Introduction to VLSI Design
Assignment 3

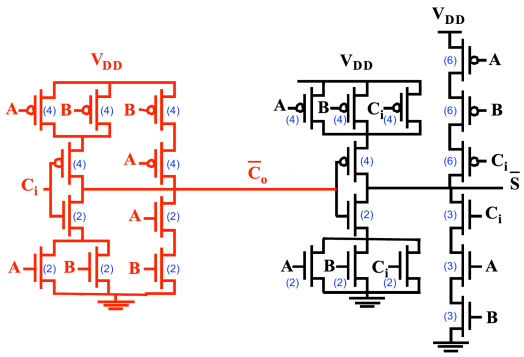
Name- Pushkal Mishra
Roll- EE20BTECH11042

Implementation of mirror adder

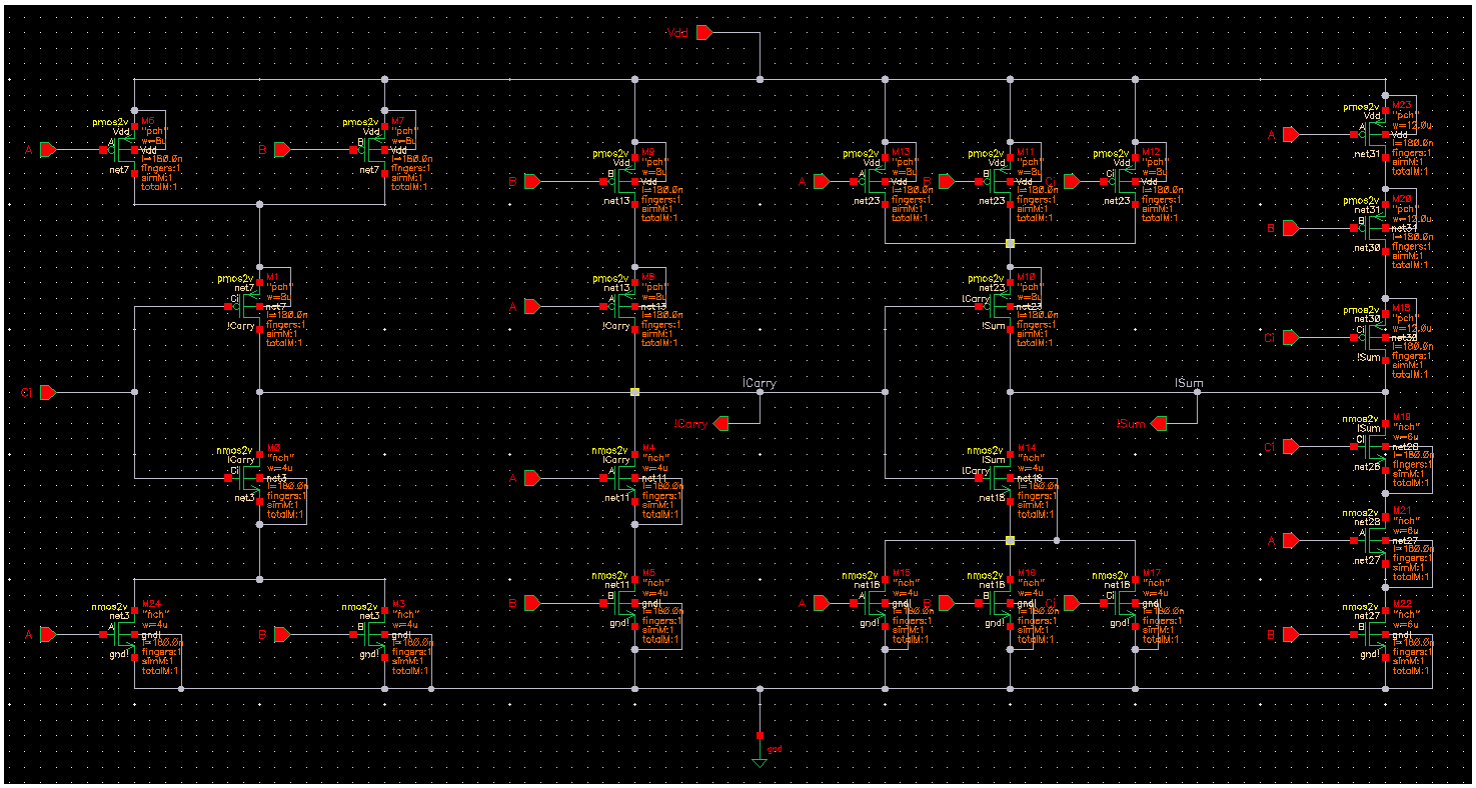
Adder is a digital circuit which performs addition of two binary numbers. We use PMOS and NMOS transistors with the pull up network having the same structure as the pull down network (hence the name mirror adder). Suppose that A and B are input bits and C_i is the carry bit, then the mirror adder logic for sum and carry output is as follows–

$$C_{out} = AB + BC_i + C_iA$$
$$S = ABC_i + !C_{out}(A + B + C_i)$$

To decide the scaling factor of transistors we can use the method prescribed in class which balances out the PUN and PDN worst case resistances to minimize the propagation delay. Sizing of transistors comes out as shown in the adjacent image.

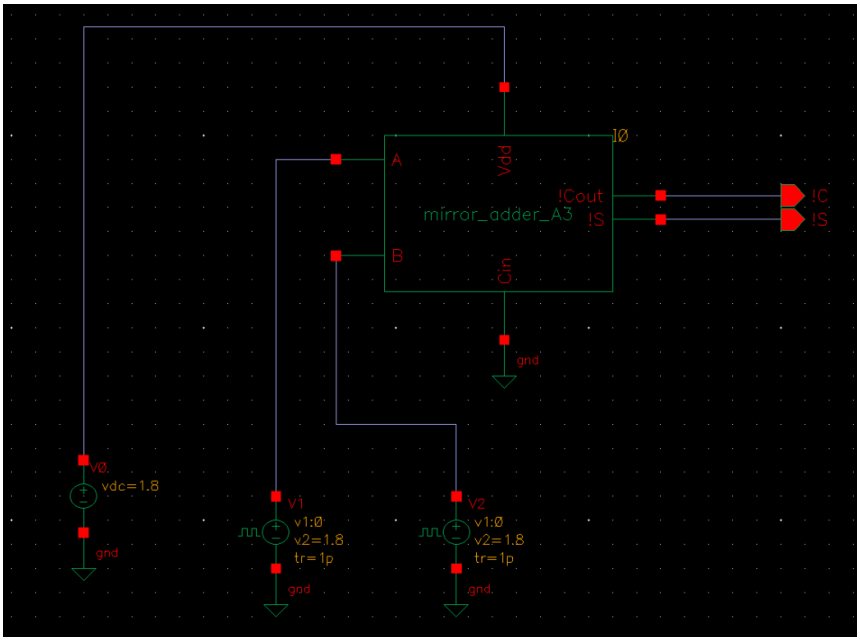


Schematic implementation of above circuit

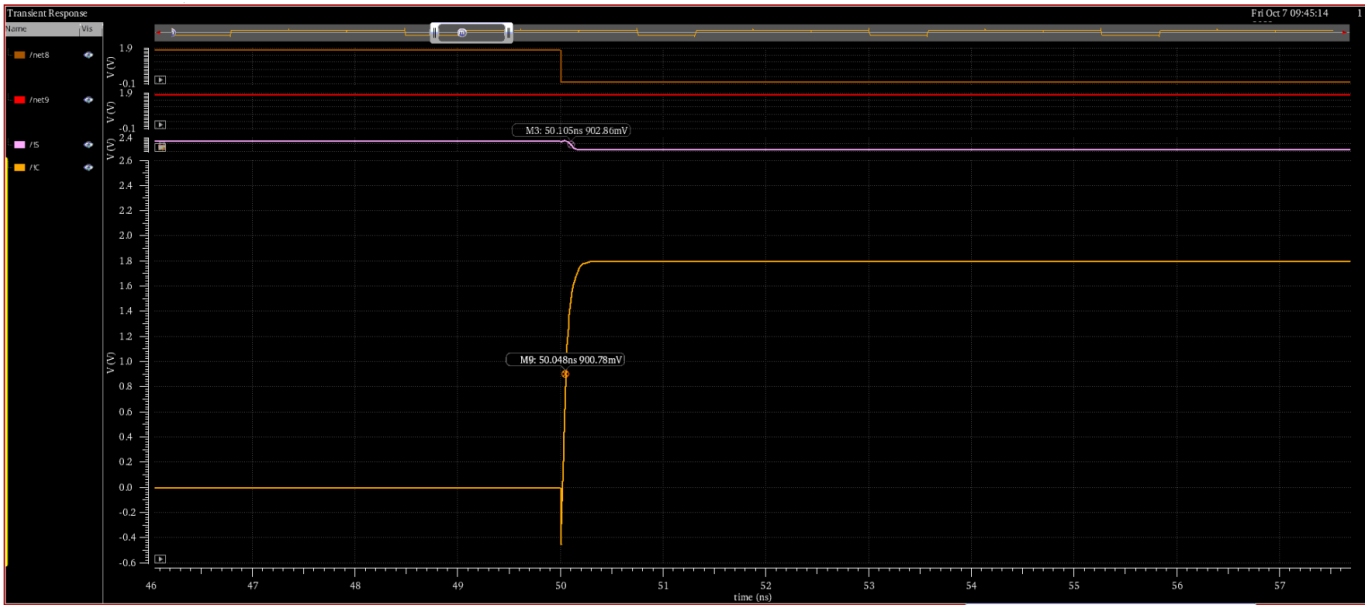


To calculate the delay in sum and carry, we can plot the outputs when there is a switch in state. The time delay in both sum and carry is the time difference of when sum (or) carry crosses 0.9V (V_{DD} / 2) and when the input crosses 0.9V.

Testbench for delay calculation

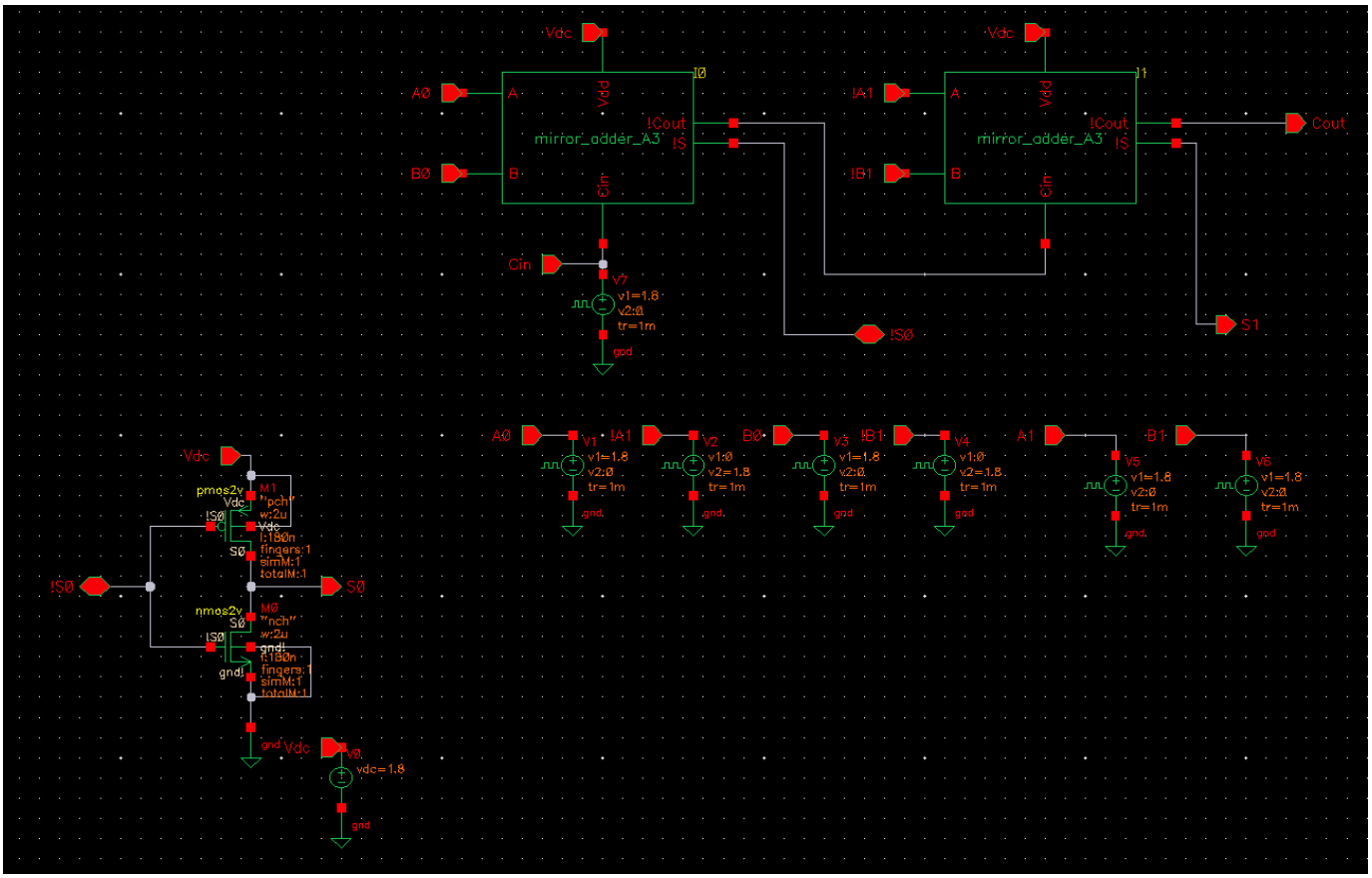


Plot with markers for 0.9V crossing

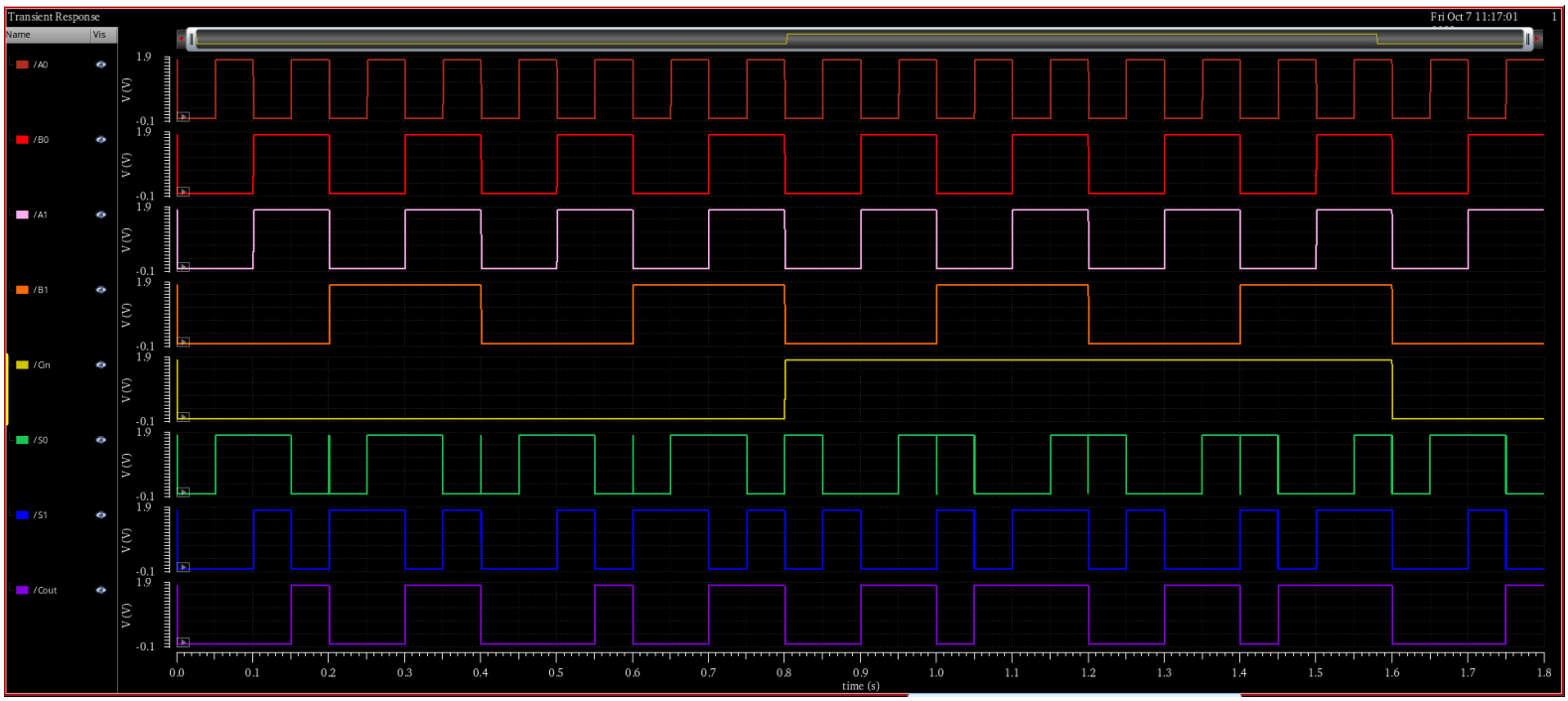


The 0.9V crossing for input is at 50ns. So delay in sum comes out to be 105ps and delay in carry comes out to be 48ps. In this implementation we receive inverted outputs from the circuit and so to make a 2 or higher bit adder we can simply feed the outputs to next stage inputs and invert alternate outputs (using inversion property of adders).

Testbench for 2 bit adder



Output of 2 bit ripple carry adder



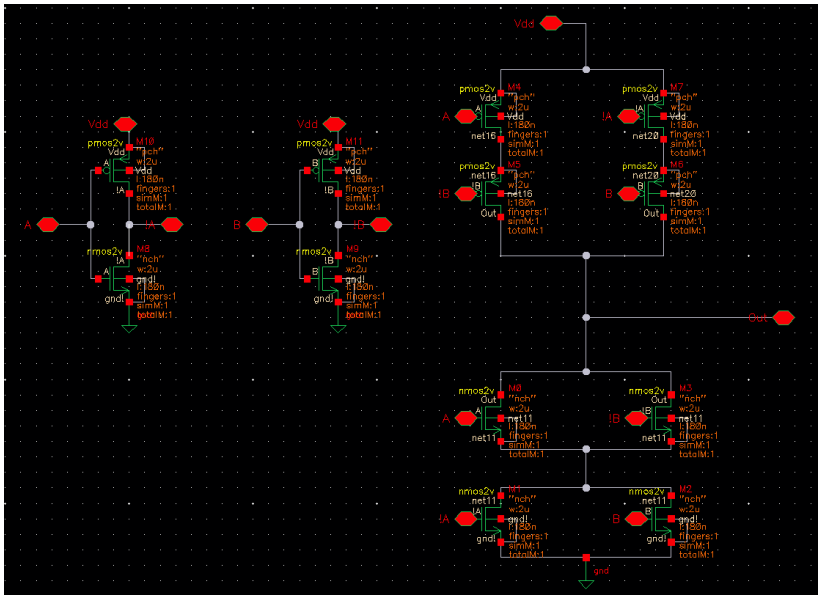
Modification of mirror adder to perform Add and Subtract using control signal

Here the inputs are in 2' s complement notation, so if we need to subtract B with A, then we can invert all binary digits in B and add 1 to the expression. Formally–

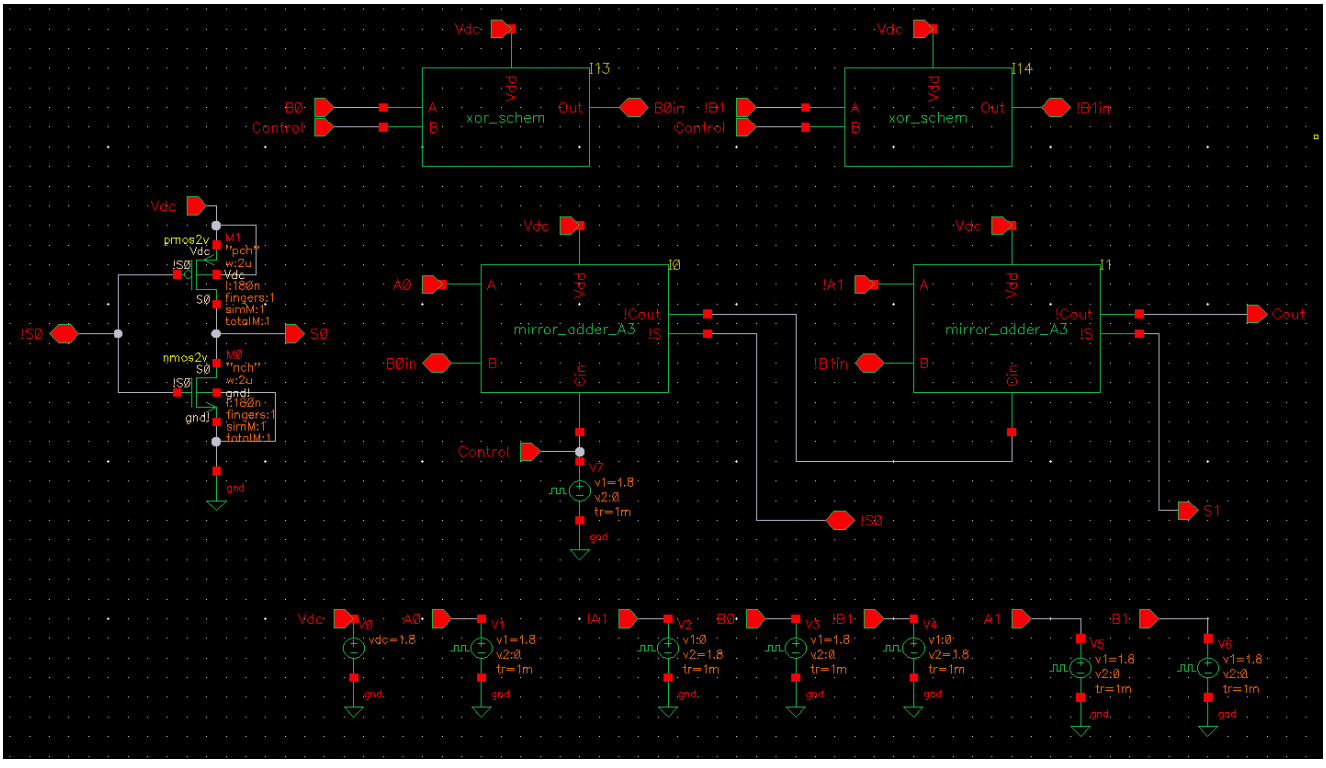
$$A_{2'} - B_{2'} = A_{2'} + \left(- B_{2'}\right) = A_{2'} +! B_{1'} + 1$$

For inverting the bits, we can use XOR because when one input of it is 0 the output is the other input value and when one input is 1 the output is the complement of the other input. Essentially it acts as a buffer when one input is 0 and inverter when one input is 1.

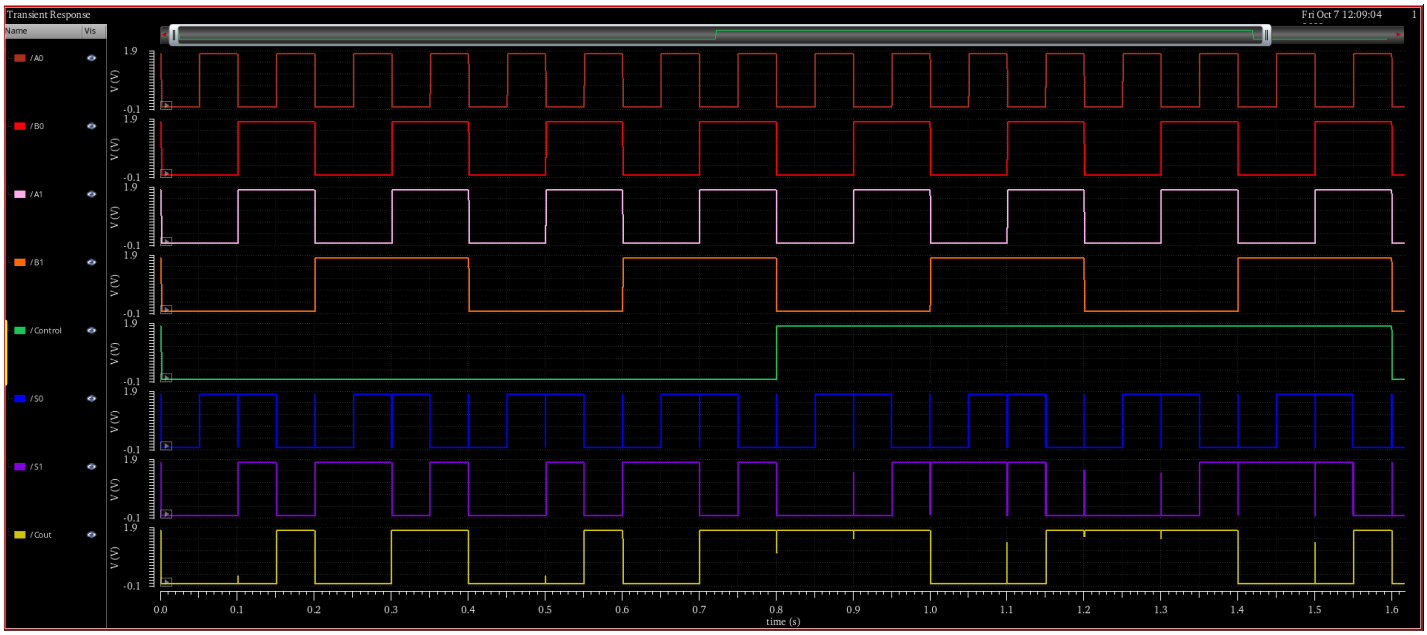
Implementation of XOR gate



Modified testbench to accommodate Add/Subtract



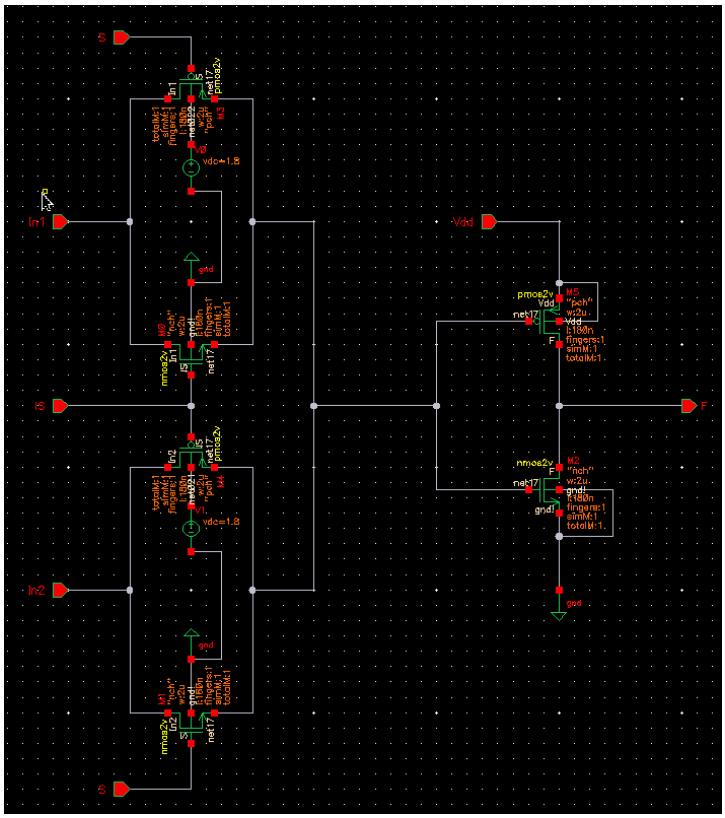
Output of the above circuit



Implementation of 4x1 multiplexer

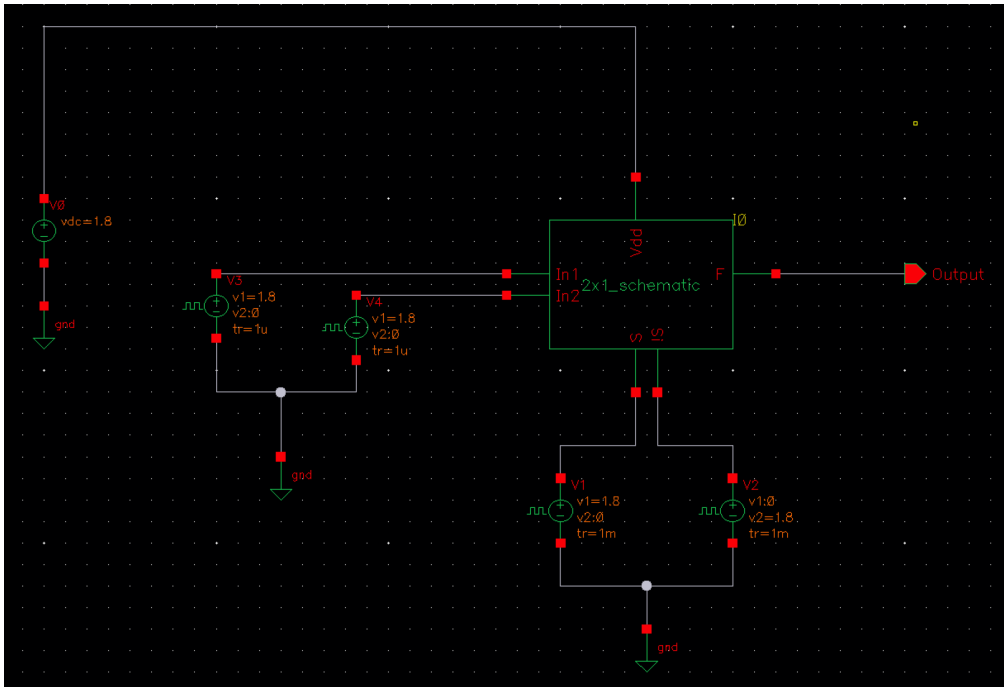
Multiplexer is a circuit which selects the input to be passed based on the provided select signals. We can implement a 4x1 multiplexer using three 2x1 multiplexers and two select signals. In general to implement a nx1 multiplexer we require (n−1) 2x1 multiplexers and $\log_2 n$ number of select signals (here assuming n is a power of 2).

