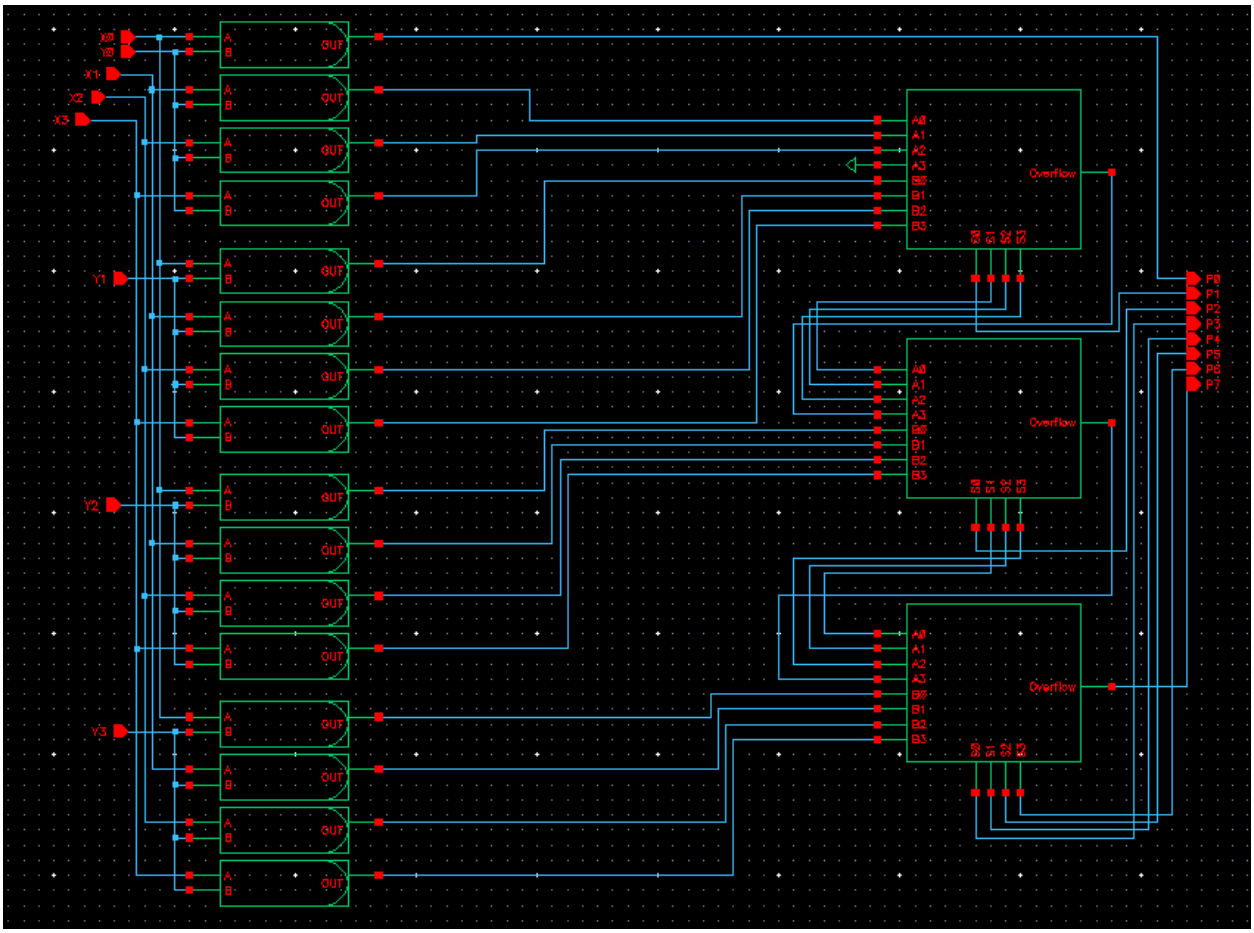
To create the full bit adder, two XORs, two ANDs, and an OR gate were used. Translating to building the logic out of NMOS, the figure below depicts it.




The x1, x2, and x3 versions of each gate scale the transistor widths by 1, 2, or 3. Increasing device width reduces effective resistance, lowering the propagation delay. This is why higher drive (x2, x3) usually shows slightly faster rise/fall times compared to x1.

## Multiplier cell

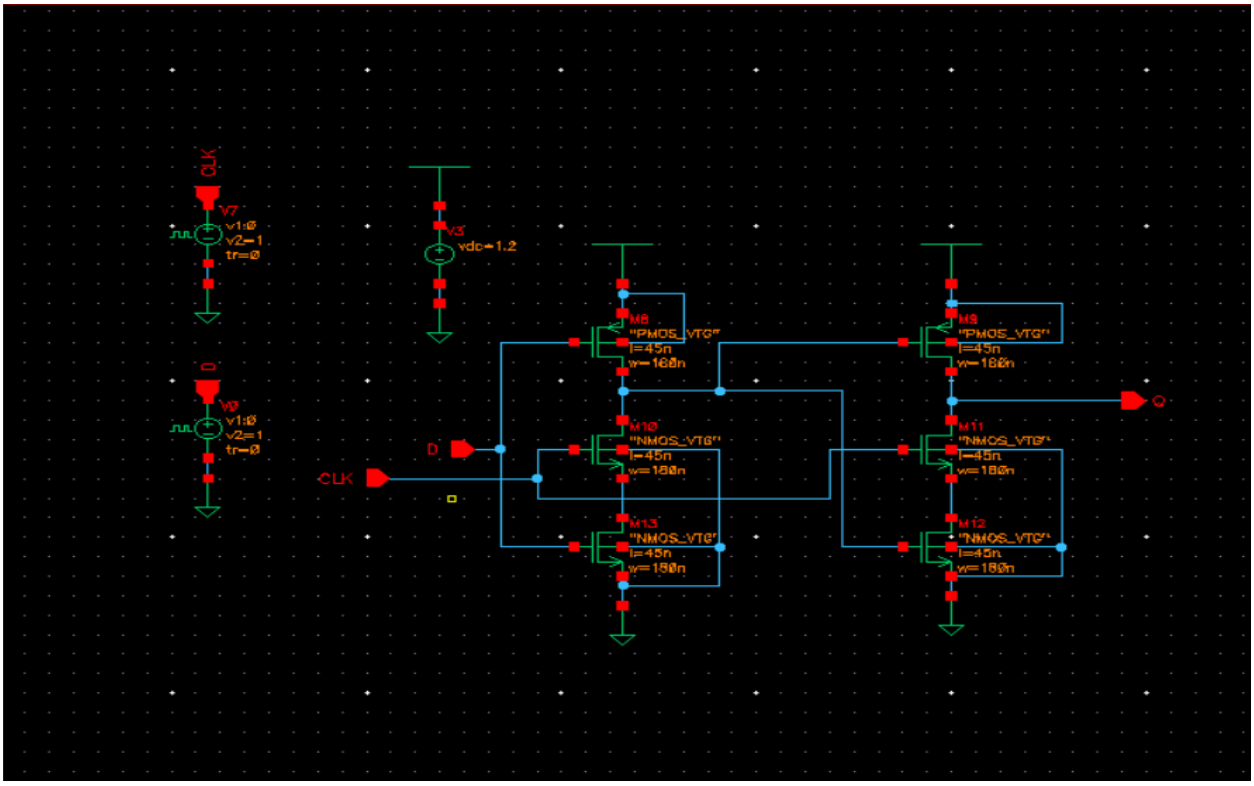
The 4-bit multiplier was built using three 4-bit adders, tied together with multiple AND gates used for most inputs.




In the overview A3 of the first 4-bit adder is tied to ground, as it acts like the Carry-In of the whole system, since it operates independently, and there wouldn't be any value for it, so it's tied to ground.

