# Lab 07 – Worksheet

| Name: Syed Asghar Abbas Zaidi | ID:07201 | Section:T6 |
| --- | --- | --- |

## Selective 2’s complement

| **Inputs to complement** | | | | **Outputs of the complement** | | |
| --- | --- | --- | --- | --- | --- | --- |
| **O** | **B2** | **B1** | **B0** | **C2** | **C1** | **C0** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 |

Write expressions for C2, C1 and C0

Table 7.2

| C2 = | (((~O)&B[2]) | (B[2] & (~B[1])&(~B[0])) | (O&(~[2])&B[0]) | (O&(~[2])&B[1])); |
| --- | --- |
| C1 = | (((~O)&B[1]) | (B[1] & (~B[0])) | (O & (~B[1])&B[0])); |
| C0 = | B[0]; |

*Provide appropriately commented code for your selective 2’s complement module*

| `timescale 1ns / 1ps  module SecondCompliment(  input O,  input [2:0] B,  output [2:0] C  );  assign C[2] = (((~O) & B[2]) | (B[2] & (~B[1]) & (~B[0])) | (O & (~B[2]) & B[0] | O & (~B[2]) & B[1]));  assign C[1] = (((~O)&B[1]) | (B[1] & (~B[0])) | (O & (~B[1])&B[0]));  assign C[0] = B[0];  endmodule |
| --- |

Write a testbench to thoroughly test designed complement module.

| `timescale 1ns / 1ps  module testbench\_AND();  reg O;  reg [2:0] B;  wire [2:0] C;  SecondCompliment module\_u\_test (O, B,C);  initial begin  #100 O = 1'b0;  #100 B = 3'b001;  #100 O = 1'b0;  #100 B = 3'b101;  #100 O = 1'b1;  #100 B = 3'b110;  #100 O = 1'b0;  #100 B = 3'b010;  end  endmodule |
| --- |

*Attach screenshot of Simulation output- make sure to scale properly for visibility of all case.*

|  |
| --- |

## Adder/Subtractor system

*Provide code for design module here*

| `timescale 1ns / 1ps  module Adder\_Subtractor\_3bit(  input [2:0] B,  input [2:0] A,  input O,  output [6:0] S,  output dp  );  wire [2:0] C;  SecondCompliment First\_Second (O, B[2:0], C[2:0]);  wire [3:0] U;  ThreeBitFullAdder TBFA( A [2:0], C [2:0], U [2:0], U[3]);  //underneath stuff.  wire Y, Y2, Y3, Y4;  not (Y, O);  //or-stuff  or (Y2,U[3],Y);  not(Y3,Y2);//Y3 selecor bit for 2nd second compliment  wire [3:0] D;  and (Y4, U[3],Y);  assign D[3] = Y4;  //Y4 will go into 7 segment  SecondCompliment Second\_Second (Y3, U[2:0], D[2:0]);  SevenSegment SS (D[3:0] , S[6:0]);  assign dp = Y2;  endmodule  module SevenSegment(  input [3:0] D,  output [6:0] S);    assign S[0] = ((~D[3])&(~D[2])&(~D[1])&D[0]) | ((~D[3])&(D[2])&(~D[1])&(~D[0])) | ((D[3])&(~D[2])&(D[1])&D[0]) | ((D[3])&(D[2])&(D[1])&D[0]);  assign S[1] = ((D[2])&(D[1])&(~D[0])) | ((D[3])&(D[1])&(D[0])) | ((D[3])&(D[2])&(~D[0])) | ((~D[3])&(D[2])&(~D[1])&D[0]);  assign S[2] = ((D[3])&(D[2])&(~D[0])) | ((D[3])&(D[2])&(D[1])) | ((~D[3])&(~D[2])&(D[1])&(~D[0]));  assign S[3] = ((D[2])&(D[1])&D[0]) | ((~D[3])&(~D[2])&(~D[1])&(D[0])) | ((~D[3])&(D[2])&(~D[1])&(~D[0])) | ((D[3])&(~D[2])&(D[1])&(~D[0]));  assign S[4] = ((~D[3])&D[0]) | ((~D[2])&(~D[1])&(D[0])) | ((~D[3])&(D[2])&(~D[1]));  assign S[5] = ((~D[3])&(~D[2])&D[0]) | ((~D[3])&(~D[2])&(D[1])) | ((~D[3])&(D[1])&D[0]) | ((D[3])&(D[2])&(~D[1])&D[0]);  assign S[6] = ((~D[3])&(~D[2])&(~D[1])) | ((~D[3])&(D[2])&(D[1])&(D[0])) | ((D[3])&(D[2])&(~D[1])&(~D[0]));    endmodule  module SecondCompliment(  input O,  input [2:0] B,  output [2:0] C  );  assign C[2] = (((~O) & B[2]) | (B[2] & (~B[1]) & (~B[0])) | (O & (~B[2]) & B[0] | O & (~B[2]) & B[1]));  assign C[1] = (((~O)&B[1]) | (B[1] & (~B[0])) | (O & (~B[1])&B[0]));  assign C[0] = B[0];  endmodule  module Adder(  input a,  input b,  input c,  output sum,  output carry  );    wire x1,x2,x3;  xor(x1,a,b);  xor g2(sum,c,x1);  and(x2,c,x1);  and(x3,a,b);  or(carry,x2,x3);  endmodule  module ThreeBitFullAdder(  input [2:0] A,  input [2:0] B,  output [2:0] Y,  output Cout  );  wire [1:0] w1;  Adder add1(A[0],B[0],0,Y[0],w1[0]);  Adder add2(A[1],B[1],w1[0],Y[1],w1[1]);  Adder add3(A[2],B[2],w1[1],Y[2],Cout);  endmodule |
| --- |

