# ES204 Digital Systems
# LAB Assignment - 2

Indian Institute of Technology, Gandhinagar
January 18, 2024

**Submission deadline: Jan 21, 2024**
**Marks : 40**

**Submission instructions:**

**(a) Only one student from the team will submit with the word doc name Rollno1_Rollno2.pdf. The PDF will contain the code, testbench and simulation results.**
**(b) Make a tar-ball / Zip of the project and upload.**

For each of the questions, write a Verilog code. You also need to create a **testbench** and show the simulation results.

1 (a) Design an 8-bit Combined Adder/Subtractor **using Full-adder modules**. A mode-bit selects whether the operation to be performed is addition or subtraction.

Example: Mode=0 Perform Add 56+32 = 88
Mode=1 Perform Subtract 56-32 = 24

The module has two 8-bit inputs (A, B) and carry_borrow_in for giving the data;
It has a Mode input to select between Add/Sub

It has one 8-bit output C and carry_borrow_out;

1 (b) After you have designed the combined adder/subtractor, now attach a "**Gray-converter**" module that converts the (9-bit) output of the combined adder/subtractor to Gray-code. Use structural approach to connect the two modules (combined adder/subtractor and Gray-code converter).

# Lab 2 - Assignment ES 204

Member 1: Birudugadda Srivibhav (22110050)
Member 2: Reddybathuni Venkat (22110220)

## # Implementation of 8-bit Combined Adder/Subtractor

### 1. Code:

```
`timescale 1ns / 1ps

module adder(
    input a, b, cin,
    output OUT, cout
);
    // sum is equal to xor(a, b, cin)
    xor G1 (OUT, a, b, cin);
    //carry is majority function
    and G2 (o1, a, b);
    and G3 (o2, b, cin);
    and G4 (o3, cin, a);
    or G5 (cout, o1, o2, o3);
endmodule

module add_or_sub_8bit(
    input Mode, carry_borrow_in,
    input [7:0] A, B,
    output [7:0] C,
    output reg carry_borrow_out
    );
    wire [7:0] B0, Carry;
    //finding 2's complement of B in case the mode = 1
    adder inst00 (.a(~B[0]), .b(1), .cin(0), .OUT(B0[0]), .cout(Carry[0]));
    adder inst01 (.a(~B[1]), .b(0), .cin(Carry[0]), .OUT(B0[1]), .cout(Carry[1]));
    adder inst02 (.a(~B[2]), .b(0), .cin(Carry[1]), .OUT(B0[2]), .cout(Carry[2]));
    adder inst03 (.a(~B[3]), .b(0), .cin(Carry[2]), .OUT(B0[3]), .cout(Carry[3]));
    adder inst04 (.a(~B[4]), .b(0), .cin(Carry[3]), .OUT(B0[4]), .cout(Carry[4]));
    adder inst05 (.a(~B[5]), .b(0), .cin(Carry[4]), .OUT(B0[5]), .cout(Carry[5]));
    adder inst06 (.a(~B[6]), .b(0), .cin(Carry[5]), .OUT(B0[6]), .cout(Carry[6]));
    adder inst07 (.a(~B[7]), .b(0), .cin(Carry[6]), .OUT(B0[7]), .cout(Carry[7]));

    reg [7:0] B_final;
    always @* begin
        //Finalizing the B value based on Mode value
        // B_final = B if ,ode = 0 and B_final = 2's complement of B
        if(Mode == 0)
        begin
        B_final = B;
        end
```

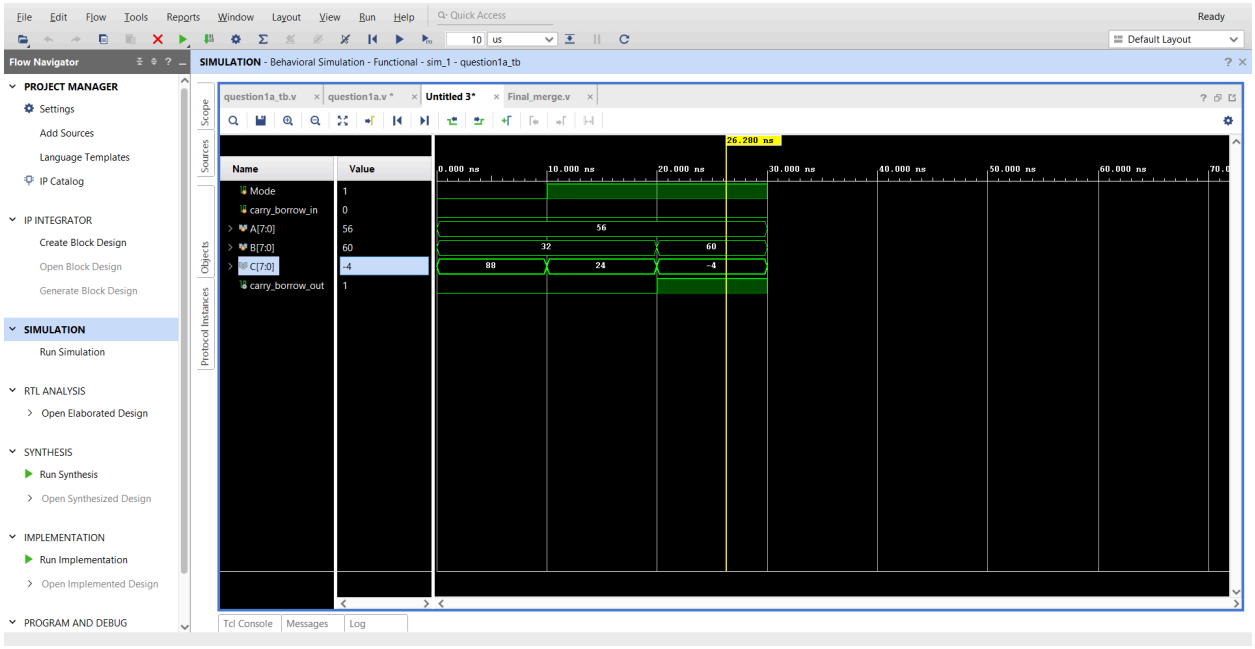
```
        else
        begin
        B_final = B0;
        end
    end
    //Applying 8 bit addition function
    wire [7:0] Carry1;
    adder inst0 (.a(A[0]), .b(B_final[0]), .cin(carry_borrow_in), .OUT(C[0]), .cout(Carry1[0]));
    adder inst1 (.a(A[1]), .b(B_final[1]), .cin(Carry1[0]), .OUT(C[1]), .cout(Carry1[1]));
    adder inst2 (.a(A[2]), .b(B_final[2]), .cin(Carry1[1]), .OUT(C[2]), .cout(Carry1[2]));
    adder inst3 (.a(A[3]), .b(B_final[3]), .cin(Carry1[2]), .OUT(C[3]), .cout(Carry1[3]));
    adder inst4 (.a(A[4]), .b(B_final[4]), .cin(Carry1[3]), .OUT(C[4]), .cout(Carry1[4]));
    adder inst5 (.a(A[5]), .b(B_final[5]), .cin(Carry1[4]), .OUT(C[5]), .cout(Carry1[5]));
    adder inst6 (.a(A[6]), .b(B_final[6]), .cin(Carry1[5]), .OUT(C[6]), .cout(Carry1[6]));
    adder inst7 (.a(A[7]), .b(B_final[7]), .cin(Carry1[6]), .OUT(C[7]), .cout(Carry1[7]));
    // Carry_borrow_out assignement as there will be a negation caused by 2's complement when
we are using 2's compelement
    always @* begin
    if(Mode == 1 & B != 8'b00000)
    begin
    carry_borrow_out = ~Carry1[7];
    end
    else
    begin
    carry_borrow_out = Carry1[7];
    end
    end
endmodule
```

## 2. Test Bench:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 19.01.2024 22:06:46
// Design Name:
// Module Name: question1a_tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
```

```
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module question1a_tb();
    reg Mode;
    reg carry_borrow_in;
    reg [7:0] A;
    reg [7:0] B;
    wire [7:0] C;
    wire carry_borrow_out;

    add_or_sub_8bit uut (
        .Mode(Mode),
        .carry_borrow_in(carry_borrow_in),
        .A(A),
        .B(B),
        .C(C),
        .carry_borrow_out(carry_borrow_out)
    );

    initial begin
        Mode = 0;
        carry_borrow_in = 0;
        A = 8'b00111000;
        B = 8'b00100000;
        #10;

        Mode = 1;
        carry_borrow_in = 0;
        A = 8'b00111000;
        B = 8'b00100000;
        #10;

        Mode = 1;
        carry_borrow_in = 0;
        A = 8'b00111000;
        B = 8'b00111100;
        #10;
        $finish();
    end
endmodule
```

## 3. Simulation Results:




# # Implementation of Gray Code Converter

## 1. Code

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 20.01.2024 10:14:46
// Design Name:
// Module Name: question1b
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module gray_converter(
    input [7:0] C,
    input carry_borrow_out,
    output [8:0] C_gray
    );
//    assign C_gray[8] = carry_borrow_out;
    and(C_gray[8], carry_borrow_out, 1);
```

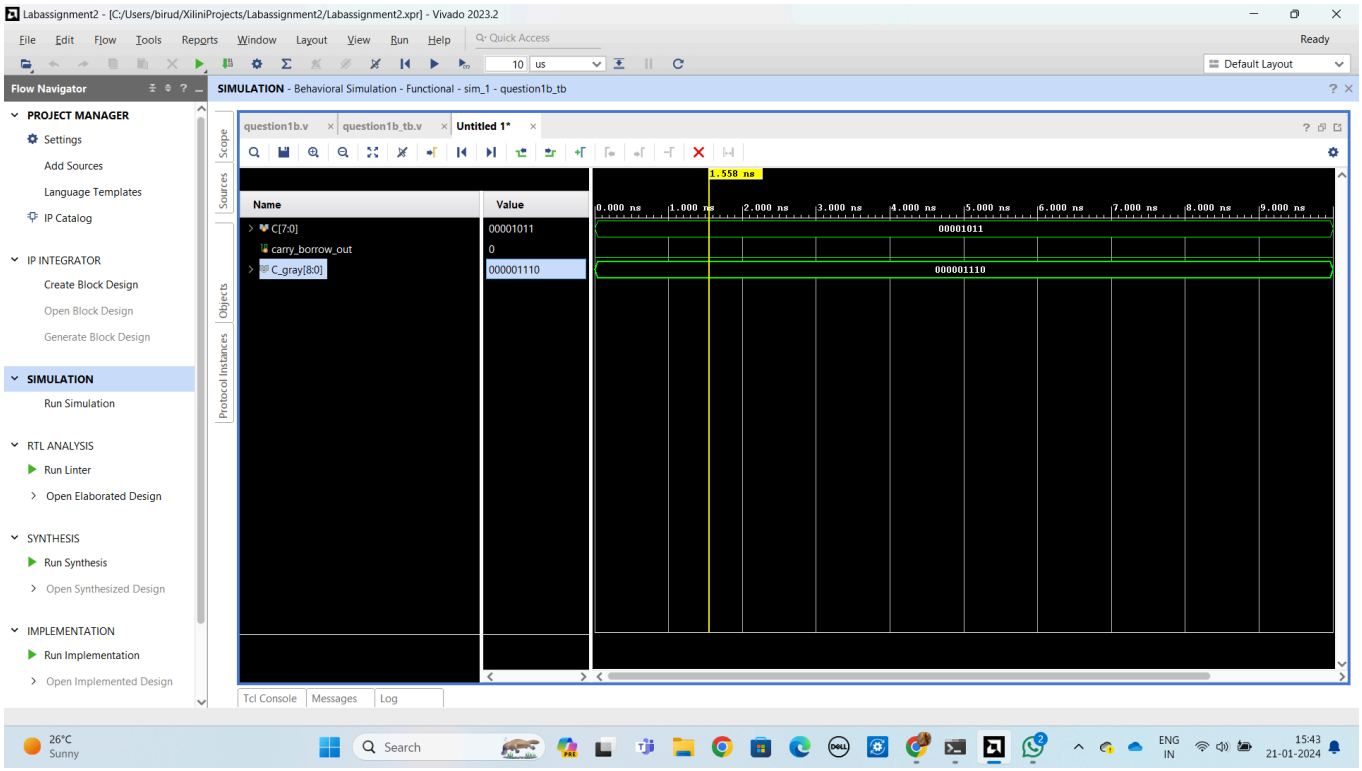
```
    xor g1(C_gray[7],carry_borrow_out, C[7]);
    xor g2(C_gray[6],C[6], C[7]);
    xor g3(C_gray[5],C[5], C[6]);
    xor g4(C_gray[4],C[4], C[5]);
    xor g5(C_gray[3],C[3], C[4]);
    xor g6(C_gray[2],C[2], C[3]);
    xor g7(C_gray[1],C[1], C[2]);
    xor g8(C_gray[0],C[0], C[1]);
endmodule
```

## 2. Test Bench:

We have checked for binary input: 00001011; the corresponding gray code is 000001110

```
`timescale 1ns / 1ps

module question1b_tb();
    reg [7:0] C;
    reg carry_borrow_out;
    wire [8:0] C_gray;
    gray_converter uut(.C(C), .carry_borrow_out(carry_borrow_out), .C_gray(C_gray));
    initial begin
        C = 8'b1011; carry_borrow_out = 0;
        #10;
        $finish();
    end
endmodule
```

