

DESIGN AND ANALYSIS OF A LOW-POWER RECONFIGURABLE ALU USING FINE-GRAINED CLOCK GATING

Project Report

Submitted by

Perumallapalli Yamini

Department of Electronics and Communication Engineering

Rajiv Gandhi University of Knowledge and Technologies

(RGUKT), Nuzvid

Andhra Pradesh, India

Abstract

This project presents the design and analysis of a low-power 8-bit reconfigurable Arithmetic Logic Unit (ALU) using fine-grained clock gating. The proposed ALU performs eight arithmetic and logical operations, including addition, subtraction, AND, OR, XOR, NOT, increment, and decrement. Fine-grained clock gating is applied at the functional-block level to reduce unnecessary switching activity in inactive sections of the ALU, thereby improving power efficiency.

The design was functionally verified using Verilog simulation and synthesized using Xilinx Vivado. The conventional and clock-gated ALU implementations were compared in terms of power consumption, with the clock-gated design showing a reduction in total on-chip power from 5.791 W to 3.610 W, corresponding to approximately 37.66

Contents

1	Introduction	3
2	Objectives	3
3	Theory and Description	3
4	Architecture and Working Principle	4
5	Verilog Implementation	5
6	Simulation and Functional Verification	6
7	Esim Implemention	7
8	Esim Simulation	9
9	Results and Discussions	13
10	Applications	13
11	Conclusion	13
12	References	13

1. Introduction

An Arithmetic Logic Unit (ALU) is a fundamental component of a digital system that performs arithmetic and logical operations on binary data. This project focuses on the design of an 8-bit reconfigurable ALU capable of performing eight operations: addition, subtraction, AND, OR, XOR, NOT, increment, and decrement. Since inactive functional sections may cause unnecessary switching activity, fine-grained clock gating is used to enable only the required section according to the selected operation. This helps reduce unnecessary switching and improve the power efficiency of the ALU.

The designed ALU is functionally verified through simulation and analyzed for power consumption. The verified design is then migrated to the open-source eSim environment as part of the FOSSEE research migration objective. The eSim implementation aims to reproduce the ALU functionality, analyze the fine-grained clock-gating behavior, and evaluate power and resource utilization under equivalent simulation conditions.

2. Objectives

1. To design an 8-bit reconfigurable ALU with fine-grained clock gating.
2. To implement arithmetic and logical operations such as addition, subtraction, AND, OR, XOR, NOT, increment, and decrement.
3. To verify the functional operation of the gated 8-bit ALU through simulation.
4. To enable only the required functional section based on the selected operation and reduce unnecessary switching activity.
5. To implement and analyze the verified gated 8-bit ALU in the eSim open-source environment.

3. Theory and Description

The ALU accepts two 8-bit operands A and B , together with a 3-bit operation-select input Op . The output Y is 8 bits wide and a carry output is provided for arithmetic operations.

The design supports eight operation codes.

Table 1: ALU Operation Codes

Op[2:0]	Operation	Description
000	ADD	$A + B$
001	SUB	$A - B$
010	AND	$A \text{ AND } B$
011	OR	$A \text{ OR } B$
100	XOR	$A \text{ XOR } B$
101	NOT	Bitwise NOT of A
110	Increment	$A + 1$
111	Decrement	$A - 1$

4. Architecture and Working Principle

The proposed architecture divides ALU functionality into operation-related sections. The operation code determines the required computation.

Fine-grained clock-gating control is used to reduce clock activity in inactive sections. The selected section produces the ALU result, while the carry output indicates arithmetic carry where applicable.

Table 2: Functional Architecture

Inputs / Control	Functional Section	Outputs
$A[7 : 0], B[7 : 0]$	Arithmetic Unit	$Y[7 : 0]$, carry
$Op[2 : 0]$	Logic Unit	$Y[7 : 0]$
CLK, RESET	Increment / Decrement Unit	$Y[7 : 0]$
Clock-gating control	Operation-specific enable	Reduced switching activity

