4-bit Magnitude Comparator

(using HDL language)

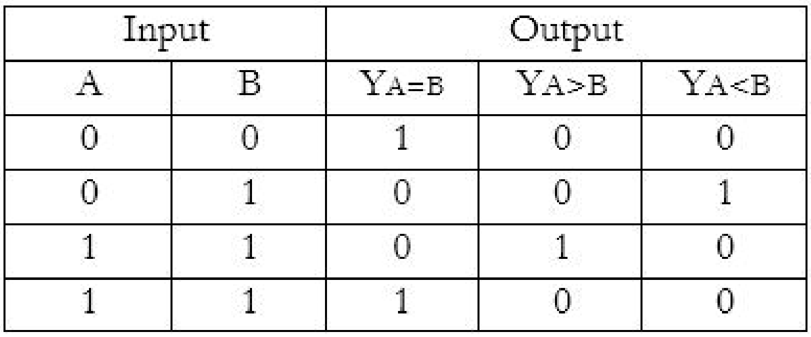
1. Abstract

The design uses Verilog code in behavioral mode, also known as RTL (Register Transfer Level) coding, to create the functional block of the comparator. This Verilog code is synthesized using the Makerchip platform and the NgVeri tool, which convert the HDL (Hardware Description Language) design into a corresponding block structure.

2. Theory

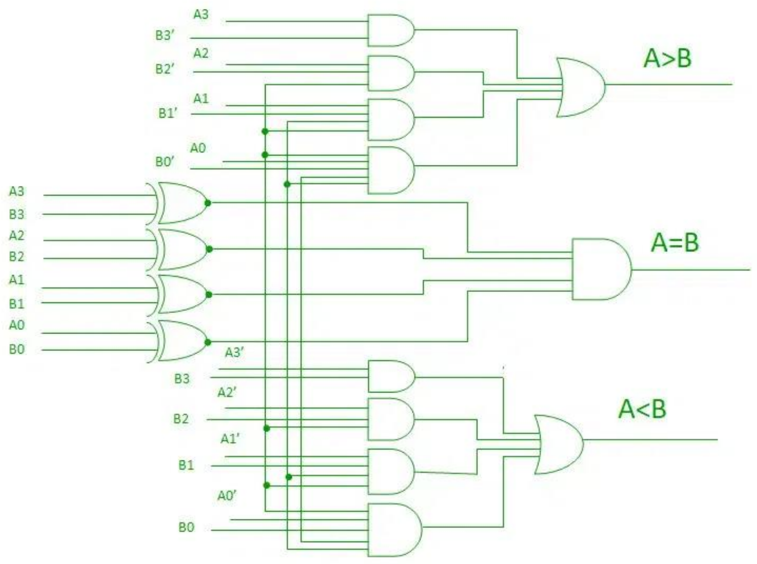
A comparator is a circuit which compares the two values. A Magnitude Comparator is a combinational circuit which compares the magnitude of two binary numbers. It decides whether a number is greater than, less than or equal to, by comparing the magnitudes of both the numbers. It is mostly used inside arithmetic operations in ALU (Arithmetic and Logic unit) which is situated inside CPU (Central Processing Unit). If two n-bit number must be compared, then the given circuit has 22n total conditions. For example – if there are 2 bit number , there will be 4 input and 16 rows in the truth table , similarly In this given circuit , if the number of bit in each number is 4 then , it is 4 bit magnitude comparator which means that it has 8 input and total of 3 output that is greater than(A>B), less than(A<B) and equal to(A=B) if A and B be two binary numbers. Let suppose that there are two number A and B, and each have 4-bit A0, A1, A2, A3 and B0, B1, B2, B3. Now, 4-bit magnitude comparator compares the magnitude of both number and provide the output accordingly that whether (A>B) or (A<B) or (A=B).

