**NEW HORIZON COLLEGE OF ENGINEERING**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING.**

**DIGITAL LOGIC DESIGN REPORT (22CSL32)**

**GROUP ASSIGNMENT**

**THIRD SEMESTER YEAR: 2024-2025**

NAME: **Ashmitha Suma Raj, Sahana H, Anvesha**

USN: **1NH24CS402-T, 1NH24CS403-T, 1NH23CS126**

SEMESTER: **3rd sem**

SECTION: **C**

SUBMITTED TO: **Mr. BHASKAR S V**

DATE:

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**FACULTY SIGNATURE**

AT1: PRACTICAL ORIENTATION ON DESIGN THINKING

TOPIC: To construct a ROM using circuits.

**Requirements: A Digital Logic Simulator.**

**PROBLEM STATEMENT:**

To design ROM (Read-Only Memory) using simple logic gates.

The ROM can also be designed using Decoders or Adders.

**REQUIREMENTS:**

1. **Memory:** The ROM must be capable of storing **16 words** each of which is **8 bits long.**
2. **Addressing:** The ROM should utilize a **4-bit address** to access each stored word. This implies there will be 16 unique addresses: 0000 to 1111.
3. **Components:** The ROM should include a **4-16 line decoder** to decode the address line. Each memory location should store **8bits** of data.

**SOLUTION TO OUR PROBLEM:**

We have implemented our Circuit using LOGISIM software. To implement the circuit we have used ROM Memory unit with clock signals, select lines, output unit and clear signal with input and output units.

**SIMULATOR USED:**

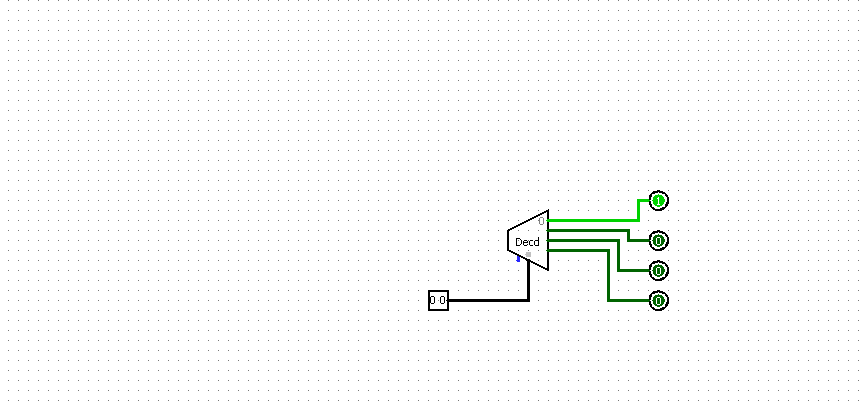
The simulator used to design our circuit is LOGISIM circuit design simulator. It is an easy to use simulator where you can design your circuit and implement and know how your circuit will run or simulate according to the input provided.

There are different components available on this software which enables you to design and implement a circuit digitally and verify it. There are some key features:

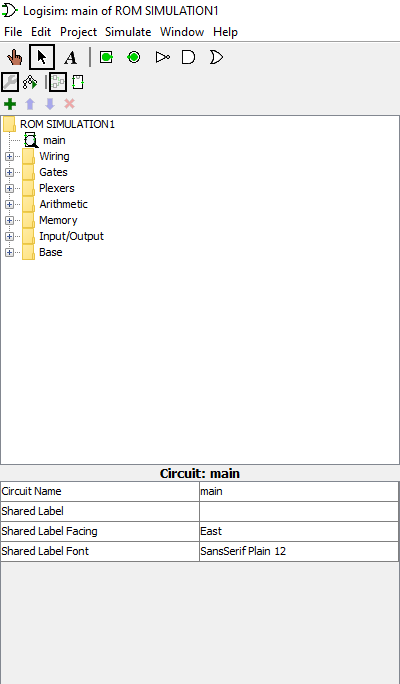
1. Graphical Interface
2. Hierarchical Design
3. Simulation
4. Component Library
5. Wire-Bundle
6. Cross-Platform

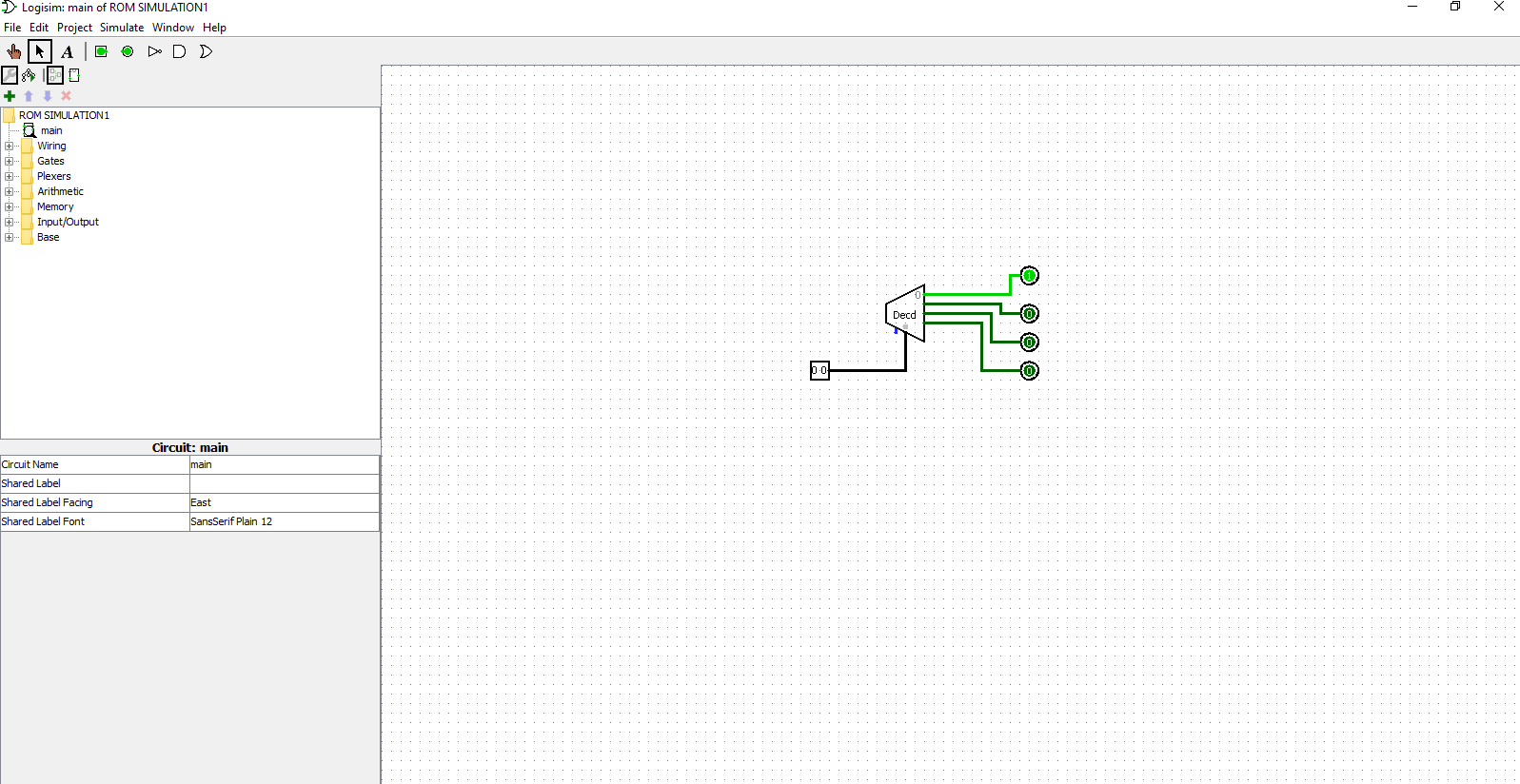
Here is a simple outlook of the software:

Sketch /Drawing area:



Objective Panel:



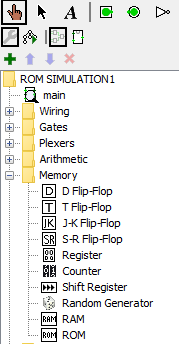


Using LOGISIM software you can implement simple basic gates, CPU or Memory units, multiplexers, de-multiplexer, decoders, encoders, arithmetic and logic gates and much more.

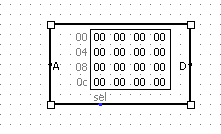
You can also simulate the circuits you have implemented and check how they will work and outputs generated according to your inputs.

**DESIGNING OUR CIRCUIT:**

We used the ROM circuit that was pre-installed in our system under the **MEMORY UNIT** available in the circuit panel.



**ROM:**

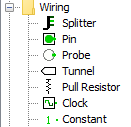


Than we fixed the input and output bits to 8 units pin or probe.



Probe(input/output)

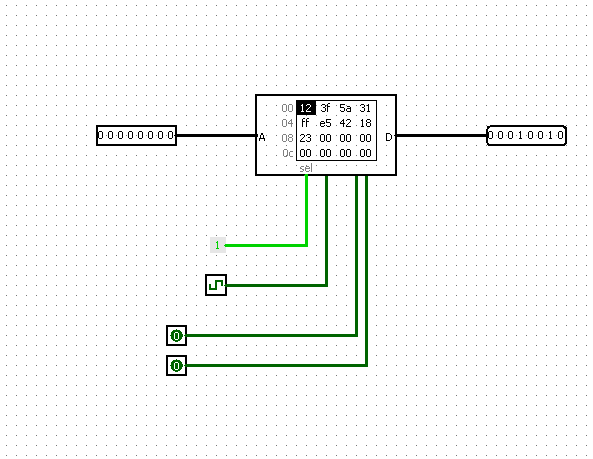
Later we implemented 1 select line and clock signal from the **WIRING UNIT.**

****

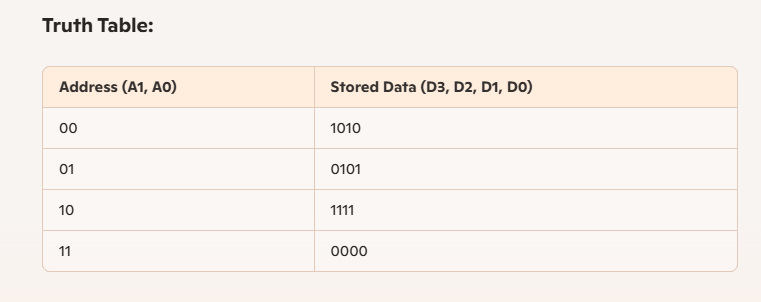
Then we used two probes to implement the output and clear units for easy simulation.

There by connecting all these components together we constructed a ROM circuit which takes 8 bit input thereby regulating its output through differing clock and inputs provided by the user.

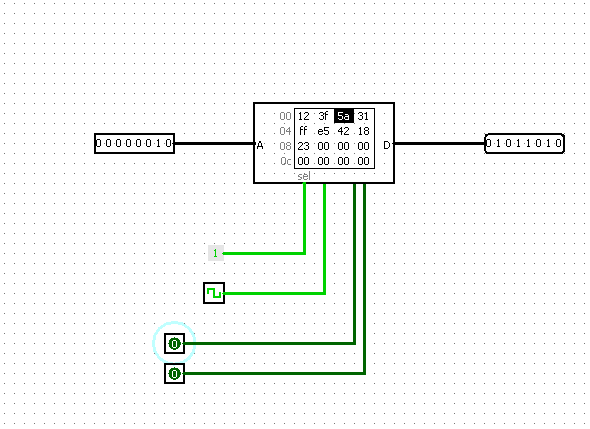
**Complete Circuit diagram:**

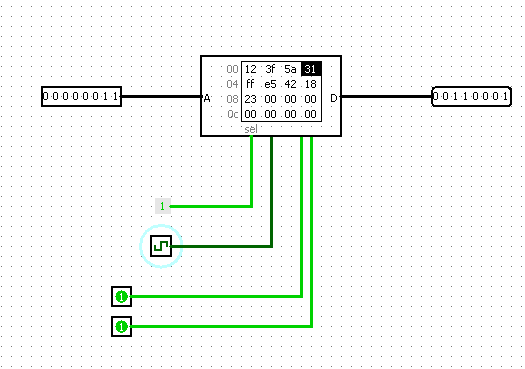


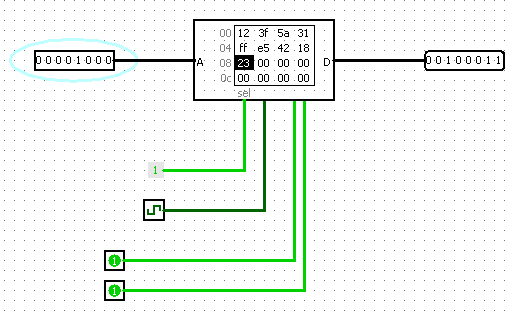
**Truth Table Of ROM Circuit:**



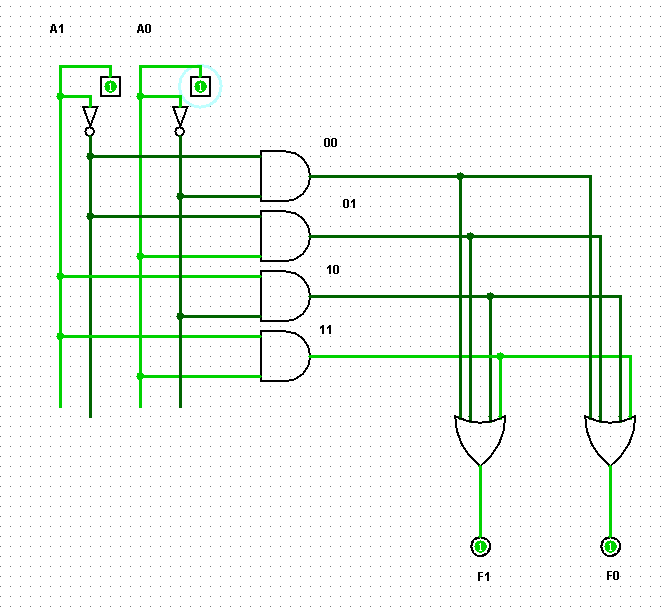
**SIMULATION PICS:**

****

****

****

**USING SIMPLE CIRCUIT:**

****

**RESULT:**

Hereby, we have implemented ROM Circuit Using LOGISIM Simulator successfully and verified its simulation through different inputs provided.

**AT2-Software Simulation of Digital Design**

**TOPIC: Implement the real time problem designed for AT-1 using Verilog Code.**

**Verilog Code for ROM Circuit:**

module rom\_2x4 (

input [1:0] addr, // 2-bit address input

output reg [3:0] data\_out // 4-bit data output

);

// Combinational logic for ROM

always @(\*) begin

case (addr)

2'b00: data\_out = 4'b1010; // Data for address 00

2'b01: data\_out = 4'b1100; // Data for address 01

2'b10: data\_out = 4'b1111; // Data for address 10

2'b11: data\_out = 4'b1001; // Data for address 11

default: data\_out = 4'b0000; // Default case

endcase

end

endmodule

**OUTPUT:** Compiled with no Syntax Error.

**Test Bench for the above Verilog Code:**

module tb\_rom\_2x4;

reg [1:0] addr; // Address input (test cases)

wire [3:0] data\_out; // Output from ROM

// Instantiate the ROM module

rom\_2x4 uut (

.addr(addr),

.data\_out(data\_out)

);

initial begin

// Test all possible addresses

$display("Address | Data Output");

addr = 2'b00; #10; $display(" %b | %b", addr, data\_out);

addr = 2'b01; #10; $display(" %b | %b", addr, data\_out);

addr = 2'b10; #10; $display(" %b | %b", addr, data\_out);

addr = 2'b11; #10; $display(" %b | %b", addr, data\_out);

$finish; // End simulation

end

endmodule

**Output:** Executed with No Syntax Error.

**WAVEFORM:**

****

**Result:** Verilog Code Executed Successfully.