Advanced Full Custom VLSI Design CND-221

Final Project - 4-bit Microprocessor Design and Implementation

Section 21 - CND221_Group 10
Serbenas seif el-eslam V23010307
Eslam asaad mahmoud V23010461
Ahmed mokhtar masoud V23010565
Ahmed Mohamed Rashad V23009920

Center of Nanoelectronics and Devices (CND) - AUC
May 2024

1. Abstract

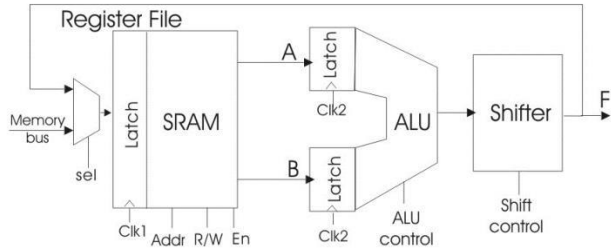
This project Aims to develop and implement 4 bits microprocessor. In this project, students will delve into the intricate aspects of microprocessor layout and circuit design. The project will emphasize hands-on experience in creating circuit designs, layout designs, and verifications for critical microprocessor components, such as the ALU, memory, control unit, and another circuitry using custom and standard cell
libraries.

2. Introduction

This project involves the schematic and layout design of a 4-bit microprocessor data path, including an ALU, a barrel shifter, and a register file using CMOS circuitry and Siemens VLSI design tools.

3. Design

3.1 **The ALU**
will have two 4-bit data inputs and several control signals that specify the ALU function to be executed in each clock cycle.
The ALU input data is loaded into the latch at the rising edge of the ALU clock (clk2), and the output is generated after a delay due propagation through the combinational logic of the ALU and the shifter. The central component of an ALU is an adder, and for this project a Carry select adder will be used.
3.2 **A shifter** provides data manipulation capabilities. A barrel shifter will be used because it has a very efficient layout and can perform n-bit shifts in a single clock cycle.
3.3**The register file** is a block of memory that provides the data inputs to and stores outputs from the ALU.
The register file at read/write address is specified by control bits included in the opcode. In this project, the register file and ALU/shifter will be synchronized by two nonoverlapping clock phases that determine when data is latched into the register file (clk1) and into the ALU (clk2).



4. Schematic

Here we will discuss the schematic of the whole microprocessor starting from top to bottom and we will delve into the building blocks of the system as we go.

4.1 SRAM:
As the requirement of the design came with multi port to read two outputs at the same time.
The SRAM array consists of 8 rows, each comprises 4 8T-SRAM cells. One SRAM cell represents 1 bit of data that could be read from, or written to. Each bit, or individual SRAM cell, is accessed by 2 complementary bit lines. The SRAM rows are accessed by a word line, that is activated using a 2-to-4 CMOS decoder. The inputs to the decoder are a 2-bit address that chooses which word line to activate, and an enable.

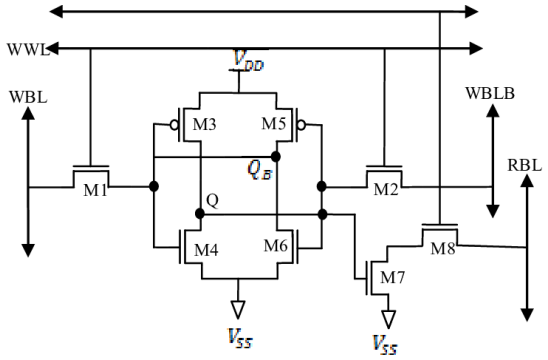


Fig2 8t SRAM.

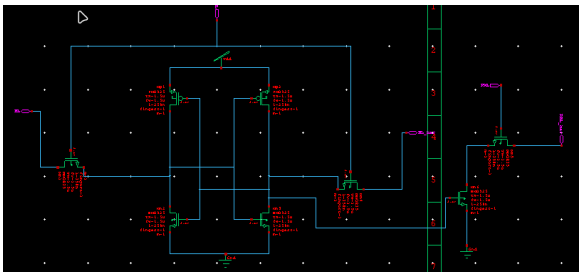
The circuit of an 8T SRAM (Static Random-Access Memory) cell consists of eight transistors configured to store a single bit of data while enhancing read stability and write efficiency.

1. **Storage Nodes (Q and $\bar{Q}$)**:
   - The bit is stored in two cross-coupled inverters formed by four transistors (2 PMOS and 2 NMOS transistors). These inverters create the bistable latch that holds the data.
2. **Write Access Transistors (M5 and M6)**:
   - Two NMOS transistors (M5 and M6) are used for accessing the storage nodes during write operations. These transistors connect the storage nodes to the bit lines (BL and $\bar{BL}$).
3. **Read Access Transistors (M7 and M8)**:
   - Two additional transistors (M7 and M8) are used to create a separate read path. This read path isolates the storage nodes from the bit lines during read operations to prevent data disturbance.
4. **Write Operation**:
   - When writing a bit, the word line (WL) is activated, turning on the write access transistors (M5 and M6).
   - The bit lines (BL and $\bar{BL}$) are driven to the desired values (BL for logic 1 and $\bar{BL}$ for logic 0, or vice versa).

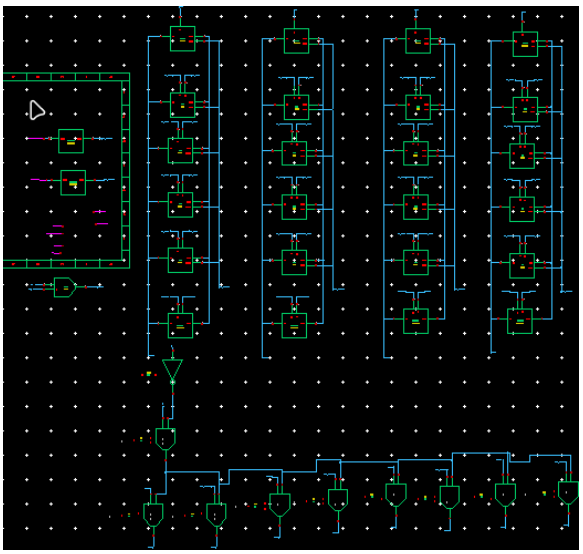
- This allows the bit lines to set the values of the storage nodes (Q and $\bar{Q}$) accordingly.

5. **Read Operation**:
   - During a read operation, the read word line (RWL) is activated, turning on the read access transistors (M7 and M8).
   - The read bit line (RBL) is precharged to a certain voltage.
   - Depending on the value stored in the storage node Q, M7 either pulls RBL down if Q is 0 (since Q is connected to M7) or leaves it high if Q is 1.
   - This change in voltage on RBL is sensed by the read circuitry, determining the stored bit.

