भारतीय प्रौद्योगिकी संस्थान गांधीनगर
पालज, गांधीनगर, गुजरात 382 055
INDIAN INSTITUTE OF TECHNOLOGY GANDHINAGAR
PALAJ, GANDHINAGAR, GUJARAT 382 055

Name : Bhavik Patel
Roll no. : 22110047

IITGN

Digital Systems
LAB Assignment-2

Question 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

One Bit Adder

It takes two 1-bit numbers and a carry and returns carry and 1 bit sum

```
`timescale 1ns / 1ps

module One_bit_adder(A, B, cin, cout , sum);
input A,B,cin;
output cout,sum;

    xor(sum, A, B, cin);
    and(f,A,B);
    and(g,A,cin);
    and(h,cin,B);
    or(cout,f,g,h);

endmodule
```

Two’s Complement Module

It takes a number and returns its it 2’s complement of the number.

```
1  `timescale 1ns / 1ps
2
3  module two_s_complement(A,out);
4  input [8:0]A;
5  output [8:0]out;
6  wire c[8:0];
7  One_bit_adder inst21(~A[0],1'b1,0,c[0],out[0]);
8  One_bit_adder inst22(~A[1],0,c[0],c[1],out[1]);
9  One_bit_adder inst23(~A[2],0,c[1],c[2],out[2]);
10 One_bit_adder inst24(~A[3],0,c[2],c[3],out[3]);
11 One_bit_adder inst25(~A[4],0,c[3],c[4],out[4]);
12 One_bit_adder inst26(~A[5],0,c[4],c[5],out[5]);
13 One_bit_adder inst27(~A[6],0,c[5],c[6],out[6]);
14 One_bit_adder inst28(~A[7],0,c[6],c[7],out[7]);
15 One_bit_adder inst29(~A[8],0,c[7],c[8],out[8]);
16
17 endmodule
18
```

Full parallel 8-bit Adder and Subtractor

```
`timescale 1ns / 1ps

module full_add_sub_8bit(A, B, mode, cout, sum);
input [7:0]A,B;
input mode;
output [7:0]sum;
output cout;
wire [8:0]c;
//wire[8:0]Anew;
wire[8:0]Bnew;
//assign Anew= {1'b0,A};
wire [8:0]Bcomp;
//assign Bcomp = 9'b0;
two_s_complement insta({1'b0,B},Bcomp);
assign Bnew= (mode==1'b1)? Bcomp:{1'b0,B};

One_bit_adder inst1(A[0],Bnew[0],0,c[0],sum[0]);
One_bit_adder inst2(A[1],Bnew[1],c[0],c[1],sum[1]);
One_bit_adder inst3(A[2],Bnew[2],c[1],c[2],sum[2]);
One_bit_adder inst4(A[3],Bnew[3],c[2],c[3],sum[3]);
One_bit_adder inst5(A[4],Bnew[4],c[3],c[4],sum[4]);
One_bit_adder inst6(A[5],Bnew[5],c[4],c[5],sum[5]);
One_bit_adder inst7(A[6],Bnew[6],c[5],c[6],sum[6]);
One_bit_adder inst8(A[7],Bnew[7],c[6],c[7],sum[7]);
One_bit_adder inst9(0,Bnew[8],c[7],c[8],cout);

endmodule
```

Test Bench

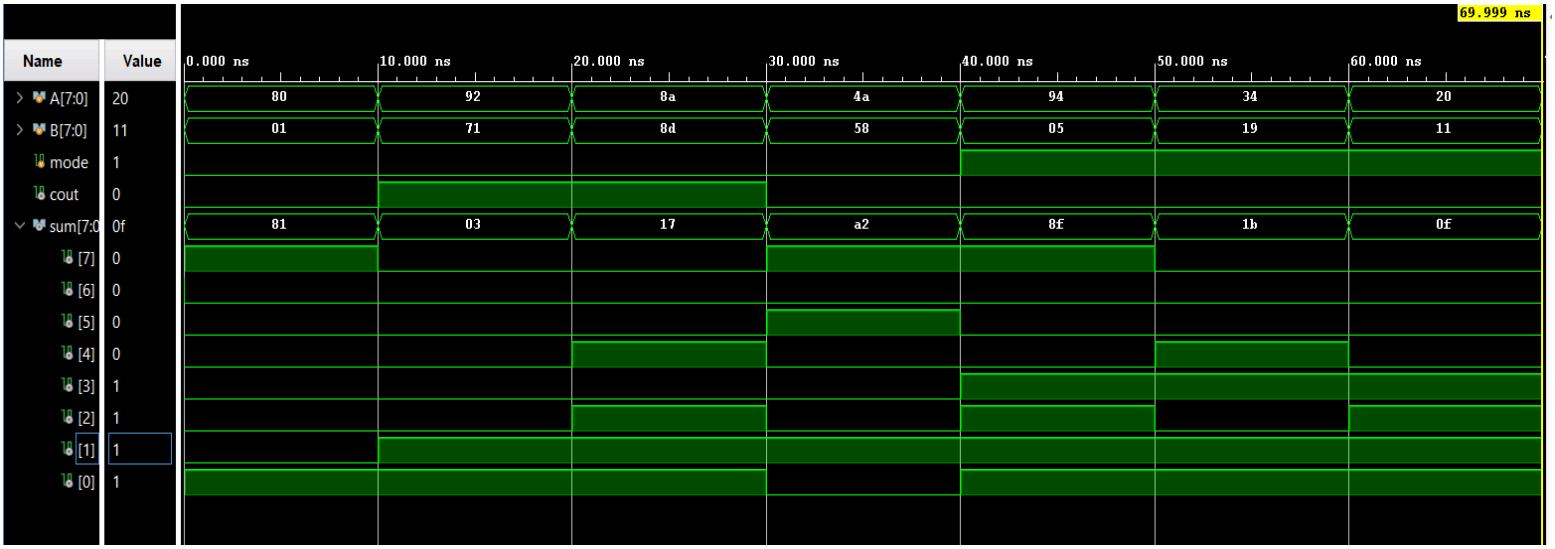
```
`timescale 1ns / 1ps
module test_bench();
reg [7:0]A,B;
reg mode;
wire cout;
wire [7:0]sum;

full_add_sub_8bit uut(A,B,mode,cout,sum);

initial
begin
mode=0;
A=8'b10000000;      B=8'b00000001;  #10;
A=8'b10010010;      B=8'b01110001;  #10;
A=8'b10001010;      B=8'b10001101;  #10;
A=8'b01001010;      B=8'b01011000;  #10;
mode=1'b1;
A=8'b10010100;      B=8'b00000101;  #10;
A=8'b00110100;      B=8'b00011001;  #10;
A=8'b00100000;      B=8'b00010001;  #10;


$finish();
end
endmodule
```

Simulation Result



Question1)(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 a structural approach to connect the two modules (combined adder/subtractor and Gray-code converter).

Gray Converter

```
`timescale 1ns / 1ps
module Grey_Code(A, B, mode, grayCode);
input [7:0]A,B;
input mode;
output [8:0]grayCode;

wire cout;
wire [7:0]sum;

full_add_sub_8bit uut0(A,B,mode,cout,sum);


and(grayCode[8],1'b1,cout);
xor(grayCode[7],cout,sum[7]);
xor(grayCode[6],sum[7],sum[6]);
xor(grayCode[5],sum[6],sum[5]);
xor(grayCode[4],sum[5],sum[4]);
xor(grayCode[3],sum[4],sum[3]);
xor(grayCode[2],sum[3],sum[2]);
xor(grayCode[1],sum[2],sum[1]);
xor(grayCode[0],sum[1],sum[0]);

endmodule
```

Test Bench

```
`timescale 1ns / 1ps
module test_bench();
reg [7:0]A,B;
reg mode;
wire [8:0]grayCode;

Grey_Code uut(A, B, mode, grayCode);
initial
begin
mode=0;
A=8'b10000000;      B=8'b00000001;  #10;
A=8'b10010010;      B=8'b01110001;  #10;
A=8'b10001010;      B=8'b10001101;  #10;
A=8'b01001010;      B=8'b01011000;  #10;
mode=1'b1;
A=8'b10010100;      B=8'b00000101;  #10;
A=8'b00110100;      B=8'b00011001;  #10;
A=8'b00100000;      B=8'b00010001;  #10;
$finish();
end
endmodule
```

Simulation Result