4-Bit ALU with Shifter Unit
VLSI Design Project

Khaled Arheby
Electrical and Electronics Engineer

April 05, 2025

Abstract

This project involves the design and implementation of a 4-bit Arithmetic Logic Unit (ALU) with an integrated Shifter Unit using the Electric VLSI design system. The ALU performs fundamental arithmetic and logic operations, while the Shifter Unit enables both left and right shift arithmetic and logic operations to support efficient data manipulation. The design employs basic CMOS logic gates, multiplexers, and full adders. Both schematic and layout views are developed and verified using SPICE simulation. This project demonstrates a practical understanding of digital logic, circuit design, and VLSI layout techniques.

Table of Contents

1. Introduction

The Arithmetic Logic Unit (ALU) is a critical component of a processor, responsible for performing arithmetic and logical operations. In this project, we design a 4-bit ALU integrated with a Shifter Unit using Electric VLSI. The goal is to implement and simulate a working ALU system at the transistor level to demonstrate key VLSI concepts, including modular circuit design, CMOS logic, and SPICE simulation.

2. Tools Used

The primary tool used in this project is Electric VLSI, an open-source software for designing and simulating VLSI circuits. Additional tools used include:
- SPICE (Simulation Program with Integrated Circuit Emphasis) for circuit simulation
- Electric's built-in schematic editor and layout editor
- Screenshot and labeling tools for documentation

3. Project Components

3.1 Basic Logic Gates

The design begins with the implementation of fundamental logic gates including NOT, NAND, and XOR. These gates are constructed using CMOS technology to serve as the building blocks for more complex components such as multiplexers, adders, and ALU units.

3.2 Multiplexers

Multiplexers are crucial in selecting between various logic and arithmetic operations. Both 2-bit and 4-bit multiplexers were designed and used for routing data paths and selecting specific functions in the ALU based on control inputs.

3.3 Arithmetic Circuits

A 1-bit full adder was implemented as the core of the arithmetic logic. By cascading four 1-bit full adders, a 4-bit ripple-carry adder was created, enabling addition of 4-bit binary numbers with carry input and output.

3.4 ALU Unit

The ALU unit combines logic and arithmetic circuits to perform multiple operations, including AND, OR, XOR, and ADD. A control signal determines the operation mode. The design includes both schematic and layout views created in Electric VLSI.

3.5 Shifter Unit

The Shifter Unit supports both logical left and right shift operations. It was implemented using a combination of wires, multiplexers, and control signals. The shift operations help manipulate data for logical and arithmetic purposes efficiently.

3.6 Control Logic

Control logic determines the operation performed by the ALU. A set of select lines (S3-S0) is used to choose between different ALU operations such as AND, OR, XOR, ADD, arithmetic and shifting operations. The logic is implemented using 3 layers of multiplexer and decoder circuits.

4. ALU circuits and components

4.1 Basic Logic Gates

4.1.1 AND Gate

- **Description: The AND gate outputs 1 only if both inputs are 1.**

Truth Table:

A	B	F = A AND B
0	0	0
0	1	0
1	0	0
1	1	1

4.1.2 OR Gate

- **Description: The OR gate outputs 1 if at least one input is 1.**
- **Truth Table:**

A	B	F = A OR B
0	0	0
0	1	1
1	0	1
1	1	1

4.1.3 XOR Gate

- **Description: The XOR gate outputs 1 if the inputs are different.**
- **Truth Table:**

A	B	F = A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

4.1.4 NOT Gate

- **Description: The NOT gate (Inverter) flips the input value.**
- **Truth Table:**

A	F = NOT A
0	1
1	0

4.2 Arithmetic Circuits

4.2.1 Full Adder

- **Description: Adds two 1-bit inputs (A and B) with a carry-in (Cin) and produces a Sum (S) and Carry-out (Cout).**
- **Truth Table:**

A	B	Cin	Sum (S)	Carry-out (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

4.2.2 Subtractor

- **Description: Computes the subtraction A - B using A + (~B) + 1 (Two's Complement).**
- **Truth Table:**

A	B	Cin (Borrow-in)	F = A - B - Cin
0	0	0	0
0	0	1	-1
0	1	0	-1
0	1	1	-2
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	-1

4.3 Multiplexers (MUX)

4.3.1 2-to-1 Multiplexer

- **Description: Selects one of two inputs based on the Select (S) signal.**
- **Truth Table:**

S	I0	I1	F = MUX Output
0	X	X	I0
1	X	X	I1

4.3.2 4-to-1 Multiplexer

- **Description: Selects one of four inputs based on S1 and S0.**
- **Truth Table:**

S1	S0	I0	I1	I2	I3	F
0	0	X	X	X	X	I0
0	1	X	X	X	X	I1
1	0	X	X	X	X	I2
1	1	X	X	X	X	I3

4.4 Shift Operations

4.4.1 Logical Shift Left (LSL)

- **Description: Moves bits left, inserting 0 at LSB.**
- **Truth Table:**

A3 A2 A1 A0	LSL Output
0001	0010
0010	0100

4.4.2 Logical Shift Right (LSR)

- **Description: Moves bits right, inserting 0 at MSB.**
- **Truth Table:**

A3 A2 A1 A0	LSR Output
1000	0100
0100	0010

4.4.3 Arithmetic Shift Right (ASR)

- **Description: Moves bits right, preserving MSB.**
- **Truth Table:**

A3 A2 A1 A0	ASR Output
1011	1101
1100	1110

4.4.4Key Differences Between Logical and Arithmetic Shifts:

Type	Fills Vacant Bit With	Preserves Sign?	Used For
Logical Left Shift (LSL)	0	No	Multiplication (Unsigned)
Logical Right Shift (LSR)	0	No	Division (Unsigned)
Arithmetic Left Shift (ASL)	0	No	Multiplication (Signed)
Arithmetic Right Shift (ASR)	Sign Bit (MSB)	Yes	Division (Signed)

5. ALU Control Table (With Carry-In as LSB):

S3	S2	S1	S0	Cin	Operation	Function Description
0	0	0	0	0	Transfer A	F = A
0	0	0	0	1	Increment A	F = A + 1
0	0	0	1	0	Addition	F = A + B
0	0	0	1	1	Add with Carry	F = A + B + Cin
0	0	1	0	0	Subtract with Borrow	F = A - B - Cin
0	0	1	0	1	Subtraction	F = A - B
0	0	1	1	0	Decrement A	F = A - 1
0	0	1	1	1	Transfer A	F = A
0	1	0	0	X	AND	F = A AND B
0	1	0	1	X	OR	F = A OR B
0	1	1	0	X	XOR	F = A XOR B
0	1	1	1	X	NOR	F = A NOR B
1	0	0	0	X	Logical Shift Left (LSL)	F = {A2, A1, A0, 0}
1	0	0	1	X	Logical Shift Right (LSR)	F = {0, A3, A2, A1}
1	0	1	0	X	Arithmetic Shift Left (ASL)	F = {A2, A1, A0, 0} (Same as LSL)
1	0	1	1	X	Arithmetic Shift Right (ASR)	F = {A3, A3, A2, A1} (MSB Preserved)

5.Design and simulation

5.1 NOT Gate Design

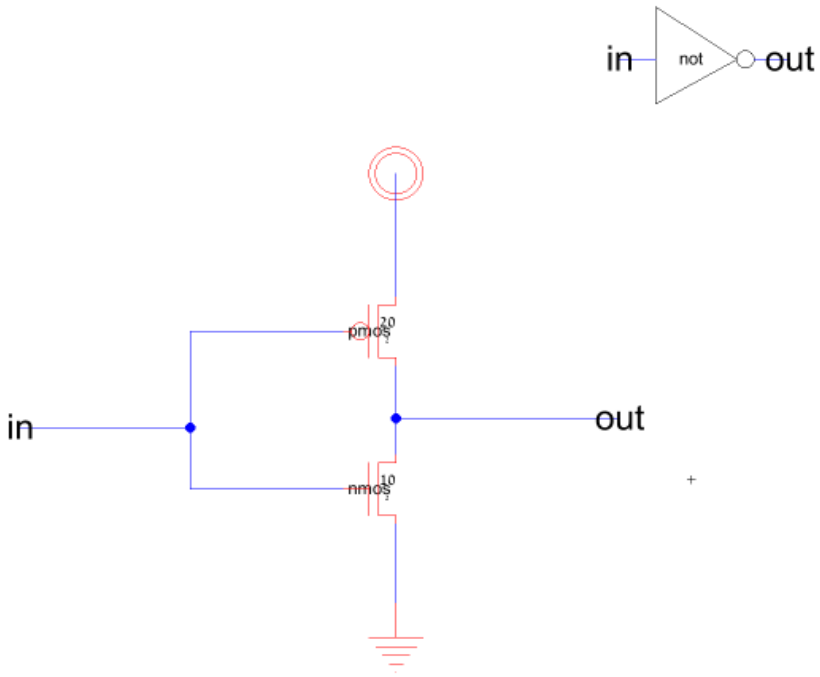

Figure 1: schematic design

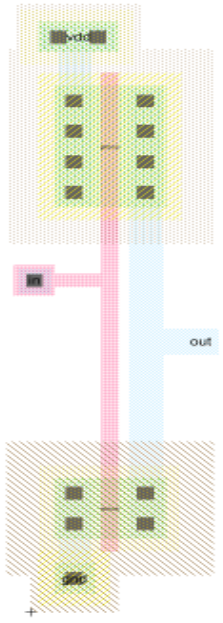

Figure 2:layout desgin

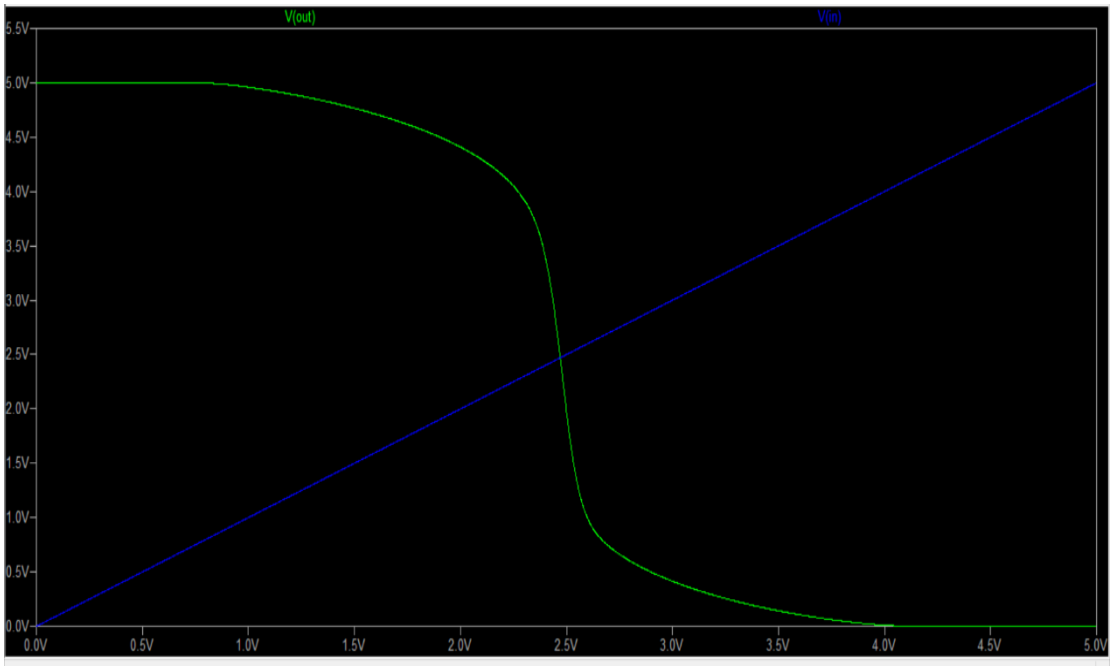

Figure 3:simulation of not gate

LVS: Hierarchical NCC every cell in the design: cell 'not{sch}' cell 'not{lay}'

Comparing: newproj:not{sch} with: newproj:not{lay}

exports match, topologies match, sizes match in 0.0 seconds.

Summary for all cells: exports match, topologies match, sizes match

NCC command completed in: 0.0 seconds.

DRC: Running DRC with area bit on, extension bit on, Mosis bit

Checking again hierarchy .... (0.0 secs)

Found 7 networks

Checking cell 'not{lay}'

No errors/warnings found

0 errors and 0 warnings found (took 0.0 secs)

5.2 NOR Gate Design:

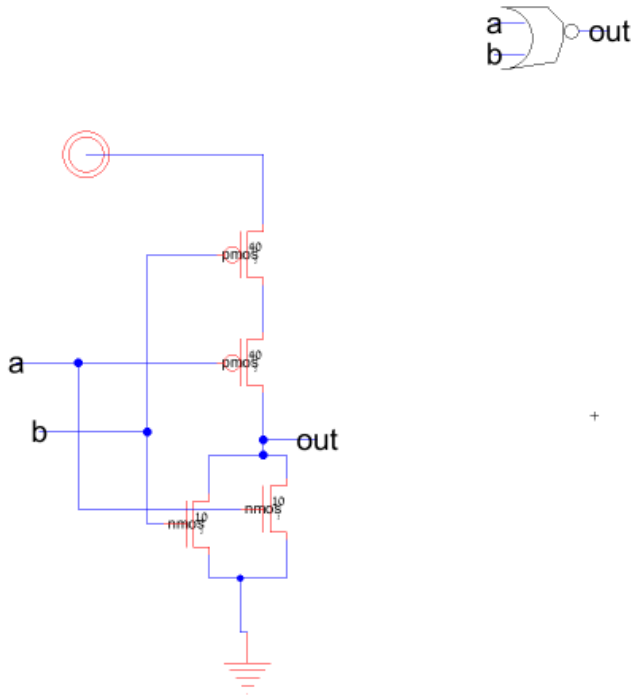

Figure 4:schematic design of nor gate

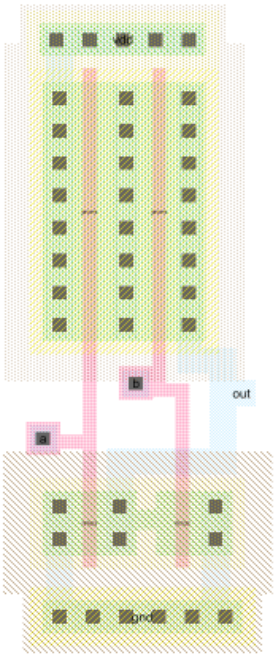

Figure 5:layout design of nor gate

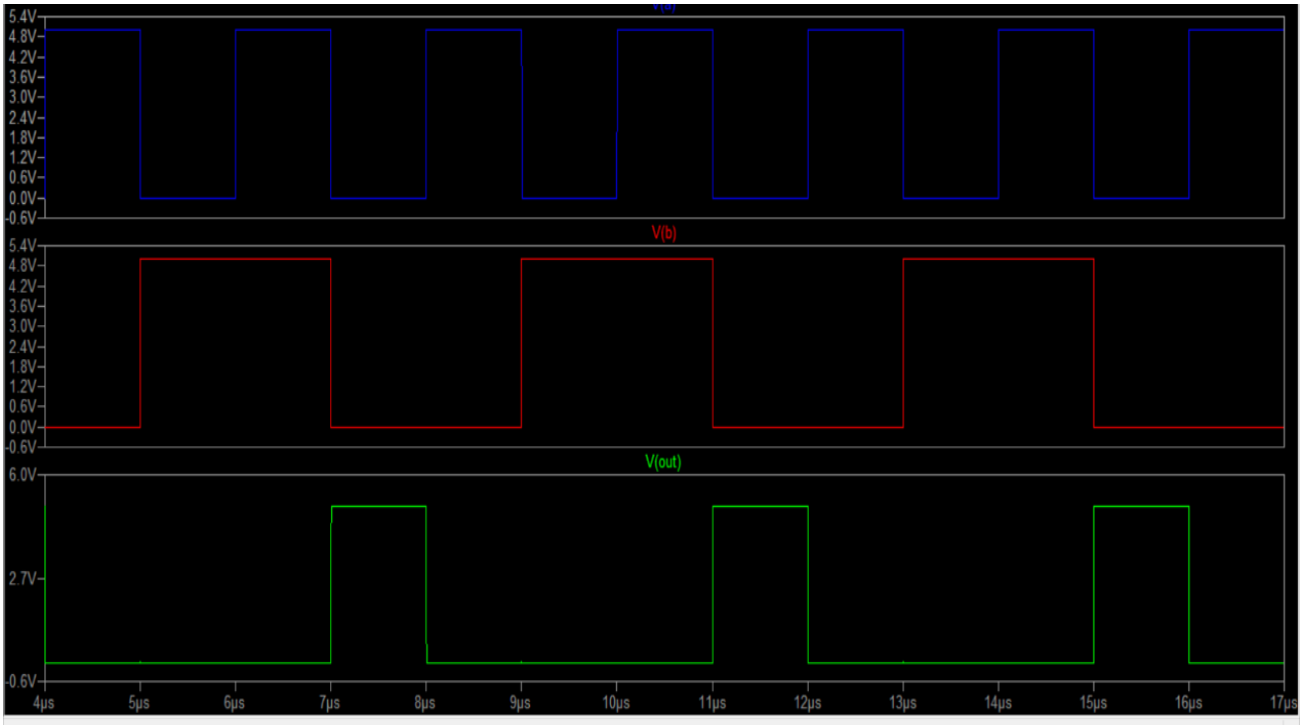

Figure 6:simulation of nor gate

LVS: Hierarchical NCC every cell in the design: cell 'nor{sch}' cell 'nor{lay}'

Comparing: newproj:nor{sch} with: newproj:nor{lay}

exports match, topologies match, sizes match in 0.002 seconds.

Summary for all cells: exports match, topologies match, sizes match

NCC command completed in: 0.002 seconds.

DRC: Running DRC with area bit on, extension bit on, Mosis bit

Checking again hierarchy .... (0.0 secs)

Found 11 networks

Checking cell 'nor{lay}'

No errors/warnings found

0 errors and 0 warnings found (took 0.011 secs)

5.3NAND Gate Design

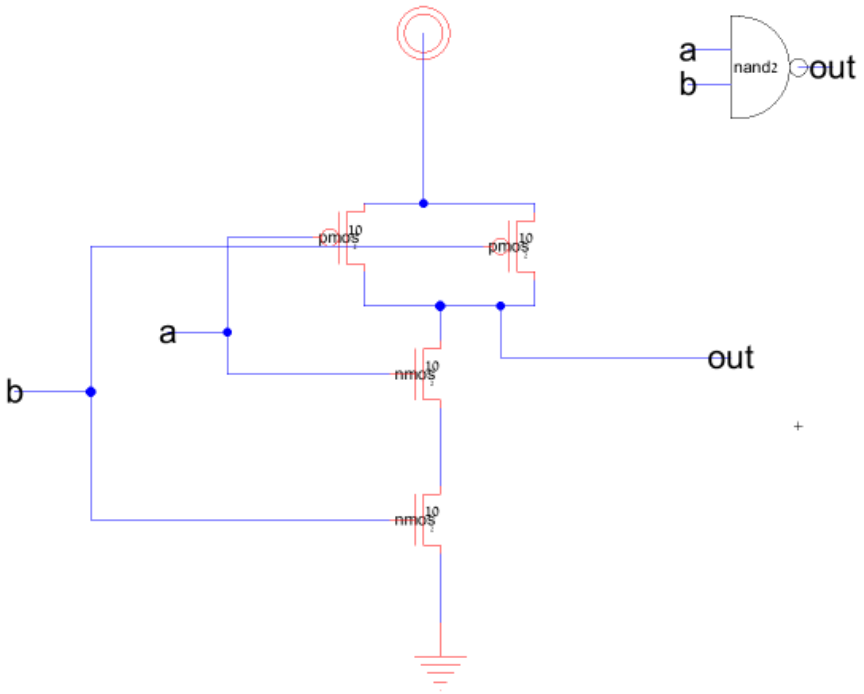

Figure 7: schematic design of nand gate

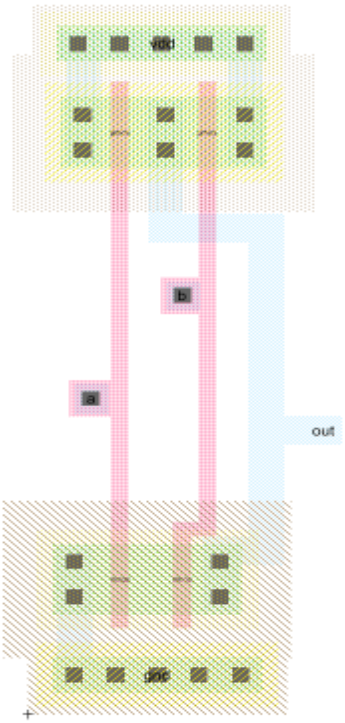

Figure 8: layout design of nand gate

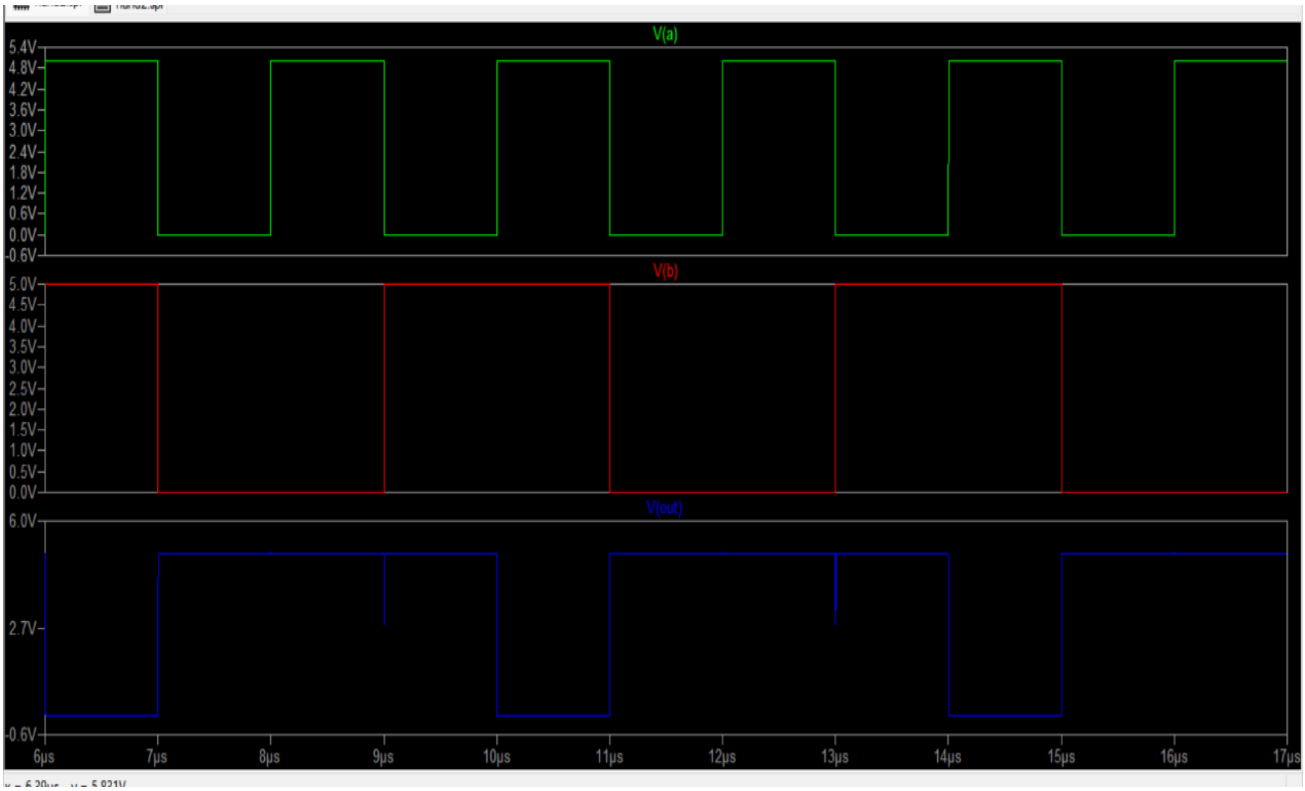

Figure 9:simulation of nand gate

LVS: Hierarchical NCC every cell in the design: cell 'nand2{sch}' cell 'nand2{lay}'