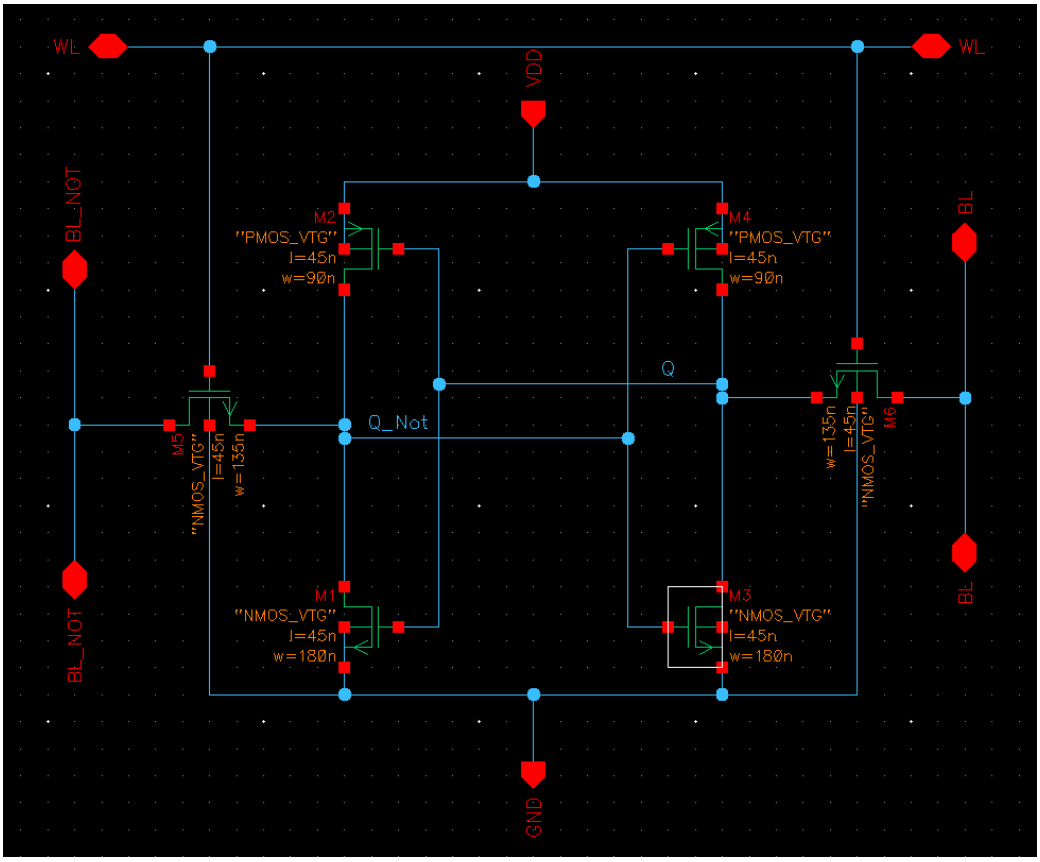
# F/F

A simple master servant design like the one demonstrated in class.



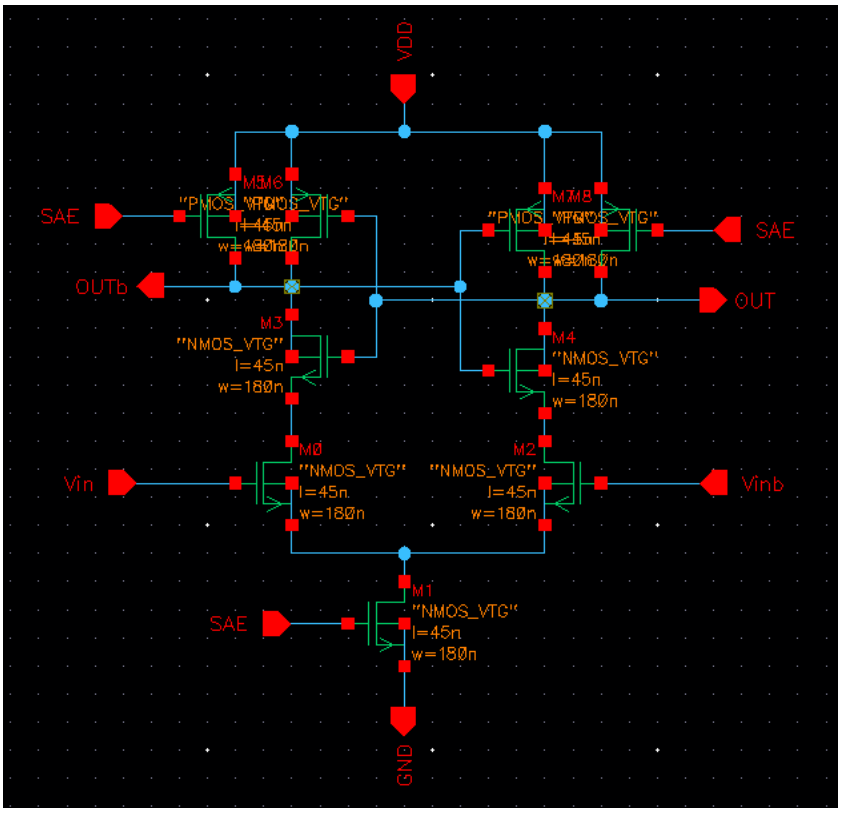
## SRAM cell

The SRAM cell is two Inverters leading into each other, with access NMOS connected to BL and BL`, controlled by WL.



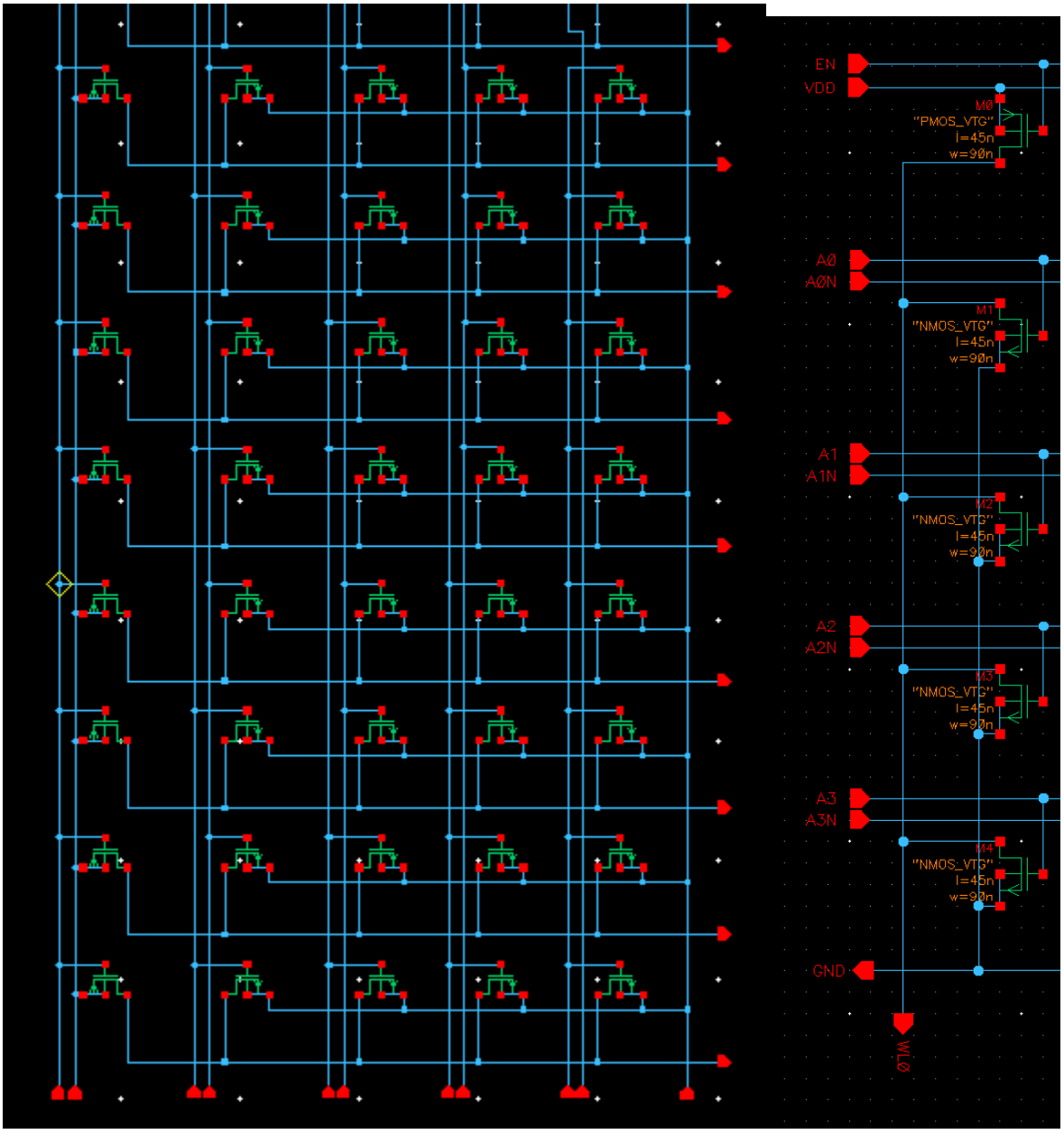

## Sense Amplifier

The Sense Amplifier is two NAND gates, with one input being BL and BL`, and the other input being the output of the other NAND gate. OUT is high and OUTb is low if BL (Vinb) is a higher voltage than BL` (Vin).




## Decoder

For proof-of-concept due to time constraints, the decoder we designed is a 16x 4-input NOR decoder. In a 32x decoder, this would be expanded to 5 inputs. There are extra inputs in the design for Vdd and Enable.




# IV. System Characterization

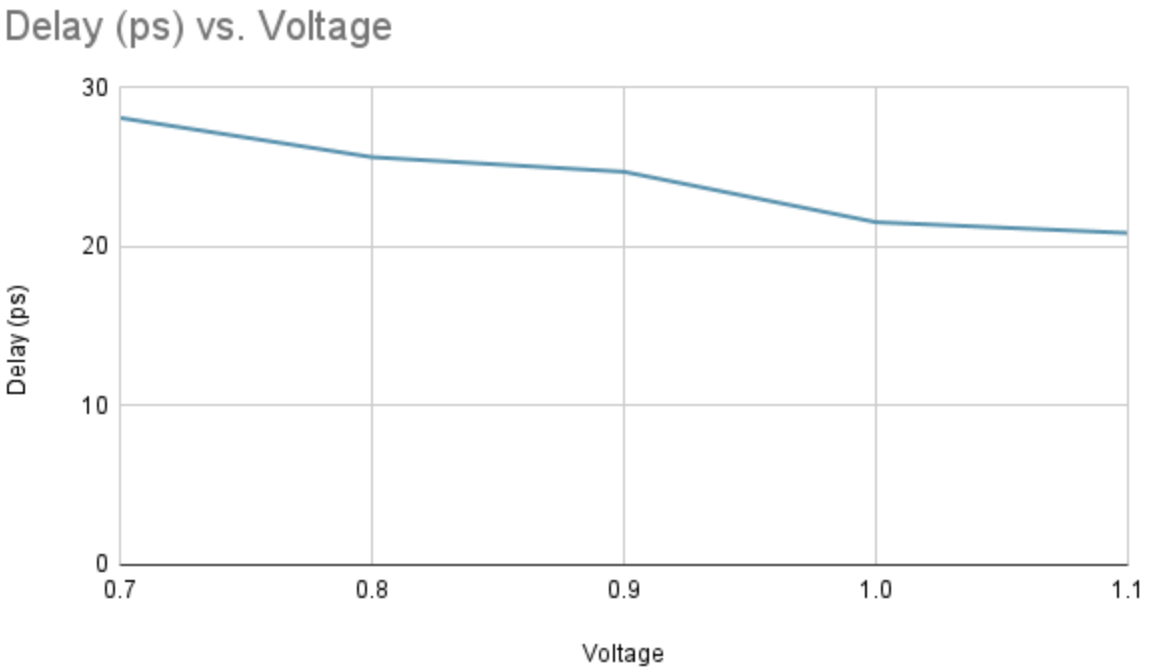
We unfortunately did not have enough time to properly put together each component into a working system level design and test it. So the most we can provide in this final report is the component level characteristics.

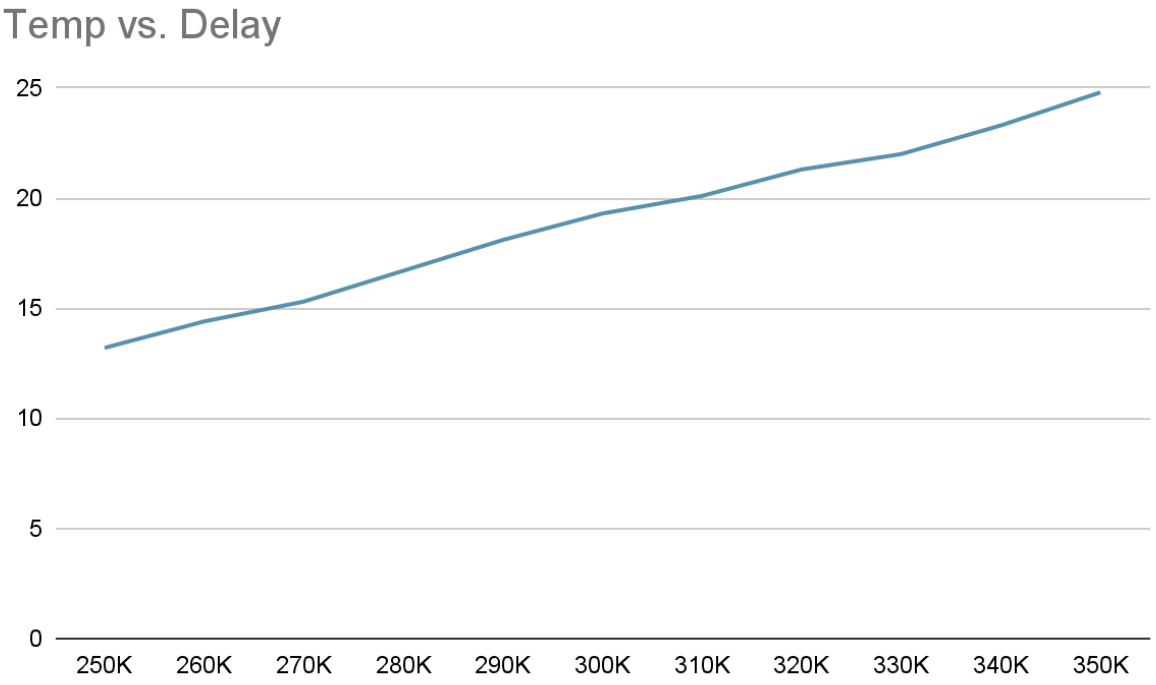
## 8-bit adder

The following table depicts the tests done to check how the 8 bit adder performs and if it works correctly. The following was measured by using stimuli, setting each input to 1.1 V but turning them off or on depending on the input needed.

| Tests | Expected | Actual |
|---|---|---|
| A=10000000<br>B=10000000 | 00000000<br>Overflow = 1 | 00000000<br>Overflow = 1<br>Figure SC(A) |
| A=00000001<br>B=00000001 | 00000010<br>Overflow = 0 | 00000010<br>Overflow = 0<br>Figure SC(B) |
| A=10000000<br>B=00000001 | 10000001<br>Overflow = 0 | 10000001<br>Overflow = 0<br>Figure SC(C) |
| A=00011111<br>B=00000001 | 00100000<br>Overflow = 0 | 00100000<br>Overflow = 0<br>Figure SC(D) |
| A=10101010<br>B=01010101 | 11111111<br>Overflow = 0 | 11111111<br>Overflow = 0<br>Figure SC(E) |

```
tran: time = 9.82 ns     (49.1 %), step = 400 ps          (2 %)
tran: time = 10.62 ns    (53.1 %), step = 400 ps          (2 %)
tran: time = 11.82 ns    (59.1 %), step = 400 ps          (2 %)
tran: time = 12.62 ns    (63.1 %), step = 400 ps          (2 %)
tran: time = 13.82 ns    (69.1 %), step = 400 ps          (2 %)
tran: time = 14.62 ns    (73.1 %), step = 400 ps          (2 %)
tran: time = 15.82 ns    (79.1 %), step = 400 ps          (2 %)
tran: time = 16.62 ns    (83.1 %), step = 400 ps          (2 %)
tran: time = 17.82 ns    (89.1 %), step = 400 ps          (2 %)
tran: time = 18.62 ns    (93.1 %), step = 400 ps          (2 %)
tran: time = 19.71 ns    (98.6 %), step = 290 ps       (1.45 %)
```



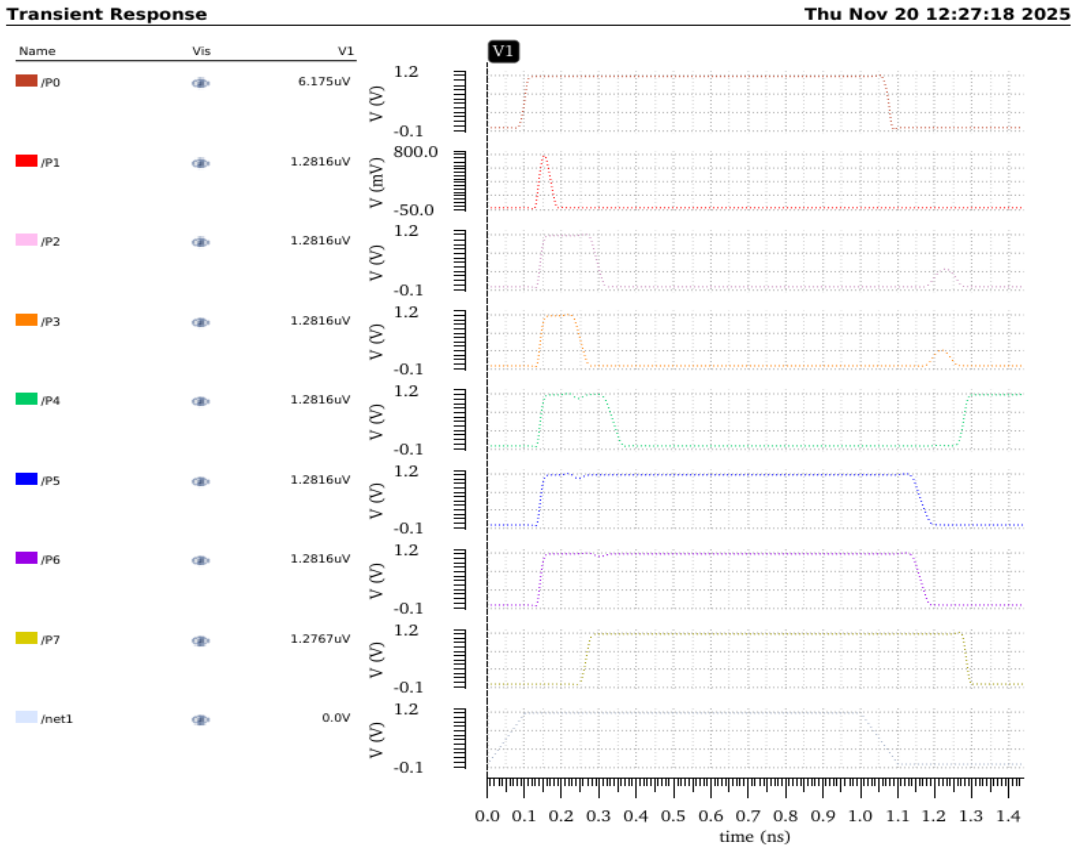




## 4-bit multiplier

The worst-case scenario for delay can be either of the following. A = 1111 B = 1111, A = 0001 B=1111 , or A = 1111 B = 0001. The primary source of delay comes from A3/B3 value. The delay being **243.39 ps**. The following are tests of logic.

| Test | Expected | Actual |
|---|---|---|
| 1111 x 1111 | 11100001 | 11100001 |
| 0000 x 0000 | 00000000 | 00000000 |
| 0101 x 0101 | 00011001 | 00011001 |
| 1100 x 0011 | 00100100 | 00100100 |
| 0110 x 0110 | 00100100 | 00100100 |

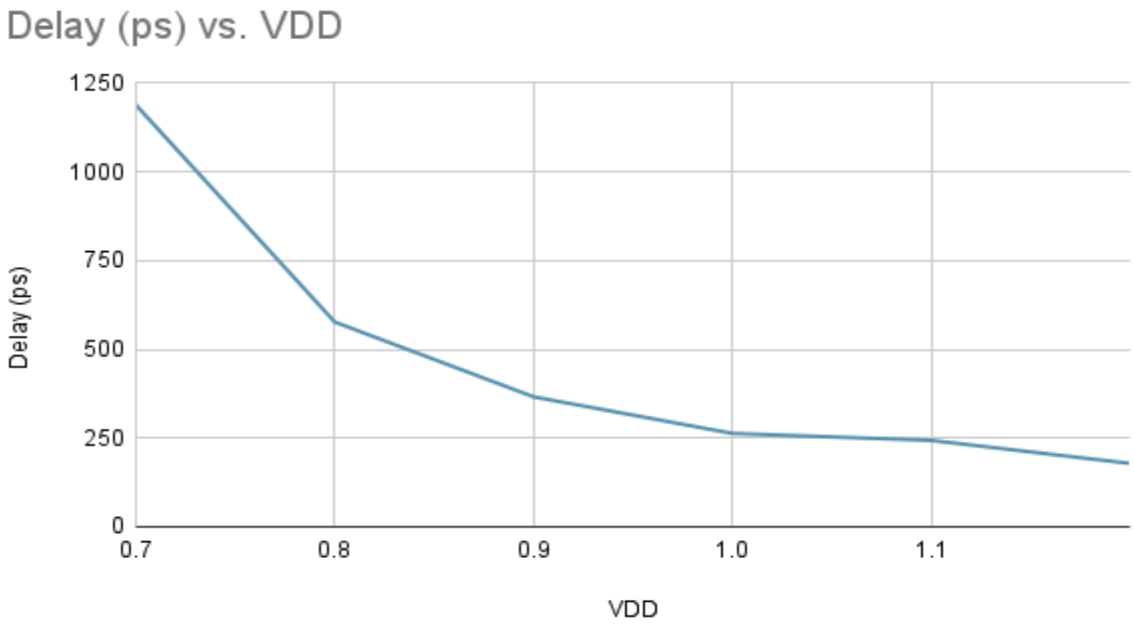
Worst-case input signals and the operational transients of different significance bits of the multiplier



Power and delay of the worst-case at nominal VDD

| VDD | Delay |
|---|---|
| 0.7 | *1.188 ns |
| 0.8 | 576.79 ps |
| 0.9 | 366.13 ps |
| 1.0 | 263.02 ps |
| 1.1 | 243.39 ps |
| 1.2 | **178.54 ps** |

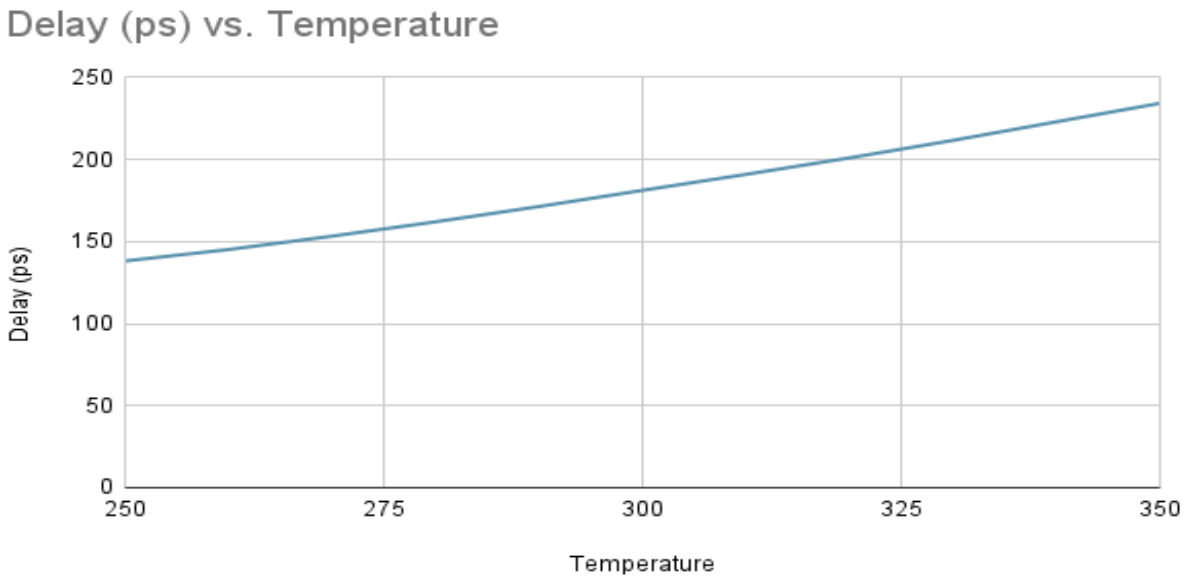
Worst-case delay at varying VDD (0.7V – 1.2V) and at room temperature (300 K)




Worst-case delay at varying temperature (250 K – 350 K) and at nominal VDD

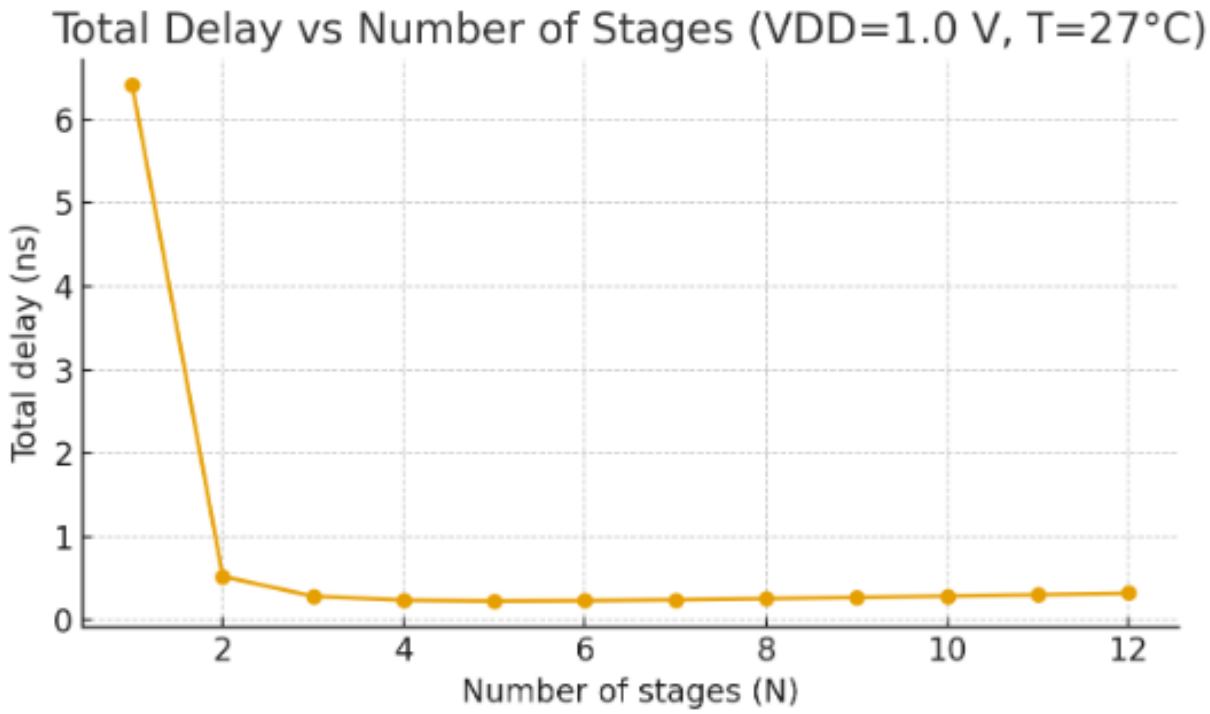
| Temperature | Delay |
|---|---|
| 250 | 138.151 ps |
| 260 | 145.1476 ps |

| | |
|---|---|
| 270 | 153.2742 ps |
| 280 | 162.03 ps |
| 290 | 171.339 ps |
| 300 | 181.2 ps |
| 310 | 190.95 ps |
| 320 | 200.98 ps |
| 330 | 211.65 ps |
| 340 | 222.967 ps |
| 350 | 234.231 ps |

