

**RAJIV GANDHI UNIVERSITY OF KNOWLEDGE
TECHNOLOGIES**



DESIGN OF ALU USING M-GDI TECHNIQUE WITH 2T XOR GATE AND DECODER

GUIDED BY:

Dr. H. Srinivasa Varaprasad

Asst. prof, ECE Dept

Batch No-16:

1. M. Sirisha (S191059)
2. B. Krishna Veni (S191112)
3. B. Akhila (S190419)
4. T. Ramya (S190078)
5. G. Sai Ram (S190986)
6. SK. Basheer Begum (S190198)

Tables of The Content:

- Abstract
- Objectives
- Introduction
- Literature Review
- Block Diagram of ALU
- Problem Statement
- Software Tools
- GDI and m-GDI and 2T-XOR
- Key Components of Proposed ALU
- Differences Between GDI, CMOS and m-GDI
- Conclusion and Future Scope
- Expected Results
- References

Abstract

Over the last few decades a lot of work has been taken to do in conventional CMOS based circuit and more impact power efficient. Various techniques, such as Transmission Logic Gate, Domino Logic, Pass transistor Logic, Double pass transistor Logic and others have been implemented in order to improve the performance of CMOS based circuits. The Gate Diffusion Input (GDI) technique is a novel approach for drastically reducing power requirements . Furthermore, GDI reduces the number of transistors, resulting in a smaller chip size giving the designs an advantage over traditional methods.

On the basis of GDI techniques, there are two types of transistors: PMOS and NMOS. Each of these has four subtypes: N,D,P,G .G,N,P serves as inputs. however GDI technique have a big issue with gate diffusion input. Since NMOS generates Logic 1 and PMOS generates Logic 0.The existing GDI Technique is the major disadvantage.

The aim of the project to designing an ALU that uses a 2-transistor XOR gate and a Modified Gate Diffusion Input (m-GDI) technique more compact and power-efficient design with modern technology. If the design an adder using GDI technique it requires 28 transistors. But in m-GDI the adder requires only 8 transistors. This is the major advantage of the project.

Compare to GDI Technique m-GDI Technique requires less number of transistors. When the count of transistors reduces automatically the chip size and power consumption will be reduced.

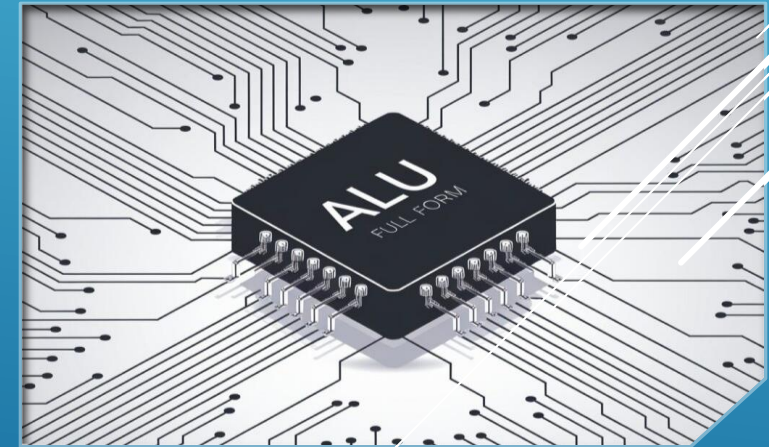
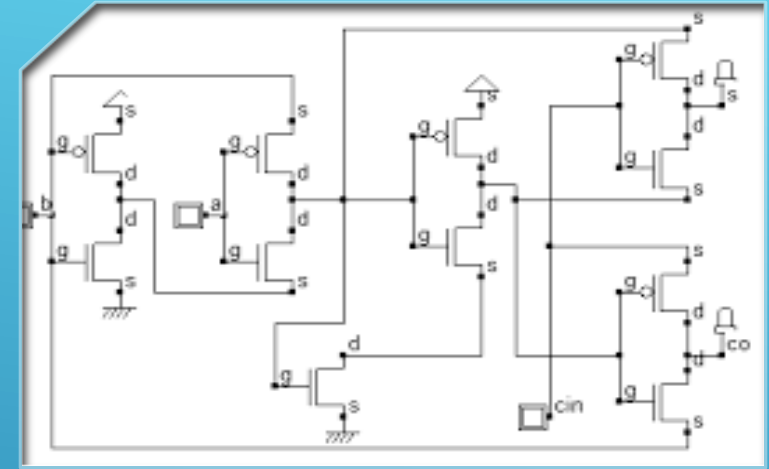
Objectives

- ❖ Our project consists of Four objectives:
- ❖ Arithmetic unit design using Modified Gate Diffusion Input and 2T-XOR
- ❖ Logic Unit design Using Modified Gate Diffusion Input(m-GDI) and 2T-XOR
- ❖ Compare m-GDI technique and GDI technique
- ❖ Code and Simulation

Introduction

The Arithmetic Logic Unit's main component is the Central Processing Unit (CPU). Arithmetic operations such as Addition, Subtraction, Multiplication, and Division are included in the CPU. AND gate, OR gate, NAND gate, XOR gate, multiplexer. All of these operations are examples of logical operations. ALU operations necessitate a much larger number of capable computers, as well as application-specific circuits and VLSI chips.

The main function of designing larger are Low-power implementation and compact implementation Dissipation and these circuits are difficult to understand and they have no exception in ALU. Over the last few decades a lot of work has been taken to do in conventional CMOS based circuit and more impact power efficient.



Problem Statement:

Design an Arithmetic Logic Unit (ALU) leveraging a 2-Transistor (2T) XOR gate and a decoder to achieve optimized performance, reduced power consumption, and minimal transistor count, addressing the limitations of conventional ALU designs in resource-constrained environments.