Comparing: newproj:nand2{sch} with: newproj:nand2{lay}

exports match, topologies match, sizes match in 0.0 seconds.

Summary for all cells: exports match, topologies match, sizes match

NCC command completed in: 0.0 seconds.

DRC: Running DRC with area bit on, extension bit on, Mosis bit

Checking again hierarchy .... (0.0 secs)

Found 11 networks

Checking cell 'nand2{lay}'

No errors/warnings found

0 errors and 0 warnings found (took 0.008 secs)

5.4 OR Gate Design:

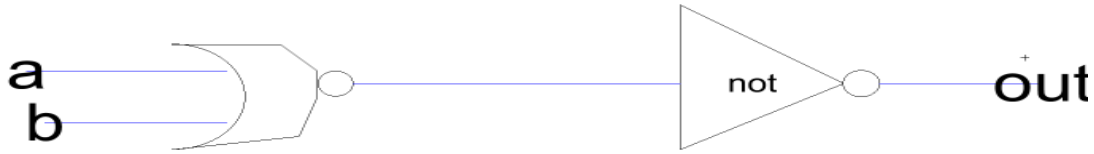

Figure 10:schematic design of or gate

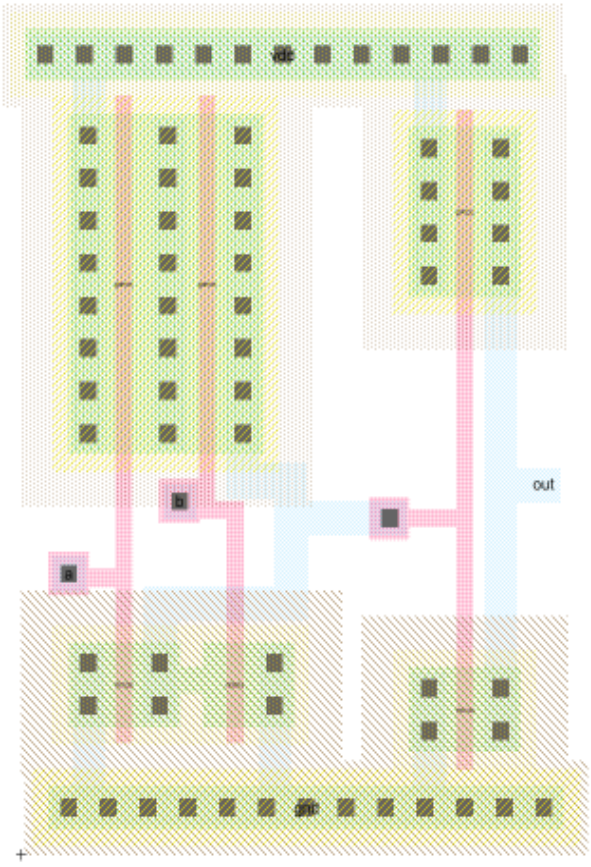

Figure 11:layout design of or gate

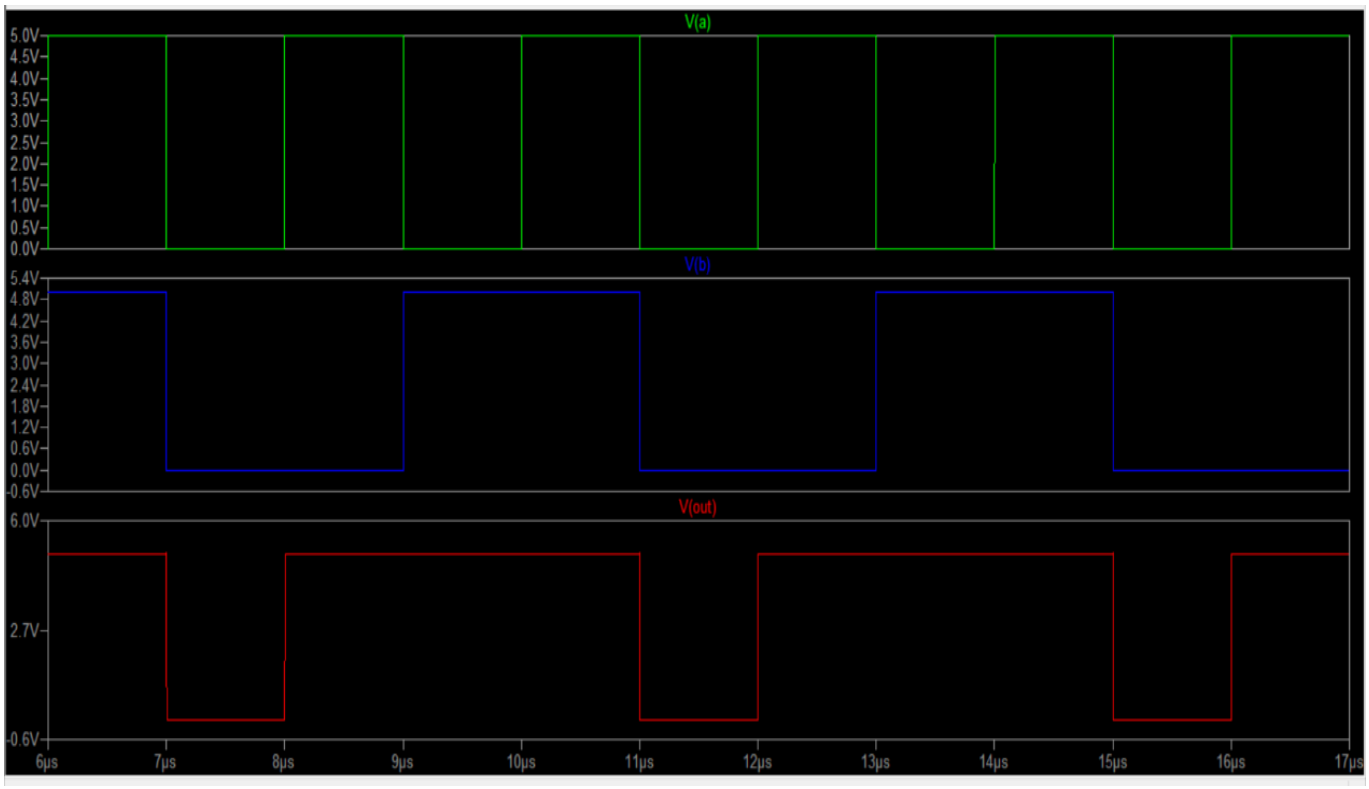

Figure 12: simulation of or gate

LVS: Hierarchical NCC every cell in the design: cell 'or{sch}' cell 'or{lay}'

Comparing: newproj:or{sch} with: newproj:or{lay}

exports match, topologies match, sizes match in 0.0 seconds.

Summary for all cells: exports match, topologies match, sizes match

NCC command completed in: 0.002 seconds.

DRC: Running DRC with area bit on, extension bit on, Mosis bit

Checking again hierarchy .... (0.01 secs)

Found 14 networks

Checking cell 'or{lay}'

No errors/warnings found

0 errors and 0 warnings found (took 0.02 secs)

5.5 AND Gate Design:

Figure 13:schematic design of and gate

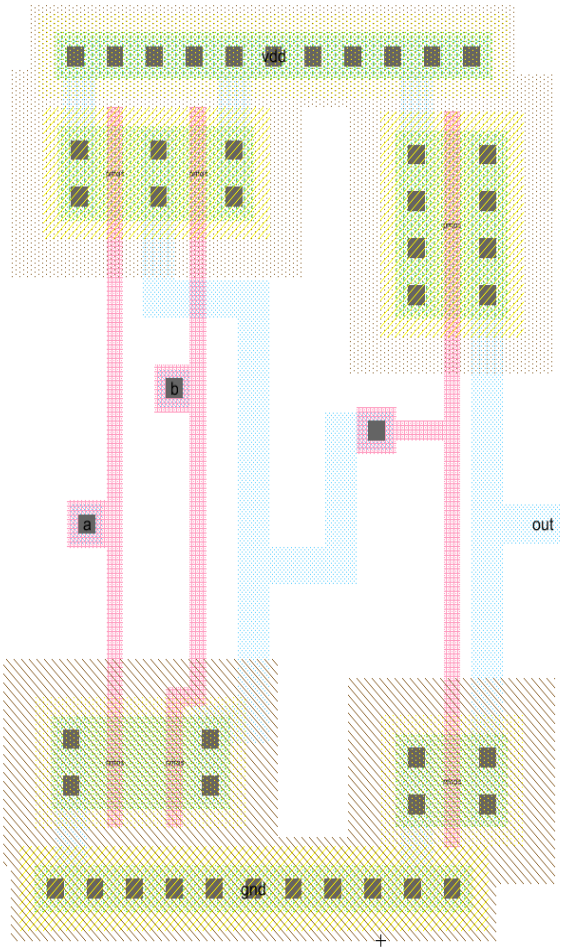

Figure 14:layout design of and gate

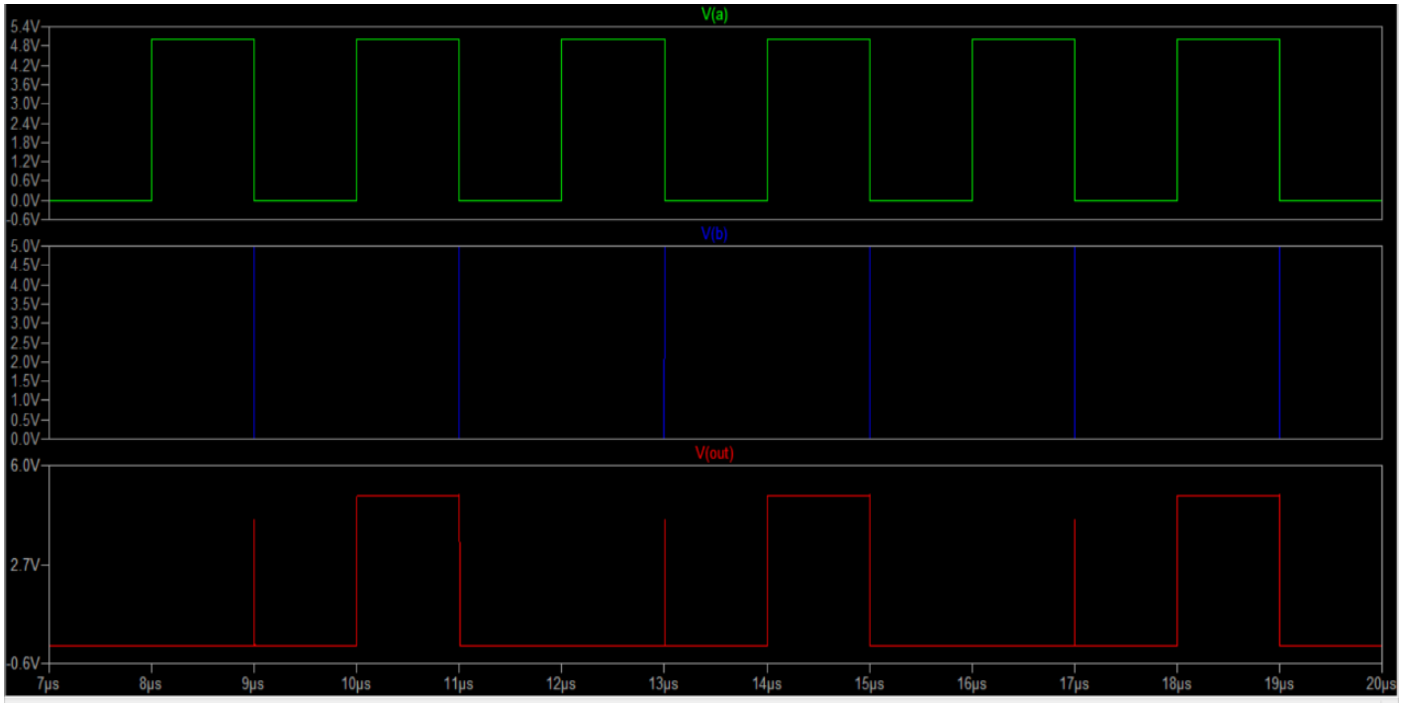

Figure 15: simulation of and gate

LVS: Hierarchical NCC every cell in the design: cell 'and{sch}' cell 'and{lay}'

Comparing: newproj:and{sch} with: newproj:and{lay}

exports match, topologies match, sizes match in 0.002 seconds.

Summary for all cells: exports match, topologies match, sizes match

NCC command completed in: 0.003 seconds.

DRC: Running DRC with area bit on, extension bit on, Mosis bit

Checking again hierarchy .... (0.0 secs)

Found 14 networks

Checking cell 'and{lay}'

No errors/warnings found

0 errors and 0 warnings found (took 0.011 secs)

5.6 XOR Gate Design:

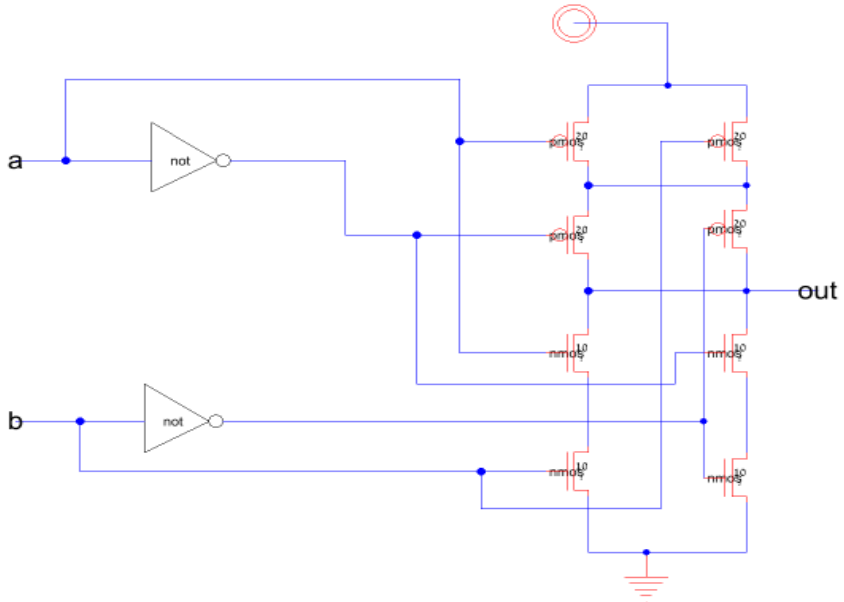

Figure 16: schematic design of xor gate

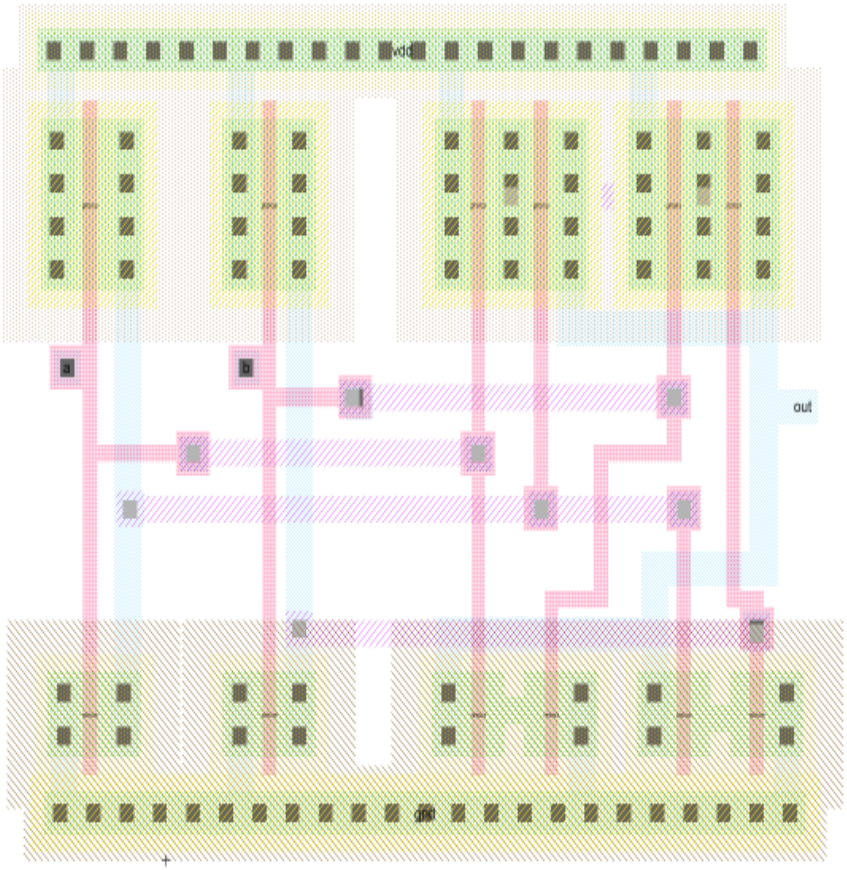

Figure 17: layout design of xor gate

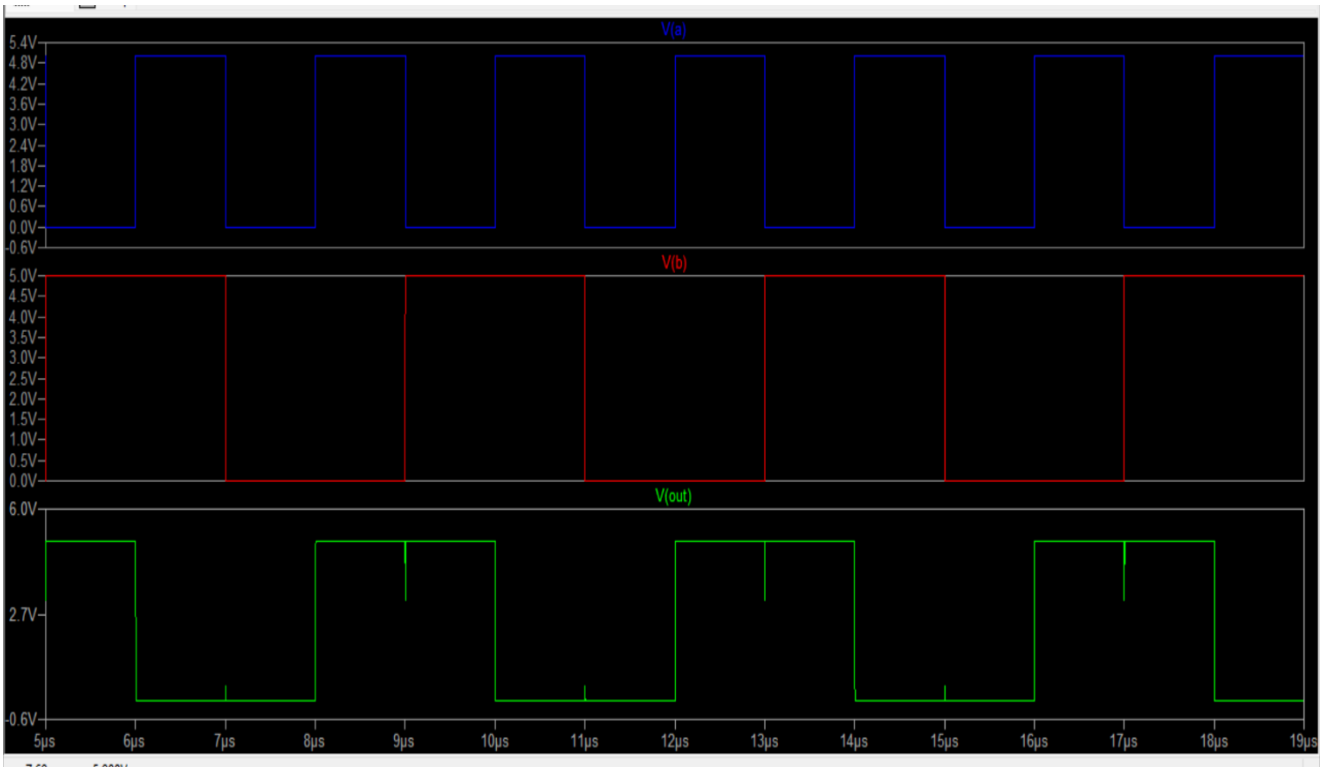

Figure 18:simulation of xor gate

LVS: Hierarchical NCC every cell in the design: cell 'xor{sch}' cell 'xor{lay}'

Comparing: newproj:xor{sch} with: newproj:xor{lay}

exports match, topologies match, sizes match in 0.003 seconds.

