



King Saud University
Computer Engineering Department



King Saud University

College of Computer and Information Sciences
Computer Engineering Department
CEN415: Introduction to VLSI Design

6-bit input ALU

by

Abdelrahman Amr (441105960)

Yamen Majed Idriss (441105995)

Fahad Khalid Al-Ali (439103021)

Mohammad Salem Al-Hebshy (439105346)

A CEN415 Project Sent to the Department of
Computer Engineering, CCIS College
King Saud University

Supervised by:

Dr. Ayed Al-Qahtani



1 Introduction

Arithmetic & Logic Unit (ALU for short) is a combinational circuit which carries out both arithmetic and logic operations on operand inputs.

2 Problem Statement and Specifications

It's required to design an ALU which is capable of doing the operations shown in table below:

Operation code		Operation
s1	s0	
0	0	A XOR B
0	1	Add: A+B
1	0	Subtract: A-B
1	1	Y Zero Detection

ALU must have (at least): two different 6-bit inputs, Carry in, 2-bit selectors, Common V_{DD} & GND, 6-bit output, Carry out signal.

Table 1. Operations of ALU

3 Motivation

The motivation is to apply most of the CMOS VLSI design fundamental concepts, such as using a layout tool to make layout of real circuit and gates.

Second motivation is to get along with the Linux Operating System.

4 Digital Logic Design

This section shows the digital logic design of the 6-bit ALU using Logisim, which is a tool for designing and simulating logic circuits.

4.1 Complete Design

Figure below shows the complete design of the circuit.

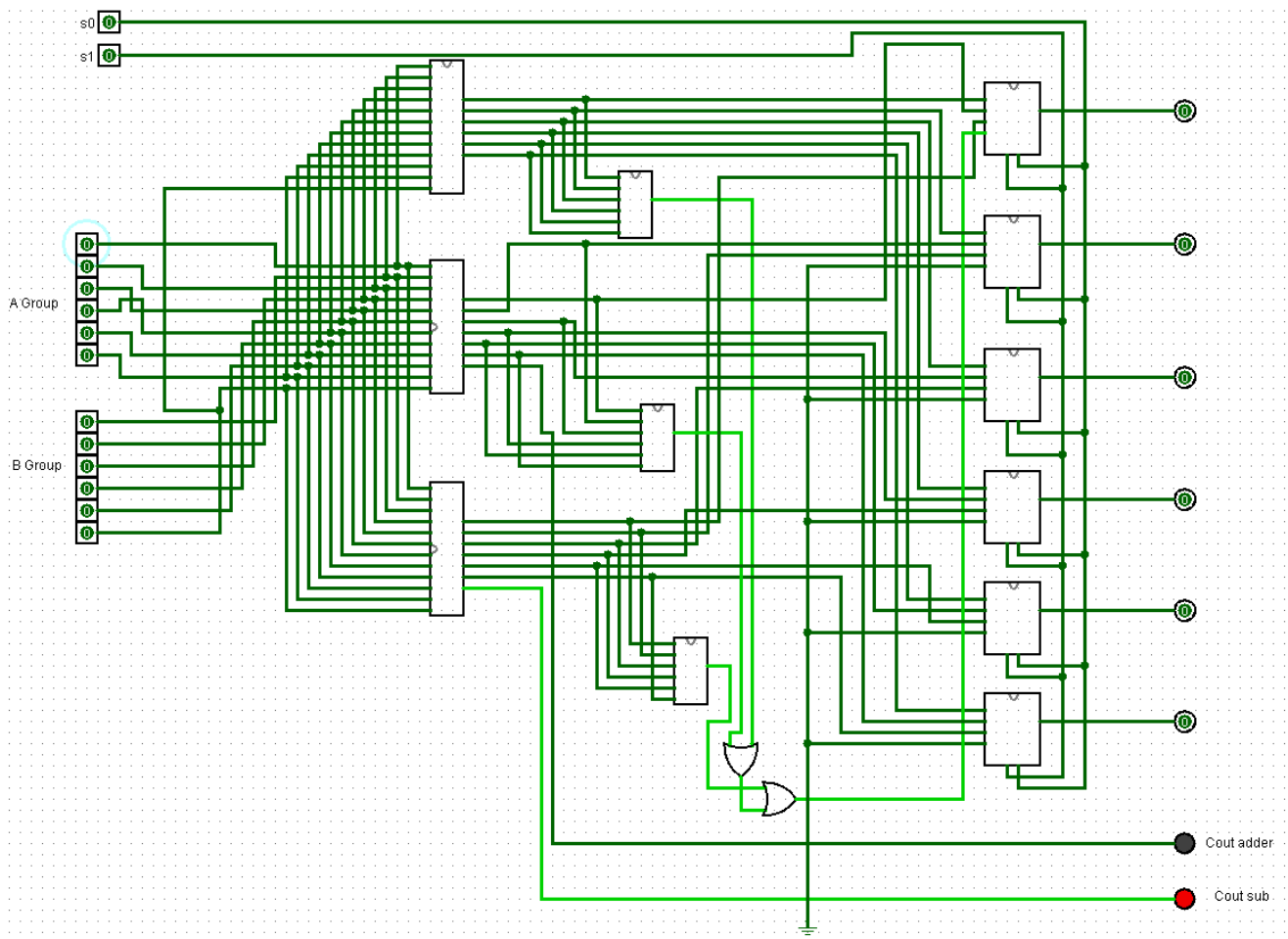


Figure 1. Complete Design

4.2 XOR Unit

Figures below show the design of the XOR unit, which is basically 6 2-input XORs connected together to form this unit.

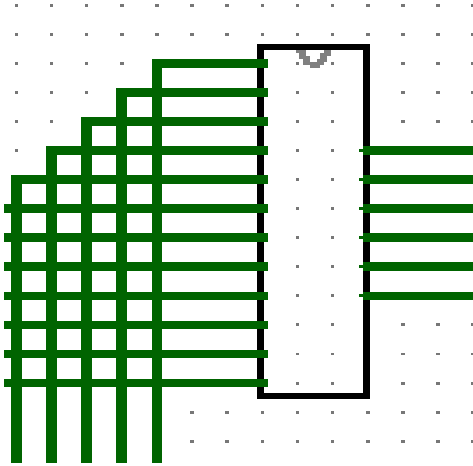


Figure 2. XOR Unit (Lvl 0)

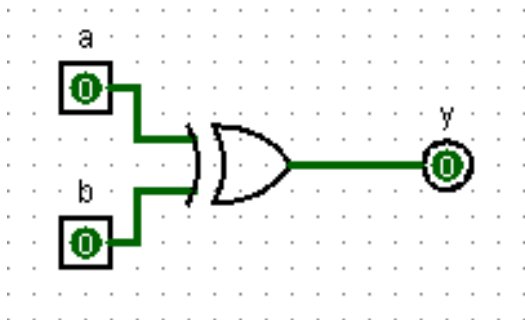


Figure 4. XOR Unit (Lvl 2)