The schematic of the system was as follows:



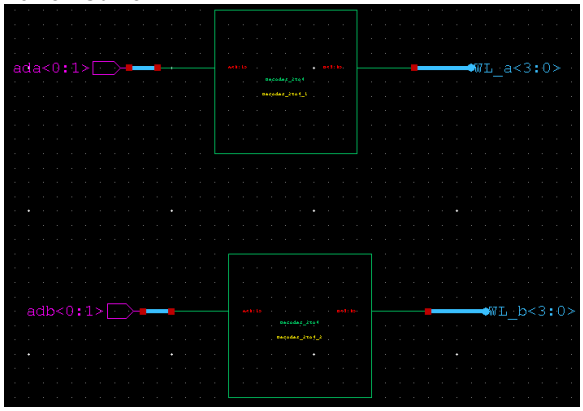
And it was arranged in a 4x4 register file array as follows:



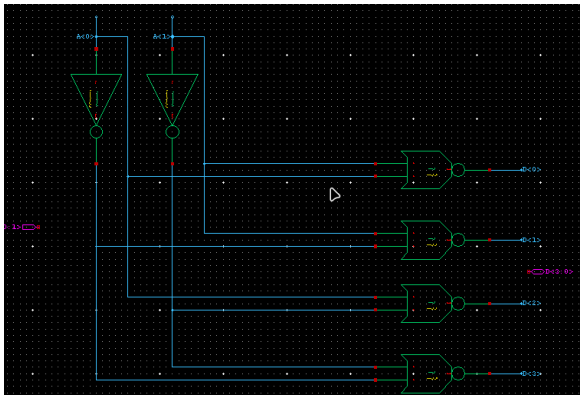
With the following blocks to be integrated by to complete the function of the SRAM :

Row decoder

The Row decoder is NOR based 2:4 Decoder. The word line is selected based on inputs given to the decoder. The designed decoder will be very efficient because of its layout. In each row, a decoder with a dynamic NOR gate is designed. It consists of NN NMOS transistors connected in parallel, where NN represents the number of address lines which has to be decoded. The NOR is a NOT-AND

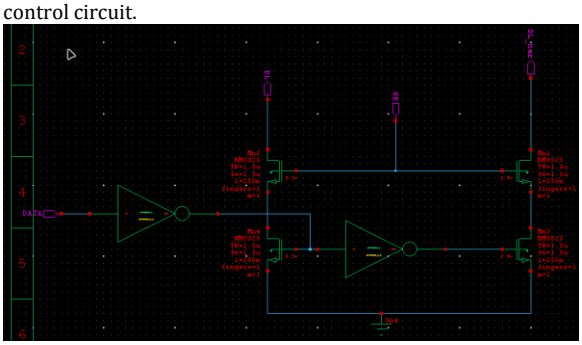


While inside of the 2x4 decoder :



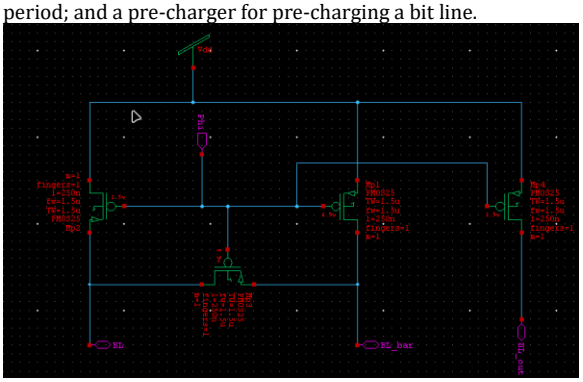
Write driver circuit.

Write driver circuits provide the data which is the input to the bit lines. The circuit is controlled by the Write Enable (WE) signal. Whenever WE = 1 (High), the data will be written in to the cell or else the data will not be written. In the memory cell, only one Write driver circuit is required for one column in the cell. Therefore, by increasing the number of transistors, the speed of the circuit increases, and it is controlled by Write control circuit.

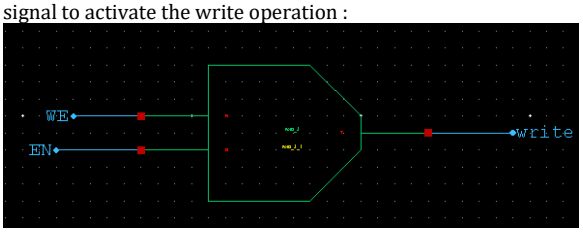


Pre-charge circuit

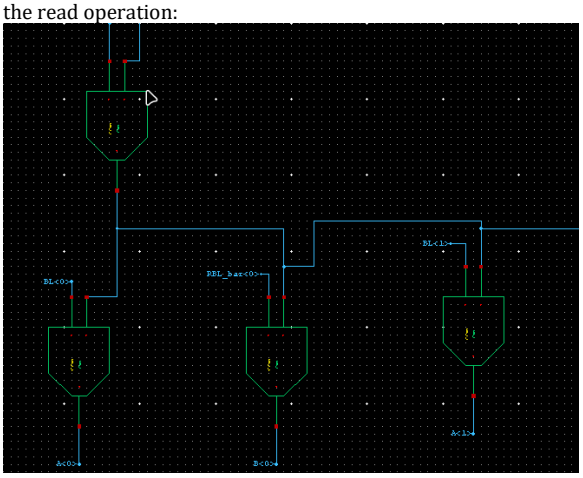
With the help of precharge circuit, the RBL line is charged to VDD. The read operation is carried out and the output obtained will be given to the sense amplifier. A precharge circuit consists of a precharge time controller to generate the precharge time control signal based on a threshold voltage of a transistor; a precharge control signal generator for generating a precharge control signal activated during a predetermined period; and a pre-charger for pre-charging a bit line.



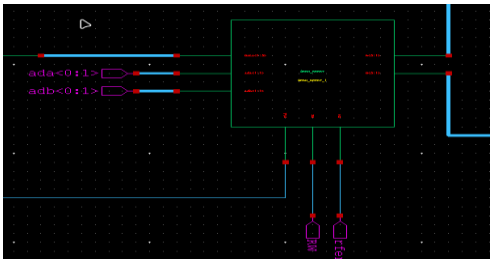
And here the enable signal is put in NAND with the write/read signal to activate the write operation :



While the invert of the signal is in chain of NANDs to decide for the read operation:



Here showcasing the symbol of the SRAM ARRAY :

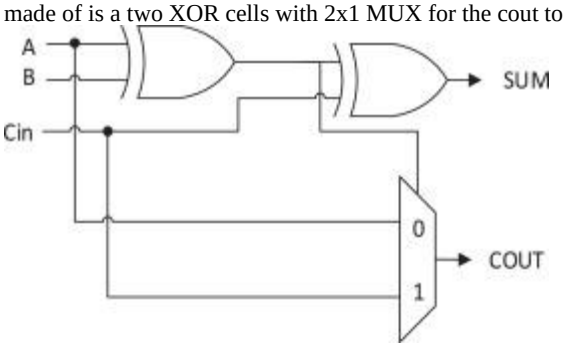


The ALU:

as for the design of the ALU it will be divided to three circuits :

1. Carry Adder
2. Logic Unit
3. Barrel Shifter

As for the carry adder the building block of the full adder is made of is a two XOR cells with 2x1 MUX for the cout to

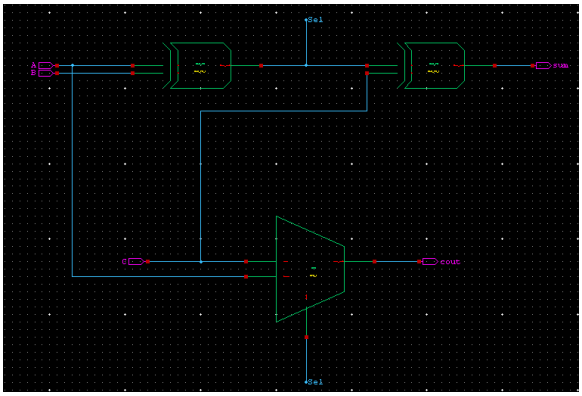


shows the shows the proposed architecture of XOR MUX Full adder design while employing the two successive stages of XOR gate for Sum operations and 2:1 non-inverting multiplexer while using for Carry operations, it takes totally 2 logic gates and 1 multiplexer.