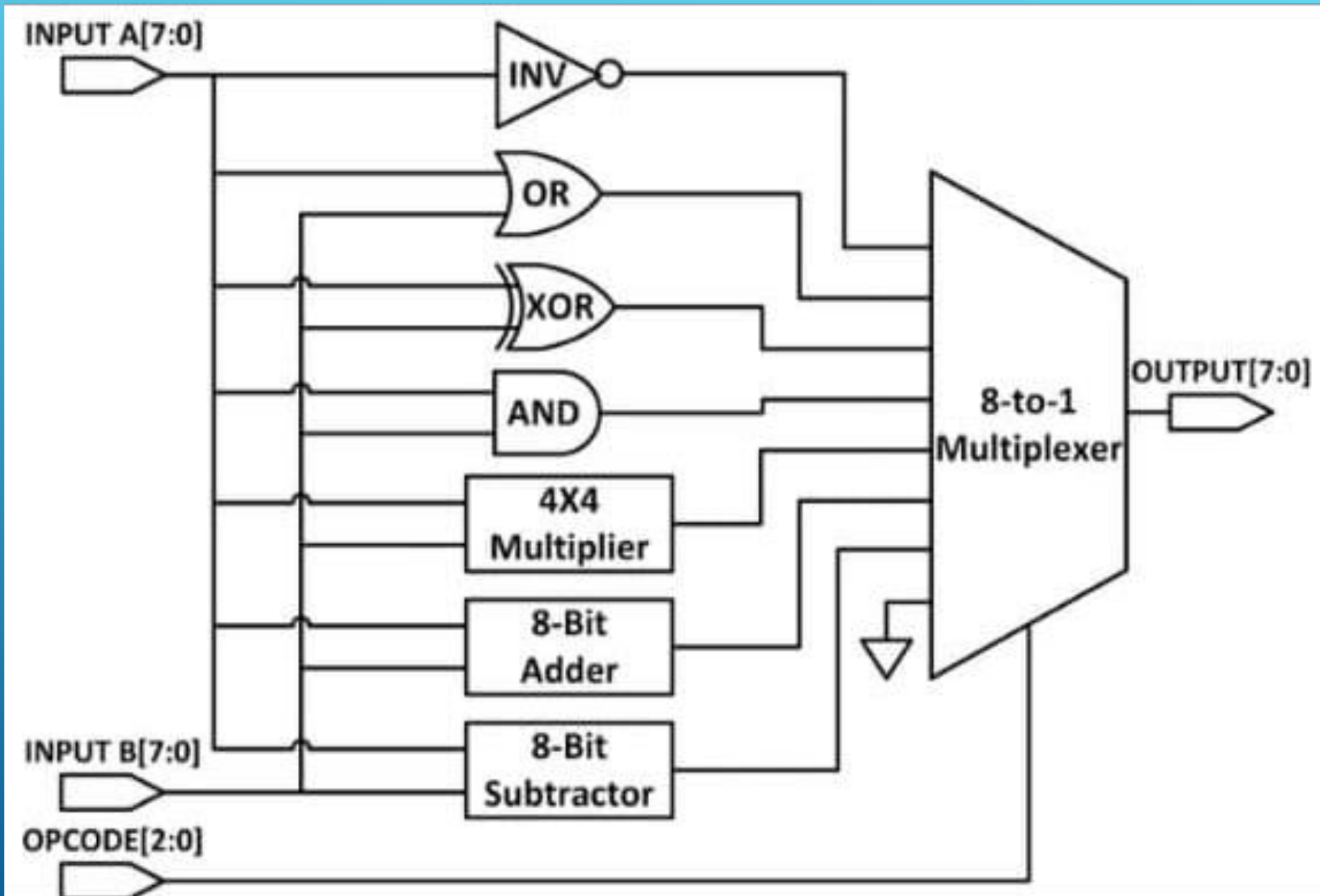
Software Tool:

Xilinx Vivado Software

Literature Review

S.NO	TITLE	JOURNAL	AUTHORS	OUTCOMES
1.	Modified Gate Diffusion Input Technique: A New Technique for enhancing performance in Full Adder circuits	IEEE(2012)	Uma, R., & Dhavachelvan, P.	Pseudo-NMOS is simple and fast but reduces noise margins and increases power consumption.
2.	9T and 8T Full Subtractor Design using Modified GDI and 3T XOR Technique	IEEE(2019)	Sarkar, S.. Sarkar. S.. Atta, A.. Pahari. T.. Majumdar. N.. & Mondal, S.	Full Subtractor is implemented by using 3T XOR gate in GDI(Gate Diffusion Input)
3.	8-Bit Arithmetic Logic Unit Design Using Modified Gate Diffusion Input(m-GDI) Technique	IJAEM(2021)	Anil Nageswar Rangapure, Dr. Kiran v	ALU design using Modified Gate Diffusion Technique
4.	Design of ALU Using 2T XOR Gate and Decoder	IJCI(2021)	Aiyyappusurendrababu, N. Ashok Kumar	ALU design using Modified Gate Diffusion Technique with less chip count

Block Diagram Of ALU



what Is GDI?

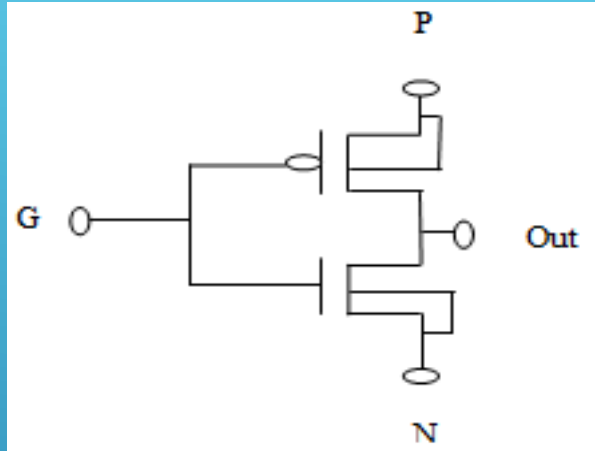


fig: Basic GDI Unit Cell

- ❖ In digital circuit design, Gate Diffusion Input (GDI) is a technique used to implement logic functions using fewer transistors compared to traditional CMOS (Complementary Metal-Oxide-Semiconductor) logic.
- ❖ It was introduced as a method to reduce power consumption, propagation delay, and area in integrated circuits (ICs), especially for low-power applications.

what Is m-GDI?

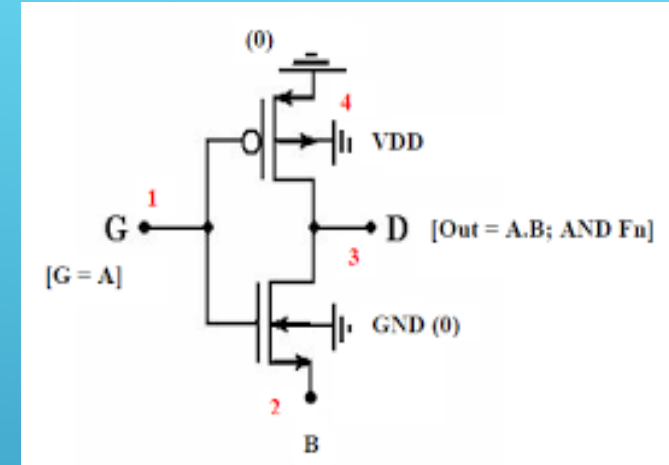


fig: Basic m-GDI Unit Cell

- ❖ Modified Gate Diffusion Input (M-GDI) is a low-power digital design technique.
- ❖ It allows the design of complex logic gates with fewer transistors than conventional CMOS logic.
- ❖ M-GDI offers improved power consumption, area efficiency, and propagation delay.

Decoder for ALU

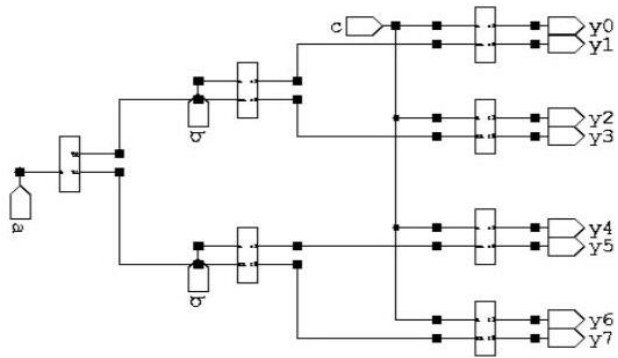


Fig: Decoder 1-8 Basic Circuit Diagram

2T-XOR Gate

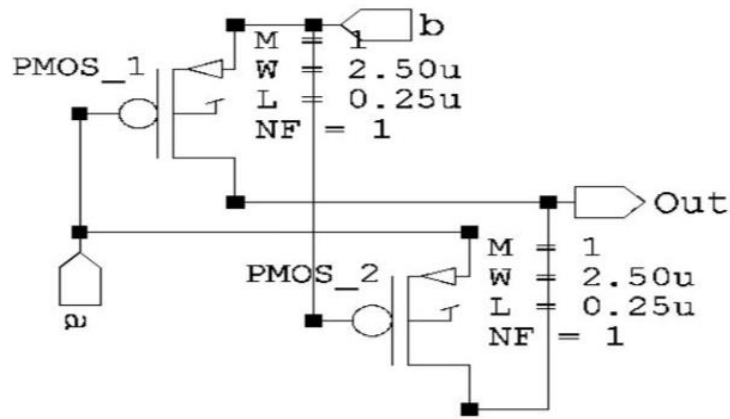


Fig: m-GDI is used in the design of a 2T XOR

- ❖ A decoder translates control signals into select lines for different ALU operations.
- ❖ The M-GDI technique can also be used to design an efficient decoder circuit with fewer transistors.

- ❖ A standard XOR gate requires multiple transistors, but using the M-GDI technique, it can be reduced to just two transistors.
- ❖ XOR gates are essential in arithmetic operations like addition (for sum bits in full adders).

Key Components for ALU Design:

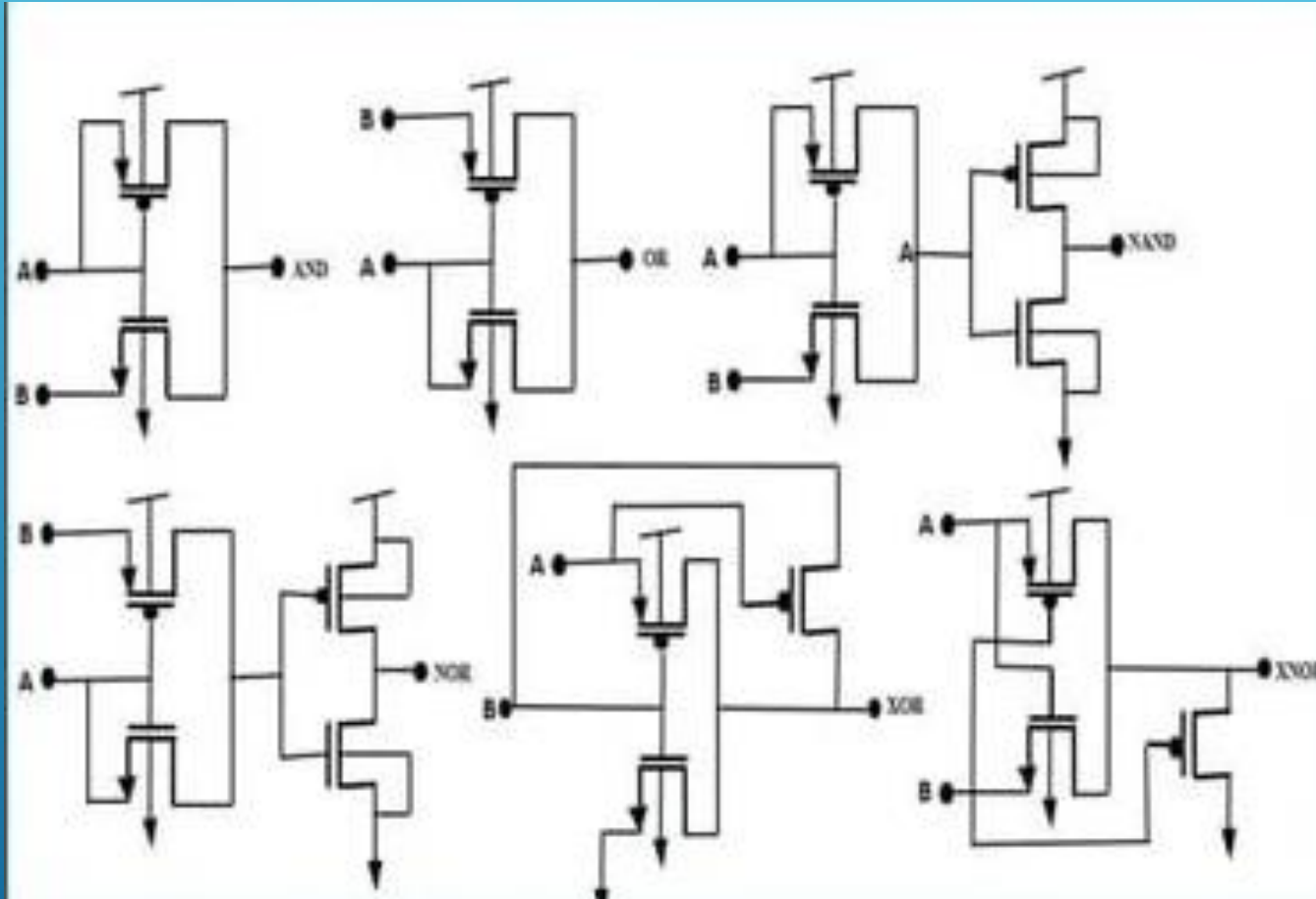
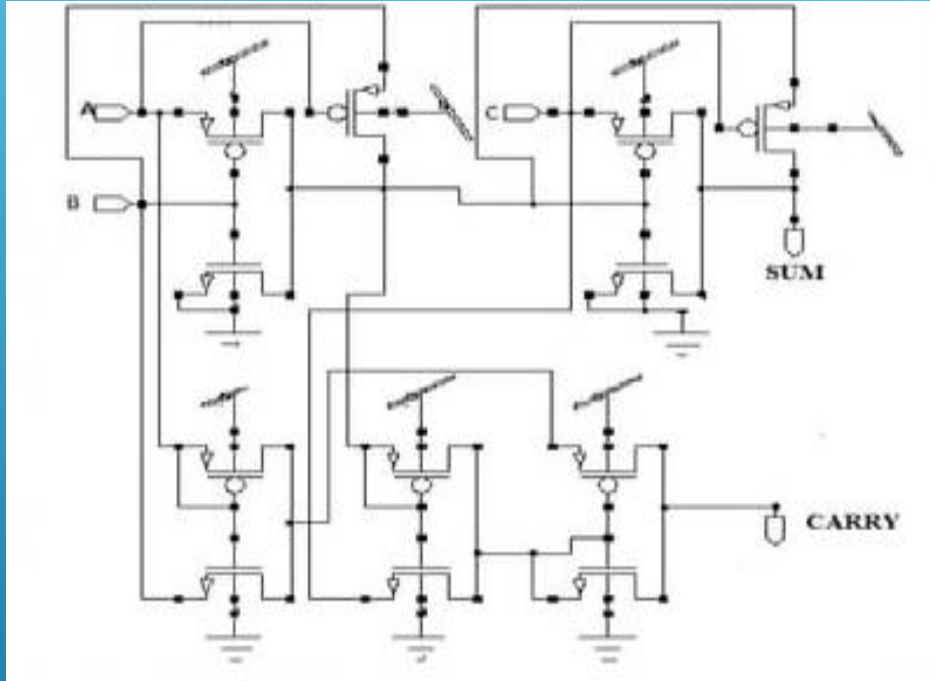


Fig: Basic logic gates using m-GDI technique

N	P	G	OUT	FUNCTION
0	B	A	$\bar{A}B$	F1
B	1	A	$\bar{A} + B$	F2
A	B	A	$A + B$	OR
B	A	A	AB	AND
C	B	A	$\bar{A}B + AC$	MUX
0	1	A	\bar{A}	NOT

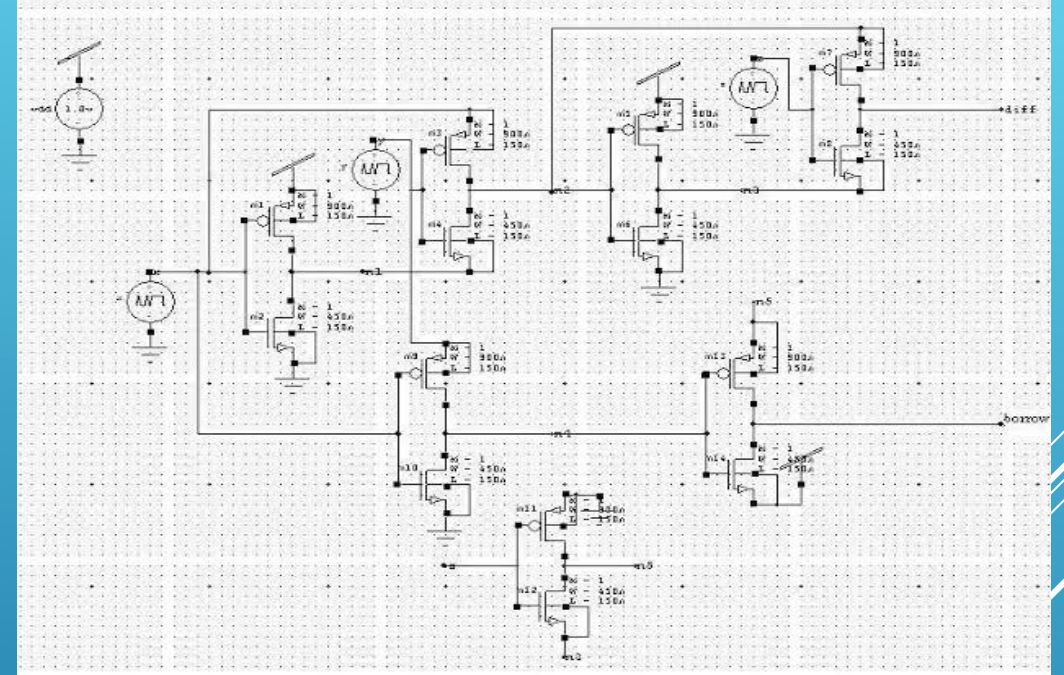
Full Adder Using m-GDI Technique:



$$\text{Sum} = A \oplus B \oplus C$$

$$\text{Carry} = AB + BC + CA$$

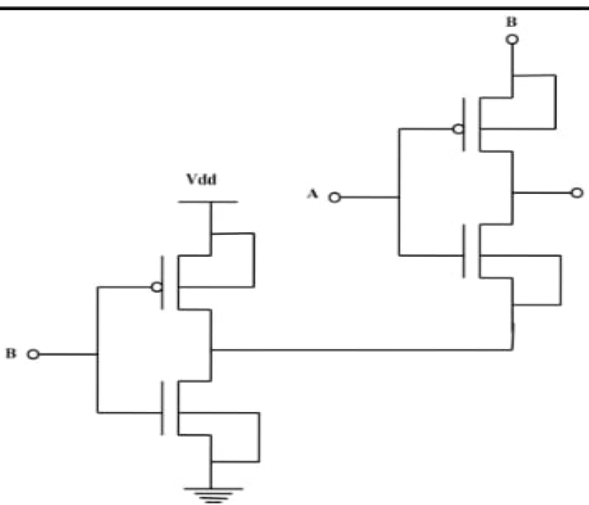
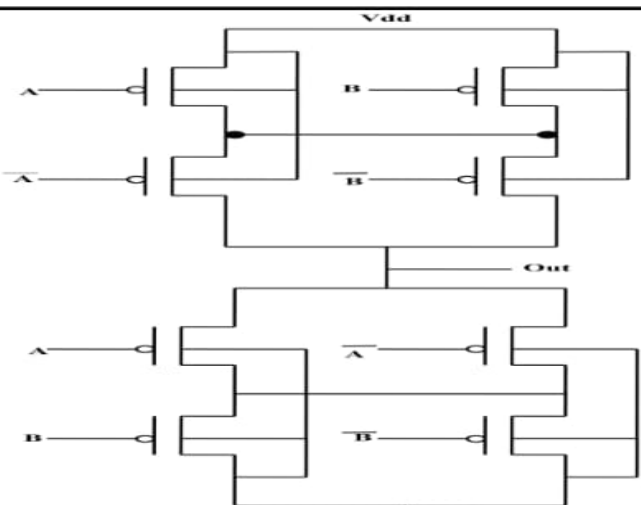
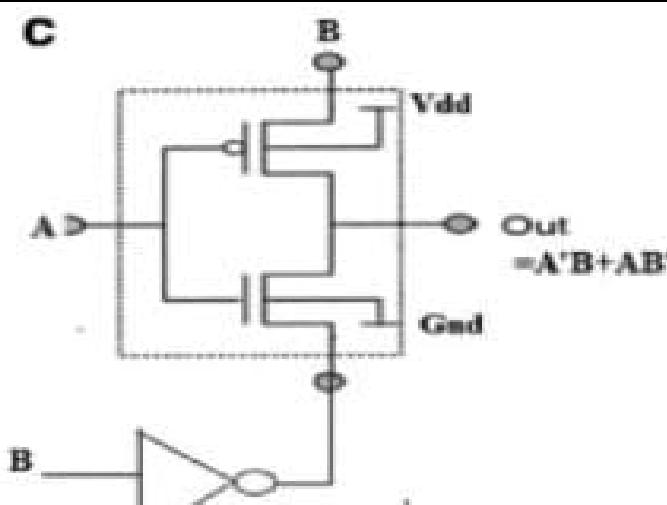
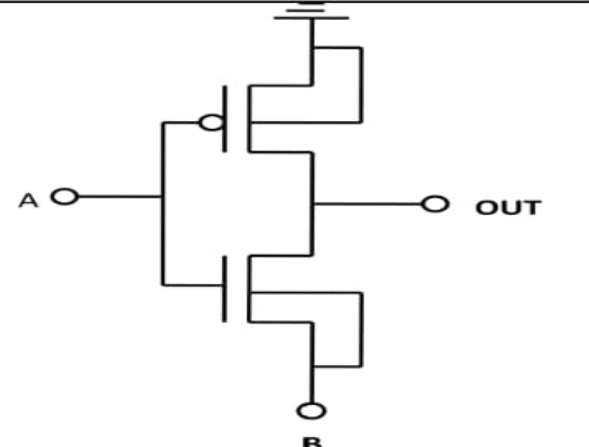
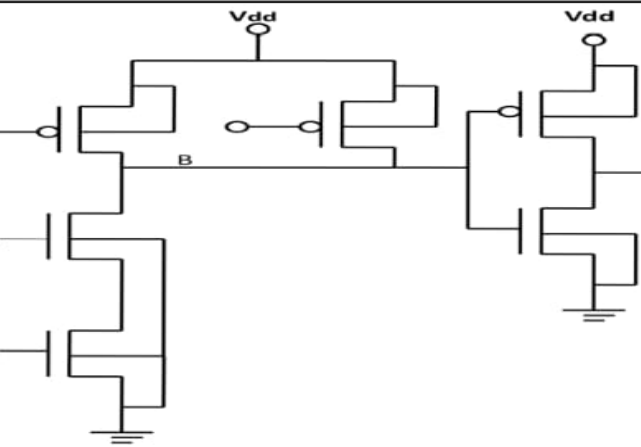
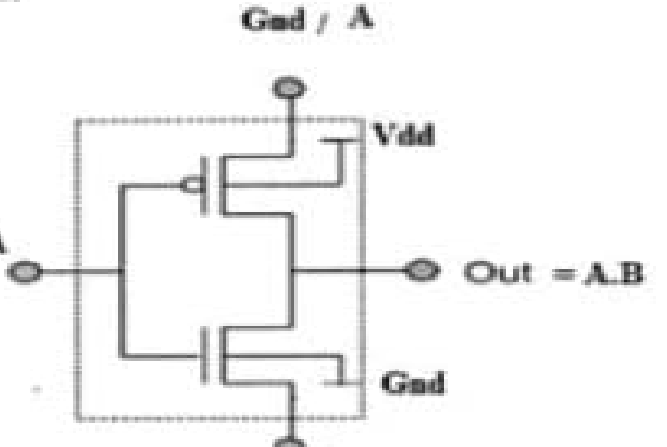
Full Subtractor Using m-GDI Technique:

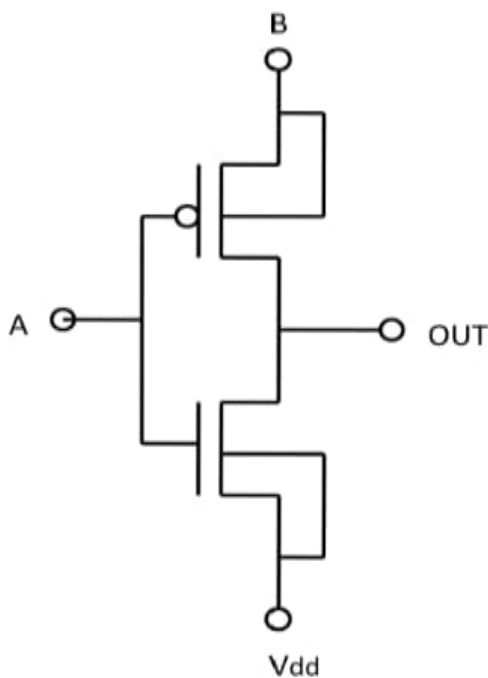
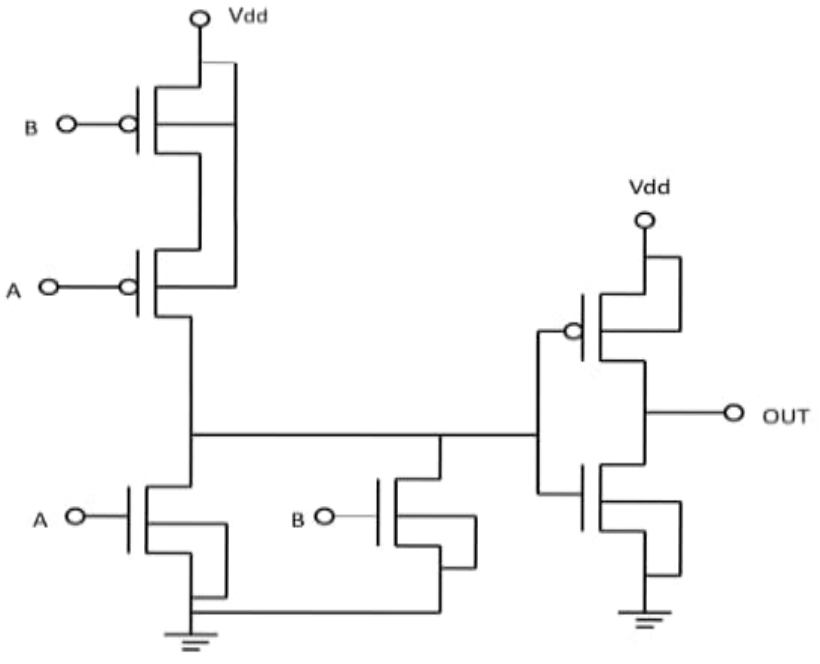
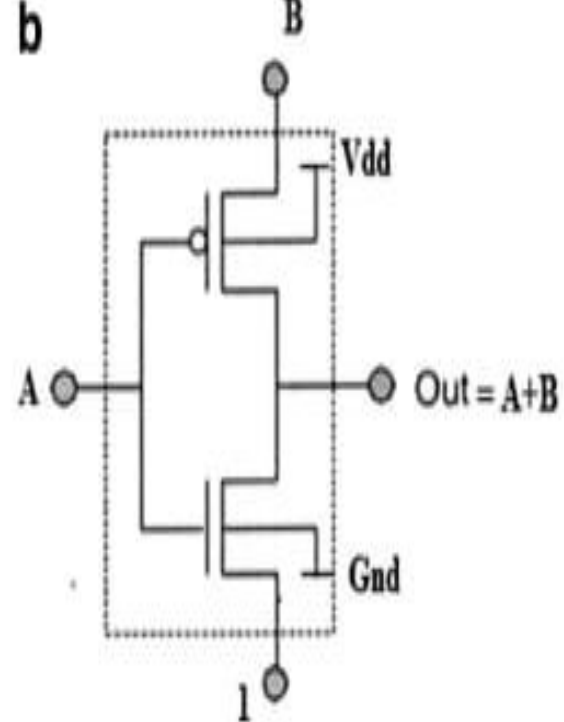


$$\text{Difference} = A \oplus B \oplus C$$

$$\text{Barrow} = A'B + BC + CA'$$

Differences Between GDI,CMOS and m-GDI:

Logic Gates	GDI	CMOS	M-GDI
XOR	 <p>Transistor count- 4</p>	 <p>Transistor count- 8 +4(for generating A' & B')</p>	 <p>Out = $A'B + AB'$</p>
AND	 <p>Transistor count- 2</p>	 <p>Transistor count-6</p>	 <p>Out = $A.B$</p>

Logic Gates	GDI	CMOS	M-GDI
OR	 <p>A GDI OR gate circuit diagram. It features a PMOS transistor at the top with its gate connected to input A and its source connected to input B. The drain of this PMOS transistor is connected to the output node OUT. A single NMOS transistor is at the bottom with its gate connected to input A and its source connected to ground (Vdd). The drain of this NMOS transistor is also connected to the output node OUT.</p> <p>Transistor count- 2</p>	 <p>A CMOS OR gate circuit diagram. The PMOS network consists of two PMOS transistors in parallel, with gates connected to inputs B and A respectively, and their sources connected to Vdd. The NMOS network consists of two NMOS transistors in series, with gates connected to inputs A and B respectively, and their sources connected to ground. The output node OUT is connected to the drains of both the PMOS and NMOS networks.</p> <p>Transistor count- 6</p>	 <p>A schematic diagram of an M-GDI OR gate, labeled 'b'. It shows a PMOS transistor with gate A and source B, and an NMOS transistor with gate A and source connected to ground (Gnd). The output node is labeled 'Out = A+B'. A dashed box encloses the PMOS transistor and the output node. A label '1' is at the bottom.</p>

Here are key differences between m-GDI, GDI and CMOS:

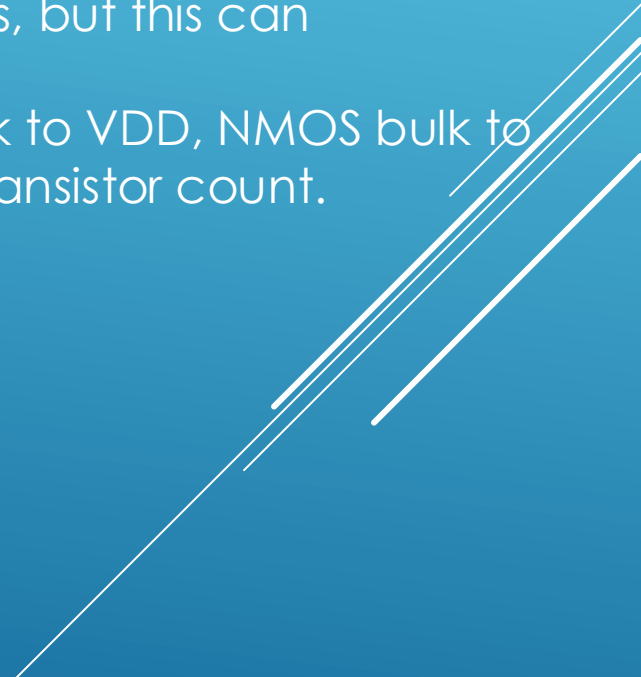
1. Number of Transistors:

- CMOS: Requires a large number of transistors to implement basic logic functions (typically twice the number of transistors, as it uses both NMOS and PMOS transistors).
- GDI: Reduces the number of transistors by using a more flexible configuration of inputs (Gate, Source, Drain, and Bulk) to create logic gates.
- m-GDI: Similar to GDI but ensures proper bulk connections (PMOS bulk to VDD and NMOS bulk to GND) for stable operation, still using fewer transistors compared to CMOS.

2. Power Consumption:

- CMOS: Consumes more power due to the higher number of transistors and switching activities.
- GDI : Consumes less power because it uses fewer transistors and can avoid static power dissipation in some configurations.
- m-GDI: Maintains the low power consumption of GDI but improves stability by avoiding issues like body effect and latch-up.

3. Bulk Connection:

- CMOS: Bulk connections are fixed (NMOS bulk to GND, PMOS bulk to VDD) and do not change with the inputs.
 - GDI: Bulk connections can vary and be connected to different input signals, but this can introduce instability or require complex fabrication.
 - m-GDI: Simplifies the GDI method by fixing the bulk connections (PMOS bulk to VDD, NMOS bulk to GND), ensuring better stability and easier fabrication, while still reducing the transistor count.
- 
- A series of white diagonal lines of varying lengths and thicknesses are positioned in the bottom right corner of the slide, creating a modern, abstract graphic element.

Source Code:

// XOR Gate - Verilog Representation of the 2T XOR Gate

```
module xor_gate (  
    input a,  
    input b,  
    output y  
);  
    assign y = a ^ b; // XOR operation  
endmodule
```

// 2:4 Decoder - Used to control which operation is selected in the ALU

```
module decoder_2x4 (  
    input [1:0] select,  
    output reg [3:0] out  
);  
    always @(*) begin  
        case (select)  
            2'b00: out = 4'b0001; // Select Operation 0 (Addition)  
            2'b01: out = 4'b0010; // Select Operation 1 (Subtraction)  
            2'b10: out = 4'b0100; // Select Operation 2 (AND)  
            2'b11: out = 4'b1000; // Select Operation 3 (XOR)  
            default: out = 4'b0000; // Default case  
        endcase  
    end  
end
```

```
// 8-bit ALU Module
module alu_8bit (
    input [7:0] A,    // 8-bit Input A
    input [7:0] B,    // 8-bit Input B
    input [1:0] select, // 2-bit selection from decoder to
choose operation
    output reg [7:0] result, // 8-bit ALU result
    output carry_out // Carry output (for addition and
subtraction)
);
    wire [7:0] sum, diff, and_op, xor_op;
    wire [3:0] control; // 4-bit control signal from decoder

    // Instantiate the 2T XOR gate module for each bit
    xor_gate xor0(.a(A[0]), .b(B[0]), .y(xor_op[0]));
    xor_gate xor1(.a(A[1]), .b(B[1]), .y(xor_op[1]));
    xor_gate xor2(.a(A[2]), .b(B[2]), .y(xor_op[2]));
    xor_gate xor3(.a(A[3]), .b(B[3]), .y(xor_op[3]));
    xor_gate xor4(.a(A[4]), .b(B[4]), .y(xor_op[4]));
    xor_gate xor5(.a(A[5]), .b(B[5]), .y(xor_op[5]));
    xor_gate xor6(.a(A[6]), .b(B[6]), .y(xor_op[6]));
    xor_gate xor7(.a(A[7]), .b(B[7]), .y(xor_op[7]));
```