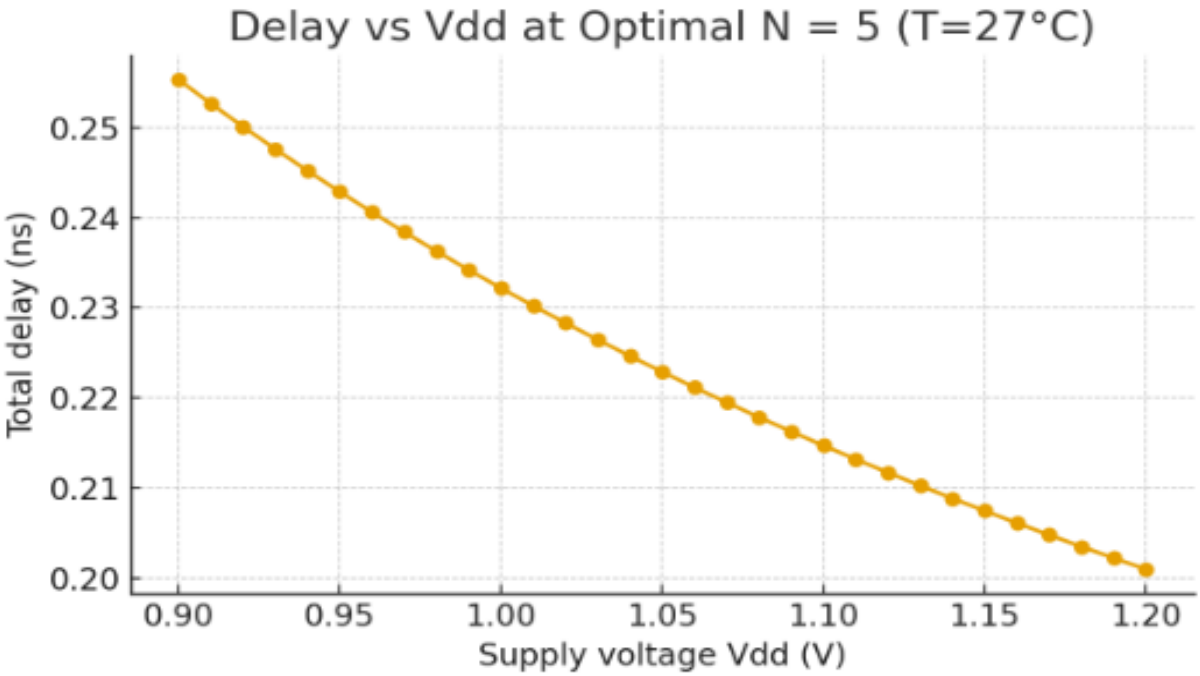



## Inverter Chain

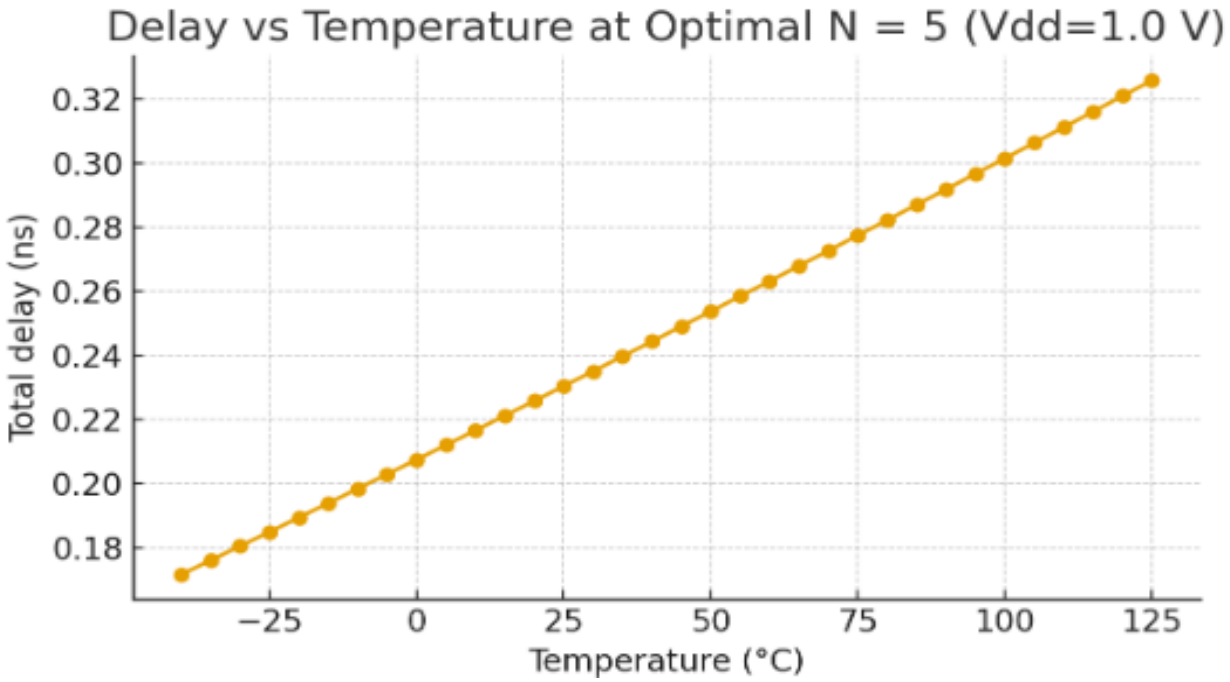
Delay vs. number of stages (N)




Delay vs. supply voltage




Delay vs. temperature




## Gate Library

Worst case rise and fall delay of each gate at VDD = 1 V and room temperature (300K)

| Gate | Drive | tpLH (ps) | tpHL (ps) | Worst-case |
|---|---|---|---|---|
| NAND2X1 | x1 | 9.9 | 7.5 | 9.9 ps |
| NAND2X2 | x2 | 9.84 | 7.45 | 9.84 ps |
| NAND2X3 | x3 | 9.81 | 7.45 | 9.81 ps |
| NAND3X1 | x1 | 13.16 | 8.13 | 13.16 ps |
| NAND3X2 | x2 | 13.0 | 8.2 | 13.0 ps |
| NAND3X3 | x3 | 12.95 | 8.16 | 12.95 ps |
| NOR2X1 | x1 | 18.11 | 6.5 | 18.11 ps |
| NOR2X2 | x2 | 18.1 | 6.46 | 18.1 ps |
| NOR2X3 | x3 | 18.1 | 6.44 | 18.1 ps |
| NOR3X1 | x1 | 30.51 | 8.9 | 30.51 ps |
| NOR3X2 | x2 | 30.56 | 8.78 | 30.56 ps |

| | | | | |
|---|---|---|---|---|
| NOR3X3 | x3 | 30.6 | 8.76 | 30.6 ps |

Worst case rise and fall delay of each XOR gate at VDD = 1V and room temperature (300K)

| | tpLH (ps) | tpHL (ps) |
|---|---|---|
| Inputs Before & After (AB) | 00 -> 01 | 10 -> 11 |
| XORx1 | 35.79 | 11.29 |
| XORx2 | 35.53 | 11.23 |
| XORx3 | 35.46 | 11.2 |

Worst case rise and fall delay of each XNOR gate at VDD = 1V and room temperature (300K)

| | tpLH (ps) | tpHL (ps) |
|---|---|---|
| Inputs Before & After (AB) | 10 -> 11 | 00 -> 01 |
| XNORx1 | 37.13 | 15.36 |
| XNORx2 | 36.85 | 15.3 |
| XNORx3 | 36.85 | 15.28 |

## F/F and Registers

Clock period = 1 ns (1 GHz). 15% = 150 ps.

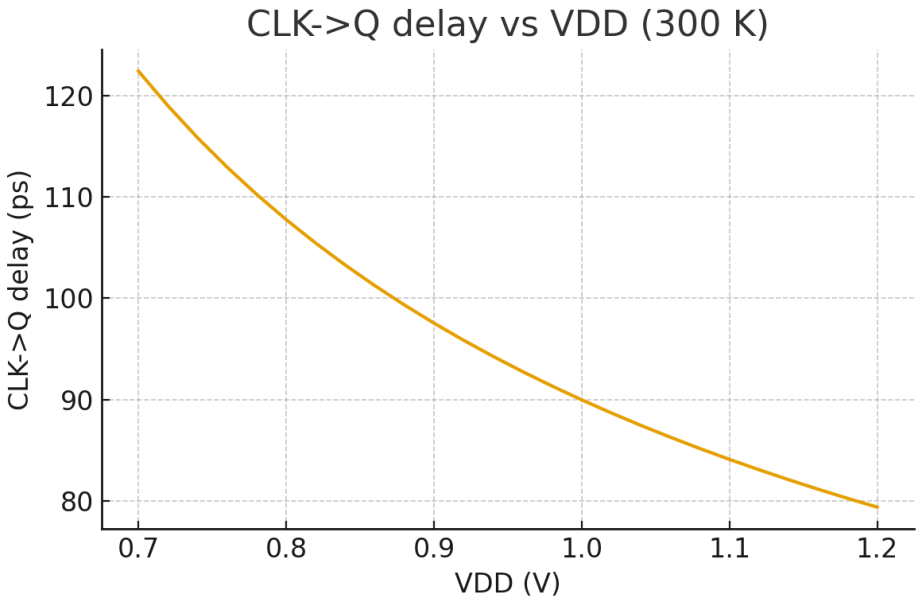
– Report setup time of F/F. It should be less than 15% of CLK period. Setup time was discussed in class lectures.

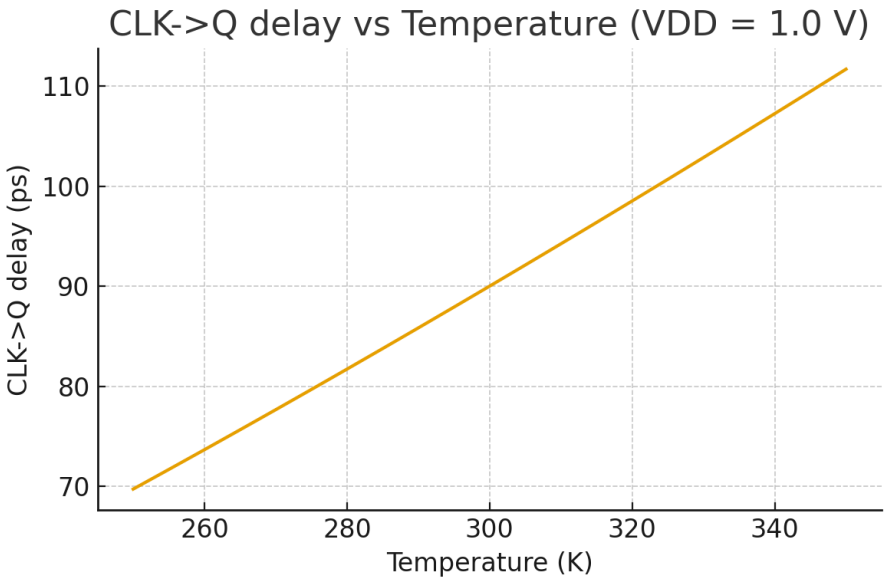
Setup time: 54.0 ps

– Report CLK-Q delay. It should be less than 15% of CLK period.

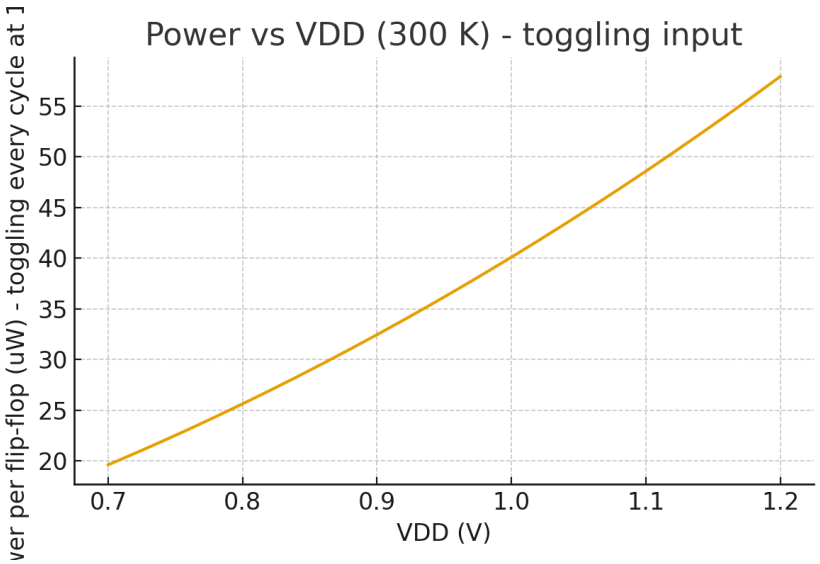
CLK→Q (1.0 V @ 300 K): 90.0 ps

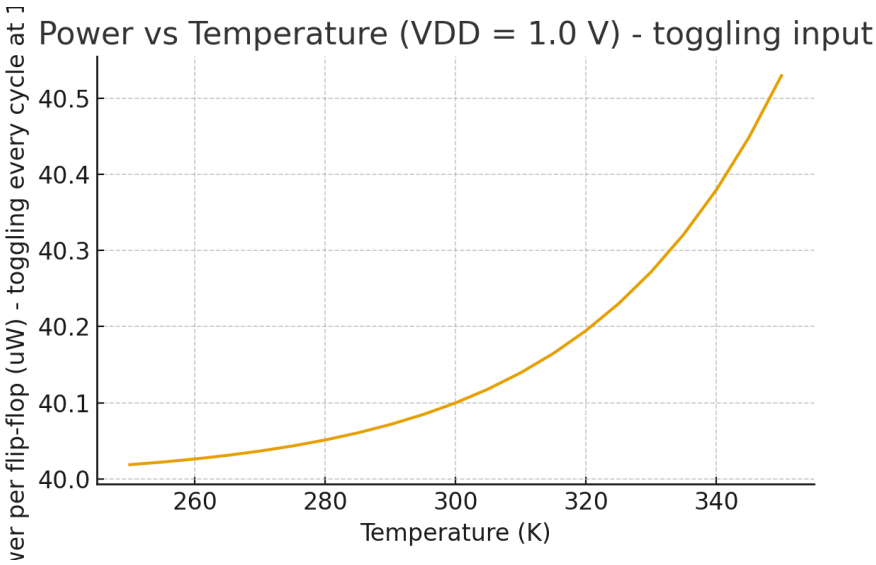
CLK-Q delay at varying VDD (0.7V – 1.2V) at room temperature, and varying temperature (250 K – 350 K) at nominal VDD






Apply a toggling input to F/F that changes from 0→1 →0 in every clock cycle. Plot power at varying VDD (0.7V – 1.2V) and room temperature (300 K). Plot power at varying temperature (250 K – 350 K) and nominal VDD.