*Add your testbench here*

| *`timescale 1ns / 1ps*  *module Simulation();*  *reg [2:0] A;*  *reg [2:0] B;*  *reg O;*  *wire [6:0] S;*  *wire dp;*  *Adder\_Subtractor\_3bit module\_u\_test (B, A , O, S, dp);*  *initial begin*  *#100 B = 3'b110;*  *A = 3'b001;*  *O = 1'b0;*  *#100 B = 3'b111;*  *A = 3'b001;*  *O = 1'b1;*  *#100 B = 3'b111;*  *A = 3'b001;*  *O = 1'b0;*  *#100 B = 3'b100;*  *A = 3'b001;*  *O = 1'b1;*    *end*  *endmodule* |
| --- |

*Attach screenshot of waveform results here*

|  |
| --- |

## Exercise

Use Figure 6.0 and work out for cases when B is zero and subtracted from A. Reflect on the output obtained and suggest ways to fix (if required).

*Space to work out on Figure 7.0*

| B[2:0] = 000  O = 1  A[2:0] = 110  C [2:0] = 000 (2’s complement of 010)  U [3:0] = 0110  **U[2:0] = 110** **-> Answer**  U[3] = 0  Selector Bit for 2nd “Selective 2’s Complement” = 1  D[2:0] = 010 (2’s complement of U[2:0])  D[3] = 0 {AND of 0 [NOT of O(1) = 0] and [Carry U(3) = 0]}  **dp = 0** (0 means off for active Low so number is negative) [This is not true] . To Fix this: |
| --- |

*Space for reflection*

|  |
| --- |

## Lab Evaluation Rubrics

**Marks Distribution:**

|  |  | **LR2**  **Code** | **LR4**  **Data**  **Collection** | **LR5**  **Results** | **LR7**  **Viva** | **LR10**  **Analysis** |
| --- | --- | --- | --- | --- | --- | --- |
| **In-lab** | **Task b** | 20 points | 15 points | 10 points | 10 points | - |
| **Task c** | 20 points | - | 10 points | - |
| **Exercise** | **Task d** | - | - | - | 15 Points |
| **Total Marks =** | **100** | 40 | 15 | 20 | 10 | 15 |

**Marks Obtained:**

|  |  | **LR2**  **Code** | **LR4**  **Data**  **Collection** | **LR5**  **Results** | **LR7**  **Viva** | **LR10**  **Analysis** |
| --- | --- | --- | --- | --- | --- | --- |
| **In-lab** | **Task b** |  |  |  |  | - |
| **Task c** |  | - |  | - |
| **Exercise** | **Task d** | - | - | - |  |
| **Total Marks =** | **100** |  |  |  |  |  |