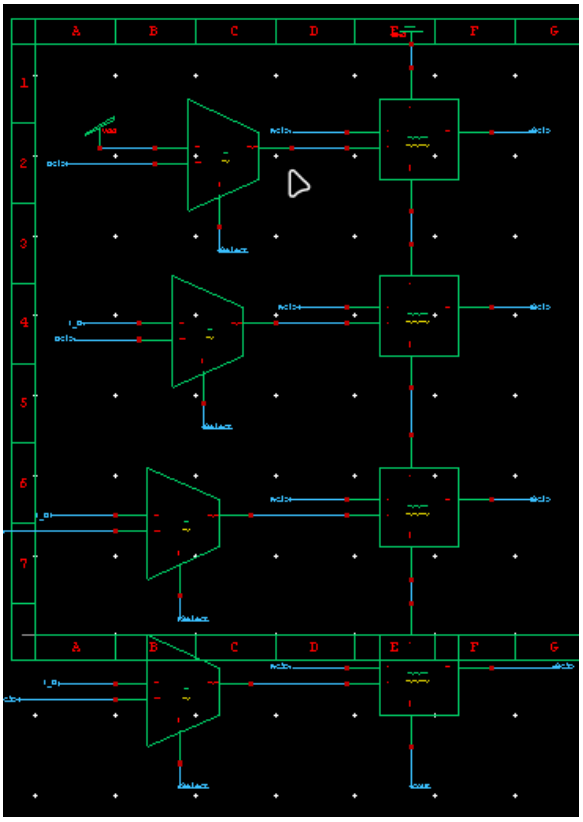
The truth table of XOR-MUX Full adder design is shown in Table 1. A Gate level structure of Conventional RCA with BEC adder de- sign Also, in this Fig XOR, AND, OR, NOT gate based structure are shown, it takes totally 7 logic gate and 1 multiplexer. The truth table of Conventional RCA Based Carry select adder is shown in Table .

Table 1
Truth table of XOR-MUX full adder.

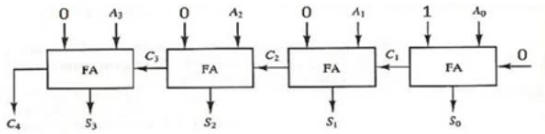
C_{IN}	A	B	SUM	COUT
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



While the whole 4 bit adder system is connected as follows:

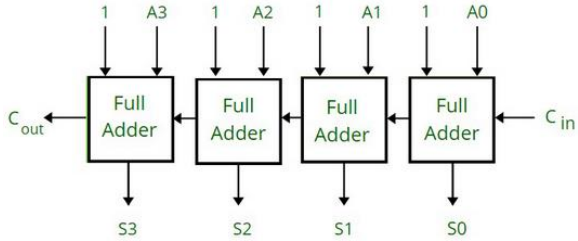


Now for the input B there's a Mux connected to decide whether to pass the B input or to perform the other two operation requested from the Adder unit : Increment and decrement :
As for the incrementer using units of full adders the design decided as following:



It adds 1 binary value to the existing binary value stored in the register or in other words we can simply say that it increases the value stored in the register by 1.
For any n-bit binary incrementer ,'n' refers to the storage capacity of the register which needs to be incremented by 1. So we require 'n' number of half adders . Thus, in case of 4 bit binary incrementer we require 4 full adders.

The Decrementer:

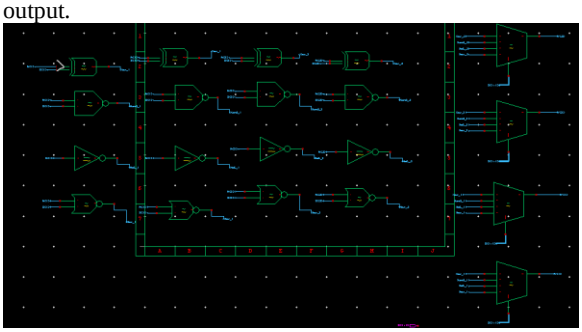


And in that fashion the first dull adder is always 1 while the others will be choosing between the 0 and 1 values to choose between increment or decrement respectively.
The value of Cin in all cases will be zero.

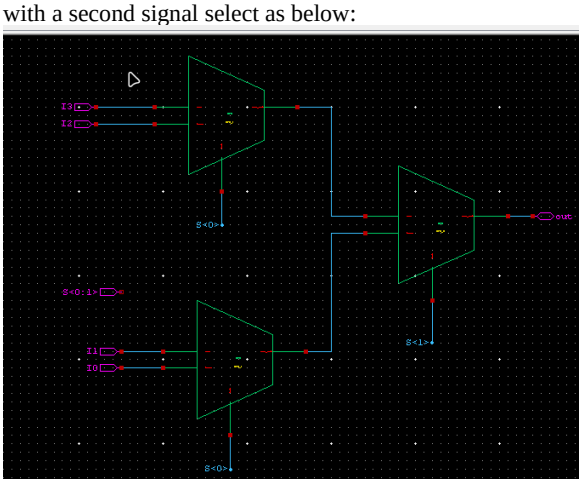
The Logic Unit
in the Logic unit the operation that we need to process is one of the following :

1- NAND
2- XOR
3- Invert
4- NOR

And this is deployed as shown below for each input bit of A and B we can decide to choose which operation to be output with the help of 4x1 MUX unit that decides which operation on the basis of the select signal that is inserted by the Opcode to choose which operation to appear on the output.

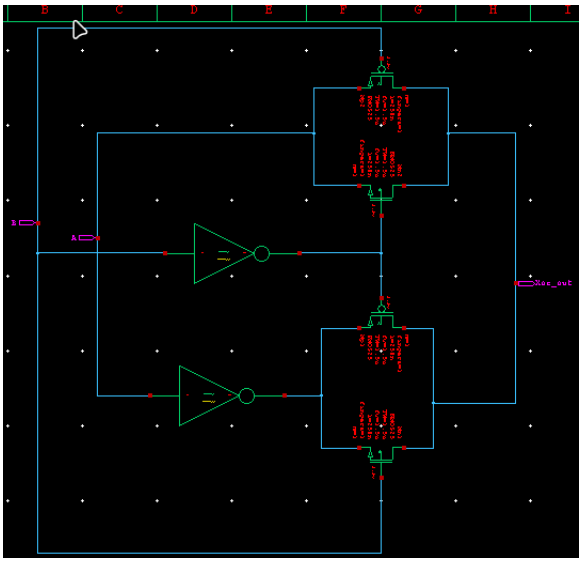


The 4x1 Mux is built on the Notion of two 2x1 Muxes on the first stage and then brached to a second phase MUX with a second signal select as below:



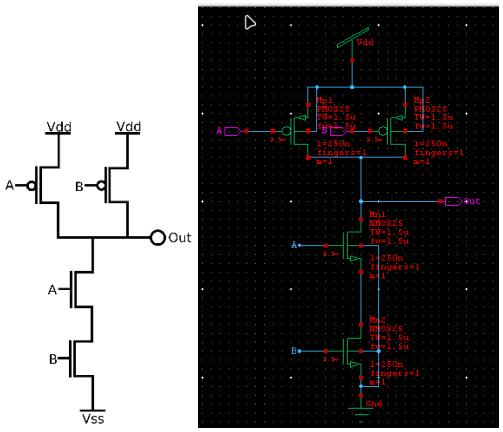
The basic building blocks of this unit is :

1- XOR:

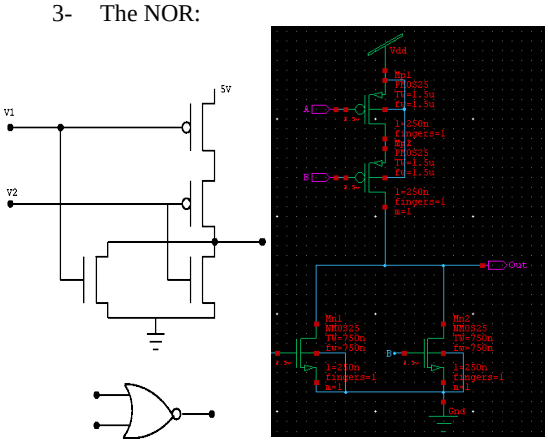


The circuit shown here realizes the XOR function of inputs A and B using just six transistors. The first stage of the circuit consists of a basic static inverter, which generates the inverse of input A. Note that the inverse of A is connected to the gate of the transmission-gate n-channel transistor, while A is connected to the gate of the p-channel transistor. Therefore, the transmission gate is conducting when A is low, and passes the value of B. This effectively realizes the (!A & B) term of the standard disjunctive expansion of the XOR function.
In order to realize the remaining (A & !B) term of the XOR expansion, the circuit uses a dirty trick. Obviously, the two transistors in the center of the schematics form an inverter that calculates !B, and whose output is connected to the output of the XOR gate. However, the source and drain connections of the transistors are not directly connected to VCC and GND, but to the input A and the output !A of the first-stage inverter. The net result is that the inverter is only active when A is high, because its power-supply is effectively switched-off when A is low.