5. Verilog Implementation

```
module alu_8bit_gated (
    input clk,
    input reset,
    input [7:0] A,
    input [7:0] B,
    input [2:0] Op,
    output reg [7:0] Y,
    output reg carry
);
wire arithmetic_en;
wire logic_en;
wire incdec_en;
reg [7:0] arithmetic_result = 8'b0;
reg [7:0] logic_result = 8'b0;
reg [7:0] incdec_result = 8'b0;
reg arithmetic_carry = 1'b0;
assign arithmetic_en =
    (Op == 3'b000) || (Op == 3'b001);
assign logic_en =
    (Op == 3'b010) || (Op == 3'b011) ||
    (Op == 3'b100) || (Op == 3'b101);
assign incdec_en =
    (Op == 3'b110) || (Op == 3'b111);
/* Arithmetic Section */
always @(posedge clk) begin
    if (reset) begin
        arithmetic_result <= 8'b0;
        arithmetic_carry <= 1'b0;
    end
    else if (arithmetic_en) begin
        case (Op)
            3'b000:
                {arithmetic_carry,
                 arithmetic_result} <= A + B;
            3'b001: begin
                arithmetic_result <= A - B;
                arithmetic_carry <= 1'b0;
            end
            default: begin
                arithmetic_result <= 8'b0;
                arithmetic_carry <= 1'b0;
            end
        endcase
    end
end
/* Logic Section */
always @(posedge clk) begin
    if (reset) begin
        logic_result <= 8'b0;
    end
    else if (logic_en) begin
        case (Op)
            3'b010:
                logic_result <= A & B;
            3'b011:
                logic_result <= A | B;
            3'b100:
                logic_result <= A ^ B;
            3'b101:
                logic_result <= ~A;
        endcase
    end
end
```

```

        default:
            logic_result <= 8'b0;
        endcase
    end
end
/* Increment and Decrement Section */
always @(posedge clk) begin
    if (reset) begin
        incdec_result <= 8'b0;
    end
    else if (incdec_en) begin
        case (Op)
            3'b110:
                incdec_result <= A + 1'b1;
            3'b111:
                incdec_result <= A - 1'b1;
            default:
                incdec_result <= 8'b0;
        endcase
    end
end
/* Output Selection */
always @(*) begin
    Y = 8'b0;
    carry = 1'b0;
    case (Op)
        3'b000,
        3'b001: begin
            Y = arithmetic_result;
            carry = arithmetic_carry;
        end
        3'b010,
        3'b011,
        3'b100,
        3'b101:
            Y = logic_result;
        3'b110,
        3'b111:
            Y = incdec_result;
        default: begin
            Y = 8'b0;
            carry = 1'b0;
        end
    endcase
end
endmodule

```

6. Simulation and Functional Verification

A Verilog testbench was used to exercise the ALU operations.

The waveform contains the following signals:

- CLK
- RESET
- A

- B
- Op
- Y
- carry

The operation code changes across the eight functions. The final test uses

$$A = FF, \quad B = 01, \quad Op = 000$$

which produces

$$Y = 00, \quad carry = 1$$

This demonstrates the 8-bit addition overflow/carry condition.

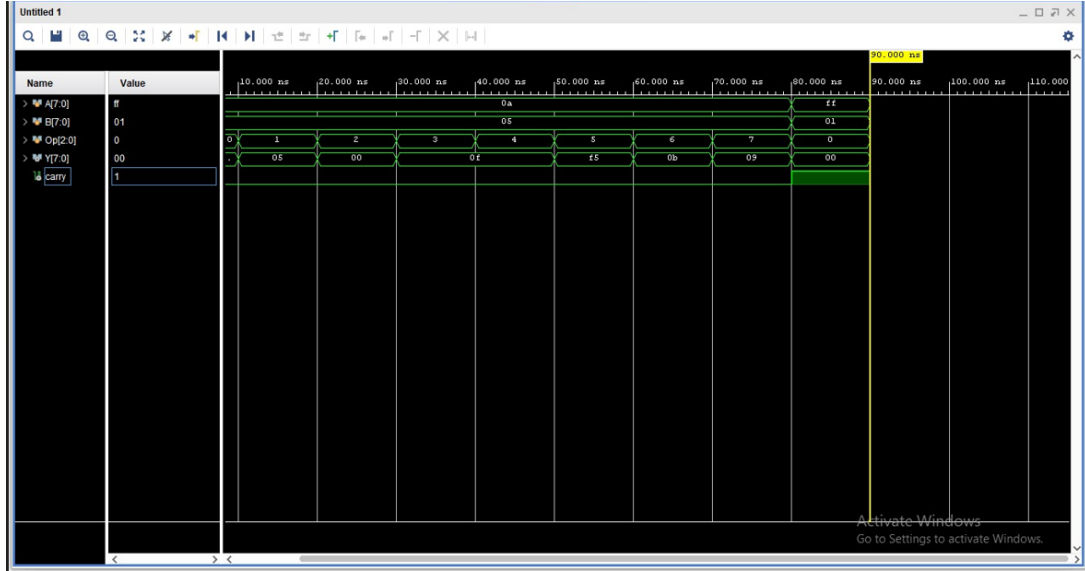


Figure 1: Simulation results in vivado

7. Esim Implementation

The 8-bit gated ALU is implemented in the eSim open-source EDA environment to verify its functional operation. The circuit accepts two 8-bit inputs, A and B, along with a 3-bit operation-select input, CLK, and RESET. The operation-select input determines the required ALU operation. The design supports eight operations: addition, subtraction, AND, OR, XOR, NOT, increment, and decrement. Fine-grained clock gating is used to control the clock activity of the corresponding functional section based on the selected operation, thereby reducing unnecessary switching activity in inactive sections. The eSim schematic is constructed by connecting the ALU functional blocks, clock-gating control, input signals, and output signals.

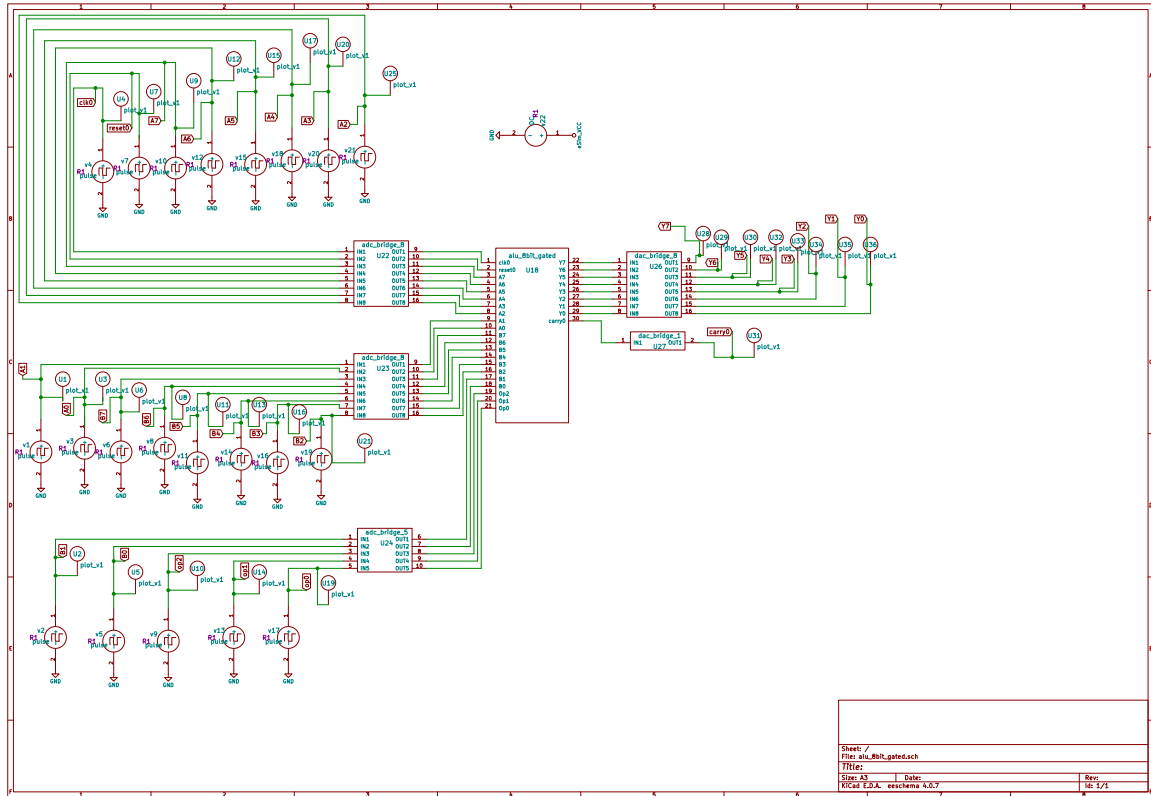


Figure 2: Circuit Diagram of the Proposed ALU

8. Esim Simulation

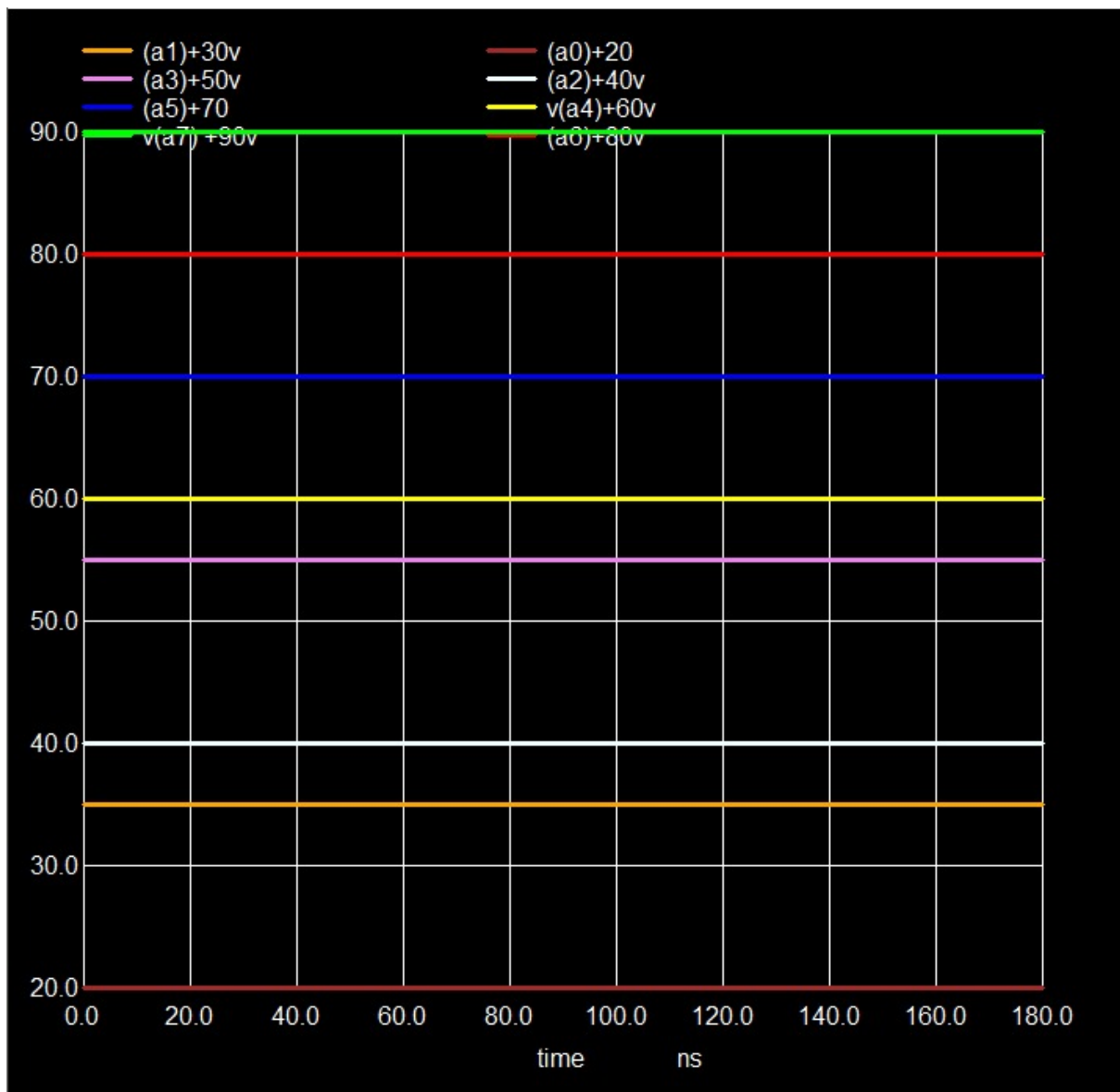


Figure 3: eSim simulation of 8-bit ALU of 'a' inputs

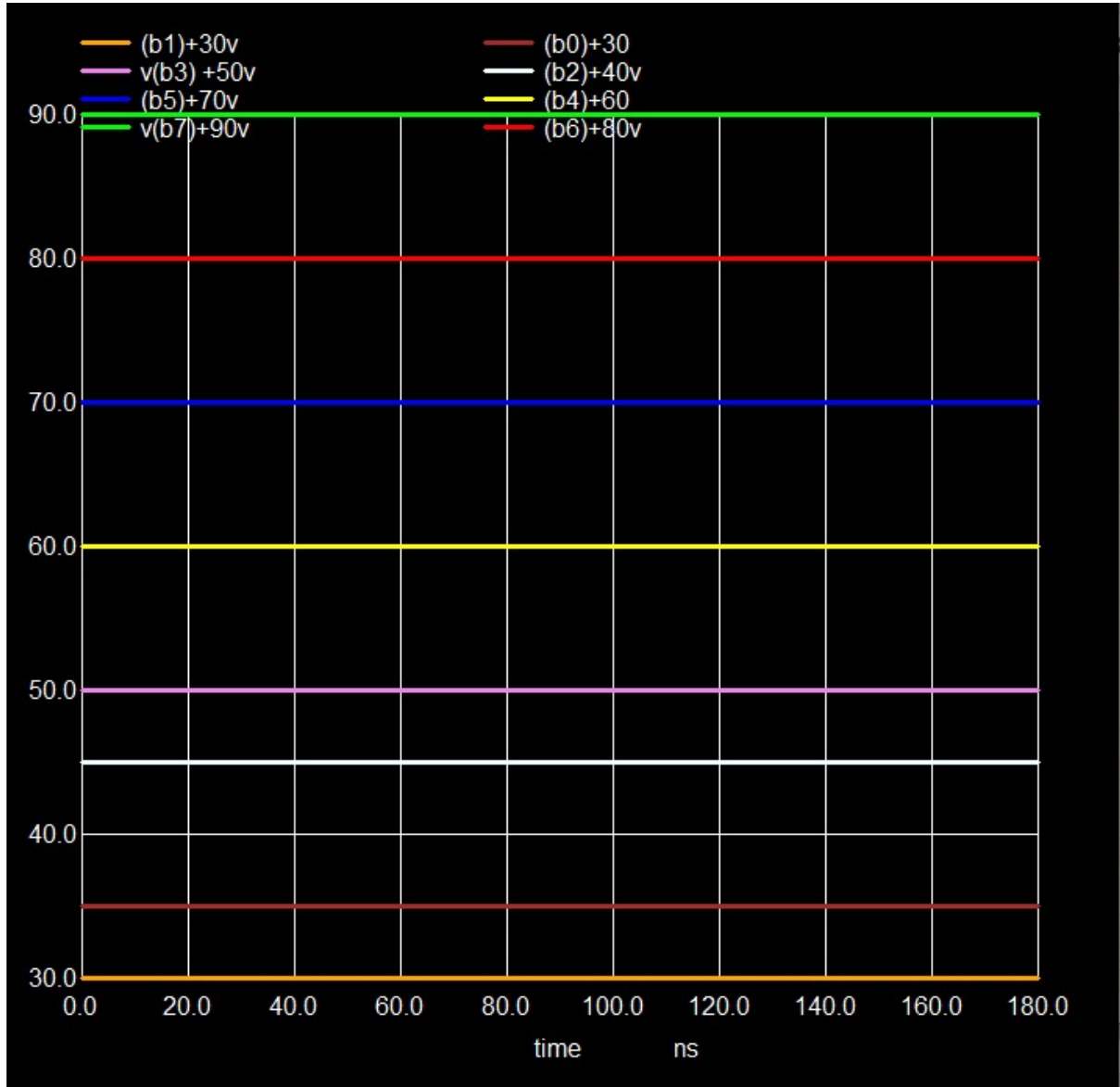


Figure 4: eSim simulation of 8-bit ALU of 'b' inputs

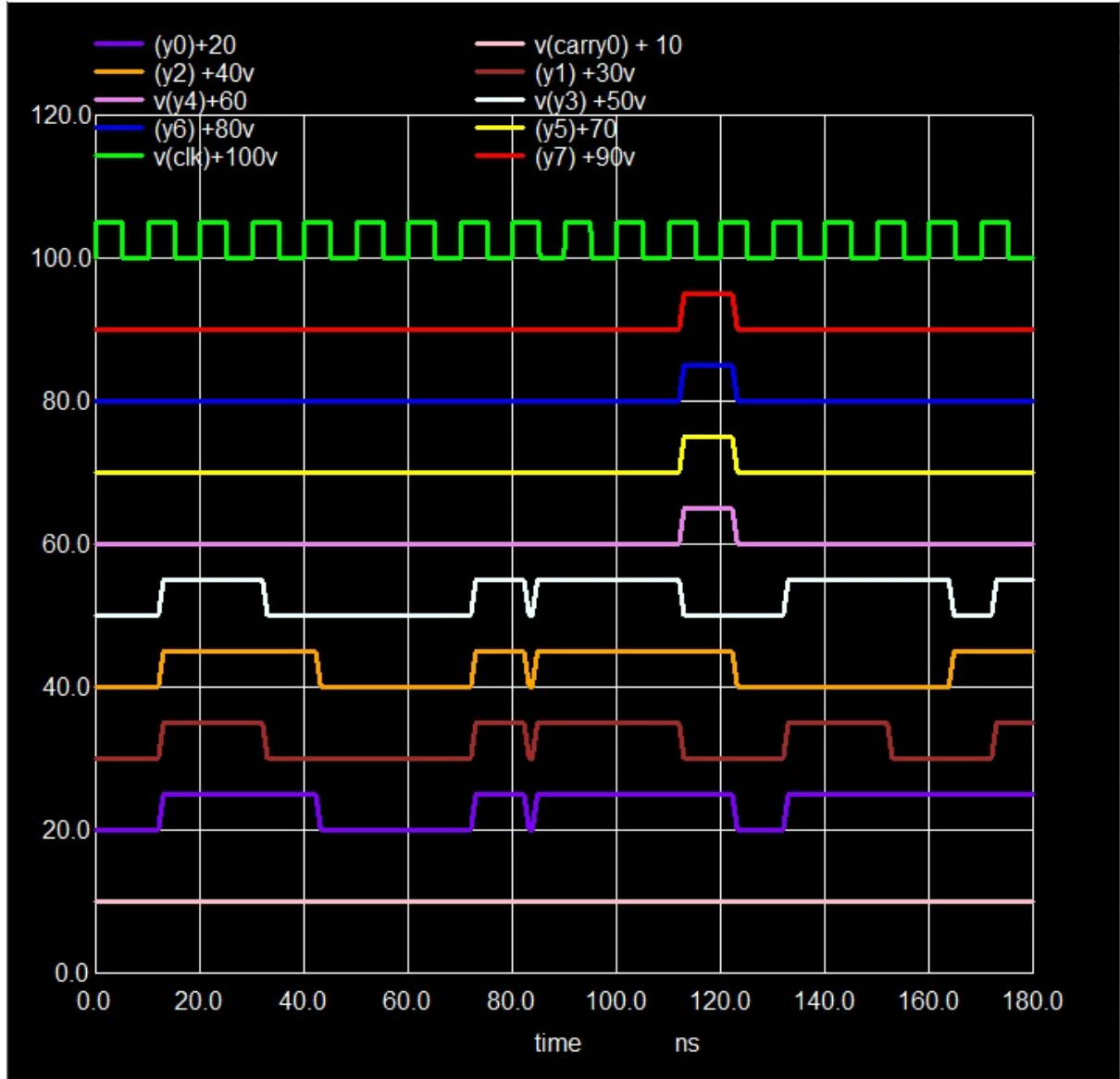
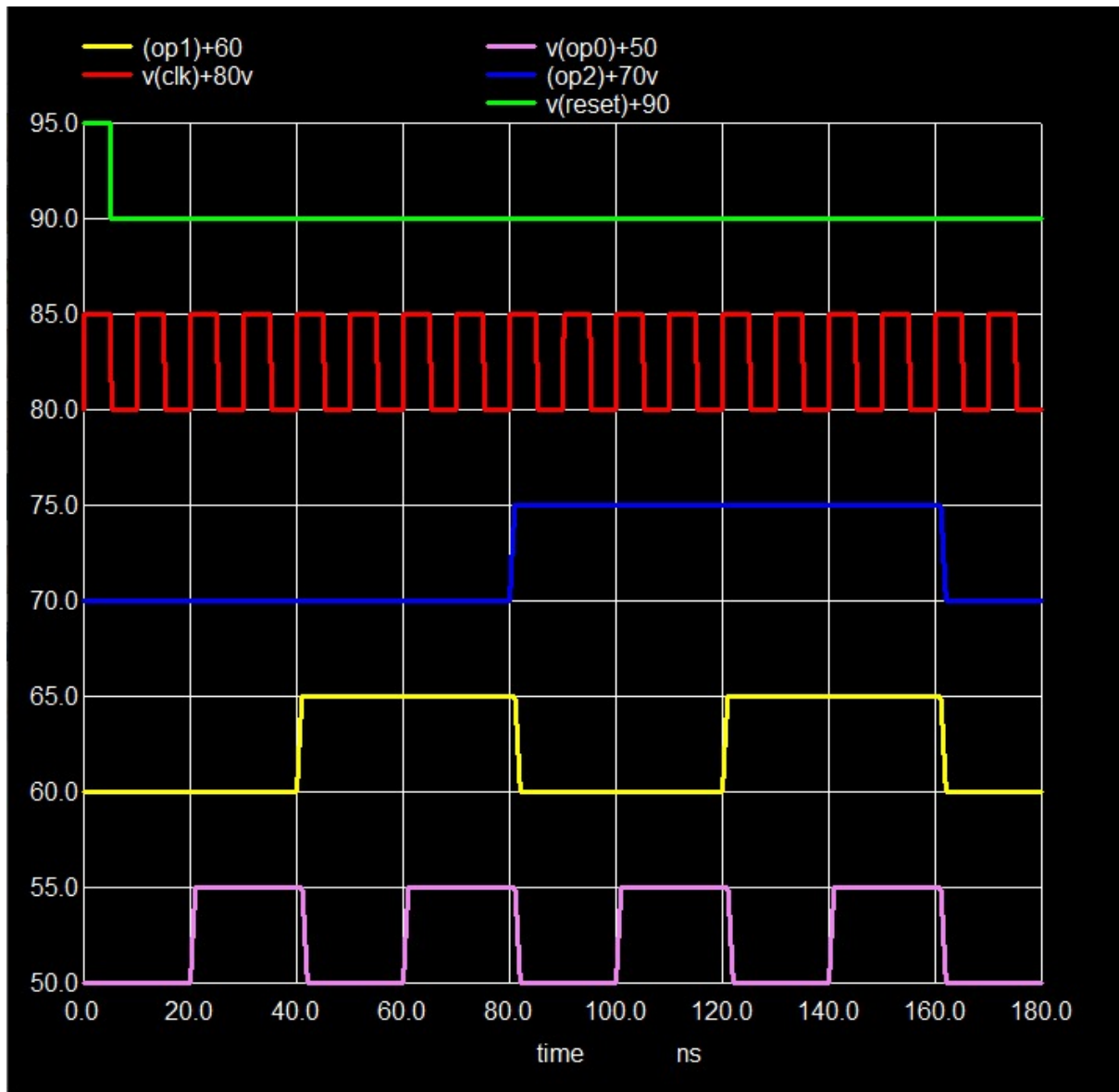


Figure 5: eSim simulation of 8-bit ALU of outputs



9. Results and Discussions

The 8-bit gated ALU was successfully implemented and functionally verified in eSim. The simulation confirms the correct operation of all eight functions: ADD, SUB, AND, OR, XOR, NOT, increment, and decrement. The fine-grained clock-gating technique enables the required functional section and reduces unnecessary switching activity.

For reference, the Vivado analysis reported total on-chip power of 5.791 W for the conventional ALU and 3.610 W for the gated ALU, corresponding to a 37.66 reduction. Dynamic power was reduced by 37.29, and static power by 44.59. These values are used only as reference results because only the gated 8-bit ALU was implemented in eSim.

10. Applications

- Low-power processor datapaths
- Embedded and portable digital systems
- Microcontrollers and digital controllers
- FPGA-based digital systems
- Battery-operated electronic devices
- Low-power arithmetic and logic processing units

11. Conclusion

The 8-bit reconfigurable ALU with fine-grained clock gating was successfully designed and functionally verified. The ALU performs eight arithmetic and logical operations, while clock gating reduces unnecessary switching activity in inactive functional sections. The eSim implementation provides functional verification of the gated ALU. The reference Vivado results indicate a reduction in total on-chip power from 5.791 W to 3.610 W, corresponding to approximately 37.66 power reduction. Since only the gated ALU is implemented in eSim, these power values are treated as reference Vivado results rather than independent eSim measurements.

12. References

- [1] eSim User Manual, FOSSEE, IIT Bombay.
- [2] M. Morris Mano, Digital Design, Pearson Education.
- [3] Xilinx Vivado Design Suite User Guide, AMD/Xilinx.