THE UNIVERSITY OF SCIENCE AND TECHNOLOGY

UNIVERSITY OF DANANG

-----□□&□□-----

REPORT

IC Design Project

Team Members: Hoàng Kim Uyên

Trần Thị Nguyệt Hà

Nguyễn Thị Phương Diệu

Hồ Xuân Đạt

Class: 19ECE

Instructor: Nguyễn Văn Cường

Table of Contents

IC DESIGN FINAL PROJECT

A. Theory Part

1. Logic Design:

a. Top-level Interfaces:

- The top-level interface refers to the interface that connects the internal circuitry of a VLSI chip with the outside world. It defines the signals and communication protocols through which the chip interacts with external components or systems. It includes various aspects such as:

i. Input/Output Signals: The interface defines the I/O signals through which the chip receives inputs from external devices and transmits outputs to them. These signals can include data lines, control lines, clock signals, power, and ground connections, and any other necessary signals for communication.

Inputs	Outputs
ph1	MemWrite
ph2	Adr[7:0]
reset	WriteData[7:0]
MemData[7:0]	

Fig.1. Top-level inputs and outputs of MIPS processor

ii. Signal Types and Formats: The interface specifies the types and formats of the signals exchanged between the chip and external components. This includes defining the data formats (such as digital, analog, or mixed-signal) and signal characteristics (such as voltage levels, timing requirements, noise considerations, and impedance matching).

iii. Communication Protocols: The interface may incorporate communication protocols that govern the exchange of data and control information between the chip and

external devices. Common protocols used in VLSI designs include SPI (Serial Peripheral Interface), I2C (Inter-Integrated Circuit), UART (Universal Asynchronous Receiver-Transmitter), and Ethernet.

iv. Pin Assignment: The top-level interface determines the assignment of physical pins or pads on the chip for connecting with external components. It specifies the function of each pin, ensuring proper connectivity and compatibility with external systems.

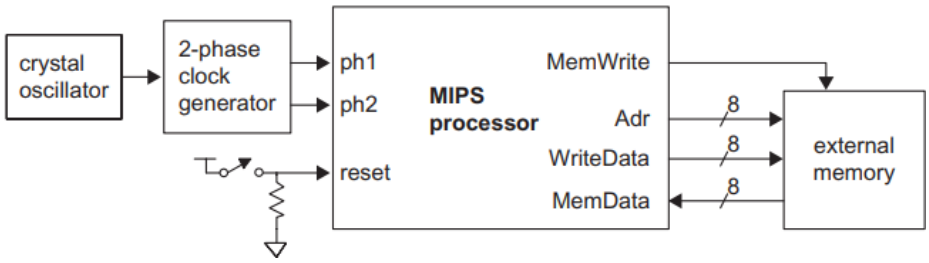


Fig.2. MIPS computer system

b. Block Diagram:

- In logic design in VLSI, a block diagram is a graphical representation that depicts the high-level structure and interconnections of functional blocks within a digital system. It provides an overview of the system's architecture and shows how the different components or modules are organized and connected.

- A block diagram typically uses rectangular blocks to represent functional units or modules, and lines or arrows to illustrate the data flow or signal connections between them. Each block represents a specific function or a collection of logic gates that perform a particular task. The interconnections between the blocks represent the flow of data or control signals between them.

c. Hierarchy:

- In logic design, hierarchy refers to the organization of a digital system into smaller, more manageable parts. This is done to simplify the design process, improve modularity, and increase design efficiency.

- The hierarchy in logic design is typically represented by a hierarchy chart or a block diagram. At the highest level, the system is represented by a single block, which contains all the other blocks that make up the system. Each block in turn may be composed of smaller blocks, and so on. The blocks at the lowest level are referred to as "primitive blocks" and are typically built using basic logic gates.

- The use of hierarchy in logic design allows designers to break down complex systems into smaller, more manageable parts, which can be designed and tested independently. This approach also allows for the reuse of common building blocks, which can save time and reduce errors. In addition, hierarchy can make it easier to debug a system, as errors can often be isolated to a specific block or level.

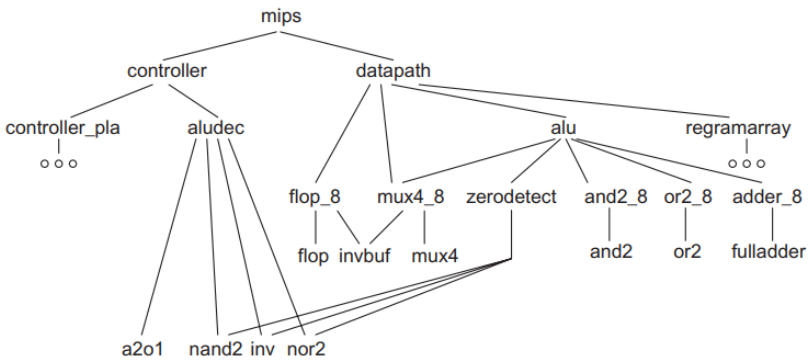


Fig.3. MIPS design hierarchy

d. Hardware Description Languages:

- Hardware Description Languages (HDLs) are specialized computer languages used to describe and design digital circuits and systems in VLSI (Very Large Scale Integration) design. HDLs allow designers to create models of digital circuits and systems, and simulate their behavior before they are physically implemented in hardware. There are two main HDLs used in VLSI design: Verilog and VHDL (VHSIC Hardware Description Language). Both languages have similar functionality and are widely used in the industry.

i. Verilog is a hardware description language that was originally developed by Gateway Design Automation in the 1980s. It is a procedural language that allows designers to describe the behavior of digital circuits using a variety of constructs, including modules,

tasks, and functions. Verilog is widely used in the industry, especially for digital design and verification.

ii. VHDL is a hardware description language that was developed by the U.S. Department of Defense in the 1980s. It is a strongly-typed language that is based on Ada, a general-purpose programming language. VHDL allows designers to describe the behavior of digital circuits using a variety of constructs, including entities, architectures, and processes. VHDL is widely used in the industry, especially for ASIC and FPGA design.

2. Circuit Design:

- Circuit design is the process of creating electronic circuits by arranging and connecting transistors in a way that enables them to perform specific logic functions.

- Given a circuit design, we can estimate the delay and power. The circuit can be represented as a schematic, or in textual form as a netlist. Common netlist formats at the transistor level include Verilog and SPICE

- One of the goals of circuit design is to choose transistor widths to meet delay requirements.

- Because a transistor gate is a good insulator, it can be modeled as a capacitor, C.

- When the transistor is ON, some current I flows between source and drain. Both the current and capacitance are proportional to the transistor width.

- Energy is required to charge and discharge the load capacitance. This is called

dynamic power because it is consumed when the circuit is actively switching. The dynamic power consumed when a capacitor is charged and discharged at a frequency f is

$$P_{\text{dynamic}} = CV_{DD}^2 f$$

- Because an OFF transistor is leaky, a small amount of current Istatic flows between power and ground, resulting in a static power dissipation of

$$P_{\text{static}} = I_{\text{static}} V_{DD}$$

- Circuit designers often draw schematics at the transistor and/or gate level. For example in the figure below.

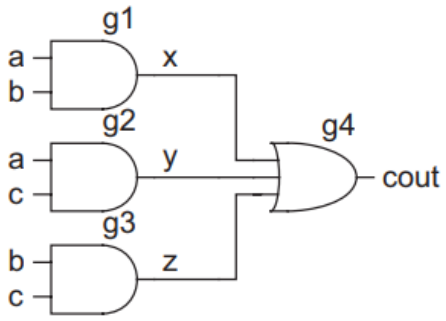


Fig.4. Schematic

- These schematics are then netlisted for simulation and verification. One common netlist format is structural Verilog HDL. The gate-level design can be netlisted as follows:

```
module alu(
      input          [3:0]     a,          //src1
      input          [3:0]     b,          //src2
      input          [2:0]     alu_control,     //function sel
      output     reg     [3:0]     result,          //result
      output zero
   );
 always @(*)  //fpga4student.com: FPga projects, Verilog projects, VHDL projects
 begin
      case(alu_control)
      3'b000: result = a + b; // add
      3'b001: result = a - b; // sub
      3'b010: result = a & b; // and
      3'b011: result = a | b; // or
      3'b100: begin if (a<b) result = 4'd1;
                    else result = 4'd0;
                    end
      default:result = a + b; // add
      endcase
 end
 assign zero = (result==4'd0) ? 1'b1: 1'b0;
 endmodule
```

Fig.5. Netlist

3. Physical Design:

a. Floorplanning:

- Floorplanning is a critical step in the physical design process of integrated circuits, where the overall layout of the chip is planned. The objective of floorplanning is to define the chip's physical boundaries and allocate the available area to different functional blocks of the design, such as the processor core, memory, I/O interface, and clocking circuitry.

- During floorplanning, the designer must consider various constraints such as power, signal integrity, area, and timing requirements. The placement of functional blocks and their connections must be carefully planned to minimize the signal delay and power consumption while ensuring that all timing requirements are met.

- Floorplanning is a crucial step in the physical design process, as it sets the foundation for subsequent design steps such as placement and routing. A well-planned floorplan can lead to a more efficient and optimized chip layout, which can result in better performance, reduced power consumption, and lower overall manufacturing costs.

- Example: Figure below shows the chip floorplan for the MIPS processor including the pad frame. The top-level blocks are the controller and datapath. A wiring channel is located between the two blocks to provide room to route control signals to the datapath. The datapath is further partitioned into wordslices. The pad frame includes 40 I/O pads, which are wired to the pins on the chip package. There are 29 pads used for signals; the remainder are VDD and GND.

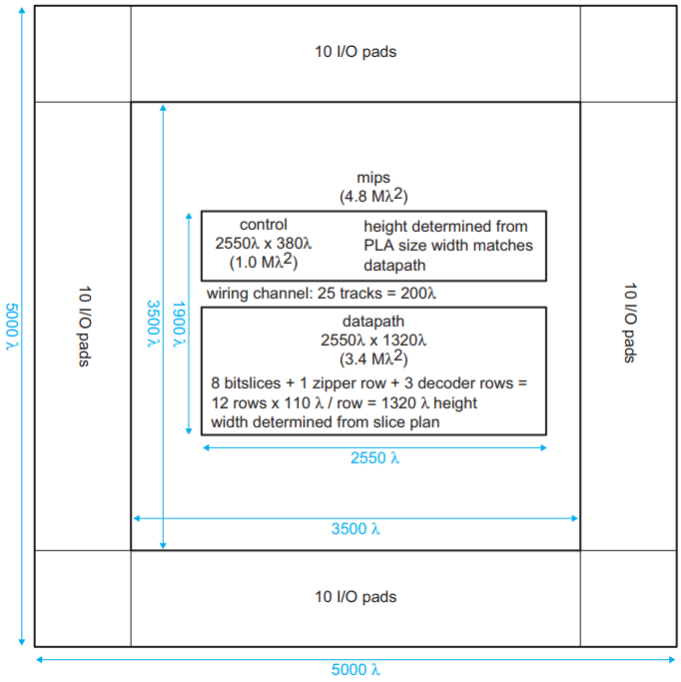


Fig.6. MIPS floor planning

- On-chip structures can be categorized as random logic, datapaths, arrays, analog, and

input/output (I/O).

- Automatic layout generators exist for memory arrays and random logic but are not as mature for datapaths. Therefore, many design methodologies ignore the potential structure of datapaths and instead lay them out with random logic tools except when performance or area are vital. Analog circuits still require careful design and simulation but tend to involve only small amounts of layout because they have relatively few transistors. I/O cells are also highly tuned to each fabrication process and are often supplied by the process vendor. Random logic and datapaths are typically built from standard cells such as inverters, NAND gates, and flip-flops. Standard cells increase productivity because each cell only needs to be drawn and verified once. Often, a standard cell library is purchased from a third party vendor.

- Another important decision during floorplanning is to choose the metal orientation.

b. Standard Cells:

- In physical design, standard cells refer to pre-designed, pre-verified, and pre-characterized logic cells that are used as building blocks in the layout of integrated circuits. Standard cells are typically designed to be modular and have a fixed height and width, making them easy to integrate into a larger chip design. By using standard cells, designers can save time and effort in the design process, as they do not need to design each logic gate from scratch. Standard cells consist of a collection of logic gates, such as AND, OR, NAND, NOR, XOR, and flip-flops, which are commonly used in digital circuitry.

- Standard cells are designed to meet specific design rules and layout guidelines, and they are characterized to provide information about their electrical and timing characteristics. This information is used in the design process to ensure that the cells meet performance and power consumption requirements and to optimize the placement and routing of the cells in the chip layout.

- Standard cells are an important component of the physical design process, as they enable efficient and predictable chip layout and can help to reduce design time and manufacturing costs. Standard cell libraries are typically provided by semiconductor foundries or third-

party vendors and are available in a range of process technologies and cell libraries optimized for different applications and performance requirements.

c. Pitch Matching

- Pitch matching is a technique used in physical design to ensure that adjacent standard cells in a chip layout have the same height and width. This technique is important because it helps to maintain a regular and uniform layout structure, which can reduce the risk of timing and noise issues in the final chip.

- In pitch matching, the height and width of adjacent standard cells are matched by adjusting the spacing between them. This spacing, known as the "pitch," is defined as the distance between the centers of two adjacent cells. By matching the pitch, designers can ensure that the cells align properly and that the interconnects between them have consistent lengths, which can help to minimize signal delay and noise.

- Pitch matching is especially important in designs that use deep submicron technologies, where variations in the manufacturing process can result in significant differences in the electrical characteristics of adjacent cells. By ensuring that adjacent cells have the same dimensions and spacing, pitch matching can help to reduce the impact of these variations and improve the overall performance and yield of the chip.

- Pitch matching is typically performed as part of the floorplanning and placement phases of the physical design process. Designers may use specialized tools and techniques to optimize the pitch matching and ensure that the final layout meets the required performance and manufacturability targets.

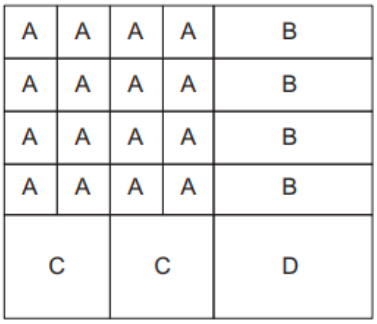


Fig.7. Pitch-matching of snap-together cells

For Example: Figure below shows the MIPS datapath in more detail. The eight horizontal bit slices are clearly visible. The zipper at the top of the layout includes three rows for the decoder that is pitch-matched to the register file in the datapath

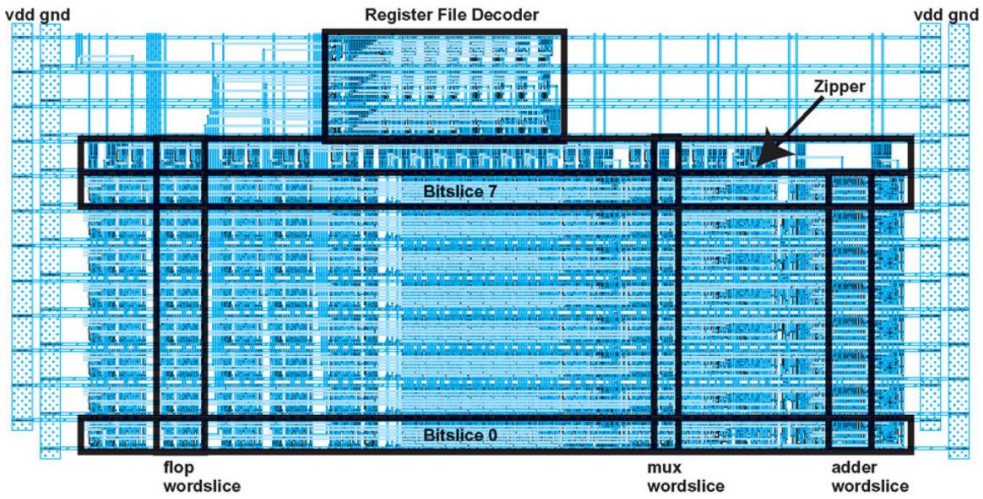


Fig.8. MIPS datapath layout

d. Slice Plans:

- In physical design, a slice plan is a high-level diagram that shows the partitioning of a chip into smaller rectangular regions, known as slices. Each slice is typically designed to contain a specific set of functional blocks, such as a memory array or a processing unit, and the interconnects needed to connect these blocks to the rest of the chip.

Example: Figure below shows a slice plan of the datapath. The diagram illustrates the ordering of wordslices and the allocation of wiring tracks within each bitslice. Dots indicate that a bus passes over a cell and is also used in that cell. Each cell is annotated with its type and width (in number of tracks).

Slice plan makes it easy to calculate wire lengths and evaluate wiring congestion before laying out the datapath. In this case, it is evident that the greatest congestion takes place over the register file, where seven wiring tracks are required.

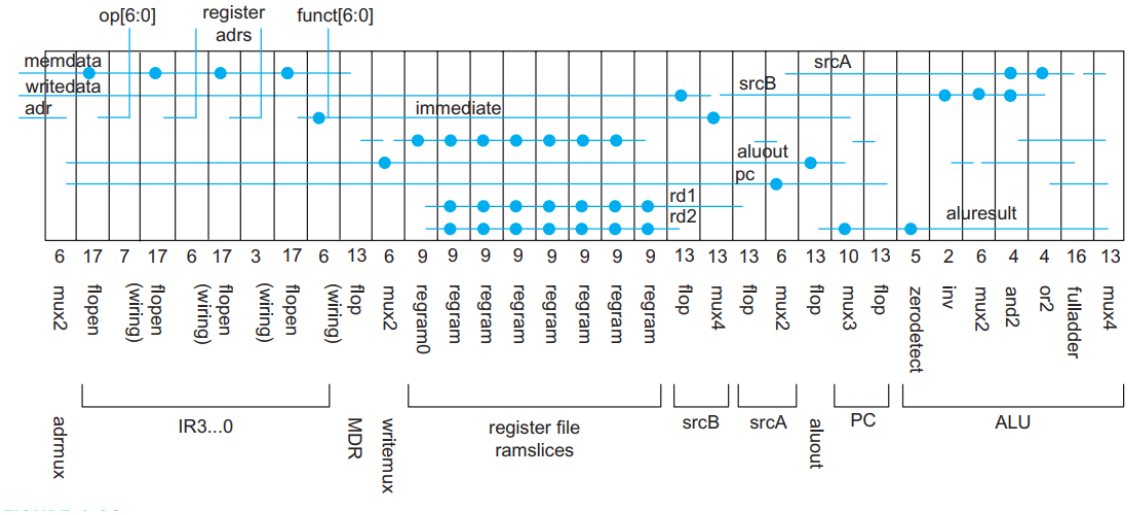


Fig.9. Slice plan of the datapath

e. Arrays:

- A programmable logic array (PLA) used for the control FSM next state and output logic.

- Any logic function can be expressed in sum-of-products form; i.e., where each output is the OR (sum) of the ANDs (products) of true and complementary inputs. The inputs and their complements are called literals. The AND of a set of literals is called a product or minterm. The outputs are ORs of minterms. The PLA consists of an AND plane to compute the minterms and an OR plane to compute the outputs

- The structure on the left is called the AND plane and the structure on the right is the OR

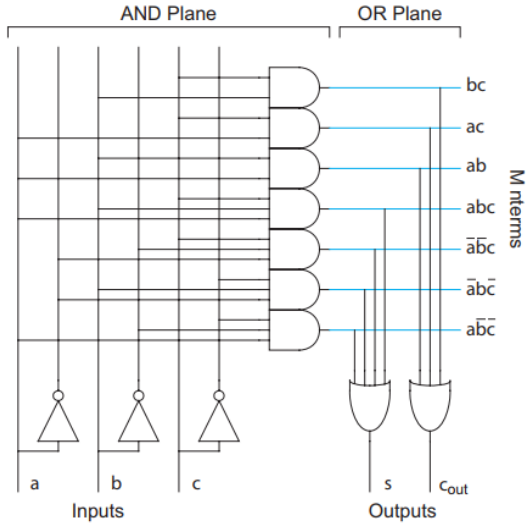


Fig.10. PLA for control FSM

f. Area Estimation:

- Area estimation is an important aspect of physical design in the field of microelectronics. It involves estimating the amount of physical space that will be required on a chip to implement a given digital circuit or system. Accurate area estimation is critical for optimizing chip performance, reducing production costs, and ensuring that the chip can be manufactured within the physical limitations of the fabrication process. In the absence of data for such comparison, the Table below lists some typical numbers.

Element	Area
random logic (2-level metal process)	$1000 - 1500\ \lambda^2$ / transistor
datapath	$250 - 750\ \lambda^2$ / transistor or $6\ WL + 360\ \lambda^2$ / transistor
SRAM	$1000\ \lambda^2$ / bit
DRAM (in a DRAM process)	$100\ \lambda^2$ / bit
ROM	$100\ \lambda^2$ / bit

Table. Typical layout densities

4. Design Verification

- Because integrated circuits are so complex, everything that can go wrong will almost certainly go wrong. Design verification is critical for discovering problems before they are manufactured, and it often accounts for half or more of the effort put into a chip.

- Verification time grows as design representations get more comprehensive. It is impractical to model a whole chip for a significant number of cycles in a circuit-level simulator such as SPICE to ensure that the layout is right. Instead, the design is often verified for functioning at the architectural level using a model written in a programming language such as C, and at the logic level by replicating the HDL description. The circuits are next reviewed to verify that they are an accurate representation of the logic, and the

layout is tested to ensure that it is an accurate representation of the circuits, as illustrated in Figure below. Timing and power criteria must also be met by the circuits and layout.

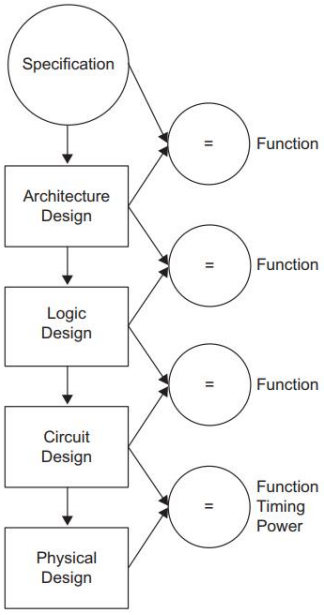


Figure 11 Design and verification sequence

B. Practical Part

- In this project, we will design ALU module of MIPS processor following the VLSI design flow.

I. Requirement Specification:

- Technology: 90nm
- Frequency: >= 0.5 GHz.
- Power: <= 5 mW.
- Area: <= 4500 µm^2
- Databus: 4 bit

II. Logic Design:

- In this step, the structure of the desired design is added to the behavioral representation of the desired design. The main specifications to be considered for logic design are logic minimization, performance enhancement, and testability.

- To do this, we use LTSpice software, draw transistor levels for each unit, each component

1. Top-level Interfaces:

Inputs	Outputs
a[3:0]	zero
b[3:0]	result[3:0]
alu_control[2:0]	

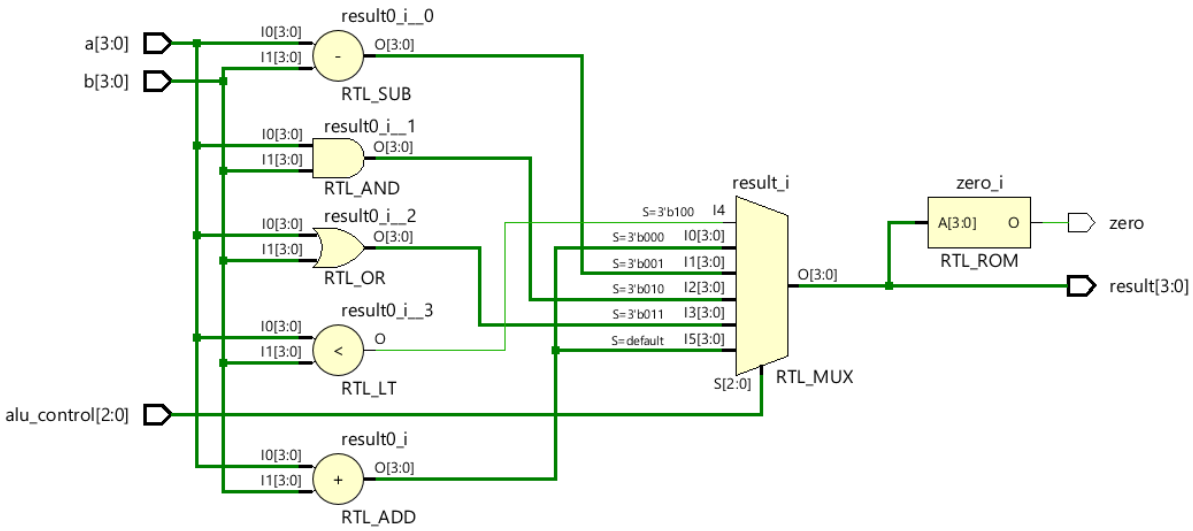
2. Verilog:

Alu.v

D:/19ECE/EE477/ALU/ALU.srcs/sources_1/new/Alu.v

```
1  //fpga4student.com: FPga projects, Verilog projects, VHDL projects
2  // Verilog code for ALU
3  module alu(
4        input          [3:0]     a,          //src1
5        input          [3:0]     b,          //src2
6        input          [2:0]     alu_control,     //function sel
7        output     reg     [3:0]     result,          //result
8        output zero
9     );
10 always @(*)  //fpga4student.com: FPga projects, Verilog projects, VHDL projects
11 begin
12       case(alu_control)
13       3'b000: result = a + b; // add
14       3'b001: result = a - b; // sub
15       3'b010: result = a & b; // and
16       3'b011: result = a | b; // or
17       3'b100: begin if (a<b) result = 4'd1;
18                     else result = 4'd0;
19                     end
20       default:result = a + b; // add
21       endcase
22 end
23 assign zero = (result==4'd0) ? 1'b1: 1'b0;
24 endmodule
```

3. Schematic:



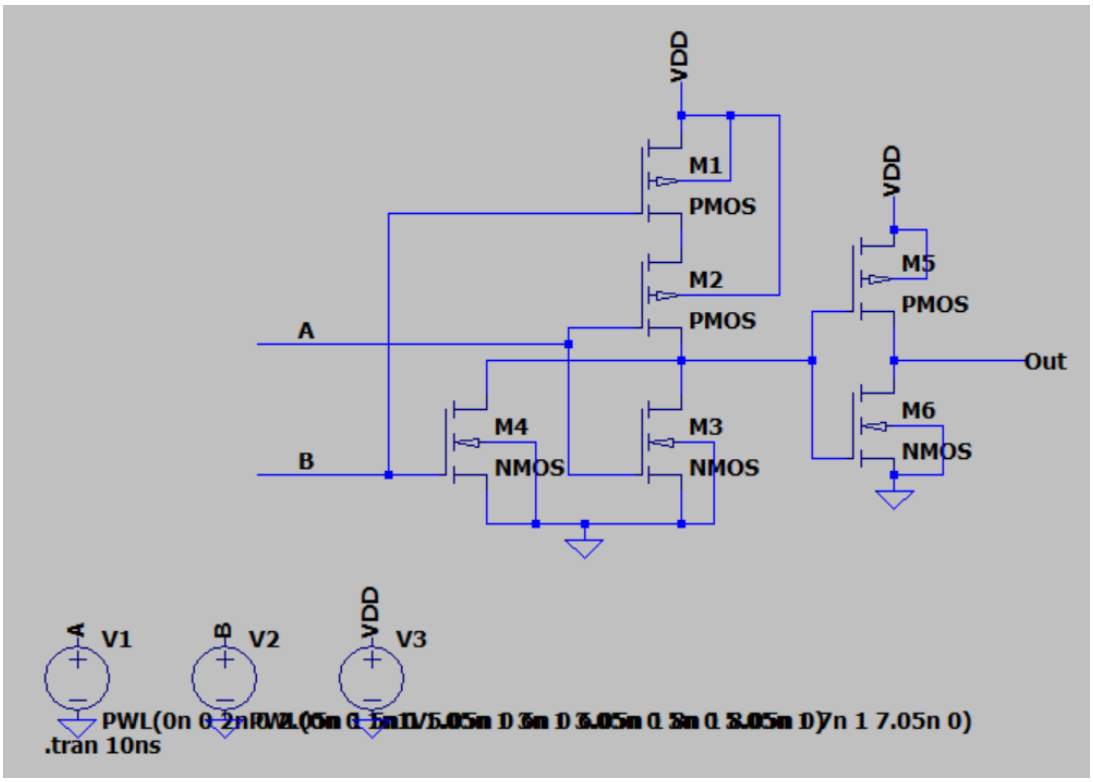
III. Circuit Design:

- In this step, the logic blocks of the desired design are replaced by the electronic circuits, which are consists of electronic devices such as resistors, capacitors, and transistors. Circuit simulation of the desired design is done at this stage, in order to verify the timing behavior of the desired system.

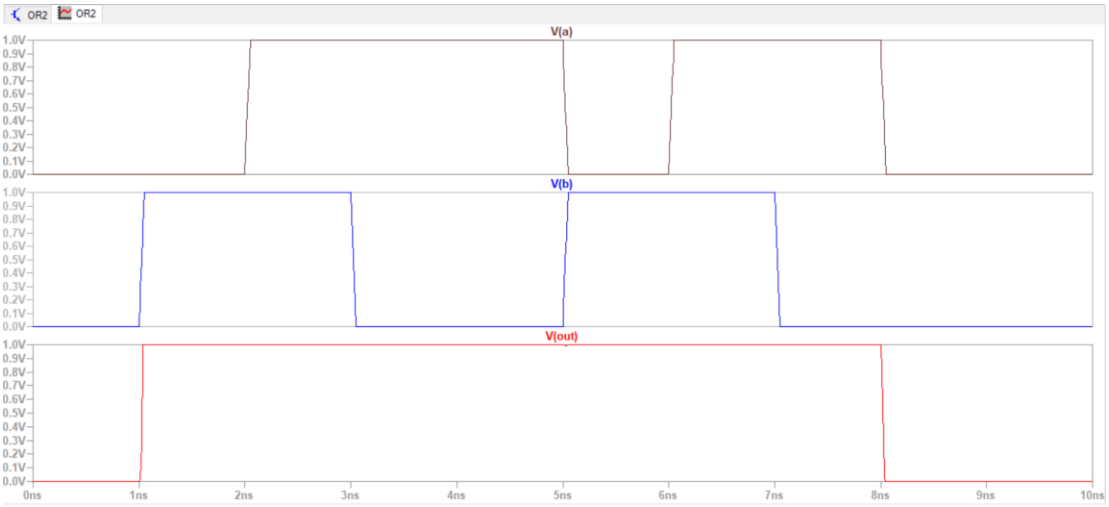
1. Circuit Schematic Design:

a. 2-input OR gate:

i. Schematic:



ii. Waveform:



iii. Power calculation:

-Dynamic power:

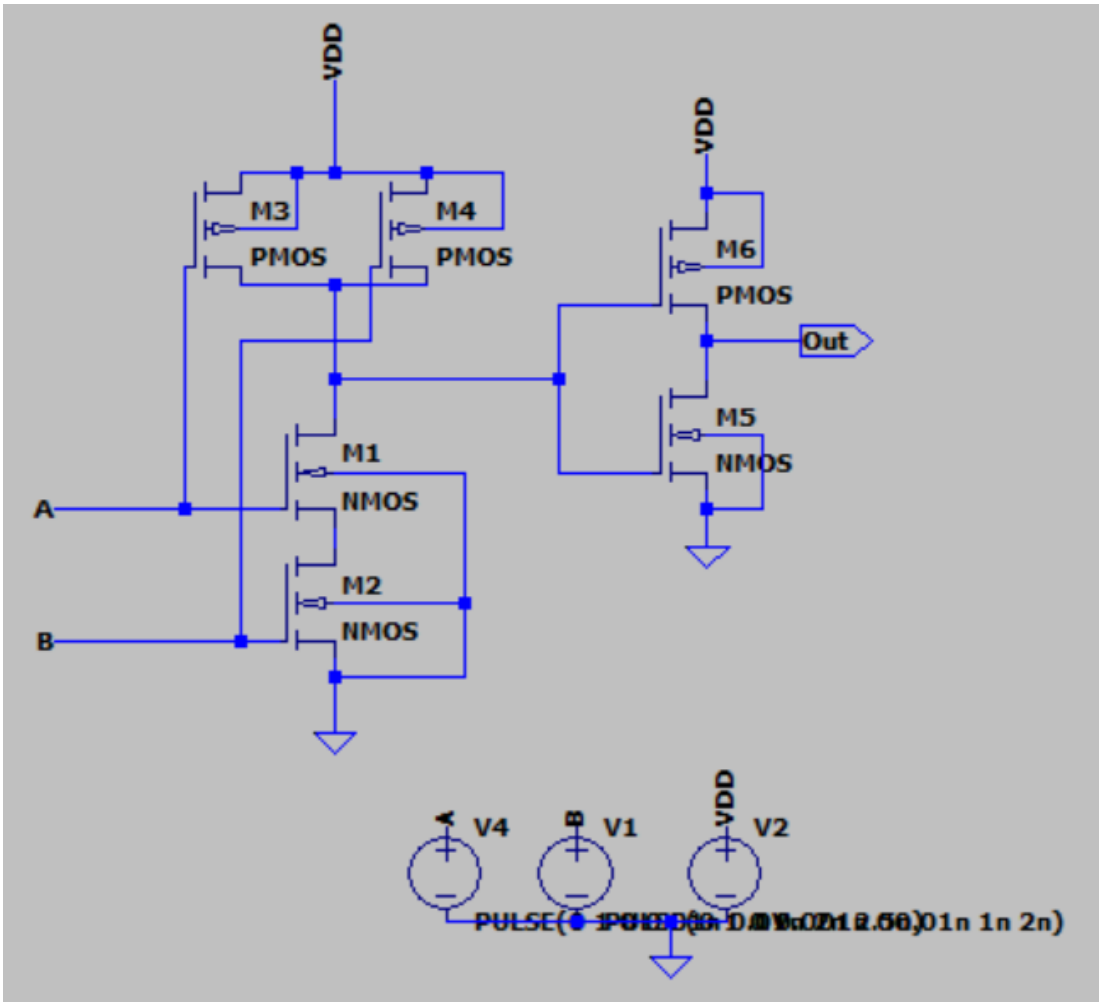
$$P_{dynamic} = V_{dd} \times \frac{I_{ave} - I_{leakage}}{Frequency} = 1V \times \frac{55.5nA - 6.1pA}{100MHz} = 0.55 \times 10^{-3} uW/MHz$$

-Leakage power:

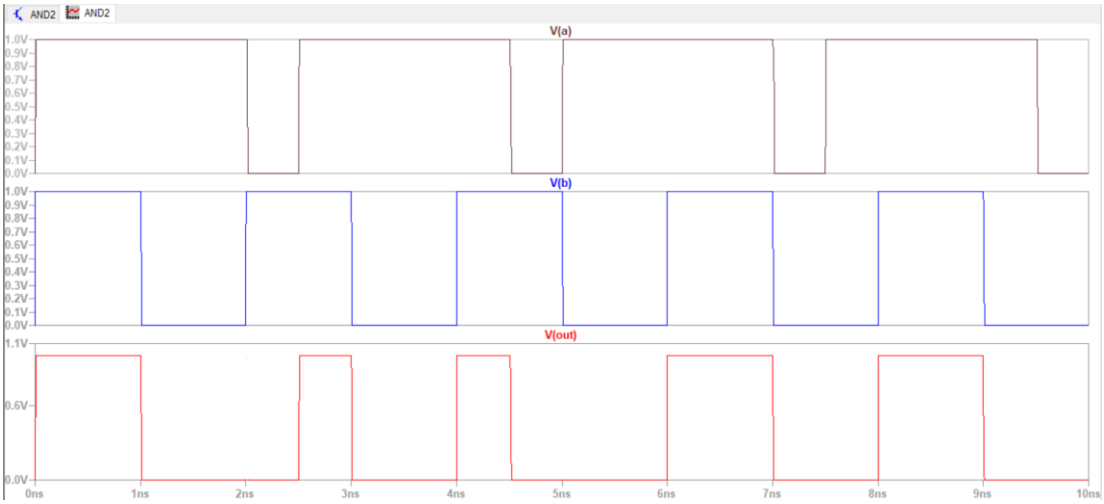
$$P_{leakage} = V_{dd} \times I_{leakage} = 1V \times 6.1pA = 6.1pW$$

b. 2-input AND gate:

i. Schematic:



ii. Waveform:



iii. Power calculation:

-Dynamic power:

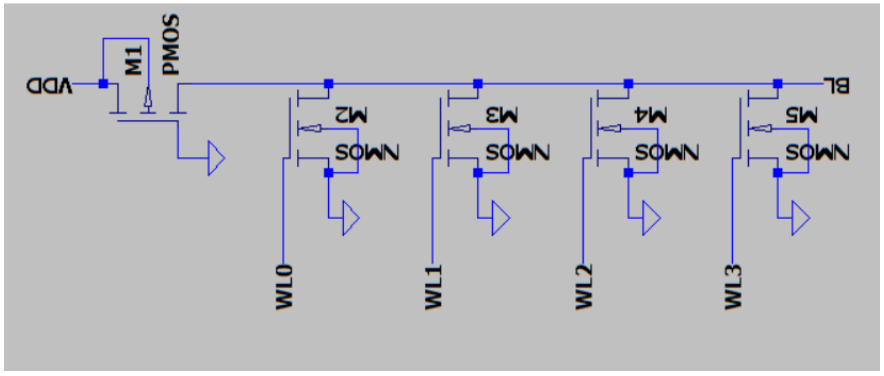
$$P_{dynamic} = V_{dd} \times \frac{I_{ave} - I_{leakage}}{Frequency} = 1V \times \frac{34.2nA - 3.8pA}{100MHz} = 0.34 \times 10^{-3} uW/MHz$$

-Leakage power:

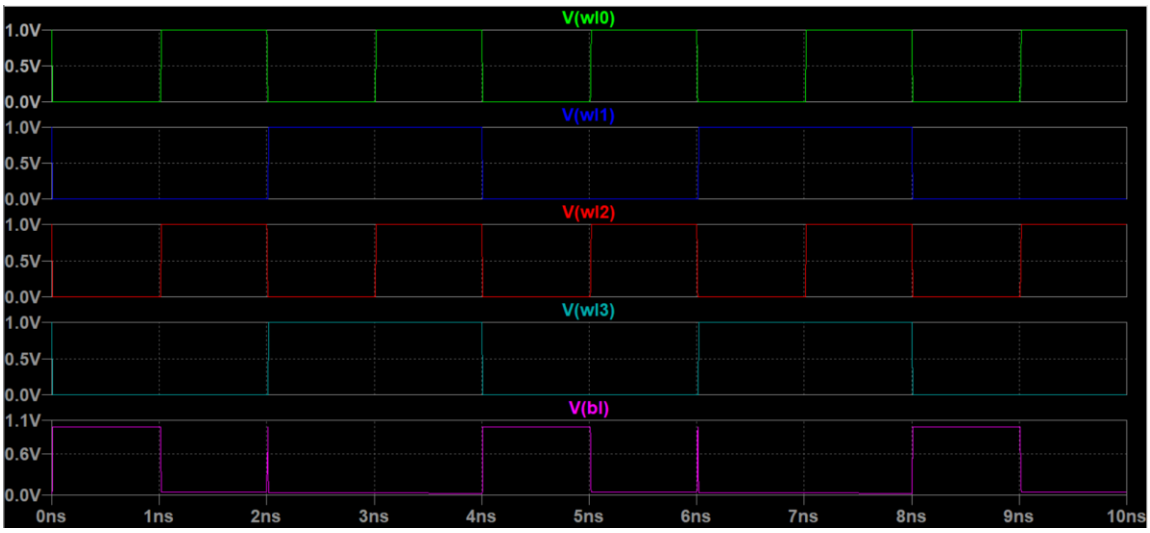
$$P_{leakage} = V_{dd} \times I_{leakage} = 1V \times 3.8pA = 3.8pW$$

c. ROM:

i. Schematic:



ii. Waveform:



iii. Power calculation:

-Dynamic power:

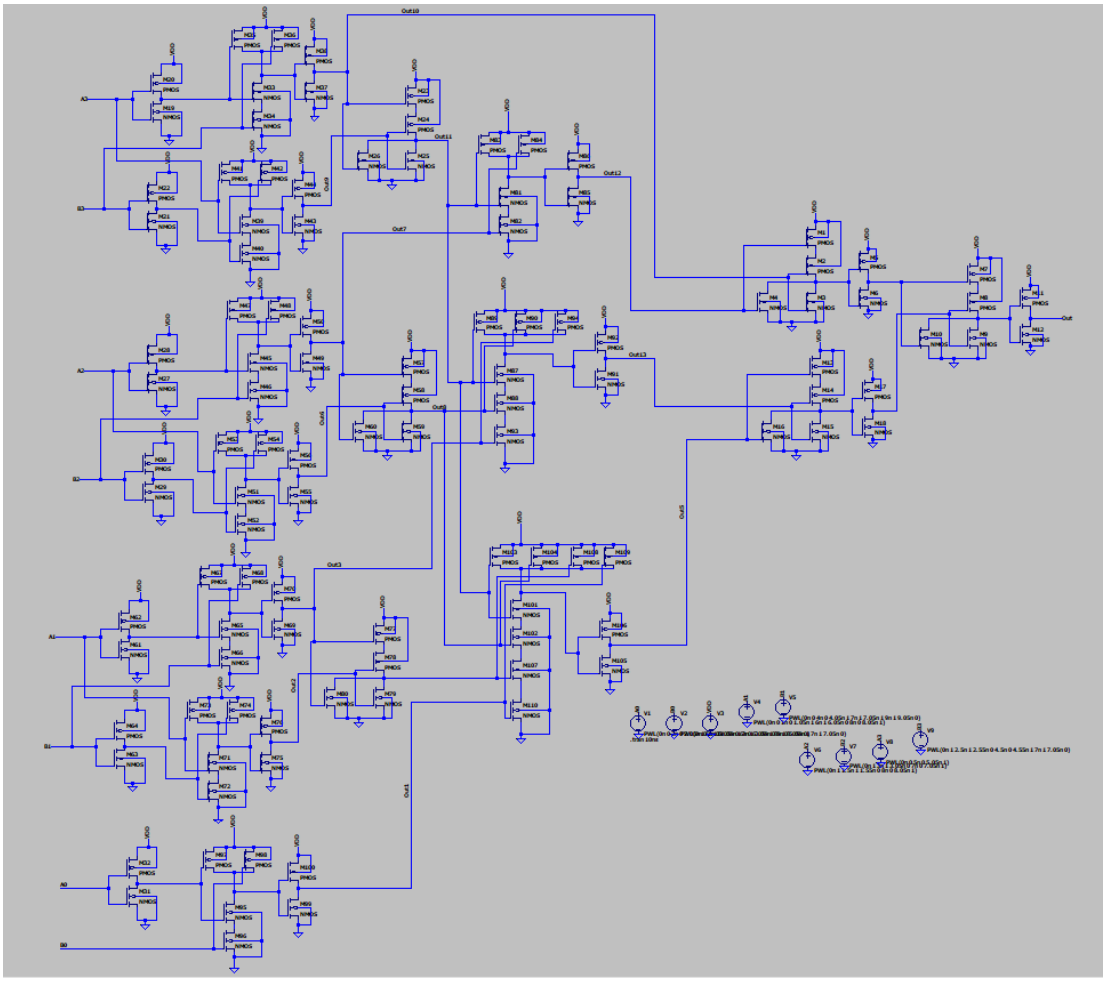
$$P_{dynamic} = V_{dd} \times \frac{I_{ave} - I_{leakage}}{Frequency} = 1V \times \frac{5.06\ uA - 8.04\ pA}{500MHz} = 1.01 \times 10^{-12}\ pW/Hz$$

-Leakage power:

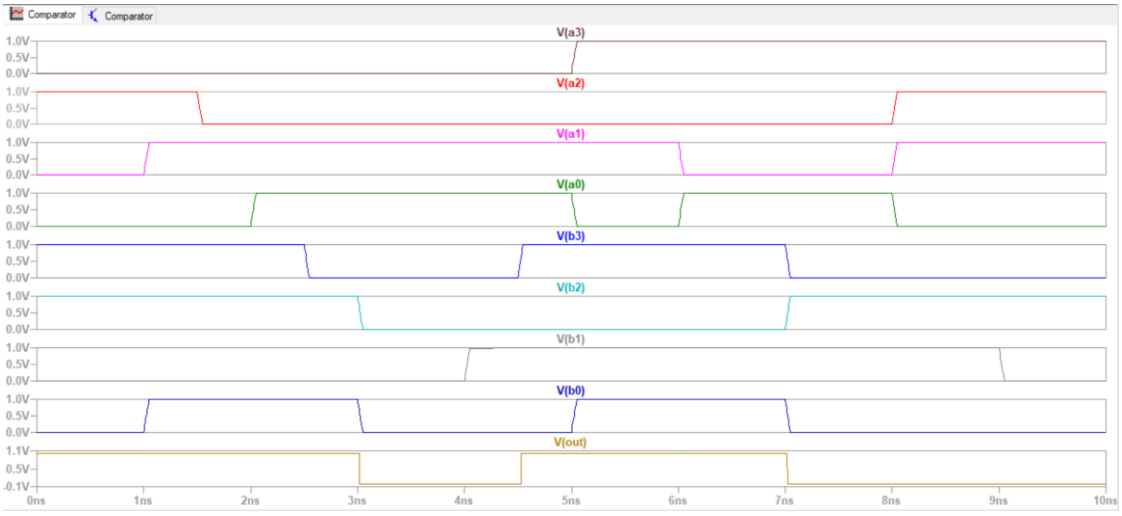
$$P_{leakage} = V_{dd} \times I_{leakage} = 1V \times 8.04\ pA = 8.04\ pW$$

d. Comparator (Less than):

i. Schematic:



ii. Waveform:



iii. Power calculation:

-Dynamic power:

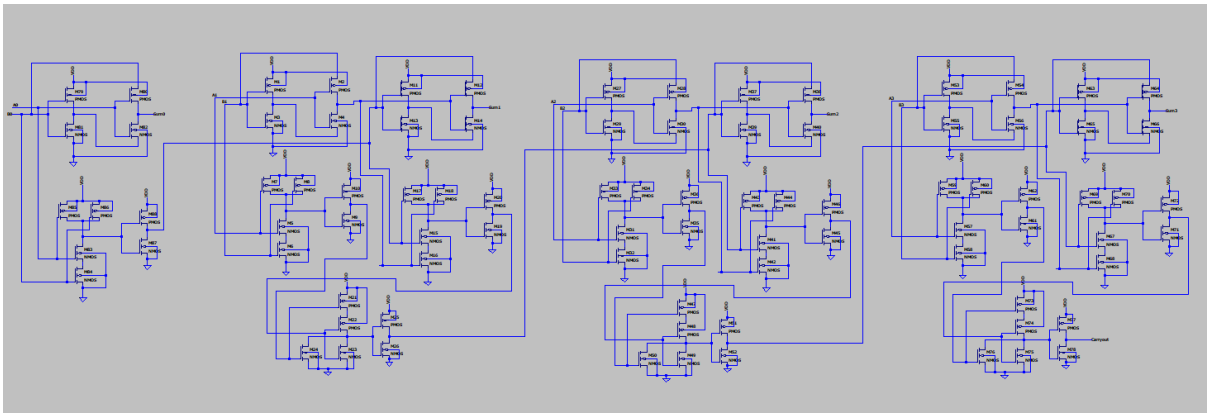
$$P_{dynamic} = V_{dd} \times \frac{I_{ave} - I_{leakage}}{Frequency} = 1V \times \frac{541.35nA - 87.4pA}{100MHz} = 5.4 \times 10^{-3} uW/MHz$$

-Leakage power:

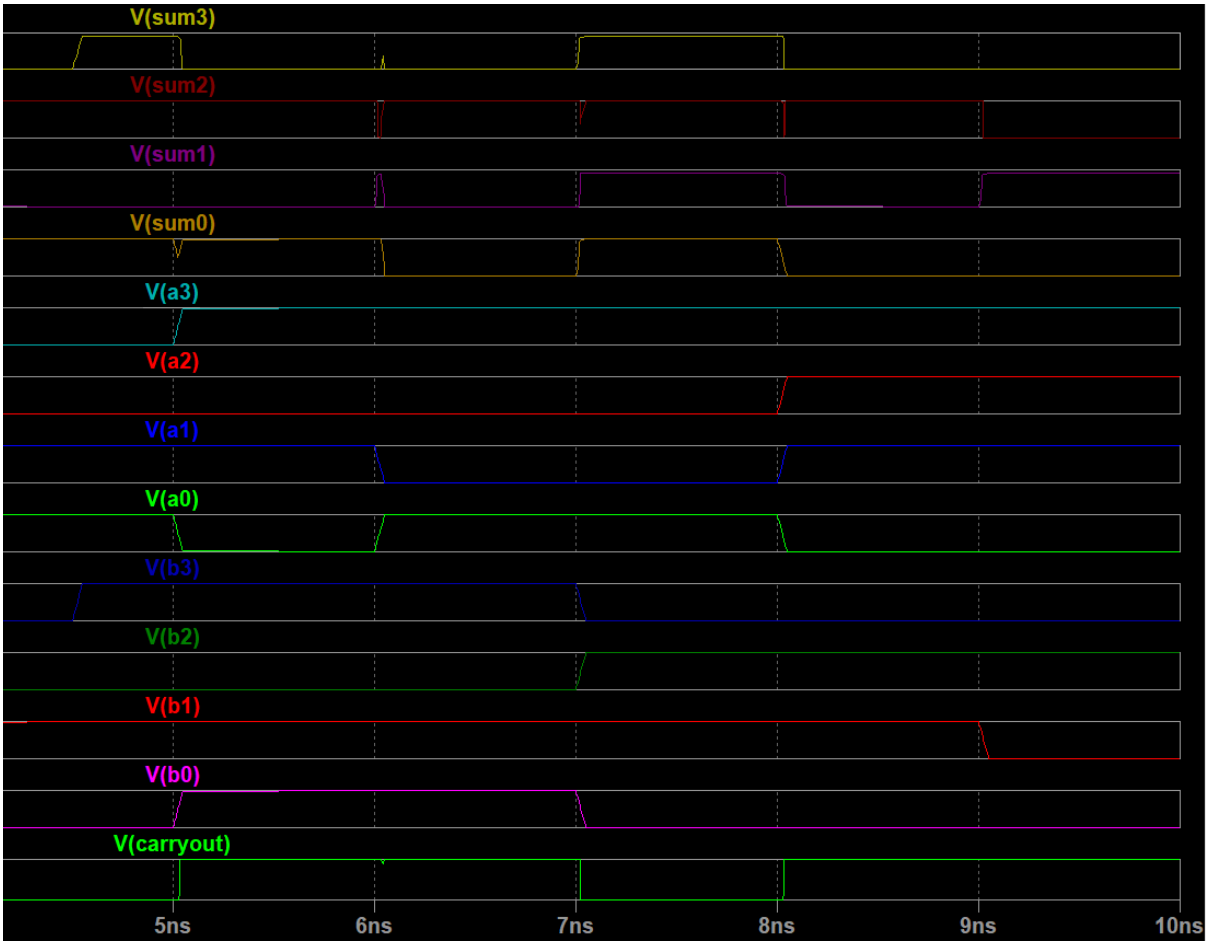
$$P_{leakage} = V_{dd} \times I_{leakage} = 1V \times 87.4pA = 87.4pW$$

e. Full-adder:

i. Schematic:



ii. Waveform:



iii. Power calculation:

-Dynamic power:

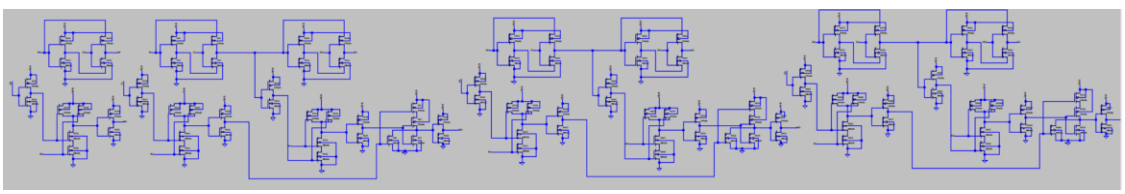
$$P_{dynamic} = V_{dd} \times \frac{I_{ave} - I_{leakage}}{Frequency} = 1V \times \frac{82.13uA - 107.38pA}{100MHz} = 0.82uW/MHz$$

-Leakage power:

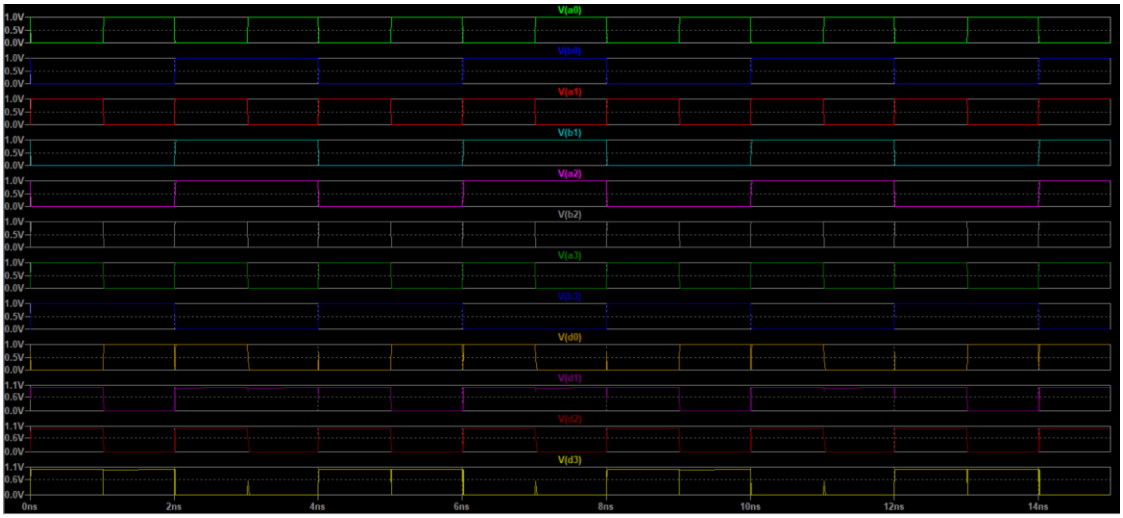
$$P_{leakage} = V_{dd} \times I_{leakage} = 1V \times 107.38pA = 107.31pW$$

f. Full-subtractor:

i. Schematic:



ii. Waveform:



iii. Power calculation:

-Dynamic power:

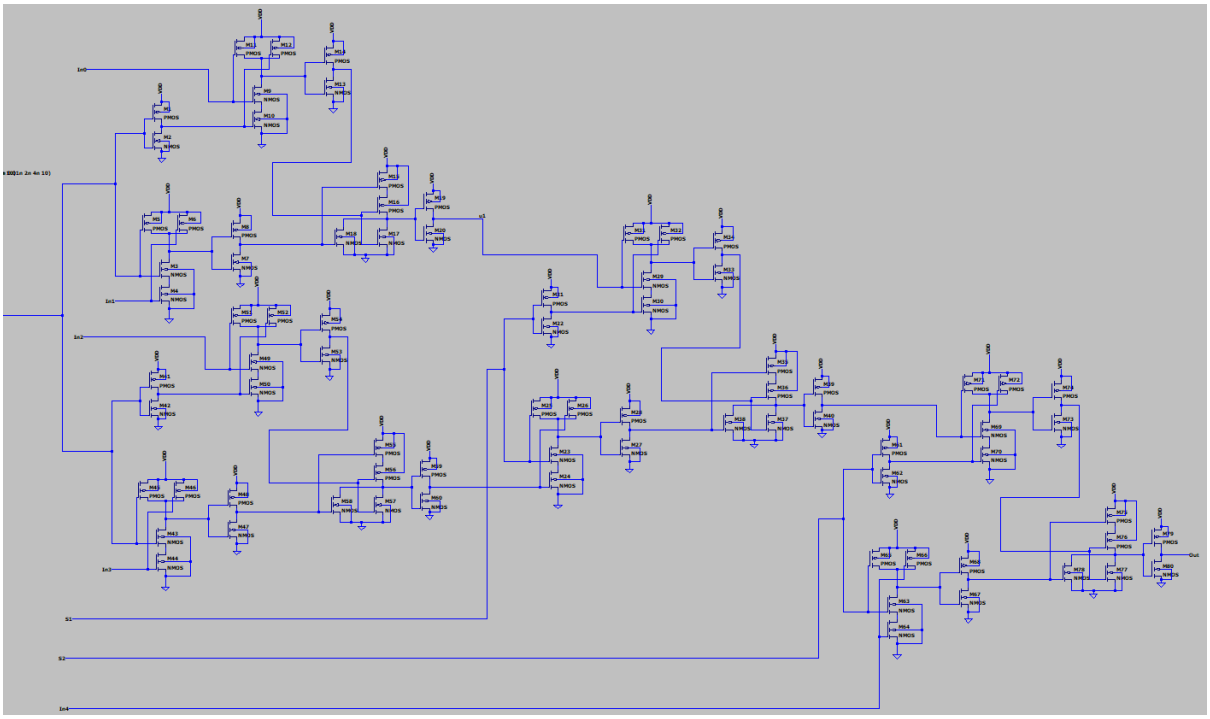
$$P_{dynamic} = V_{dd} \times \frac{I_{ave} - I_{leakage}}{Frequency} = 1V \times \frac{723.08nA - 121.77pA}{500MHz} = 1.44 \times 10^{-3} pW/Hz$$

-Leakage power:

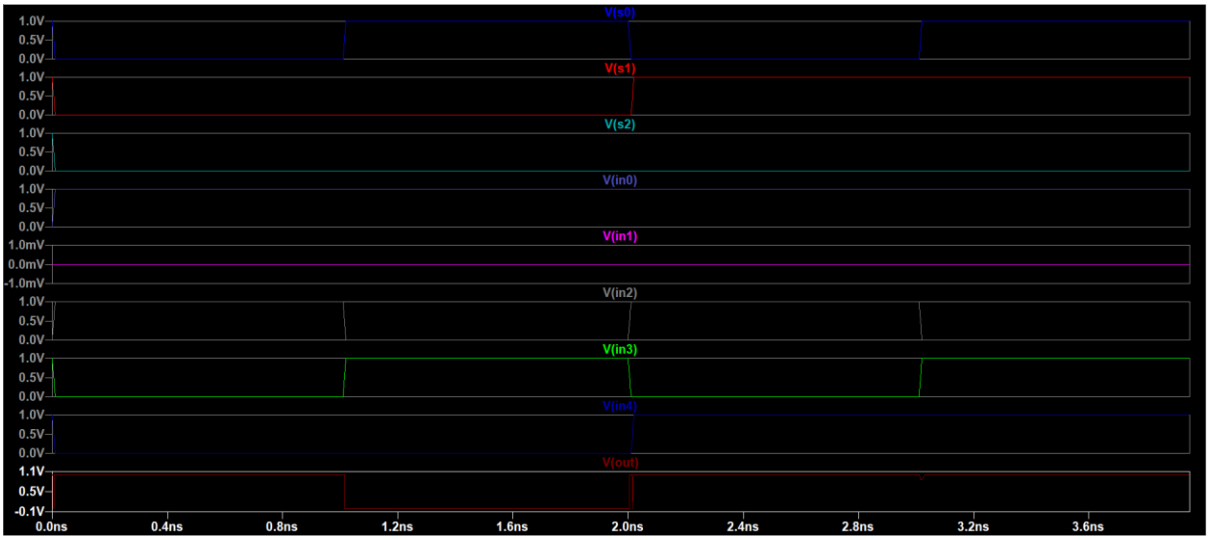
$$P_{leakage} = V_{dd} \times I_{leakage} = 1V \times 121.77pA = 121.77\ pW$$

g. Mux 5-1:

i. Schematic:



ii. Waveform:



iii. Power calculation:

-Dynamic power:

$$P_{dynamic} = V_{dd} \times \frac{I_{ave} - I_{leakage}}{Frequency} = 1V \times \frac{245.25\ nA - 69.89\ pA}{500MHz} = 4.9 \times 10^{-1}\ fW/Hz$$

-Leakage power:

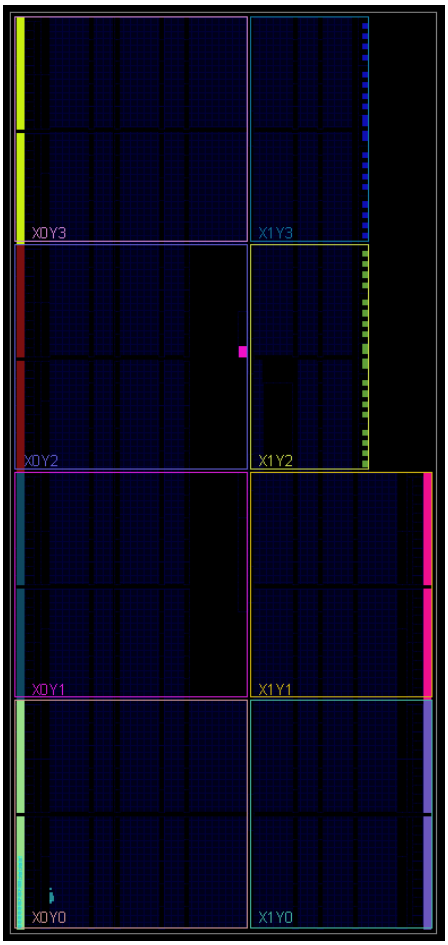
$$P_{leakage} = V_{dd} \times I_{leakage} = 1V \times 69.89\ pA = 69.89\ pW$$

IV. Physical Design:

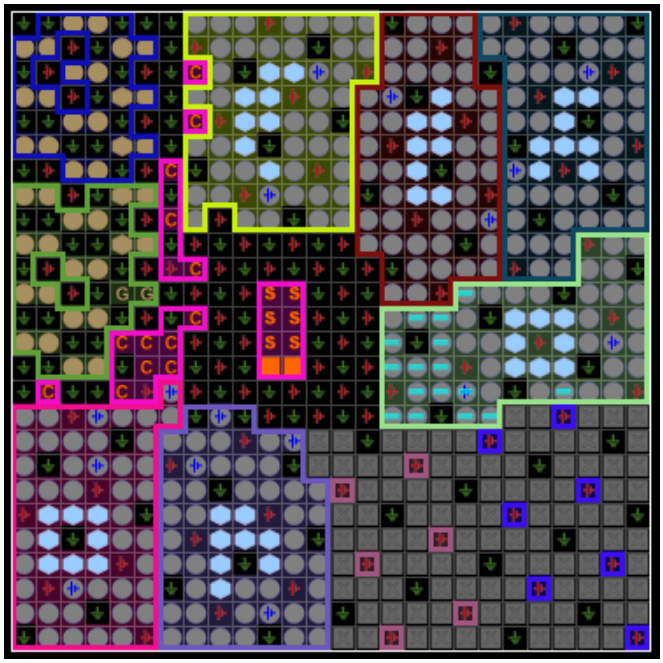
Our priorities for the physical design are as follows, ranked from highest to lowest:

- Priority 1: functionality.
  Since our design is fairly complex, our highest priority is to have a functioning design that operates correctly as we intended. This means that all operations: add, subtract, and, or, and compare (less than) should produce the correct result.
- Priority 2: power consumption.
  For our second priority, we want to reduce the power consumption of the ALU as much as possible, since we find our design to be simpler to optimize for power. To reduce power consumption, we will try different methods that mainly concerns reducing the area of the circuit.
- Priority 3: speed.
  Finally, we want the design to operate quickly and accurately. Hence, we will try to reduce the delay by various methods, such as sizing, input reordering, delay matching, etc.