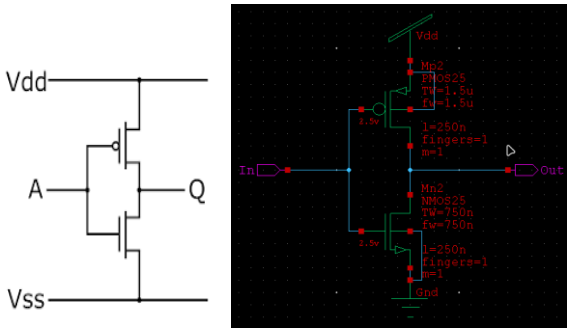
2- NAND



3- The NOR:



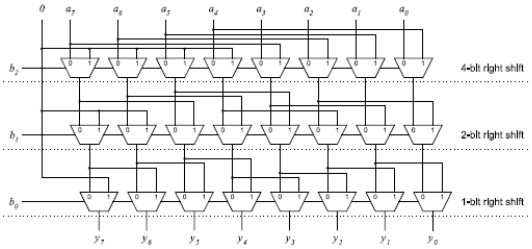
The Inverter:



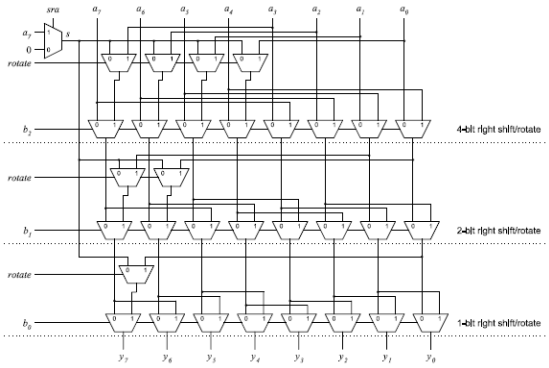
The Shifter:

Shifting and rotating data is required in several applications including arithmetic operations, variable-length coding, and bit-indexing. Consequently, barrel shifters, which are capable of shifting or rotating data in a single cycle, are commonly found in both digital signal processors and general-purpose processors

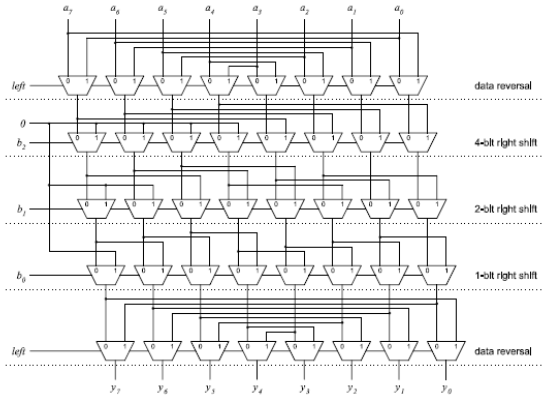
we took the same structure of the MUX based barrel shifter with the structure of the shifter as follows:



And then to be a shifter and rotater it's proposed to be in the same as following:

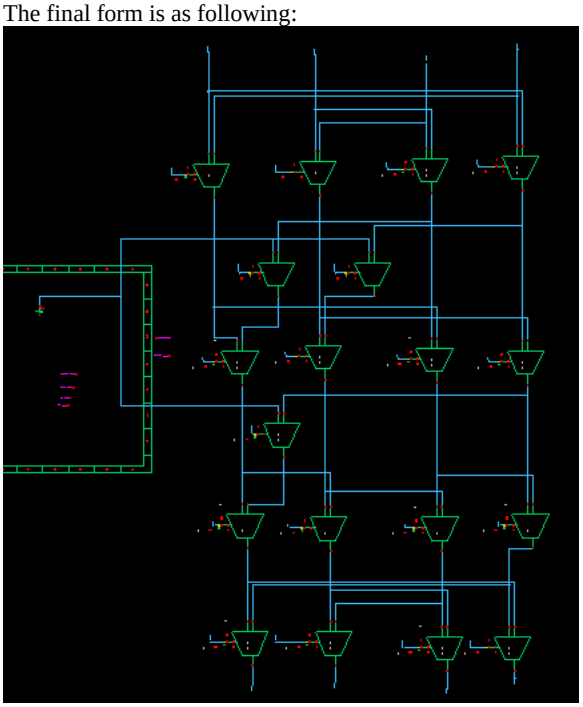


Now we need to add the left option to the shifter and rotate operation :



By adding the data reversal decoding line in the start and the end of the array

The final form is as following:

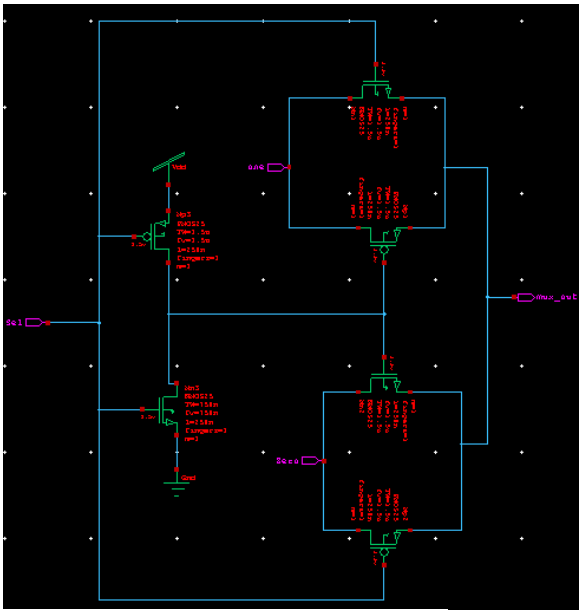


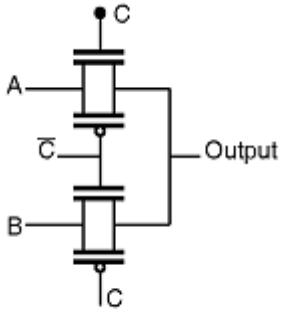
This design can perform the following :

- Pass (no shift)
- Shift left 1.
- Shift left 2.
- Shift right 1.
- Shift right 2.
- Rotate right 1.
- Rotate left 1.

The building block of this design is the following:

- 2x1 MUX



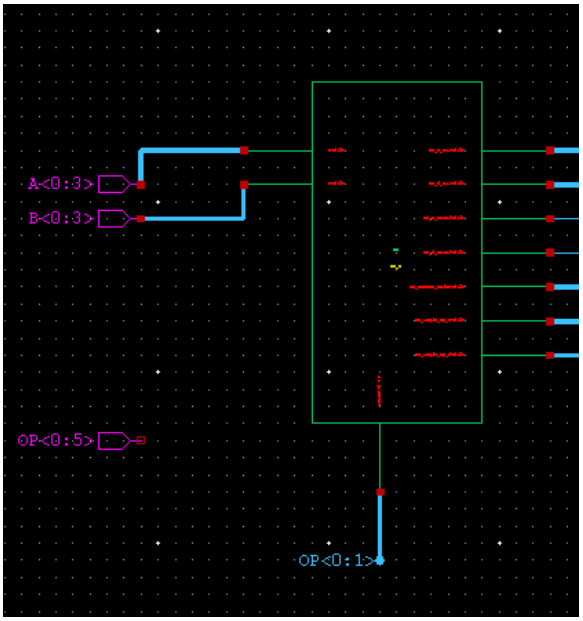


2 : 1 MUX using transmission gate

When the control signal C is low then the upper transmission gate turns OFF and it will not allow A to pass through it, at the same time the lower transmission gate is 'ON' and it allows B to pass through it so the output = B.