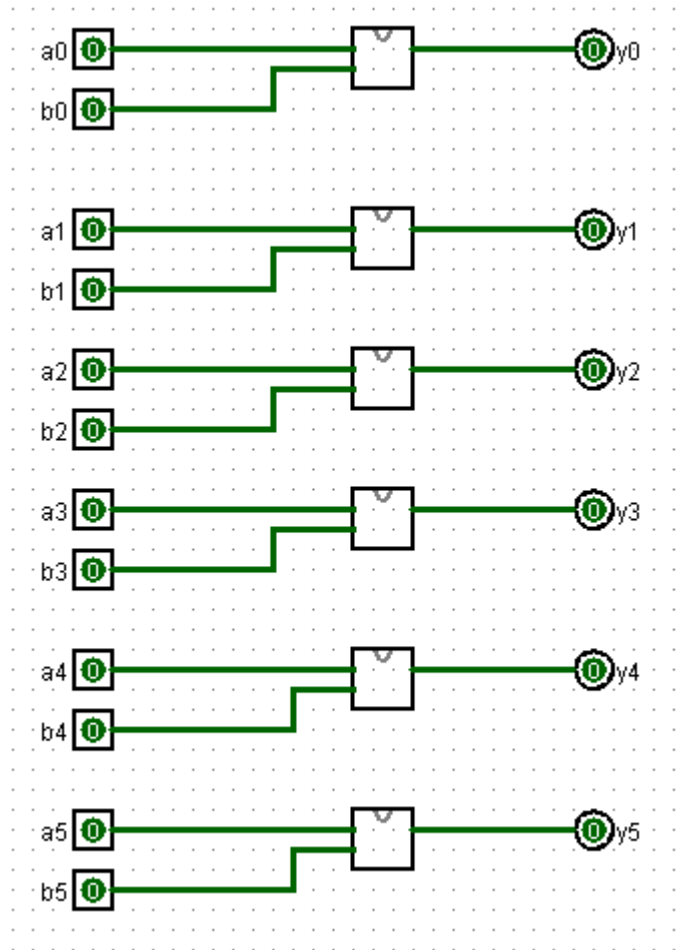


Figure 3. XOR Unit (Lvl 1)

4.3 Adder

The adder used is of type (Carry Ripple Adder), which is a two 6-bit inputs adder composed of 6 full adders having each carry out signal connected to the next adders as a carry in, figures below present the design of the adder used.

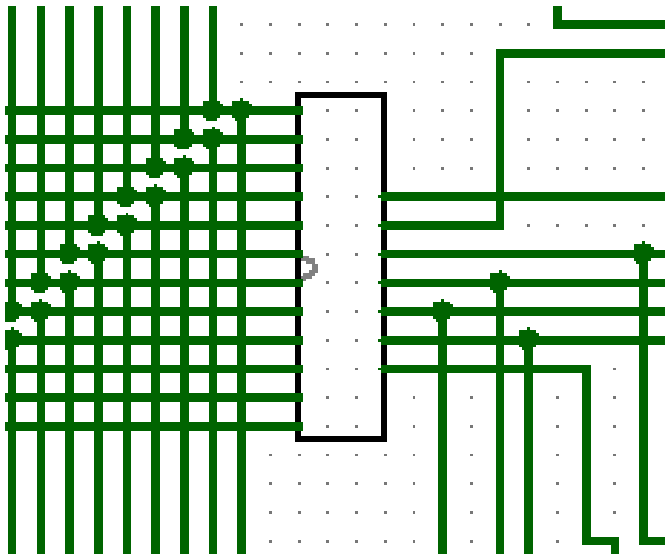


Figure 5. Carry Ripple Adder (Lvl 0)

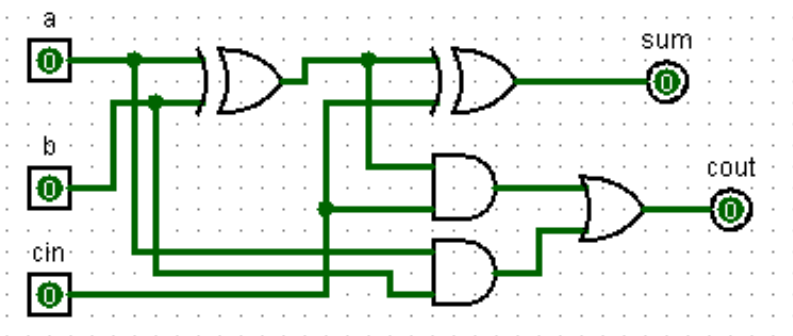


Figure 7. Carry Ripple Adder (Lvl 2)

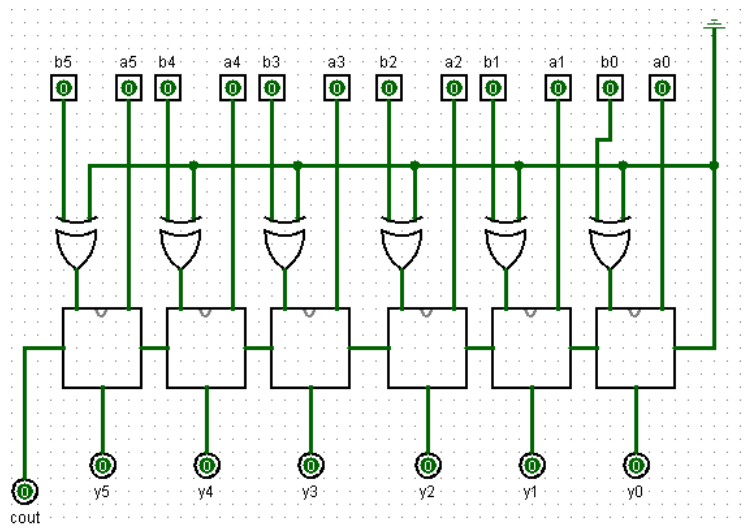


Figure 6. Carry Ripple Adder (Lvl1)

Figure 7. represents the Full-Adder, Figure 6. Shows 6 Full-Adders combined together to formulate the big unit shown in Figure 5.

4.4 Subtractor

In logic design we know that $A - B = A + (\sim B)$, this technique is used in this subtractor by inverting the inputs of B and then adding them to A, so same addition unit is used but with a little bit of modification, figures below show the subtraction unit.