Summary for all cells: exports match, topologies match, sizes match

NCC command completed in: 0.003 seconds.

DRC: Running DRC with area bit on, extension bit on, Mosis bit

Checking again hierarchy .... (0.0 secs)

Found 23 networks

Checking cell 'xor{lay}'

No errors/warnings found

0 errors and 0 warnings found (took 0.04 secs)

5.7 2bit MUX Design:

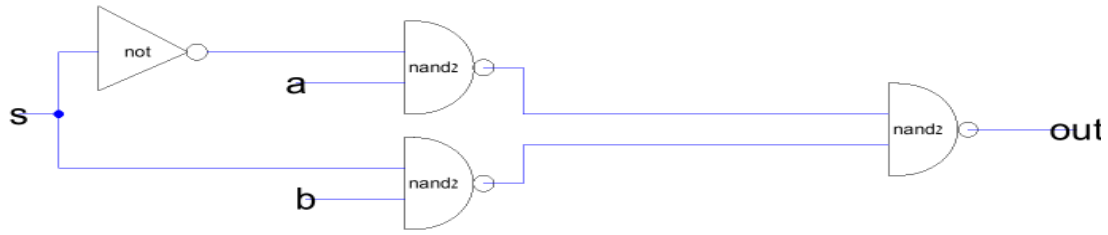

Figure 19: schematic design of 2bit mux

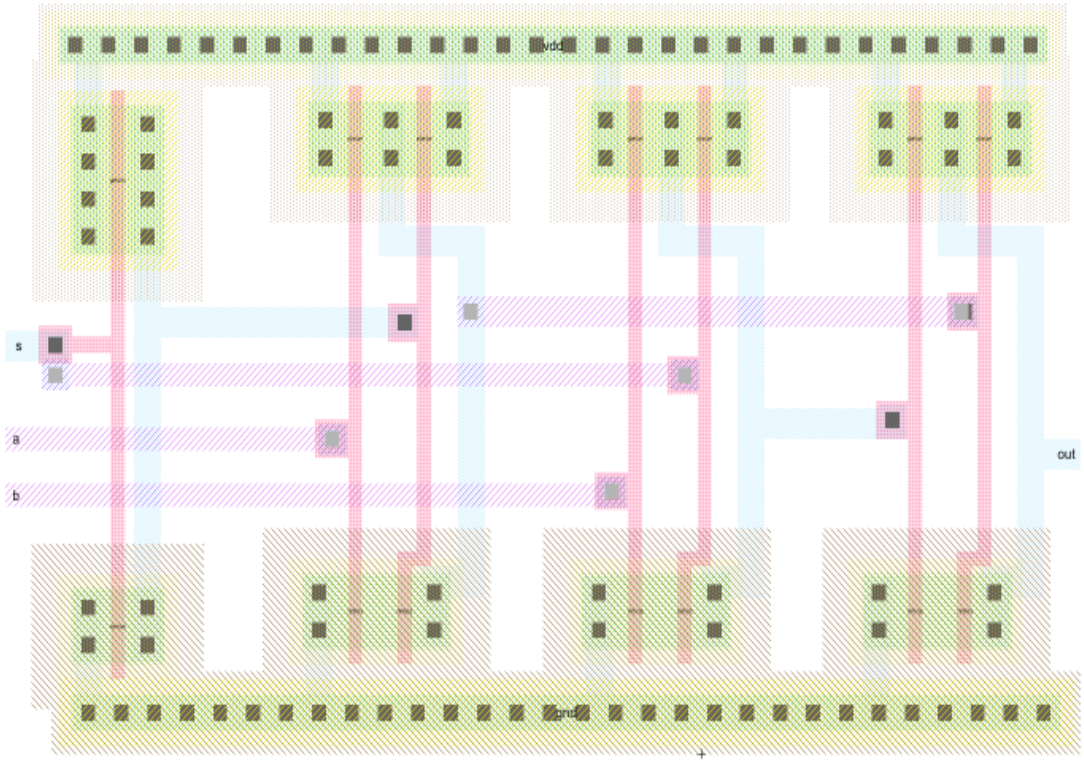

Figure 20: layout design of 2bit mux

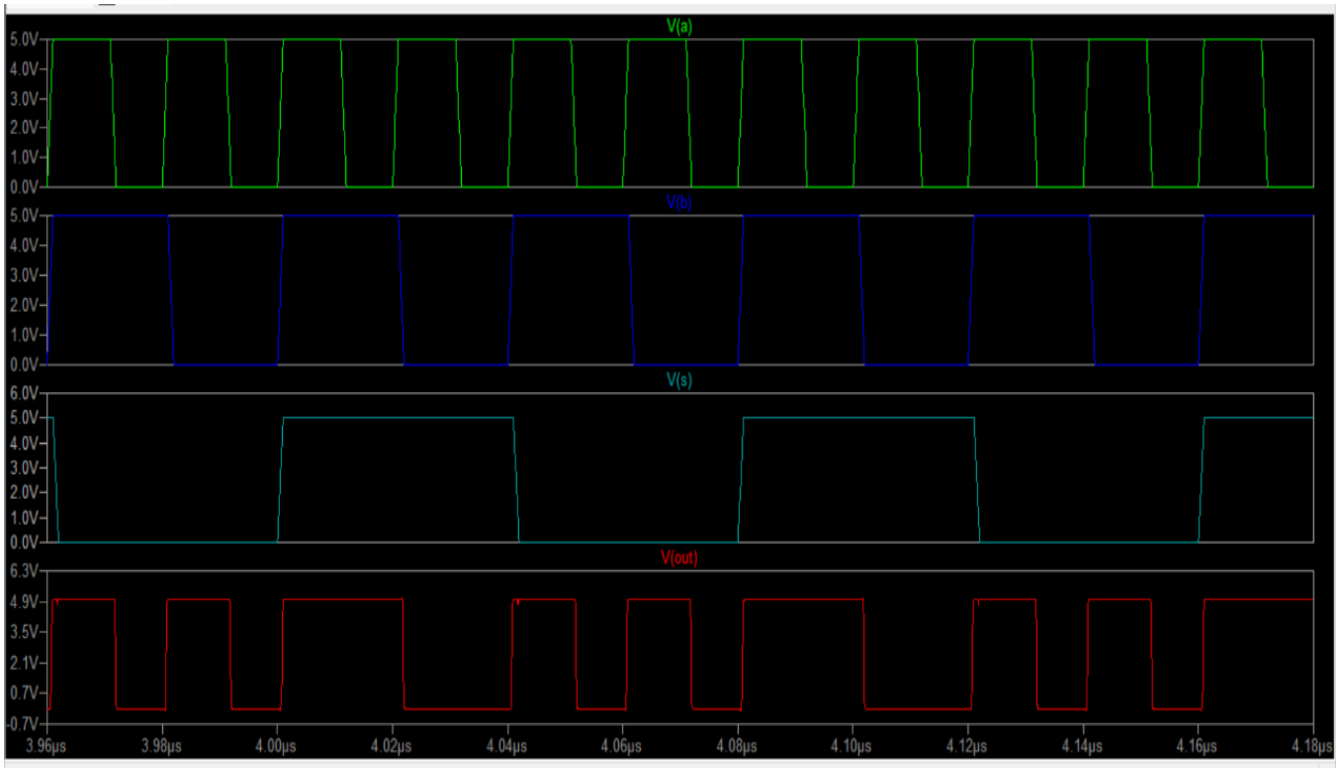

Figure 21:simulation of 2bit mux

LVS: Hierarchical NCC every cell in the design: cell '2mux1{sch}' cell '2mux1{lay}'

Comparing: newproj:2mux1{sch} with: newproj:2mux1{lay}

exports match, topologies match, sizes match in 0.067 seconds.

Summary for all cells: exports match, topologies match, sizes match

NCC command completed in: 0.083 seconds.

DRC: Running DRC with area bit on, extension bit on, Mosis bit

Checking again hierarchy .... (0.0 secs)

Found 27 networks

Checking cell '2mux1{lay}'

No errors/warnings found

0 errors and 0 warnings found (took 0.048 secs)

5.8 4bit MUX Design:

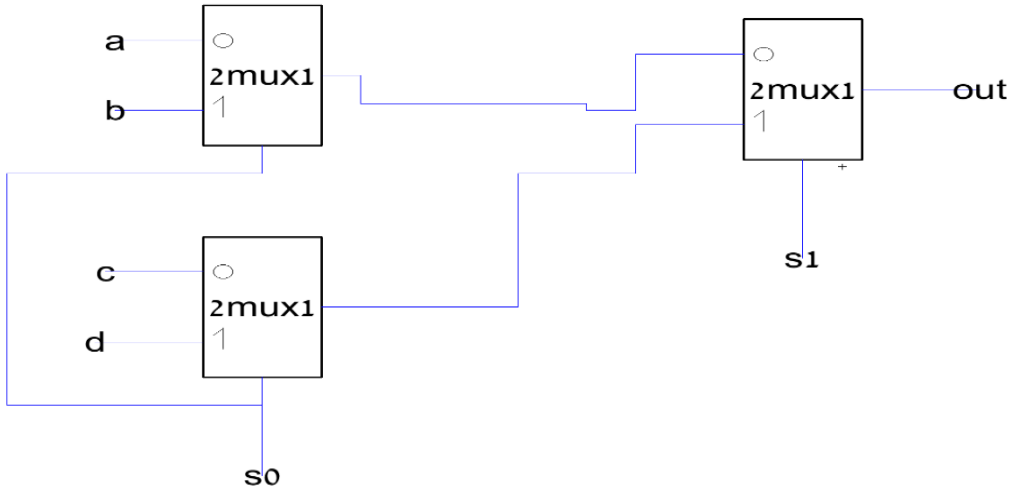

Figure 22: schematic design of 4bit mux

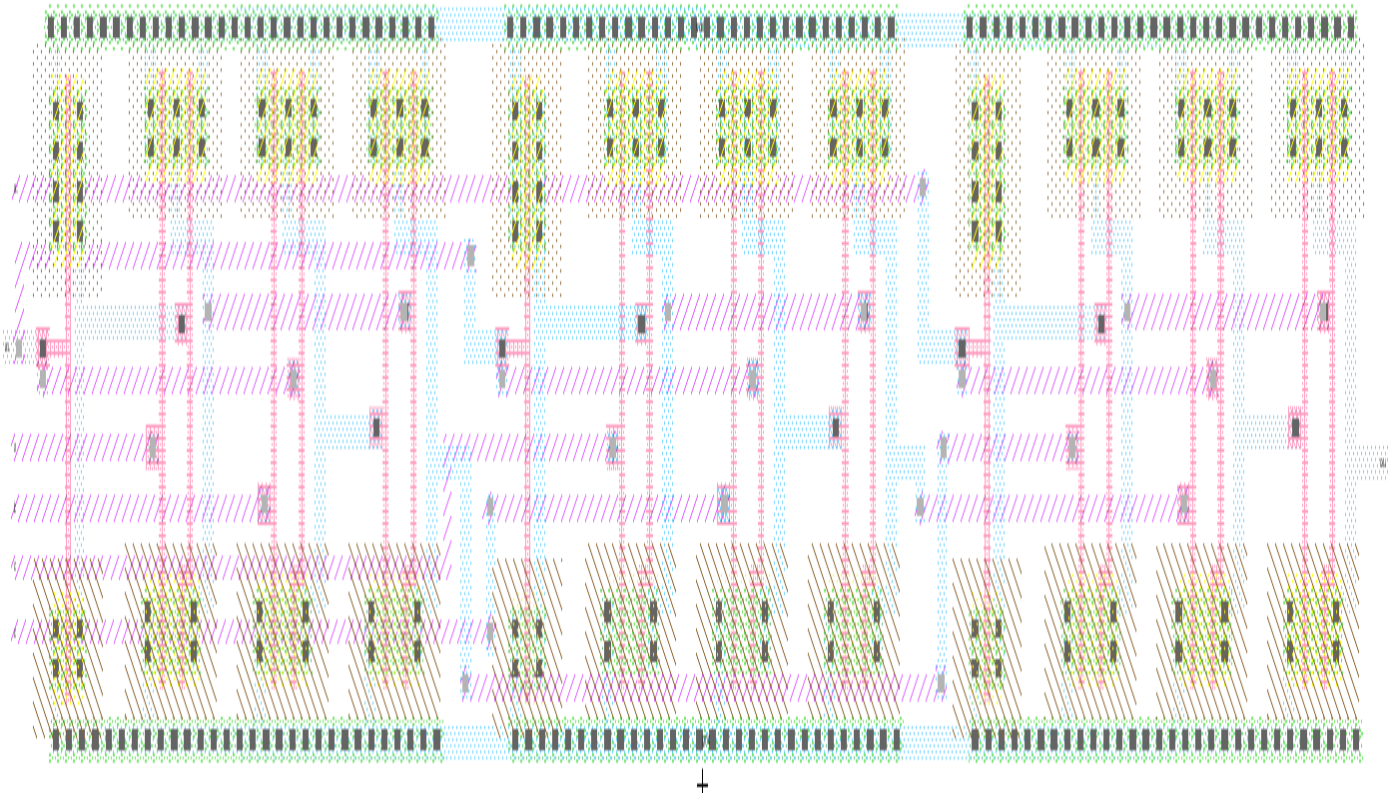

Figure 23: layout design of 4bit mux

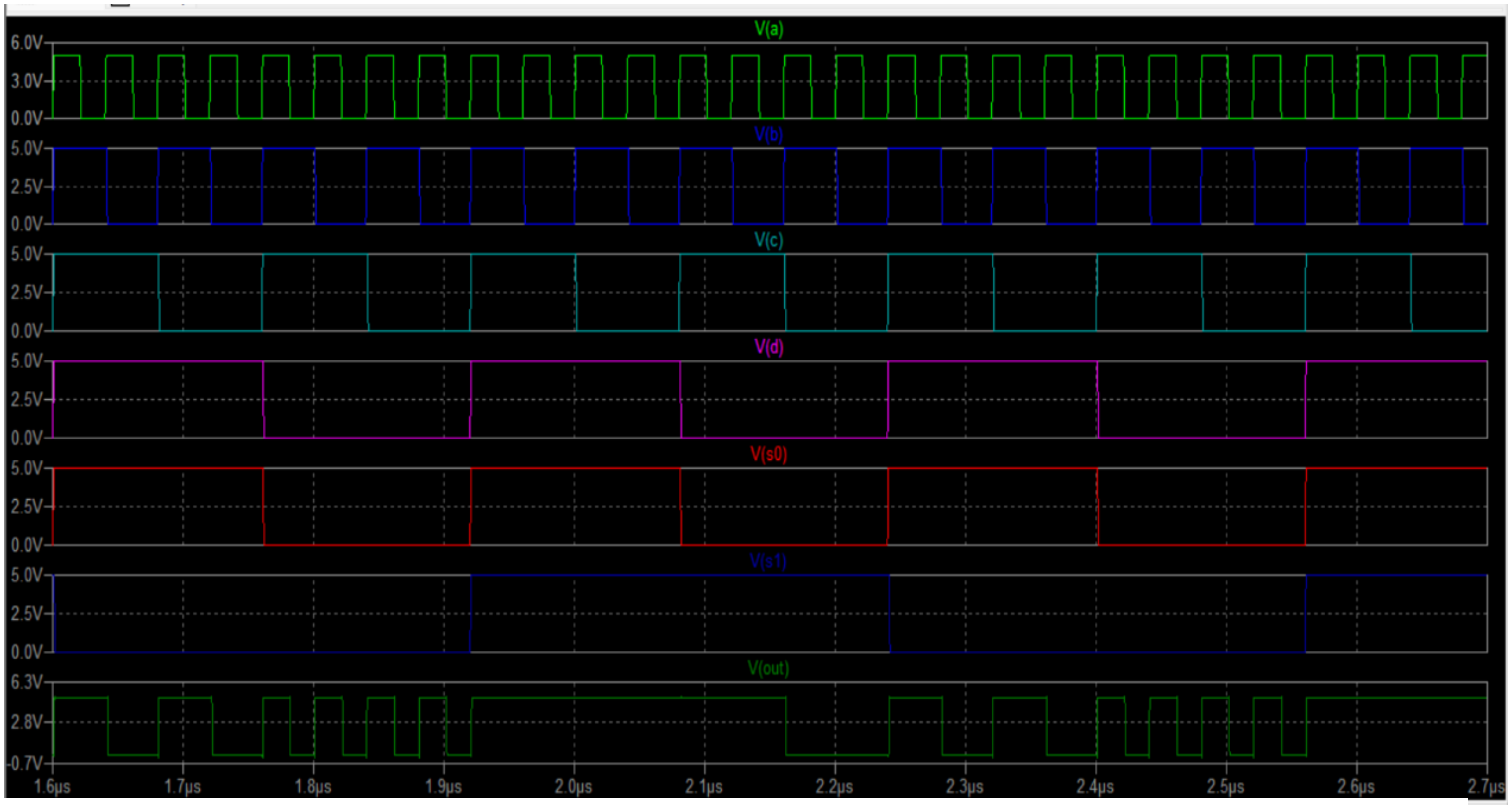

Figure 24:simulation of 4bit mux

LVS: Hierarchical NCC every cell in the design: cell '4mux1{sch}' cell '4mux1{lay}'

Comparing: newproj:2mux1{sch} with: newproj:2mux1{lay}

exports match, topologies match, sizes match in 0.012 seconds.

Comparing: newproj:4mux1{sch} with: newproj:4mux1{lay}

exports match, topologies match, sizes match in 0.005 seconds.

Summary for all cells: exports match, topologies match, sizes match

NCC command completed in: 0.022 seconds.

DRC: Running DRC with area bit on, extension bit on, Mosis bit

Checking again hierarchy .... (0.0 secs)

Found 12 networks

Checking cell '4mux1{lay}'

No errors/warnings found

0 errors and 0 warnings found (took 0.038 secs)

5.9 FULL ADDER Design:

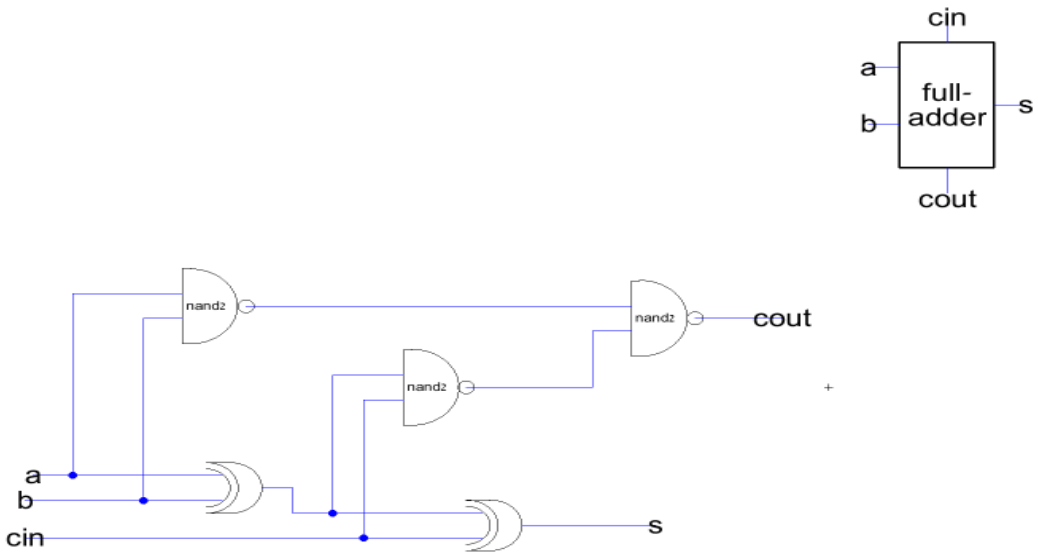

Figure 25: schematic design of full adder

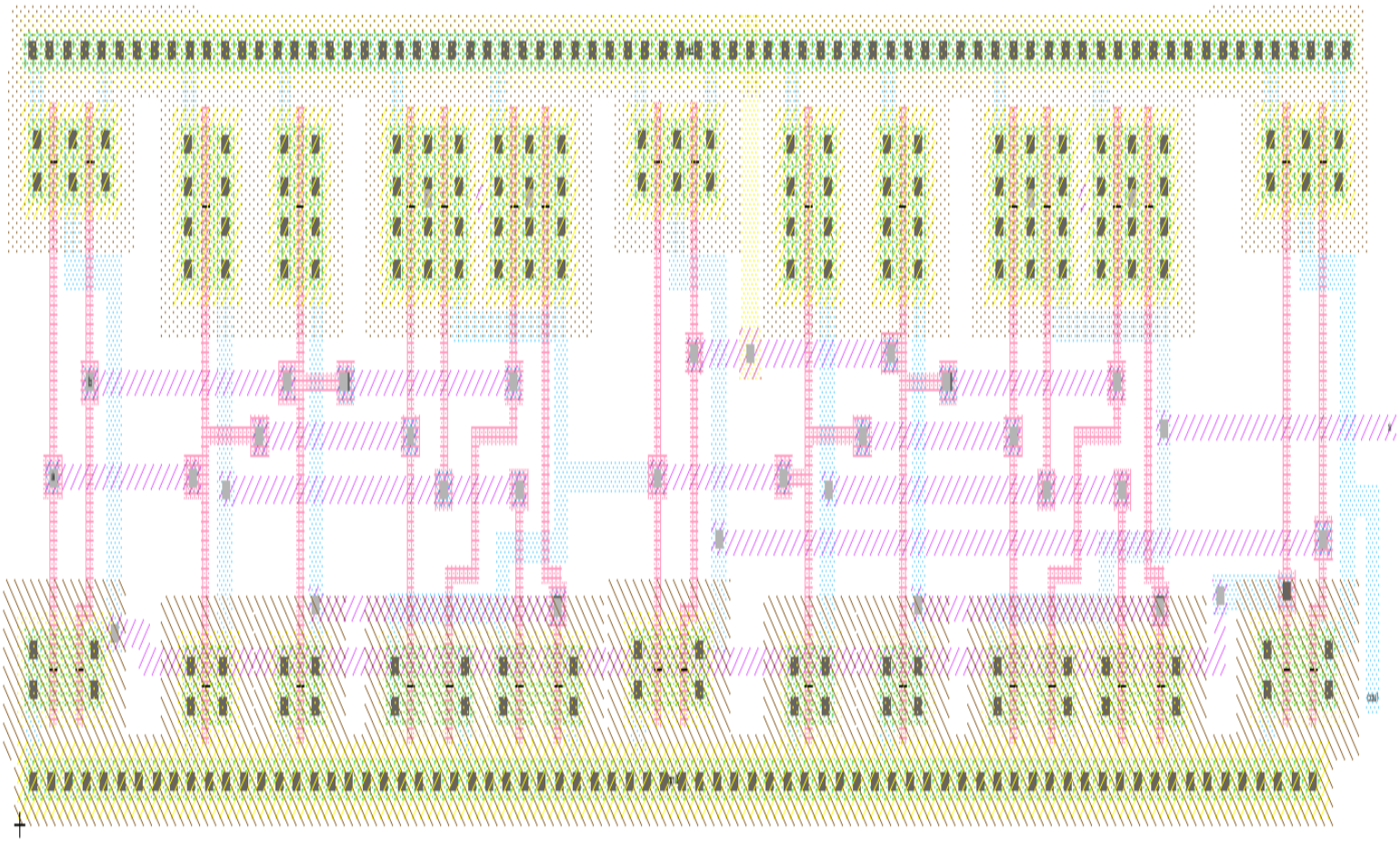

Figure 26: layout design of full adder

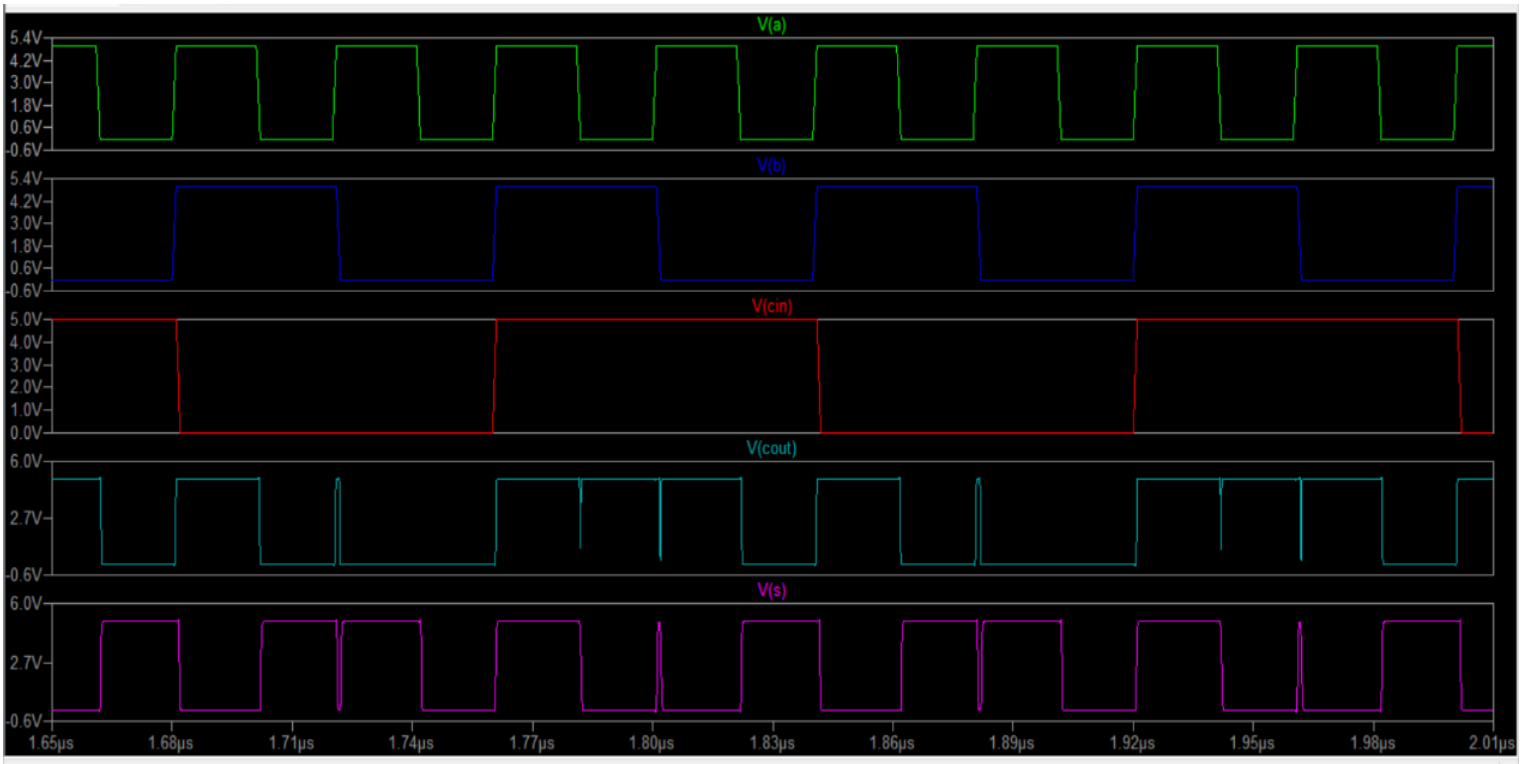

Figure 27: simulation of full adder

LVS: Hierarchical NCC every cell in the design: cell 'full-adder{sch}' cell 'full-adder{lay}'

Comparing: newproj:full-adder{sch} with: newproj:full-adder{lay}

exports match, topologies match, sizes match in 0.006 seconds.

Summary for all cells: exports match, topologies match, sizes match

NCC command completed in: 0.007 seconds.

DRC: Running DRC with area bit on, extension bit on, Mosis bit

Checking again hierarchy .... (0.0 secs)

Found 60 networks

Checking cell 'full-adder{lay}'

No errors/warnings found

0 errors and 0 warnings found (took 0.134 secs)

5.10 1bit AU Design:

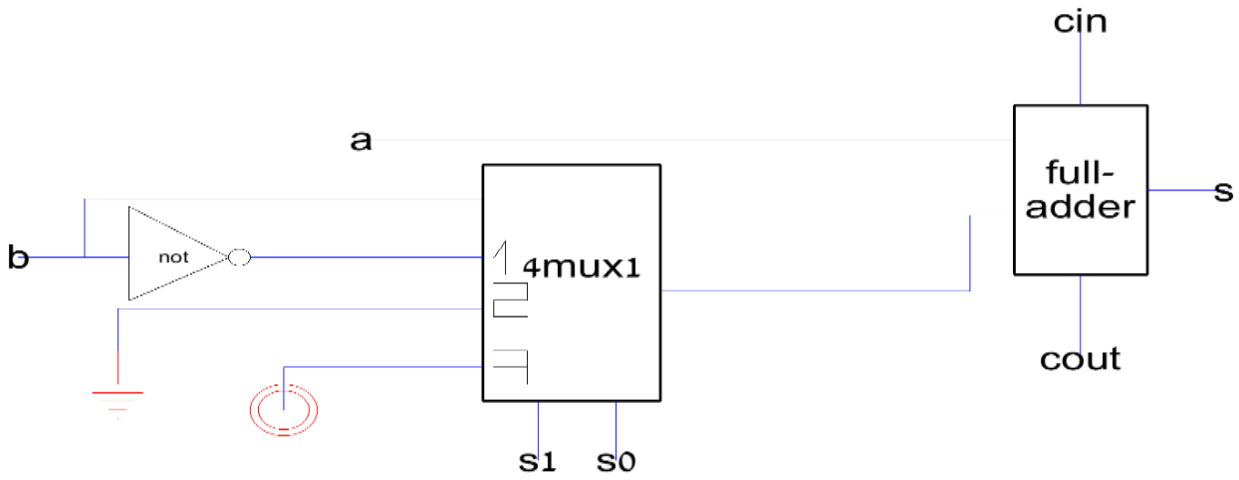

Figure 28: schematic design of 1bit arithmetic unit

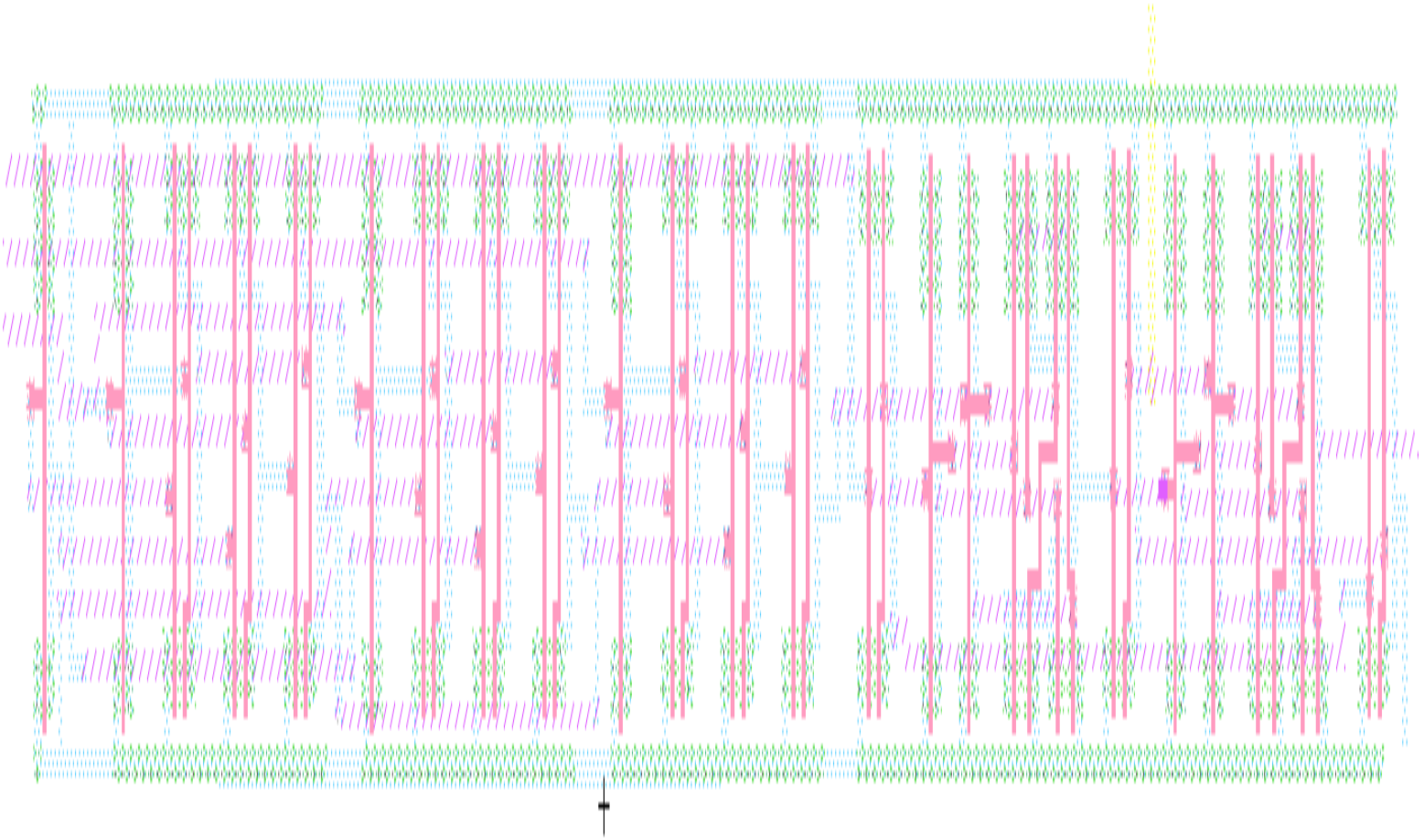

Figure 29: layout design of 1bit arithmetic unit

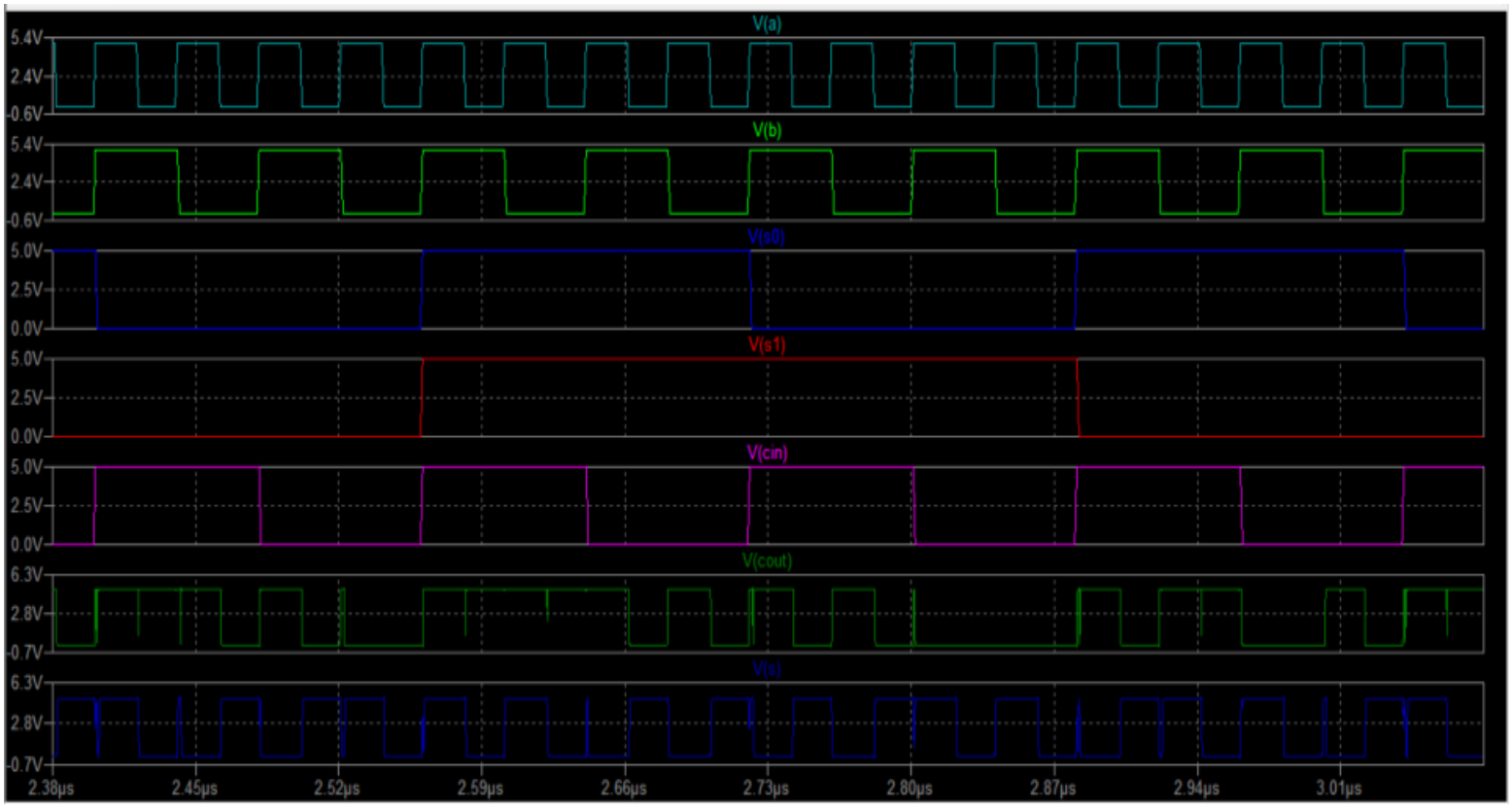

Figure 30:simulation of 1bit arithmetic unit

This circuit implements the arithmetic portion of the ALU, based on the design from the *Mano Computer Architecture* book. It performs operations such as addition, subtraction (using 2's complement), increment, and transfer of input values. The unit takes two 1-bit inputs (A and B) and a carry-in (Cin), and generates a 1-bit output along with a carry-out. It uses a full adder combined with logic controlled by select lines to determine the operation. This module is central to executing basic arithmetic instructions in the ALU.

LVS: Hierarchical NCC every cell in the design: cell '1bit-AU{sch}' cell '1bit-AU{lay}'

Comparing: newproj:not{sch} with: newproj:not{lay}

exports match, topologies match, sizes match in 0.003 seconds.

Comparing: newproj:2mux1{sch} with: newproj:2mux1{lay}

exports match, topologies match, sizes match in 0.0 seconds.

Comparing: newproj:4mux1{sch} with: newproj:4mux1{lay}

exports match, topologies match, sizes match in 0.0 seconds.

Comparing: newproj:full-adder{sch} with: newproj:full-adder{lay}

exports match, topologies match, sizes match in 0.002 seconds.

Comparing: newproj:1bit-AU{sch} with: newproj:1bit-AU{lay}

exports match, topologies match, sizes match in 0.0 seconds.

Summary for all cells: exports match, topologies match, sizes match

NCC command completed in: 0.02 seconds.