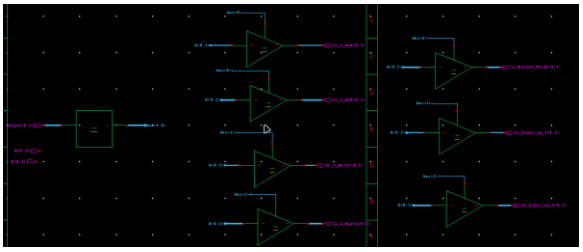
All of that is controlled by input controller unit that decide based on the opcode <0:1> which input will be passed to activate which unit of the three above.

The control unit:



The control unit presented is designed to be a low-power which works only while an operation decoded; otherwise it will be off. It consists of a 2-4 bit decoder that decodes the OP code and the output values will be the enable signals of the tri-state buffers which control the operand transmission to the following blocks. There's two outputs reserved for future to implement the Multiplier cell.

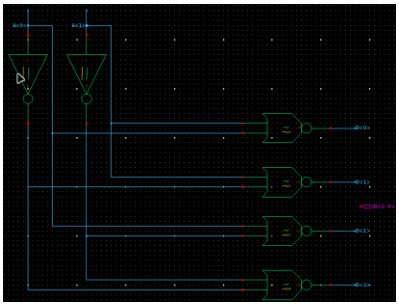
The building blocks of such unit is:



- The 2x4 Decoder:

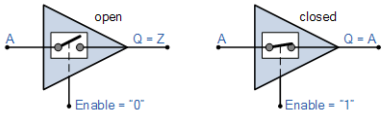
In this cell, Each output is having one product term. So, there are four product terms in total. We can implement these four product terms by using four AND gates having three inputs each & two inverters. The circuit diagram of 2 to 4 decoder is shown in the following figure.

Inputs		Outputs			
A_1	A_0	Y_3	Y_2	Y_1	Y_0
x	x	0	0	0	0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

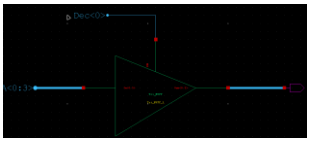


Tri-State Buffer:

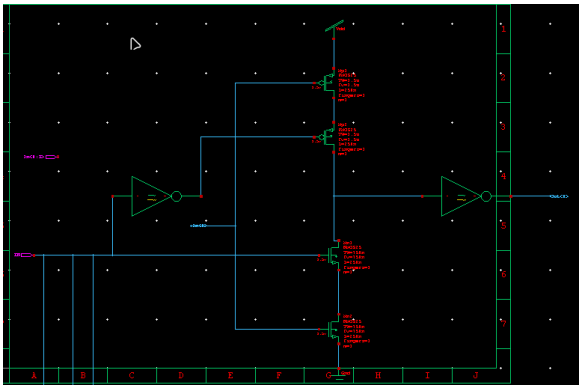
A Tri-state Buffer can be thought of as an input controlled switch with an output that can be electronically turned "ON" or "OFF" by means of an external "Control" or "Enable" (EN) signal input. This control signal can be either a logic "0" or a logic "1" type signal resulting in the Tri-state Buffer being in one state allowing its output to operate normally producing the required output or in another state were its output is blocked or disconnected.



Then a tri-state buffer requires two inputs. One being the data input and the other being the enable or control input as shown.

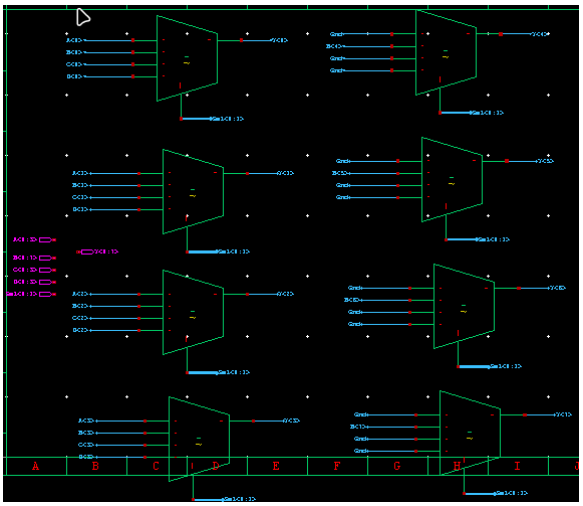


When activated into its third state it disables or turns "OFF" its output producing an open circuit condition that is neither at a logic "HIGH" or "LOW", but instead gives an output state of very high impedance, **High-Z**, or more commonly Hi-Z. Then this type of device has two logic state inputs, "0" or a "1" but can produce three different output states, "0", "1" or " Hi-Z " which is why it is called a "Tri" or "3-state" device.

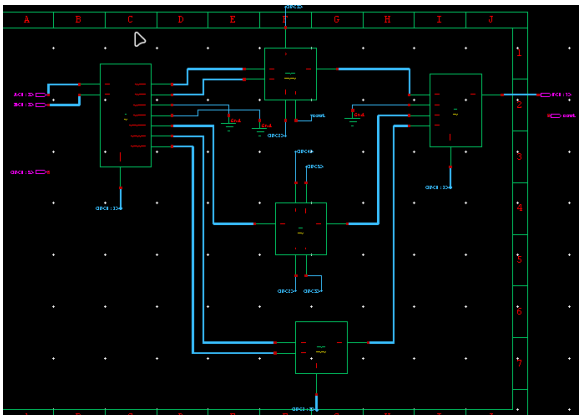


That was to decide the input passing to each unit, as for the output it's decided by a 4x1 MUX unit to decide which output will be passed outside of the ALU unit.

The multiplexer used in the MUX selection block is (4 to 1) 8-bit MUX. OP[3:2] is the selector that chooses which output will be represented as depicted in Figure 10. The MUX is combined from eight 4 to 1 (1-bit) MUXs

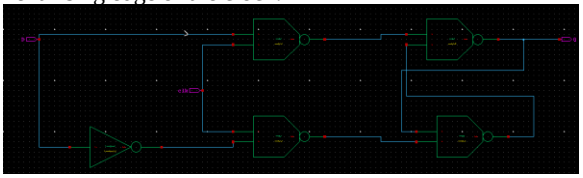


An the whole ALU Unit schematic is as follows



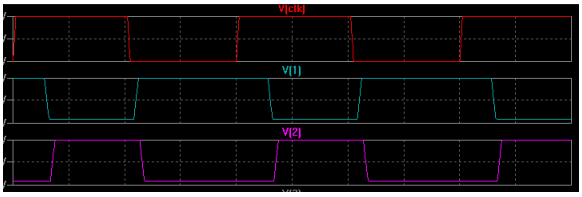
The Latches:

D latches are also known as transparent latches and are implemented using two inputs: D (Data) and a clock signal. The output of the latch follows the input at the D terminal as long as the clock signal is high. When the clock signal goes low, the output of the latch is stored and held until the next rising edge of the clock.

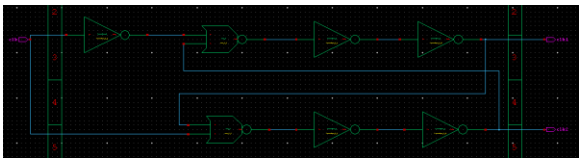


That was the schematic of a one bit D-Latch, as it was repeated 4 times to make the D-latch4bit unit that was decided before the SRAM and the inputs of the ALU.

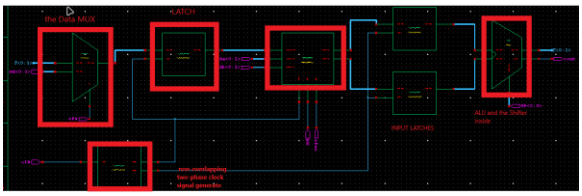
The two phase Generator:
This NOR-flipflop based circuit implements a **non-overlapping two-phase clock signal generator** and can be used to derive a two-phase clock signal from a single and possibly non-symmetrical clock signal
While the upper circuit uses 'typical' gate-delays in the nanosecond range, the bottom circuit uses slowed-down gate-delays of 0.2 seconds per gate. This should make it easier to observe the idea behind the circuit.