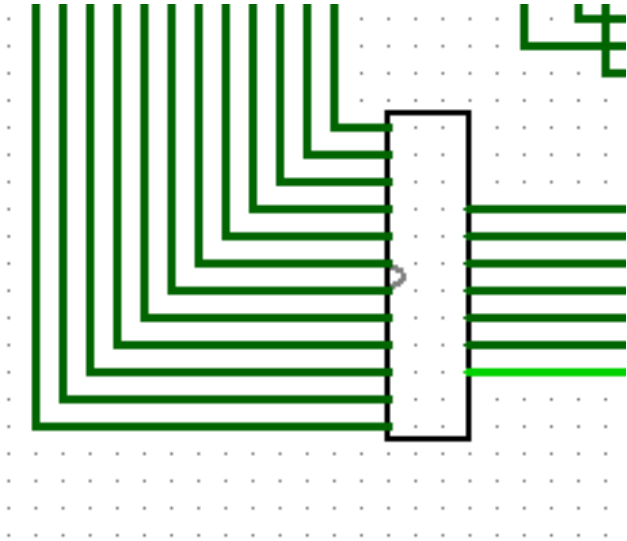


Figure 8. Subtraction Unit (Lvl 0)

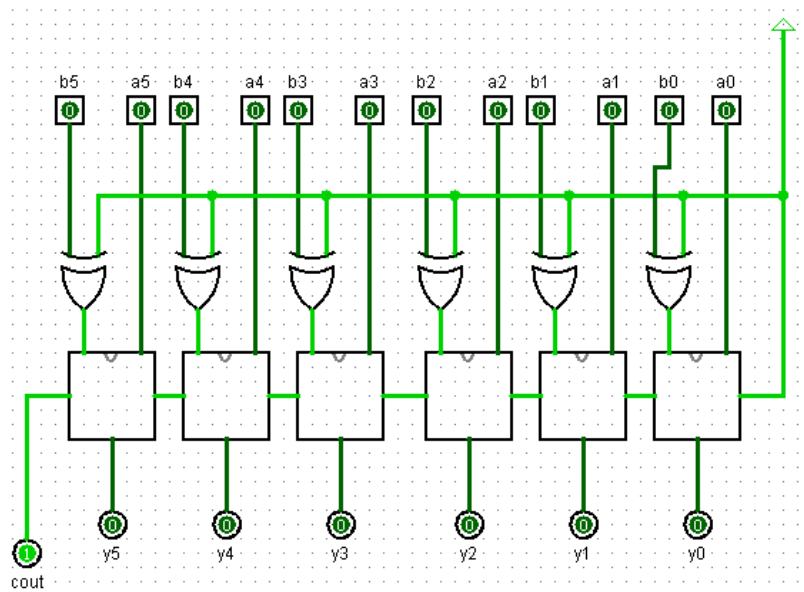


Figure 9. Subtraction Unit (Lvl 1)

4.5 Zero Detection

This unit's job is to raise a flag in 1 bit whenever the output is equal to zero, it's composed from AND, NOR and NOT gates, figures below show this unit and its inner design.

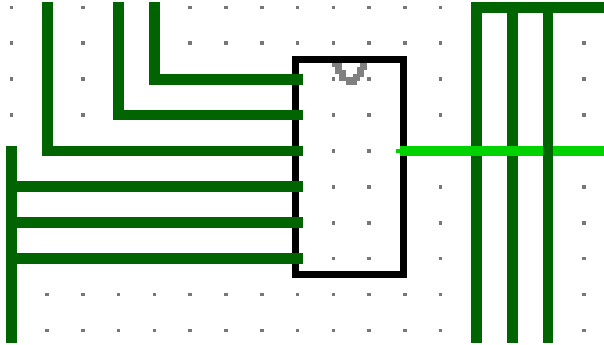


Figure 10. Zero Detection Unit (Lvl 0)

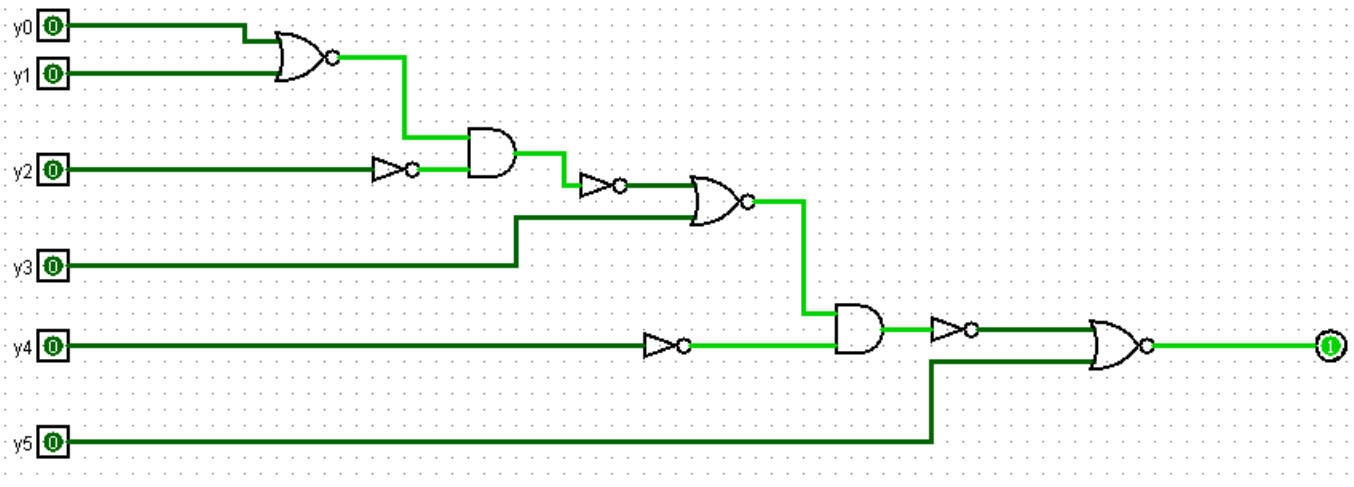


Figure 11. Zero Detection Unit (Lvl 1)

4.6 Multiplexer

The final design component of this section is the multiplexer, the multiplexer used is a 4:1 multiplexer made out of 3 smaller 2:1 multiplexers, figures below present both levels of design for this unit.

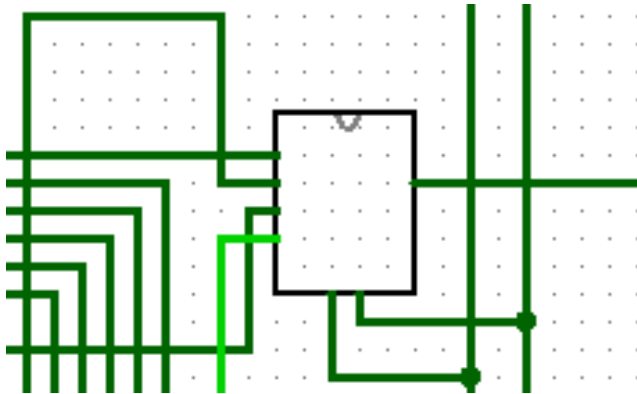


Figure 12. Multiplexer (Lvl 0)