3. Truth Table (ex. 2bit)



Input		Output		
A	B	$Y_{A=B}$	$Y_{A>B}$	$Y_{A<B}$
0	0	1	0	0
0	1	0	0	1
1	1	0	1	0
1	1	1	0	0

4. Circuit

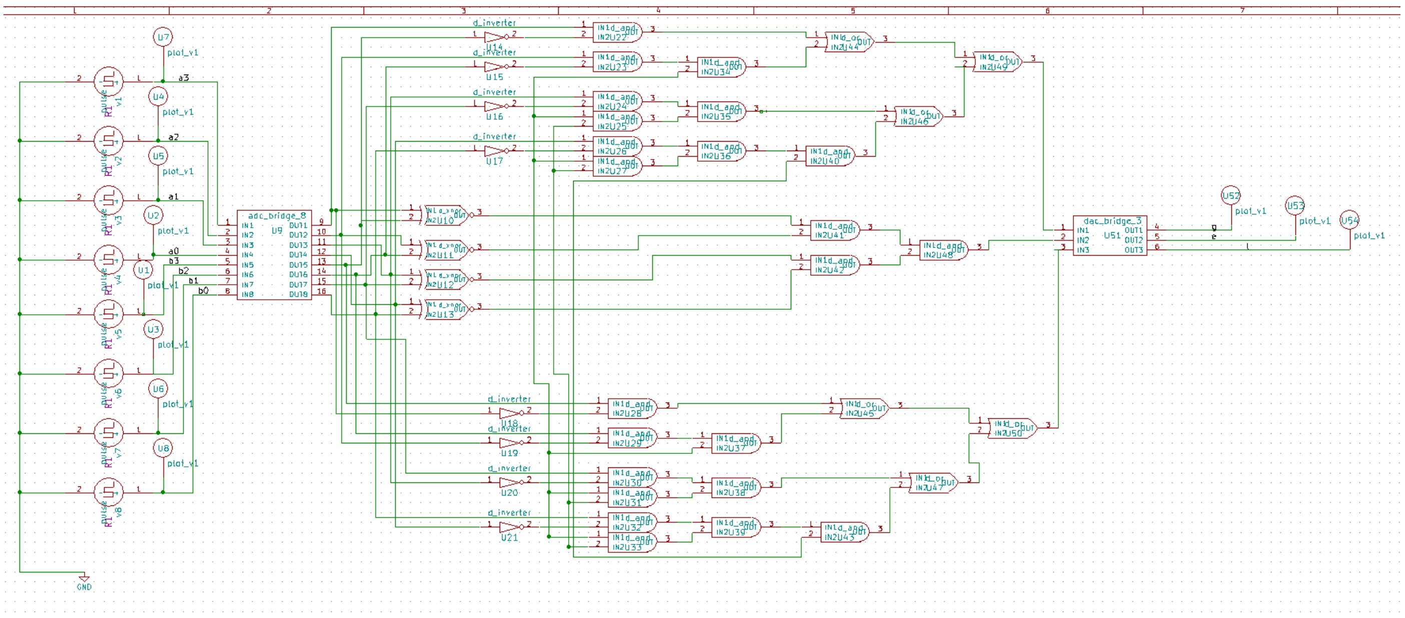


5. Verilog code for Comparator:

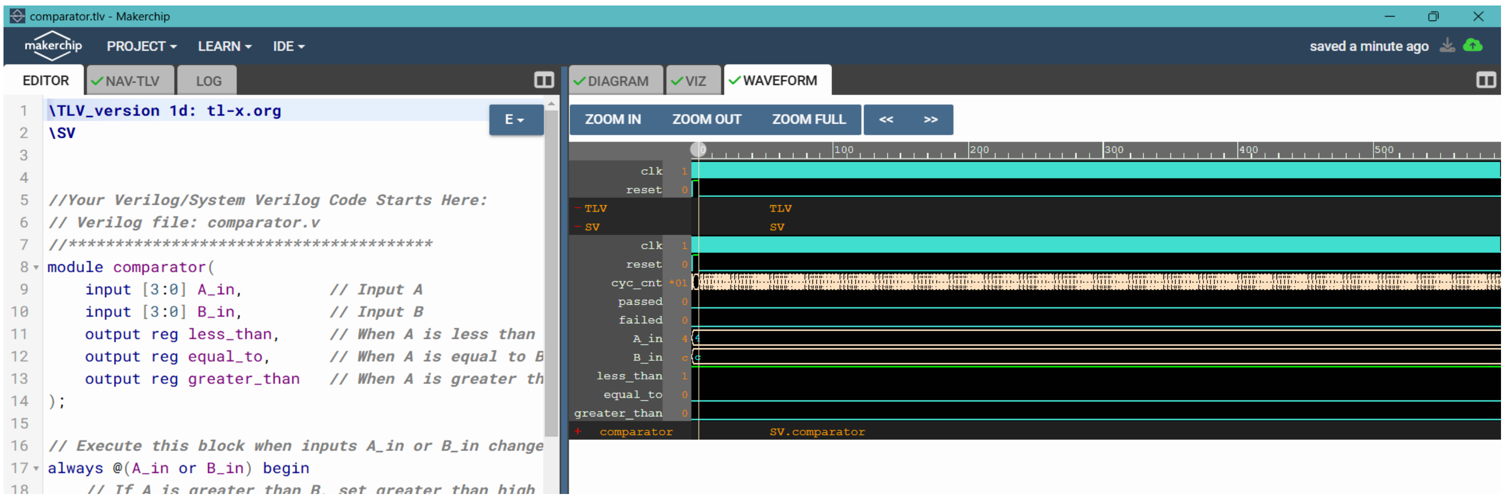
```
\TLV_version 1d: tl-x.org

\SV

//Your Verilog/System Verilog Code Starts Here:

// Verilog file: comparator.v

//**************************************

module comparator(

 input [3:0] a_in,        // Input A

 input [3:0] b_in,        // Input B

 output reg l,     // When A is less than B, it is high

 output reg e,      // When A is equal to B, it is high

 output reg g;   // When A is greater than B, it is high

// Execute this block when inputs A_in or B_in change

always @(a_in or b_in) begin

// If A is greater than B, set g high

if (a_in > b_in) begin

 l = 1'b0;

e = 1'b0;

g = 1'b1;

 end

// If A is equal to B, set e high
```

```
 else if (a_in == b_in) begin

  l = 1'b0;

 e= 1'b1;

 g= 1'b0;

 end

 // Otherwise, set l high (A < B)

 else begin

 l= 1'b1;

 e= 1'b0;

g = 1'b0;

end

end

endmodule

//Top Module Code Starts here:

                    module top(input logic clk, input logic reset, input logic [31:0] cyc_cnt, output logic passed, output logic failed);

logic  [3:0] a_in;//input

logic  [3:0] b_in;//input

logic  l;//output

logic  e;//output

logic  g;//output

//The $random() can be replaced if user wants to assign values

assign A_in = $random();

assign B_in = $random();

comparator comparator(.a_in(a_in), .b_in(b_in), .l(l), .e(e), .g(g));


\TLV

//Add \TLV here if desired

\SV

endmodule
```