DRC: Running DRC with area bit on, extension bit on, Mosis bit

Checking again hierarchy .... (0.0 secs)

Found 12 networks

Checking cell '1bit-AU{lay}'

No errors/warnings found ,0 errors and 0 warnings found (took 0.076 sec

5.11 1bit LU Design:

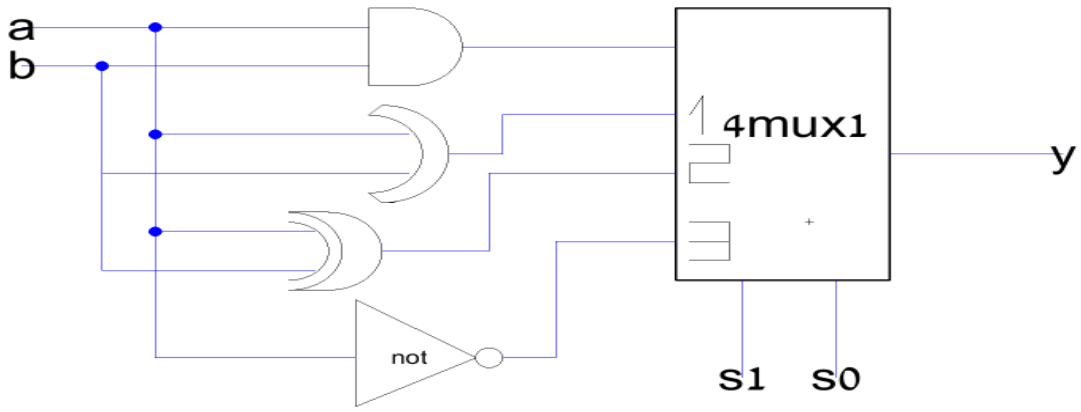

Figure 31: schematic design of 1bit logic unit

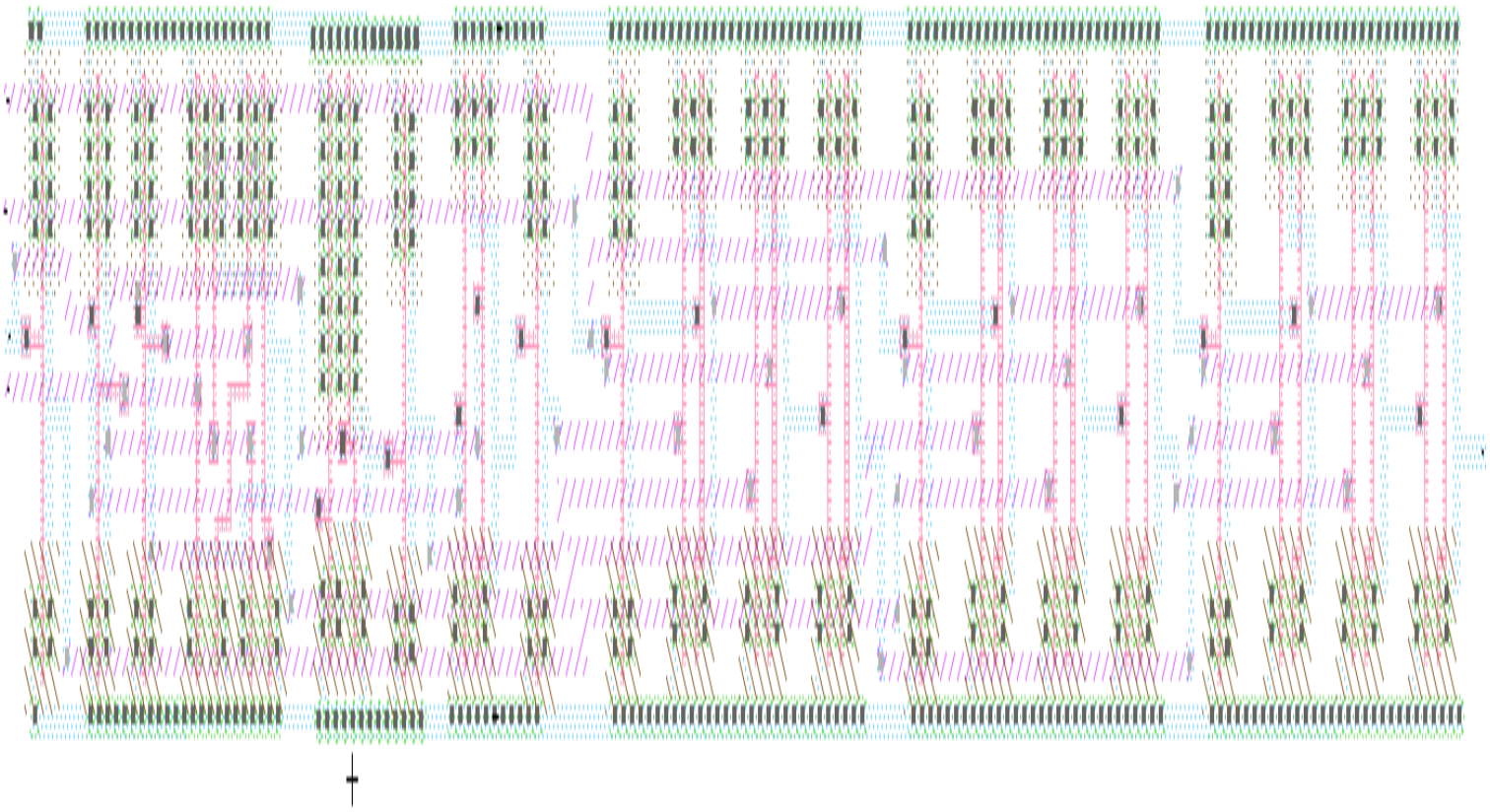

Figure 32: layout design of 1bit logic unit

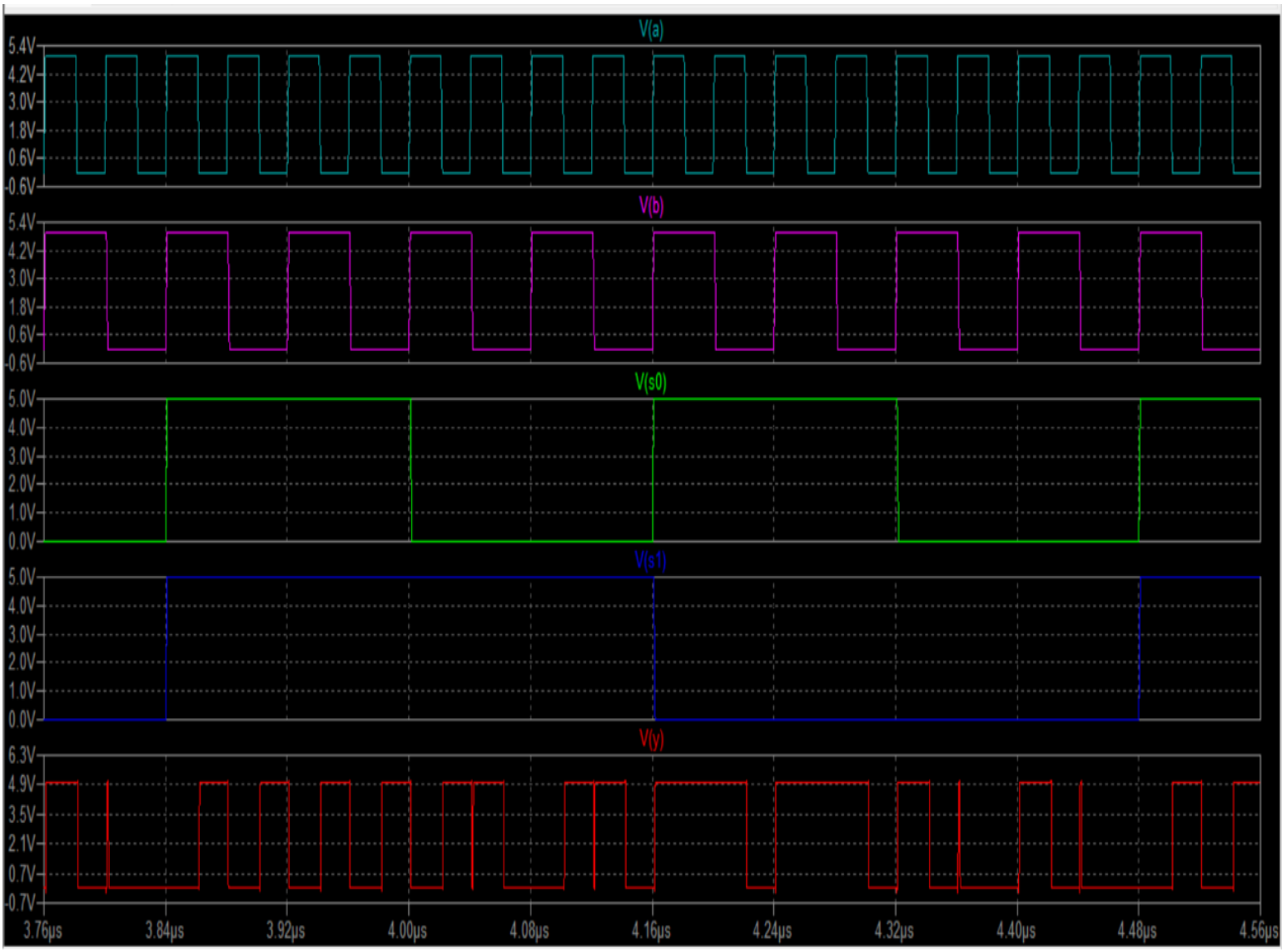

Figure 33: simulation of 1bit logic unit

This circuit is based on the design presented in the *Mano Computer Architecture* book. It performs bitwise logical operations—AND, OR, XOR, and NOT—on the 1-bit inputs A and B. Each operation is executed in parallel using basic logic gates. The outputs of these gates are connected to a 4-bit multiplexer, which selects one of the operations to pass to the output based on control signals. This approach provides a compact and modular solution for implementing logic functions, aligned with classical ALU architecture principles.

LVS: Hierarchical NCC every cell in the design: cell '1bit-LU{sch}' cell '1bit-LU{lay}'

Comparing: newproj:not{sch} with: newproj:not{lay}

exports match, topologies match, sizes match in 0.0 seconds.

Comparing: newproj:2mux1{sch} with: newproj:2mux1{lay}

exports match, topologies match, sizes match in 0.003 seconds.

Comparing: newproj:4mux1{sch} with: newproj:4mux1{lay}

exports match, topologies match, sizes match in 0.002 seconds.

Comparing: newproj:and{sch} with: newproj:and{lay}

exports match, topologies match, sizes match in 0.003 seconds.

Comparing: newproj:or{sch} with: newproj:or{lay}

exports match, topologies match, sizes match in 0.001 seconds.

Comparing: newproj:xor{sch} with: newproj:xor{lay}

exports match, topologies match, sizes match in 0.0 seconds.

Comparing: newproj:1bit-LU{sch} with: newproj:1bit-LU{lay}

exports match, topologies match, sizes match in 0.001 seconds.

Summary for all cells: exports match, topologies match, sizes match

NCC command completed in: 0.015 seconds.

DRC: Running DRC with area bit on, extension bit on, Mosis bit

Checking again hierarchy .... (0.0 secs)

Found 12 networks 0 errors and 0 warnings found (took 0.004 secs)

5.12 4bit arithmetic and logic shifter:

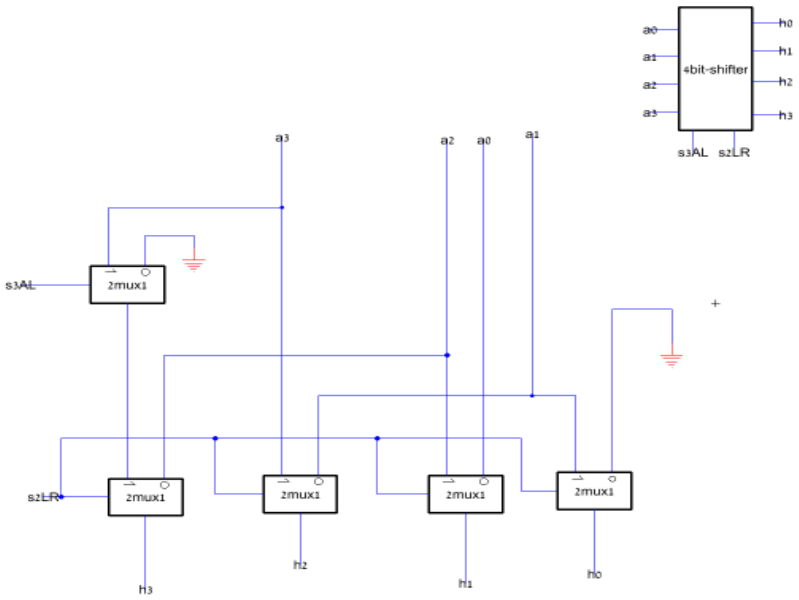

Figure 34: schematic design of 4bit shifter

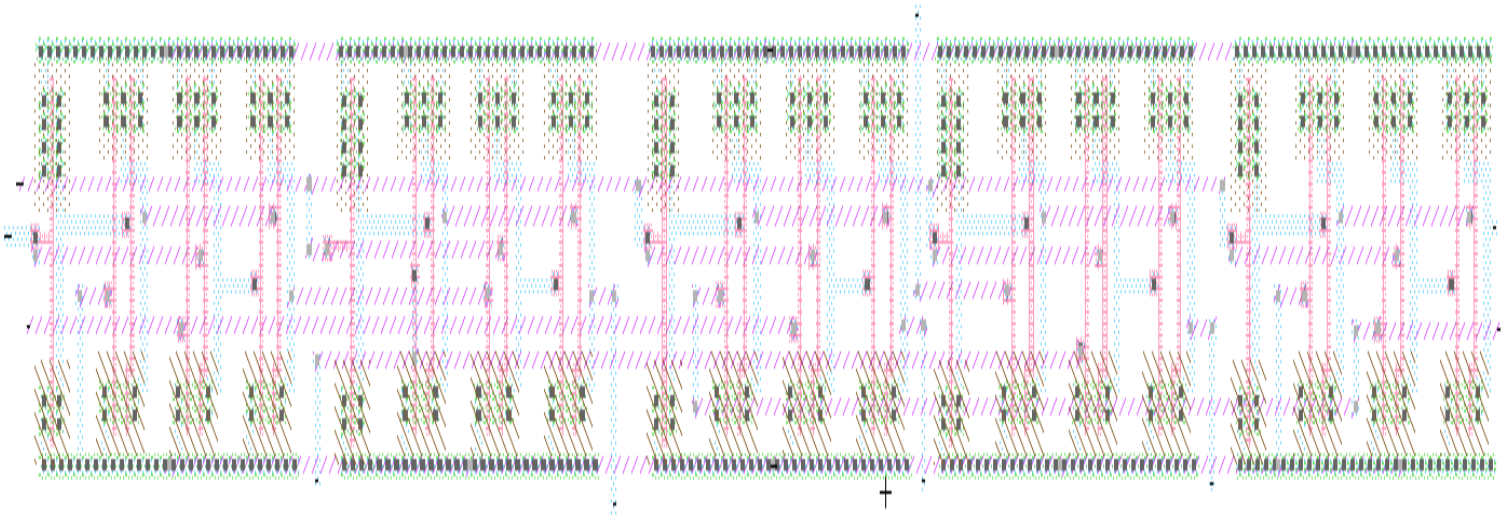

Figure 35:layout design of 4bit shifter

This circuit implements a 4-bit bidirectional shifter capable of performing both **arithmetic** and **logical** shifts to the left or right.

Functionality

- **Inputs:**
    - a3–a0: 4-bit data input.
    - sAL: Selects between **arithmetic** and **logical** shift.
    - sLR: Determines the **shift direction** — left or right.
- **Outputs:**
    - h3–h0: 4-bit shifted output.

Working Principle

Each output bit is controlled by a **2:1 multiplexer**. The lower-level muxes perform the actual shifting of bits either to the left or right, depending on sLR. For example:

- A logical **left shift** moves bits toward higher indices, inserting 0 at LSB.
- A logical **right shift** moves bits toward lower indices, inserting 0 at MSB.
- An **arithmetic right shift** keeps the sign bit (MSB) constant to preserve number sign in 2's complement representation.

Role of the Top 2:1 Mux (sAL Control)

The **topmost 2:1 mux**, controlled by sAL, selects between two shift behaviors:

- When sAL = 0: selects **logical shift** behavior (inserts 0 at MSB on right shift).
- When sAL = 1: selects **arithmetic shift** behavior (inserts the original MSB value at MSB on right shift, preserving sign).

This allows the circuit to flexibly switch between arithmetic and logical shifts using a compact and modular design. The result is a shifter that supports:

- Logical shift left (LSL)
- Logical shift right (LSR)
- Arithmetic shift right (ASR)

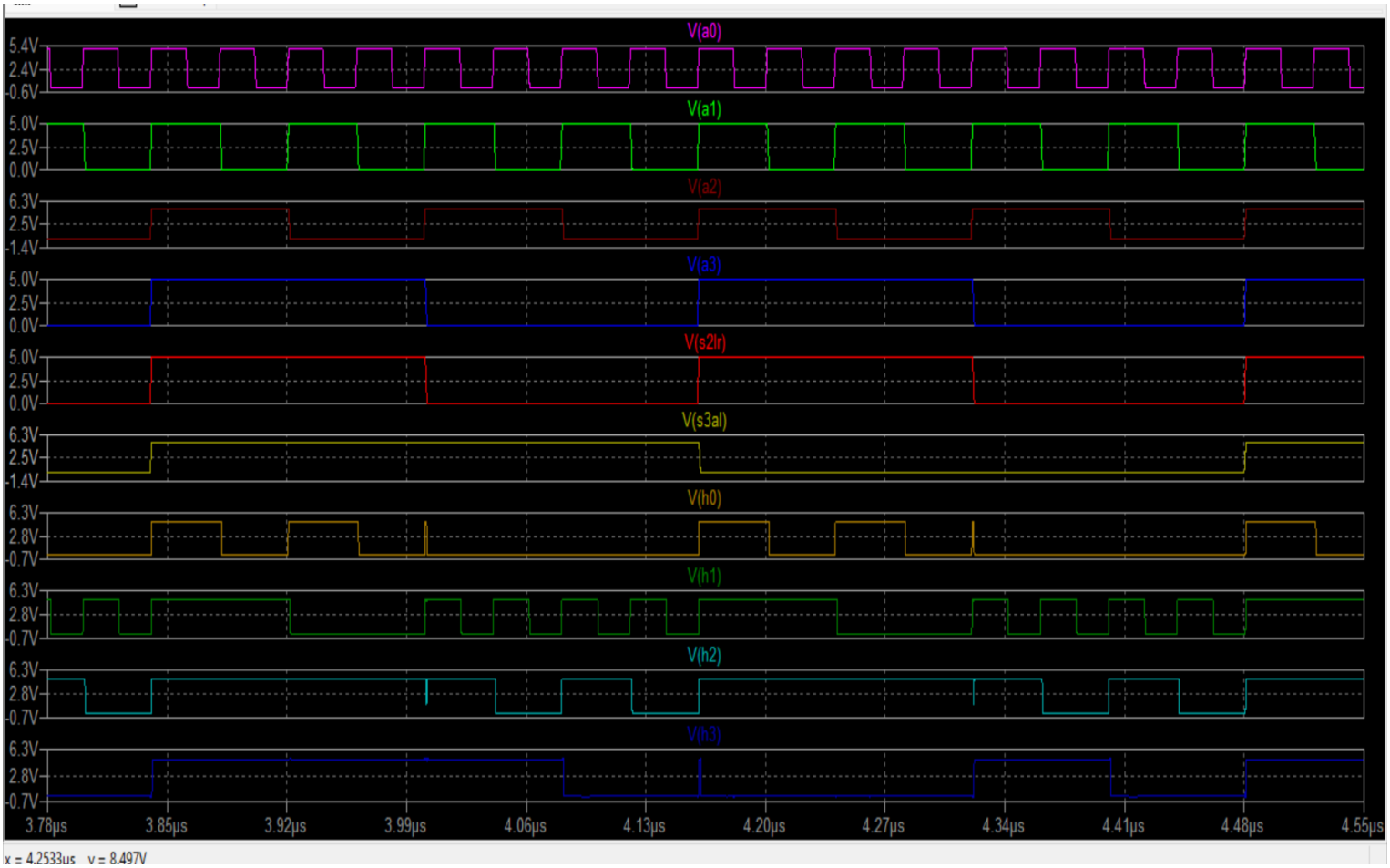

Figure 36 : simulation of shifter unit (s3=1 arithmetic shift ,s3=0 logical shift ,(s2=1 shift right , s2=0 shift left)

DRC: Running DRC with area bit on, extension bit on, Mosis bit

Checking again hierarchy .... (0.0 secs)

Found 114 networks

Checking cell '4bit-shifter{lay}'

No errors/warnings found

0 errors and 0 warnings found (took 0.596 secs)

LVS: Hierarchical NCC every cell in the design: cell '4bit-shifter{sch}' cell '4bit-shifter{lay}'

Comparing: newproj:4bit-shifter{sch} with: newproj:4bit-shifter{lay}

exports match, topologies match, sizes match in 0.029 seconds.

Summary for all cells: exports match, topologies match, sizes match

NCC command completed in: 0.035 seconds.