1. Floorplanning:



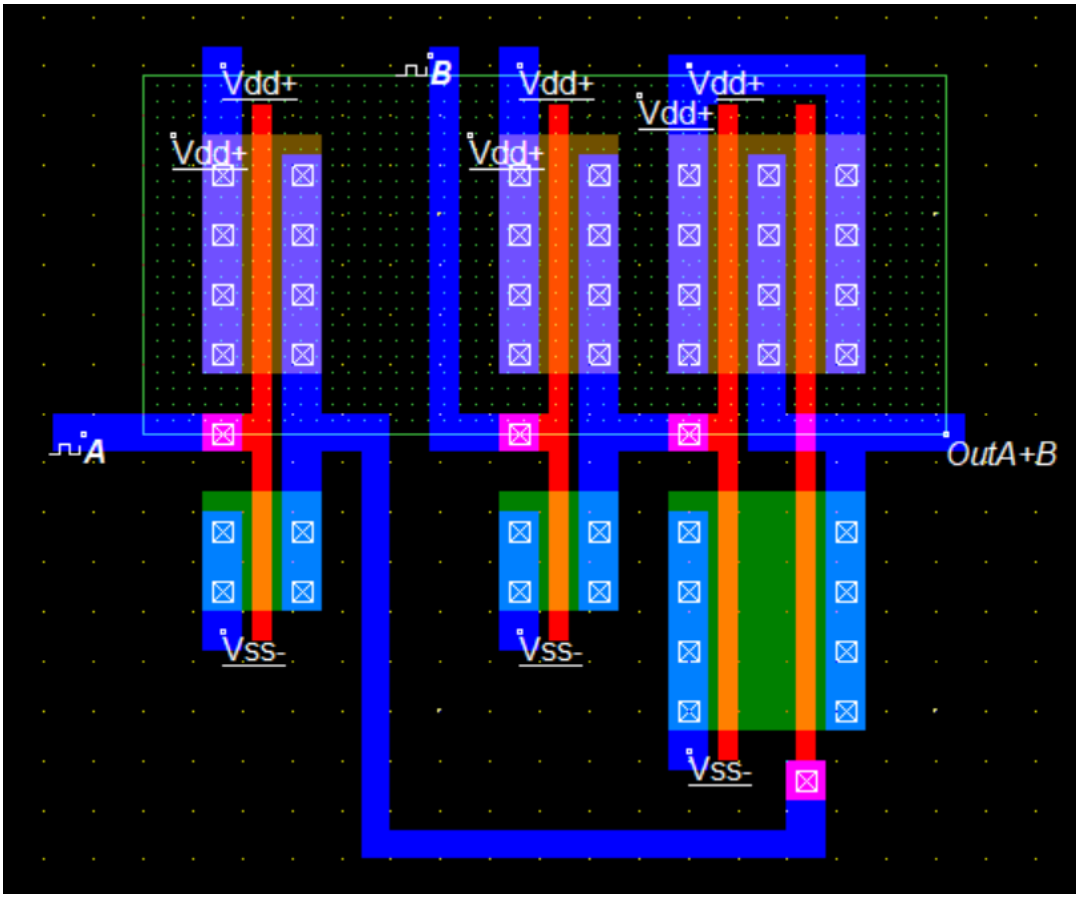
2. I/O Planning:



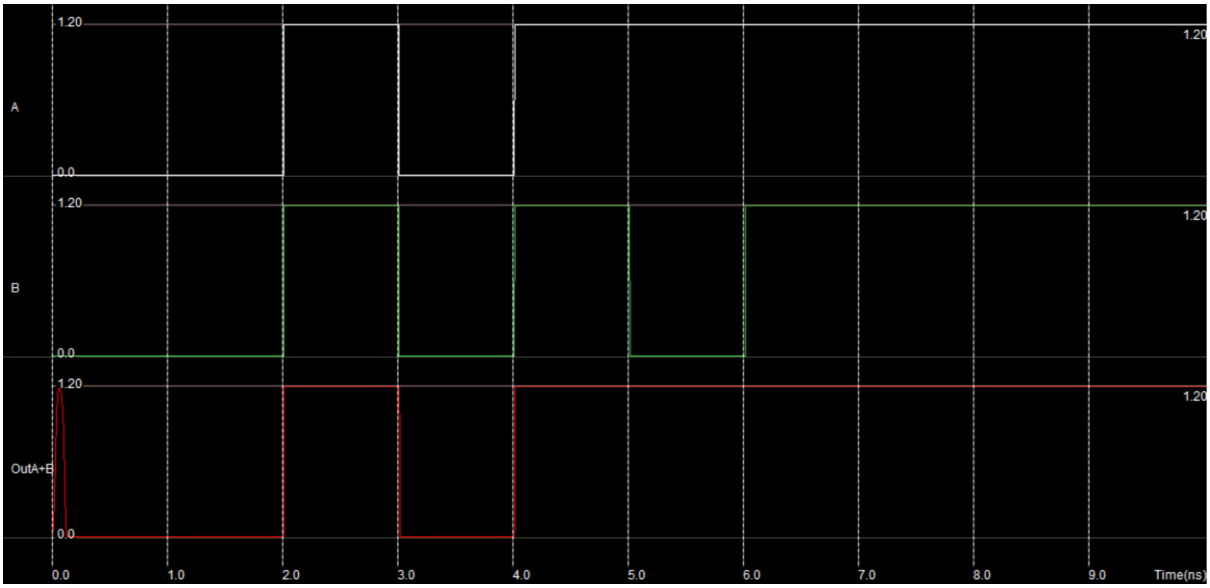
3. Layout:

a. 2-input OR gate:

i. Layout:

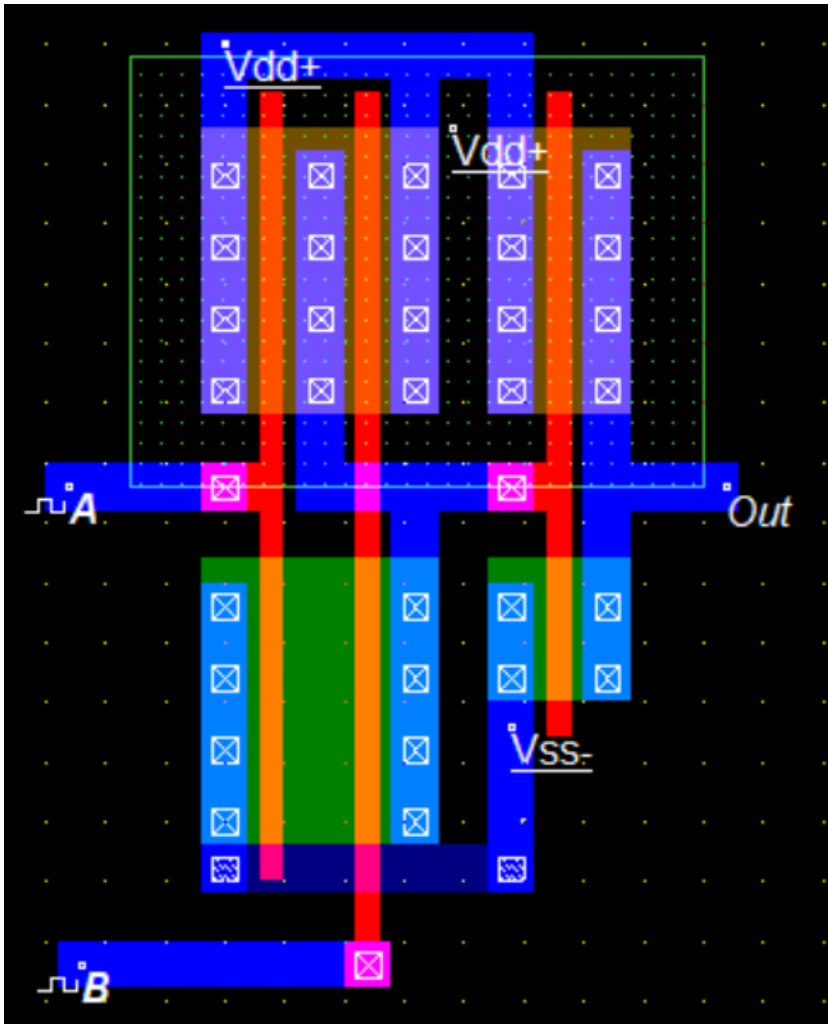


ii. Waveform:

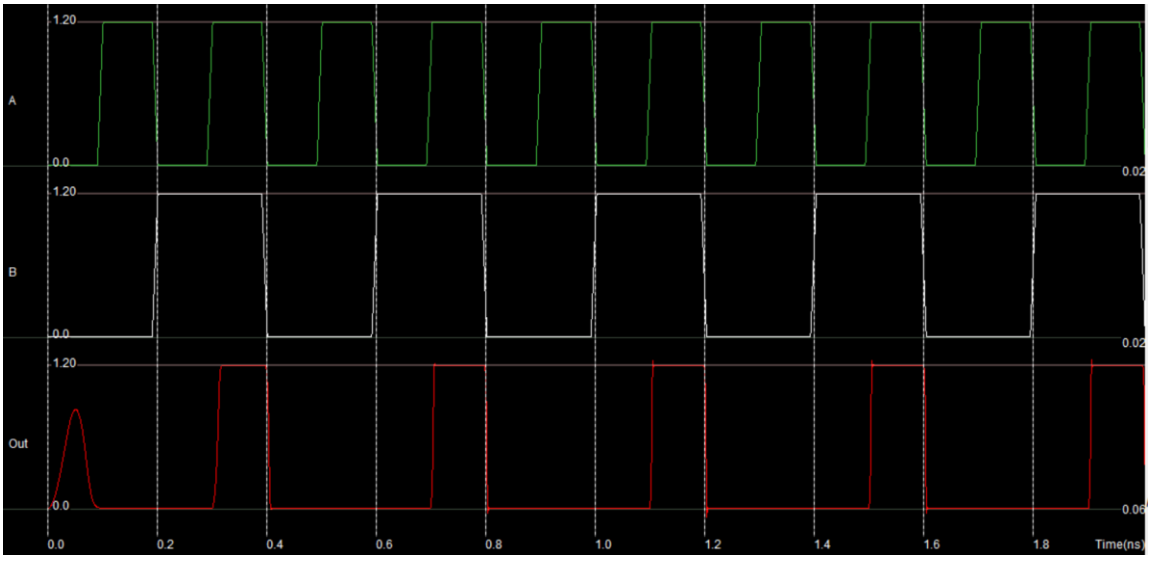


b. 2-input AND gate:

i. Layout:

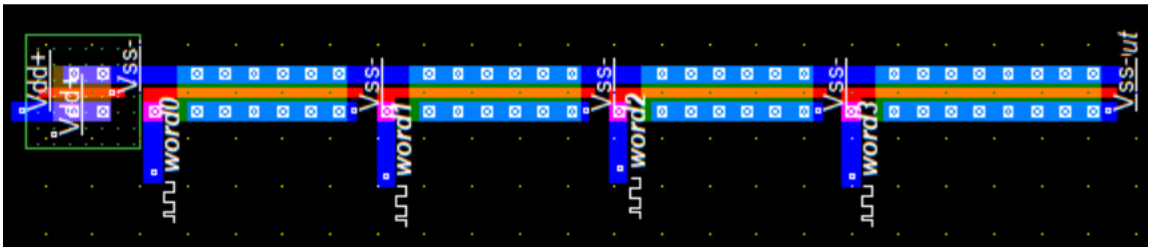

Vdd+
Vdd+
A
Out
Vss-
B

ii. Waveform:

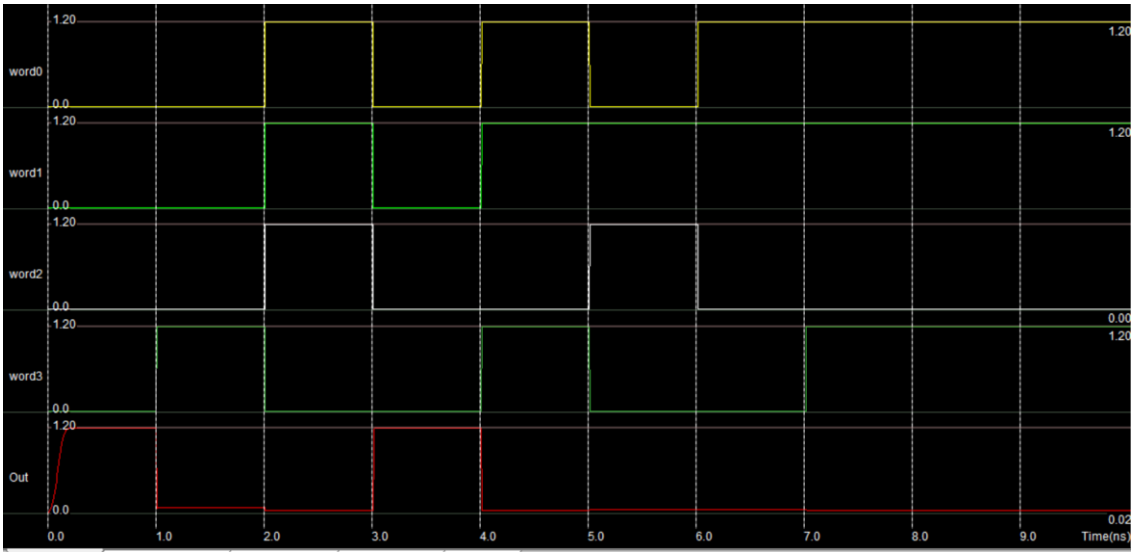

1.20
A
0.0
1.20
B
0.0
1.20
Out
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
Time(ns)

c. ROM:

i. Layout:

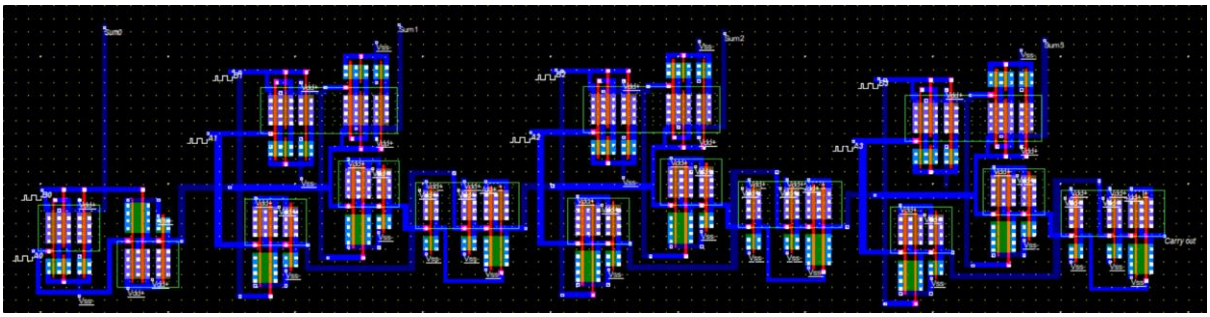


ii. Waveform:

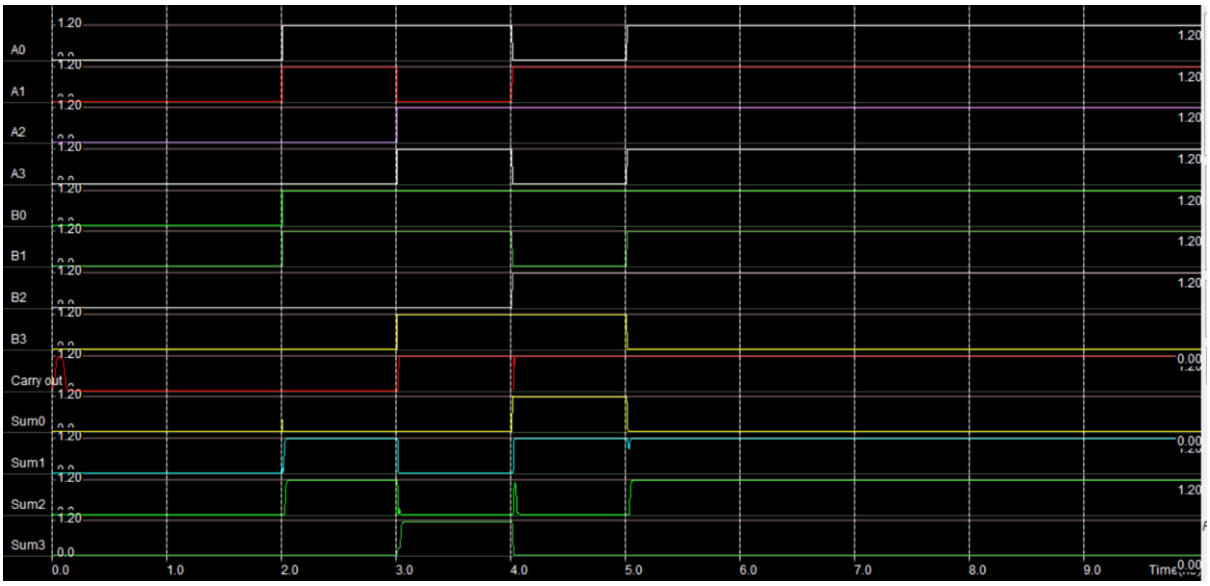


d. Full-adder:

i. Layout:

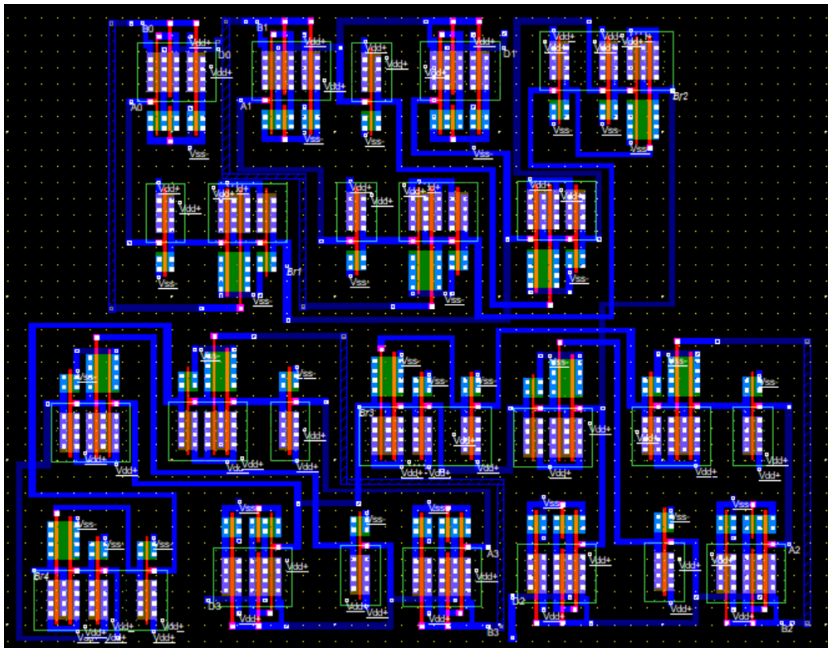


ii. Waveform:

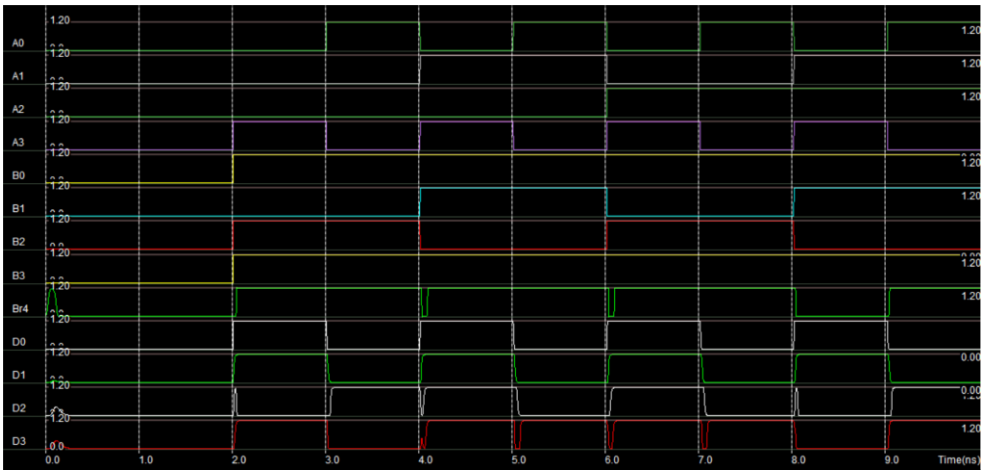


e. Full-subtractor:

i. Layout:

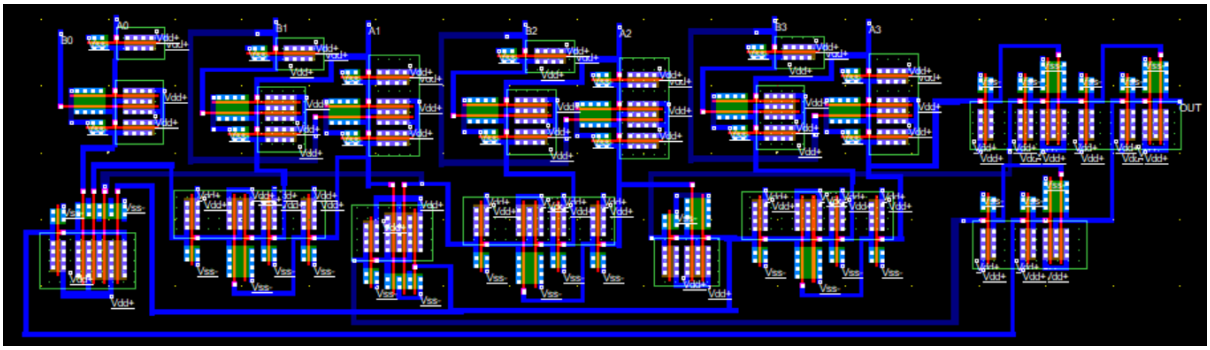


ii. Waveform:

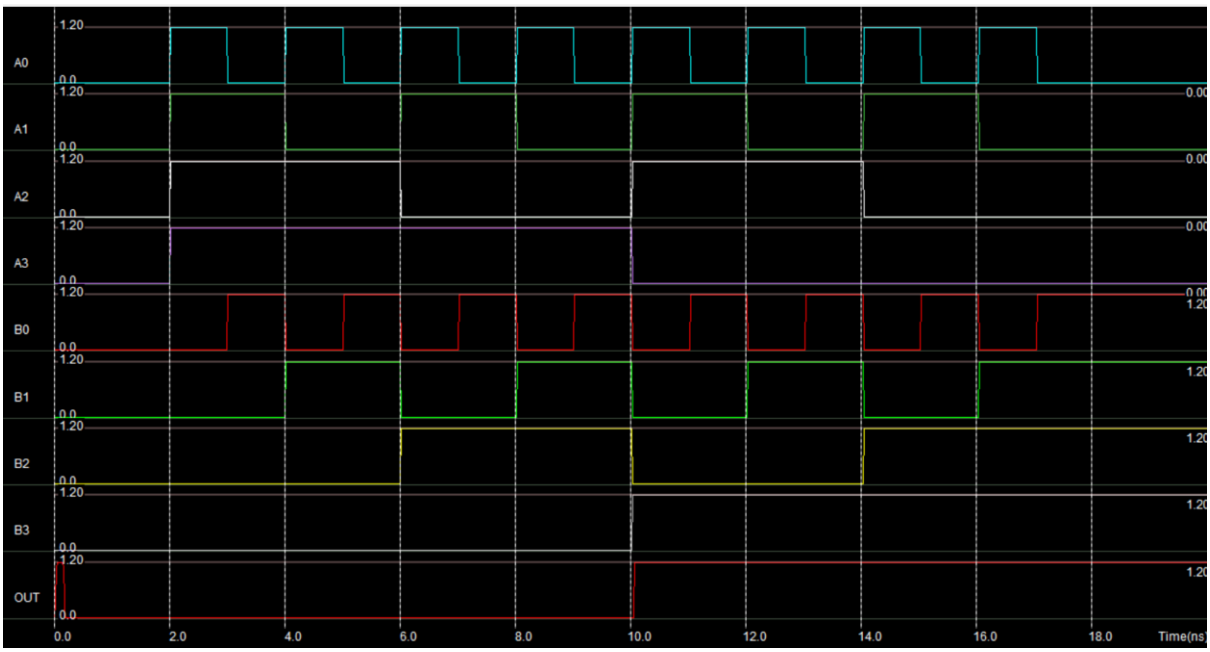


f. Comparator (Less than):

i. Layout:

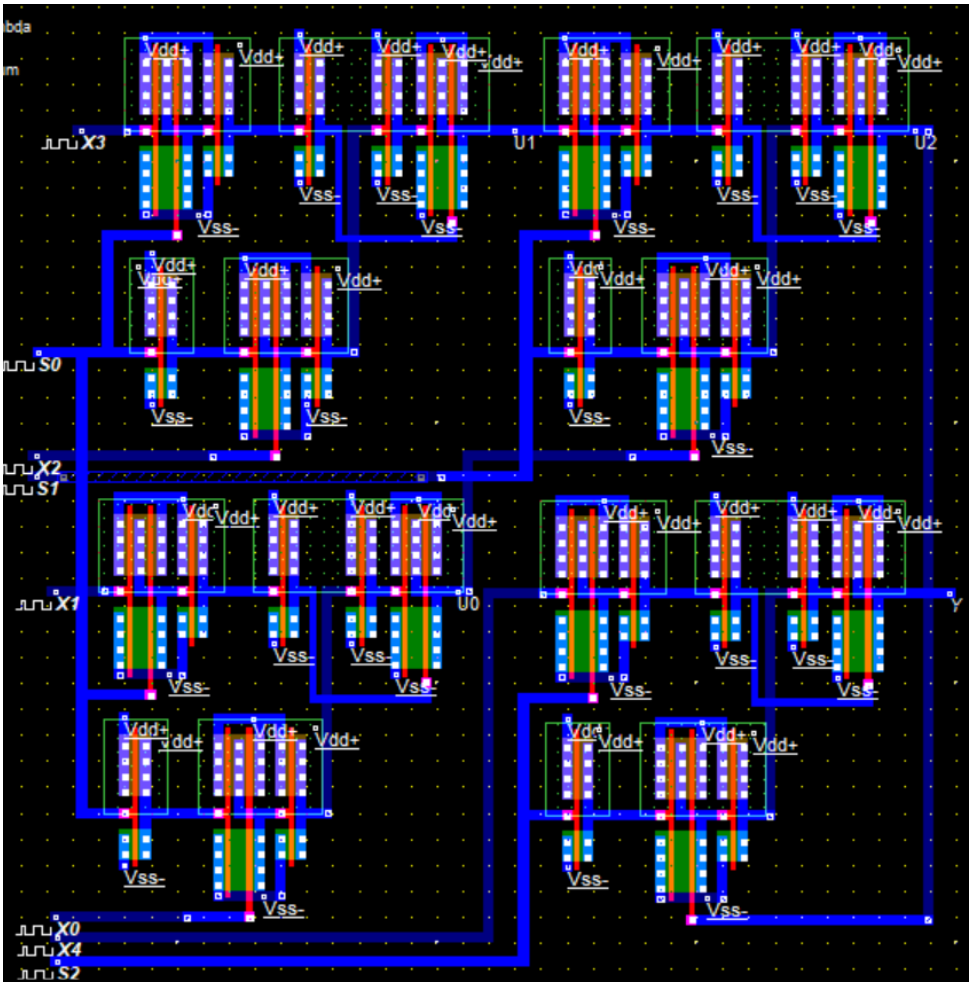


ii. Waveform:

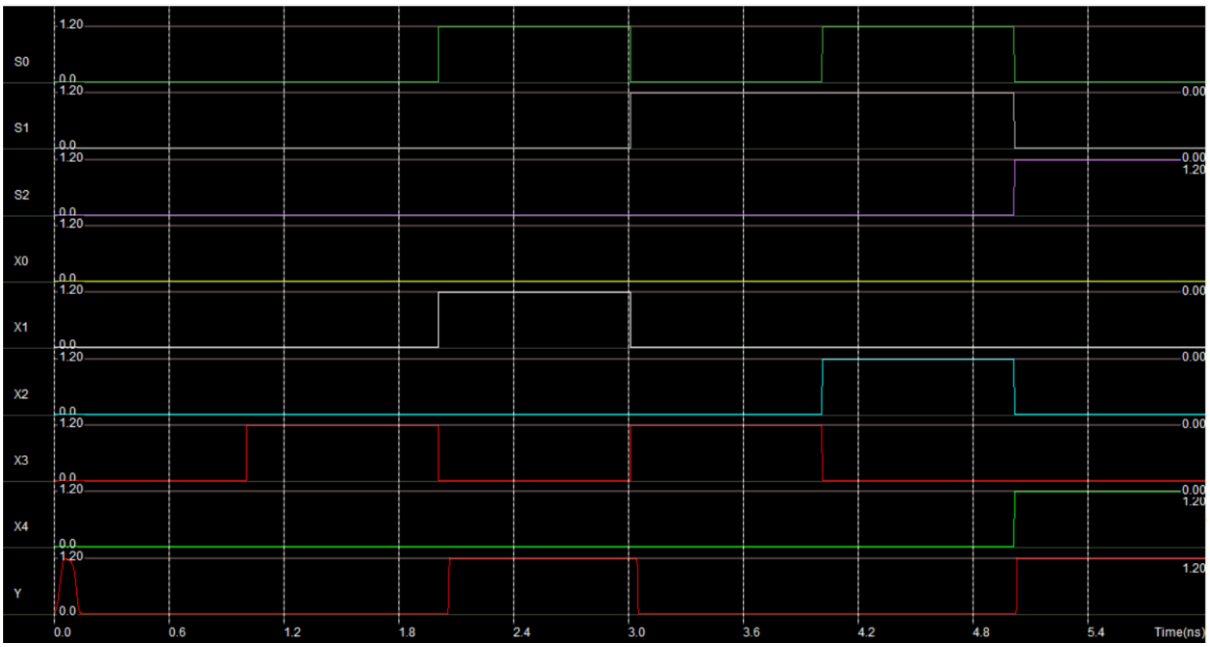


g. Mux 5-1:

i. Layout:

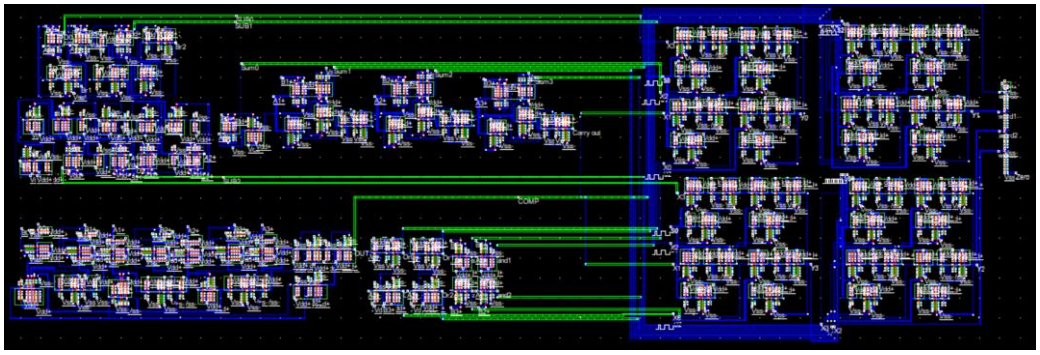


ii. Waveform:

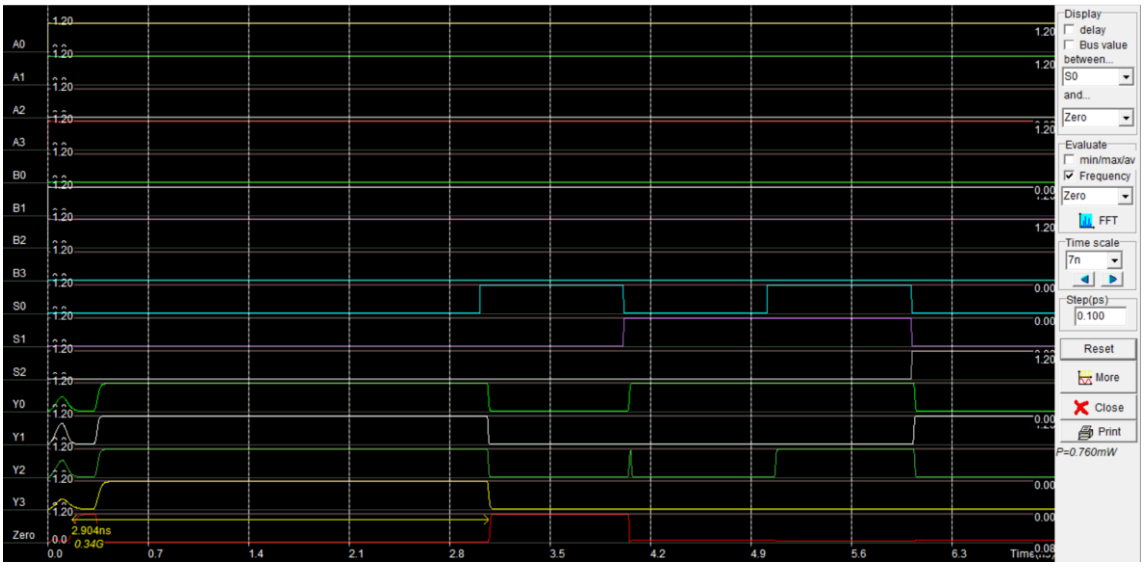


h. ALU:

i. Layout:



ii. Waveform:



V. Design Verification:

Verification is an important step in VLSI design flow. For this project, our verification process will include the following steps:

- Step 1: Logic design verification.
- Step 2: Circuit design verification.
- Step 3: Physical design verification.

1. Logic design verification.

There are several methods to verify the logic design of our project. One such method that we employ is to use testbench to simulate the logic of our circuit in Verilog. These can be found in section II of this report.

The waveform of our testbench showed that our logic design is correct. Timing and power analysis also showed that our design met the power and timing specification.

2. Circuit design verification.

After implementing and simulating our Logic design in Verilog, we then verify our circuit design. We use LTSpice to construct the circuit, and then simulate them to see if our circuit agrees with our logic design. The implementation and results can be found in section III.

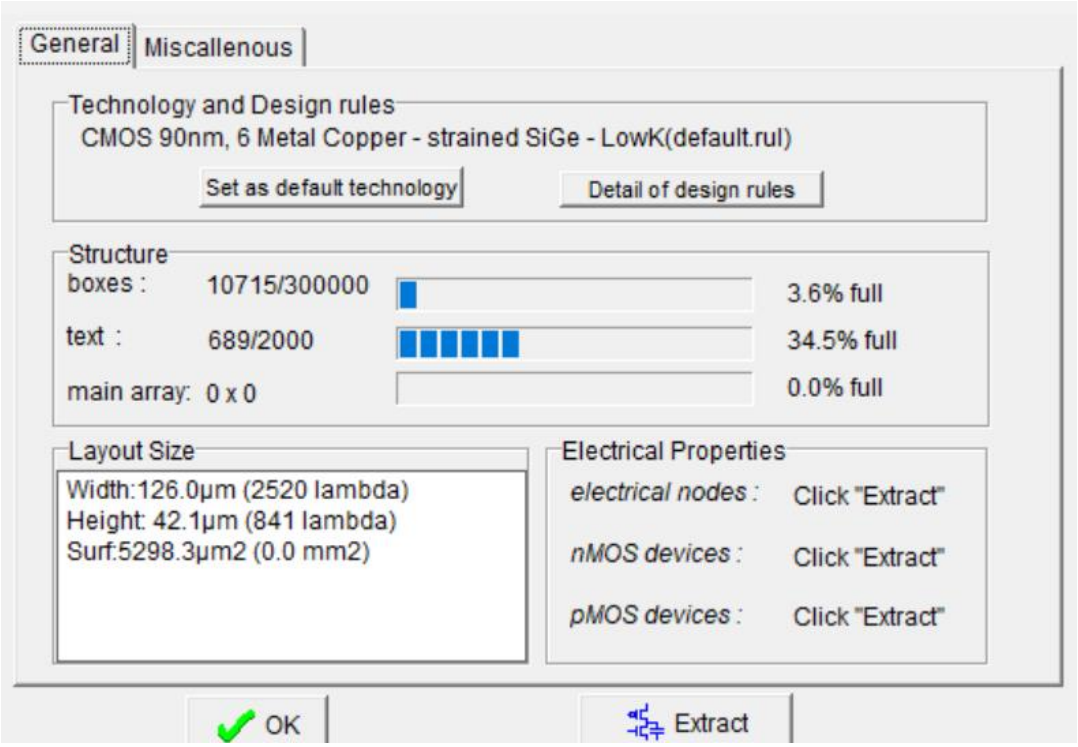
It is evident from our simulation that our circuit is correct and is functioning as expected. However, timing and power specification is not analyzed.

3. Physical design verification.

The final verification step involves verification of the physical design. Our physical design and simulation was constructed in Microwind and DSCH.

The following results were observed for our physical design.

- Functionality: the waveform observed in section IV demonstrates that our physical design is functioning correctly as we expected. All 5 operations of the ALU were correct, and the results including the result of the operation and the Zero signal agrees with our theoretical assumptions.
- Area: unfortunately, our physical design couldn't meet the area specification that we've set. Initially, our design had a surface area of about 6200 µm^2. After many optimization steps, we were able to reduce the area to about 5300 µm^2. However, this design still couldn't meet our initial specification.



- Speed: our design is capable of comfortably functioning at a period of less than 2 ns. This means that the operating frequency of our design is higher than 0.5 GHz. We found that even at 2 GHz, our physical design still performed well.