## 3. Simulation Results:




# # Final Implementation of 8-bit Combined Adder/Subtractor with Gray Code Converter

## 1. Code

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 20.01.2024 10:37:07
```

```
// Design Name:
// Module Name: Final_merge
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module gray_converter(
    input [7:0] C,
    input carry_borrow_out,
    output [8:0] C_gray
    );
//    assign C_gray[8] = carry_borrow_out;
    and(C_gray[8], carry_borrow_out, 1);
    xor g1(C_gray[7],carry_borrow_out, C[7]);
    xor g2(C_gray[6],C[6], C[7]);
    xor g3(C_gray[5],C[5], C[6]);
    xor g4(C_gray[4],C[4], C[5]);
    xor g5(C_gray[3],C[3], C[4]);
    xor g6(C_gray[2],C[2], C[3]);
    xor g7(C_gray[1],C[1], C[2]);
    xor g8(C_gray[0],C[0], C[1]);
endmodule
module adder(
    input a, b, cin,
    output OUT, cout
);
    // sum is equal to xor(a, b, cin)
    xor G1 (OUT, a, b, cin);
    //carry is majority function
    and G2 (o1, a, b);
    and G3 (o2, b, cin);
    and G4 (o3, cin, a);
    or G5 (cout, o1, o2, o3);
endmodule
module Final_merge(
    input Mode, carry_borrow_in,
    input [7:0] A, B,
    output [8:0] C_gray,
    output reg carry_borrow_out
    );
    wire [7:0] B0, Carry;
//finding 2's complement of B in case the mode = 1
```

```
    adder inst00 (.a(~B[0]), .b(1), .cin(0), .OUT(B0[0]), .cout(Carry[0]));
    adder inst01 (.a(~B[1]), .b(0), .cin(Carry[0]), .OUT(B0[1]), .cout(Carry[1]));
    adder inst02 (.a(~B[2]), .b(0), .cin(Carry[1]), .OUT(B0[2]), .cout(Carry[2]));
    adder inst03 (.a(~B[3]), .b(0), .cin(Carry[2]), .OUT(B0[3]), .cout(Carry[3]));
    adder inst04 (.a(~B[4]), .b(0), .cin(Carry[3]), .OUT(B0[4]), .cout(Carry[4]));
    adder inst05 (.a(~B[5]), .b(0), .cin(Carry[4]), .OUT(B0[5]), .cout(Carry[5]));
    adder inst06 (.a(~B[6]), .b(0), .cin(Carry[5]), .OUT(B0[6]), .cout(Carry[6]));
    adder inst07 (.a(~B[7]), .b(0), .cin(Carry[6]), .OUT(B0[7]), .cout(Carry[7]));