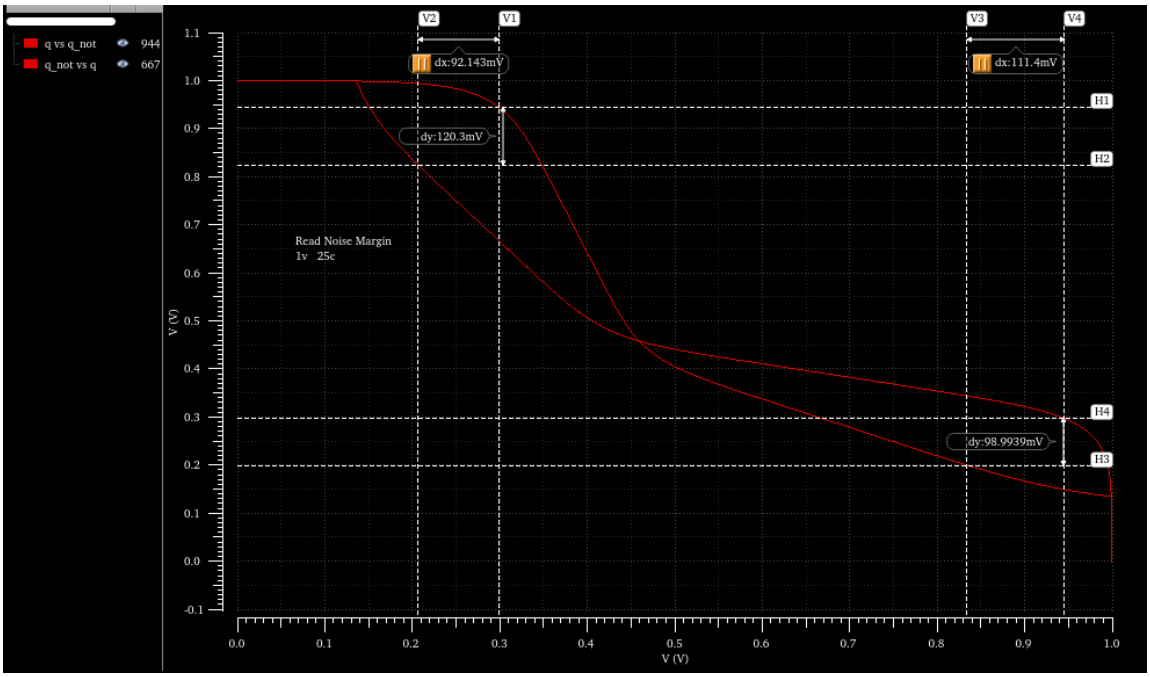


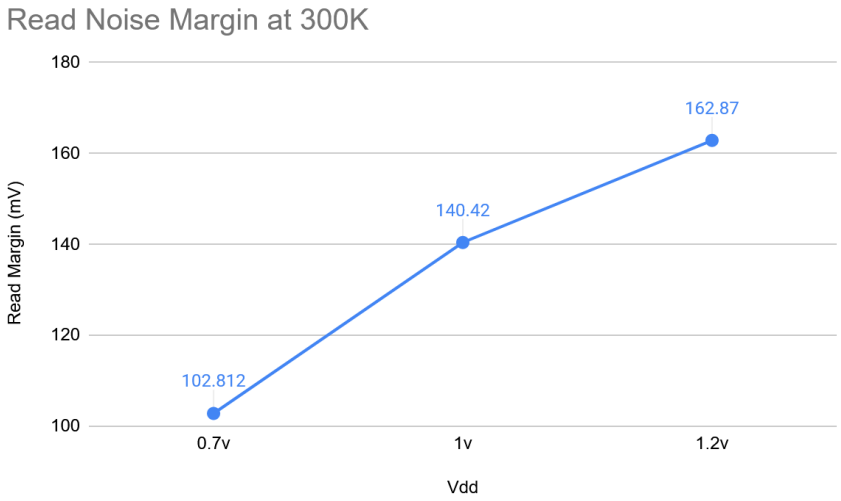


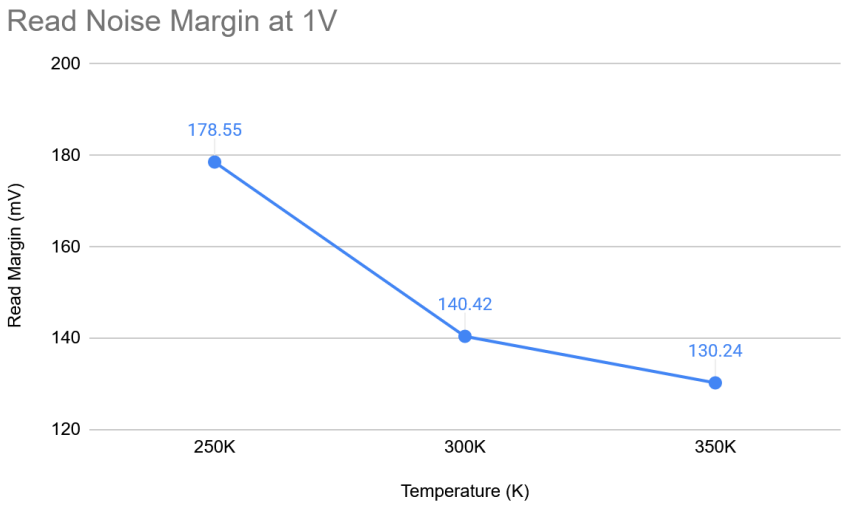

## SRAM cell

| | Transistor Width |
|---|---|
| Pull-Up | 90nm |
| Access | 135nm |
| Pull-Down | 180nm |

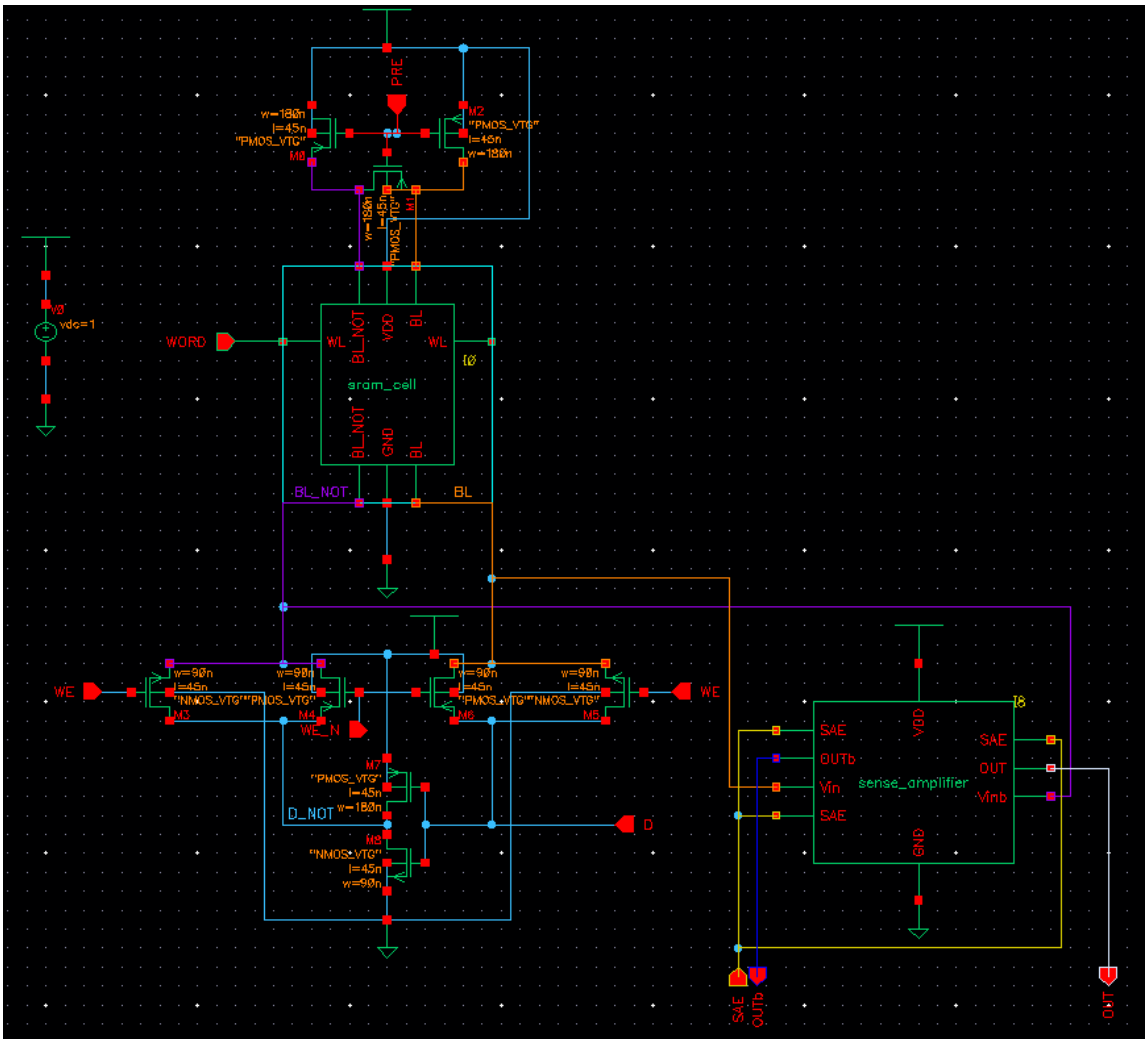
Read Noise Margin was tested by setting WL, BL, and BL` to Vdd, and graphing $V_Q$ against $V_{Q`}$ while applying a varying voltage of 0V-Vdd to $V_{Q`}$.


q vs q_not
944
q_not vs q
667
V2
V1
V3
V4
dx:92.143mV
dx:111.4mV
H1
H2
H4
H3
dy:120.3mV
dy:98.9939mV
Read Noise Margin
1v 25c
V (V)
V (V)

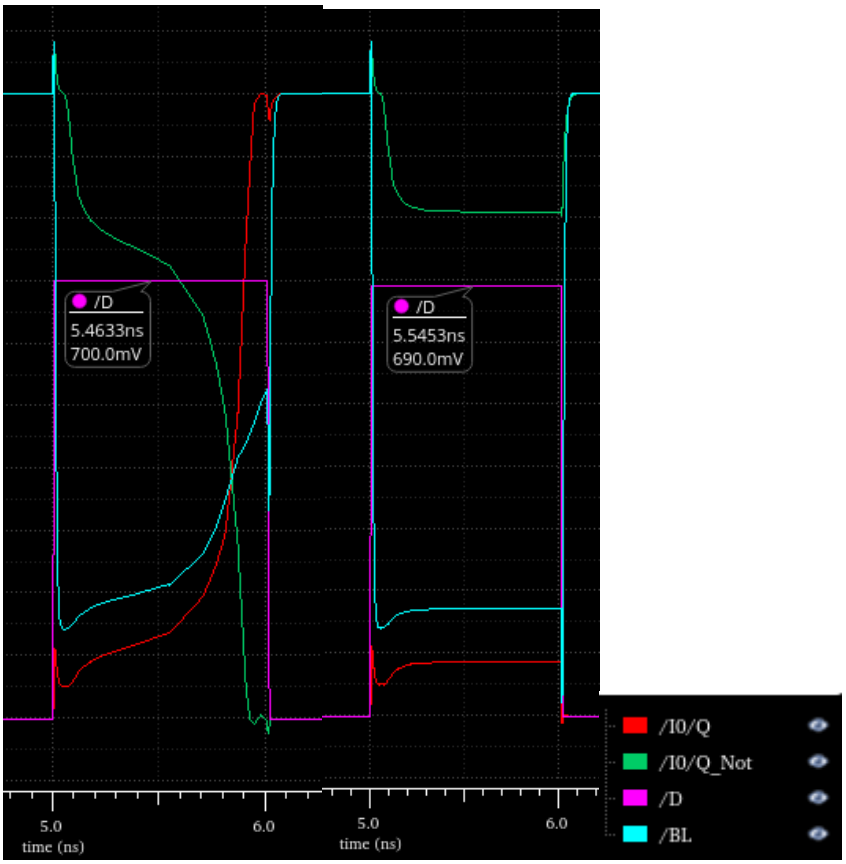

Read Noise Margin at 300K
180
160
140
120
100
Read Margin (mV)
162.87
140.42
102.812
0.7v
1v
1.2v
Vdd