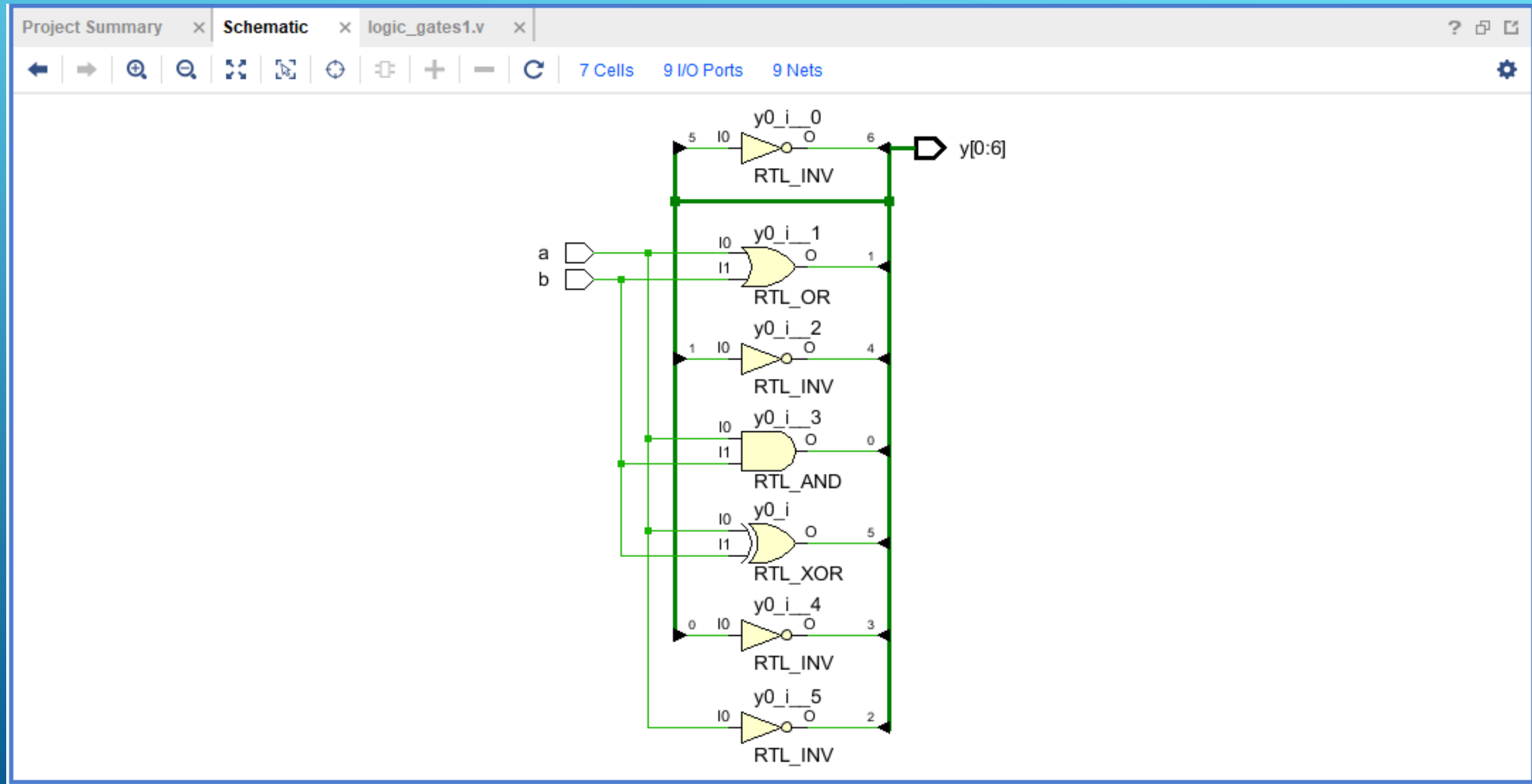
```
// Perform Arithmetic Operations
assign {carry_out, sum} = A + B; // Addition
assign diff = A - B;           // Subtraction
```

```
// Perform Logical Operations
assign and_op = A & B;         // AND
Operation
```

```
// Instantiate the 2:4 decoder
decoder_2x4 dec(.select(select),
.out(control));
```

```
// ALU Result Selection Based on Decoder
Output
always @(*) begin
    case (control)
        4'b0001: result = sum; // Addition
        4'b0010: result = diff; // Subtraction
        4'b0100: result = and_op; // AND
        4'b1000: result = xor_op; // XOR
        default: result = 8'b00000000; // Default:
No operation
    endcase
end
endmodule
```