    reg [7:0] B_final;
    always @* begin
        //Finalizing the B value based on Mode value
        // B_final = B if ,ode = 0 and B_final = 2's complement of B
        if(Mode == 0)
        begin
        B_final = B;
        end
        else
        begin
        B_final = B0;
        end
    end
        //Applying 8 bit addition function
    wire [7:0] C;
    wire [7:0] Carry1;
    adder inst0 (.a(A[0]), .b(B_final[0]), .cin(carry_borrow_in), .OUT(C[0]), .cout(Carry1[0]));
    adder inst1 (.a(A[1]), .b(B_final[1]), .cin(Carry1[0]), .OUT(C[1]), .cout(Carry1[1]));
    adder inst2 (.a(A[2]), .b(B_final[2]), .cin(Carry1[1]), .OUT(C[2]), .cout(Carry1[2]));
    adder inst3 (.a(A[3]), .b(B_final[3]), .cin(Carry1[2]), .OUT(C[3]), .cout(Carry1[3]));
    adder inst4 (.a(A[4]), .b(B_final[4]), .cin(Carry1[3]), .OUT(C[4]), .cout(Carry1[4]));
    adder inst5 (.a(A[5]), .b(B_final[5]), .cin(Carry1[4]), .OUT(C[5]), .cout(Carry1[5]));
    adder inst6 (.a(A[6]), .b(B_final[6]), .cin(Carry1[5]), .OUT(C[6]), .cout(Carry1[6]));
    adder inst7 (.a(A[7]), .b(B_final[7]), .cin(Carry1[6]), .OUT(C[7]), .cout(Carry1[7]));
    // Carry_borrow_out assignement as there will be a negation caused by 2's complement when
we are using 2's compelement
    always @* begin
        if(Mode == 1)
        begin
        carry_borrow_out = ~Carry1[7];
        end
        else
        begin
        carry_borrow_out = Carry1[7];
        end
        end
    //Applying gray converter
    gray_converter inst8(.C(C), .carry_borrow_out(carry_borrow_out), .C_gray(C_gray));
endmodule
```

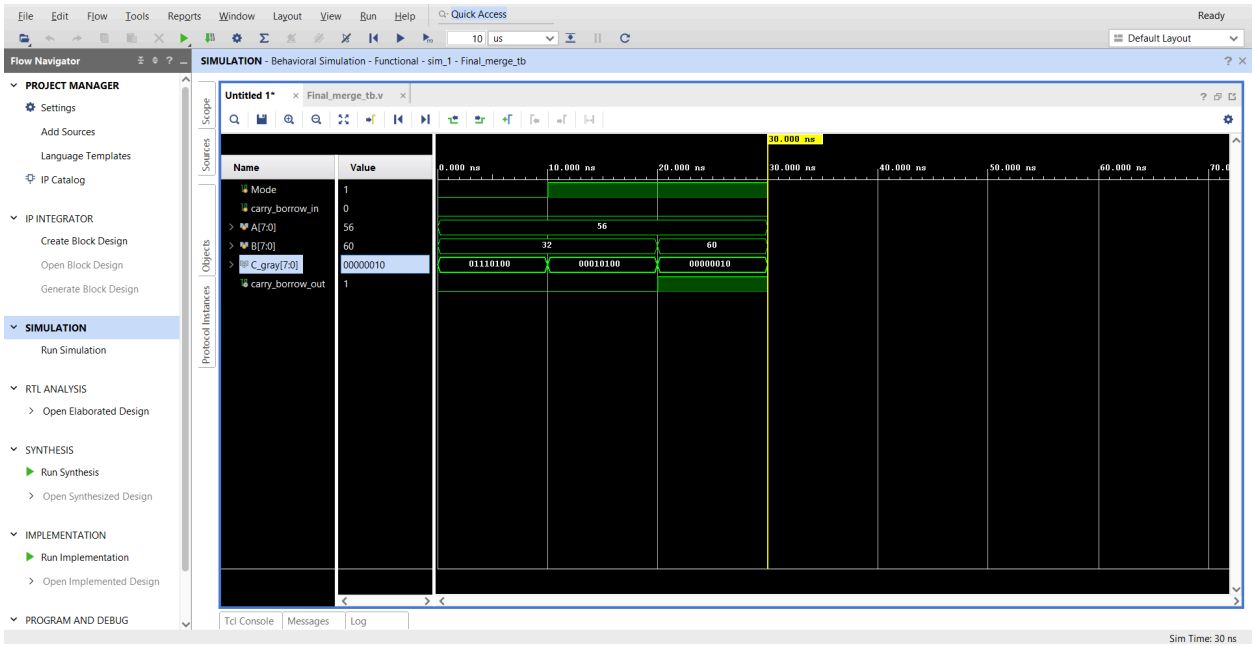
## 2. Test Bench

```
`timescale 1ns / 1ps

module Final_merge_tb();
    reg Mode;
    reg carry_borrow_in;
    reg [7:0] A;
    reg [7:0] B;
    wire [7:0] C_gray;
    wire carry_borrow_out;

    Final_merge uut (
        .Mode(Mode),
        .carry_borrow_in(carry_borrow_in),
        .A(A),
        .B(B),
        .C_gray(C_gray),
        .carry_borrow_out(carry_borrow_out)
    );

    initial begin
        Mode = 0;
        carry_borrow_in = 0;
        A = 8'b00111000;
        B = 8'b00100000;
        #10;

        Mode = 1;
        carry_borrow_in = 0;
        A = 8'b00111000;
        B = 8'b00100000;
        #10;

        Mode = 1;
        carry_borrow_in = 0;
        A = 8'b00111000;
        B = 8'b00111100;
        #10;
        $finish();
    end
endmodule
```

## 3. Simulation Results:

The test bench is written for 56 + 32 = 88 and 56 - 32 = 24

Binary code of 88 = 1011000, Gray Code of 88 = 1110100
Binary code of 24 = 11000, Gray Code of 24 = 10100


File Edit Flow Tools Reports Window Layout View Run Help
Quick Access
Ready
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
SIMULATION - Behavioral Simulation - Functional - sim_1 - Final_merge_tb
Untitled 1*
Final_merge_tb.v
Scope
Sources
Objects
Protocol Instances
Name
Value
Mode
1
carry_borrow_in
0
A[7:0]
56
B[7:0]
60
C_gray[7:0]
00000010
carry_borrow_out
1
30.000 ns
0.000 ns
10.000 ns
20.000 ns
30.000 ns
40.000 ns
50.000 ns
60.000 ns
56
32
60
01110100
00010100
00000010
Tcl Console
Messages
Log
Sim Time: 30 ns