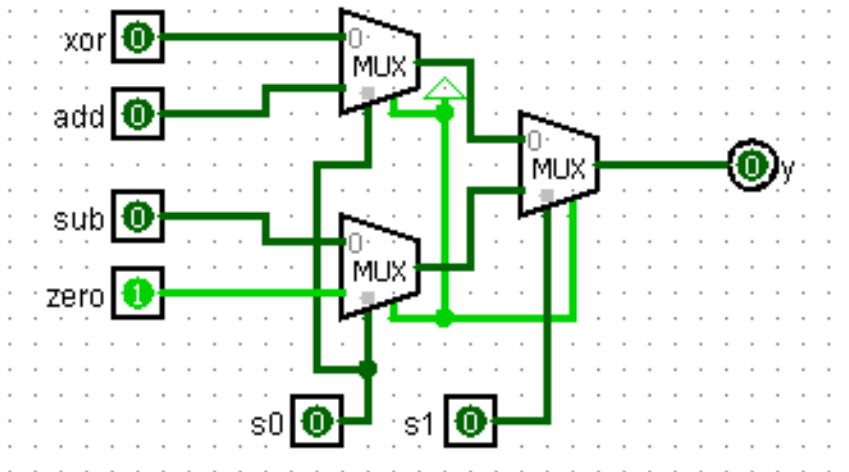
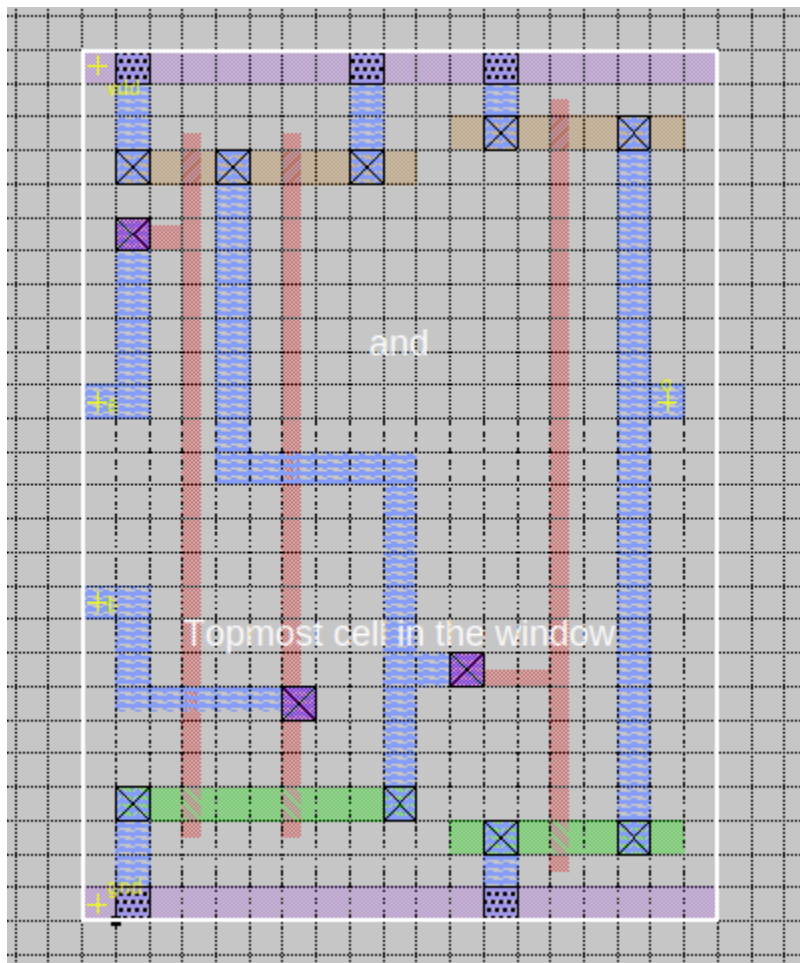


Figure 13. Multiplexer (Lvl 1)

5 Stick Diagrams

This section of the report dedicated to Magic, stick diagrams of each component including XOR, NOR, NOT, OR, AND, NAND, Full-Adder, Subtractor, Multiplexer, Zero Detection and finally the complete design will be shown in this section.

5.1 AND



2-inputs AND having
A and B as inputs
and O as an output

Figure 14. AND Stick Diagram

5.2 NOT

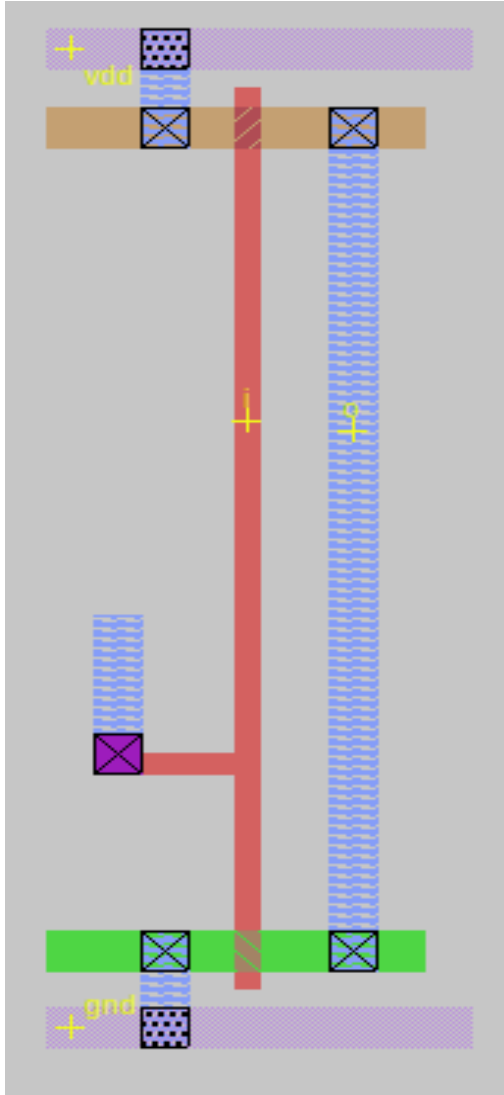


Figure 15. NOT Stick Diagram

5.3 NAND

NAND gate is basically formed from AND + NOT as shown in the figure below.