And it's schematic:



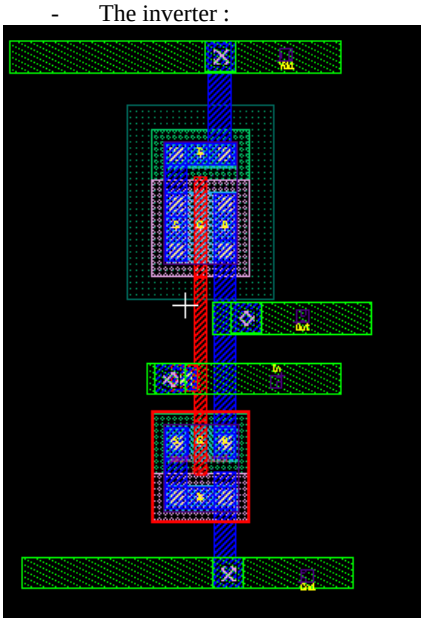
And the whole system schematic is as follows:



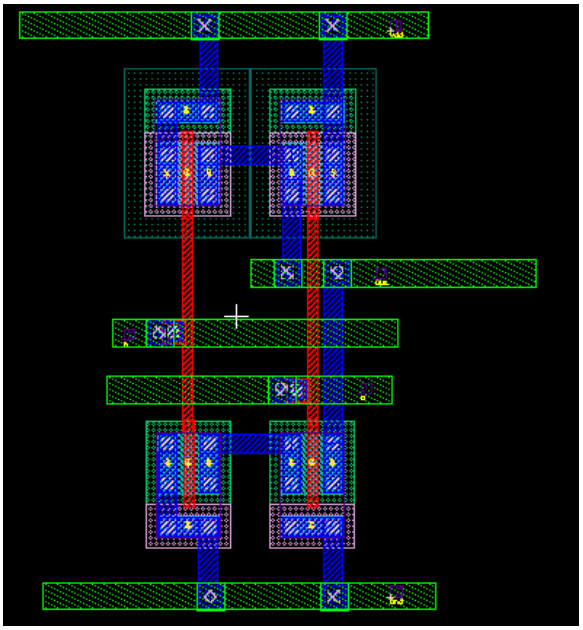
5 Layout:

Here we will showcase the Layout of the building block of the systems and then will go up to top design:

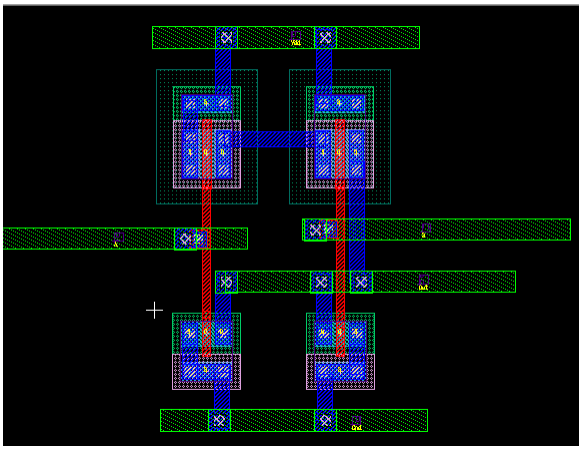
- The inverter :



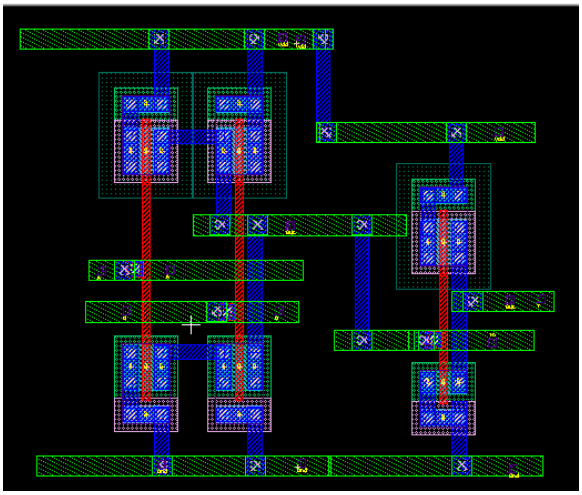
The NAND:



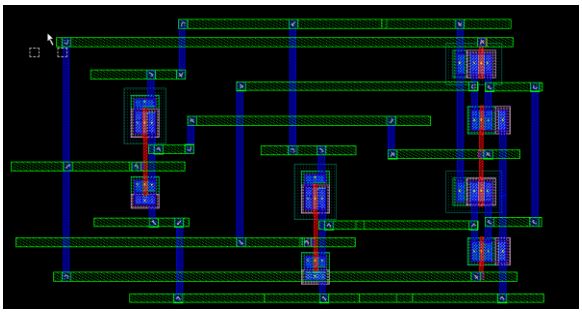
The NOR:



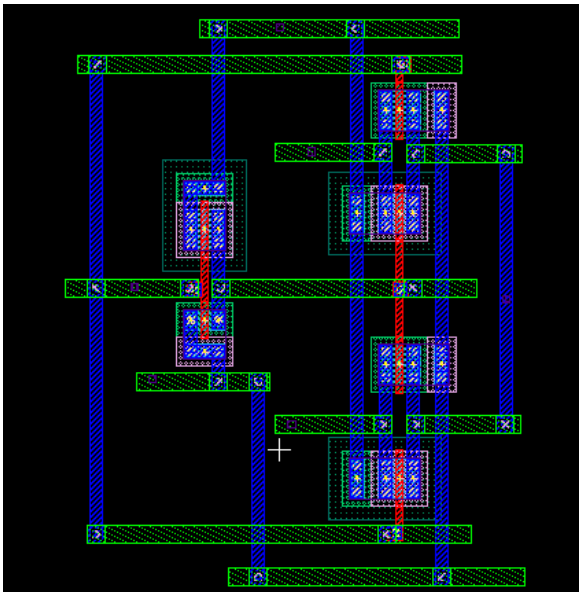
The AND:



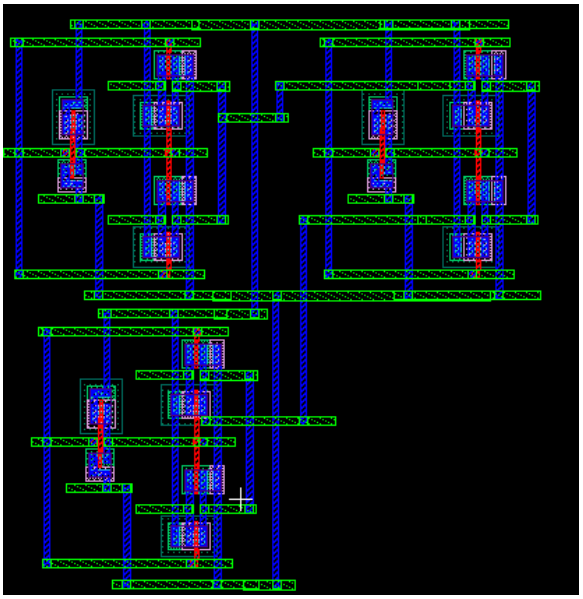
The XOR :



