Digital Project Report

Spring 23

Submitted to:

Dr. Sameh Assem Ibrahim

Arithmetic and Logical Unit Design Project

Team 27

NAME	ID	Department
Fady Mohamed Abdelfatah	2000964	ECE
Abdallah Karim Mohammed	2000993	ECE
Karen Ehab Alfons	2001645	ECE
Omar Mohamed Aboelfetoh	2001109	CSE
Abdallah Yasser Mohamed	2001034	CSE
Salma Waleed Ismail	2000308	CSE
Bigoal Shereen Soliman	2001632	CSE
Omar Tamer Yehia	2001709	CSE
Mohamed Fareed Mohamed	2000933	CSE
Abdulaziz Hamad Albdowi	2002265	ECE

The Code:

1- ALU Code:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

ENTITY Alu_unit IS
   PORT(
      a,b : IN STD_LOGIC_VECTOR(3 downto 0);
      sel : IN STD_LOGIC_VECTOR(3 downto 0);
      Result_OP : OUT STD_LOGIC_VECTOR(7 downto 0)
      );
END Alu_unit ;

ARCHITECTURE Arch OF Alu_unit  IS
   signal logic : STD_LOGIC_VECTOR(3 downto 0);
   signal arithop : STD_LOGIC_VECTOR(7 downto 0);
BEGIN
   -- Logic Unit
   WITH sel SELECT
      logic  <= NOT a WHEN "1000",
         NOT b WHEN "1001",
         a AND b WHEN "1010",
         a OR b WHEN "1011",
```

```
        a XOR b WHEN "1100",
        a XNOR b WHEN "1101",
        a NAND b WHEN "1110",
        a NOR b WHEN OTHERS;

    -- Arithmetic Unit
    process(a,b,sel)
    begin
      case sel is
        when "0000" => arithop <= STD_LOGIC_VECTOR( (a) + "00000001");
        when "0001"=> arithOP <= STD_LOGIC_VECTOR( (a) - "00000001" );
        when "0010"=> arithOP <= STD_LOGIC_VECTOR( (a) * "0010" );
        when "0011"=> arithOP <= STD_LOGIC_VECTOR( (b) + "00000001" );
        when "0100"=> arithOP <= STD_LOGIC_VECTOR( (b) - "00000001" );
        when "0101"=> arithOP <= STD_LOGIC_VECTOR( (b) * "0010" );
        when "0110"=> arithOP <= STD_LOGIC_VECTOR( (a) + (b) +
"00000000");
        when others => arithOP <= STD_LOGIC_VECTOR( (a) * "0100" );
      end case ;
    end process;

    -- MUX
    Result_OP <= ( "0000" & logic) when sel(3)= '1' else
      arithOP when sel(3) = '0' ;

END Arch;
```

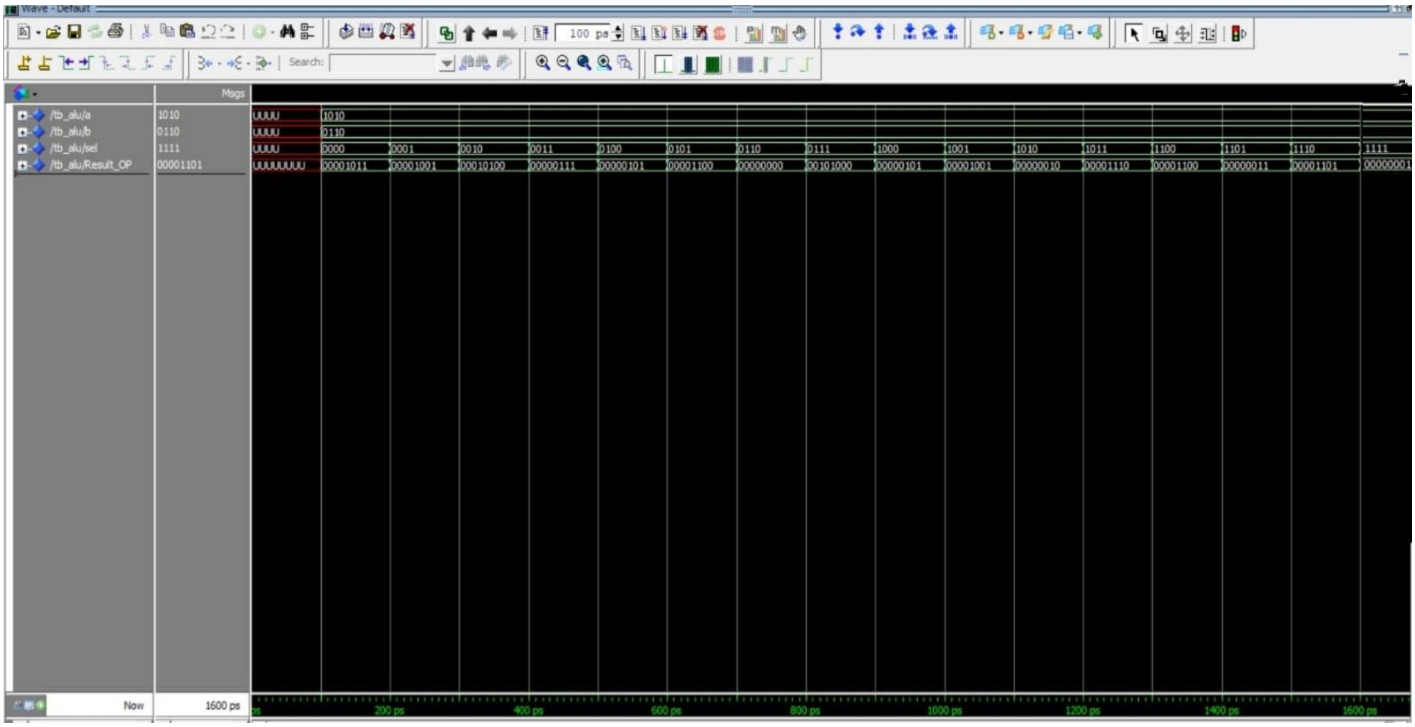
2-Testbench Code:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE IEEE.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;

ENTITY tb_ALU IS
END tb_ALU;

ARCHITECTURE behavior OF tb_ALU IS
   COMPONENT Alu_unit
      PORT(
         a,b : IN STD_LOGIC_VECTOR(3 downto 0);
         sel : IN STD_LOGIC_VECTOR(3 downto 0);
         Result_OP : OUT STD_LOGIC_VECTOR(7 downto 0)
      );
   END COMPONENT;

   --Inputs
   signal a : STD_LOGIC_VECTOR(3 downto 0) ;
   signal b : STD_LOGIC_VECTOR(3 downto 0) ;
   signal sel : STD_LOGIC_VECTOR(3 downto 0) ;
   --Outputs
   signal Result_OP : STD_LOGIC_VECTOR(7 downto 0);

   BEGIN
   uut: ALU_unit PORT MAP (
      a => a,
      b => b,
```

```
    sel => sel,
    Result_OP => Result_OP
);
-- Stimulus process
stim_proc: process --processing in nano sec for simulation
begin
    wait for 100 ps;
    a <="1010";
    b <="0110";
    sel <="0000" ;
    wait for 100 ps;
    sel <="0001" ;
    wait for 100 ps;
    sel <="0010" ;
    wait for 100 ps;
    sel <="0011" ;
    wait for 100 ps;
    sel <="0100" ;
    wait for 100 ps;
    sel <="0101" ;
    wait for 100 ps;
    sel <="0110" ;
    wait for 100 ps;
    sel <="0111" ;
    wait for 100 ps;
    sel <="1000" ;
    wait for 100 ps;
    sel <="1001" ;
    wait for 100 ps;
    sel <="1010" ;
    wait for 100 ps;
```

```
        sel <="1011" ;
        wait for 100 ps;
        sel <="1100" ;
        wait for 100 ps;
        sel <="1101" ;
        wait for 100 ps;
        sel <="1110" ;
        wait for 100 ps;
        sel <="1111" ;
        wait;
    end process;
END;
```

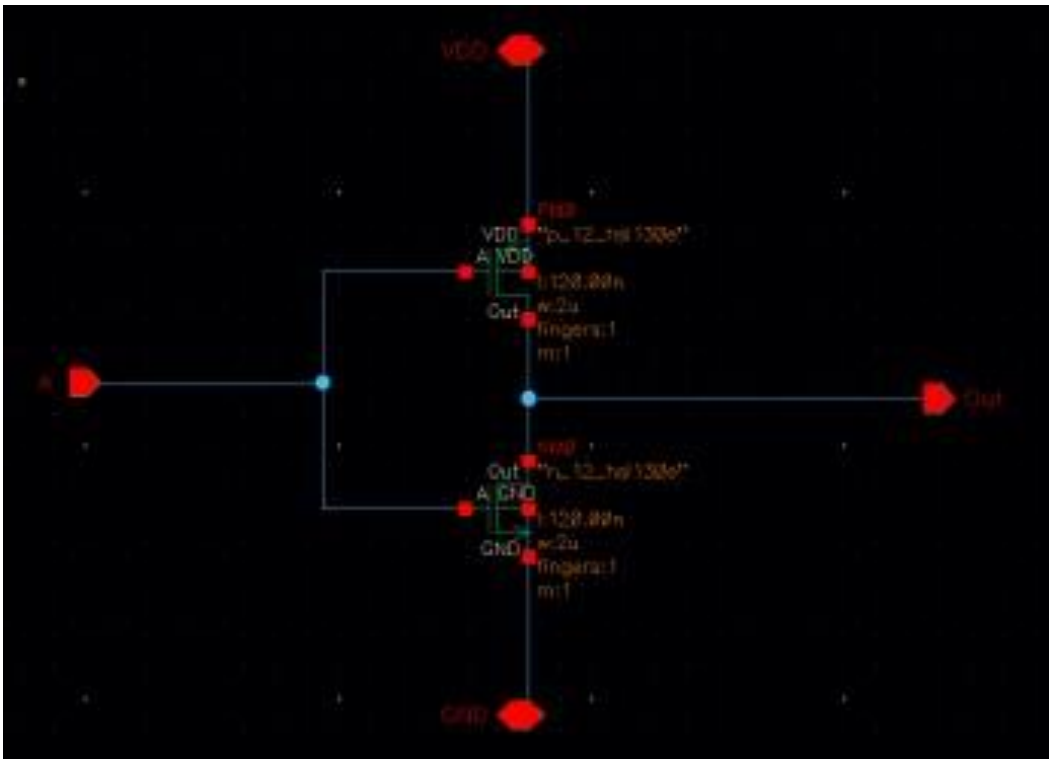
3- Testbench Results:



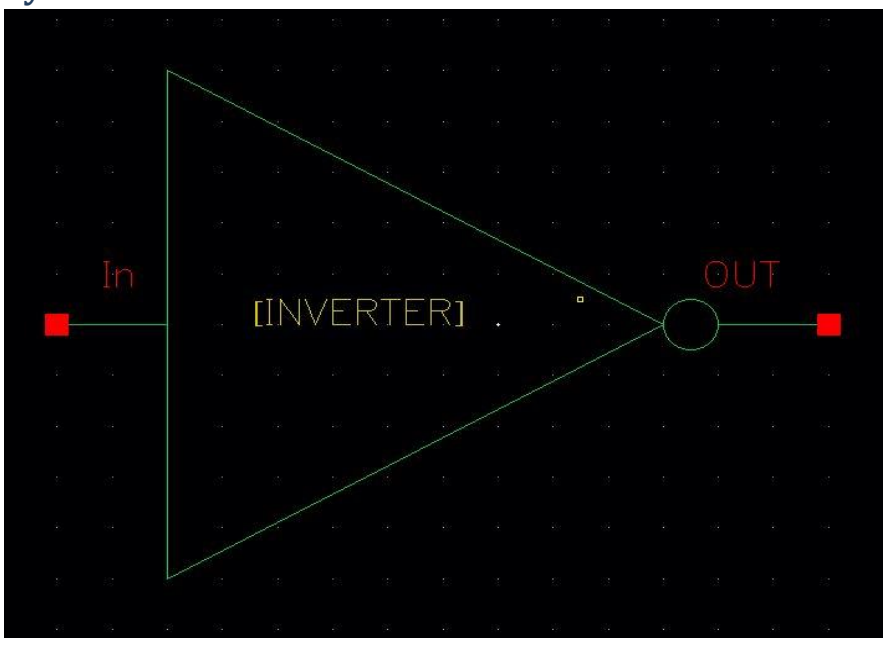
Cadence Schematics, Symbols and Output waveforms:

1) Inverter (NOT):

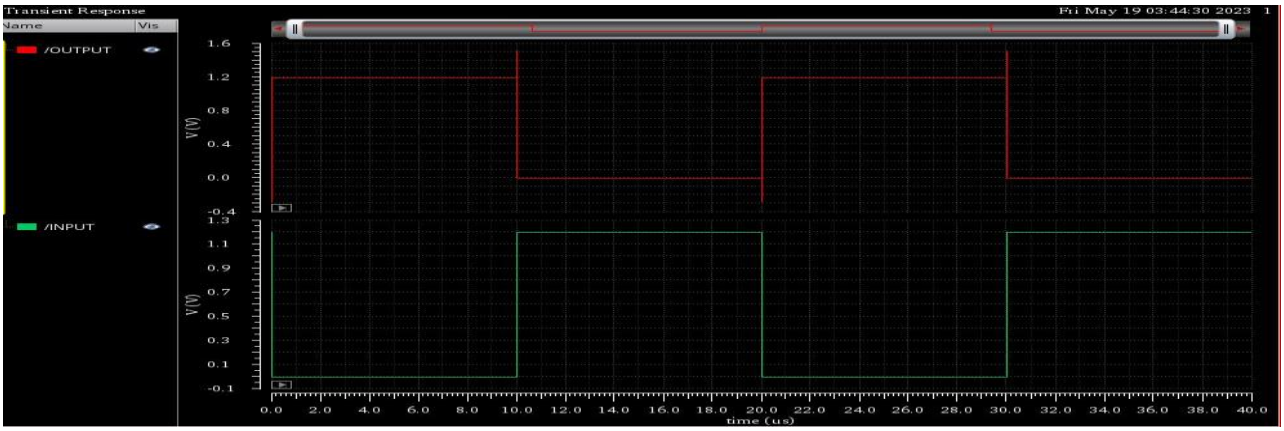
Schematic:



Symbol:

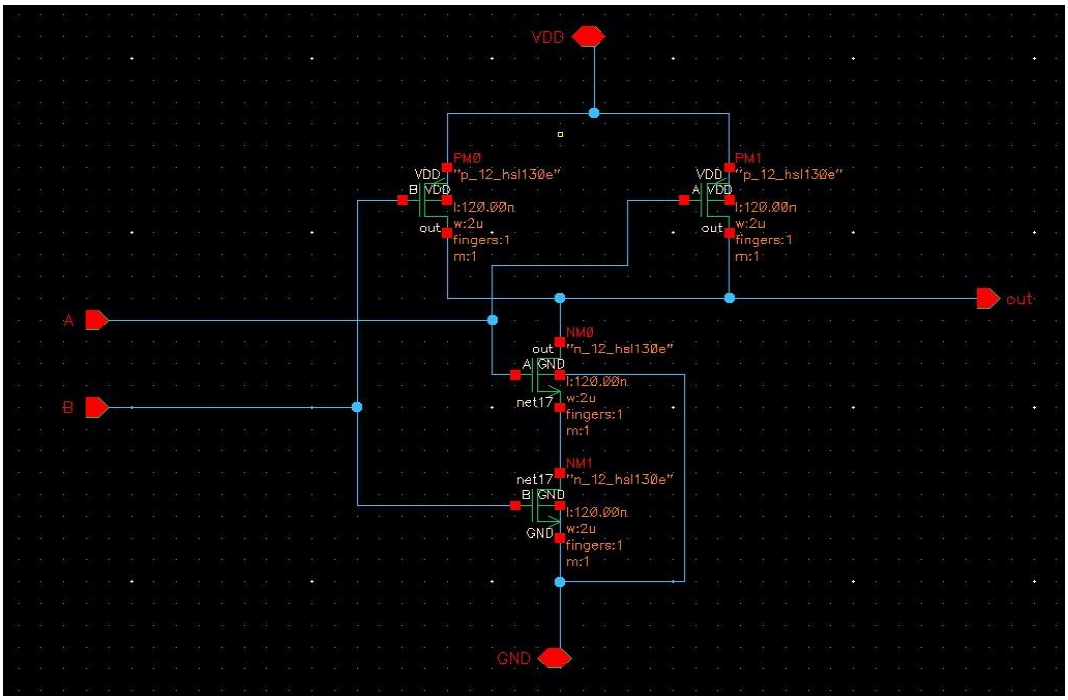


Output waveform:

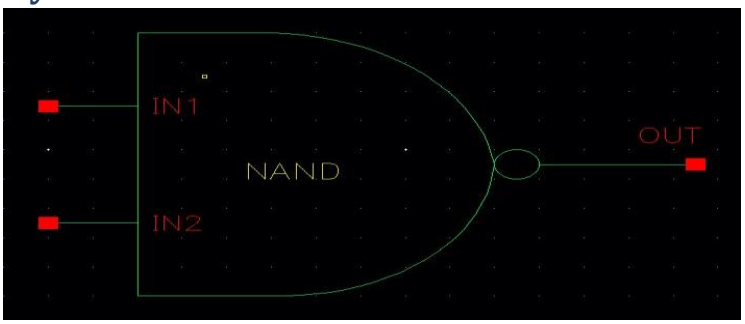


2- NAND:

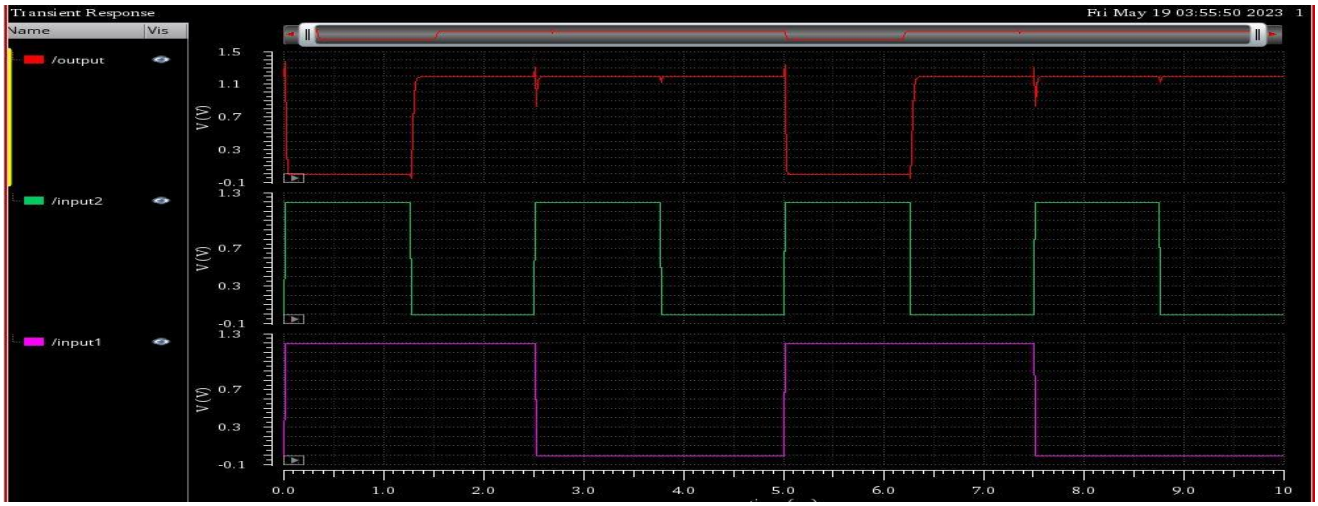
Schematic:



Symbol :

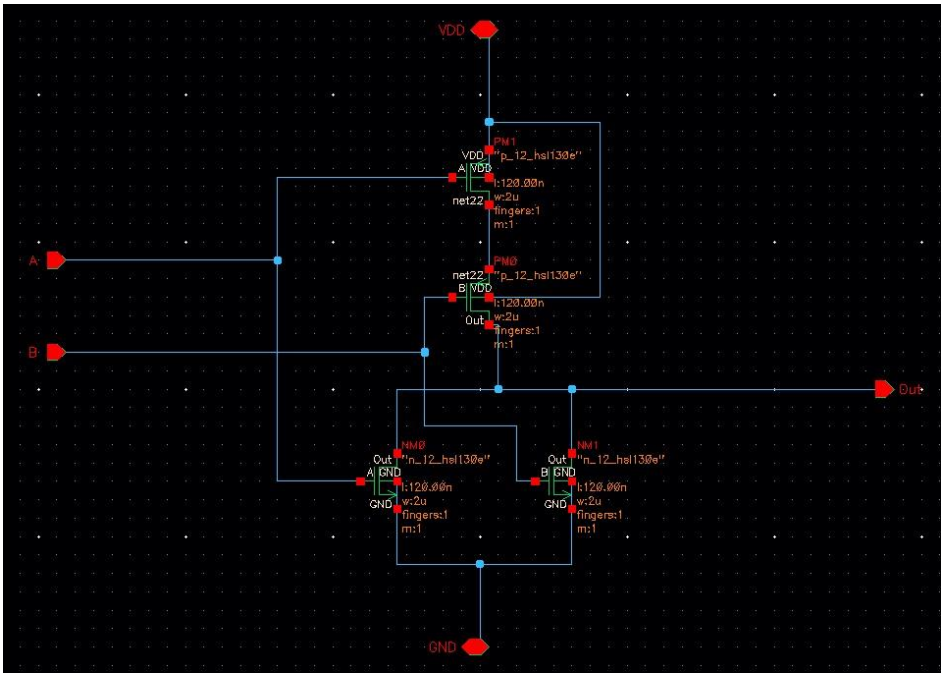


Output waveform:

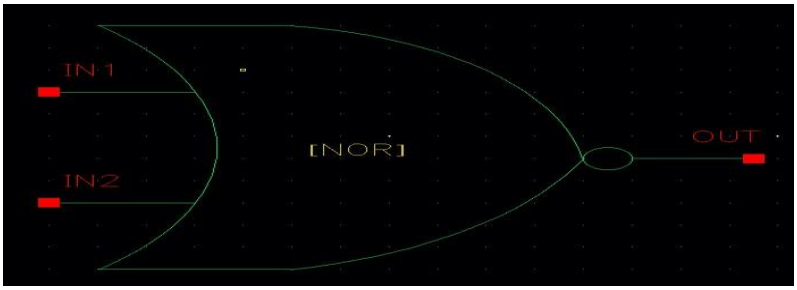


3- NOR:

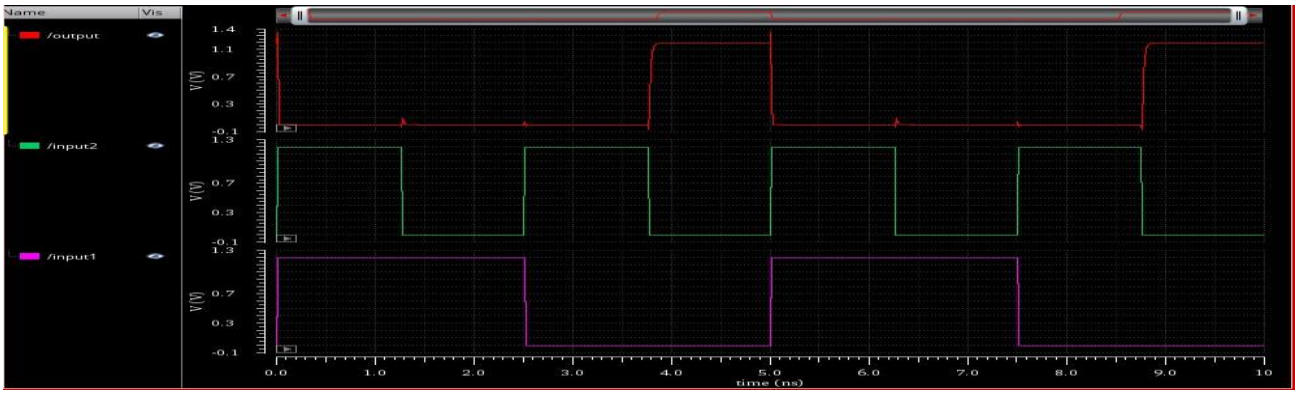
Schematic:



Symbol :

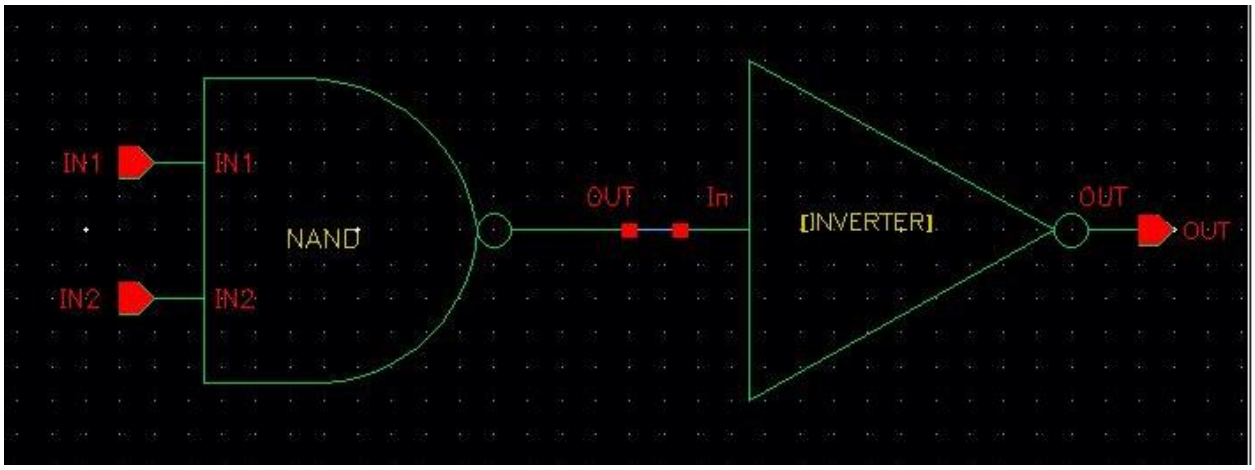


Output waveform:

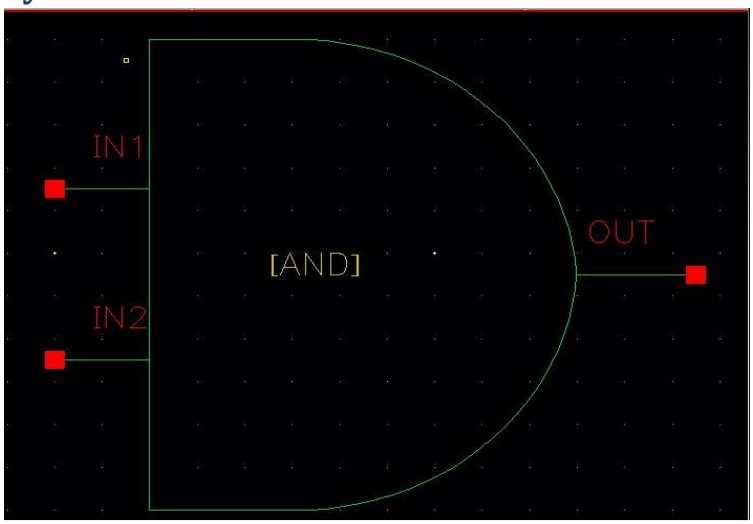


4-AND:

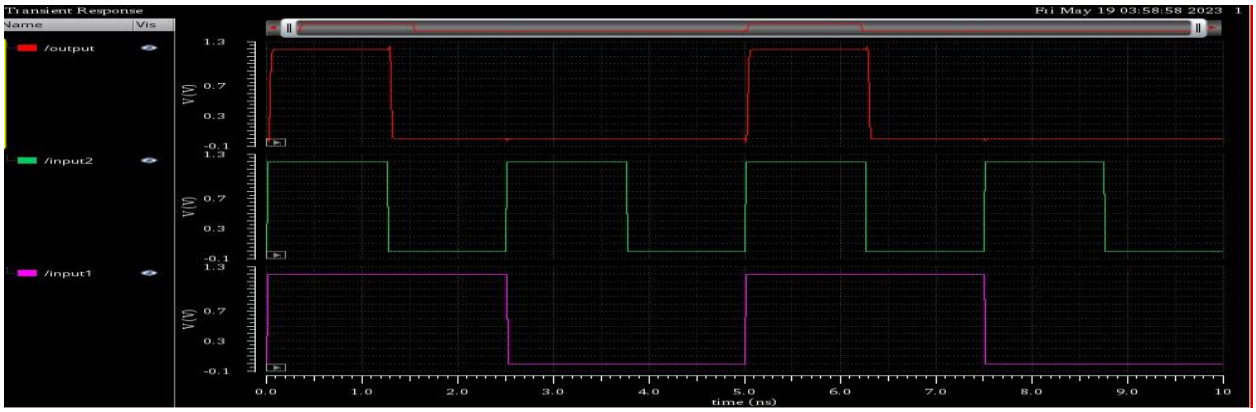
Schematic:



Symbol:

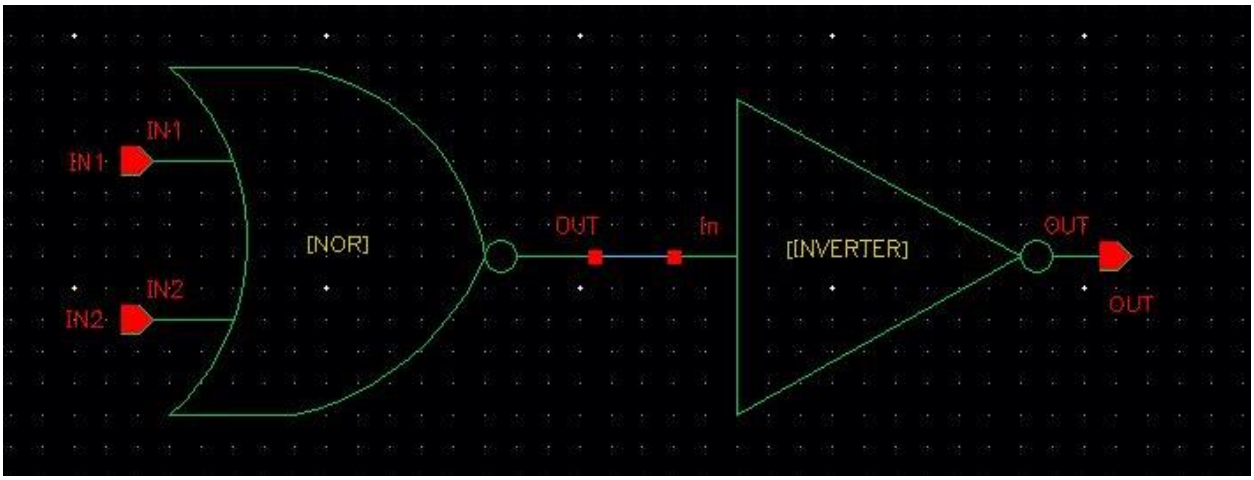


Output waveform:

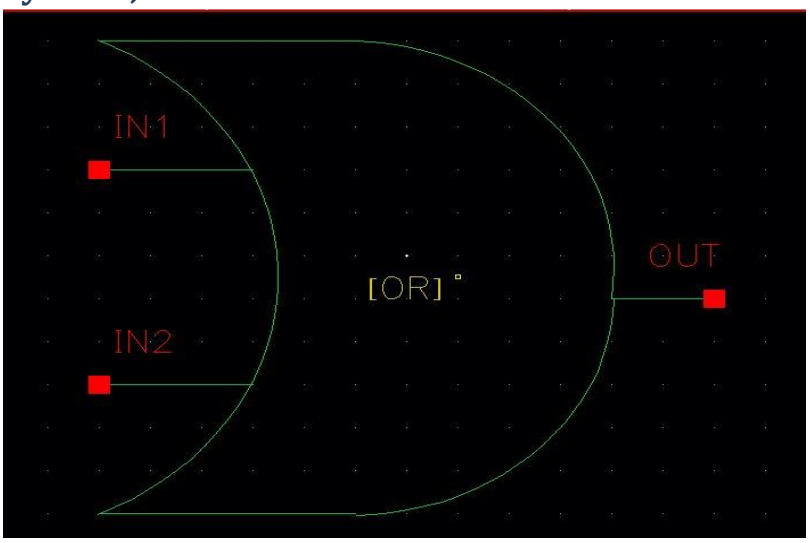


5-OR:

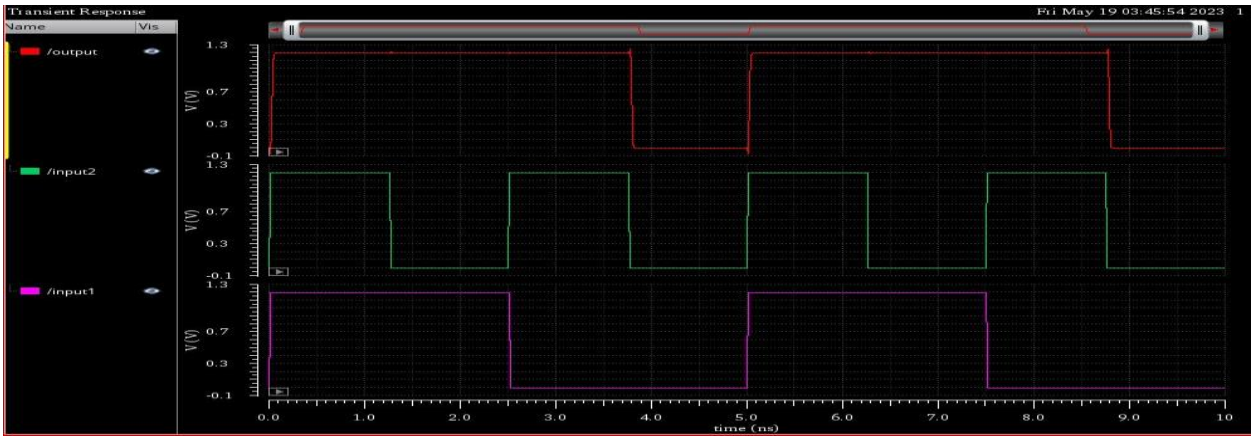
Schematic:



Symbol;

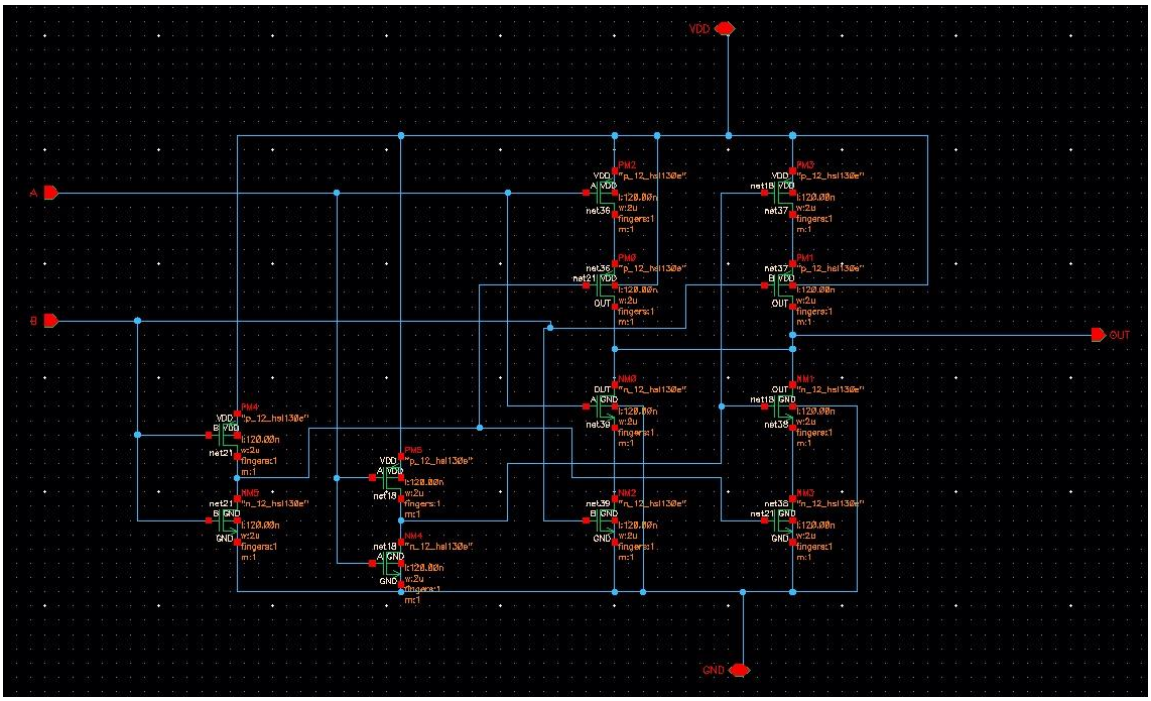


Output waveform:

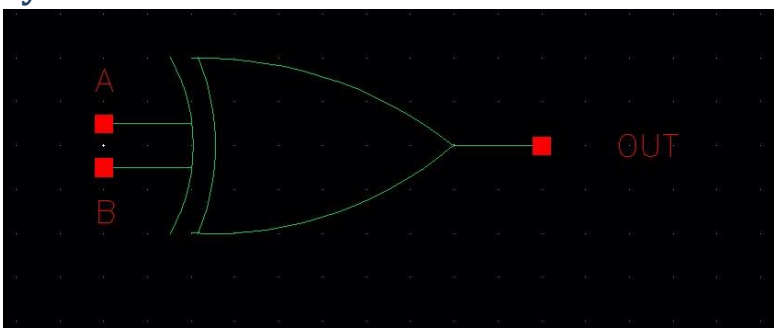


6-XOR:

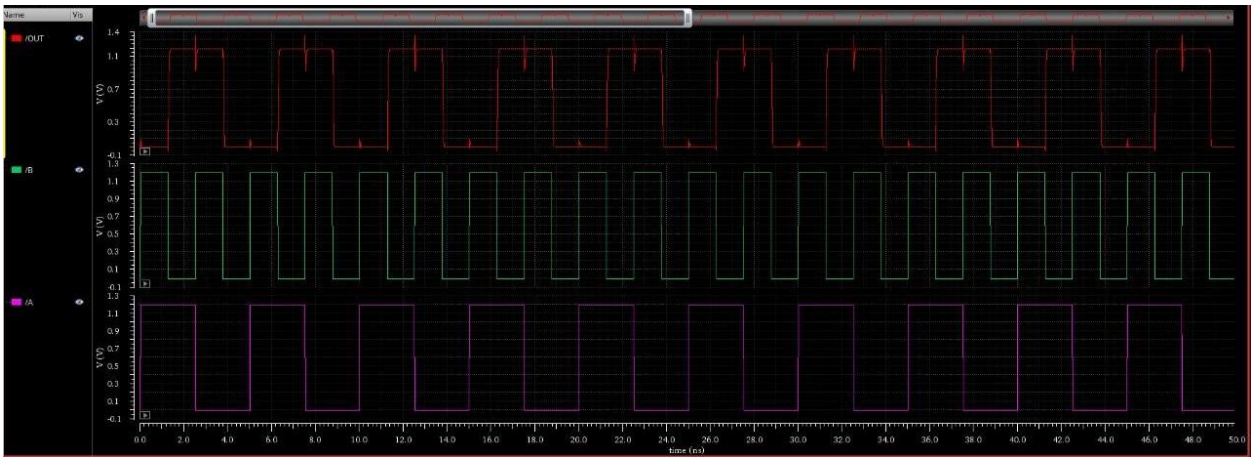
Schematic:



Symbol:

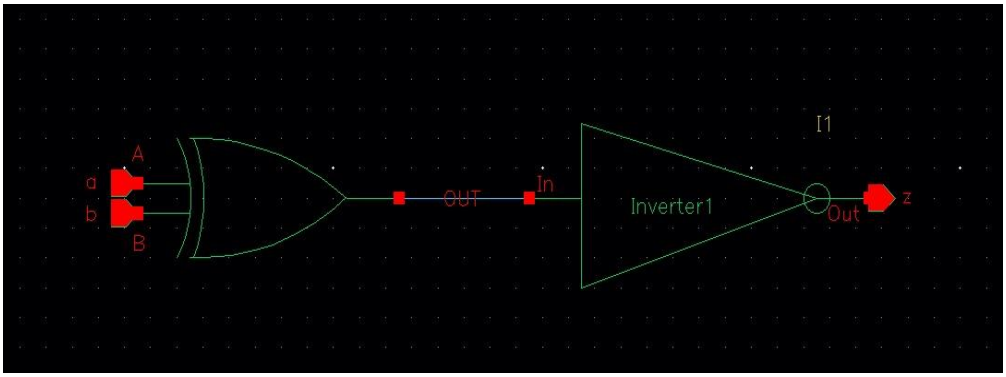


Output waveform:

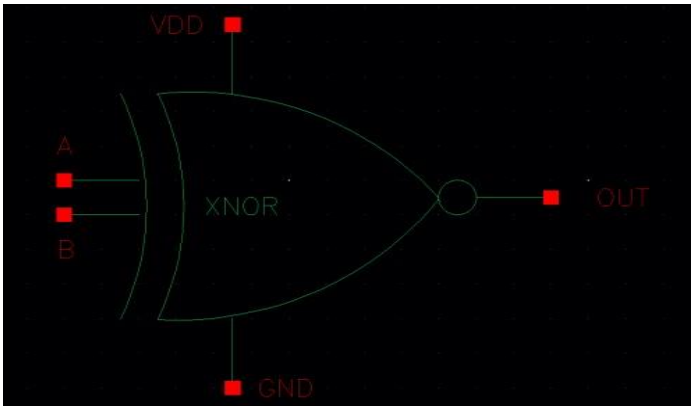


7- XNOR

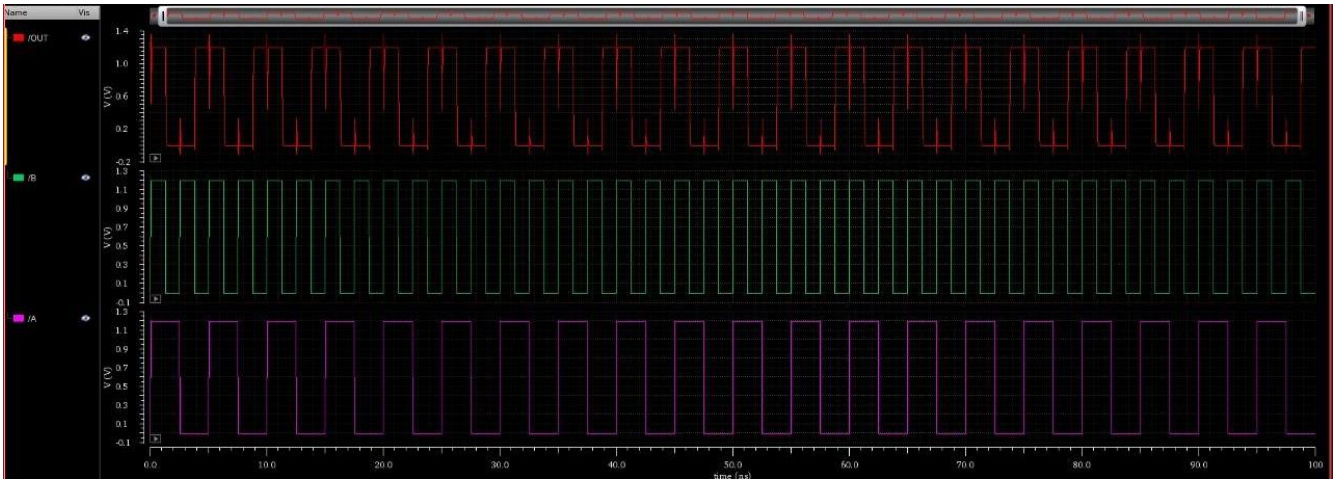
Schematic:



Symbol:

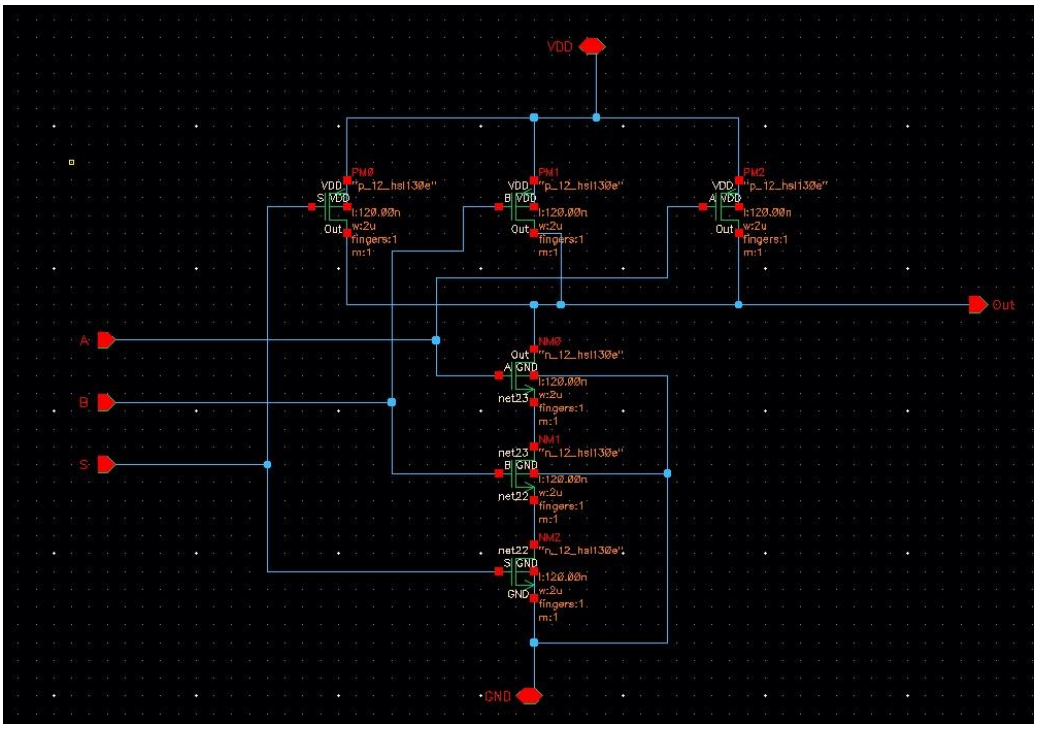


Output waveform:

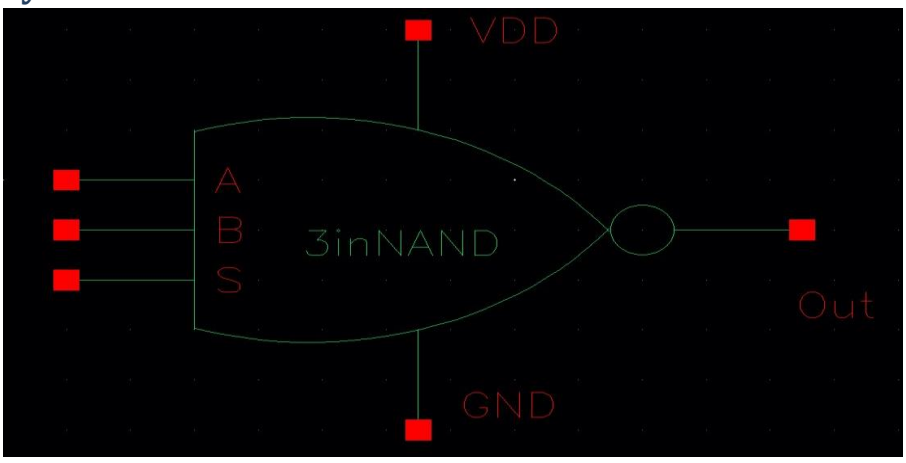


8- 3in NAND (for the 4x1 MUX):

Schematic:

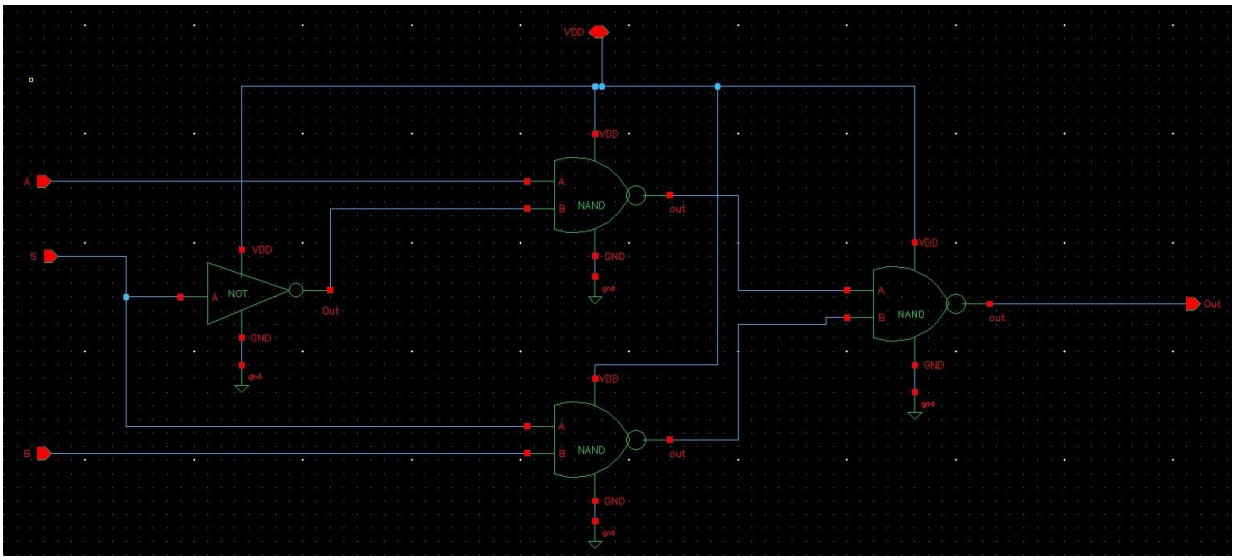


Symbol:

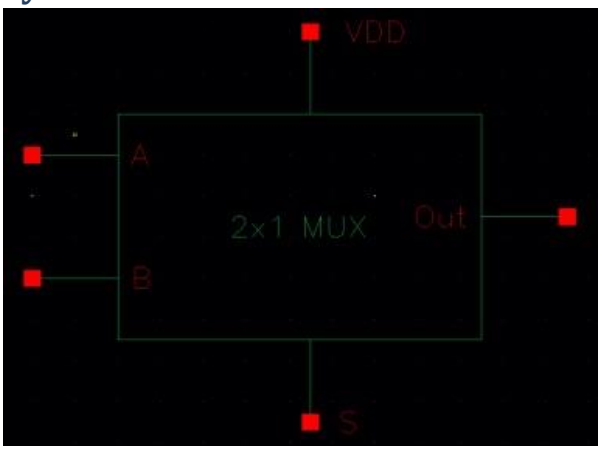


9- 2x1 MUX:

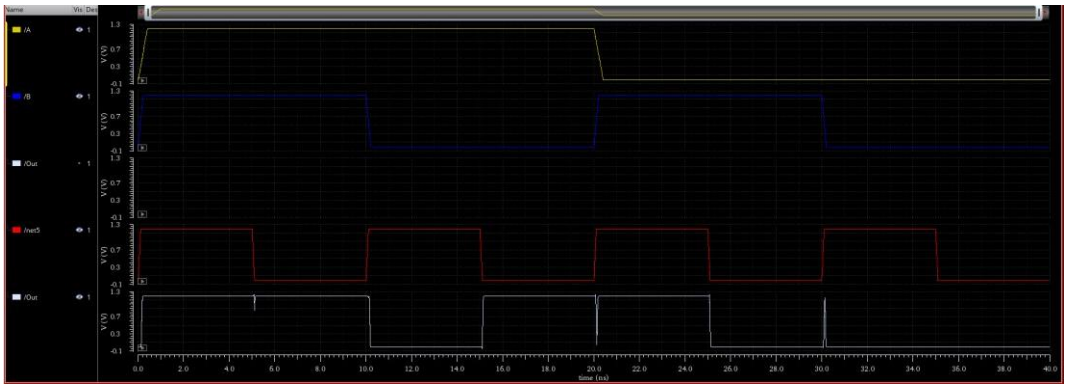
Schematic:



Symbol:

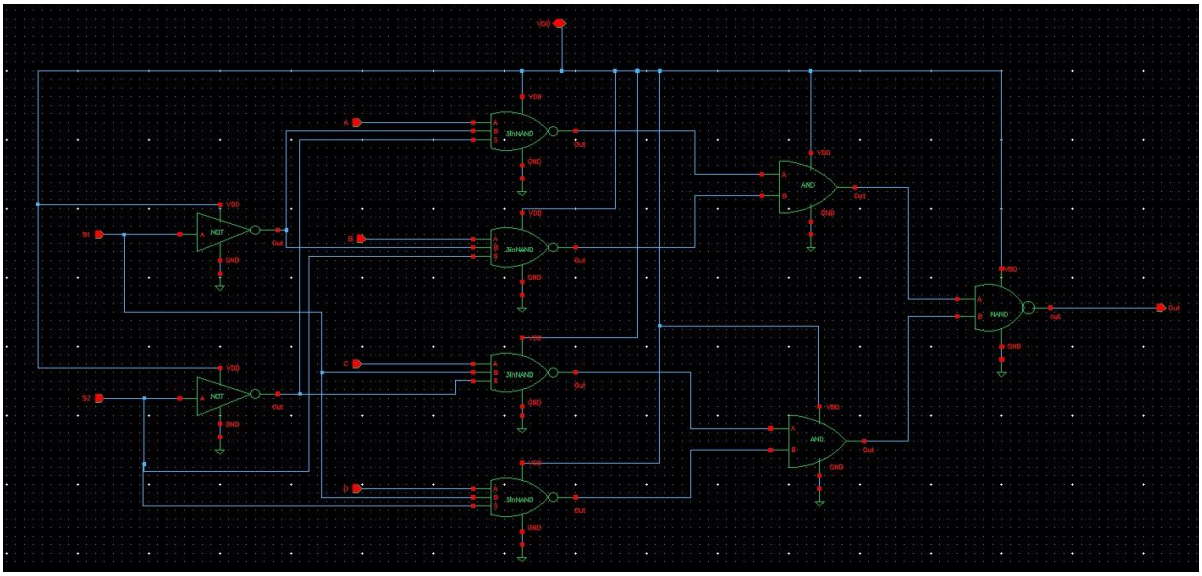


Output waveform:

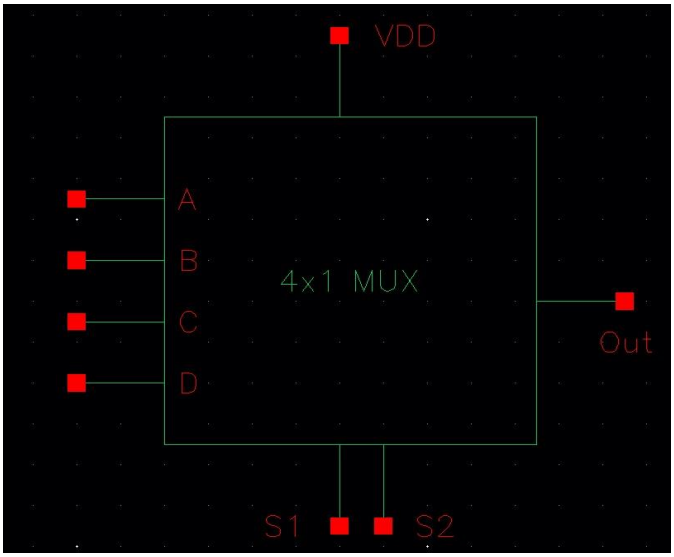


10- **4x1 MUX:**

Schematic:

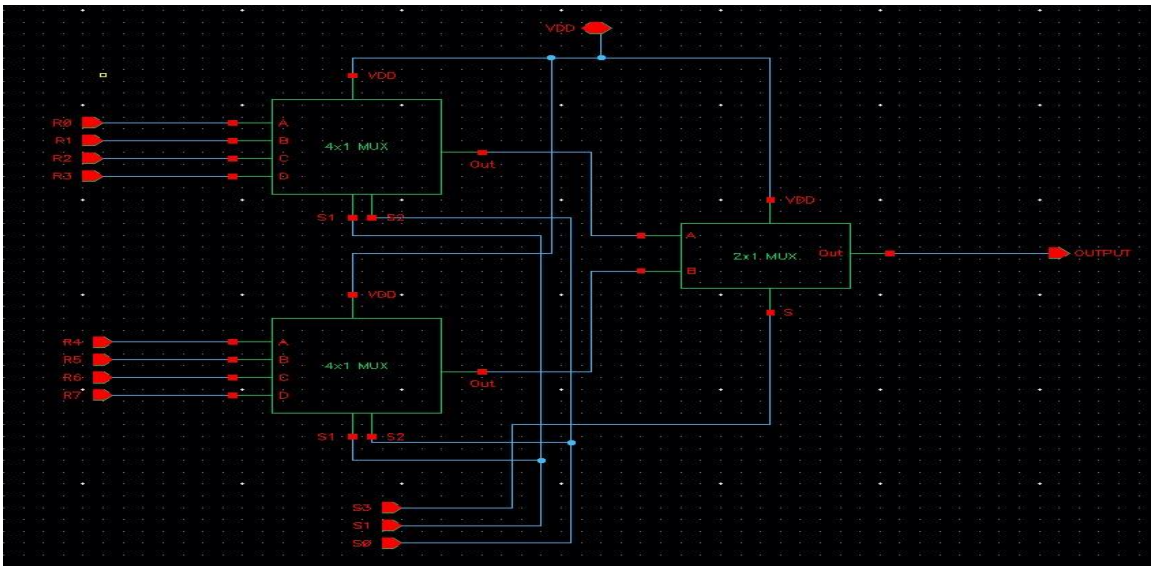


Symbol:

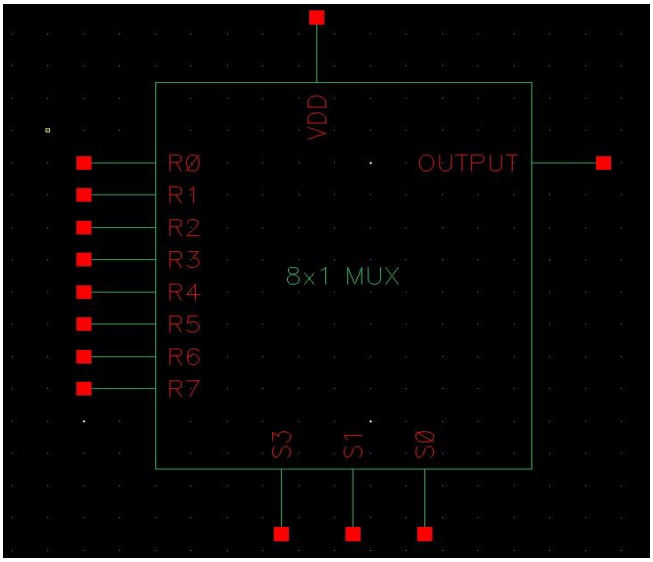


11- 8x1 MUX:

Schematic:

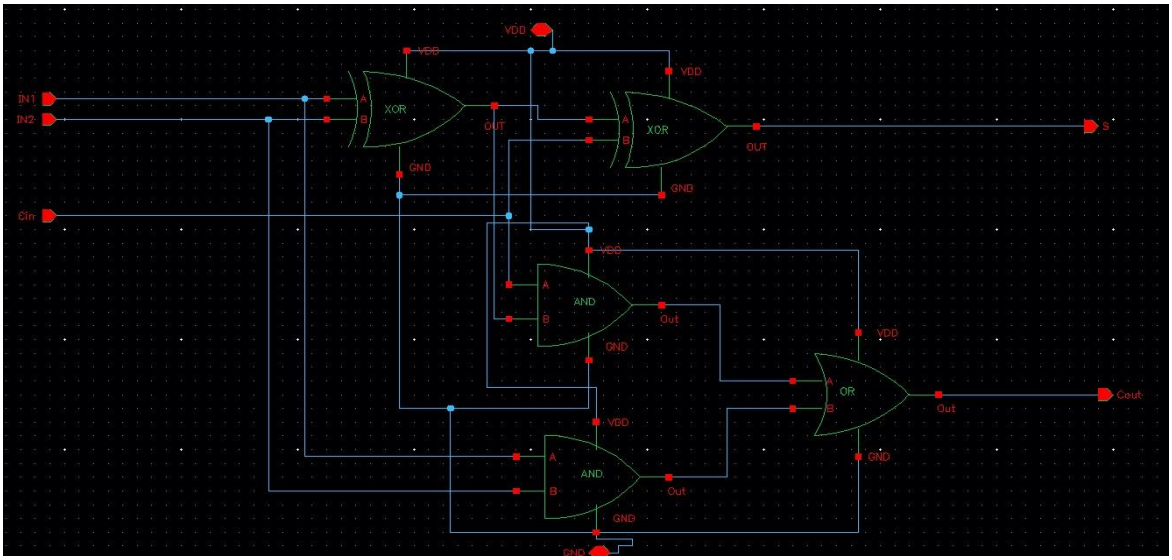


Symbol:

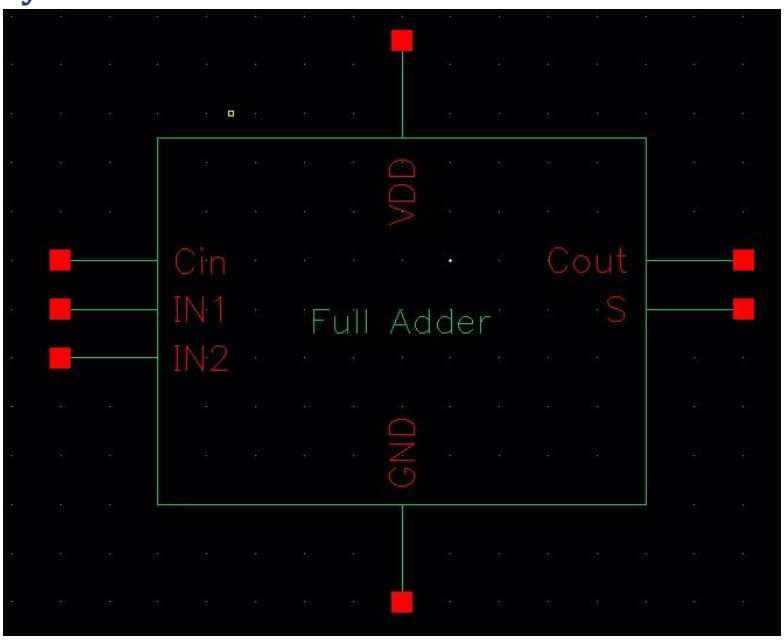


12- Full Adder:

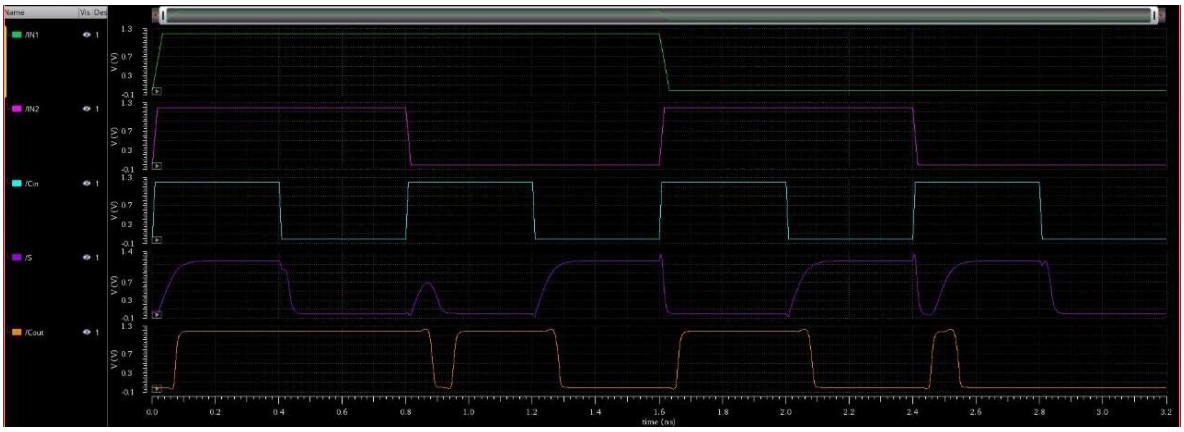
Schematic:



Symbol:



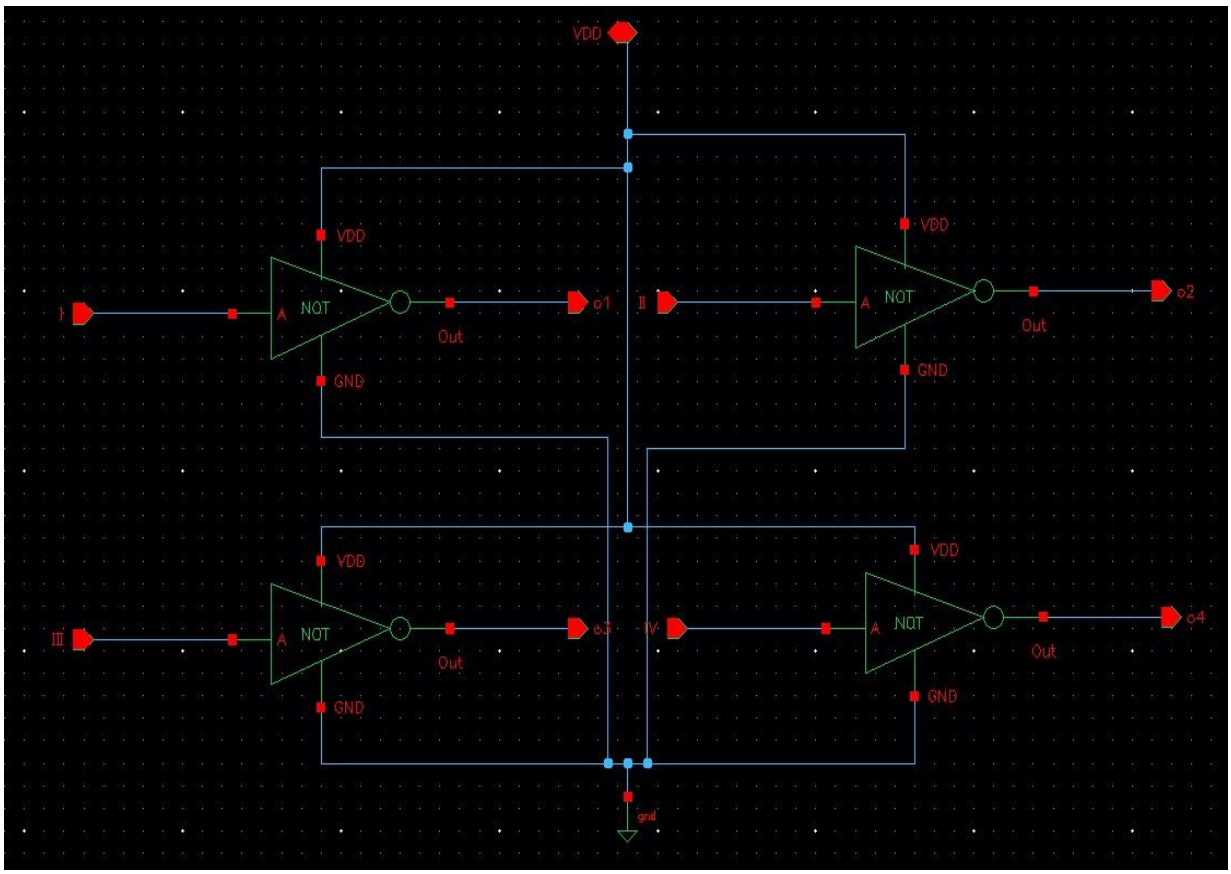
Output waveform:



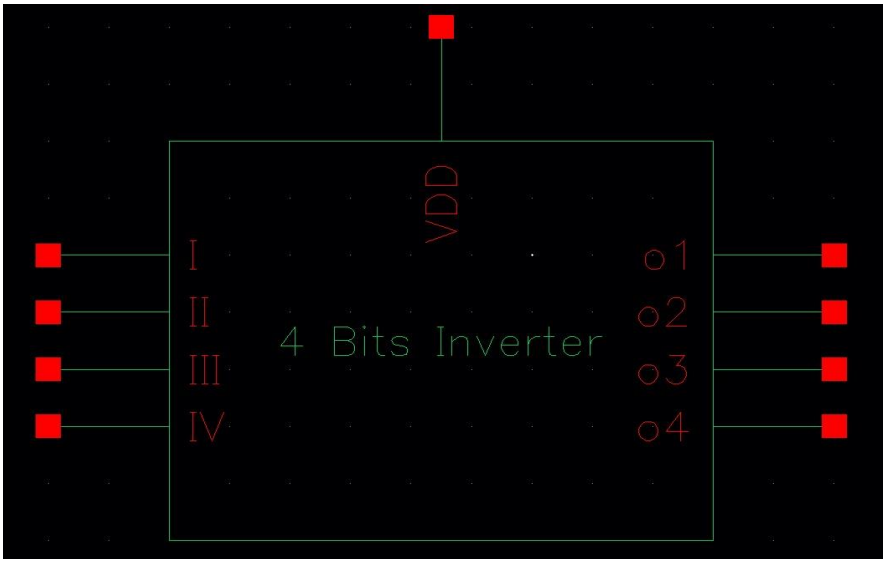
We made every 4 gates into one block to use it easier:

4 bit NOT:

Schematic:

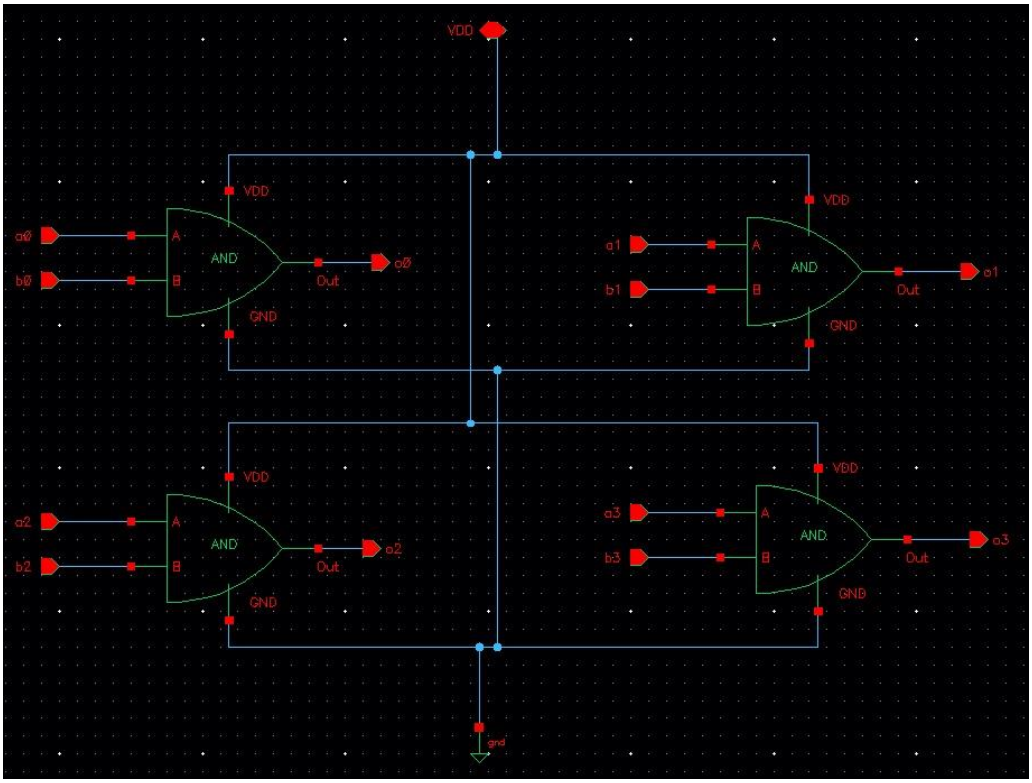


Symbol:

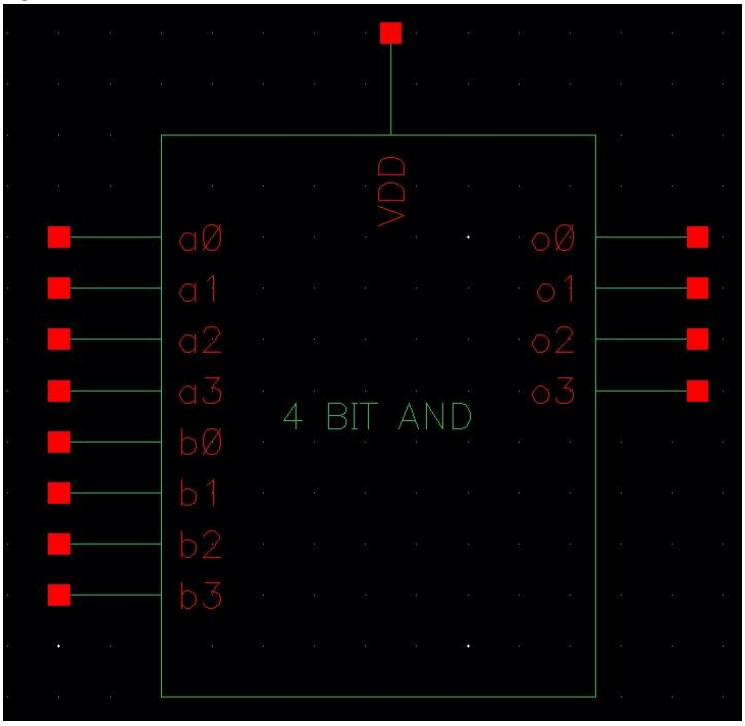


4 bit And:

Schematic:

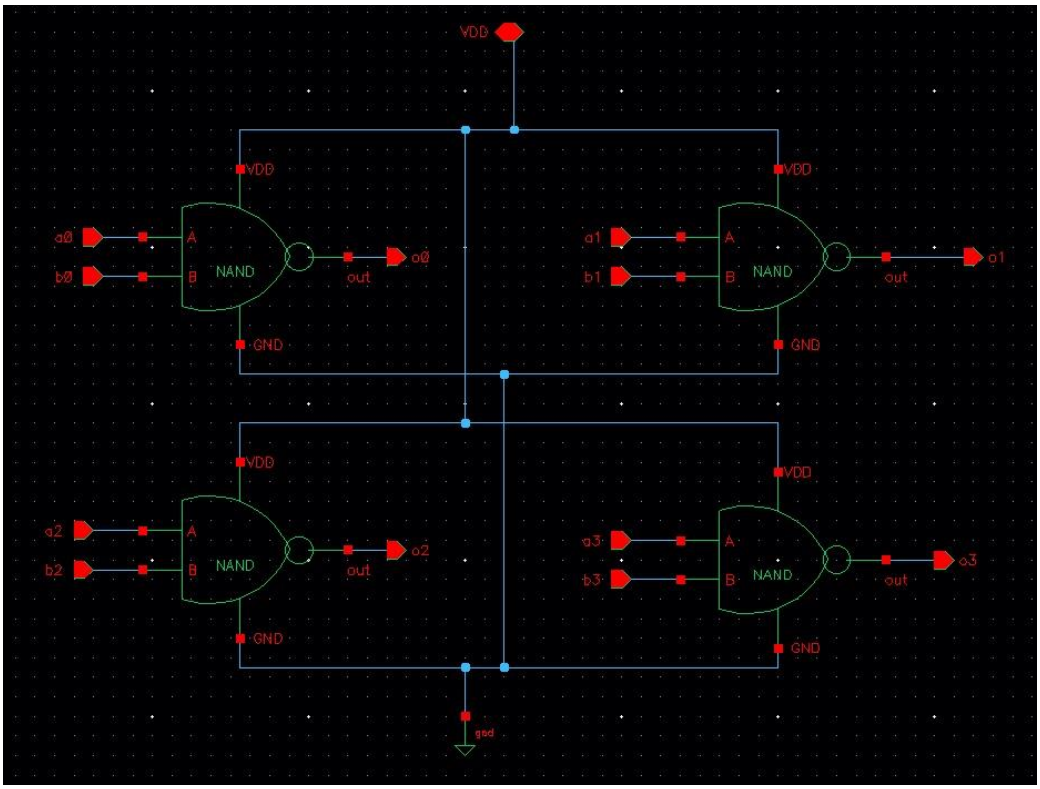


Symbol:

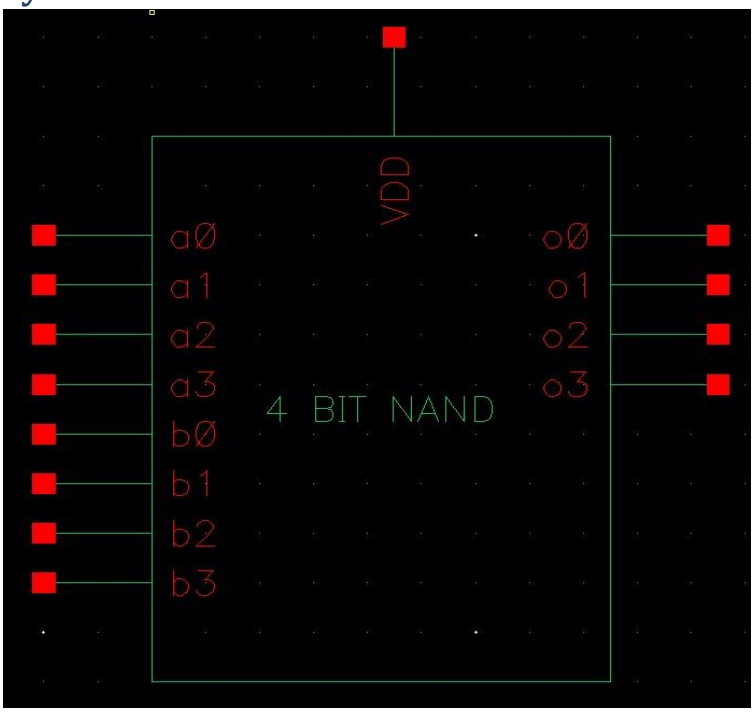


4 bit Nand:

Schematic:

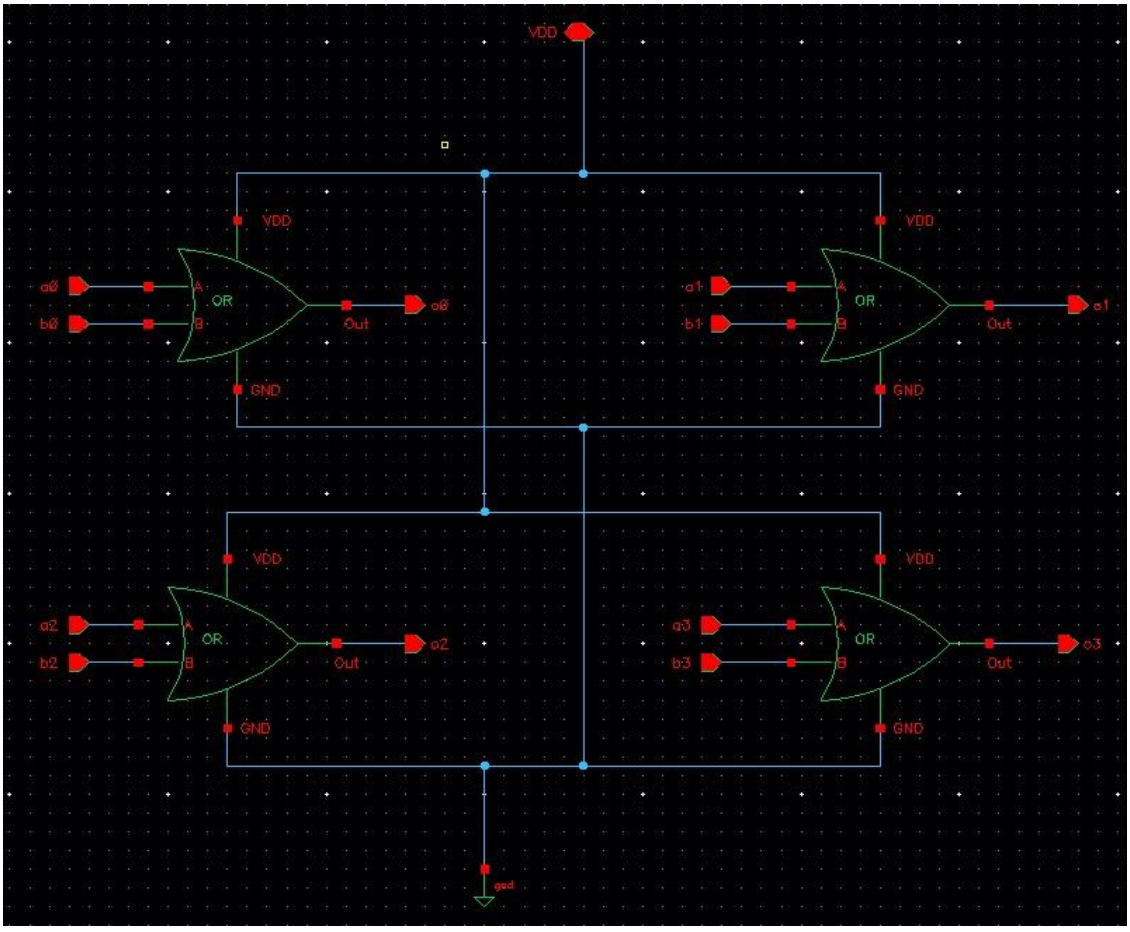


Symbol:

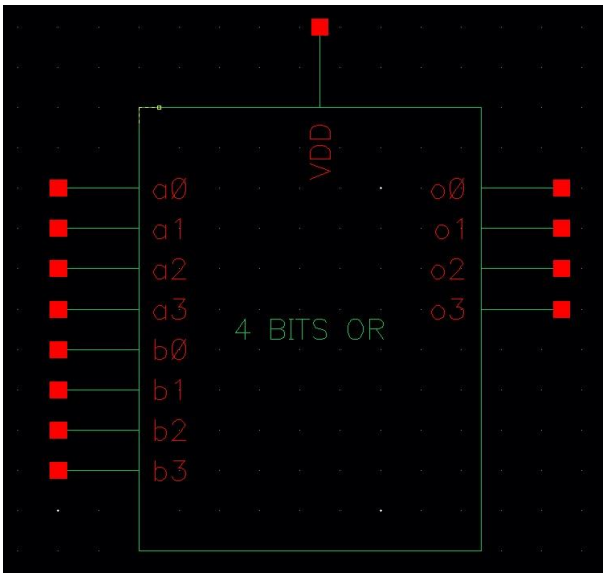


4 bit OR:

Schematic:

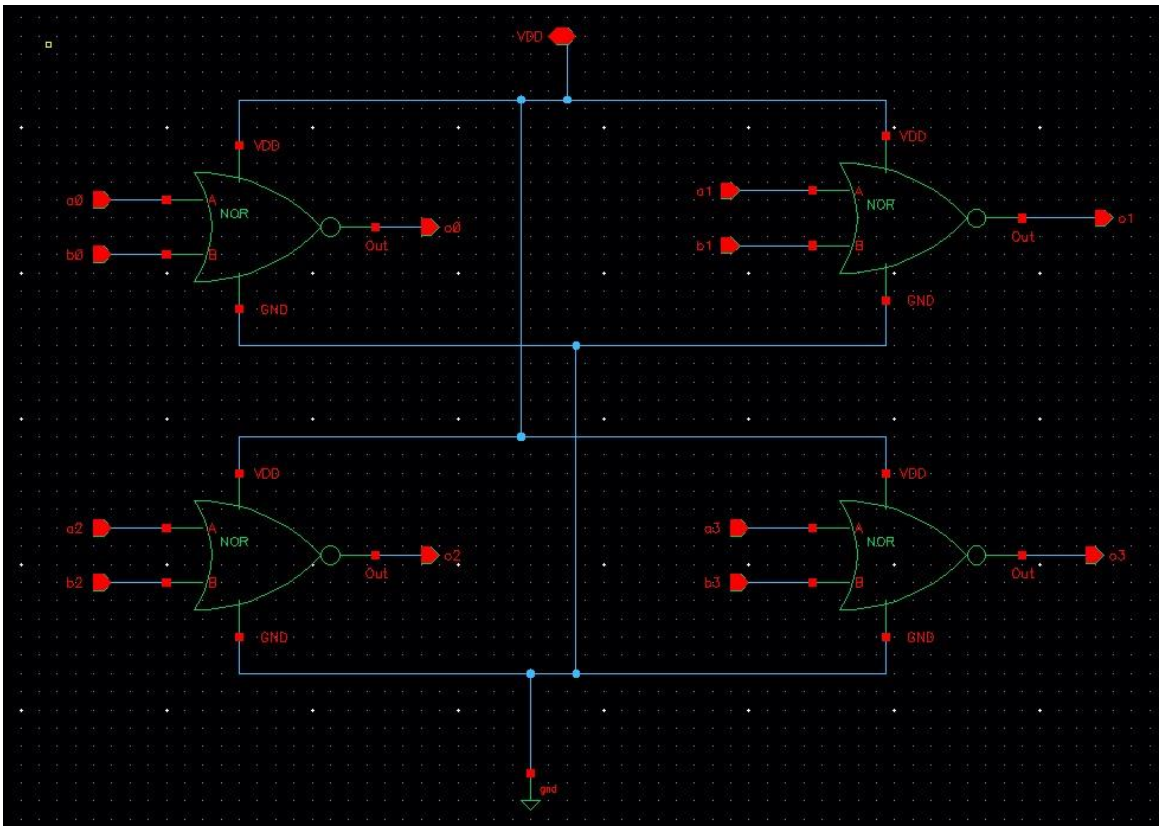


Symbol:

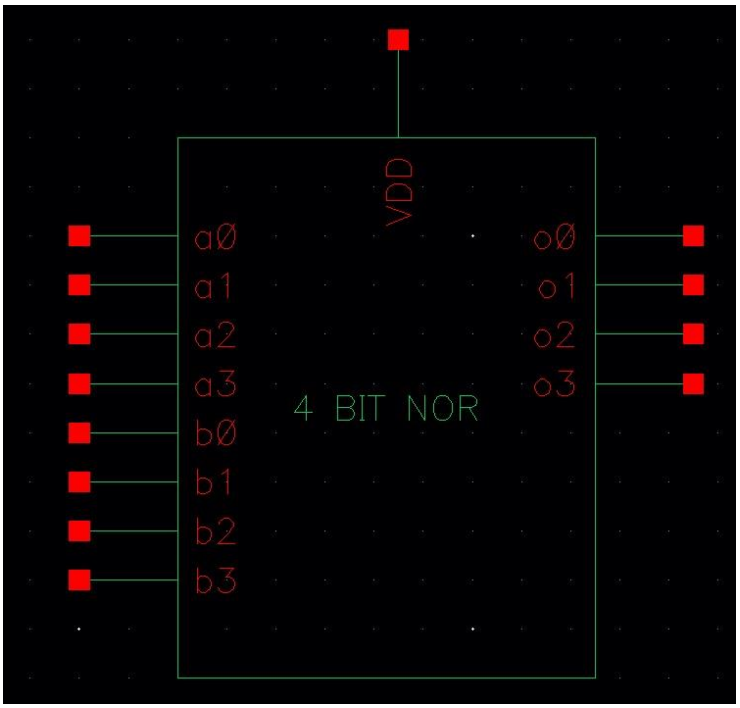


4 bit NOR:

Schematic:

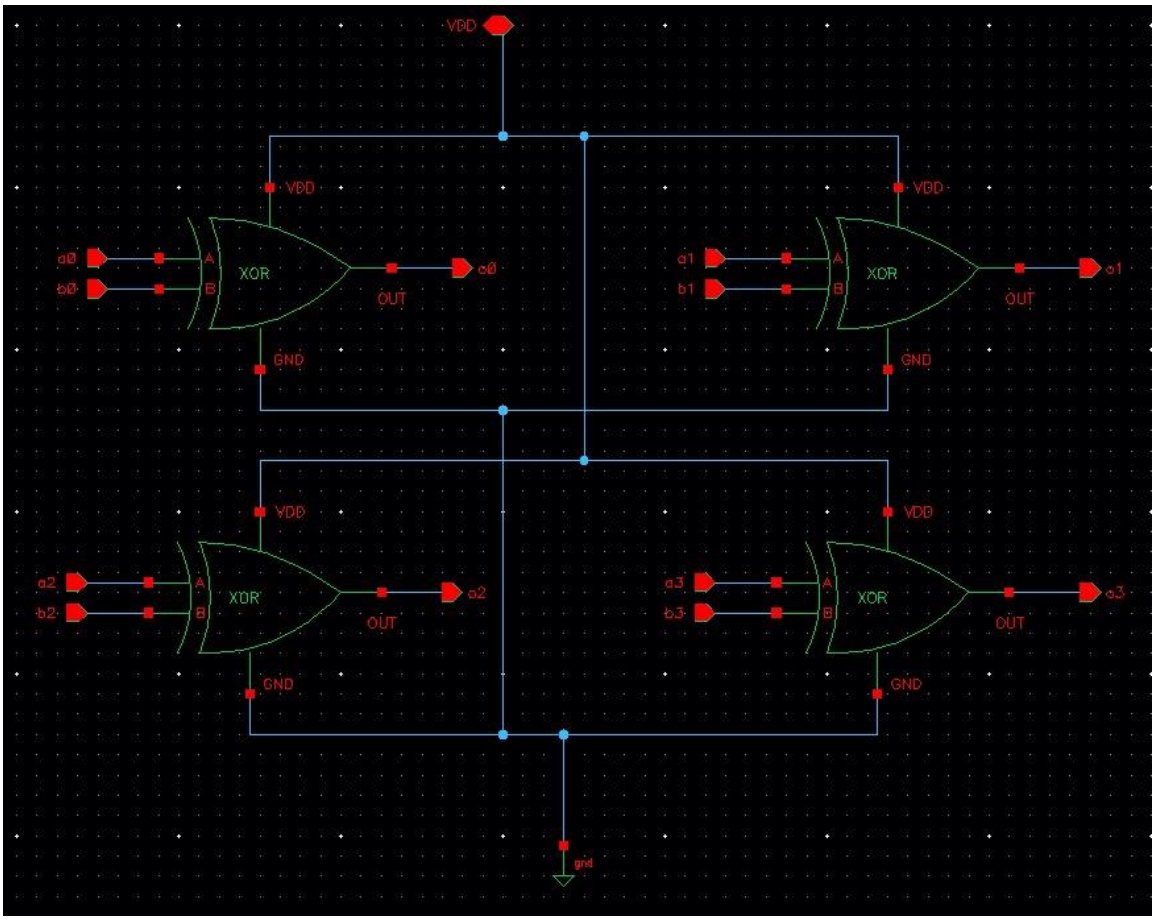


Symbol:

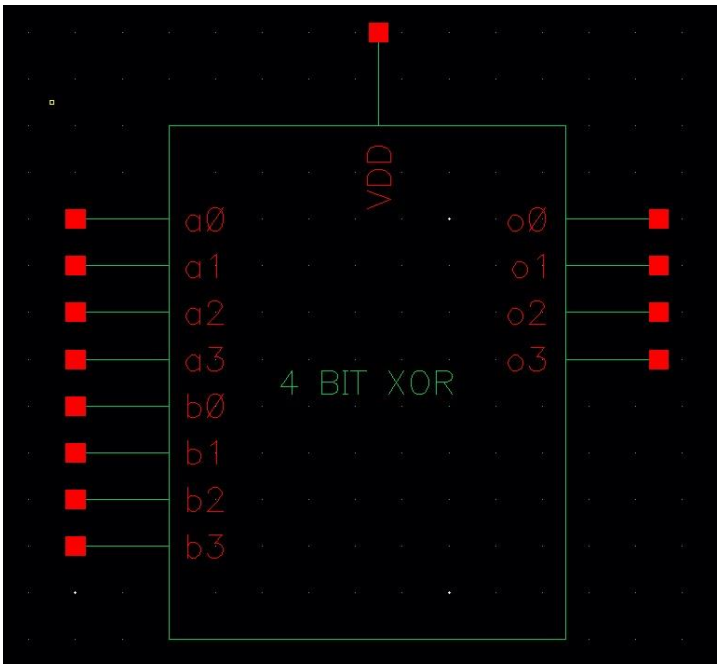


4 bit XOR:

Schematic:

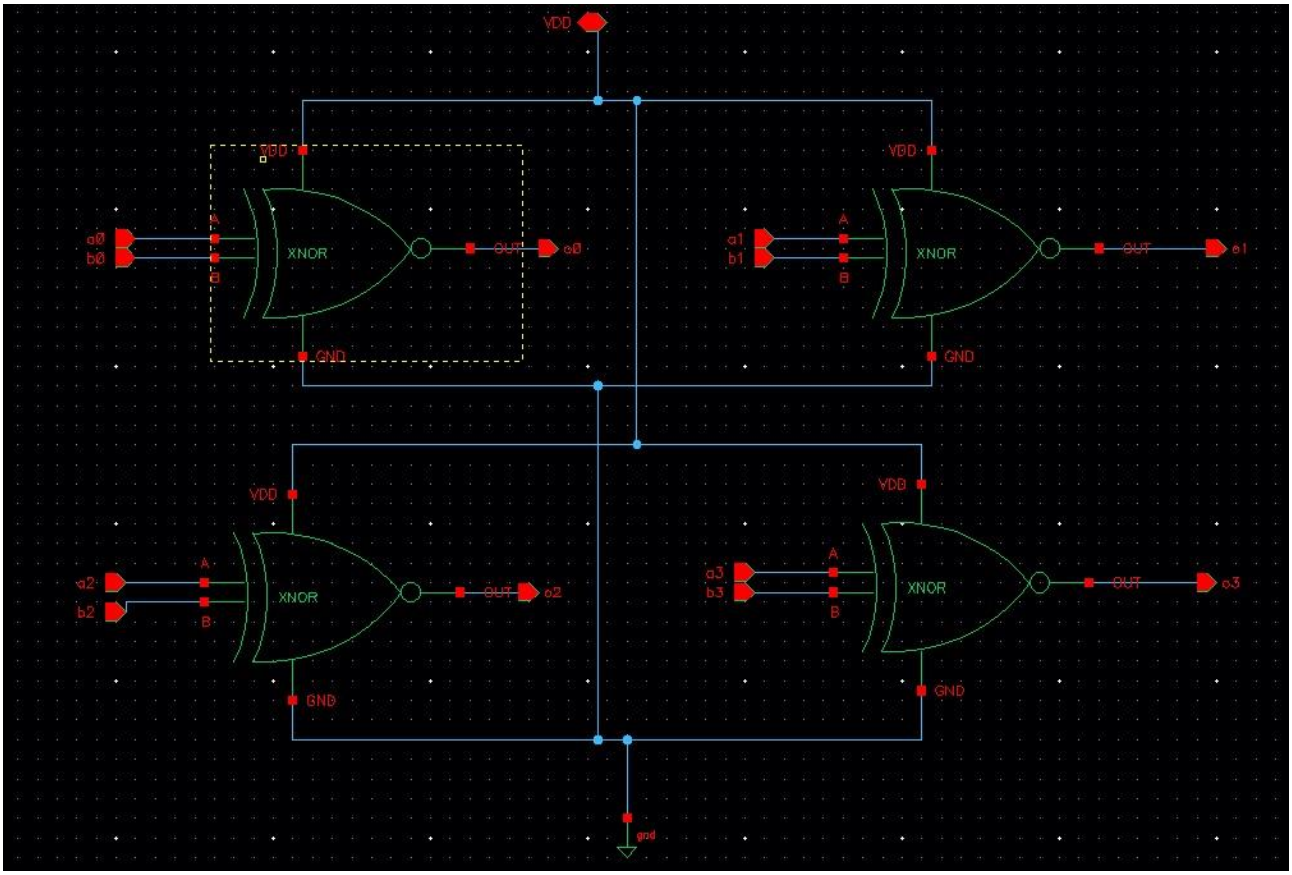


Symbol:

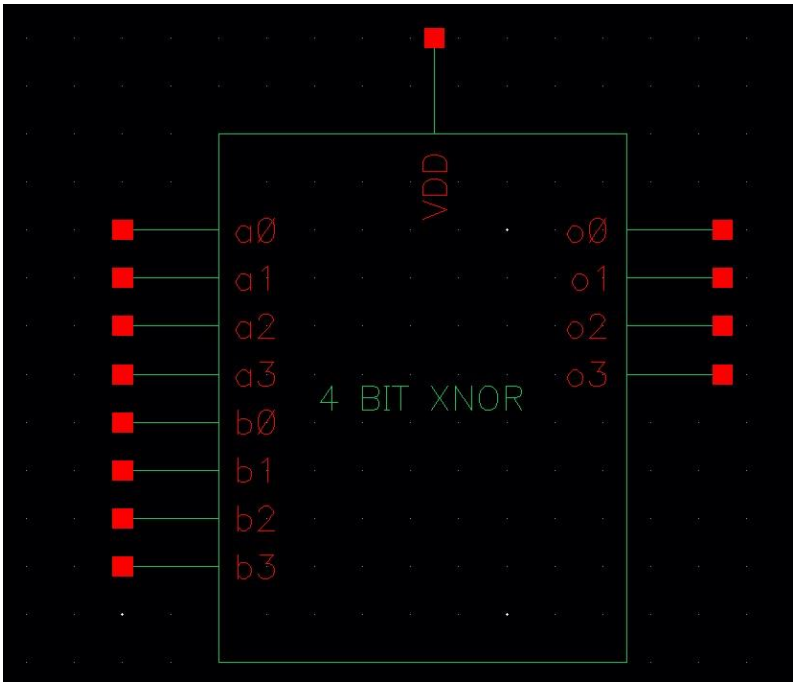


4 bit XNOR:

Schematic:

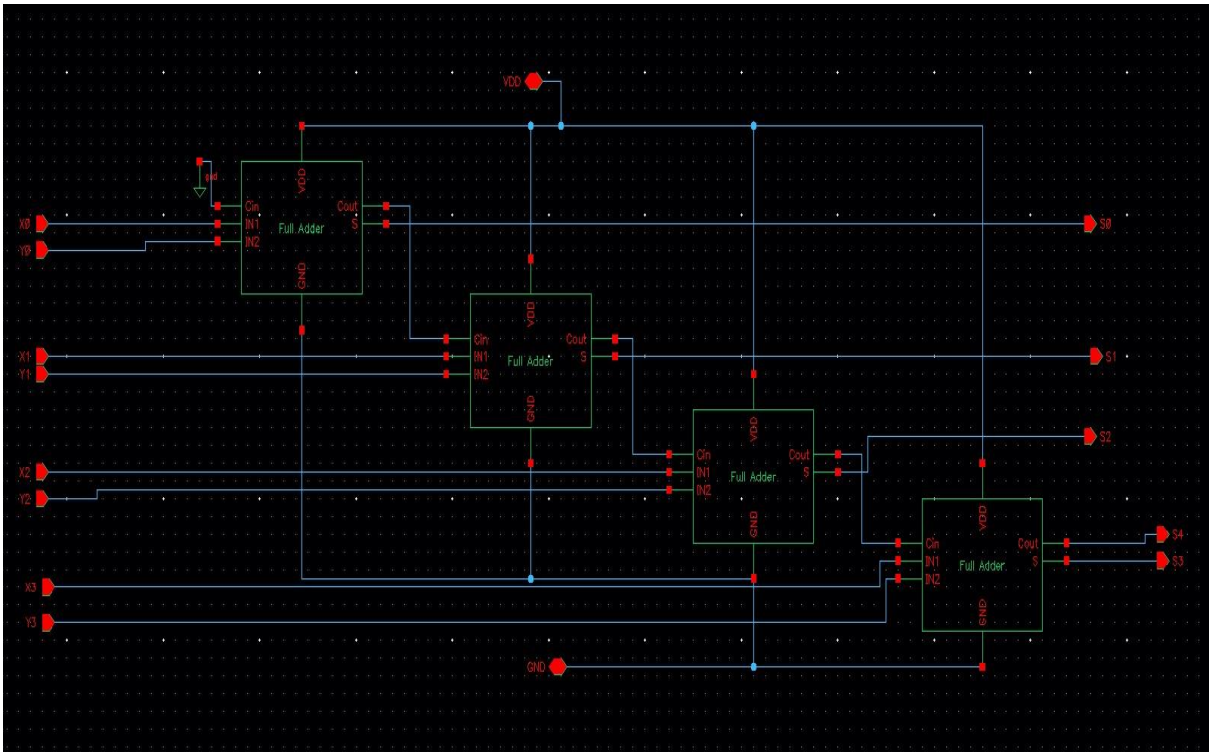


Symbol:

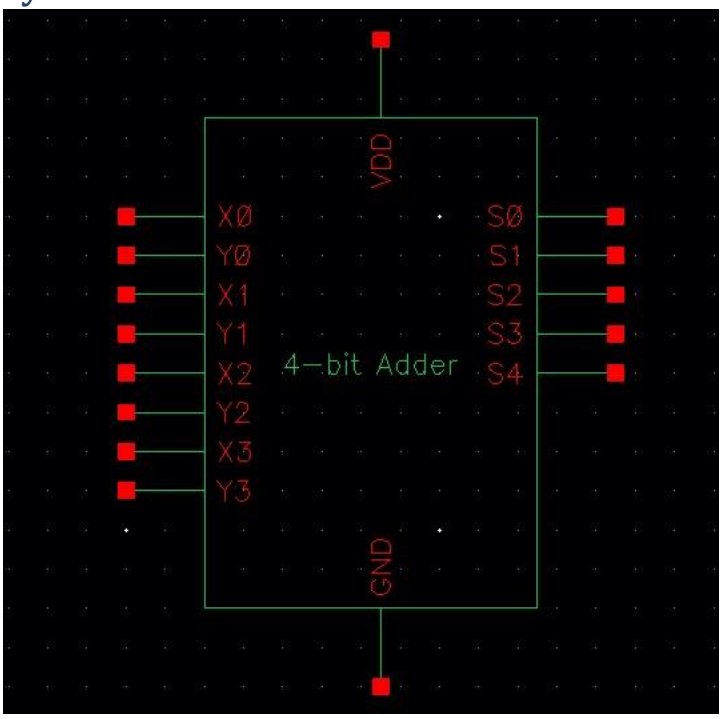


4 bit Adder:

Schematic:

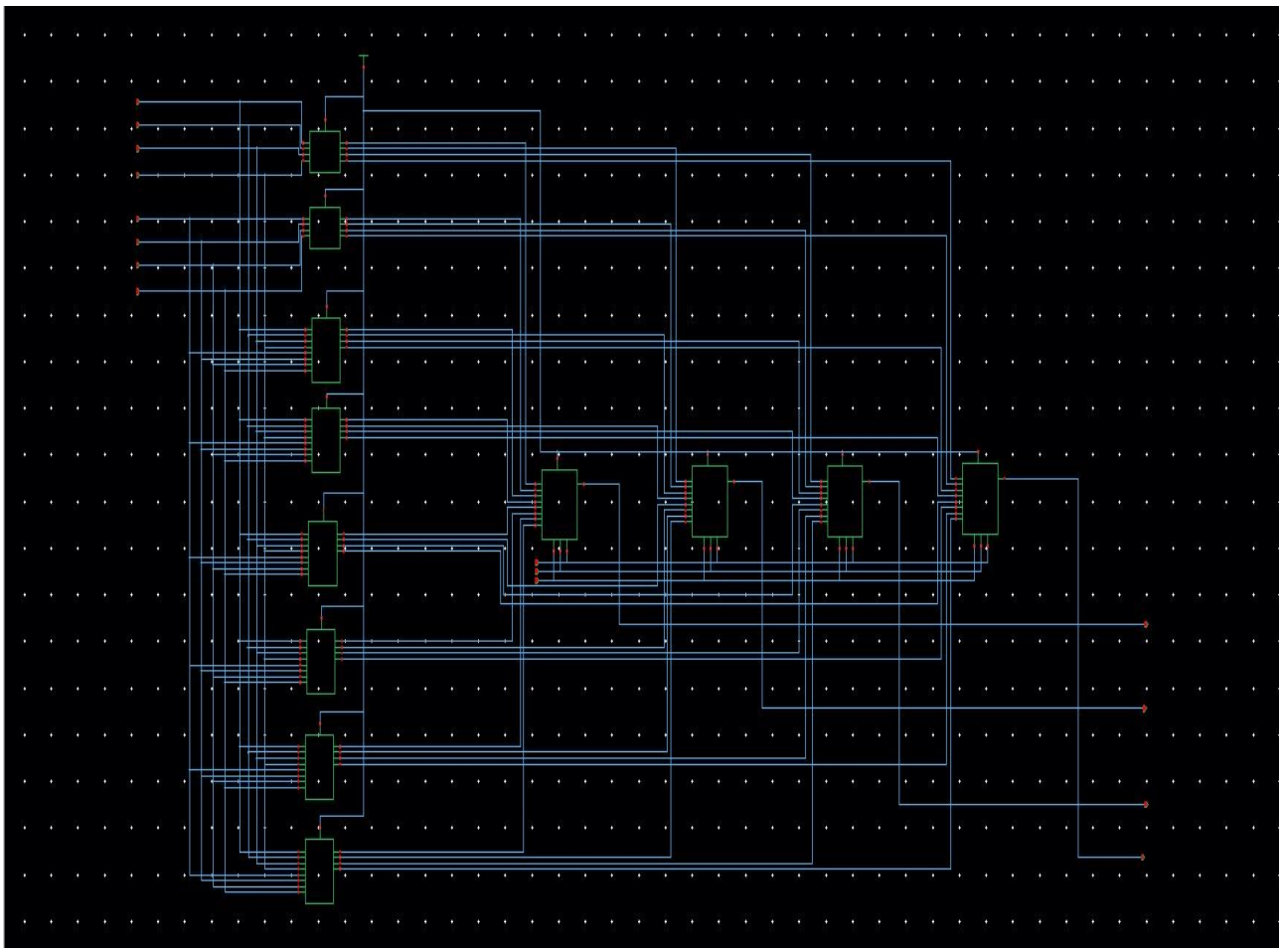


Symbol:

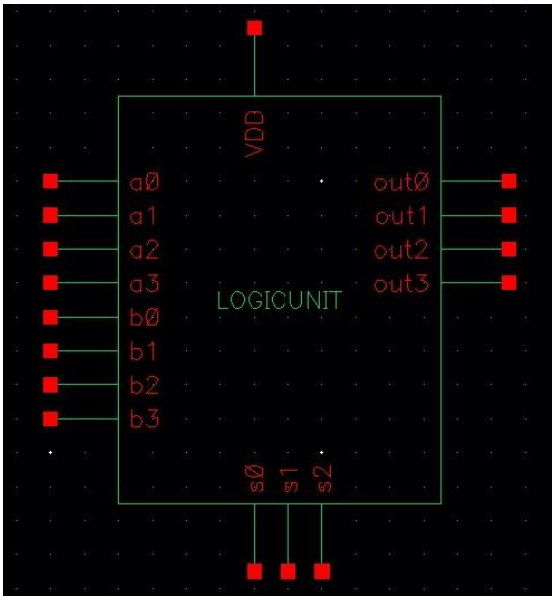


Logic Unit Full:

Schematic:

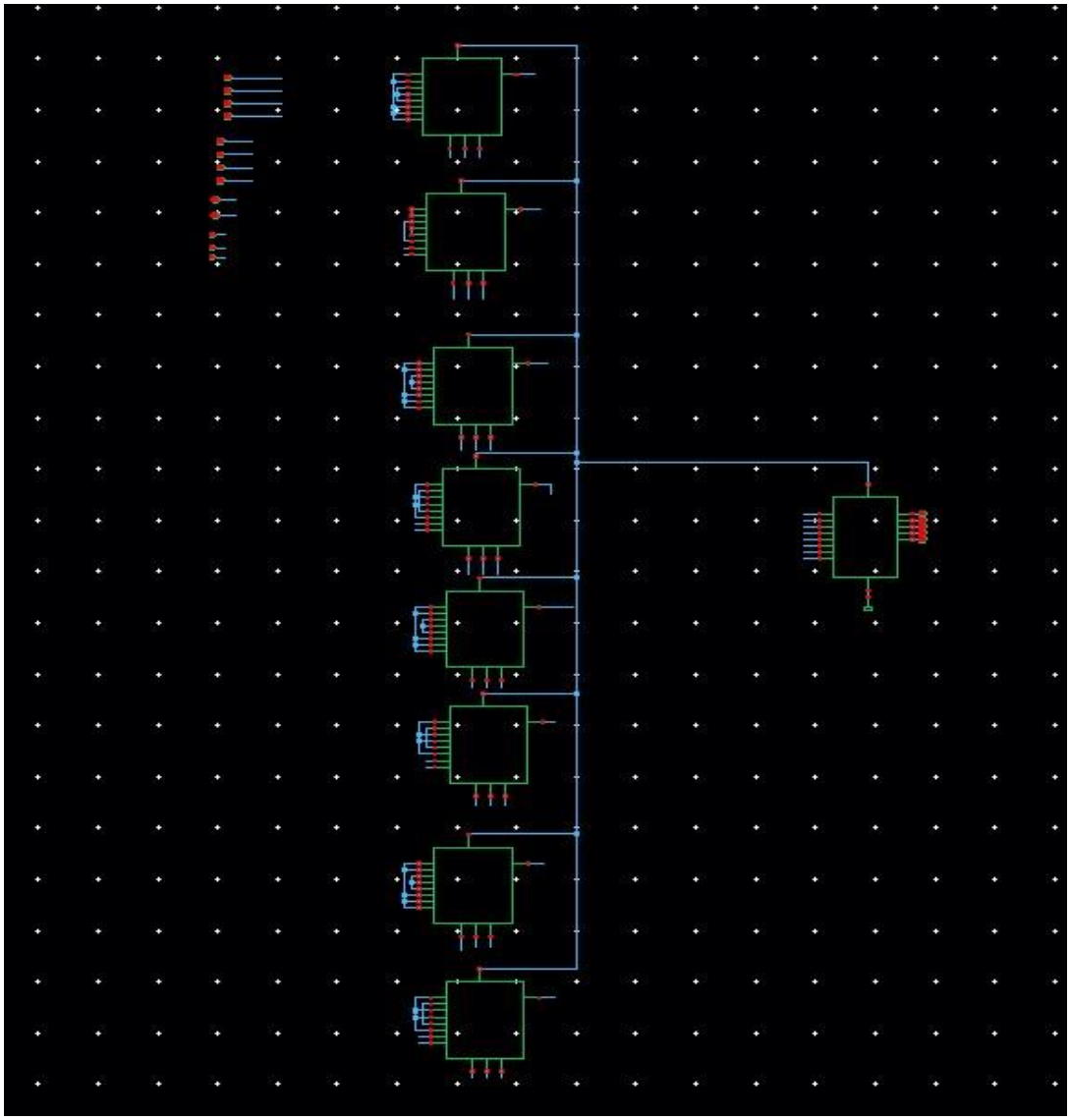


Symbol:

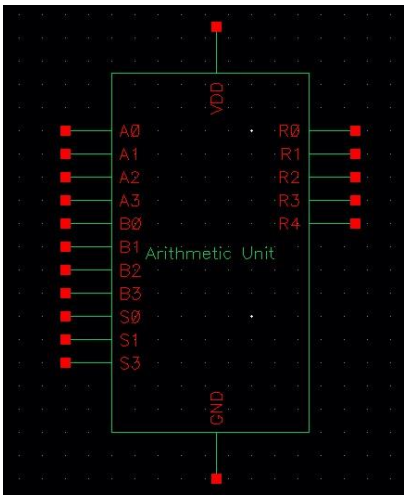


Arthmetic Unit Full:

Schematic:

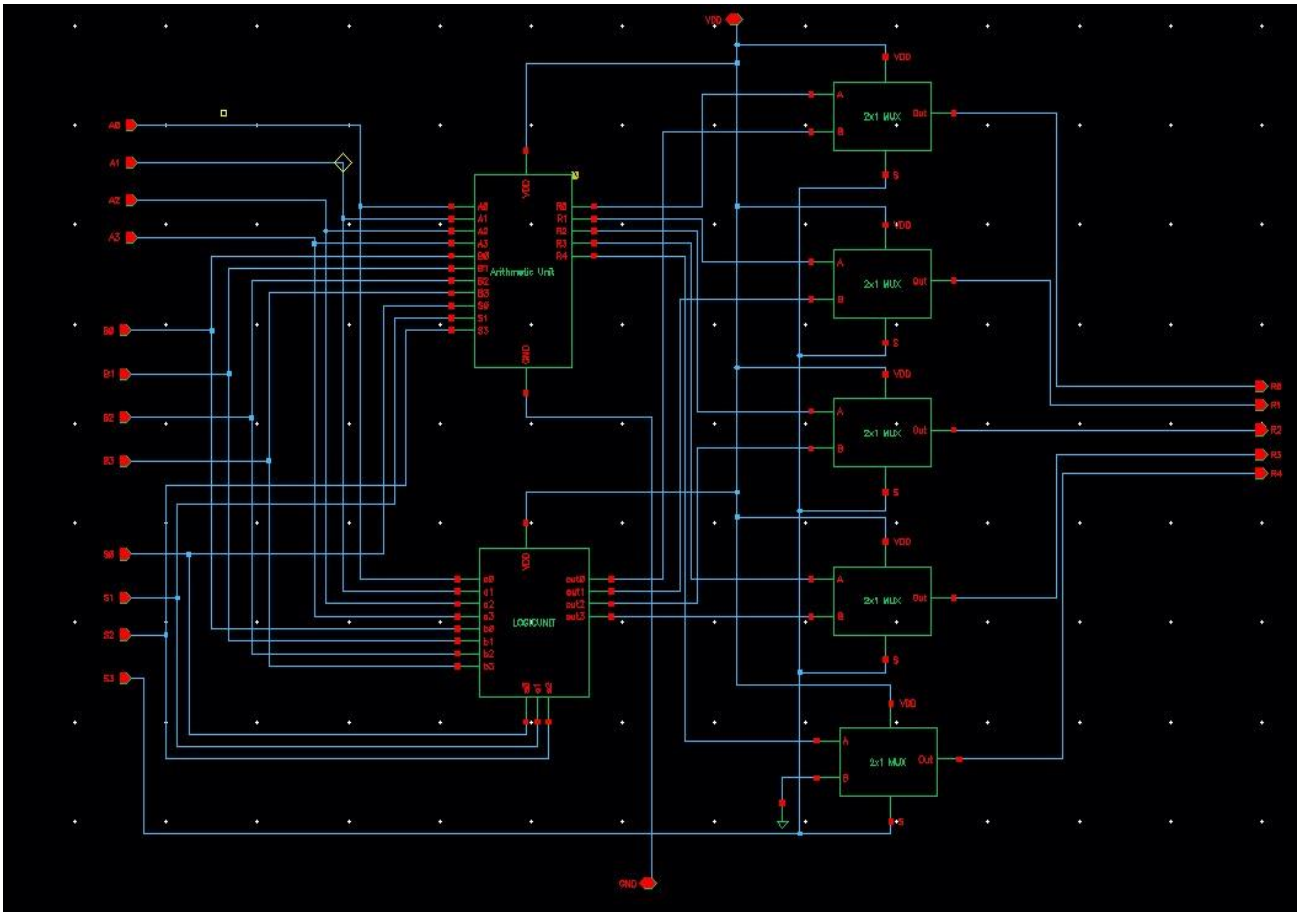


Symbol:

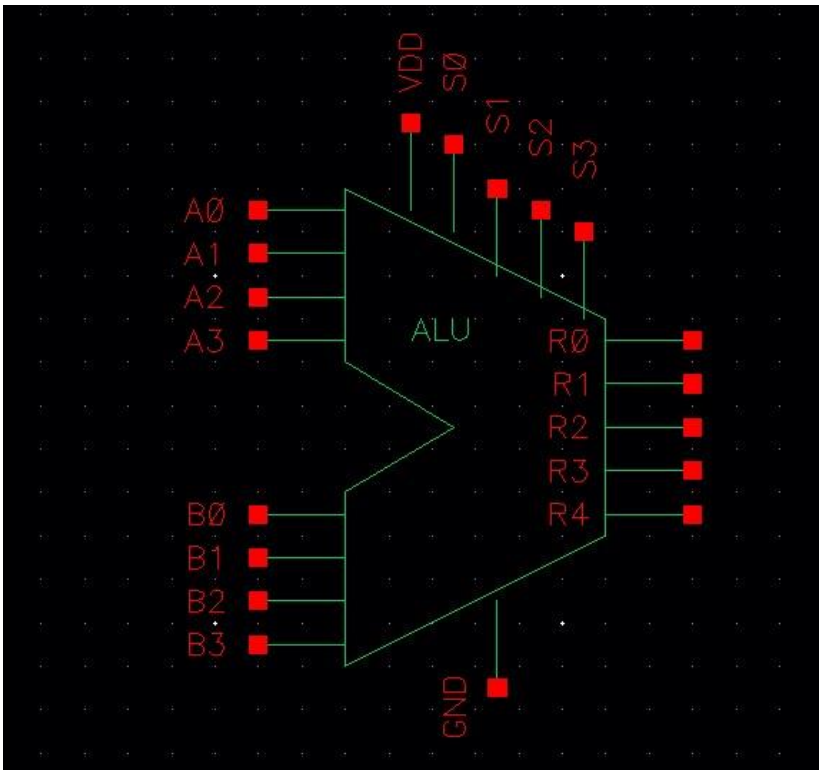


ALU Unit :

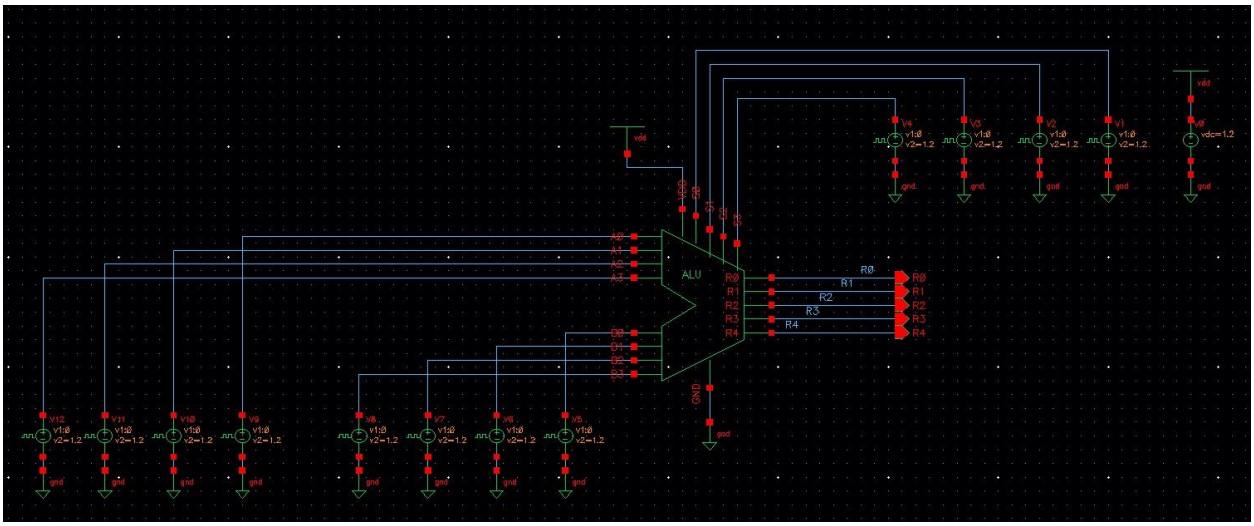
Schematic:



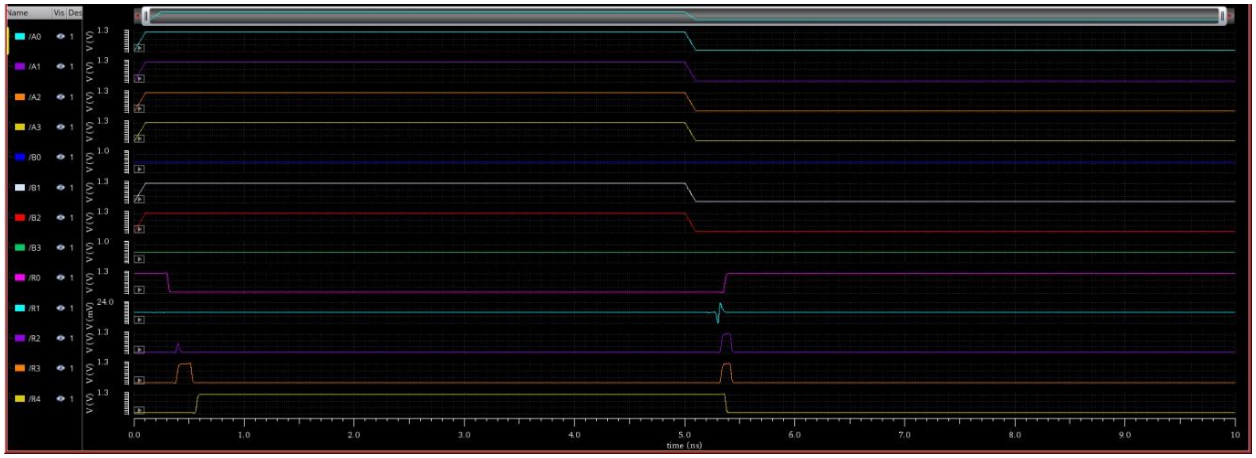
Symbol:



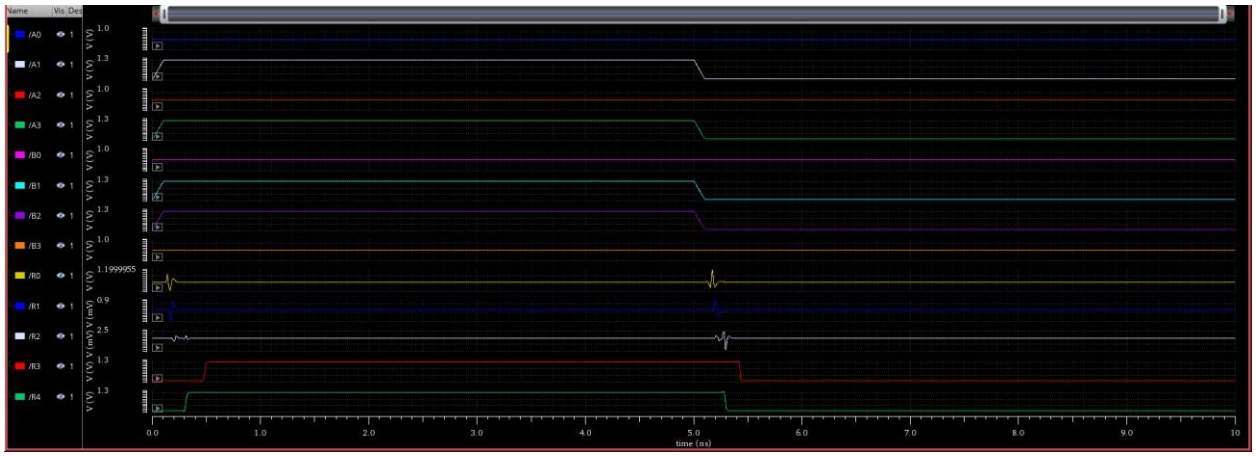
TEST BENCH ON EVERY OPERATION:



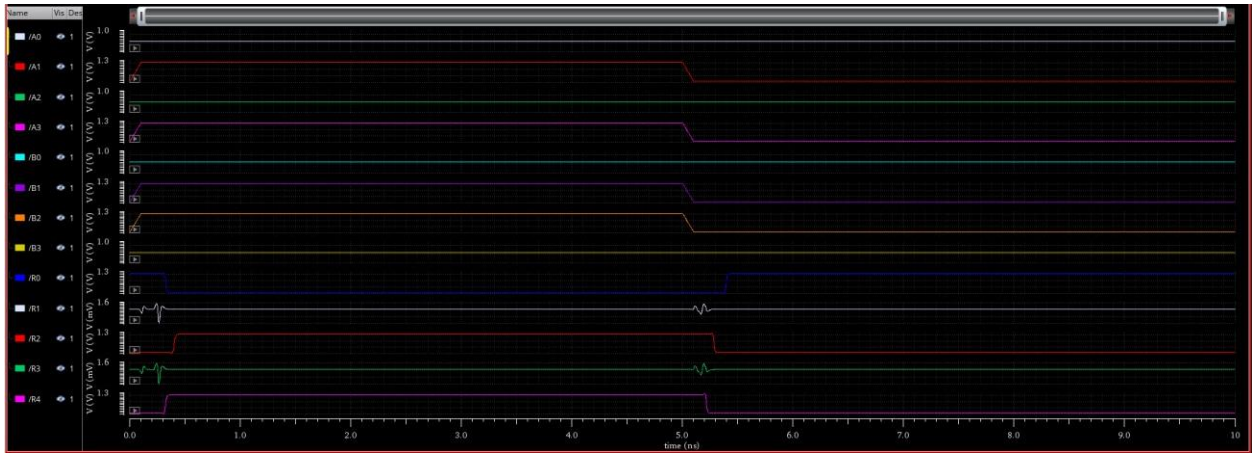
1) INPUTS: A (1111) Operation: 0000 Increment A Result 10000



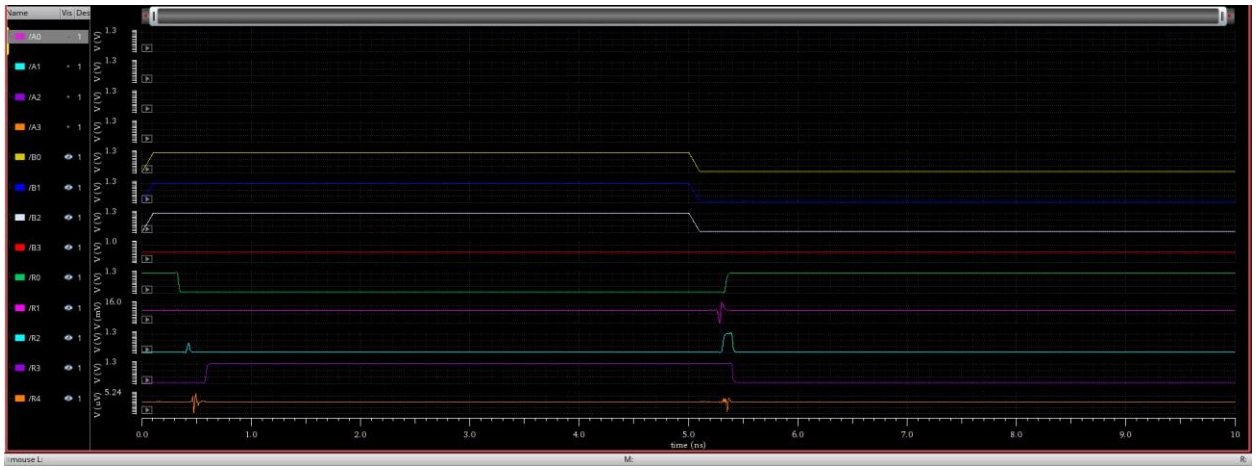
2) INPUTS: A (1010) Operation: 0001 Decrement A Result 1001



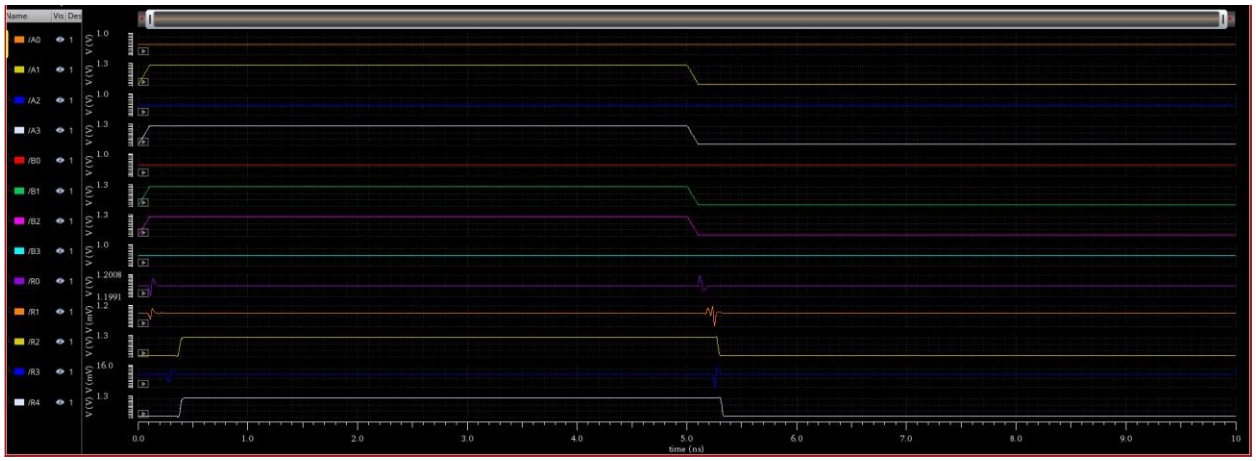
3) INPUTS: A (1010) Operation: 0010 Multiply A *2 Result 10100



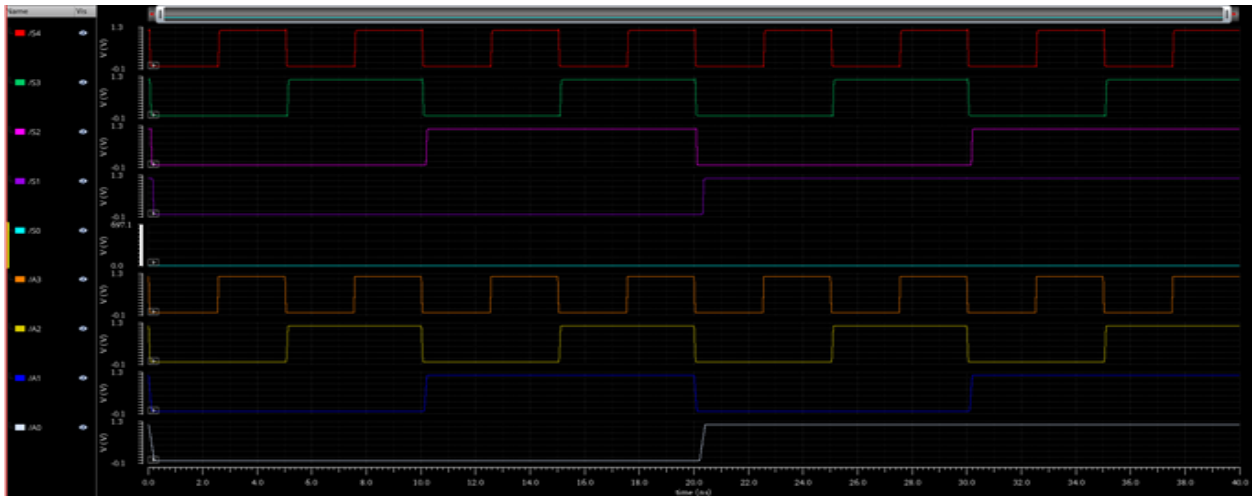
4) INPUTS: B (0111) Operation: 0011 Increment B Result: 01000



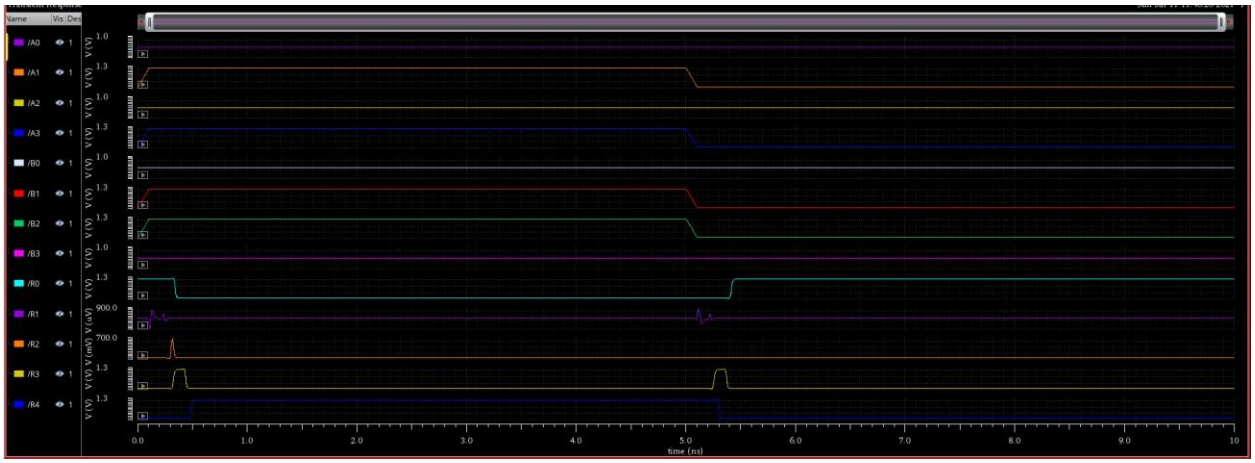
5) INPUTS: B (0110) Operation: 0100 Decrment B Result: 0101



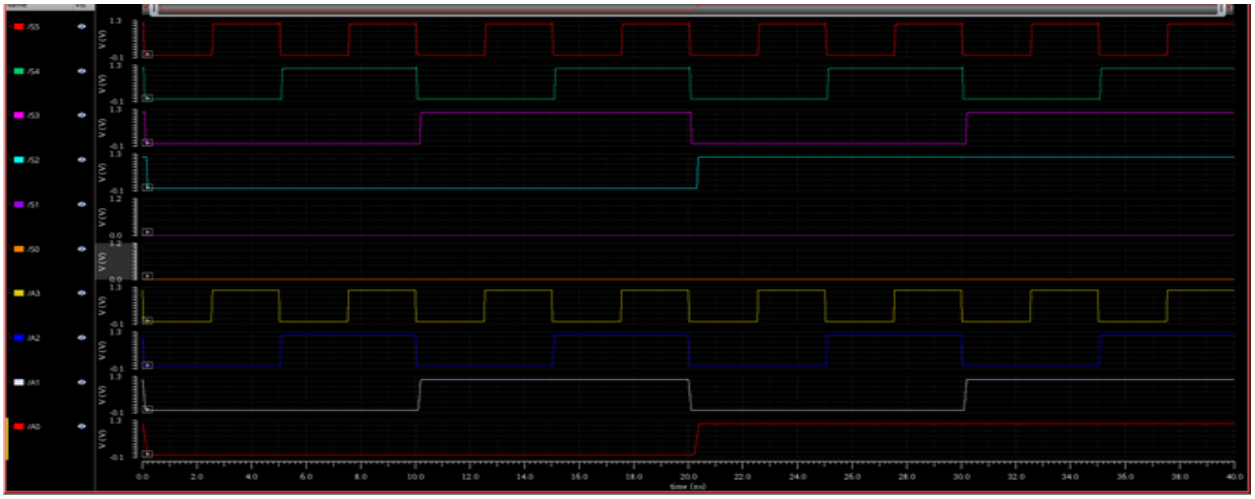
6) INPUTS: B (0110) Operation: 0101 Multiply B *2 Result 1100



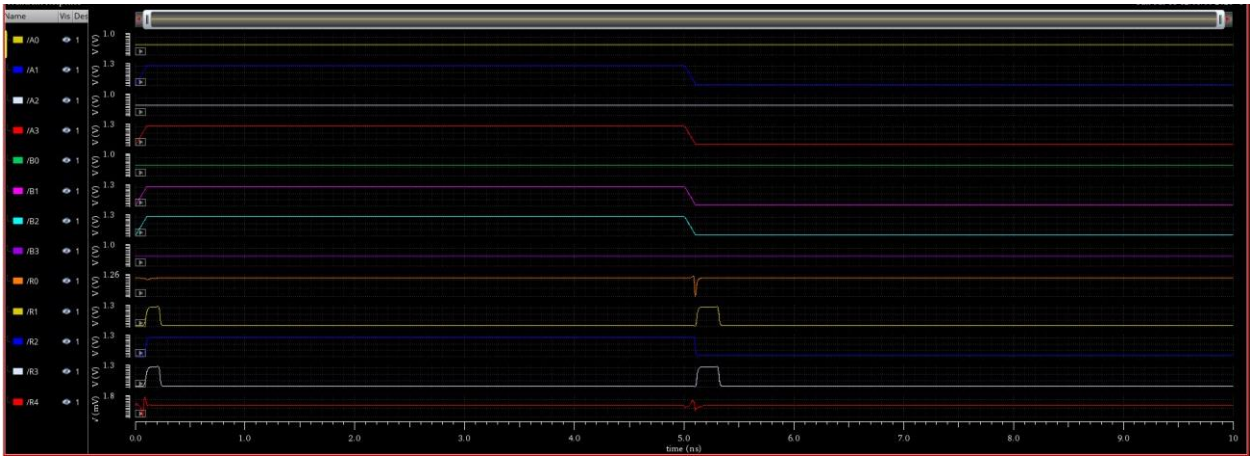
7) INPUTS: A (1010) B(0110) Operation: 0110 Add A+B Result 10000



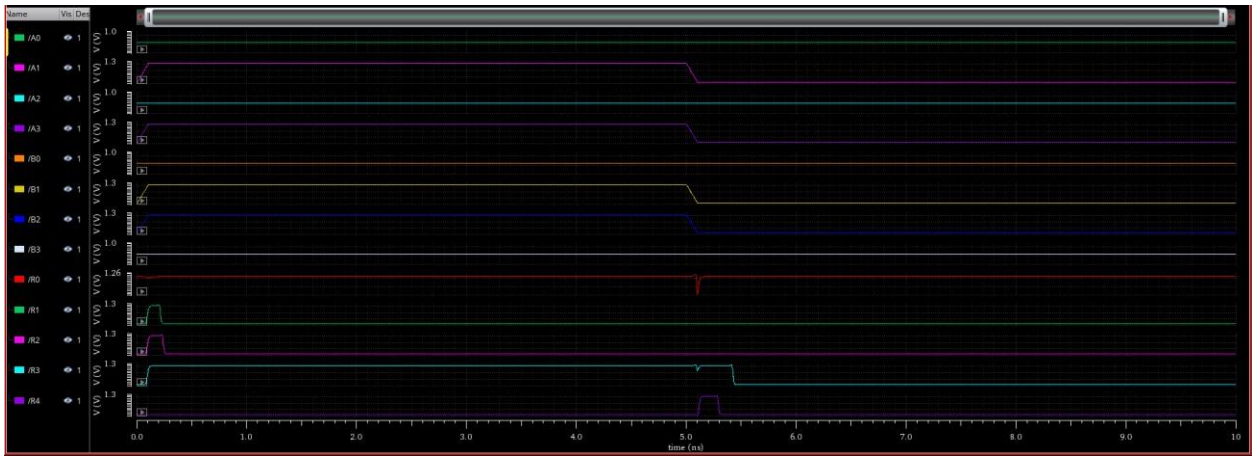
8) INPUTS: A (1010) Operation: 0111 Multiply A *4 Result 101000



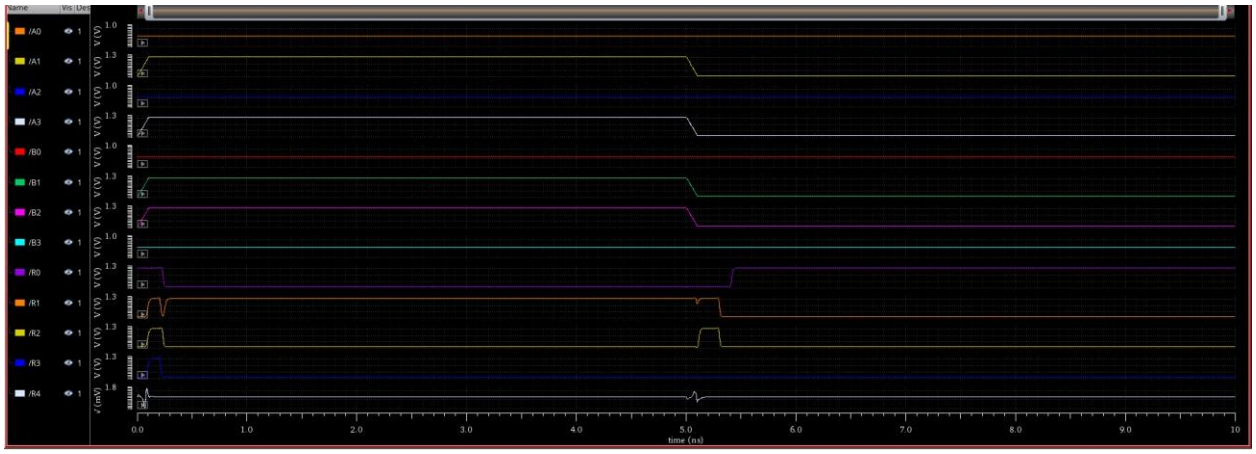
9) INPUTS: A (1010) Operation: 1000 Complement A Result 0101



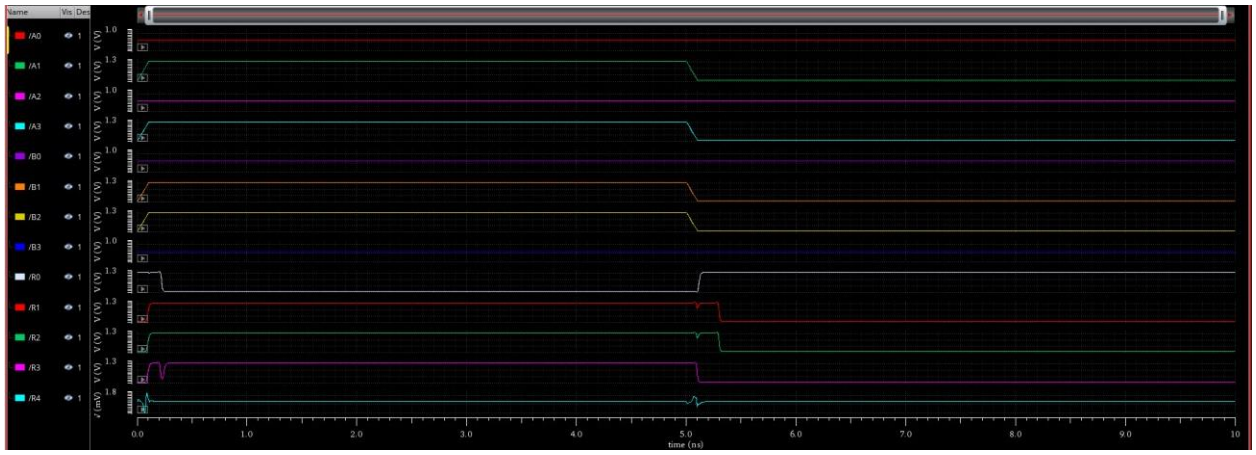
10) INPUTS: B (0110) Operation: 1001 Complement B Result 1001



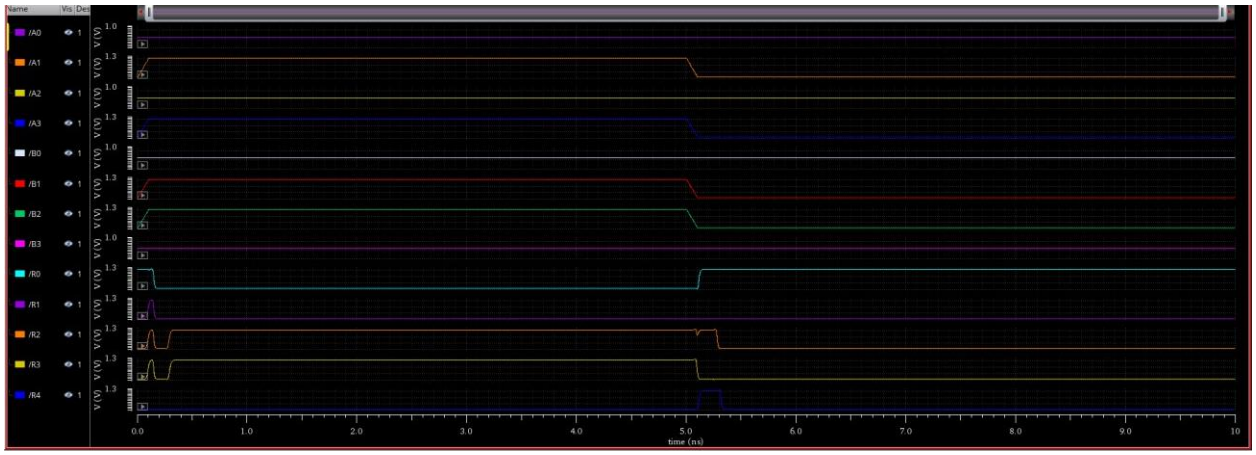
11) INPUTS: A (1010) B (0110) Operation: 1010 A AND B Result 0010



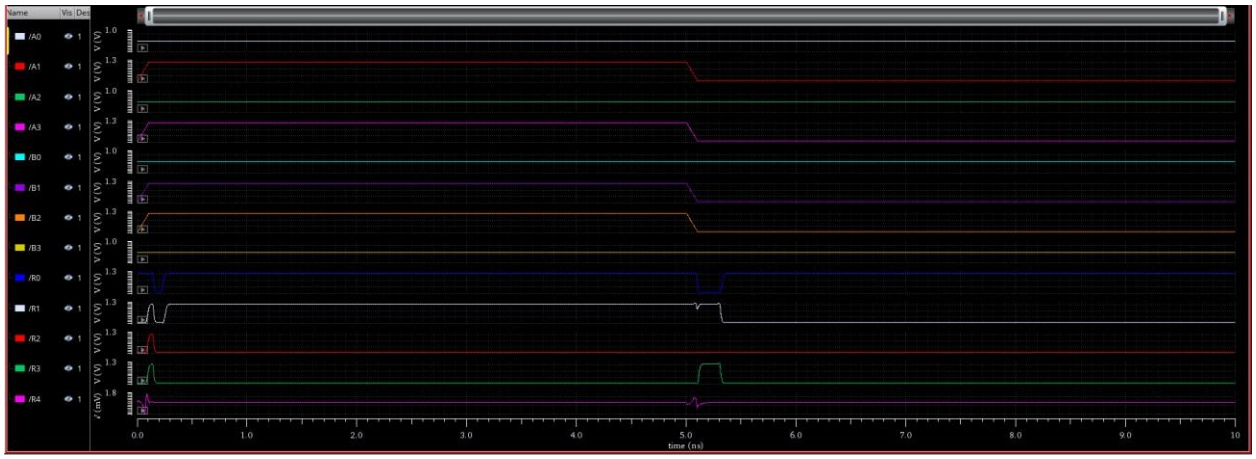
12) INPUTS: A (1010) B (0110) Operation: 1011 A OR B Result 1110



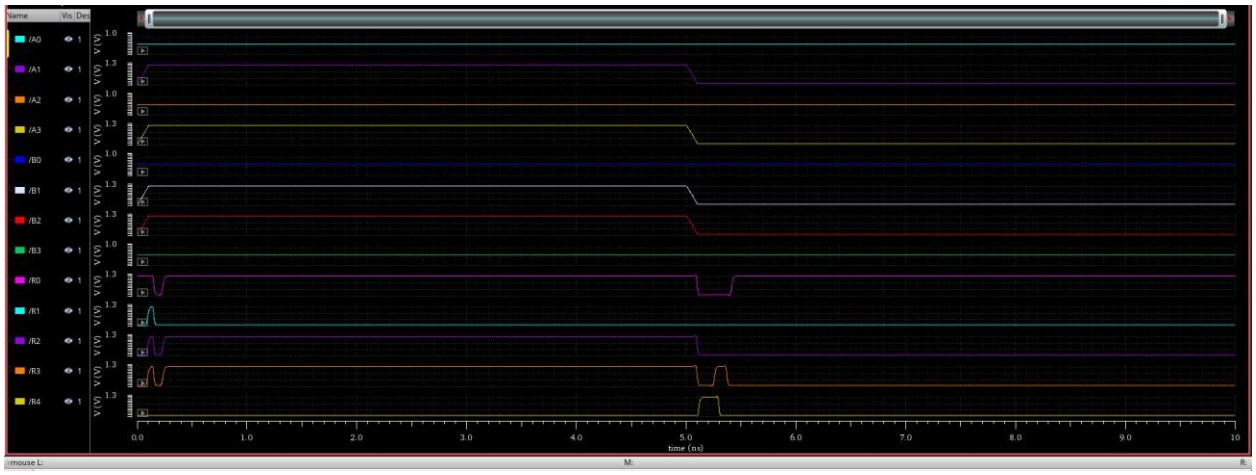
13) INPUTS: A (1010) B (0110) Operation: 1100 A XOR B Result 1100



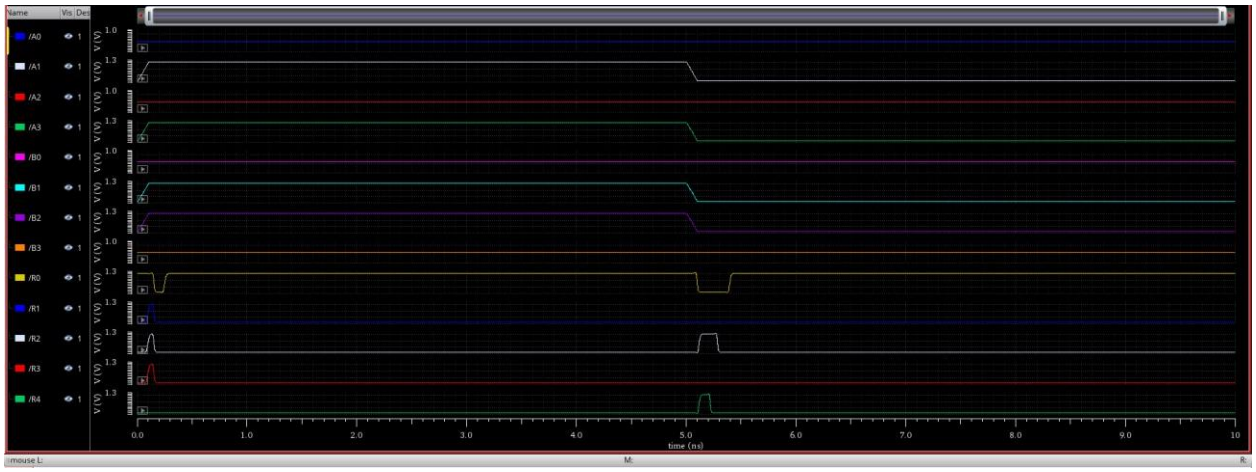
14) INPUTS: A (1010) B (0110) Operation: 1101 A XNOR B Result 0011



15) INPUTS: A (1010) B (0110) Operation: 1110 A NAND B Result 1101

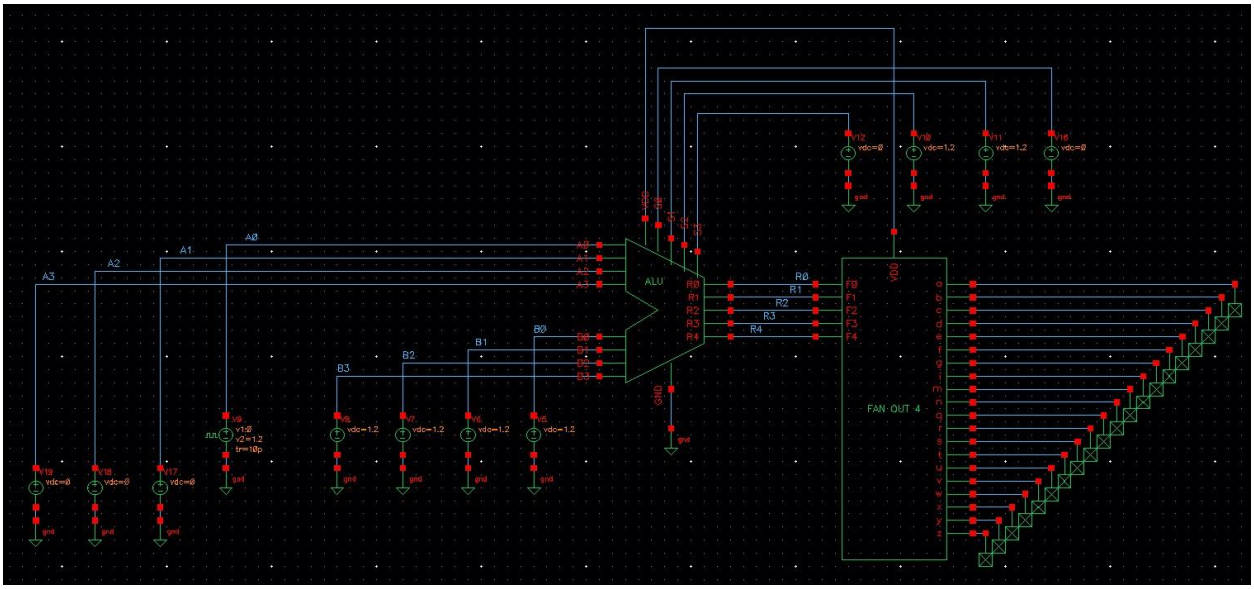


16) INPUTS: A (1010) B (0110) Operation: 1111 A NOR B Result 0001



Worst Delay Calculation:

We connected the Fan out of 4:



We found that the worst delay path (the most number of transistors) is at the functionality (A+B) so the selector is (0110)

To get the worst delay:

Bit A0 was a pulse input (figure 1)

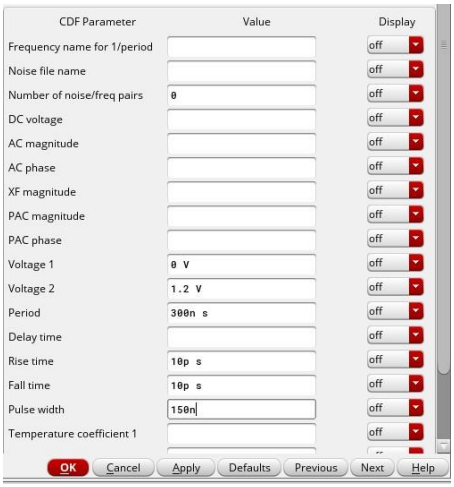
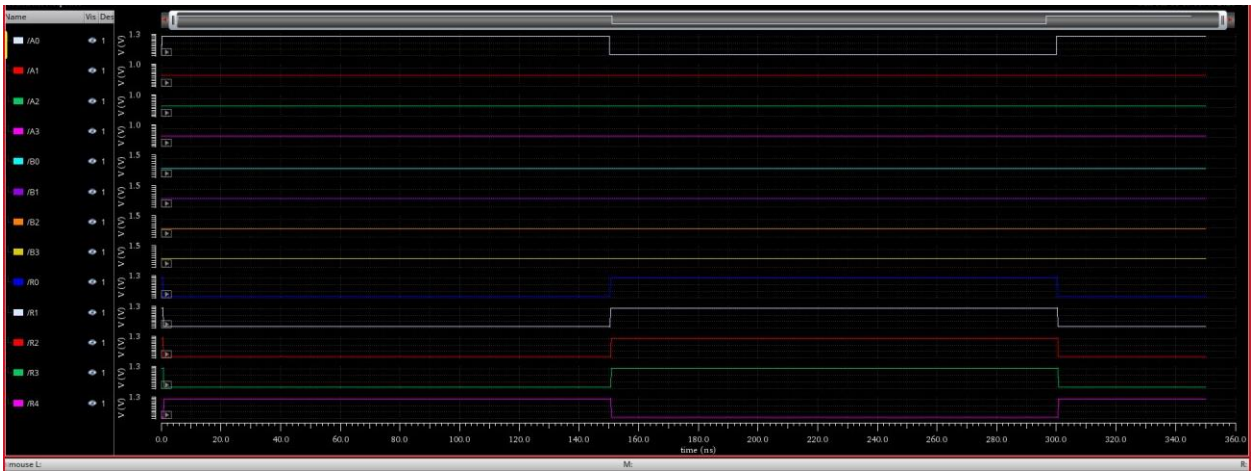
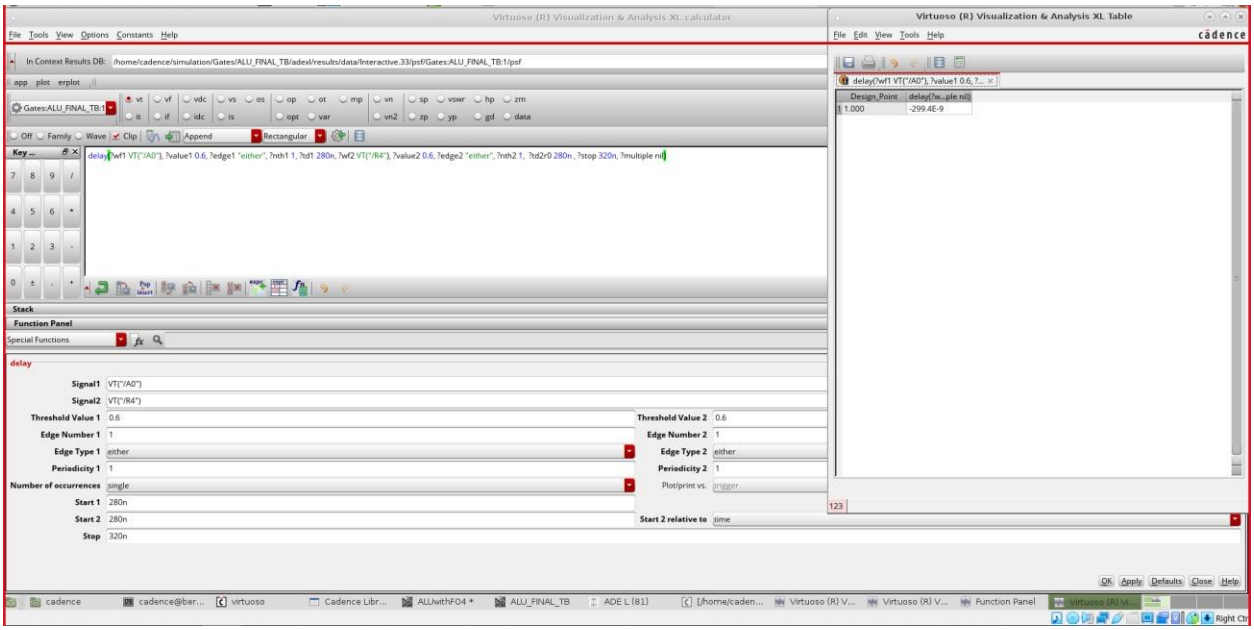


Figure 1

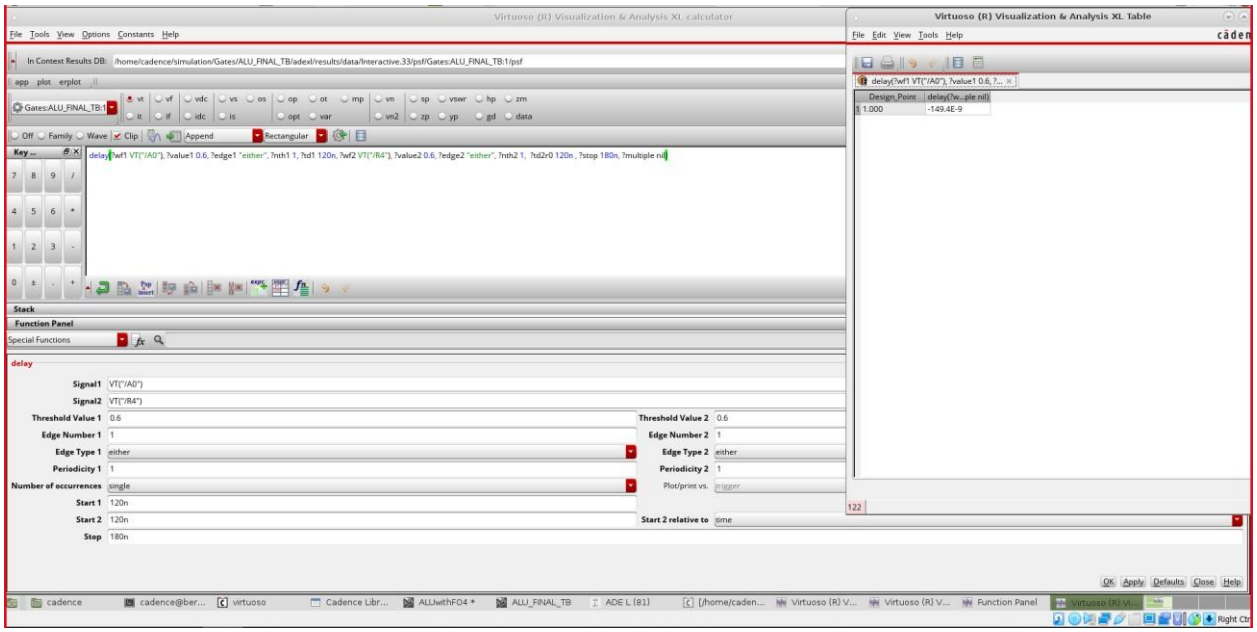
Wave form of the worst delay input:



Where TPLH was 300ns for the input pattern for A is (0001). And B (1111).



Where TPHL was 150ns for the input pattern for A is (0000). And B (1111).



So the average is (150+300)/2 = 225ns

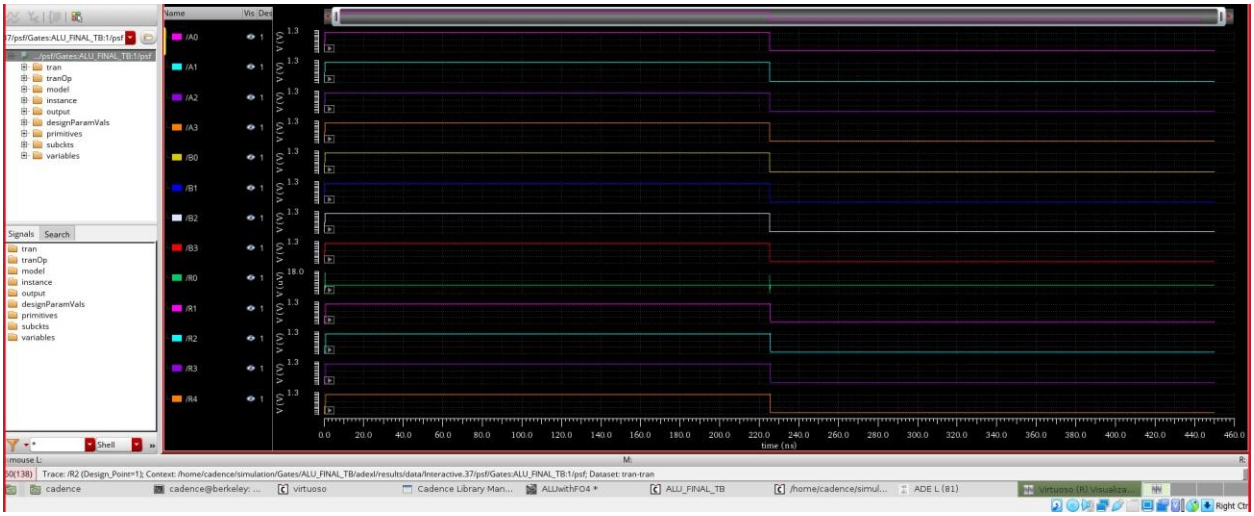
Maximum frequency of the operation = 1 / (2*225n) =

2.222MHZ

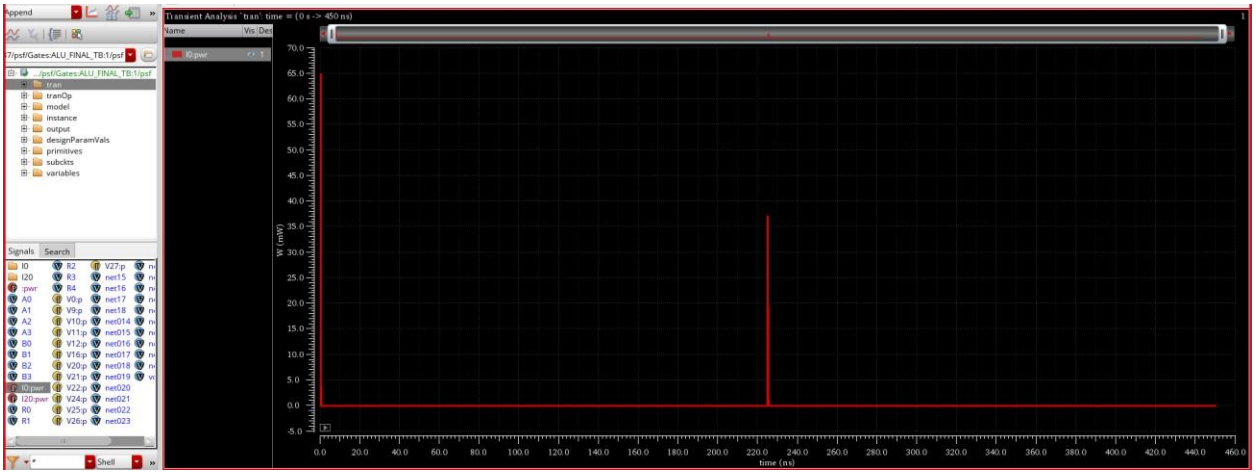
Power consumption:

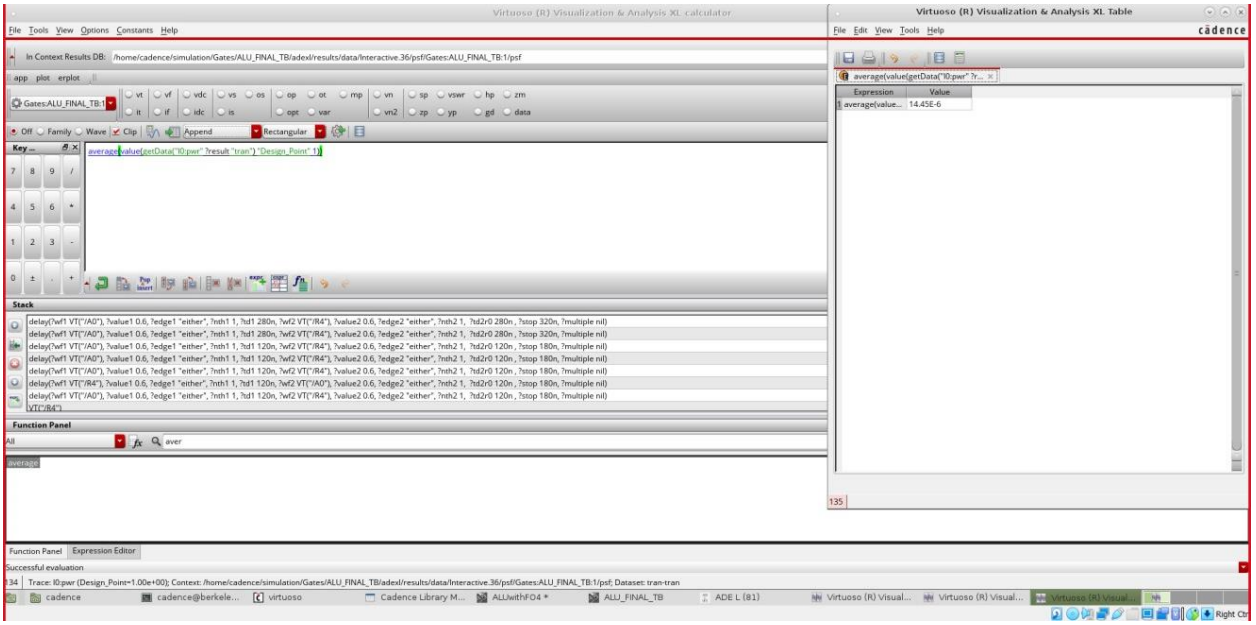
Using maximum frequency operation . this is used to plot power waveform. For a 450ns transient simulation with a functionality (A+B)-> (0110)

With inputs A and B changing from (1111) to (0000) Simultaneously



For which they are pulse inputs with period 450ns with 50% duty cycle and 10ps rising and falling time.





Total Power Consumption is = 14.45 uW