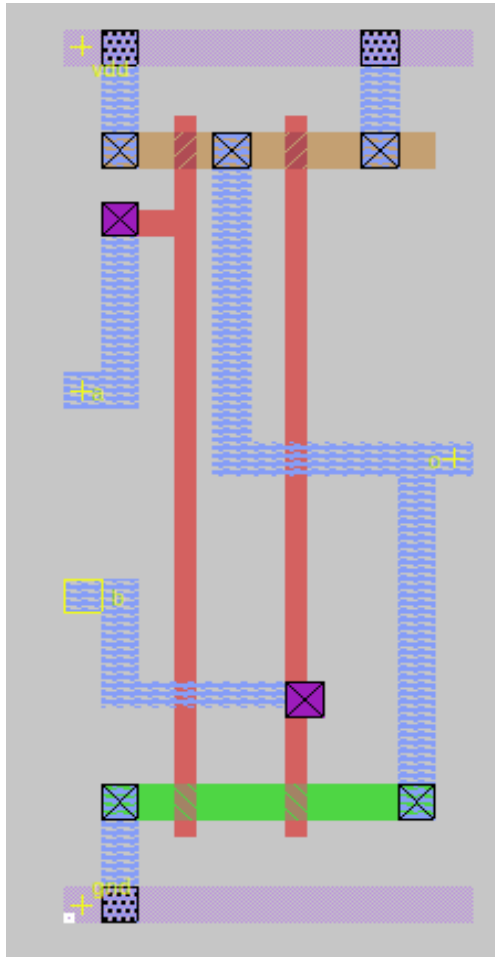


Figure 16. NAND Stick Diagram

2-inputs AND + NOT combined together to make a 2-inputs NAND gate



5.4 OR

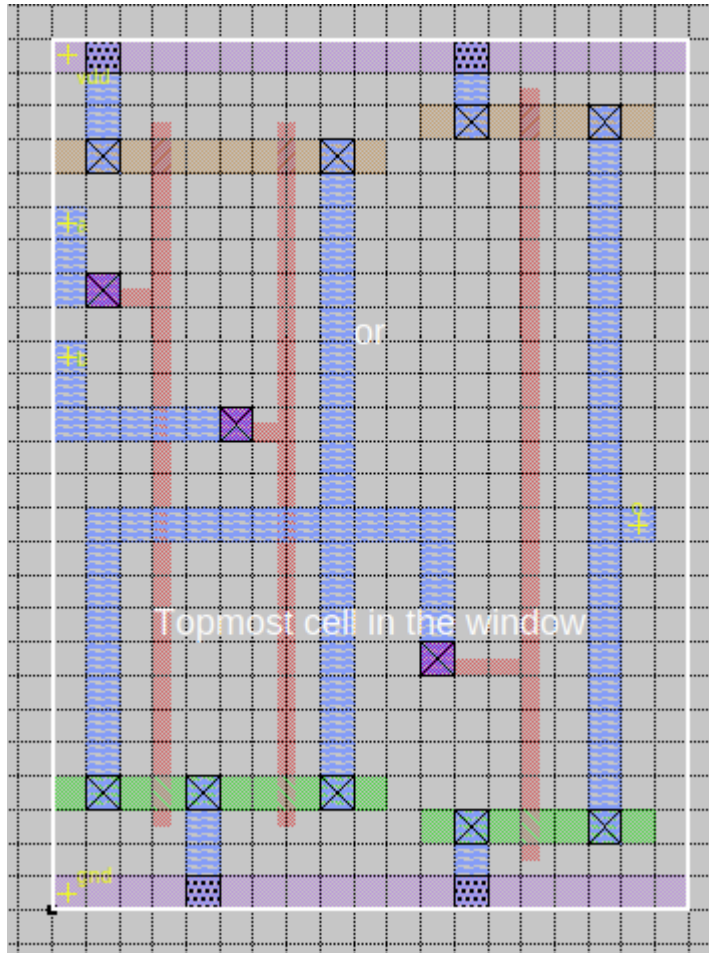


Figure 17. OR Stick Diagram



5.5 NOR

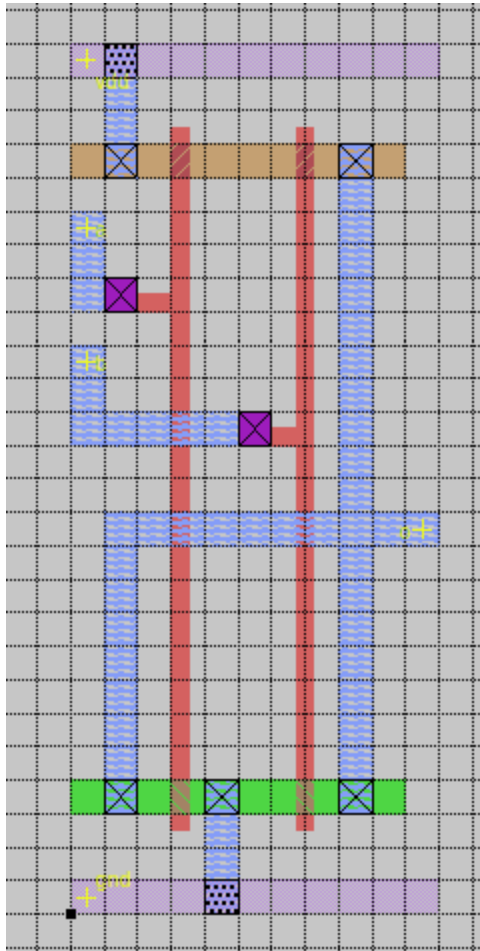


Figure 18. NOR Stick Diagram



5.6 XOR

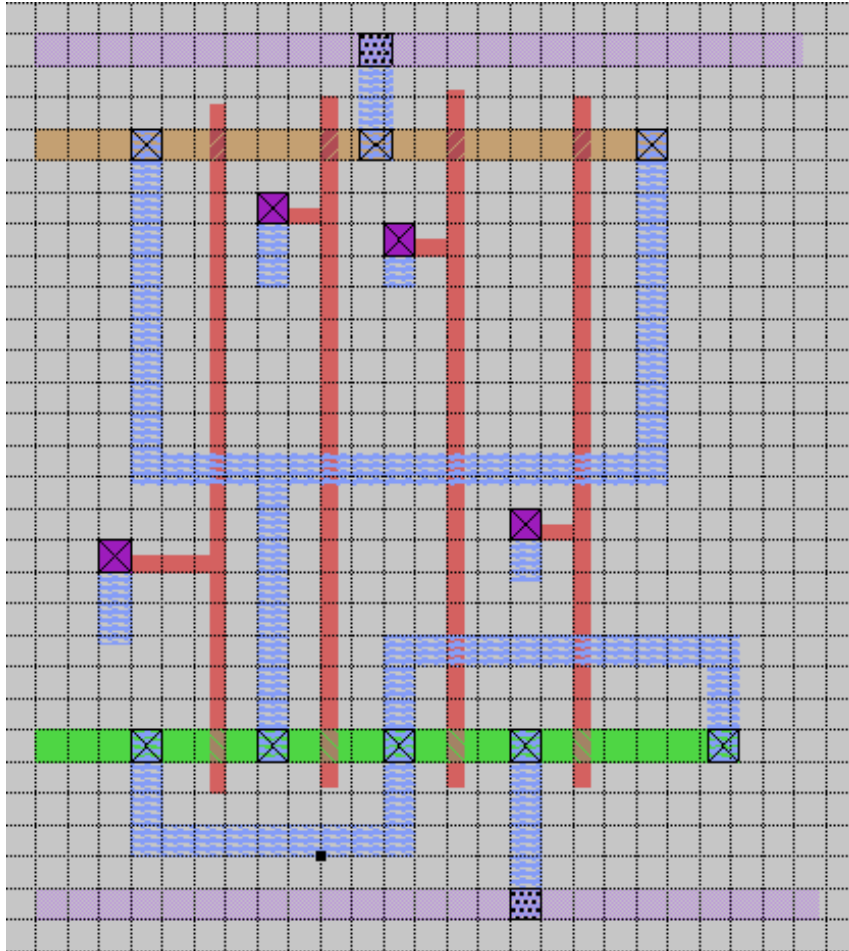


Figure 19. XOR Stick Diagram

5.7 Full Adder

Two 1-bit addition unit ($A+B$)