6. Schematic Diagram

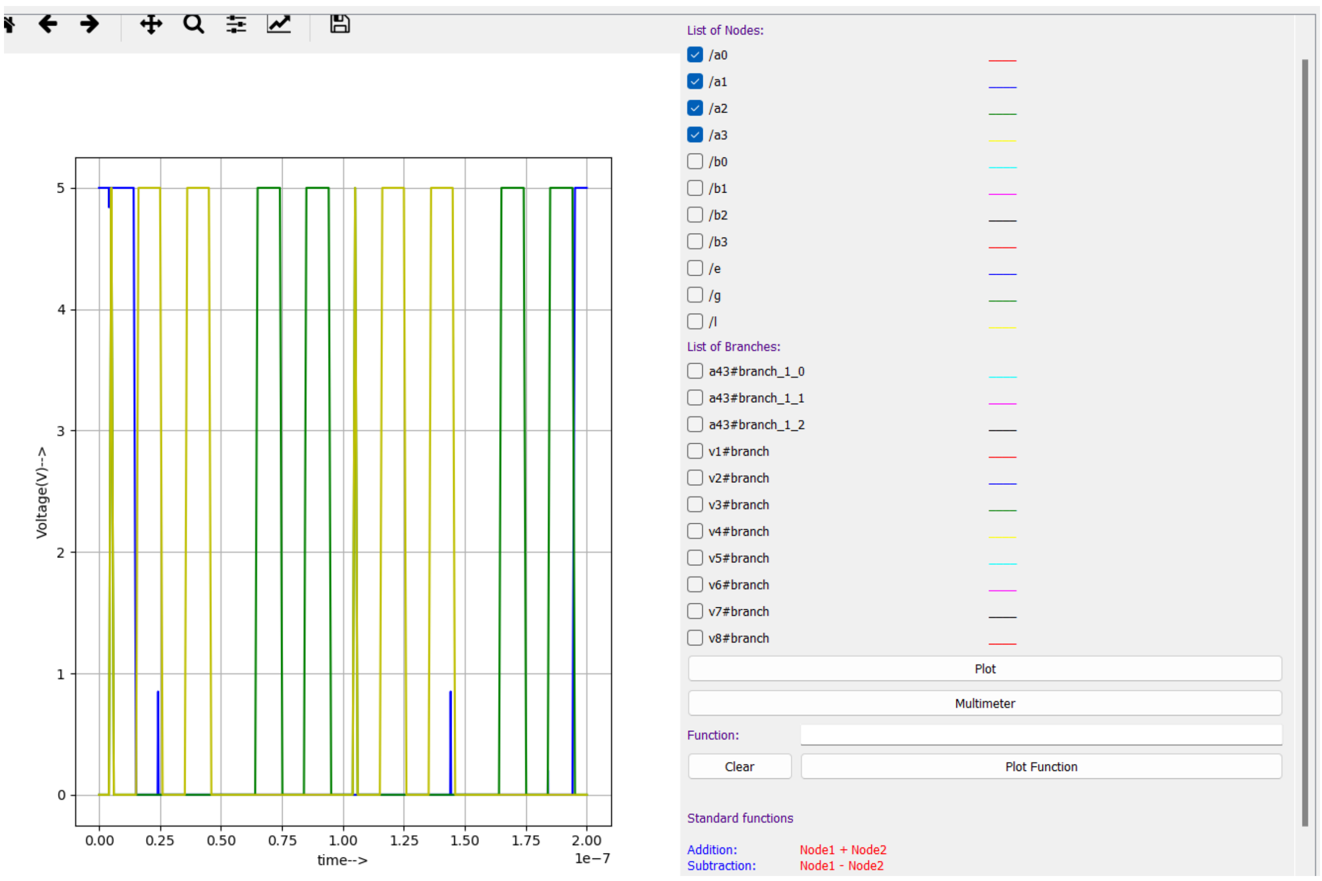


7.Makerchip plot:



```
\TLV_version 1d: tl-x.org
\SV


//Your Verilog/System Verilog Code Starts Here:
// Verilog file: comparator.v
//***************************************
module comparator(
    input [3:0] A_in,          // Input A
    input [3:0] B_in,          // Input B
    output reg less_than,      // When A is less than
    output reg equal_to,       // When A is equal to B
    output reg greater_than    // When A is greater th
);

// Execute this block when inputs A_in or B_in change
always @(A_in or B_in) begin
    // If A is greater than B, set greater_than high
```

6.Output Waveforms



List of Nodes:

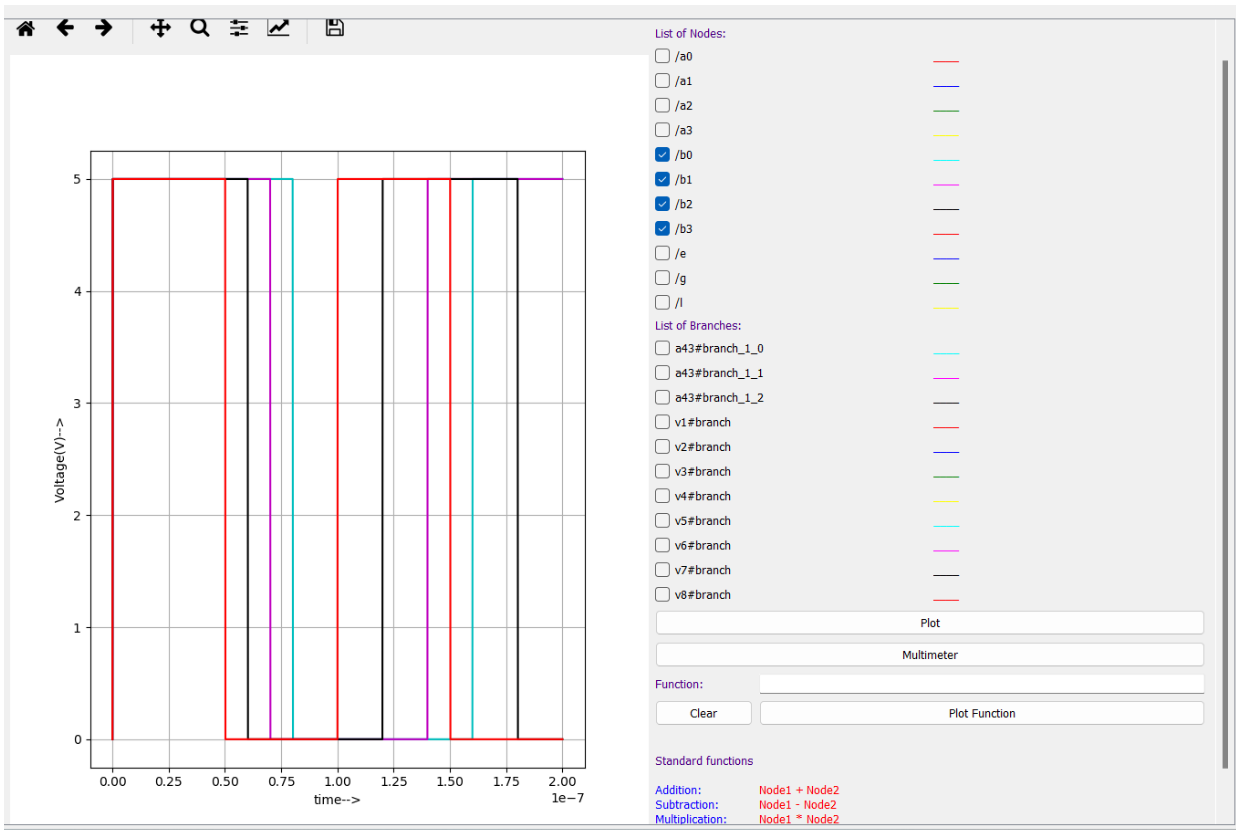
- ✓ /a0
- ✓ /a1
- ✓ /a2
- ✓ /a3
- ☐ /b0
- ☐ /b1
- ☐ /b2
- ☐ /b3
- ☐ /e
- ☐ /g
- ☐ /l

List of Branches:

- ☐ a43#branch_1_0
- ☐ a43#branch_1_1
- ☐ a43#branch_1_2
- ☐ v1#branch
- ☐ v2#branch
- ☐ v3#branch
- ☐ v4#branch
- ☐ v5#branch
- ☐ v6#branch
- ☐ v7#branch
- ☐ v8#branch

Addition: Node1 + Node2
Subtraction: Node1 - Node2

A3, A2, A1, A0



List of Nodes:

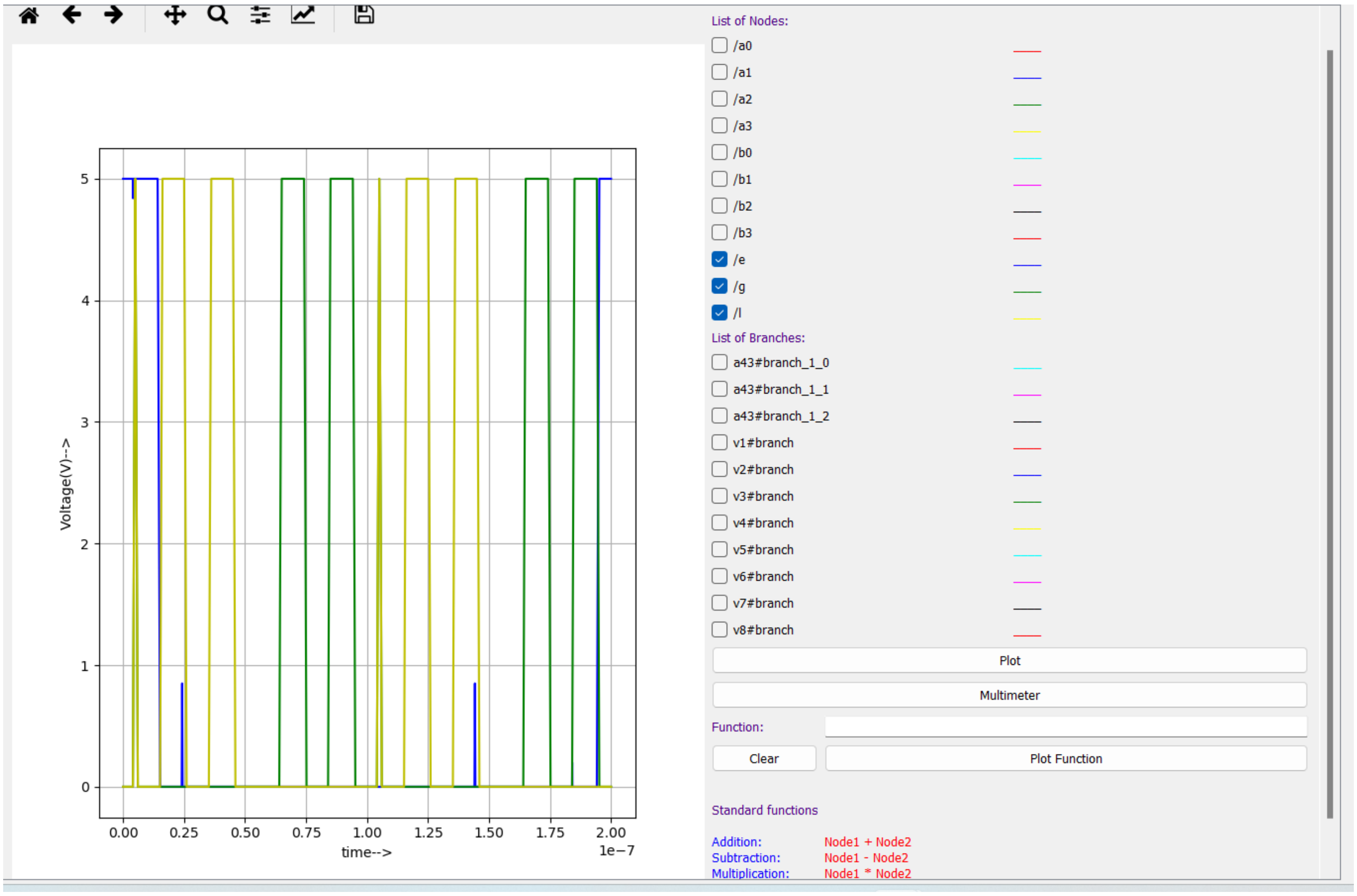
- ☐ /a0
- ☐ /a1
- ☐ /a2
- ☐ /a3
- ✓ /b0
- ✓ /b1
- ✓ /b2
- ✓ /b3
- ☐ /e
- ☐ /g
- ☐ /l

List of Branches:

- ☐ a43#branch_1_0
- ☐ a43#branch_1_1
- ☐ a43#branch_1_2
- ☐ v1#branch
- ☐ v2#branch
- ☐ v3#branch
- ☐ v4#branch
- ☐ v5#branch
- ☐ v6#branch
- ☐ v7#branch
- ☐ v8#branch

Addition: Node1 + Node2
Subtraction: Node1 - Node2
Multiplication: Node1 * Node2

B3, B2, B1, B0



List of Nodes:

- [ ] /a0
- [ ] /a1
- [ ] /a2
- [ ] /a3
- [ ] /b0
- [ ] /b1
- [ ] /b2
- [ ] /b3
- [✓] /e
- [✓] /g
- [✓] /l

List of Branches:

- [ ] a43#branch_1_0
- [ ] a43#branch_1_1
- [ ] a43#branch_1_2
- [ ] v1#branch
- [ ] v2#branch
- [ ] v3#branch
- [ ] v4#branch
- [ ] v5#branch
- [ ] v6#branch
- [ ] v7#branch
- [ ] v8#branch

Plot

Multimeter

Function:

Clear | Plot Function

Standard functions

Addition:	Node1 + Node2
Subtraction:	Node1 - Node2
Multiplication:	Node1 * Node2

(A>B), (A=B), (A<B)

References

[1] https://www.geeksforgeeks.org/magnitude-comparator-in-digital-logic/

[2] https://www.researchgate.net/publication/275726067_Performance_Analysis_of_Magnitude_Comparator_using_Different_Design_Techniques

[3] https://eevibes.com/digital-logic-design/how-to-design-a-4-bit-magnitude-comparator-circuit/

Project Submitted by:

Shanthi Priya K
Rajiv Gandhi University of Knowledge and Technologies,
IIIT Nuzvid,
Eluru ,521202