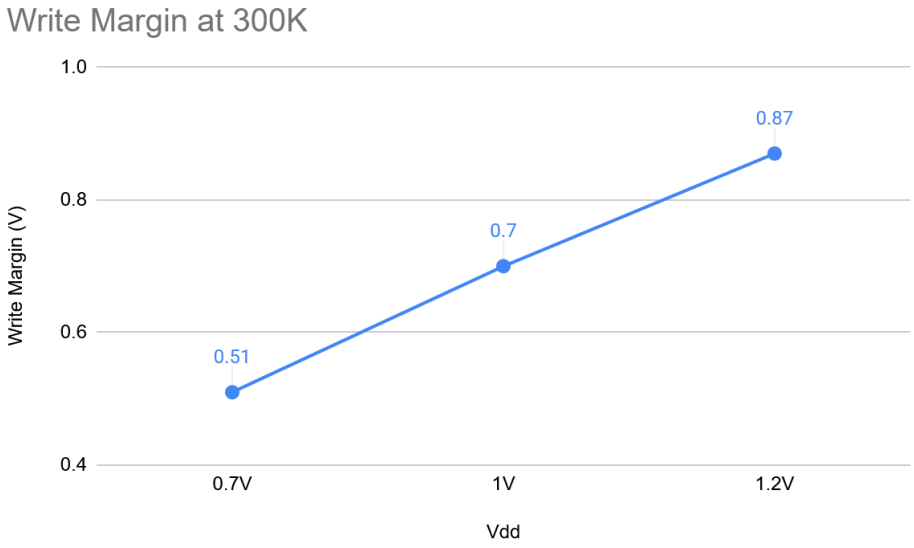


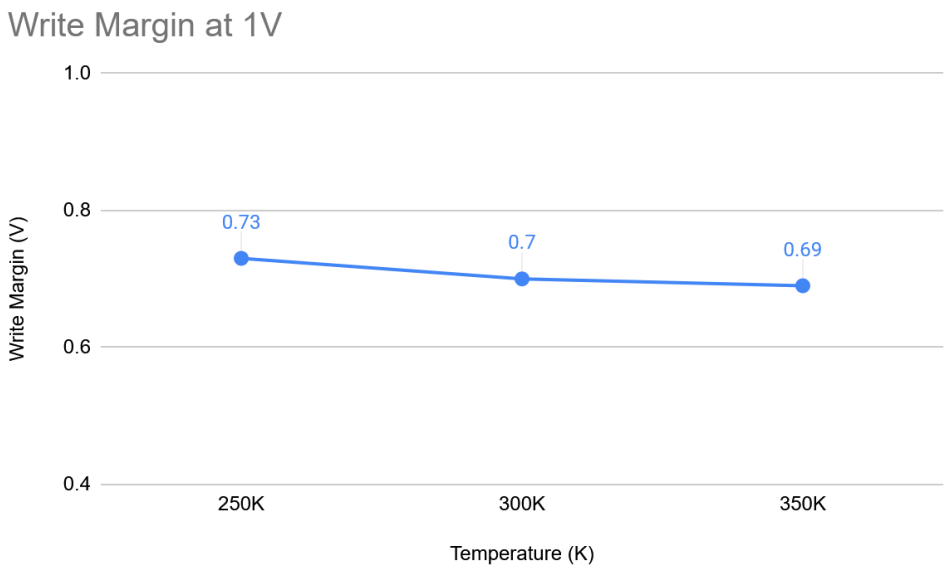

## SRAM Read/Write Tests



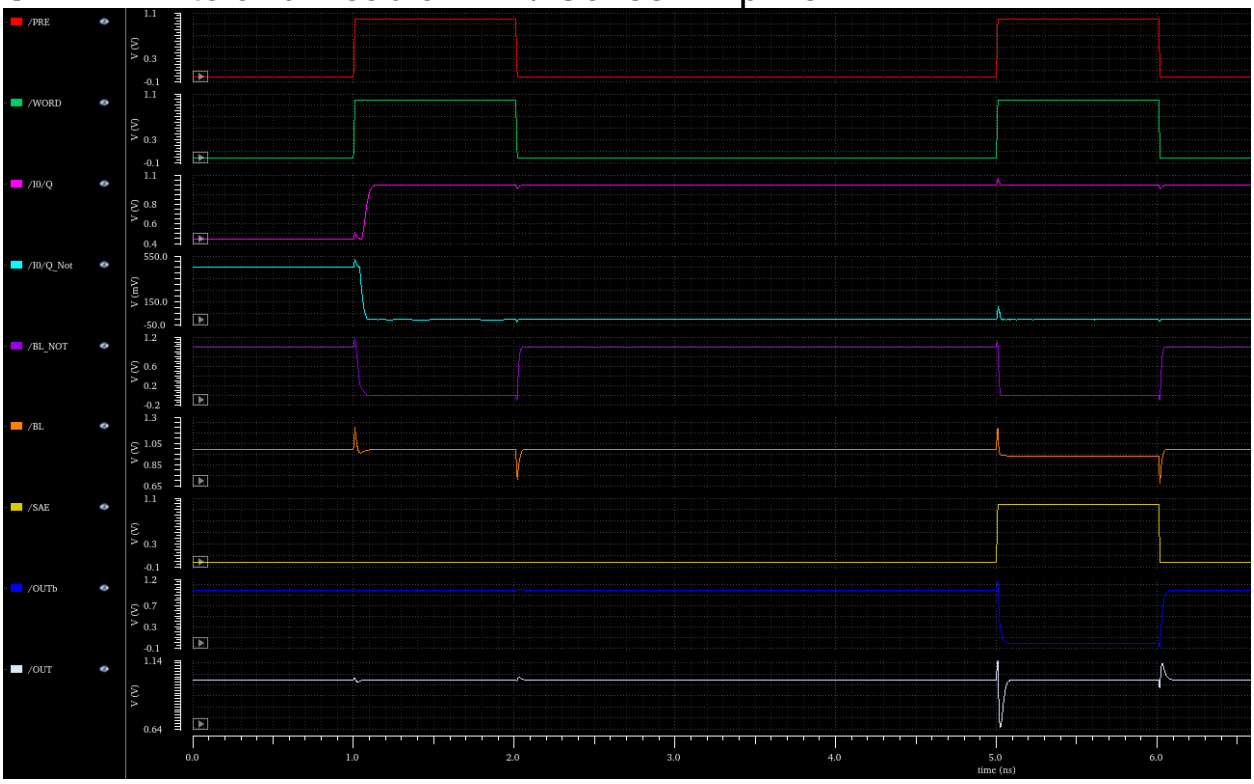

## SRAM Write Margin = 0.7v @ 1v, 300K


/D
5.4633ns
700.0mV
/D
5.5453ns
690.0mV
5.0
time (ns)
6.0
5.0
time (ns)
6.0
/I0/Q
/I0/Q_Not
/D
/BL


Write Margin at 300K
1.0
0.8
0.6
0.4
Write Margin (V)
0.87
0.7
0.51
0.7V
1V
1.2V
Vdd




SRAM Write and Read a ‘1’ w/ Sense Amplifier




SRAM Write and Read a ‘0’ w/ Sense Amplifier


Decoder Testing


# V. Task Organization

- Dalton Reynolds: Inverter Chain, F/F, Registers, report 1, presentation, final report
- Tessa Urwin: Cell library, report 1, presentation, final report

- Julian O'Hern: Cell library, SRAM cell, Sense Amplifier, Decoder, report 1, presentation, final report
- Lucas Bigos: Adder & Multiplier, report 1, presentation, final report

# VI. Conclusions

Through the design and simulation of the MAC datapath components in this project, we were able to apply what we've learned in ECE 467 about CMOS logic and circuit design in a hands-on way. We successfully designed and simulated components, such as inverters, logic gates, adders, registers, and SRAM cells.

We tested how transistor sizing and gate complexity impacted performance leading us to observe that increasing the drive strength (scaling the width) slightly reduced propagation delay, confirming that wider transistors improve switching speed.

We tested each component at different temperatures to find the optimal operating region for each. Proving that lower temperatures in general provided better performance across the board.

We were unable to finish the SRAM array due to time constraints, but we were able to design a smaller version of the Row decoder, and test a single SRAM cell in a column that would mimic an array column (the difference being the array column would have 32 cells instead of 1). Other differences include increased capacitance in the full array, and an extra input to make the x16 decoder a x32 decoder. Testing with the components we did make shows that the desired functionality is present, and expanding to a full array would be possible.

Although we didn't have time to fully integrate the complete MAC datapath, the individual blocks performed as expected. If we had more time, we would focus on earlier integration and timing verification across the full datapath. Overall, this project strengthened our understanding of transistor-level digital design, delay analysis, and how fundamental building blocks can form complex systems when integrated.