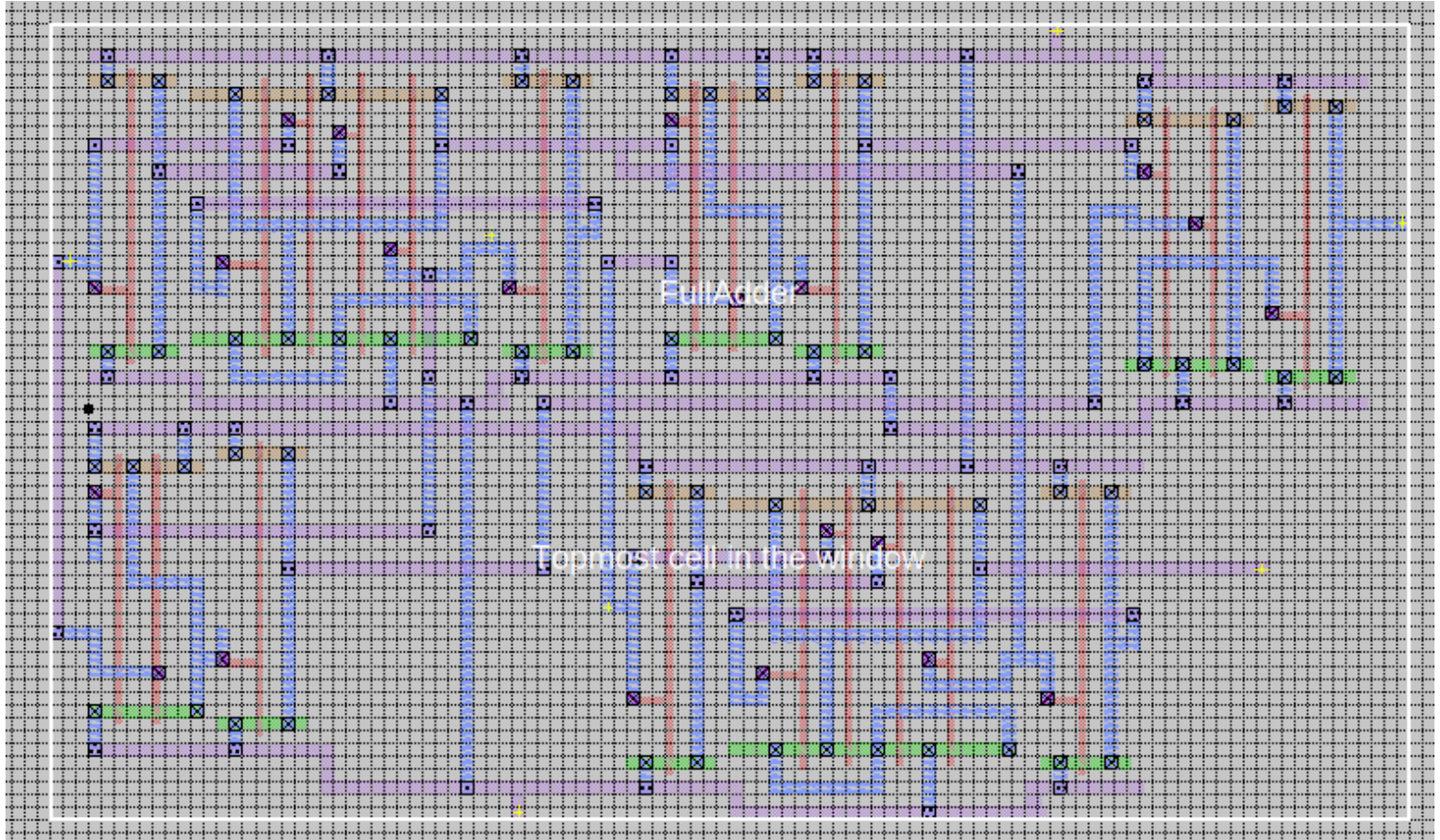


Figure 20. Full Adder Stick Diagram

5.8 Carry Ripple Adder

As mentioned before, the (Carry Ripple Adder) is an adder composed of 6 Full-Adders combined together in a certain way to model this big addition unit.

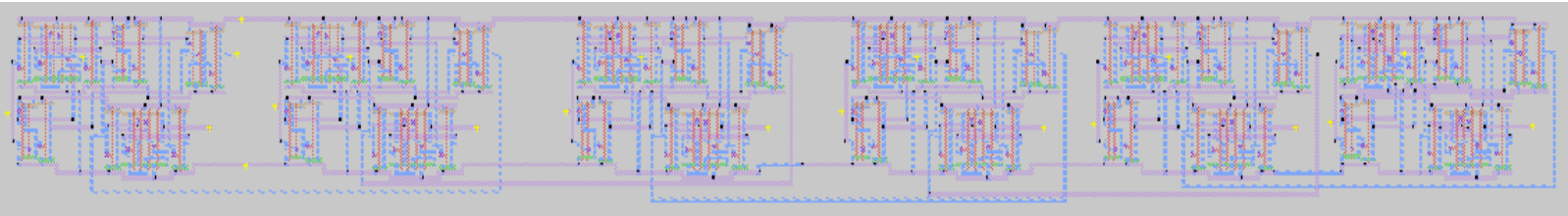


Figure 21. Carry Ripple Adder Stick Diagram

5.9 Subtractor

Subtraction unit is very similar to the previous unit since the subtraction technique uses addition.