Schematic Diagram

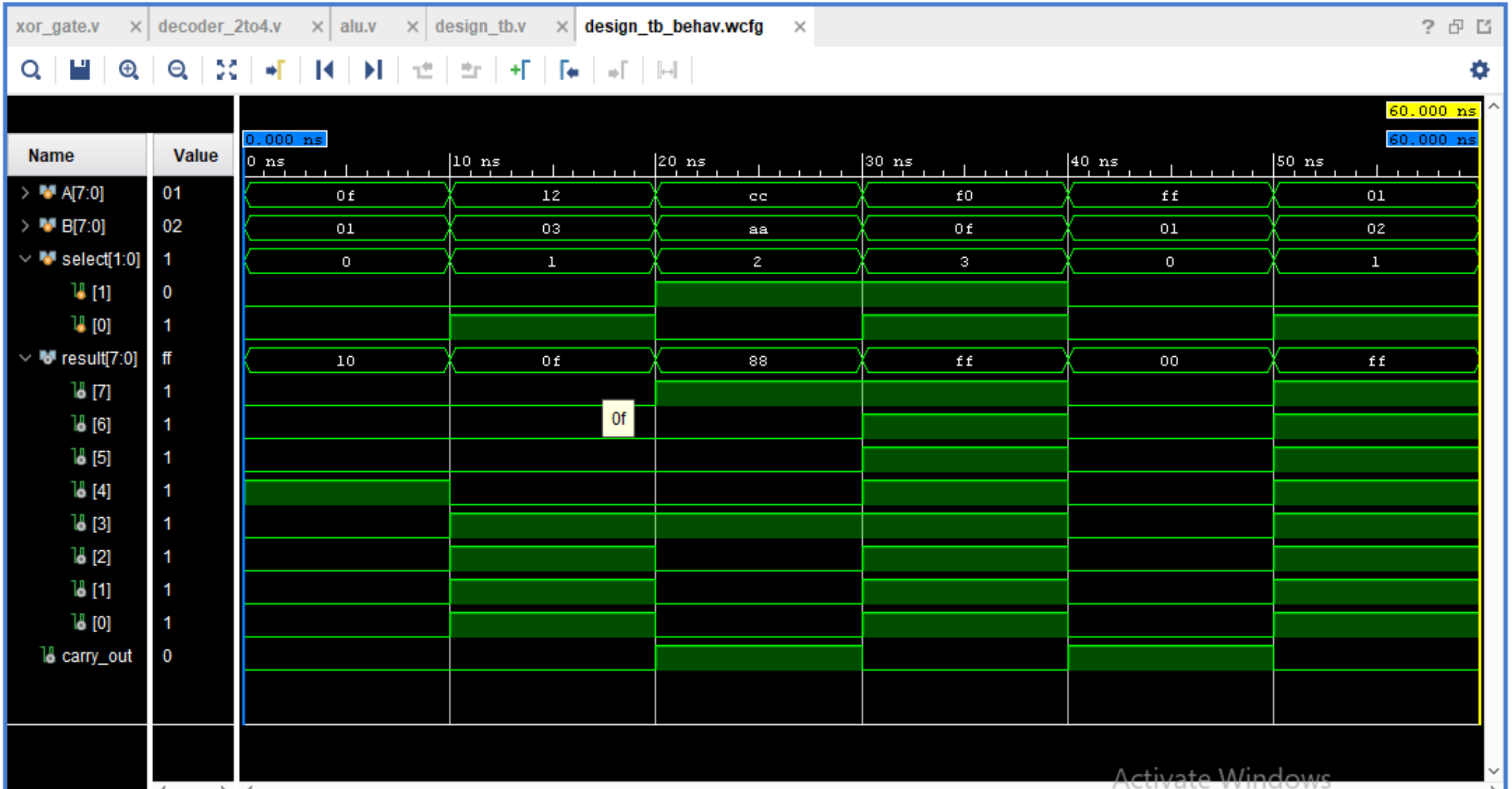


Testbench Code

```
module design_tb;
// Inputs
  reg [7:0] A;
  reg [7:0] B;
  reg [1:0] select;
// Outputs
  wire [7:0] result;
  wire carry_out;
// Instantiate the ALU module
  alu uut( .A(A), .B(B), .select(select), .result(result), .carry_out(carry_out) );
// Test vectors and stimulus initial begin
// Monitor outputs
  $monitor("Time: %0d A=%b B=%b select=%b -> result=%b carry_out=%b", $time, A, B, select, result,
  carry_out); // Initialize Inputs and apply test cases
// Test Case 1: Addition (A + B)
  A = 8'b00001111; // 15 in decimal
  B = 8'b00000001; // 1 in decimal
  select = 2'b00; // Select addition operation
  #10; // Wait 10 time units
// Test Case 2: Subtraction (A - B)
  A = 8'b00010010; // 18 in decimal
  B = 8'b00000011; // 3 in decimal
  select = 2'b01; // Select subtraction operation
  #10; // Wait 10 time units
```

```
// Test Case 3: AND Operation (A & B)
A = 8'b11001100; // A = 204
B = 8'b10101010; // B = 170
select = 2'b10; // Select AND operation
#10; // Wait 10 time units
// Test Case 4: XOR Operation (A ^ B)
A = 8'b11110000; // A = 240
B = 8'b00001111; // B = 15
select = 2'b11; // Select XOR operation
#10; // Wait 10 time units
// Test overflow in addition
A = 8'b11111111; // A = 255
B = 8'b00000001; // B = 1
select = 2'b00; // Select addition operation
#10; // Wait 10 time units
// Test Case 5: Subtraction with negative result
A = 8'b00000001; // A = 1
B = 8'b00000010; // B = 2
select = 2'b01; // Select subtraction operation
#10; // Wait 10 time units
// Test complete
$finish; // End the simulation
end
endmodule
```


Simulation Results



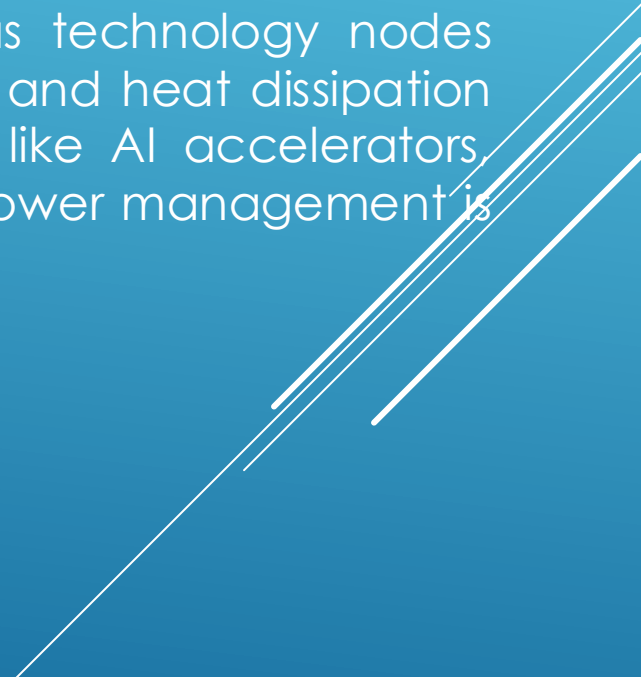
Conclusion:

Here we are implement the Arithmetic unit design using Modified Gate Diffusion Input and 2T-XOR, Logic Unit design Using Modified Gate Diffusion Input(m-GDI) and 2T-XOR , Compare m-GDI technique and GDI technique and Verilog Source Code.

Several white diagonal lines of varying lengths and thicknesses are positioned in the bottom right corner of the slide, creating a modern, abstract graphic element.

Future Scope:

The future scope for ALU design using the m-GDI technique with a 2T XOR gate and decoder is promising, especially in the context of low-power, high-performance computing. This approach can be further optimized for use in advanced microprocessors, embedded systems, and portable devices, where energy efficiency and space are critical. Additionally, as technology nodes continue to shrink, m-GDI-based designs can help mitigate power leakage and heat dissipation challenges. The technique can also be explored for use in applications like AI accelerators, Internet of Things (IoT) devices, and wearable electronics, where efficient power management is crucial.

Several white lines of varying lengths and slopes are positioned in the bottom right corner of the slide, creating a modern, abstract graphic element.

Expected Results:

Design of ALUs	Power(nW)	Delay [(Critical Path)-(ns)]	Count of Transistors
8 Bit ALU [8]	5.04	1.1	125
Proposed System	3.5	0.7	120

References:

- ❖ “Design of High Speed, Area Optimized and Low Power Arithmetic and Logic Unit”, Advances in Industrial Engineering and Management, Vol.6, No. 1 (2017), 26-31 by Md.Afreen Begum and S. Sweta.
- ❖ “Modified Gate Diffusion Input Technique: A New Techniqu for Enhancing Performance in Full Adder Circuits”, Procedia Technology 6 (2012) 74 –81 by P.Dhabachelvan and R.Uma
- ❖ “Gate-Diffusion Input (GDI): A Power-Efficient Method for Digital Combinatorial Circuits”, IEEE Transactions On Very Large Scale Integration (VLSI) Systems, Vol. 10, No. 5, October2002 by Israel A.Wagner ,Alexander Fish and ArkadiyMorgenshtein.

The background is a deep blue gradient. It is filled with a complex network of thin, glowing blue lines that crisscross and curve across the frame. Interspersed among these lines are numerous small, bright blue dots and larger, soft blue bokeh-like circles, creating a sense of depth and movement, similar to a starry night sky or a digital data visualization.

THANK YOU