5.13 1bit ALU Design:

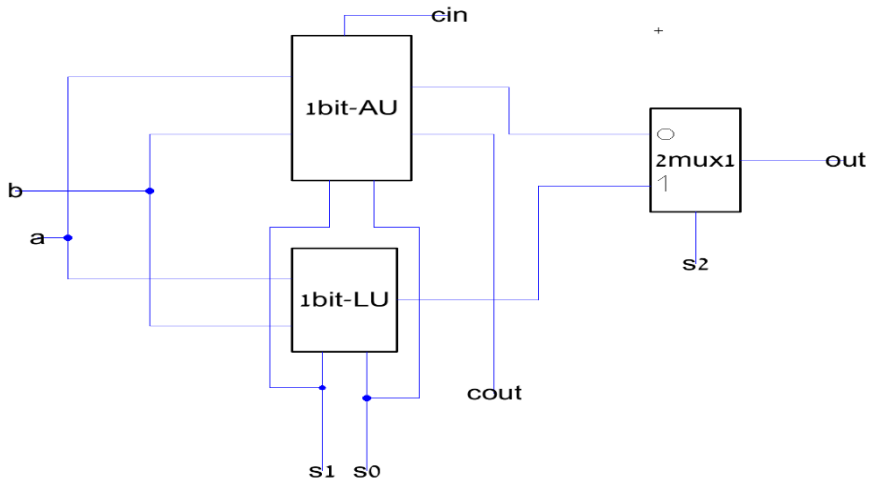

Figure 37: schematic design of 1bit alu

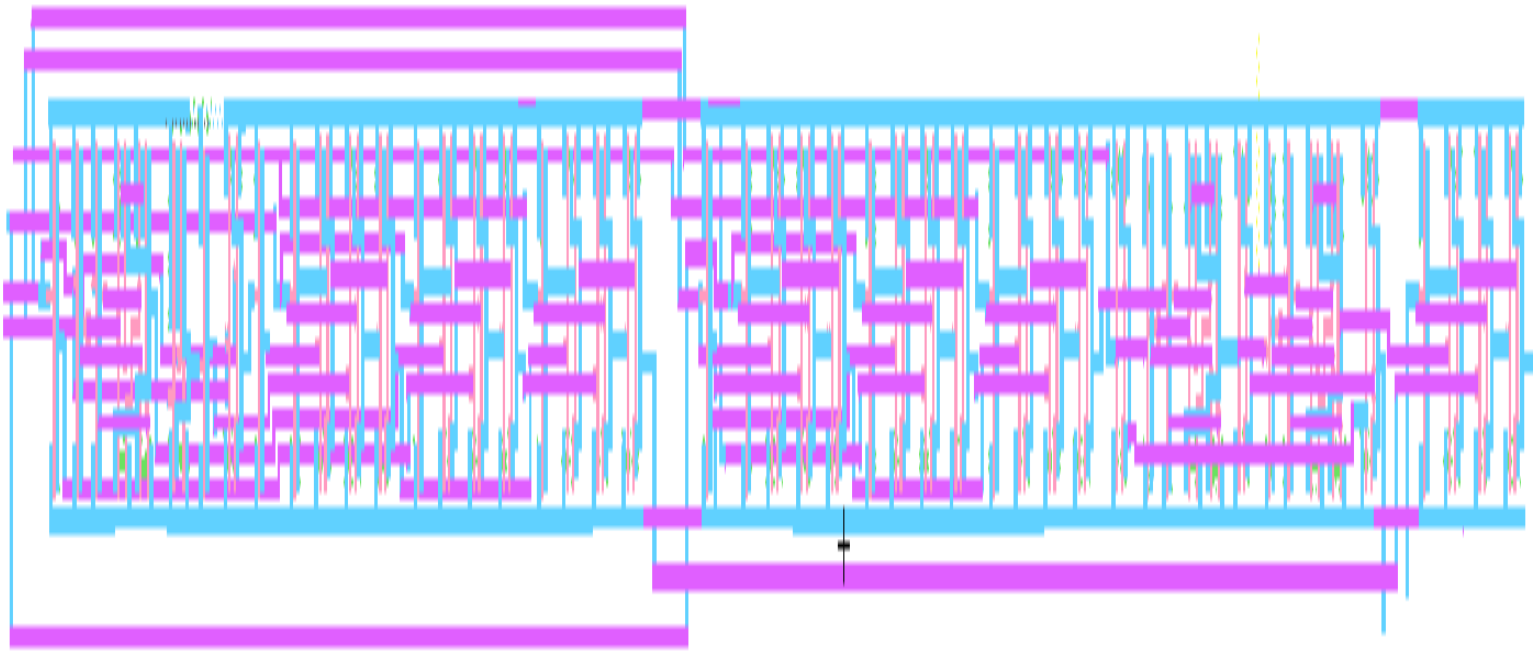

Figure 38: layout design of 1bit alu

This figure shows the design of a single-bit Arithmetic and Logic Unit (ALU) .The circuit combines a **1-bit Arithmetic Unit (AU)** and a **1-bit Logic Unit (LU)**, with a **2:1 multiplexer** to select between their outputs based on control signal S2.

- **Inputs**:
    - a and b: The primary data inputs.
    - cin: Carry-in for arithmetic operations.
    - s1, s0: Control signals for the logic unit to choose the specific arithmetic or logical operation (e.g., AND, OR, XOR, NOT).
    - s2: Control signal that determines whether the output comes from the Arithmetic Unit (s2 = 0) or Logic Unit (s2 = 1).
- **Outputs**:
    - out: Final result of the selected operation (arithmetic or logic).
    - cout: Carry-out from the Arithmetic Unit for multi-bit arithmetic chaining.

Functionality

This unit allows performing both arithmetic (like addition or increment) and logic operations (like AND, OR) at the 1-bit level. The use of the multiplexer allows switching between arithmetic and logic results dynamically during execution. When scaled up and combined, four of these units form a complete 4-bit ALU system.

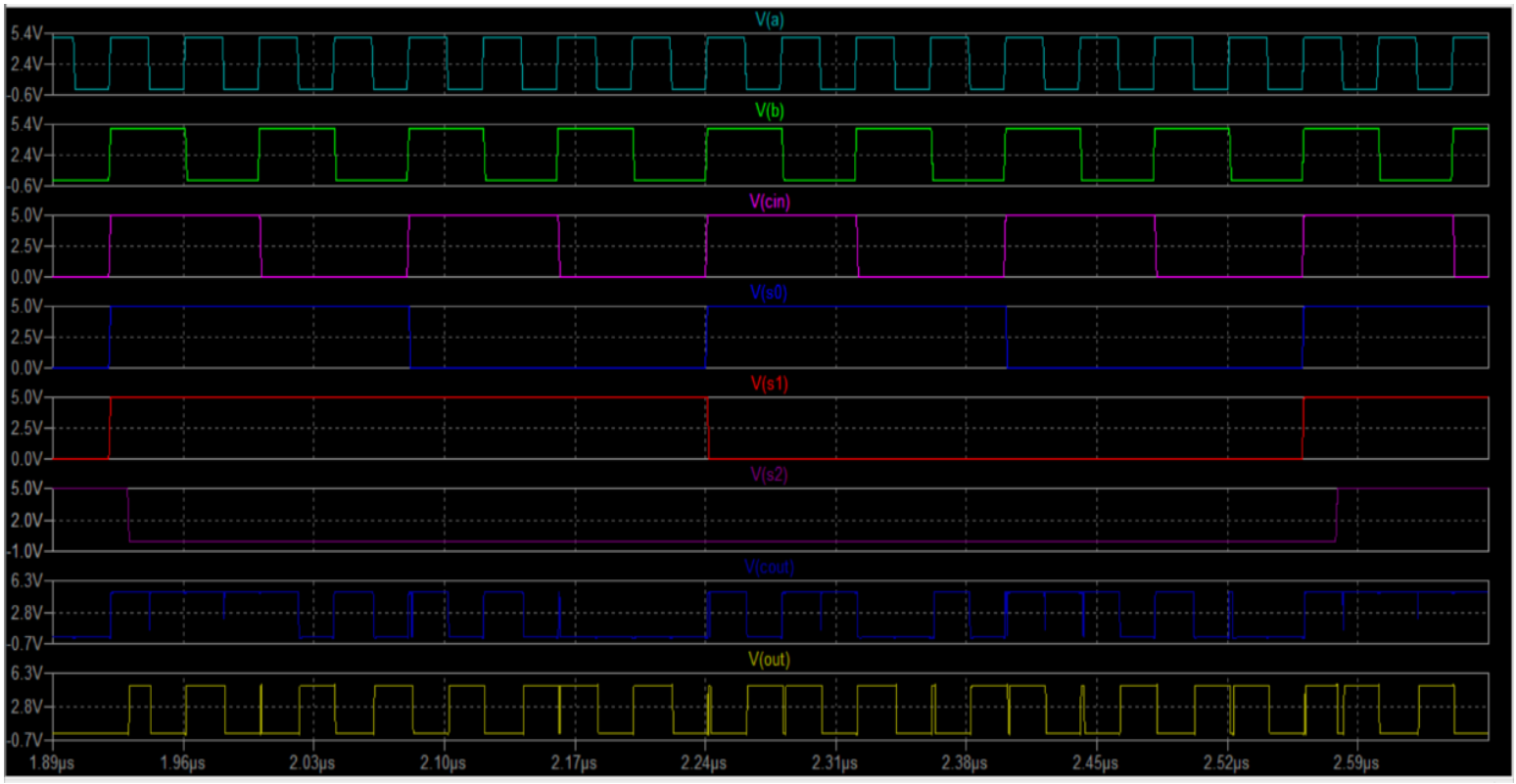

Figure 39 : simulation of 1bit alu (s2=0)

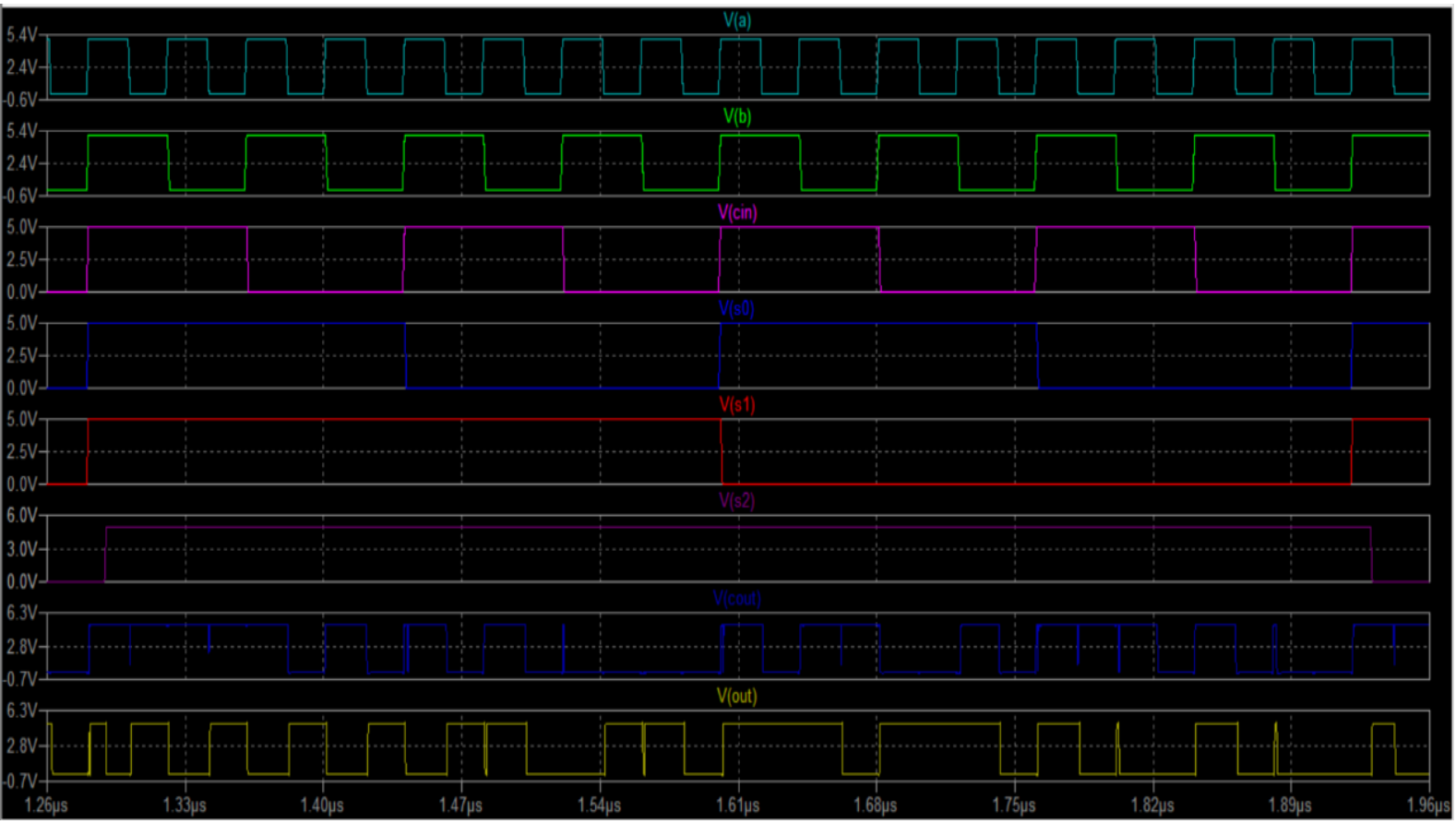

Figure 40: simulation of 1bit alu (s2=1)

LVS: Hierarchical NCC every cell in the design: cell '1bitALU{sch}' cell '1bitALU{lay}'

Comparing: newproj:not{sch} with: newproj:not{lay}

exports match, topologies match, sizes match in 0.001 seconds.

Comparing: newproj:2mux1{sch} with: newproj:2mux1{lay}

exports match, topologies match, sizes match in 0.0 seconds.

Comparing: newproj:4mux1{sch} with: newproj:4mux1{lay}

exports match, topologies match, sizes match in 0.004 seconds.

Comparing: newproj:xor{sch} with: newproj:xor{lay}

exports match, topologies match, sizes match in 0.002 seconds.

Comparing: newproj:full-adder{sch} with: newproj:full-adder{lay}

exports match, topologies match, sizes match in 0.004 seconds.

Comparing: newproj:1bit-AU{sch} with: newproj:1bit-AU{lay}

exports match, topologies match, sizes match in 0.0 seconds.

Comparing: newproj:and{sch} with: newproj:and{lay}

exports match, topologies match, sizes match in 0.0 seconds.

Comparing: newproj:or{sch} with: newproj:or{lay}

exports match, topologies match, sizes match in 0.001 seconds.

Comparing: newproj:1bit-LU{sch} with: newproj:1bit-LU{lay}

exports match, topologies match, sizes match in 0.001 seconds.

Comparing: newproj:1bitALU{sch} with: newproj:1bitALU{lay}

exports match, topologies match, sizes match in 0.0 seconds.

Summary for all cells: exports match, topologies match, sizes match

NCC command completed in: 0.023 seconds.

DRC: Running DRC with area bit on, extension bit on, Mosis bit

Checking again hierarchy .... (0.0 secs)

Found 13 networks

Checking cell '1bitALU{lay}'

No errors/warnings found

0 errors and 0 warnings found (took 0.184 secs)

5.14 4bit ALU Design:

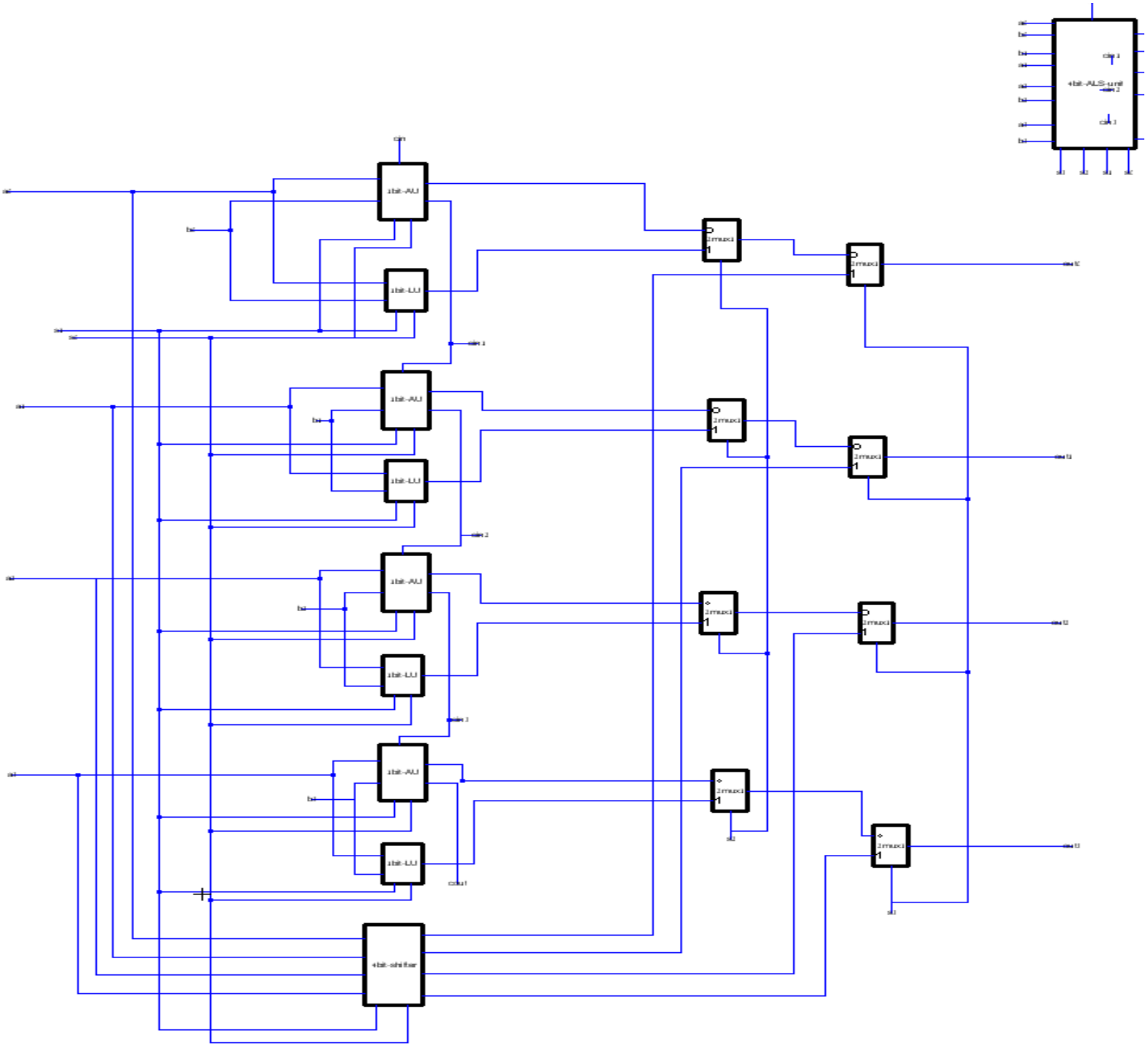

Figure 41: schematic design of 4bit arithmetic logic shift unit

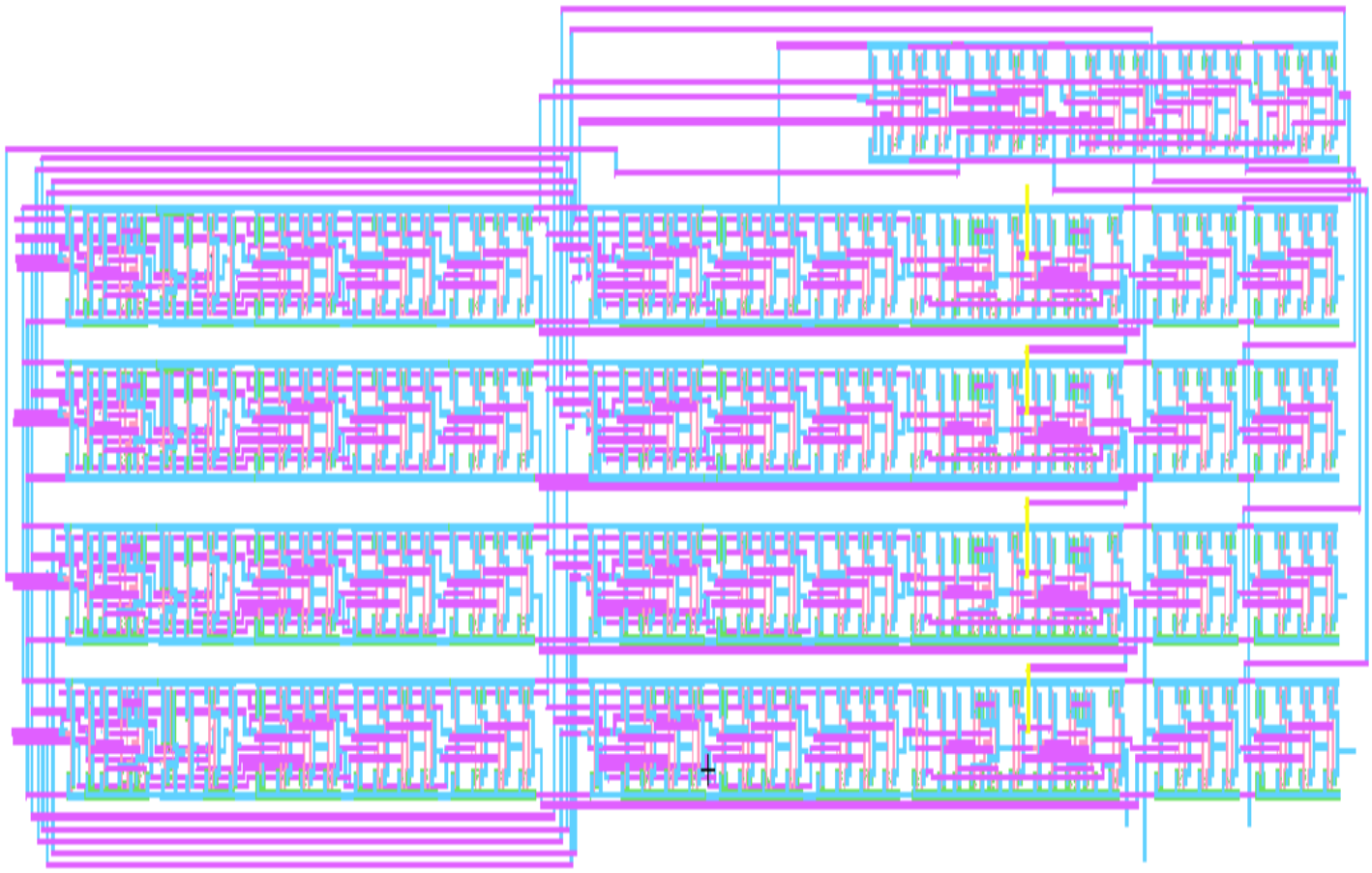

Figure 42: layout design of 4bit arithmetic logical and shift unit

This figure illustrates the full design of a 4-bit Arithmetic Logic Shift Unit (ALSU), based on the architecture described in the *Mano Computer Architecture* textbook. The design integrates **four 1-bit Arithmetic and Logic Units (ALUs)** and a **4-bit Shifter Unit**, providing flexible and programmable computation at the 4-bit level.