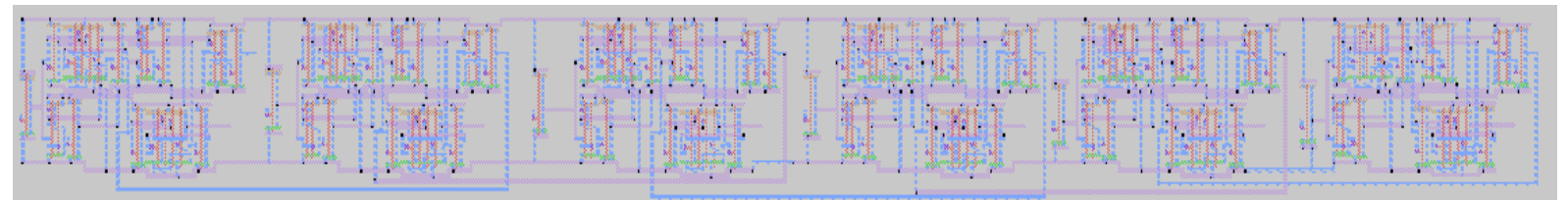


Figure 22. Subtractor Stick Diagram

5.10 Zero Detection

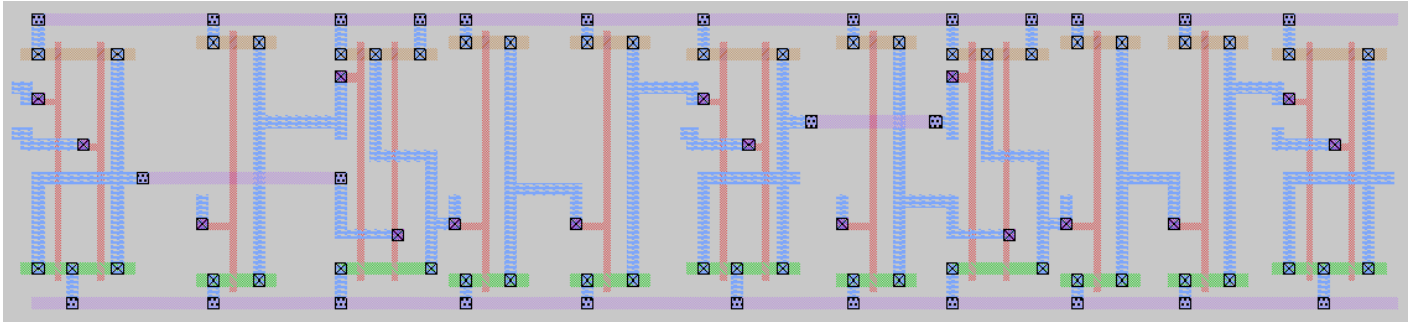


Figure 23. Zero Detection Stick Diagram

5.11 Transmission Gate (Multiplexer)

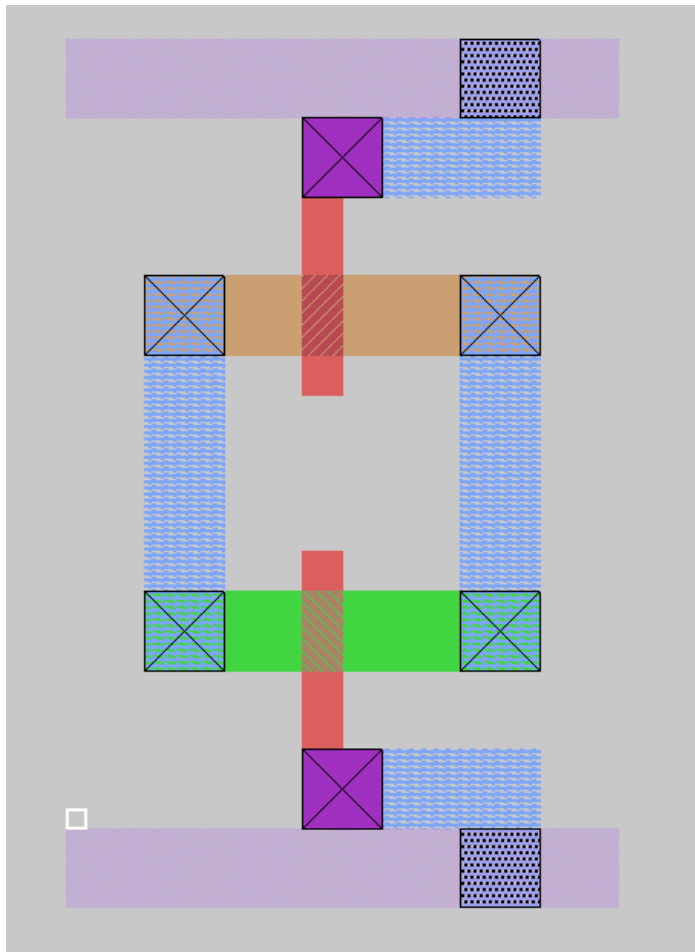


Figure 24. Transmission Gates Stick Diagram

5.12 Complete Design

After integrating all components together, finally the ALU is completed, and its stick diagram is shown below.



Figure 25. Complete Design (ALU) Stick Diagram

6 Testing

Verifying correctness of functionality is a crucial part of any circuit design, this section will be particularly dedicated to testing and results.

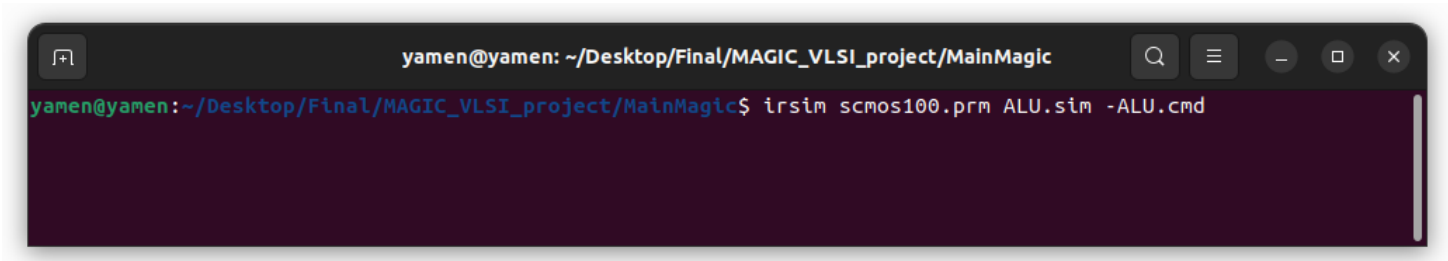
Notation used:

$A = \{A_5A_4A_3A_2A_1A_0\}$, $B = \{B_5B_4B_3B_2B_1B_0\}$, $O = \{O_5O_4O_3O_2O_1O_0\}$

Where both A and B are inputs and O is the output.

6.1 Test 1

Using IRSIM to test the circuit.

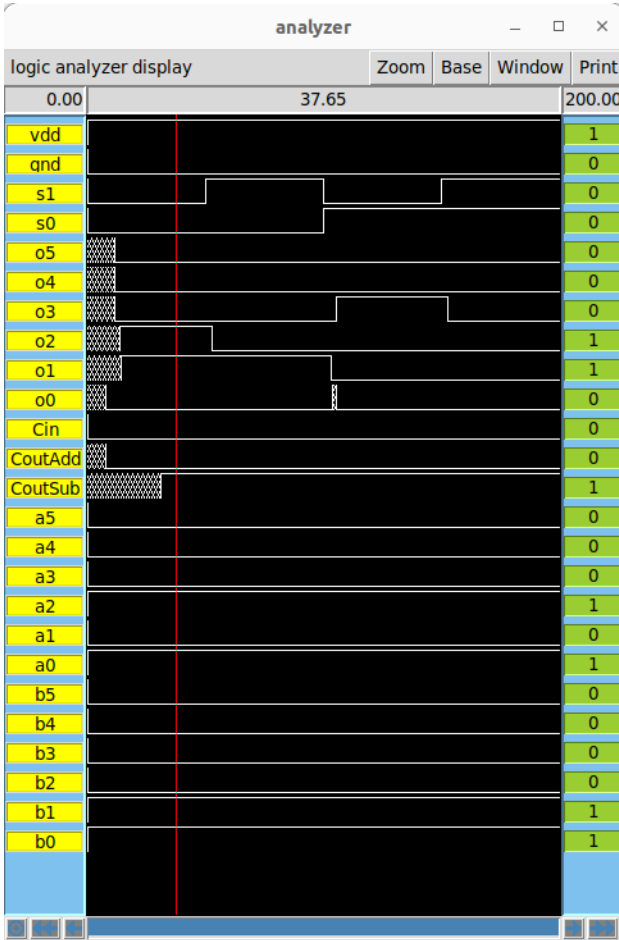


```
yamen@yamen: ~/Desktop/Final/MAGIC_VLSI_project/MainMagic
yamen@yamen:~/Desktop/Final/MAGIC_VLSI_project/MainMagic$ irsim scmos100.prm ALU.sim -ALU.cmd
```