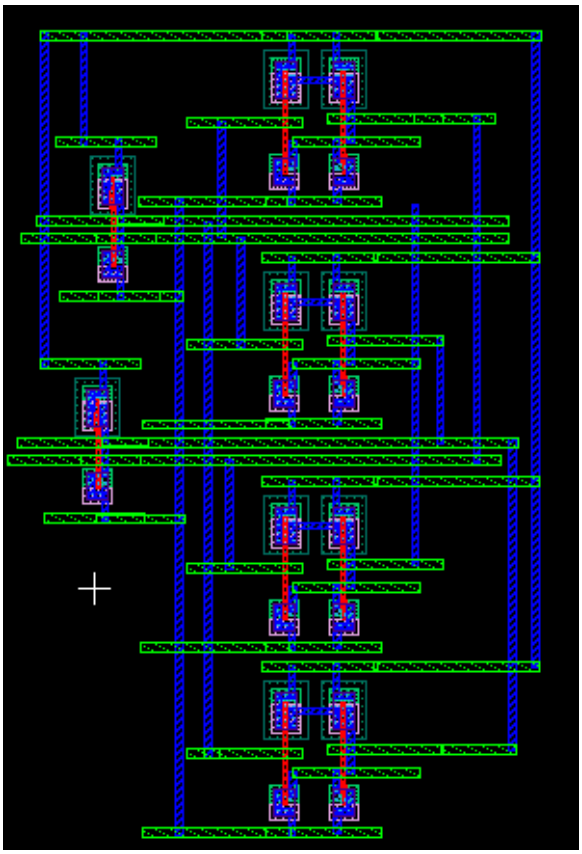
The MUX:



The 4x1 MUX:

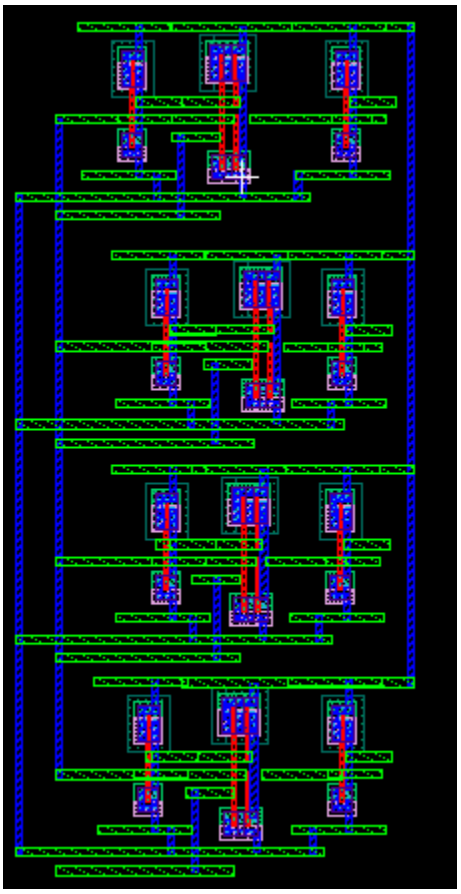


The 2x4 Decoder:



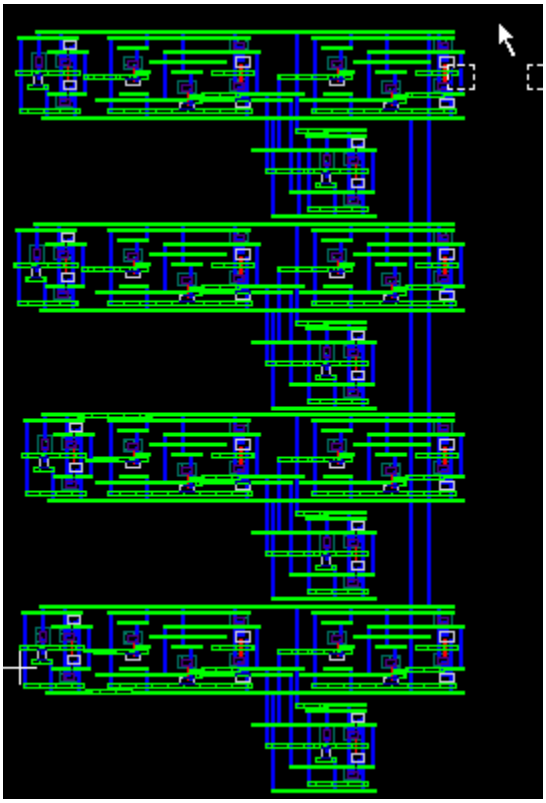
ALU building Blocks Layout:

Tri state Buffer :

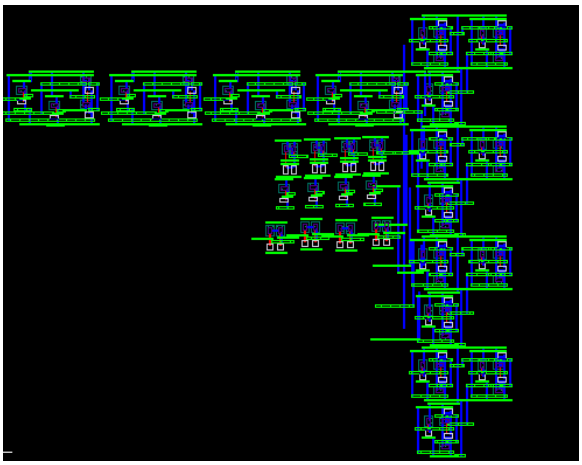


The One bit adder :

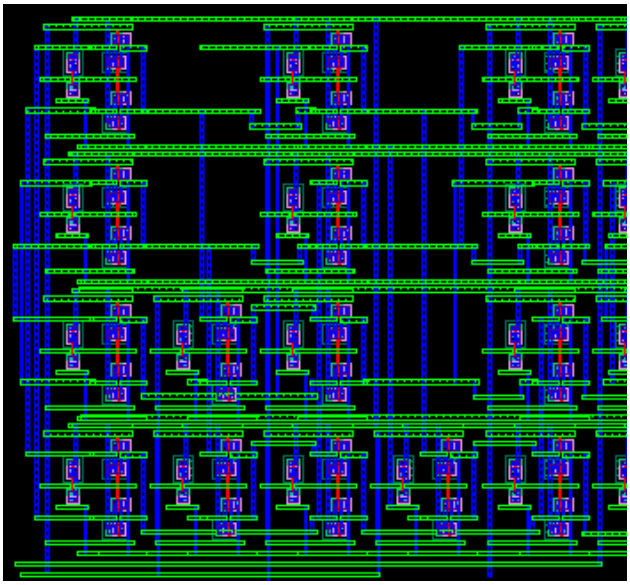
The Adder Unit:



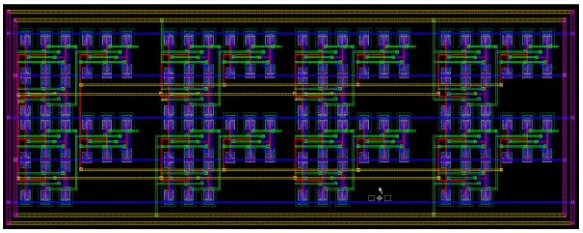
The Logic Operation Unit:



The barrel shifter:

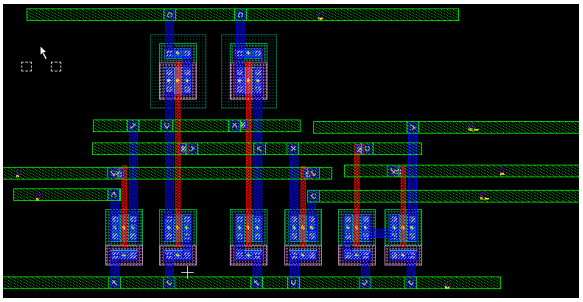


The MUX Selector:

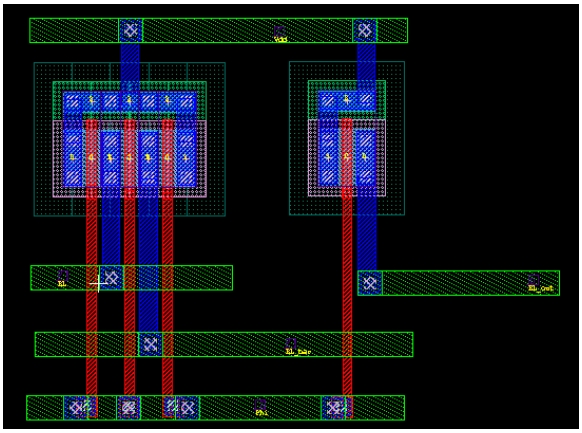


The SRAM building Blocks layout :