Component Overview:

- **1-Bit ALUs**: Perform arithmetic (e.g., addition, increment) and logic (e.g., AND, OR) operations.
- **4-Bit Shifter Unit**: Performs logical or arithmetic shifts left or right.
- **First Layer of 2:1 Multiplexers (MUX1)**: Controlled by signal S2 to select between the **Arithmetic** and **Logic** output from each 1-bit ALU.
- **Second Layer of 2:1 Multiplexers (MUX2)**: Controlled by signal S3 to select between the **output of the ALU** (from MUX1) or the **output of the Shifter Unit**.

Control Logic Explanation:

The **first control layer (MUX1)** determines the functional path inside the ALU:

- If S2 = 0: The output from the **Arithmetic Unit** is selected.
- If S2 = 1: The output from the **Logic Unit** is selected.

However, the **second control layer (MUX2)** serves as a **global override**:

- If S3 = 0: The result from MUX1 (i.e., arithmetic or logic) is passed to the output.
- If S3 = 1: The **output from the Shifter Unit** is selected **regardless of the MUX1 selection**, effectively giving the **Shifter Unit highest priority** in operation control.

This hierarchy allows S3 to force a shift operation no matter what the arithmetic or logic configuration is—providing a simple but effective control mechanism.

Operation Selection Summary:

- S3 = 1: Use the **Shifter Unit**, controlled by S0, S1 (e.g., shift left/right, logic/arithmetic shift).
- S3 = 0 and:
    - S2 = 0: Perform **Arithmetic** operation selected by S0, S1.
    - S2 = 1: Perform **Logic** operation selected by S0, S1.

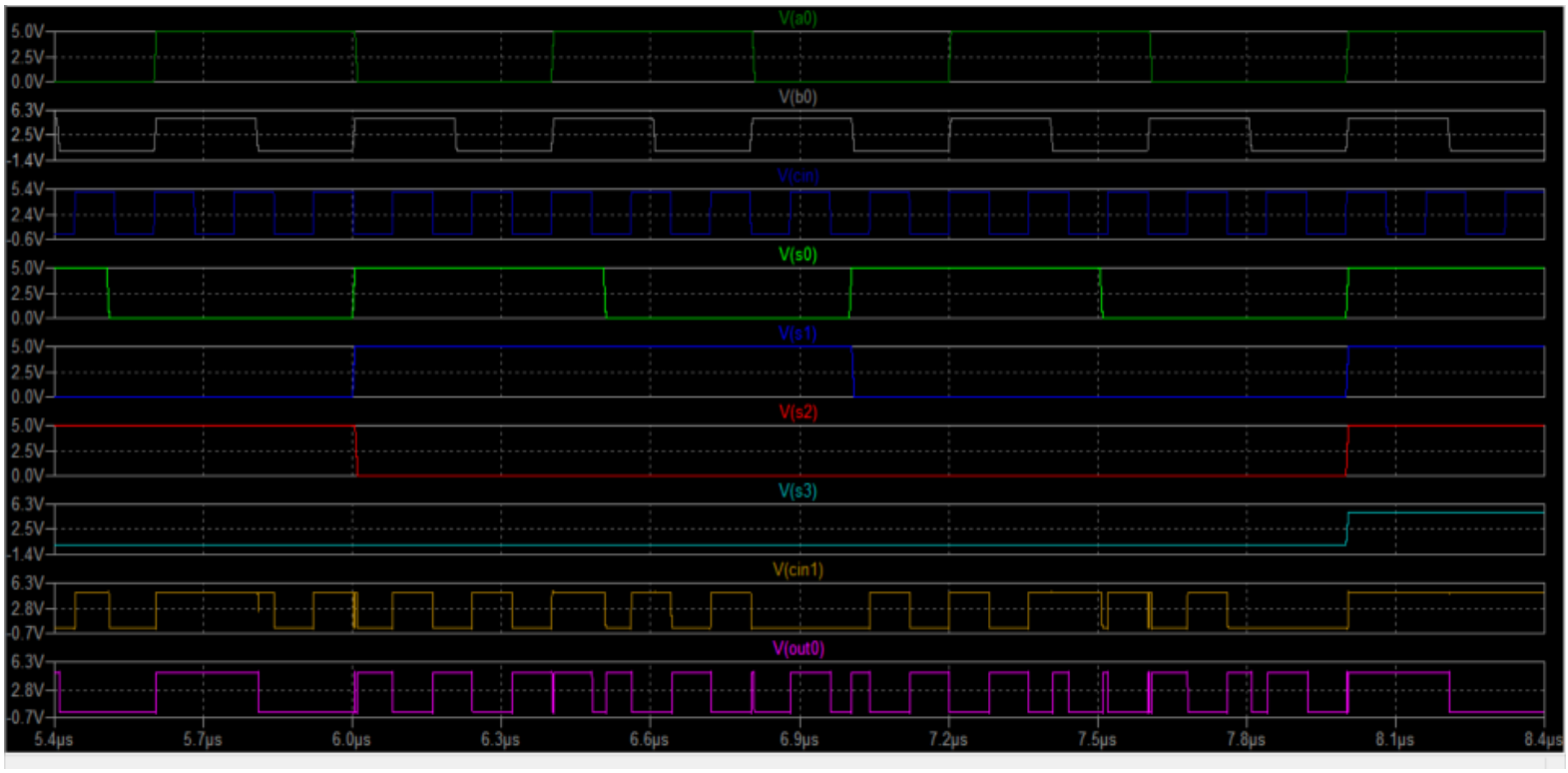

Figure 43: (s3=0, s2=0, a0, ,b0 , cin1=cout of first bit)

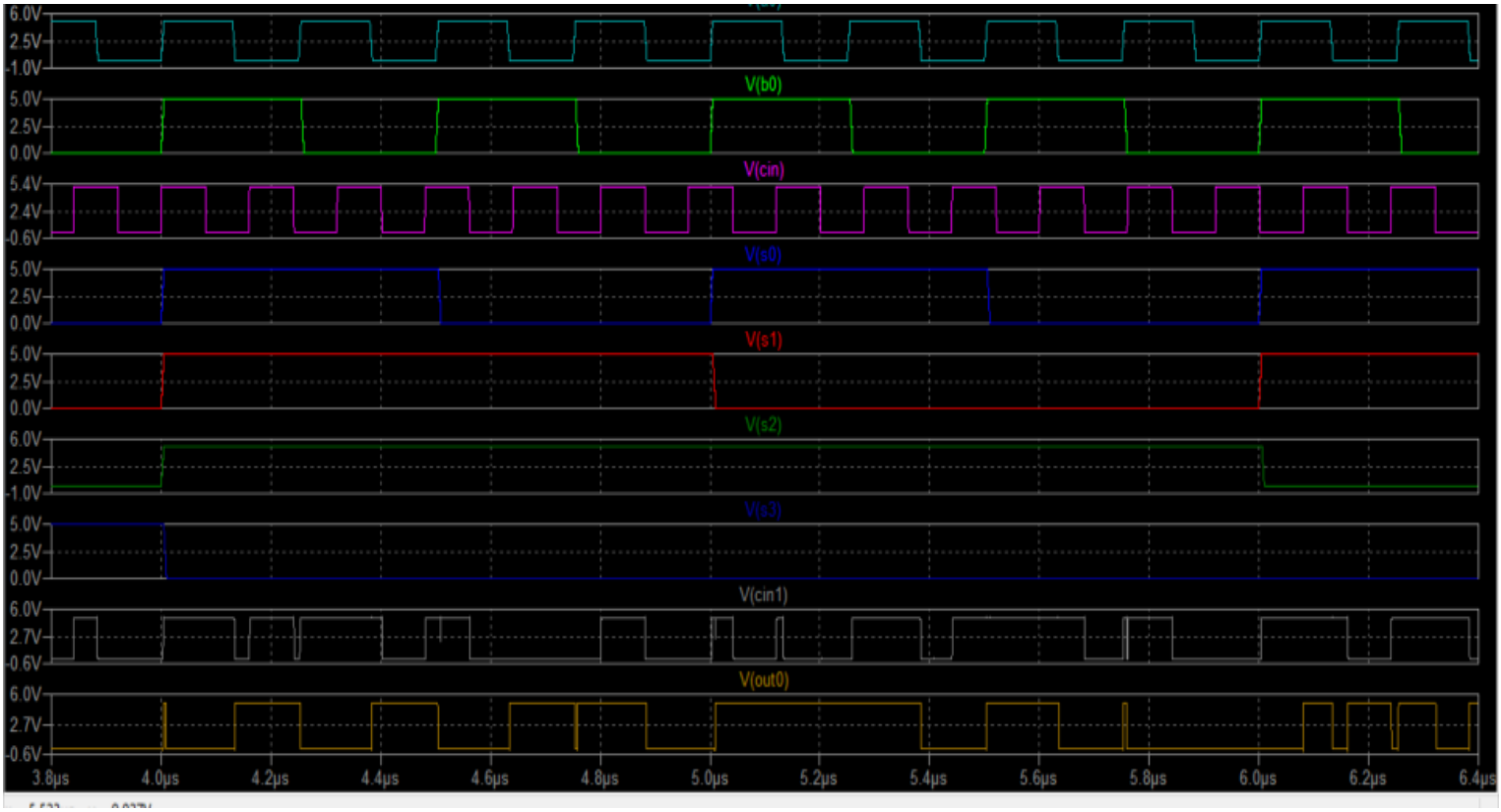

Figure 44:(s3=0, s2=1, a0, ,b0 , cin1=cout of first bit)

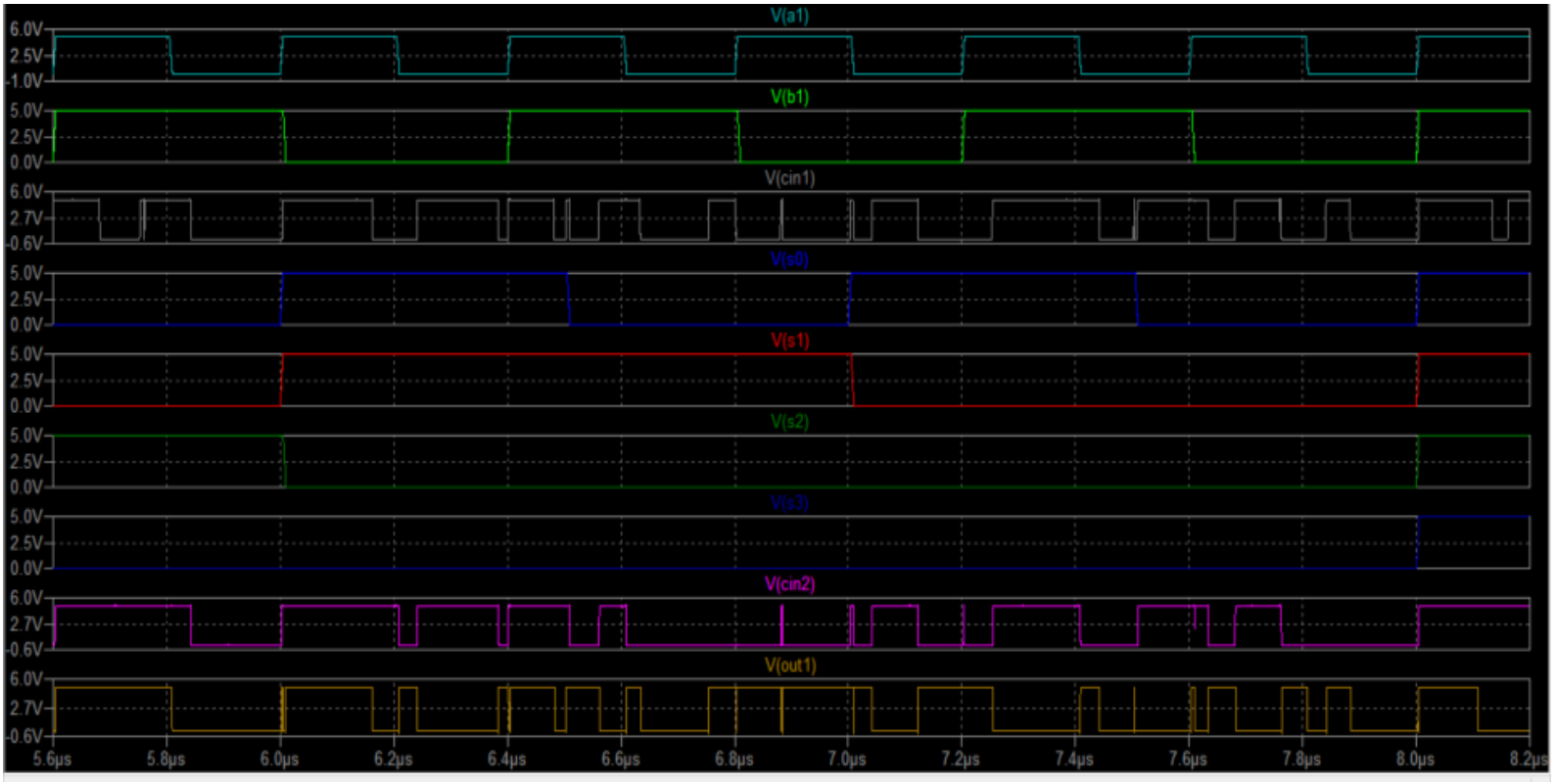

Figure 45:(s3=0, s2=0, a1, ,b1 , cin2=cout of second bit)

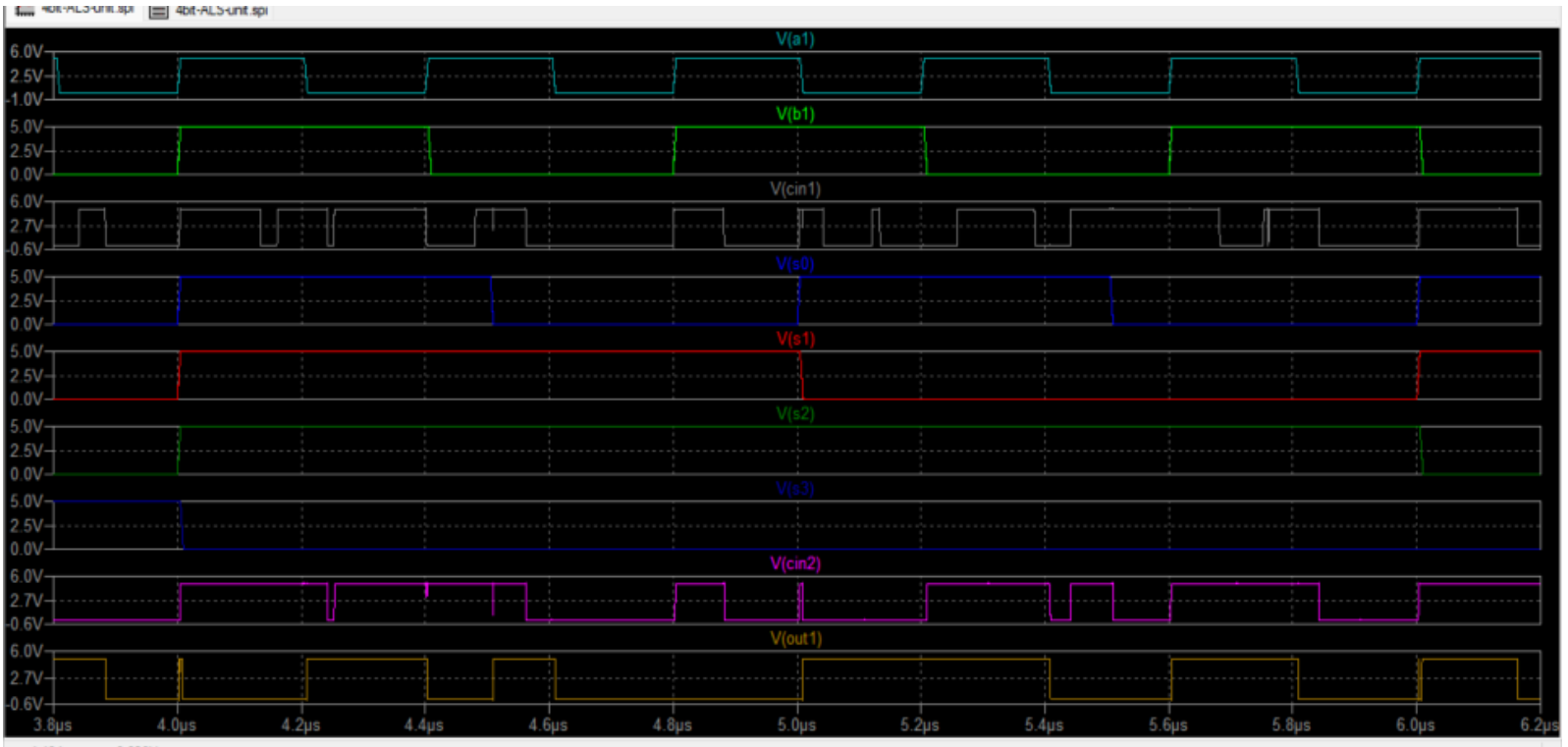

Figure 46:(s3=0, s2=1, a1 ,b1 , cin2=cout of second bit)

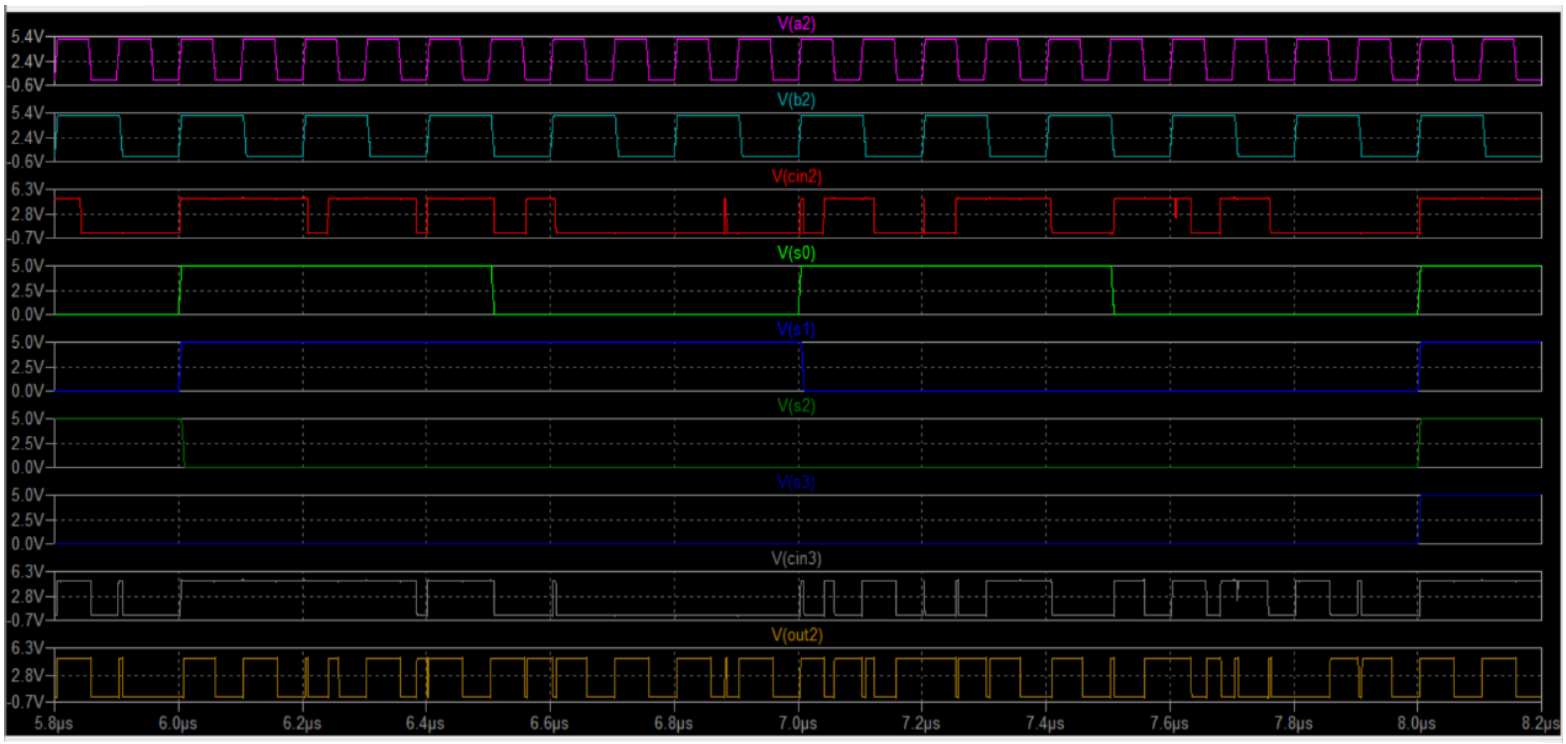

Figure 47:(s3=0, s2=0, a2, ,b2 , cin3=cout of third bit)