Figure 26. Using IRSIM

$A = \{000101\} = (5)_{10}$, $B = \{000011\} = (3)_{10}$

$A \oplus B = \{000010\}$, $A + B = \{001000\}$, $A - B = \{000010\}$



When $S_1S_0 = \{00\}$

$O = \{000110\}$

When $S_1S_0 = \{01\}$

$O = \{001000\}$

When $S_1S_0 = \{10\}$

$O = \{000010\}$

When $S_1S_0 = \{11\}$

$O = \{000000\}$

```

tkcon 2.3 Main
File Console Edit Interp Prefs History Help
Using default name "Gnd" for ground net.
ALU.sim: Ignoring lumped-resistance ('R' construct)

Read ALU.sim lambda:1.00u format:MIT
382 nodes, 1514 aliases; transistors: n-channel=380 p-channel=380
parallel txtors:none
b0=1 b1=1 b2=0 b3=0 b4=0 b5=0 a0=1 a1=0 a2=1 a3=0 a4=0 a5=0 CoutSub=1
CoutAdd=0 Cin=0 o0=0 o1=1 o2=1 o3=0 o4=0 o5=0 s0=0 s1=0 gnd=0 vdd=1
time = 50.000ns
b0=1 b1=1 b2=0 b3=0 b4=0 b5=0 a0=1 a1=0 a2=1 a3=0 a4=0 a5=0 CoutSub=1
CoutAdd=0 Cin=0 o0=0 o1=1 o2=0 o3=0 o4=0 o5=0 s0=0 s1=1 gnd=0 vdd=1
time = 100.000ns
b0=1 b1=1 b2=0 b3=0 b4=0 b5=0 a0=1 a1=0 a2=1 a3=0 a4=0 a5=0 CoutSub=1
CoutAdd=0 Cin=0 o0=0 o1=0 o2=0 o3=1 o4=0 o5=0 s0=1 s1=0 gnd=0 vdd=1
time = 150.000ns
b0=1 b1=1 b2=0 b3=0 b4=0 b5=0 a0=1 a1=0 a2=1 a3=0 a4=0 a5=0 CoutSub=1
CoutAdd=0 Cin=0 o0=0 o1=0 o2=0 o3=0 o4=0 o5=0 s0=1 s1=1 gnd=0 vdd=1
time = 200.000ns
Main console display active (Tcl8.6.12 / Tk8.6.12)
(MainMagic) 3 %
  
```

Figure 28. Main Console Results

6.2 Test 2

$A = \{010000\} = (16)_{10}$, $B = \{010000\} = (16)_{10}$.

$A \oplus B = \{000000\}$, $A + B = \{100000\}$, $A - B = \{000000\}$

When $S_1S_0 = \{00\}$, $O = \{000000\}$

When $S_1S_0 = \{01\}$, $O = \{100000\}$

When $S_1S_0 = \{10\}$, $O = \{000000\}$

When $S_1S_0 = \{11\}$, $O = \{000001\}$

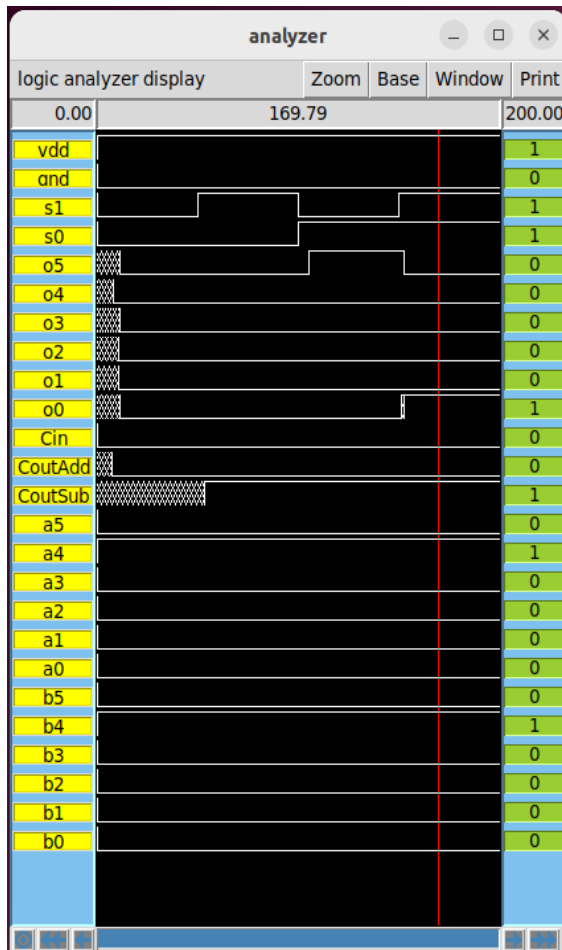


Figure 30. Analyzer Results

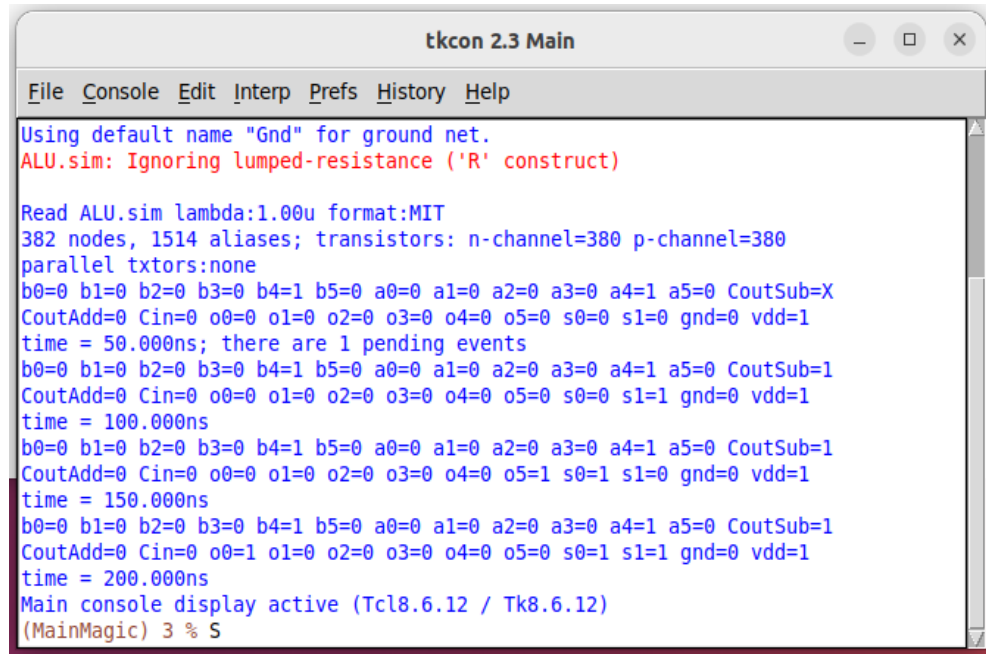


Figure 29. Main Console Results



7 CMD Generator Tool

Nearly all (.CMD) files were generated automatically by using a Python script which can find all logical combinations of outputs depending on inserted input, more details can be found by hovering over and visiting the URL below:

https://github.com/abdo20050/cmd_generator.