DESIGN AN 8-TO-1 MULTIPLEXER USING TRANSMISSION GATE LOGIC

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

FINAL REPORT

Submitted in partial fulfilment of course

EEE F313 – Analog & Digital VLSI Design

FIRST SEMESTER 2017-18

BY

Mohit Garg	*2015A3PS0231P*
Digvijay Bansal	*2015A3PS0187P*
Mohit Padhye	*2015A3PS0137P*
Mayank Raj	*2015A8PS0488P*

SUBMITTED TO

Dr. Anu Gupta

Professor, Department of Electrical and Electronics Engineering

Dr. Nitin Chaturvedi

Assistant Professor, Department of Electrical and Electronics Engineering

Problem 14 - A

Multiplexer using transmission gate logic style

Design an 8-to-1 multiplexer using transmission gate logic at 200MHz frequency with 500fF load capacitance.

Specifications and Assumptions:

- TSMC018 technology
- L = 180 nm for all transistors
- λ = 0.09 µm
- Layout for capacitance and resistor are not drawn
- Body of PMOS is connected to VDD and body of NMOS is connected to GND
- A single VDD and GND are used throughout the circuit
- Operating frequency of 200MHz is specified.
- Load Capacitance is given 500fF.
- The assumed ambient temperature for which the results are presented is 27°C
- The ratio of mobility of majority charge carriers in NMOS to PMOS is µn=2µp

Layout (with DRC Check)

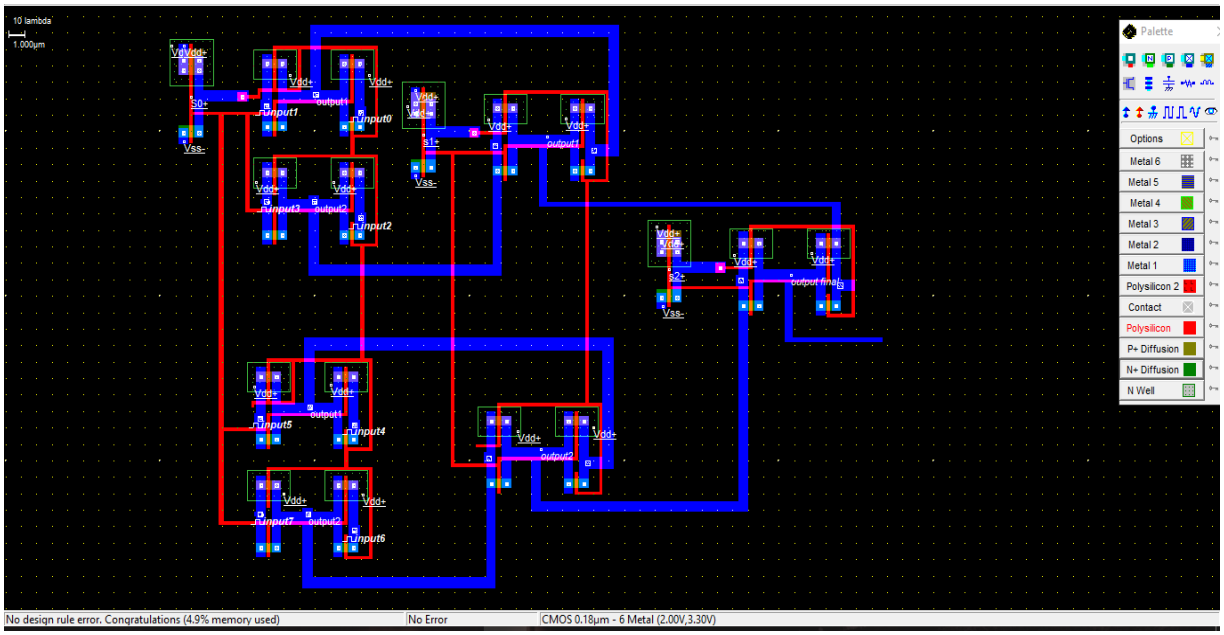


Fig. Layout for 8:1 Multiplexer using Transmission Gate Logic

Schematic

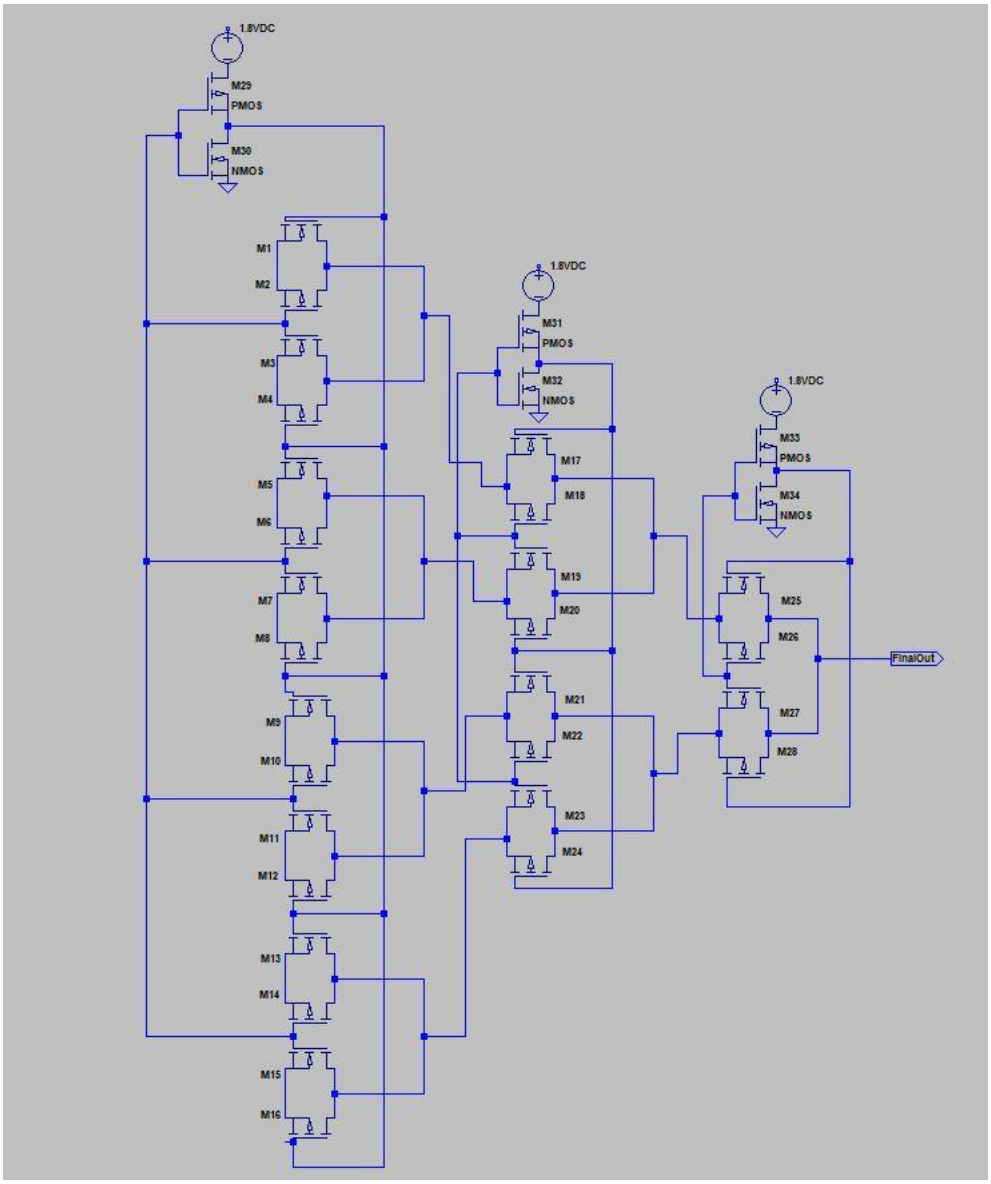


Fig. Schematic of the 8:1 multiplexer using Transmission Gate Logic

No. of Transistors:

Transmission Gates – 14 (14 NMOS – W/L = 3; 14 PMOS – W/L = 3)

Inverters - 3(3 NMOS – W/L = 4; 3 PMOS – W/L = 8)

W/L Calculations

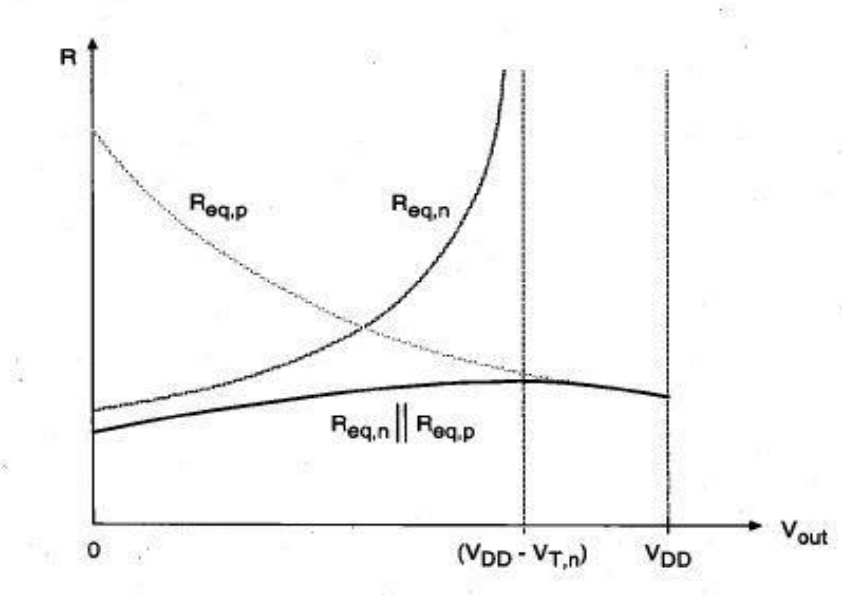


Fig. Equivalent resistance of the CMOS transmission gate, plotted as a function of the Output voltage.

As seen from above graph, $R_{eq,n} || R_{eq,p}$ is maximum when NMOS is OFF and PMOS is in Linear Region.

$$R_{eq} = \frac{2}{\mu_p C_{ox} \frac{W}{L} (|2(V_{DD} - |V_{Tp}|) - (V_{DD} - V_{out})|)}$$

As W/L Increases, R_T Decreases → Delay ($0.69R_{on}C_L$).

But $C \approx C_{jsw}$. (P) + C_{jA}. (A)

P ∝ W and A ∝ W.

So, C ∝ W.

As W/L Increases, R_T Decreases → But Capacitance Increases.

Thus an optimum point is found using HSPICE.

As shown in figure, on increasing W/L from 2 to 4, delay decreases but it is quite small when W/L goes from 3 to 4.

Thus, to prevent MUX from occupying disproportionately large area, we choose W/L = 3 for transmission gates.

In the graph below, the red curve denotes the input waveform with a rise time and fall time of 0.1ns each. The yellow curves denote the output waveform, each representing a different value of channel width from 0.6µm to 1.8µm. As W/L increases, the delay decreases by a very negligible amount. However, as a trade-off, the size of the transistor increases in a much larger proportion, with a very marginal decrease in delay. Hence, minimum size of W=0.6µm has been chosen for the transmission gates MOSFETs.

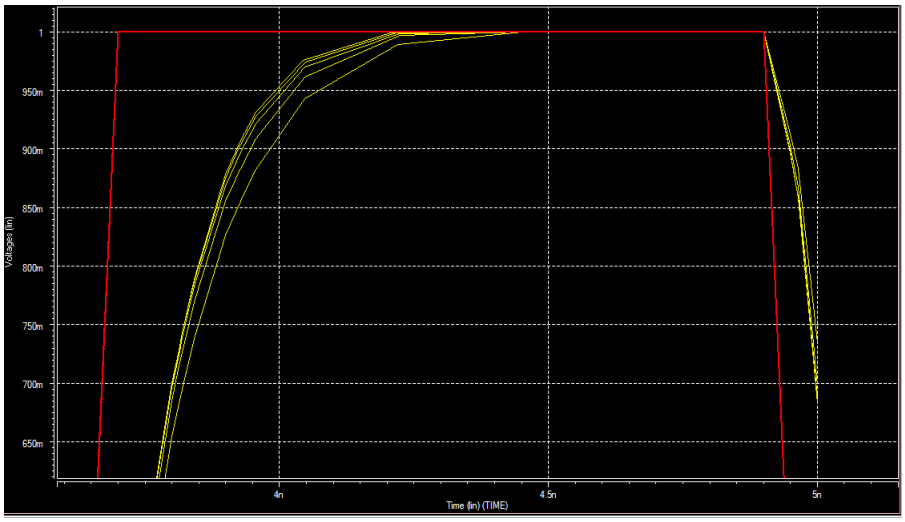


Fig. Graph showing transient analysis to choose proper sizing of transistors

Similarly, the graph plotting the transient analysis to choose W/L for the inverters is shown ahead. The red curve represents the input whereas the yellow curves represent the output curve, each for a different value of channel width. We can see from the graph that as W increases, the delay reduces significantly. After increasing to an appreciable value, however, the increase becomes very marginal. We can afford to increase the widths of a few transistors by a small amount if we are getting appreciably reduced delay. Hence, the W/L of the NMOS has been chosen to be 0.8µm and that of PMOS is 1.6µm. A pull-up to pull-down ratio of 2:1 has been maintained for stable voltage transfer characteristics of the CMOS inverters.

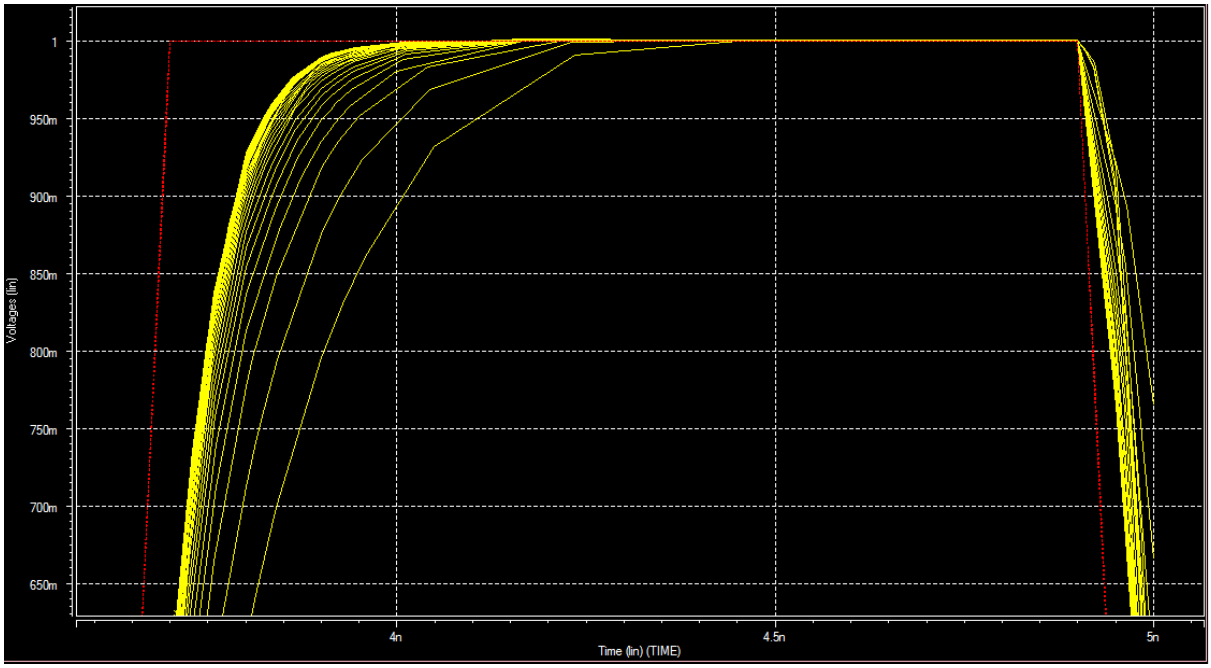


Fig. Graph showing transient analysis to choose proper sizing of transistors

Output Waveforms

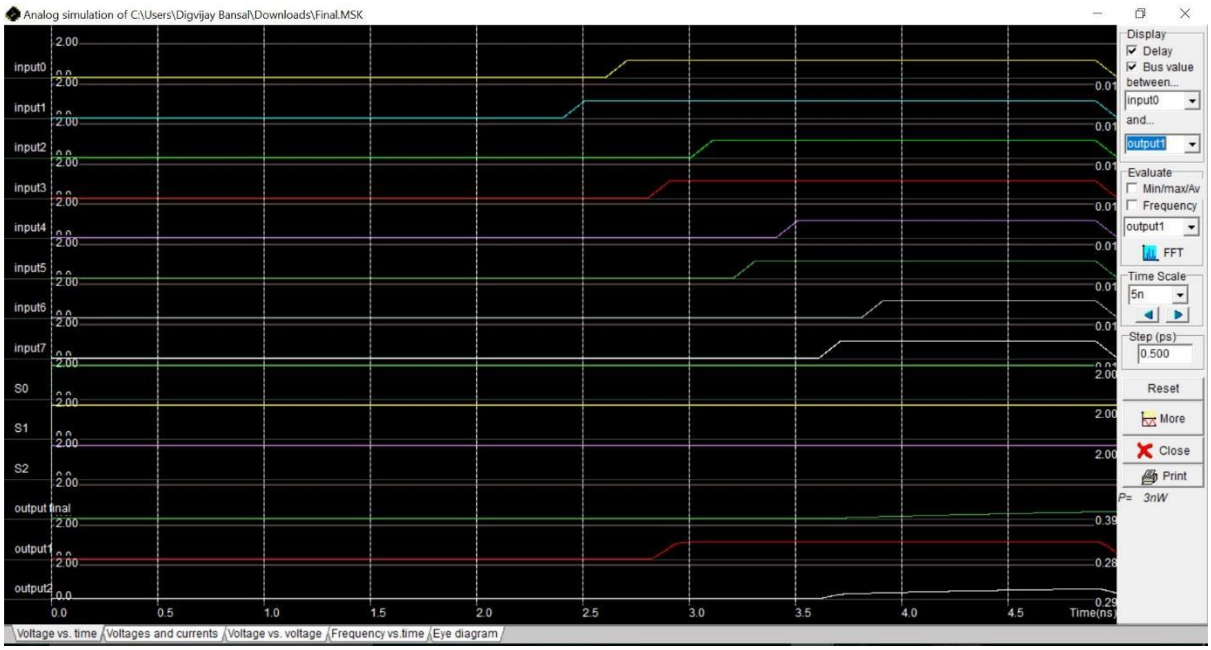


Fig - Output Waveforms of 8:1 MUX (Output Final = Input 7)

With Load Capacitance (500fF).
The select lines are given – 111 as Input.

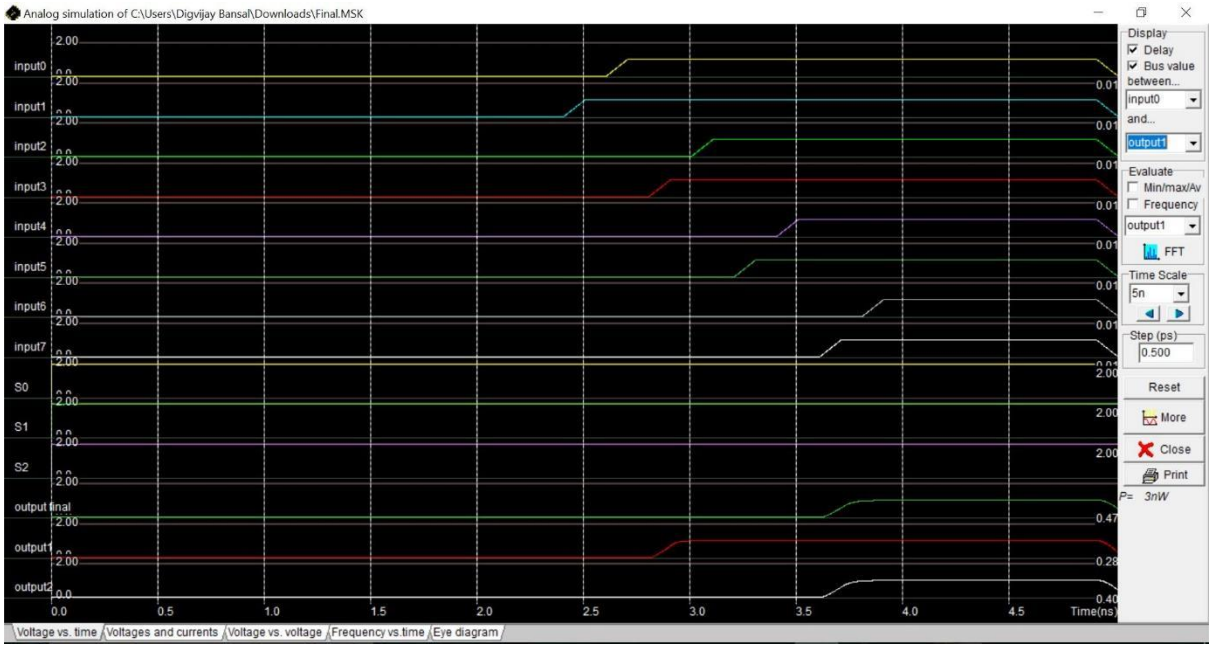
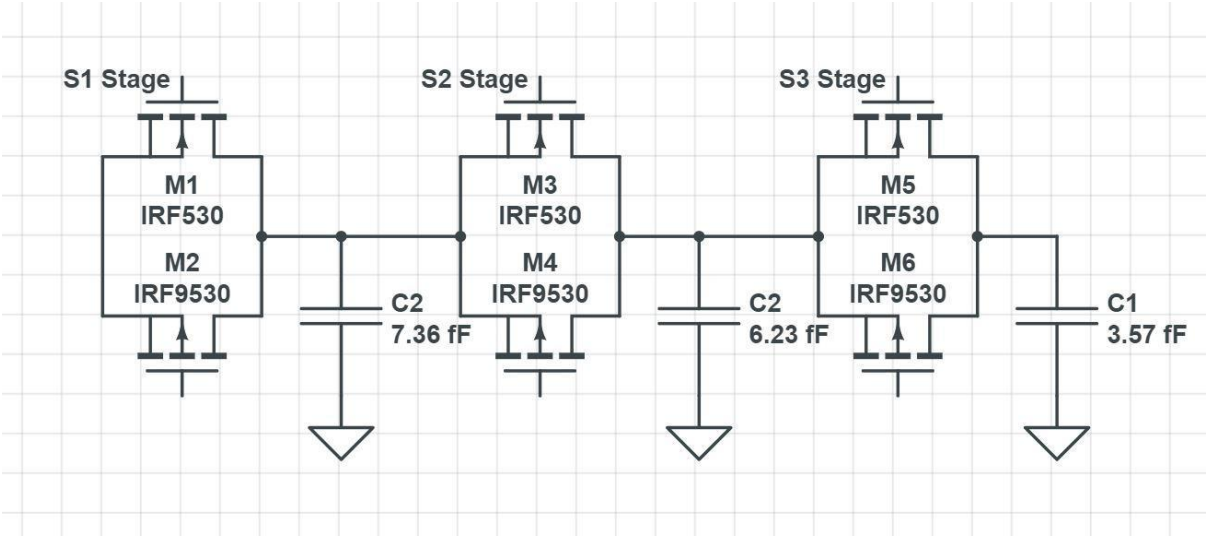


Fig - Output Waveforms of 8:1 MUX (Output Final = Input 7)

Without Load Capacitance (500fF).
The select lines are given – 111 as Input.

Delay Calculations

According to the schematic of the 8:1 MUX, every input has to go via the same path of 3 transmission gates to reach the final output as per the inputs given in the select lines.
So, delay calculation will include 3 transmission gates and their intermediate capacitances, the calculation is done via Wire Delay Model.



Without Load Capacitance (500fF).
The select lines are given – 111 as Input.

$$R_{eq} = \frac{2}{\mu_p C_{ox} \frac{W}{L} (|2(V_{DD} - |V_{Tp}|) - (V_{DD} - V_{out})|)}$$

Wire Delay = $0.69R_1(C_1 + C_2 + C_3) + R_2(C_2 + C_3) + R_3(C_3)$

$R_{1, 2, 3}$ = 9.6618 KΩ

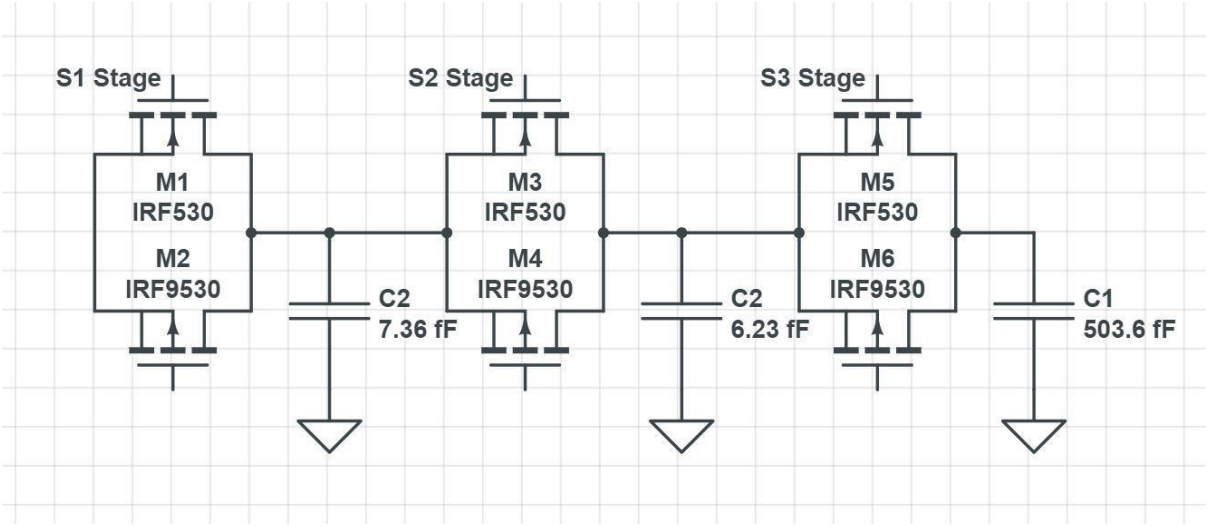
Capacitance values have been extracted from

C_1 = 7.36fF

C_2 = 6.23fF

C_3 = 3.57fF

τ = 0.2035ns



With Load Capacitance (500fF).
The select lines are given – 111 as Input.

$$R_{eq} = \frac{2}{\mu_p C_{ox} \frac{W}{L} (|2(V_{DD} - |V_{Tp}|) - (V_{DD} - V_{out})|)}$$

Wire Delay = $0.69R_1(C_1 + C_2 + C_3) + R_2(C_2 + C_3) + R_3(C_3)$

$R_{1,2,3}$ = 9.6618 KΩ

C_1 = 7.36fF

C_2 = 6.23fF

C_3 = 503.57fF

τ = 10.20ns

The SPICE model files used in the layout are as follows:

```
* n-mos model 3 :
 * low leakage
 .model n1 nmos level=3 vto=0.50 uo=115.000 tox= 4.0e-9
 +ld =-0.020u theta=0.200 gamma=0.350
 +phi=0.500 kappa=0.080 vmax=100.00k
 +cgso=100.0p cgdo=100.0p
 +cgbo= 60.0p cjsw=240.0p
 *
 * p-mos model 3:
 * low leakage
 .model p1 pmos level=3 vto=-0.60 uo=30.000 tox= 3.0e-9
 +ld =0.010u theta=0.300 gamma=0.400
 +phi=0.200 kappa=0.010 vmax=100.00k
 +cgso=100.0p cgdo=100.0p
 +cgbo= 60.0p cjsw=240.0p
 *
```

The threshold voltage for the NMOS is 0.5V and that for the PMOS is -0.6V.

Practical Values of Delay

Graphs have been plotted in SPICE to measure the value of propagation delay. Delay has been calculated till half the maximum value of the voltage. Following figure shows the transient analysis for the output and the input. The red curve is the input whereas the yellow curve is the output.

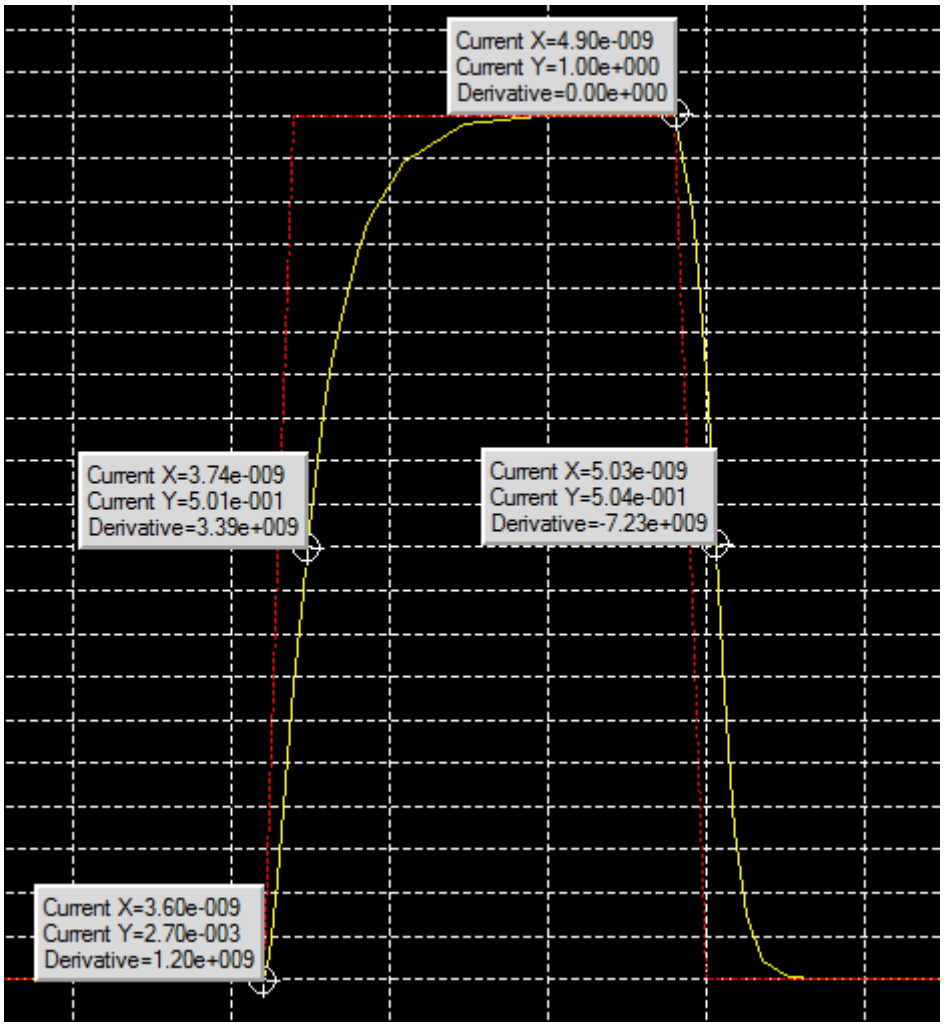


Fig. Time response of input and output for delay calculation

The low to high time propagation delay is:
$\boldsymbol{\tau}p_{LH}$ = 3.74ns - 3.60ns = 0.14ns

The high to low time propagation delay is:
$\boldsymbol{\tau}p_{HL}$ = 5.03ns - 4.9ns = 0.13ns

Hence, the average time propagation delay is:
$\boldsymbol{\tau}p = (\boldsymbol{\tau}p_{LH} + \boldsymbol{\tau}p_{HL})/2 = (0.14ns + 0.13ns)/2 = 0.135ns$

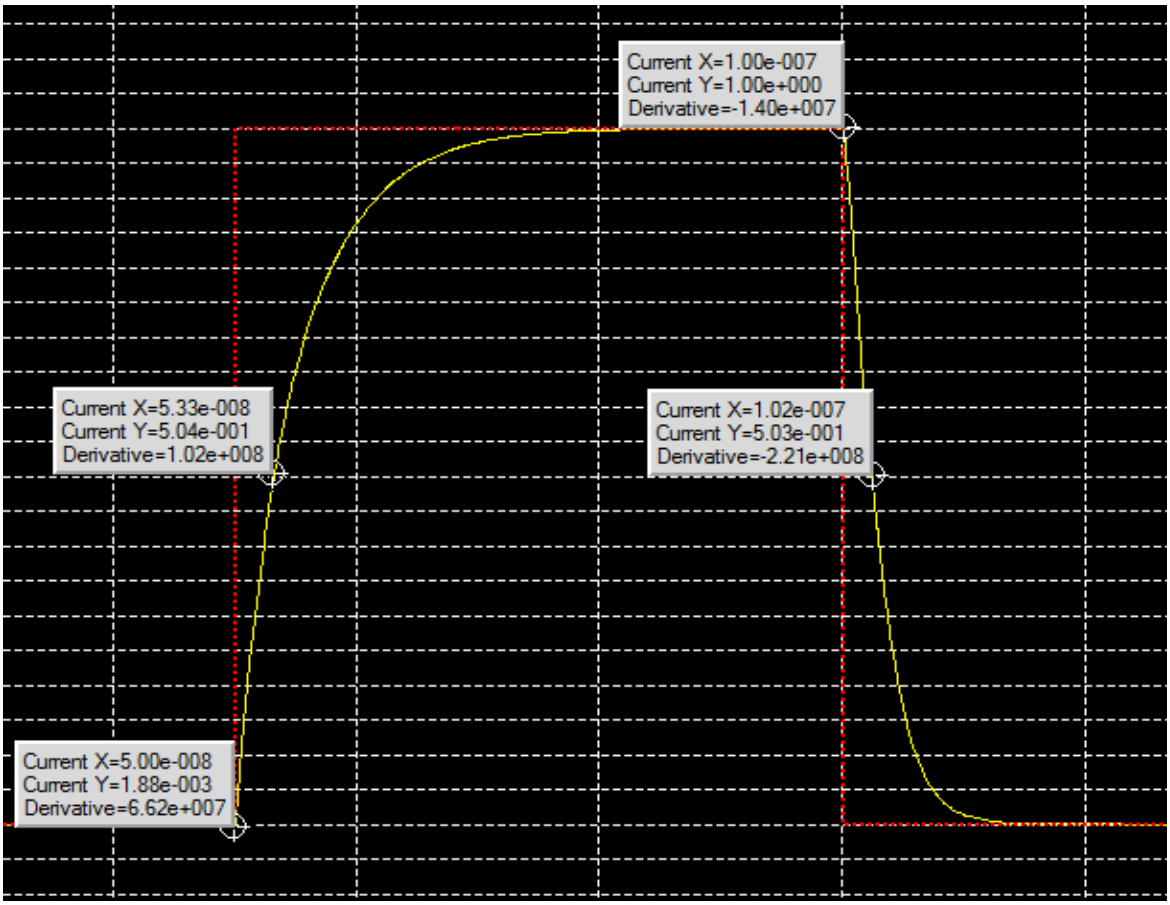


Fig. Time response of input and output for delay measurement

The low to high time propagation delay is:
$\boldsymbol{\tau}p_{LH} = 53.3ns - 50ns = 3.3ns$

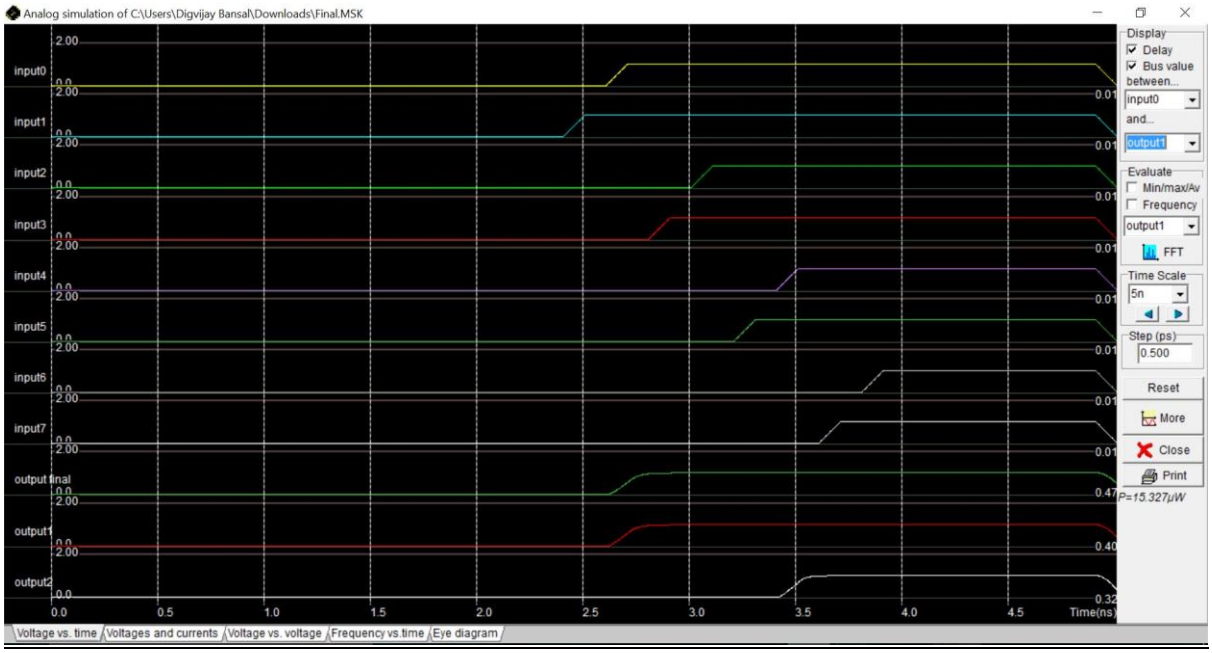
The high to low time propagation delay is:
$\boldsymbol{\tau}p_{HL} = 102ns - 100ns = 2.0ns$

Hence, the average time propagation delay is:
$\boldsymbol{\tau}p = (\boldsymbol{\tau}p_{LH} + \boldsymbol{\tau}p_{HL})/2 = (3.3ns + 2ns)/2 = 2.65ns$

RESULTS:

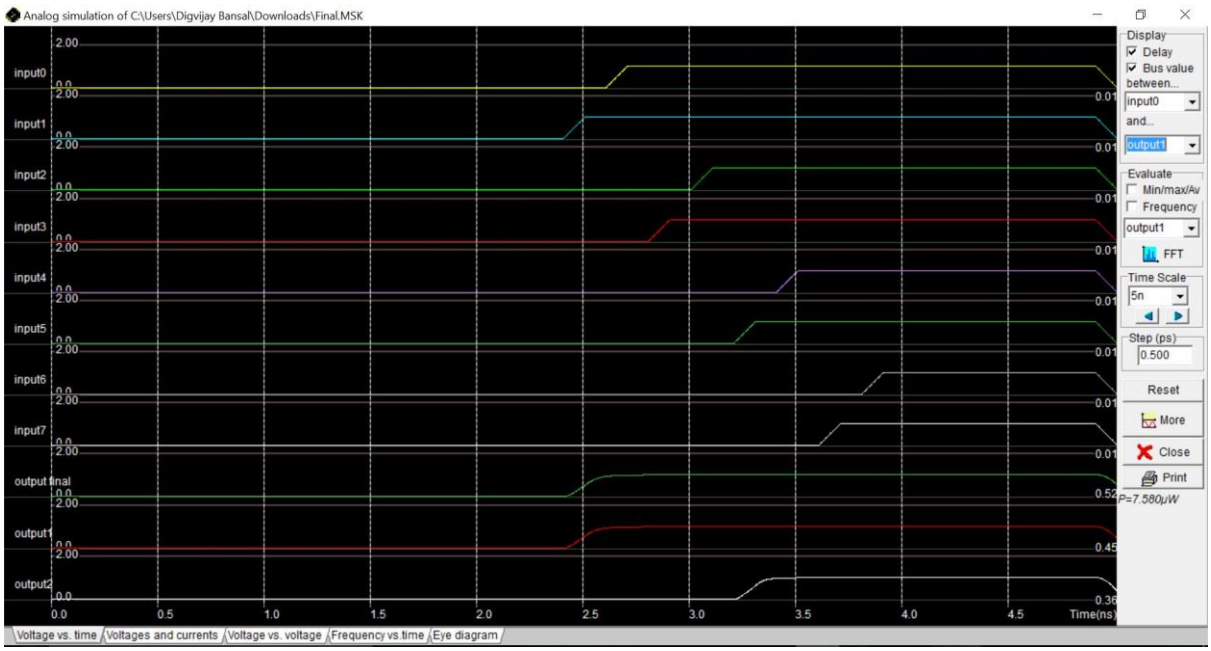
Parameter	**Calculated Value (Worst Case)**	**Measured Value**
Time propagation delay	0.2035ns	0.135ns
Time propagation delay (with 500fF load)	10.20ns	2.65ns

Power Consumption



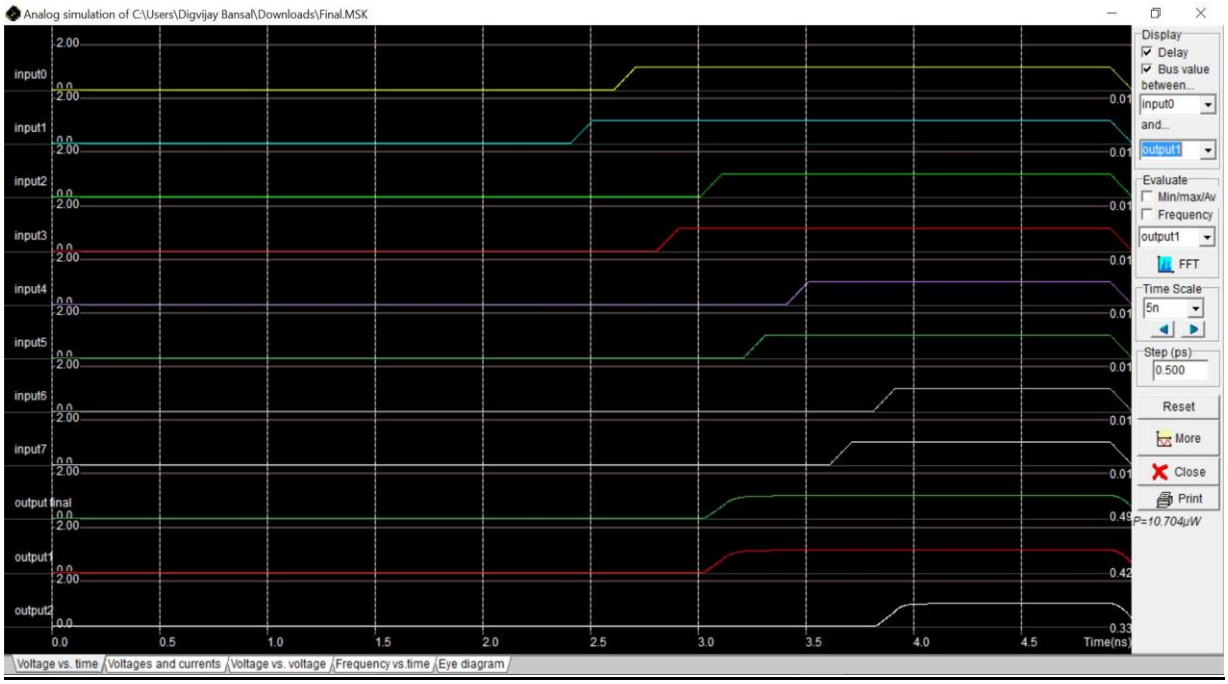
Select Lines - 000

Power Consumption – 15.327µW



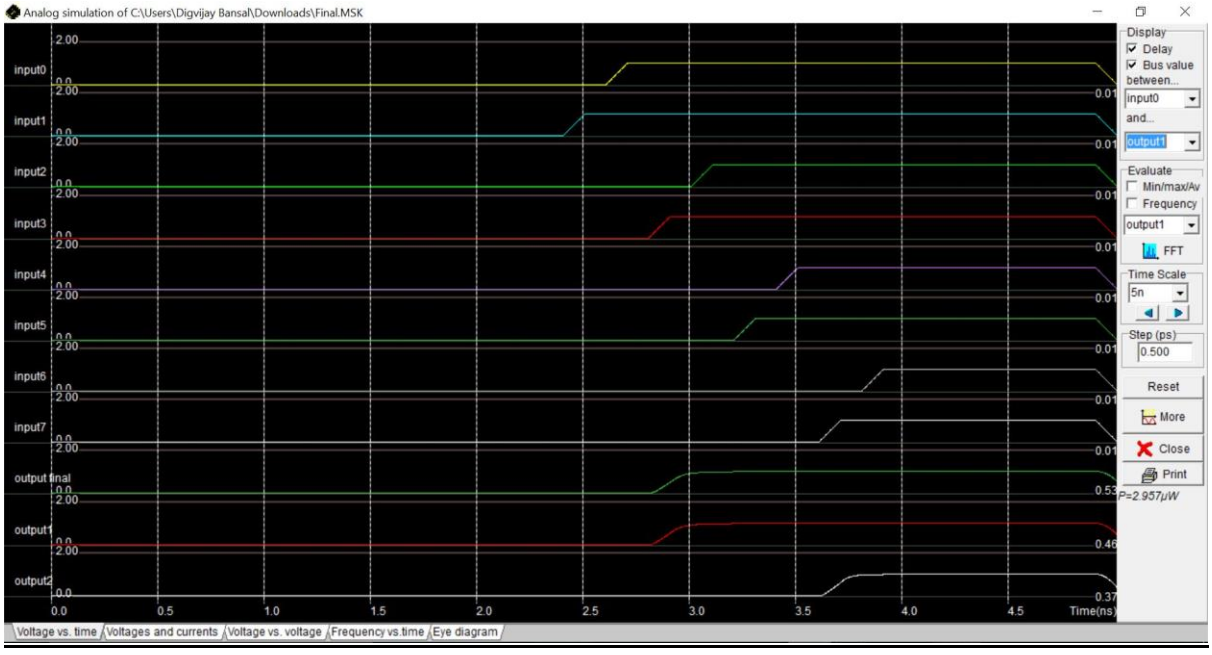
Select Lines - 001

Power Consumption – 7.580µW



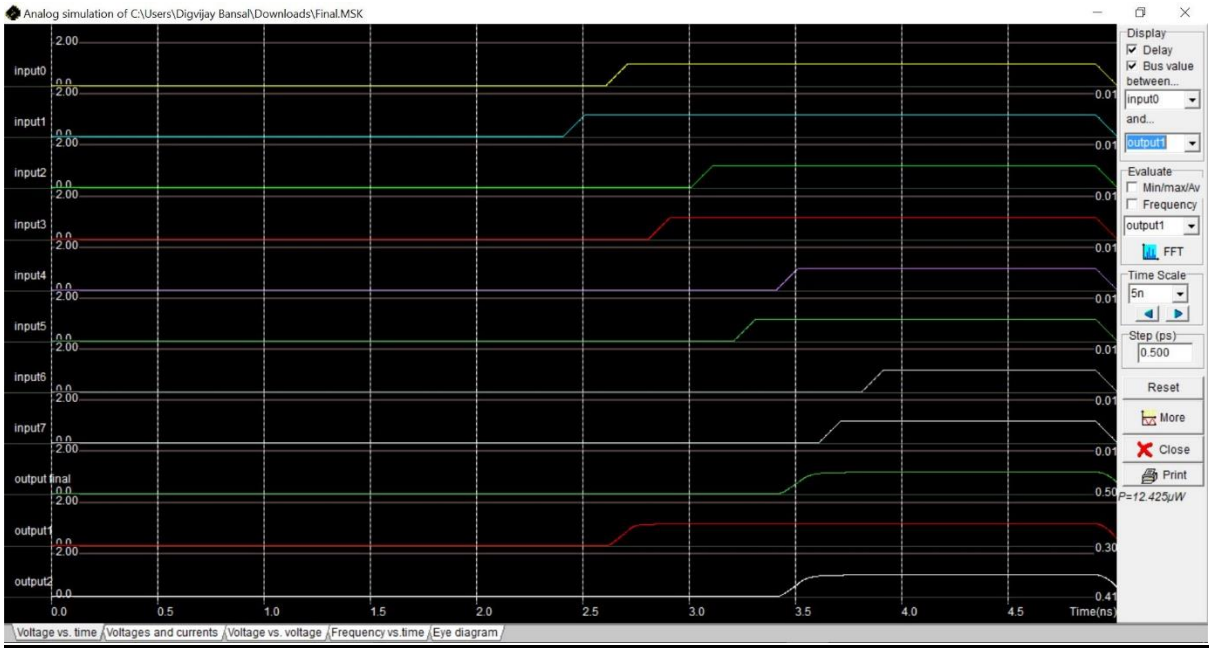
Select Lines – 010

Power Consumption – 10.704µW



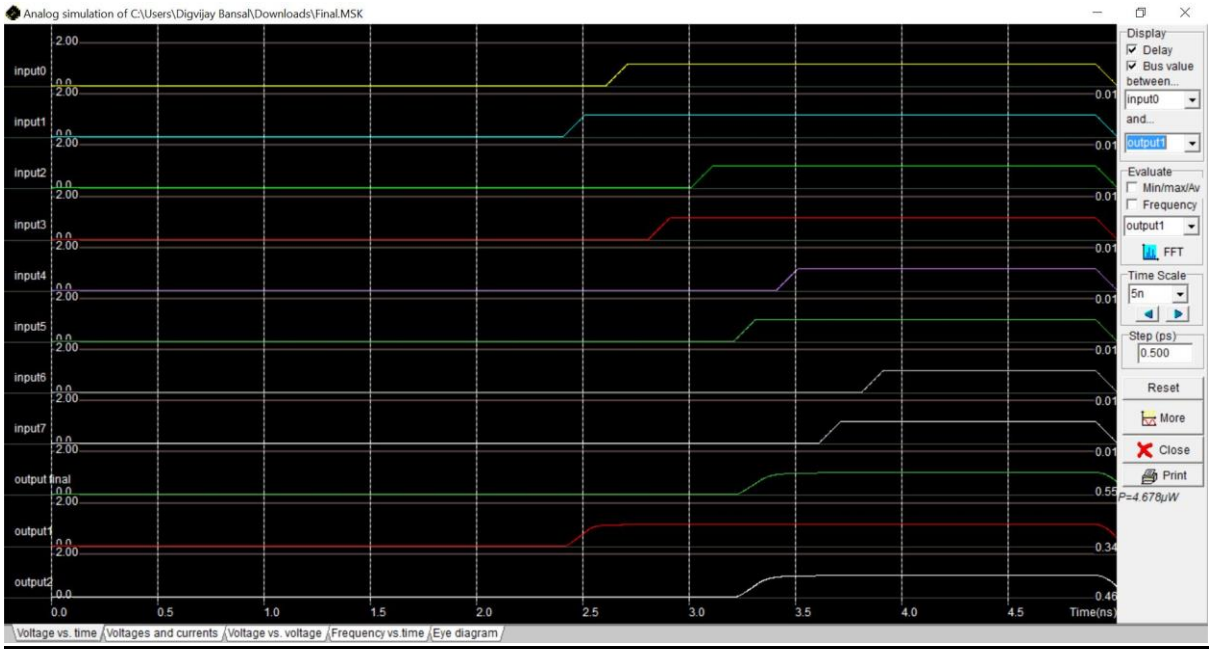
Select Lines - 011

Power Consumption – 2.957µW



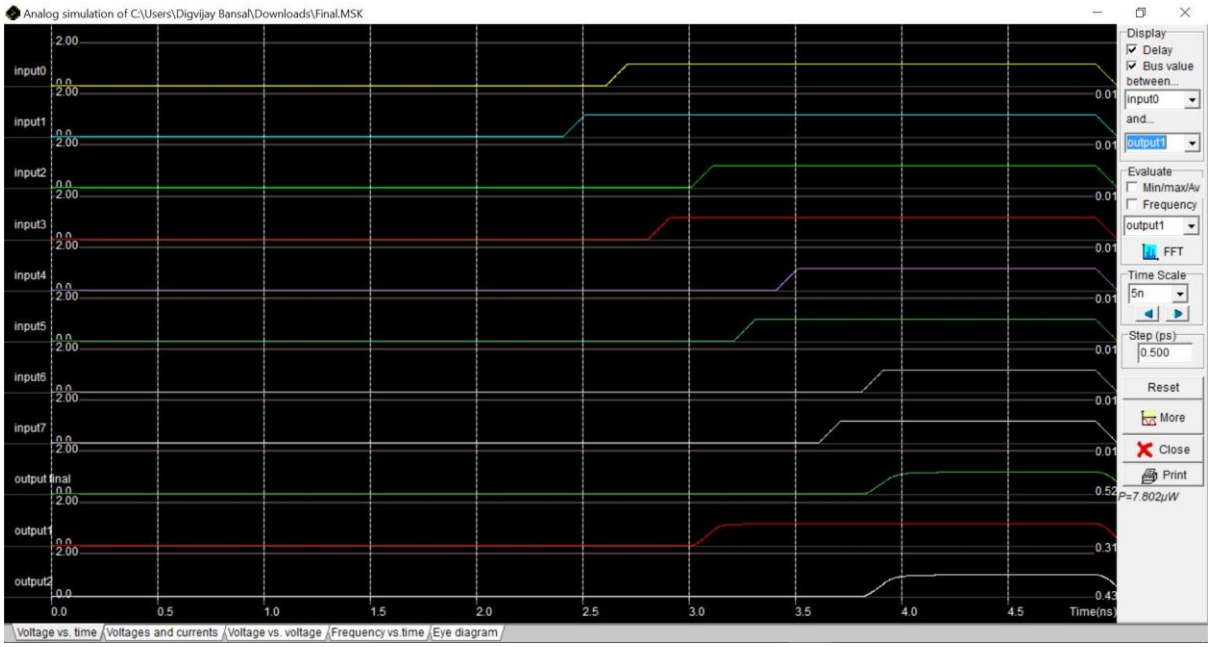
Select Lines - 100

Power Consumption – 12.425µW



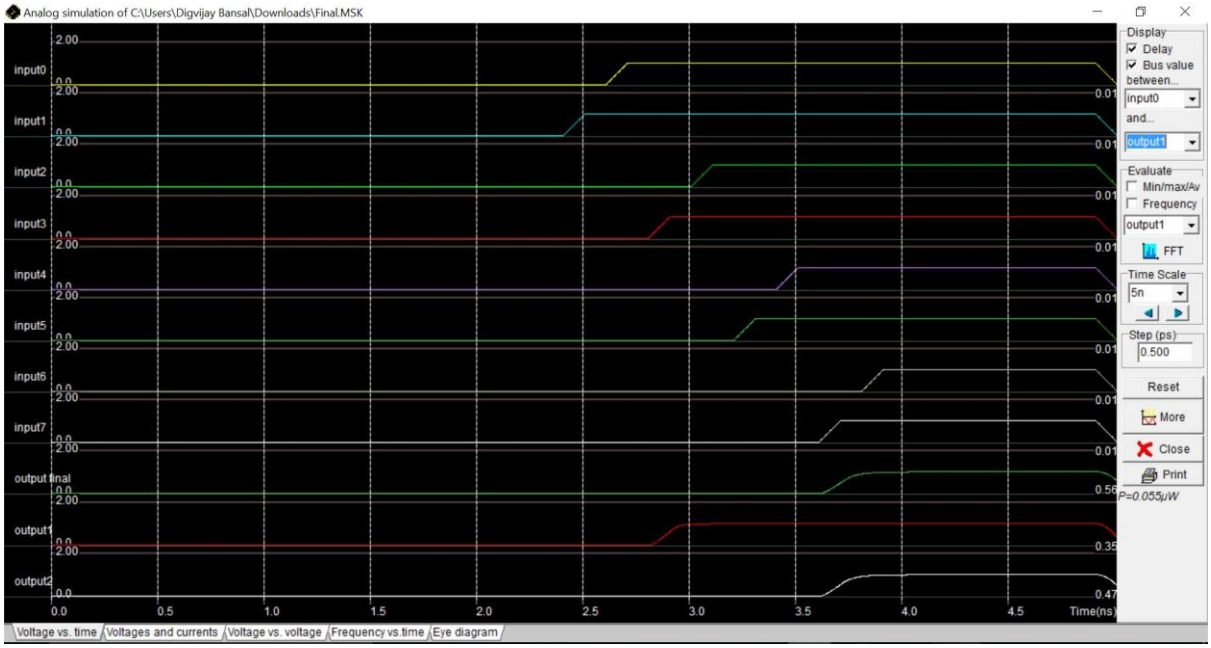
Select Lines - 101

Power Consumption – 4.678µW



Select Lines - 110

Power Consumption – 7.802µW



Select Lines - 111

Power Consumption – 0.055µW

Problem 14 – B

Verilog Program

Implement Non-restoring division algorithm for dividing six bit number by a three bit number. Use the modelling style of your choice.

Algorithm as per Digital Design Course:

Non- Restoration Algorithm

Non-restoration method eliminates the need for restoring.

Initializing the Registers.

- Divisor is loaded into M.
- Dividend loaded into Q.
- Initialize A to Zero.

Algorithm

- If sign of A is positive shift A & Q left by one bit and subtract M from A.

OR

- If sign of A is negative, shift A &Q by one bit and add M to A.
- If MSB of result is 1, then Q0 is set to zero otherwise set to one.
- Repeat steps 1-2 n (number of bits in dividend) times.
- At the end if sign of A is negative add M to A to get the correct remainder.

Verilog Code

```
module final_algo(dividend,divisor,rem,quo); //Initializing the module,
                                              Three Inputs


  input [5:0] dividend;  //6-Bit Register
  input [2:0] divisor;// 3-Bit Register
  output [5:0] rem;// 6-Bit Register, Remainder
  output [5:0] quo;// 6-Bit Register, Quotient

  integer i; // Declaring Integer for Loop
  reg [5:0] Q; // Intermediate Registers for Value Holding (Dividend)
  reg [5:0] M; // Intermediate Registers for Value Holding (Divisor)
  reg [5:0] A; // Intermediate Registers for Value Holding

  always@(dividend or divisor)
  // Always Checks For Non-Negativity Condition
  begin
    M = 0; M = M + divisor; // As Divisor is 3 Bit and M is 6 Bit
    Q = dividend;
    A = 0;// Initializing with Zero, As Per Algorithm
    for(i=0; i<6; i=i+1)//Looping No. of Bits(6) in Dividend Times
    begin
```

```
        if(A[5]==0)  //Checking The Sign of the Dividend(Positive)
        begin
            A = A << 1;   //Left Shift the Bits of a Register
            A[0] = Q[5];  //As Verilog Shifts Circularly, Manual
Shifting
            Q = Q << 1;   // Left Shift the Bits of Q Register
            Q = Q & 6'b111110; //As Verilog Shifts Circularly, Manually
           Inserting Zero at End
            A = A - M; // Subtracting A from M
        end
        else //The Loop When A is Negative
        begin
            A = A << 1; //Left Shift the Bits of a Register
            A[0] = Q[5];//As Verilog Shifts Circularly, Manual Shifting
            Q = Q << 1; // Left Shift the Bits of Q Register
            Q = Q & 6'b111110; //As Verilog Shifts Circularly, Manually
          Inserting Zero at End
            A = A + M; // Adding A and M
        end
        if(A[5]==1)// Checking MSB
        begin
            Q = Q & 6'b111110; //Setting LSB of Q = 0
        end
        else
        begin
            Q = Q | 6'b000001; //Setting LSB of Q = 1
        end
      end
      if(A[5]==1) // Checking MSB
      begin
          A = A + M; // Adjusting the Remainder
      end
   end
   assign quo = Q; // Assigning Values to Quotient
   assign rem = A; // Assigning Values to Remainder
endmodule
```

Verilog Example

Dividend: 6'b0000111

Divisor: 3'b010

Quotient: 6'b000011

Remainder: 6'b000001

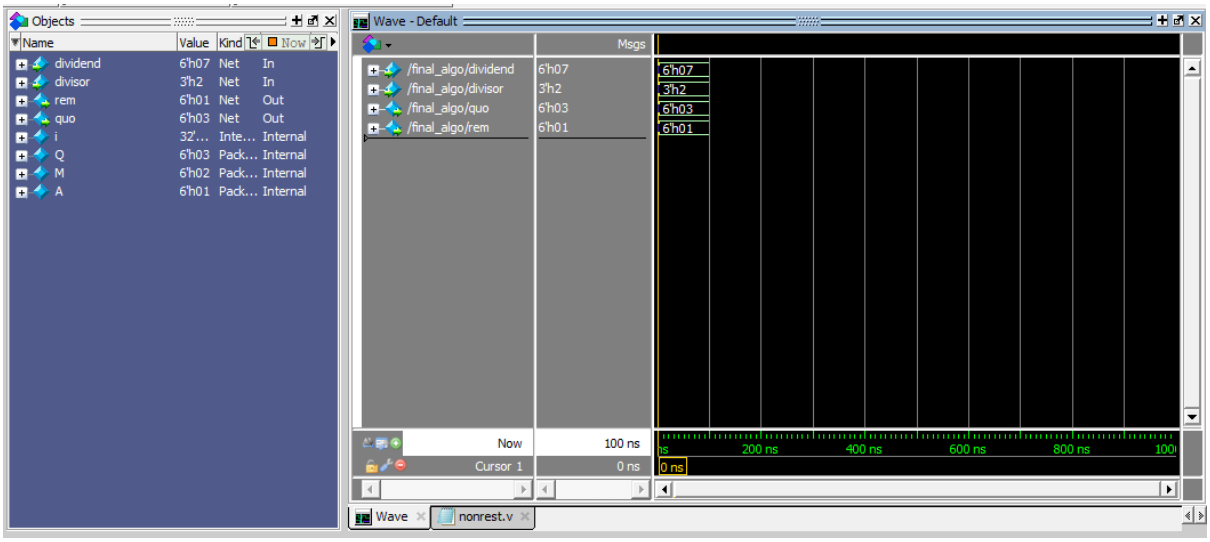


Fig -Final output of Non Restoration Division Algorithm.