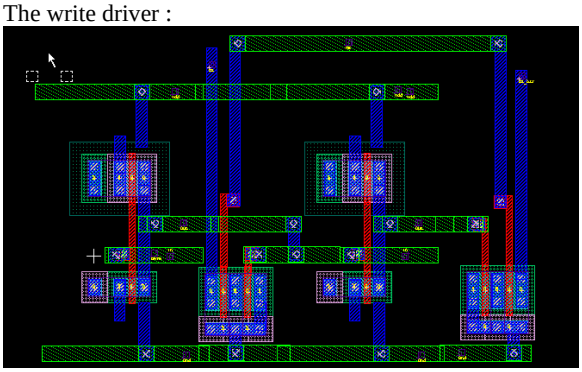
The 8t SRAM cell :



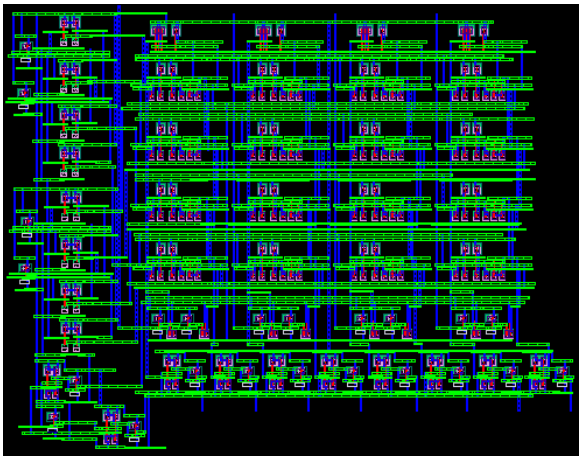
The Precharge unit :



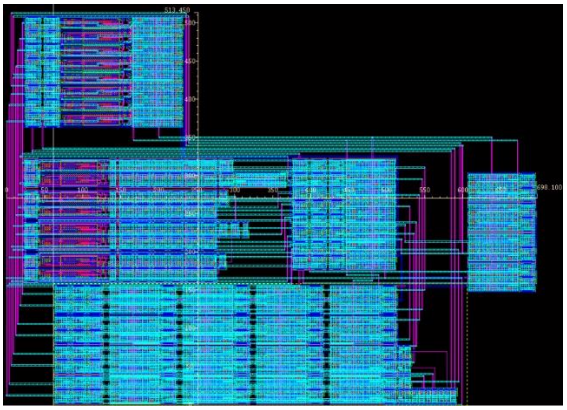
The write driver :



The whole SRAM ARRAY:

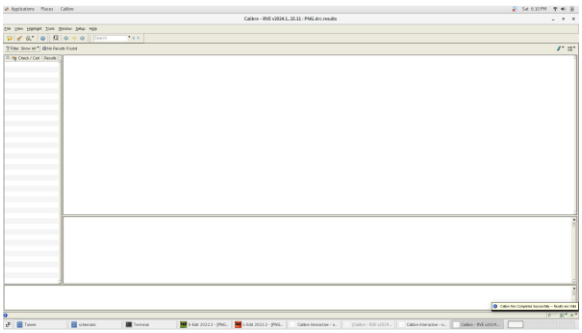


Full Design Layout



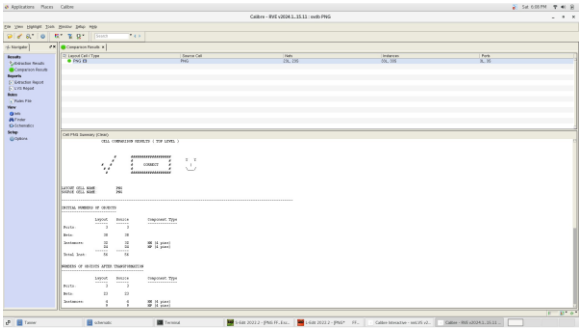
6 Verification (DRC and LVS)

The system was divided into building blocks, and it was verified for both DRC and LVS block by block till the full system integration.



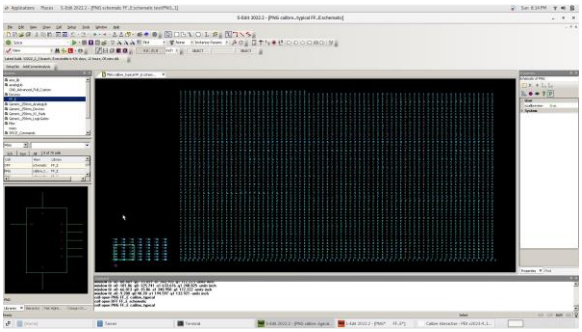
DRC clean Results

And also for LVS:

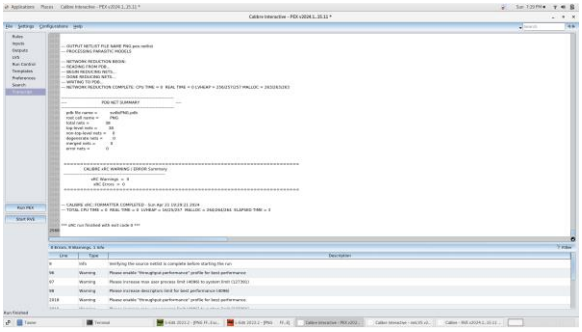


LVS Clean Results

And for PEX results :



Parasitic Extraction

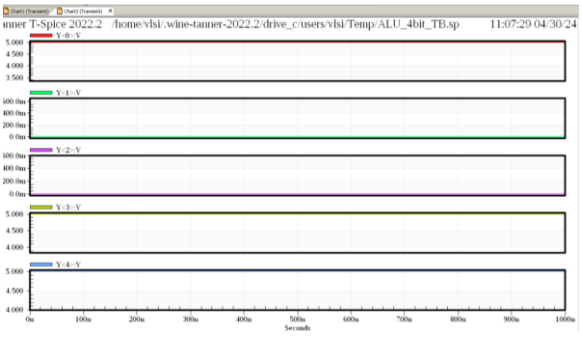


PEX tool clean results

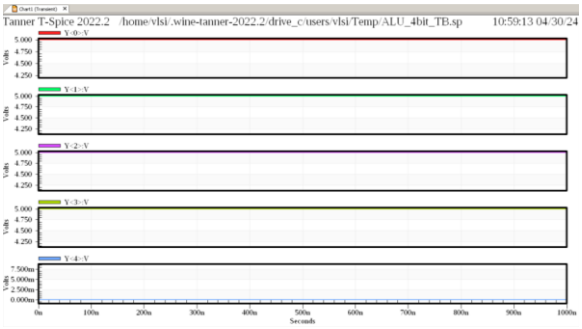
7 Simulation

As for the simulation :ALU simulation :

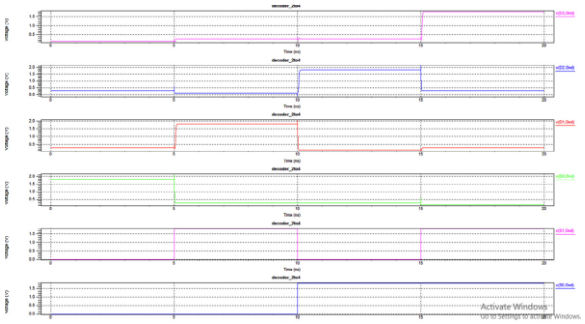
Case 1, Adding A+B = 10001



Case 2, Shifting A>>B



2x4 Decoder :



4x4 SRAM array waveform: