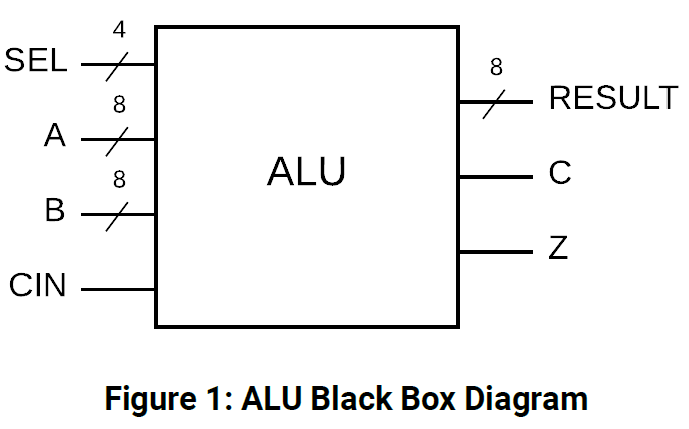
CPE 233: RAT assignment 4, ALU

Prof. Bridget Benson

Luis Gomez, Matt Bailey

# BBD



# Behavior

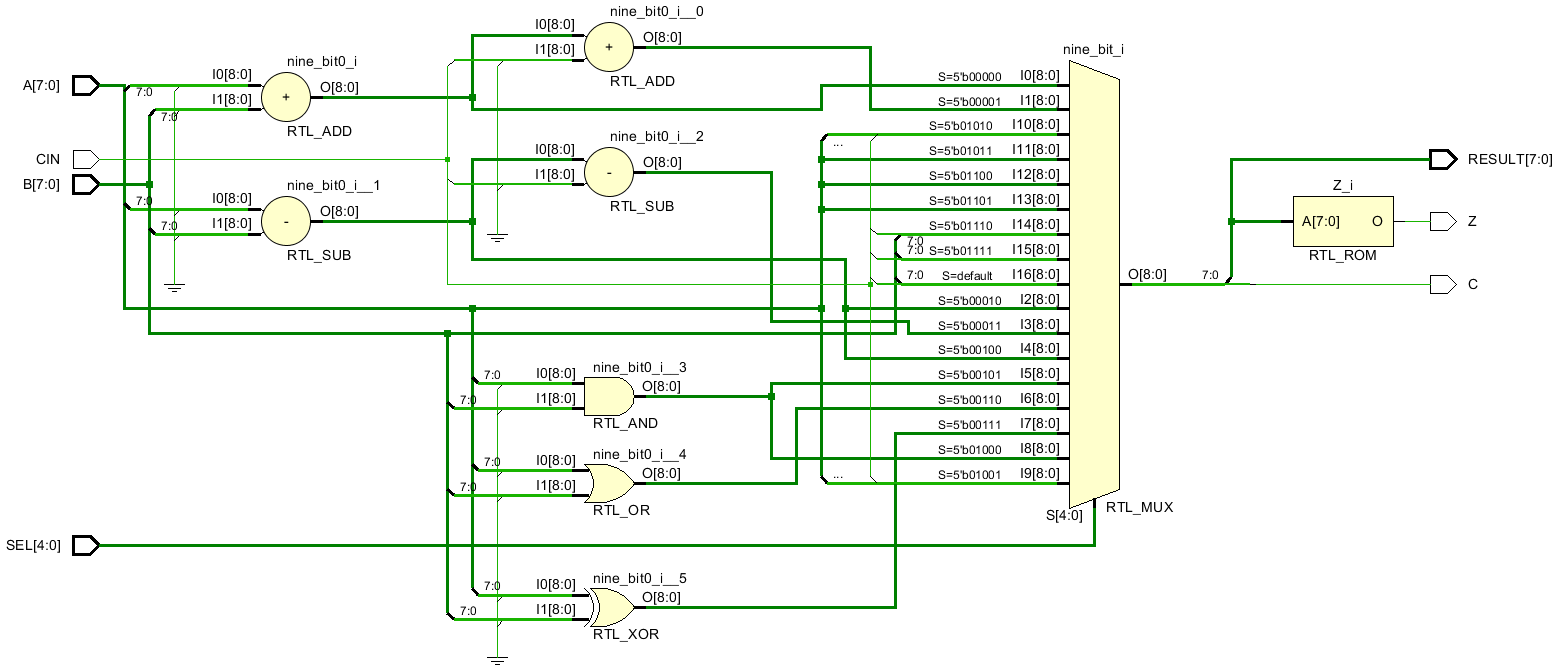
In this assignment, we built an Arithmetic Logic Unit in System Verilog. The module behavior can be described as following:

**Arithmetic Logic Unit (ALU)**: A combinatorial circuit that supports 16 instructions for the RAT architecture computer. The instructions include arithmetic operations, bit-wise operations, and logical-shift operations. The ALU also provides carry and zero flags for use in these operations. The tables below outline the inputs, outputs, and operations provided by the module:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SEL** | **Instruction** |  | **SEL** | **Instruction** |
| 0000 | **ADD** |  | 1000 | **TEST** |
| 0001 | **ADDC** |  | 1001 | **LSL** |
| 0010 | **SUB** |  | 1010 | **LSR** |
| 0011 | **SUBC** |  | 1011 | **ROL** |
| 0100 | **CMP** |  | 1100 | **ROR** |
| 0101 | **AND** |  | 1101 | **ASR** |
| 0110 | **OR** |  | 1110 | **MOV** |
| 0111 | **EXOR** |  | 1111 | **unused** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **bit width** | **Inputs** |  | **bit width** | **Outputs** |
| 8 | **A** |  | 8 | **RESULT** |
| 8 | **B** |  | 1 | **Z** |
| 4 | **SEL** |  | 1 | **C** |
| 1 | **CIN** |  |  |  |

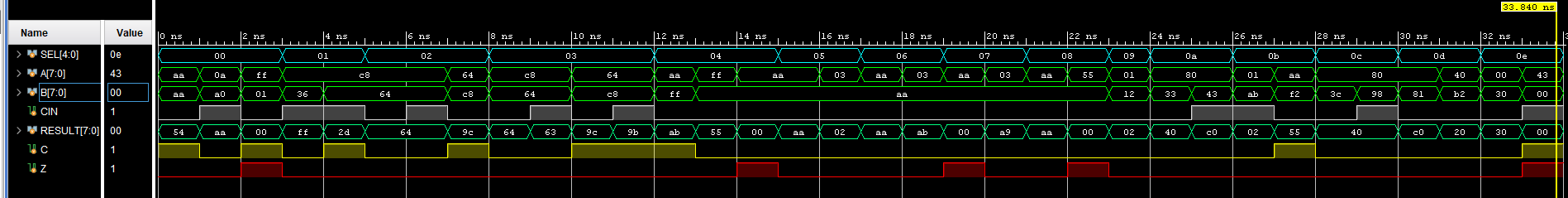
# Structural Design



# Verification

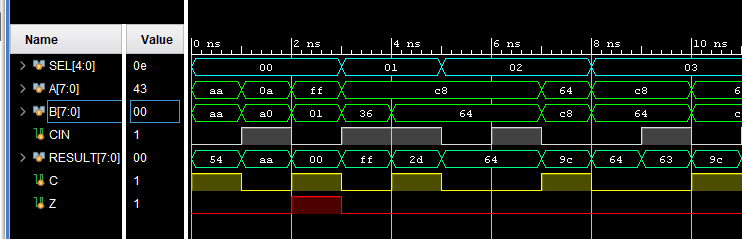
To verify the behavior of our Arithmetic Logic Unit, we created a simulation testbench that ran a total of 34 different test cases. Below we have provided a table of test case inputs, expected outputs, and testbench outputs. Further below we have provided the complete color-coded waveform, broken into three images for visual clarity. Because of the sheer number of tests, we cannot go into detail to justify the chosen parameters for each case. That said, we can generalize the description of the tests below as edge-case tests and basic functionality tests.

*Waveform in its entirety*

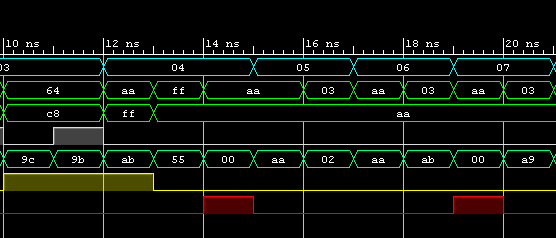


*Waveform snippets on next page…*

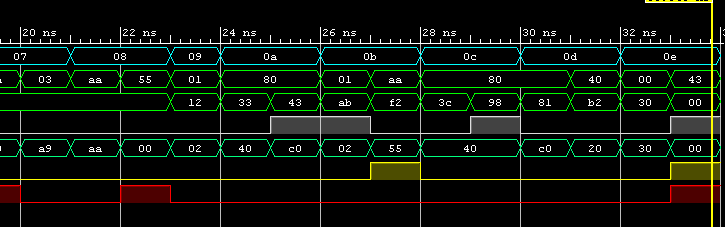
*Waveform snippet 1: [0-10.5] ns*



*Waveform snippet 2: [10-20.5] ns*



*Waveform snippet 3: [20-34] ns*



*Test Cases*

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Arguments** |  |  | **Expected** |  |  | **Testbench** |  |  | **Simulation** |
| **Test #** | **Function** | **SEL** | **A** | **B** | **Cin** | **Result (hex)** | **C** | **Z** | **Result (hex)** | **C** | **Z** | **Time (ns)** |
| **1** | **ADD** | **0000** | **0xAA** | **0xAA** | **0** | 0x54 | 1 | 0 | 0x54 | 1 | 0 | 0 |
| **2** |  |  | **0x0A** | **0xA0** | **1** | 0xAA | 0 | 0 | 0xAA | 0 | 0 | 1 |
| **3** |  |  | **0xFF** | **0x01** | **0** | 0x00 | 1 | 1 | 0x00 | 1 | 1 | 2 |
| **4** | **ADDC** | **0001** | **0xC8** | **0x36** | **1** | 0xFF | 0 | 0 | 0xFF | 0 | 0 | 3 |
| **5** |  |  | **0xC8** | **0x64** | **1** | 0x2D | 1 | 0 | 0x2D | 1 | 0 | 4 |
| **6** | **SUB** | **0010** | **0xC8** | **0x64** | **0** | 0x64 | 0 | 0 | 0x64 | 0 | 0 | 5 |
| **7** |  |  | **0xC8** | **0x64** | **1** | 0x64 | 0 | 0 | 0x64 | 0 | 0 | 6 |
| **8** |  |  | **0x64** | **0xC8** | **0** | 0x9C | 1 | 0 | 0x9C | 1 | 0 | 7 |
| **9** | **SUBC** | **0011** | **0xC8** | **0x64** | **0** | 0x64 | 0 | 0 | 0x64 | 0 | 0 | 8 |
| **10** |  |  | **0xC8** | **0x64** | **1** | 0x63 | 0 | 0 | 0x63 | 0 | 0 | 9 |
| **11** |  |  | **0x64** | **0xC8** | **0** | 0x9C | 1 | 0 | 0x9C | 1 | 0 | 10 |
| **12** |  |  | **0x64** | **0xC8** | **1** | 0x9B | 1 | 0 | 0x9B | 1 | 0 | 11 |
| **13** | **CMP** | **0100** | **0xAA** | **0xFF** | **0** | 0xAB | 1 | 0 | 0xAB | 1 | 0 | 12 |
| **14** |  |  | **0xFF** | **0xAA** | **0** | 0x55 | 0 | 0 | 0x55 | 0 | 0 | 13 |
| **15** |  |  | **0xAA** | **0xAA** | **0** | 0x00 | 0 | 1 | 0x00 | 0 | 1 | 14 |
| **16** | **AND** | **0101** | **0xAA** | **0xAA** | **0** | 0xAA | 0 | 0 | 0xAA | 0 | 0 | 15 |
| **17** |  |  | **0x03** | **0xAA** | **0** | 0x02 | 0 | 0 | 0x02 | 0 | 0 | 16 |
| **18** | **OR** | **0110** | **0xAA** | **0xAA** | **0** | 0xAA | 0 | 0 | 0xAA | 0 | 0 | 17 |
| **19** |  |  | **0x03** | **0xAA** | **0** | 0xAB | 0 | 0 | 0xAB | 0 | 0 | 18 |
| **20** | **EXOR** | **0111** | **0xAA** | **0xAA** | **0** | 0x00 | 0 | 1 | 0x00 | 0 | 1 | 19 |
| **21** |  |  | **0x03** | **0xAA** | **0** | 0xA9 | 0 | 0 | 0xA9 | 0 | 0 | 20 |
| **22** | **TEST** | **1000** | **0xAA** | **0xAA** | **0** | 0xAA | 0 | 0 | 0xAA | 0 | 0 | 21 |
| **23** |  |  | **0x55** | **0xAA** | **0** | 0x00 | 0 | 1 | 0x00 | 0 | 1 | 22 |
| **24** | **LSL** | **1001** | **0x01** | **0x12** | **0** | 0x02 | 0 | 0 | 0x02 | 0 | 0 | 23 |
| **25** | **LSR** | **1010** | **0x80** | **0x33** | **0** | 0x40 | 0 | 0 | 0x40 | 0 | 0 | 24 |
| **26** |  |  | **0x80** | **0x43** | **1** | 0xC0 | 0 | 0 | 0xC0 | 0 | 0 | 25 |
| **27** | **ROL** | **1011** | **0x01** | **0xAB** | **1** | 0x02 | 0 | 0 | 0x02 | 0 | 0 | 26 |
| **28** |  |  | **0xAA** | **0xF2** | **0** | 0x55 | 1 | 0 | 0x55 | 1 | 0 | 27 |
| **29** | **ROR** | **1100** | **0x80** | **0x3C** | **0** | 0x40 | 0 | 0 | 0x40 | 0 | 0 | 28 |
| **30** |  |  | **0x80** | **0x98** | **1** | 0x40 | 0 | 0 | 0x40 | 0 | 0 | 29 |
| **31** | **ASR** | **1101** | **0x80** | **0x81** | **0** | 0xC0 | 0 | 0 | 0xC0 | 0 | 0 | 30 |
| **32** |  |  | **0x40** | **0xB2** | **0** | 0x20 | 0 | 0 | 0x20 | 0 | 0 | 31 |
| **33** | **MOV** | **1110** | **0x00** | **0x30** | **0** | 0x30 | 0 | 0 | 0x30 | 0 | 0 | 32 |
| **34** |  |  | **0x43** | **0x00** | **1** | 0x00 | 1 | 1 | 0x00 | 1 | 1 | 33 |

TESTBENCH (3 pgs)

`timescale 1ns / 1ps

module ALU\_tb();

logic [4:0] SEL; // input

logic [7:0] A, B; // input

logic CIN; // input

logic [7:0] RESULT = 0; // output

logic C = 0, Z = 0; // output

ALU ALU\_DUT(.\*);

initial begin // 34 Tests Total

// Test 1: ADD

SEL = 0; A = 8'hAA; B = 8'hAA; CIN = 0;

#1; // nanosecond

// Test 2:

SEL = 0; A = 8'h0A; B = 8'hA0; CIN = 1;

#1;

// Test 3:

SEL = 0; A = 8'hFF; B = 8'h01; CIN = 0;

#1;

// Test 4: ADDC

SEL = 1; A = 8'hC8; B = 8'h36; CIN = 1;

#1;

// Test 5:

SEL = 1; A = 8'hC8; B = 8'h64; CIN = 1;

#1;

// Test 6: SUB

SEL = 2; A = 8'hC8; B = 8'h64; CIN = 0;

#1;

// Test 7:

SEL = 2; A = 8'hC8; B = 8'h64; CIN = 1;

#1;

// Test 8:

SEL = 2; A = 8'h64; B = 8'hC8; CIN = 0;

#1;

// Test 9: SUBC

SEL = 3; A = 8'hC8; B = 8'h64; CIN = 0;

#1;

// Test 10:

SEL = 3; A = 8'hC8; B = 8'h64; CIN = 1;

#1;

// Test 11:

SEL = 3; A = 8'h64; B = 8'hC8; CIN = 0;

#1;

// Test 12:

SEL = 3; A = 8'h64; B = 8'hC8; CIN = 1;

#1;

// Test 13: CMP

SEL = 4; A = 8'hAA; B = 8'hFF; CIN = 0;

#1;

// Test 14:

SEL = 4; A = 8'hFF; B = 8'hAA; CIN = 0;

#1;

// Test 15:

SEL = 4; A = 8'hAA; B = 8'hAA; CIN = 0;

#1;

// Test 16: AND

SEL = 5; A = 8'hAA; B = 8'hAA; CIN = 0;

#1;

// Test 17:

SEL = 5; A = 8'h03; B = 8'hAA; CIN = 0;

#1

// Test 18: OR

SEL = 6; A = 8'hAA; B = 8'hAA; CIN = 0;

#1

// Test 19:

SEL = 6; A = 8'h03; B = 8'hAA; CIN = 0;

#1

// Test 20: EXOR

SEL = 7; A = 8'hAA; B = 8'hAA; CIN = 0;

#1

// Test 21:

SEL = 7; A = 8'h03; B = 8'hAA; CIN = 0;

#1

// Test 22: TEST

SEL = 8; A = 8'hAA; B = 8'hAA; CIN = 0;

#1

// Test 23:

SEL = 8; A = 8'h55; B = 8'hAA; CIN = 0;

#1

// Test 24: LSL

SEL = 9; A = 8'h01; B = 8'h12; CIN = 0;

#1

// Test 25: LSR

SEL = 10; A = 8'h80; B = 8'h33; CIN = 0;

#1

// Test 26:

SEL = 10; A = 8'h80; B = 8'h43; CIN = 1;

#1

// Test 27: ROL

SEL = 11; A = 8'h01; B = 8'hAB; CIN = 1;

#1

// Test 28:

SEL = 11; A = 8'hAA; B = 8'hF2; CIN = 0;

#1

// Test 29: ROR

SEL = 12; A = 8'h80; B = 8'h3C; CIN = 0;

#1

// Test 30:

SEL = 12; A = 8'h80; B = 8'h98; CIN = 1;

#1

// Test 31: ASR

SEL = 13; A = 8'h80; B = 8'h81; CIN = 0;

#1

// Test 32:

SEL = 13; A = 8'h40; B = 8'hB2; CIN = 0;

#1

// Test 33: MOV

SEL = 14; A = 8'h00; B = 8'h30; CIN = 0;

#1

// Test 34:

SEL = 14; A = 8'h43; B = 8'h00; CIN = 1;

#1

$finish;

end

endmodule

# Source Code

ALU

`timescale 1ns / 1ps

module ALU(

input [4:0] SEL,

input [7:0] A, B,

input CIN,

output logic [7:0] RESULT,

output logic C, Z

);

logic [8:0] nine\_bit;

always\_comb

begin

case(SEL)

0: nine\_bit = {1'b0,A} + {1'b0,B}; // ADD

1: nine\_bit = {1'b0,A} + {1'b0,B} + {8'b0,CIN}; // ADDC

2: nine\_bit = {1'b0,A} - {1'b0,B}; // SUB

3: nine\_bit = {1'b0,A} - {1'b0,B} - {8'b0,CIN}; // SUBC

4: nine\_bit = {1'b0,A} - {1'b0,B}; // CMP

5: nine\_bit = {1'b0,A} & {1'b0,B}; // AND

6: nine\_bit = {1'b0,A} | {1'b0,B}; // OR

7: nine\_bit = {1'b0,A} ^ {1'b0,B}; // EXOR

8: nine\_bit = {1'b0,A} & {1'b0,B}; // TEST

9: nine\_bit = {A[7:0], CIN}; // LSL

10: nine\_bit = {A[0], CIN, A[7:1]}; // LSR

11: nine\_bit = {A[7], A[6:0], A[7]}; // ROL

12: nine\_bit = {A[0], A[0], A[7:1]}; // ROR

13: nine\_bit = {A[0], A[7], A[7:1]}; // ASR

14: nine\_bit = {CIN, B[7:0]}; // MOV

15: nine\_bit = {CIN, B[7:0]}; // MOV

default: nine\_bit = {CIN, B[7:0]}; // MOV

endcase

end

always\_comb

begin

C = nine\_bit[8];

end

always\_comb

begin

if(nine\_bit[7:0] == 0)

Z = 1;

else Z = 0;

end

always\_comb

begin

RESULT = nine\_bit[7:0];

end

endmodule