Schematic for 2x1 multiplexer

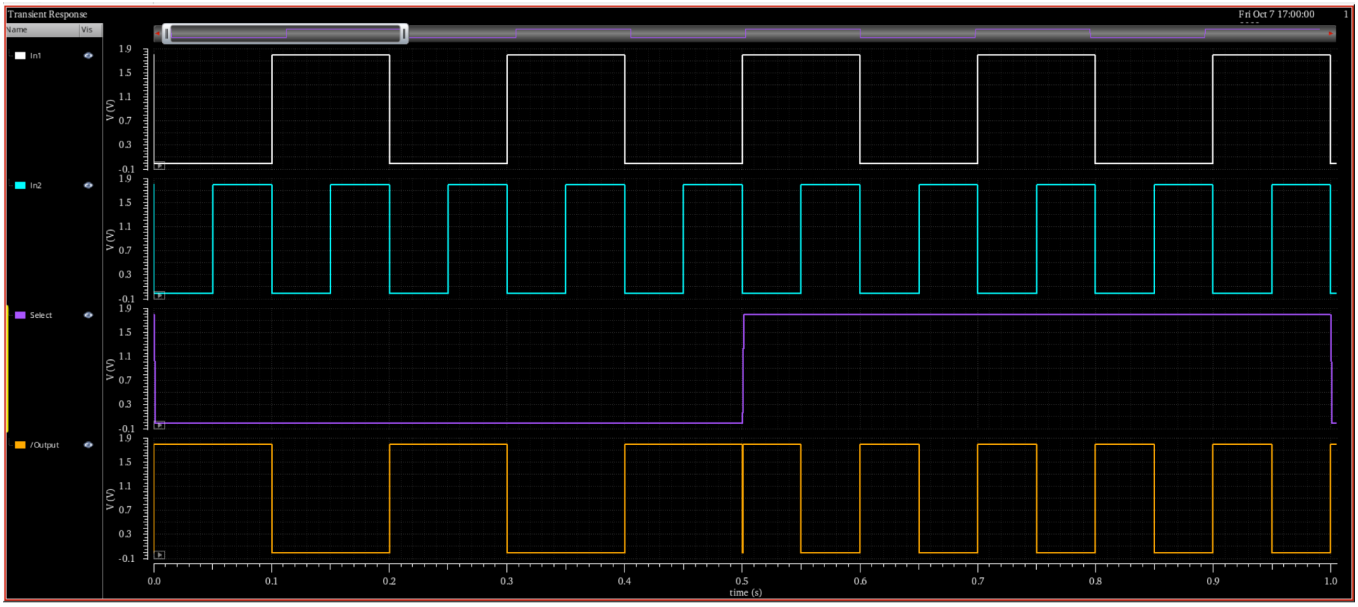


Here we have implemented a multiplexer using pass transistor logic with an inverter to drive the external load.

Testbench for 2x1 multiplexer

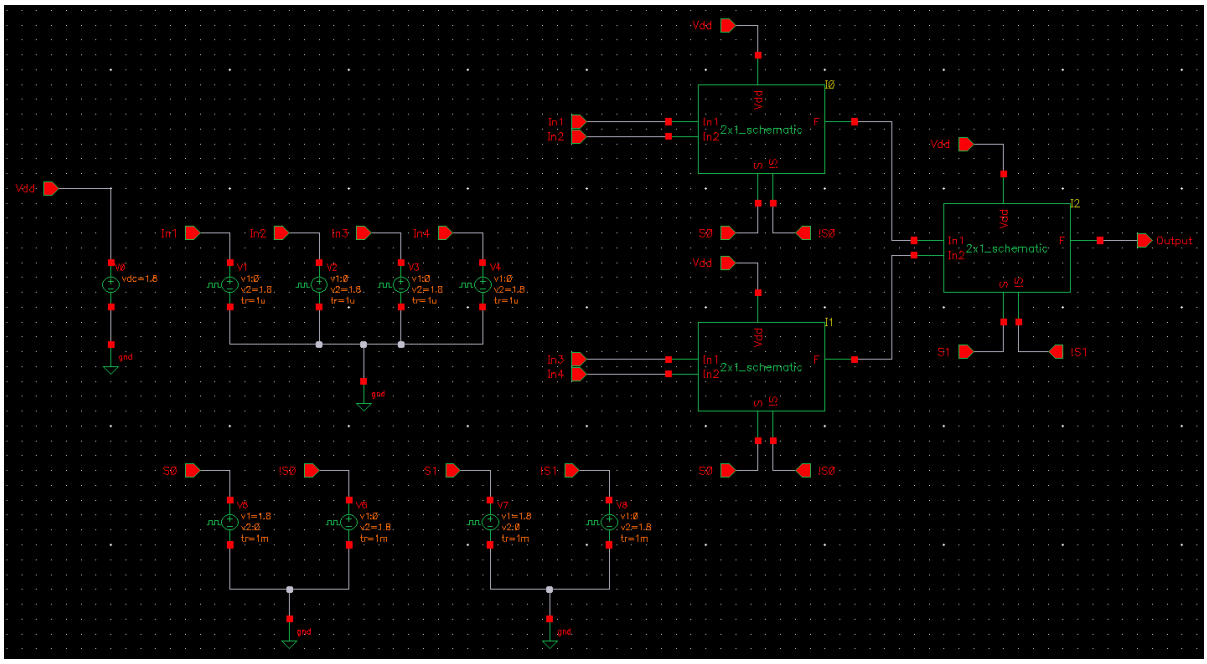


Output for the above testbench



The inverter causes the output to be an inverted version of input in a 2x1 mux (as seen in the above image) but for 4x1 mux the output is the same as input since the input is being inverted twice. In a more general sense when n is an even power of 2, the output is correct and when it is an odd power of 2 the output is an inverted form of input.

Testbench for 4x1 multiplexer



Output plot for the above testbench