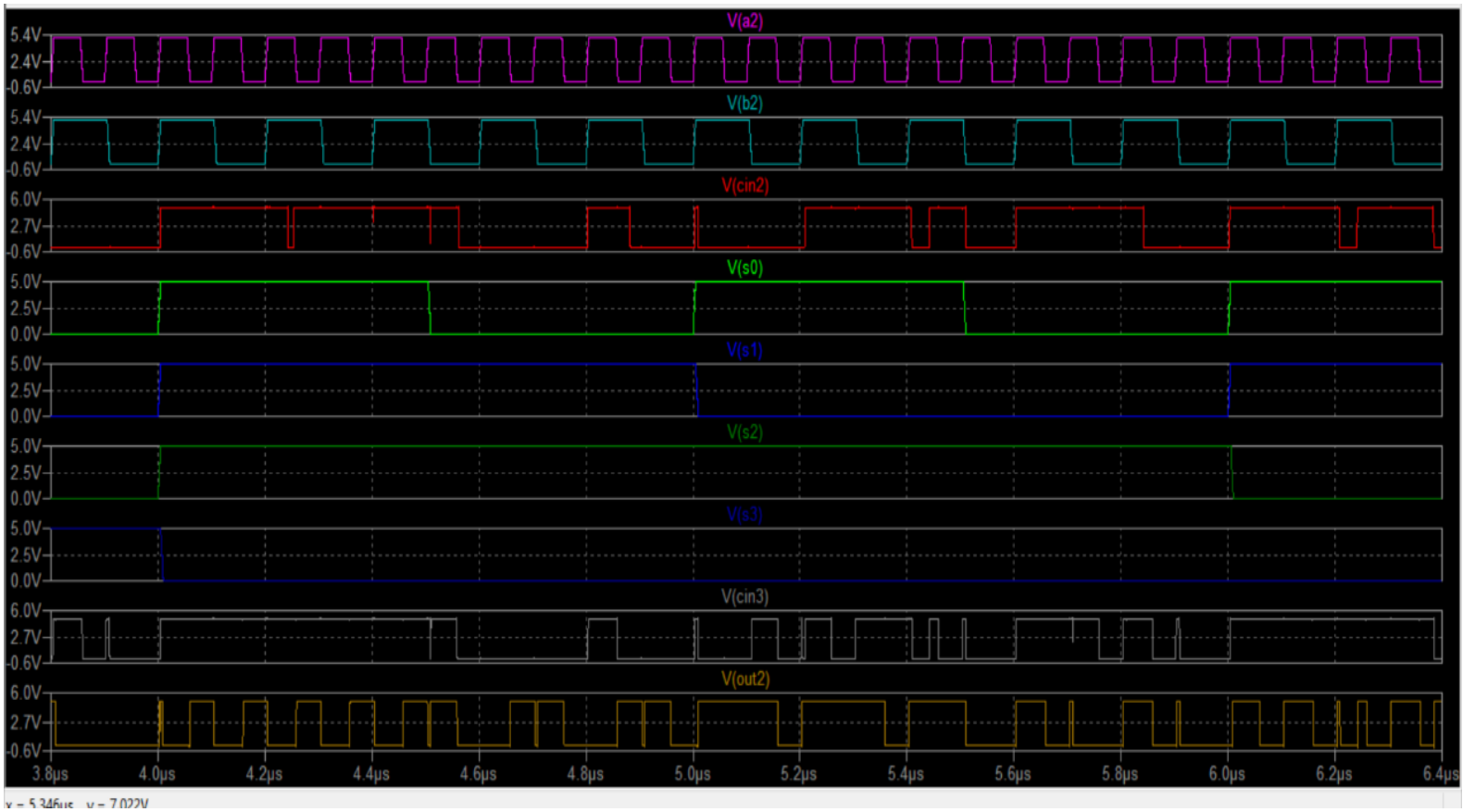

Figure 48:(s3=0, s2=1, a2, ,b2 , cin3=cout of third bit)

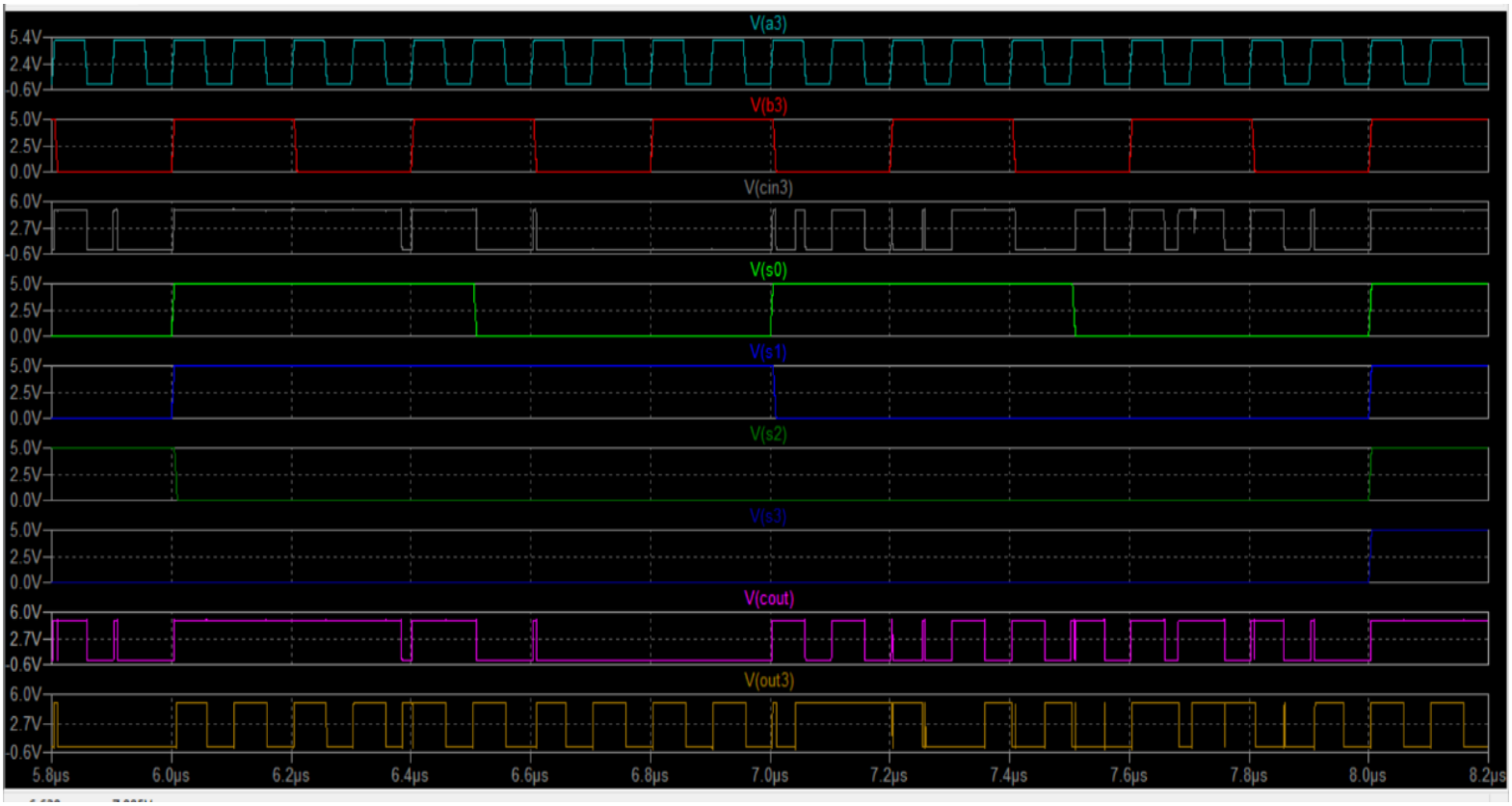

Figure 49:(s3=0, s2=0, a3, ,b3)

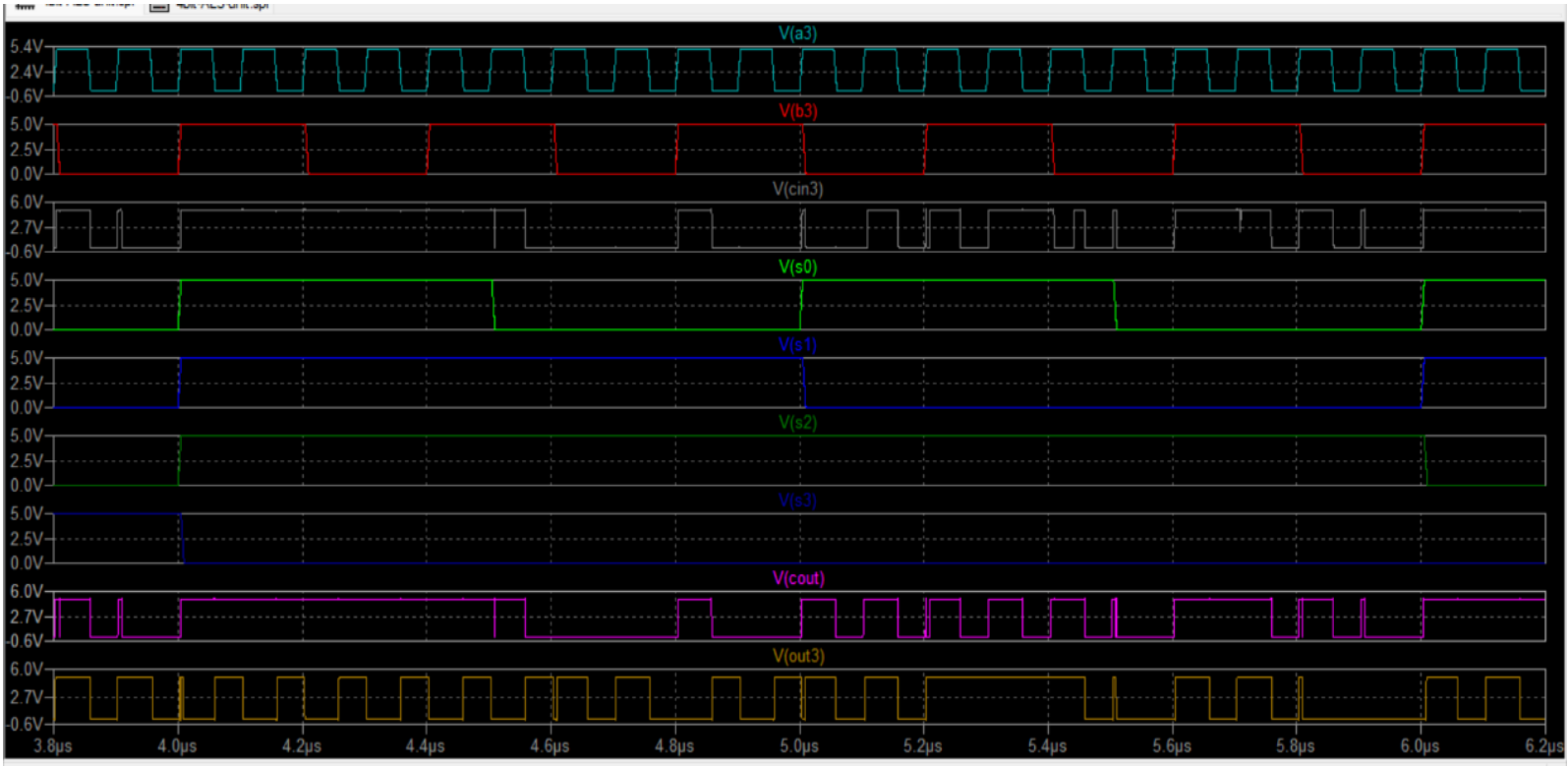

Figure 50:(s3=0, s2=1, a3, ,b3)

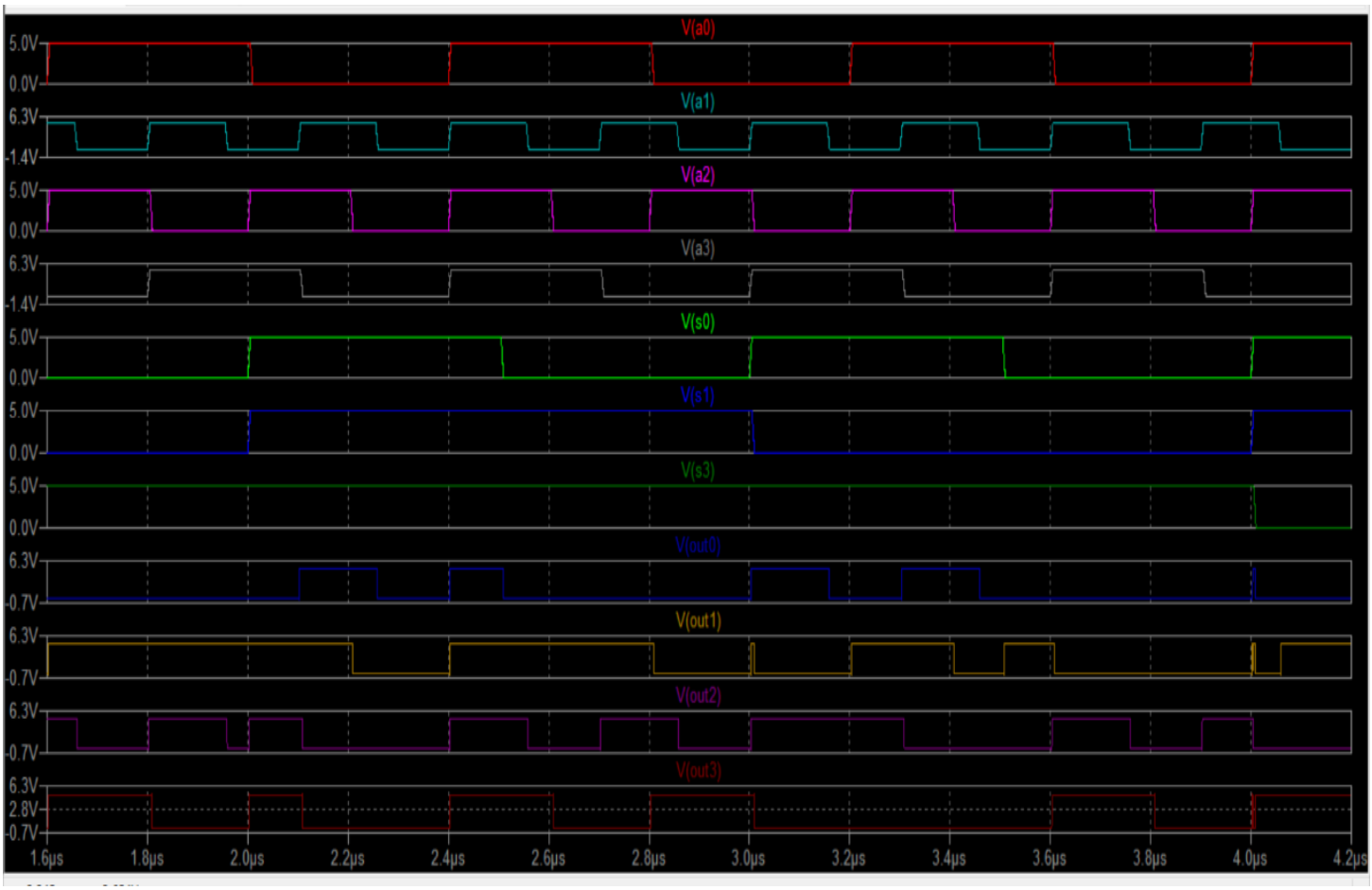

Figure 51:simulation of shifter unit

DRC: Running DRC with area bit on, extension bit on, Mosis bit

Checking again hierarchy .... (0.02 secs)

Found 41 networks

Checking cell '4bit-ALS-unit{lay}'

No errors/warnings found

0 errors and 0 warnings found (took 1.952 secs)

LVS: Hierarchical NCC every cell in the design: cell '4bit-ALS-unit{sch}' cell '4bit-ALS-unit{lay}'

Comparing: newproj:not{sch} with: newproj:not{lay}

exports match, topologies match, sizes match in 0.029 seconds.

Comparing: newproj:2mux1{sch} with: newproj:2mux1{lay}

exports match, topologies match, sizes match in 0.005 seconds.

Comparing: newproj:4mux1{sch} with: newproj:4mux1{lay}

exports match, topologies match, sizes match in 0.002 seconds.

Comparing: newproj:xor{sch} with: newproj:xor{lay}

exports match, topologies match, sizes match in 0.004 seconds.

Comparing: newproj:full-adder{sch} with: newproj:full-adder{lay}

exports match, topologies match, sizes match in 0.009 seconds.

Comparing: newproj:1bit-AU{sch} with: newproj:1bit-AU{lay}

exports match, topologies match, sizes match in 0.003 seconds.

Comparing: newproj:and{sch} with: newproj:and{lay}

exports match, topologies match, sizes match in 0.0 seconds.

Comparing: newproj:or{sch} with: newproj:or{lay}

exports match, topologies match, sizes match in 0.0 seconds.

Comparing: newproj:1bit-LU{sch} with: newproj:1bit-LU{lay}

exports match, topologies match, sizes match in 0.008 seconds.

Comparing: newproj:4bit-shifter{sch} with: newproj:4bit-shifter{lay}

exports match, topologies match, sizes match in 0.011 seconds.

Comparing: newproj:4bit-ALS-unit{sch} with: newproj:4bit-ALS-unit{lay}

exports match, topologies match, sizes match in 0.004 seconds.

Summary for all cells: exports match, topologies match, sizes match

NCC command completed in: 0.09 seconds.

6. Results and Observations

The final implementation **Results and Observations** of the 4-bit Arithmetic Logic Shift Unit (ALSU) was successfully constructed using modular design principles. The circuit was tested for a variety of input combinations to validate its ability to perform arithmetic, logic, and shift operations. The key findings and observations from the testing phase are as follows:

1. Correct Functionality of Individual Units:
   Each 1-bit Arithmetic and Logic Unit (ALU) was verified independently to ensure it correctly performs basic operations such as addition, AND, OR, and XOR. The propagation of carry between bits was correctly handled through the chained cout and cin signals.

2. ALU Control with S2:
   The first layer of multiplexers, controlled by S2, successfully selected between arithmetic and logic results. Changing S2 from 0 to 1 reliably switched the output of the ALU from arithmetic to logic operations.

3. Shift Unit Priority via S3:
   The second layer of multiplexers, controlled by S3, demonstrated higher priority in operation selection. When S3 = 1, the output of the Shifter Unit was directed to the final output, overriding the ALU outputs regardless of the setting of S2. This behavior is consistent and confirms the correct implementation of control dominance for shift operations.

4. Operation Selection via S0, S1:
   The inputs S0 and S1 effectively controlled the internal functions of the arithmetic, logic, and shift units. Specific combinations of S0 and S1 reliably selected operations such as addition, subtraction, AND, OR, logical shift left, and logical shift right.

5. Modularity and Scalability:
   The modular design using reusable 1-bit ALU blocks and separate shifter logic allowed for clear and scalable implementation. The system could be extended to 8-bit or higher by replicating the structure and adjusting the shifter accordingly.

6. No Conflicts in Output Routing:
   Thanks to proper control signal isolation, no data collision or undefined behavior was observed at the output layer. The hierarchical MUX control logic ensured that only one source (ALU or Shifter) drives the final output at any time.

7. Timing Behavior:
   The system behaved consistently in a simulated environment. Though propagation delay was not measured explicitly, visual simulation showed stable transitions with correct output for all combinations tested.

7.Conclusion

The design and implementation of the 4-bit Arithmetic Logic Shift Unit (ALSU), based on the *Mano Computer Architecture* model, successfully demonstrate a functional and modular approach to combining arithmetic, logic, and shift operations in a single unit. Through the use of hierarchical multiplexers and a clearly defined control logic, the system efficiently selects between arithmetic/logic operations and shift functions with reliable priority handling.

The use of two control layers—where S2 determines the type of ALU operation and S3 gives overriding control to the shifter—ensures flexible operation control while maintaining predictable behavior. The control inputs S0 and S1 allow precise selection of specific internal functions within the ALU and Shifter components, confirming the design's alignment with typical instruction decoding patterns.

This project highlights not only the theoretical principles taught in computer architecture but also their practical application in digital circuit design. The modular structure enables future expansion and integration into more complex systems such as CPUs or embedded controllers. Overall, the ALSU circuit functions as intended and provides a solid foundation for advanced exploration in digital system design and VLSI implementation.