

# Embedded Systems

## CS429 Lab4

Name: Dipean Dasgupta

ID:202151188

### Task1 Solution:

Module code:

```
primitive mux_udp(out,s0,s1,i0,i1,i2,i3);
output out;
input s0,s1;
input i0,i1,i2,i3;
table

//s0  s1   i0   i1   i2   i3 :  out;
  0    0   0    ?   ?    ? :  0;
  0    0   1    ?   ?    ? :  1;
  0    1   ?    0   ?    ? :  0;
  0    1   ?    1   ?    ? :  1;
  1    0   ?    ?   0    ? :  0;
  1    0   ?    ?   1    ? :  1;
  1    1   ?    ?   ?    0 :  0;
  1    1   ?    ?   ?    1 :  1;

endtable
endprimitive

module UDP_MUX(out,s0,s1,i0,i1,i2,i3);
output out;
input s0,s1;
input i0,i1,i2,i3;
mux_udp u1(out,s0,s1,i0,i1,i2,i3);
endmodule
```

### Testbench

```
module tb_mux_udp;

    // Inputs
    reg [0:0] i0,i1,i2,i3;
    reg [0:0] s0,s1;

    // Output
    wire out;
```

```

mux_udp uut (
    .i0(i0),
    .i1(i1),
    .i2(i2),
    .i3(i3),
    .s0(s0),
    .s1(s1),
    .out(out)
);
    reg clk = 0;
    always #5 clk = ~clk;

    // Stimulus
    initial begin
        $dumpfile("test_lab4_1.vcd");
        $dumpvars(0, tb_mux_udp);

        // Test case 1
        i0=1'b1;i1=1'b1;i2=1'b0;i3=1'b1;
        s0=1'b0;s1=1'b0;
        #10;

        $finish;
    end

    always @(posedge clk) begin
        $display("s0 = %b, s1 = %b, i0 = %b, i1 = %b, i2=%b, i3=%b, out=%b ", s0, s1,
i0, i1, i2, i3, out);
    end

endmodule

```

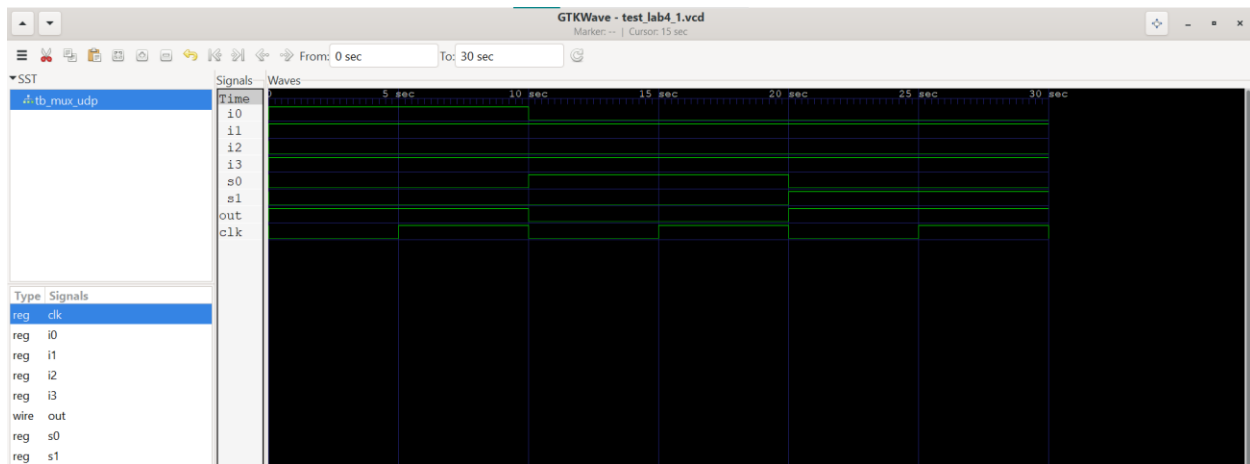
## OUTPUT:

```

[Running] lab4_1.v
VCD info: dumpfile test_lab4_1.vcd opened for output.
s0 = 0, s1 = 0, i0 = 1, i1 = 1, i2=0, i3=1, out=1
s0 = 1, s1 = 0, i0 = 0, i1 = 1, i2=0, i3=1, out=0
s0 = 0, s1 = 1, i0 = 0, i1 = 1, i2=0, i3=1, out=1
lab4_1.v:68: $finish called at 30 (1s)
[Done] exit with code=0 in 0.168 seconds

```

## Waveform:



## Task2 Solution:

### RTL Module code:

```
module Alu4b_rtl (
    input [3:0] A,
    input [3:0] B,
    input [2:0] ALUOp,
    output [3:0] Result
);

assign
Result = (ALUOp == 3'b000) ? A + B :
        (ALUOp == 3'b001) ? A - B :
        (ALUOp == 3'b010) ? (A < B) ? 4'b1 : 4'b0 :
        (ALUOp == 3'b011) ? (A == B) ? 4'b1 : 4'b0 :
        (ALUOp == 3'b100) ? (B[1:0] == 2'b00) ? A : (B[1:0] == 2'b01) ?
(A << 1) : (B[1:0] == 2'b10) ? (A << 2) : (A << 3) :
        (ALUOp == 3'b101) ? (B[1:0] == 2'b00) ? A : (B[1:0] == 2'b01) ?
(A >> 1) : (B[1:0] == 2'b10) ? (A >> 2) : (A >> 3) :
        (ALUOp == 3'b110) ? ~(A & B) :
        (ALUOp == 3'b111) ? ~(A | B) :
        4'bxxxx; // Unknown operation
endmodule
```

```
// Testbench Code
module Alu4b_rtl_tb();
reg[3:0]A,B;
reg[2:0]ALUOp;
wire[3:0]Result;

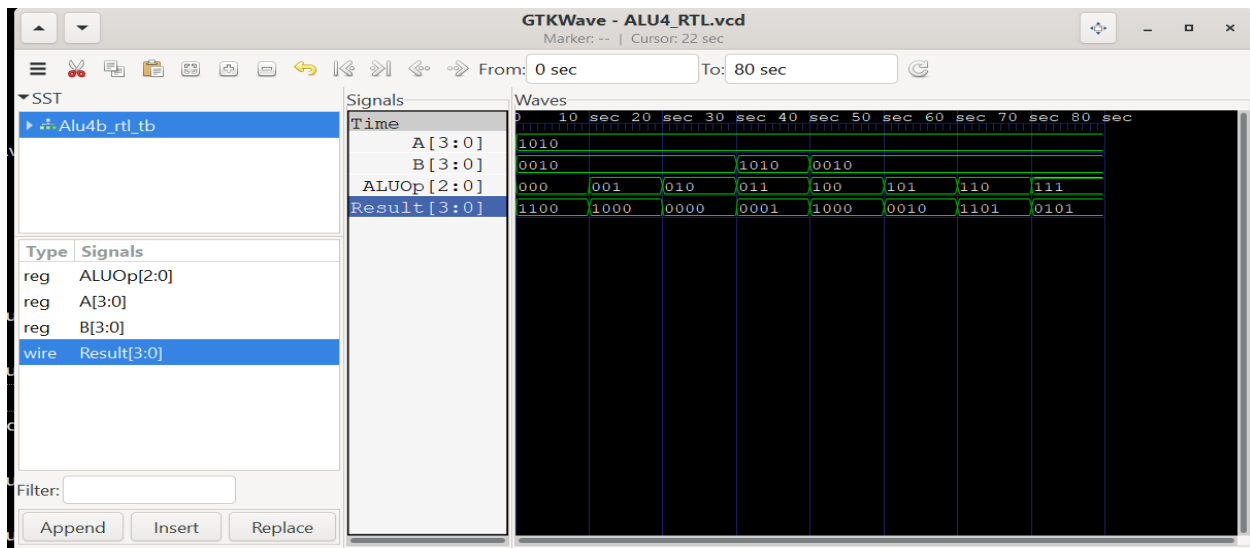
Alu4b_rtl a1(A,B,ALUOp,Result);
initial
begin
    $dumpfile("ALU4_RTL.vcd");
    $dumpvars(0,Alu4b_rtl_tb);
    ALUOp=3'b000;A=4'b1010;B=4'b0010;
    #10;
    ALUOp=3'b001;A=4'b1010;B=4'b0010;
    #10;
    ALUOp=3'b010;A=4'b1010;B=4'b0010;
    #10;
    ALUOp=3'b011;A=4'b1010;B=4'b1010;
    #10;
    ALUOp=3'b100;A=4'b1010;B=4'b0010;
    #10;
    ALUOp=3'b101;A=4'b1010;B=4'b0010;
    #10;
    ALUOp=3'b110;A=4'b1010;B=4'b0010;
    #10;
    ALUOp=3'b111;A=4'b1010;B=4'b0010;
    #10;
end

initial begin
    $monitor("ALUOp = %b, A = %b, B = %b, Result = %b",ALUOp,A,B,Result);
end
endmodule
```

**Result:**

```
[Running] lab4_2A.v
VCD info: dumpfile ALU4_RTL.vcd opened for output.
ALUOp = 000, A = 1010, B = 0010, Result = 1100
ALUOp = 001, A = 1010, B = 0010, Result = 1000
ALUOp = 010, A = 1010, B = 0010, Result = 0000
ALUOp = 011, A = 1010, B = 1010, Result = 0001
ALUOp = 100, A = 1010, B = 0010, Result = 1000
ALUOp = 101, A = 1010, B = 0010, Result = 0010
ALUOp = 110, A = 1010, B = 0010, Result = 1101
ALUOp = 111, A = 1010, B = 0010, Result = 0101
[Done] exit with code=0 in 0.32 seconds
```

## OUTPUT:



## Behavioural Module Code:

```
module alu_4bit (  
    input wire [3:0] a,  
    input wire [3:0] b,  
    input wire [2:0] alu_opcode,  
    output reg [3:0] result,  
    output reg zero_flag  
);  
  
always @(*)  
begin  
    case(alu_opcode)  
        3'b000: result = a + b;  
        3'b001: result = a>b? a-b : b-a;  
        3'b010: result = (a < b) ? 4'b1 : 4'b0;  
        3'b011: result = (a == b) ? 4'b1 : 4'b0;  
        3'b100: result = a << 1;  
        3'b101: result = a >> 1;  
        3'b110: result = ~(a & b);  
        3'b111: result = ~(a | b);  
        default: result = 4'b0; // Default  
    endcase  
    // Zero flag (set if result is zero)  
    zero_flag = (result == 4'b0) ? 1'b1 : 1'b0;  
end  
endmodule
```

```

// Testbench
module tb_alu_4bit;
    // I/p
    reg [3:0] a, b;
    reg [2:0] alu_opcode;
    //o/p
    wire [3:0] result;
    wire zero_flag;

    alu_4bit uut (
        .a(a),
        .b(b),
        .alu_opcode(alu_opcode),
        .result(result),
        .zero_flag(zero_flag)
    );

    initial begin
        $dumpfile("test_lab4_2.vcd");
        $dumpvars(0, tb_alu_4bit);
        //case 1: Addition
        a = 4'b1101;
        b = 4'b0011;
        alu_opcode = 3'b000;
        #10;
        //case 2: Subtraction
        a = 4'b1101;
        b = 4'b0011;
        alu_opcode = 3'b001;
        #10;
        //case 3: Less than comparison
        a = 4'b1101;
        b = 4'b1111;
        alu_opcode = 3'b010;
        #10;
        //case 4: Equal to comparison
        a = 4'b1101;
        b = 4'b1101;
        alu_opcode = 3'b011;
        #10;
        //case 5: Shift left
        a = 4'b1101;
        alu_opcode = 3'b100;
        #10;
    end

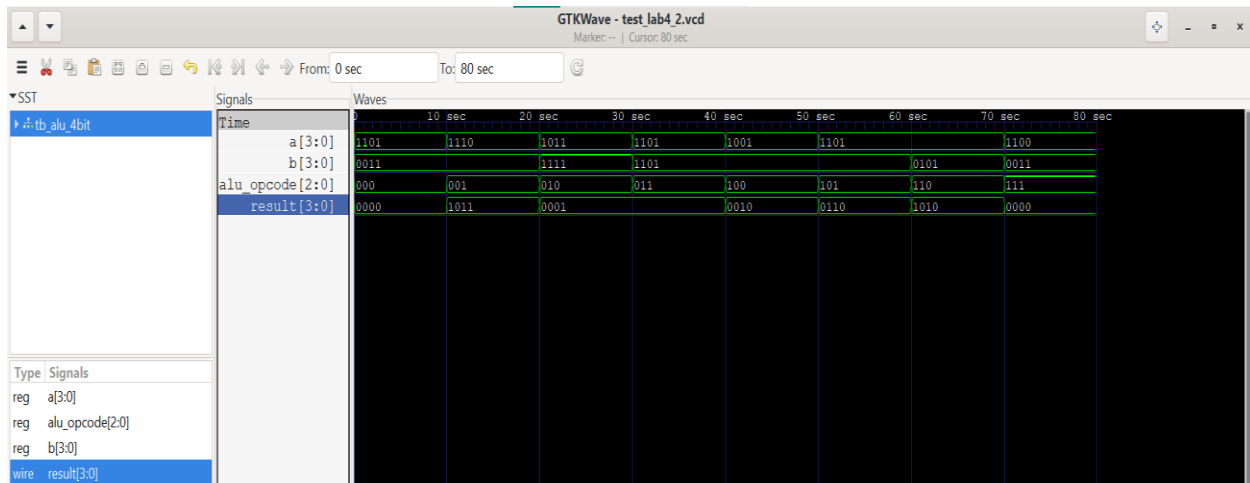
```

```

//case 6: Shift right
a = 4'b1101;
alu_opcode = 3'b101;
#10;
//case 7: NAND operation
a = 4'b1101;
b = 4'b0011;
alu_opcode = 3'b110;
#10;
//case 8: NOR operation
a = 4'b1101;
b = 4'b0011;
alu_opcode = 3'b111;
#10;
$finish;
end
endmodule

```

## OUTPUT:



## TASK3 Solution:

```
module full_adder (
    input wire a, b, cin,
    output wire sum, cout
);
    assign sum = a ^ b ^ cin;
    assign cout = (a & b) | (b & cin) | (a & cin);
endmodule

module ripple_carry_adder (
    input wire [3:0] a, b,
    output wire [3:0] sum,
    output wire cout
);

    wire [3:0] carry;
    full_adder fa0 (.a(a[0]), .b(b[0]), .cin(1'b0), .sum(sum[0]),
.cout(carry[0]));
    full_adder fa1 (.a(a[1]), .b(b[1]), .cin(carry[0]), .sum(sum[1]),
.cout(carry[1]));
    full_adder fa2 (.a(a[2]), .b(b[2]), .cin(carry[1]), .sum(sum[2]),
.cout(carry[2]));
    full_adder fa3 (.a(a[3]), .b(b[3]), .cin(carry[2]), .sum(sum[3]),
.cout(cout));

endmodule
```

## Testbench

```
// Testbench
module tb_ripple_carry_adder;
    // I/P
    reg [3:0] a, b;
    // O/P
    wire [3:0] sum;
    wire cout;

    // Instantiation
    ripple_carry_adder uut (
        .a(a),
        .b(b),
        .sum(sum),
        .cout(cout)
    );
endmodule
```

```

initial begin
$dumpfile("test_lab4_3.vcd");
$dumpvars(0, tb_ripple_carry_adder);
    //case 1
    a = 4'b0101;
    b = 4'b1100;
    #10;

    //case 2
    a = 4'b1010;
    b = 4'b0111;
    #10;

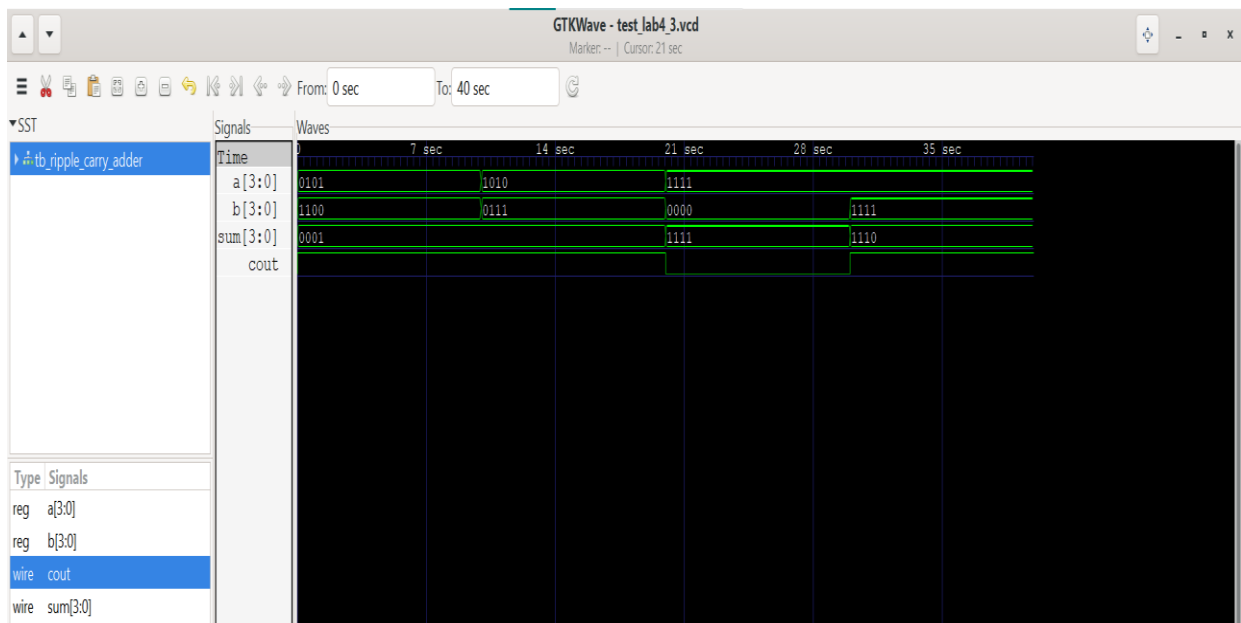
    //case 3
    a = 4'b1111;
    b = 4'b0000;
    #10;

    //case4
    a = 4'b1111;
    b = 4'b1111;
    #10;

    $finish;
end
endmodule

```

**OUTPUT:**



## TASK4 Solution:

Module code:

```
module full_adder (
    input wire a, b, cin,
    output wire sum, cout
);
    assign sum = a ^ b ^ cin;
    assign cout = (a & b) | (b & cin) | (a & cin);
endmodule

module ripple_carry_adder (
    input wire [3:0] a, b,
    output wire [3:0] sum,
    output wire cout
);

    wire [3:0] carry;
    full_adder fa0 (.a(a[0]), .b(b[0]), .cin(1'b0), .sum(sum[0]),
.cout(carry[0]));
    full_adder fa1 (.a(a[1]), .b(b[1]), .cin(carry[0]), .sum(sum[1]),
.cout(carry[1]));
    full_adder fa2 (.a(a[2]), .b(b[2]), .cin(carry[1]), .sum(sum[2]),
.cout(carry[2]));
    full_adder fa3 (.a(a[3]), .b(b[3]), .cin(carry[2]), .sum(sum[3]),
.cout(cout));

endmodule

// 2-input 4-bit BCD Adder Module
module bcd_adder_4bit (
    input wire [3:0] a, b,
    output reg [3:0] sum,
    output reg overflow
);

    wire [3:0] temp_sum;
    wire cout;

    ripple_carry_adder u_adder (.a(a), .b(b), .sum(temp_sum), .cout(cout));

    always @(*)
    begin
```

```

        // BCD addition condition: sum < 10
        if (temp_sum >= 4'b1010) begin
            overflow = 1'b1;
        end
        else begin
            overflow = 1'b0;
        end

        // BCD correction: add 6 to 4-bit binary sum if > 9
        if (temp_sum >= 4'b1010) begin
            sum = temp_sum + 4'b0110;
        end
        else begin
            sum = temp_sum;
        end
    end
end
endmodule

```

## Testbench;

```

module tb_bcd_adder_4bit;

    // I/P
    reg [3:0] a, b;

    // O/P
    wire [3:0] sum;
    wire overflow;

    bcd_adder_4bit uut (
        .a(a),
        .b(b),
        .sum(sum),
        .overflow(overflow)
    );

    initial begin
        $dumpfile("test_lab4_4.vcd");
        $dumpvars(0, tb_bcd_adder_4bit);
        //case 1
        a = 4'b1001; // 9 in BCD
        b = 4'b0001; // 1 in BCD
        #10;
        //case 2
    end
endmodule

```

```

a = 4'b1100; // 12 in BCD
b = 4'b0110; // 6 in BCD
#10;

//case 3
a = 4'b0101; // 5 in BCD
b = 4'b0101; // 5 in BCD
#10;

//case 4
a = 4'b1111; // 15 in BCD
b = 4'b0001; // 1 in BCD
#10;
$finish;

end
endmodule

```

## OUTPUT:

