NAME: ANUMITA SAMADDAR

ENROLMENT NUMBER: 2022ETB013

6TH SEMESTER

COMPUTER ORGANISATION AND ARCHITECTURE ASSIGNMENT

Q) Design a 4 bit ALU that can perform the following operations:

1) 4 bit addition Opcode: 000

2)4 bit subtraction Opcode: 001

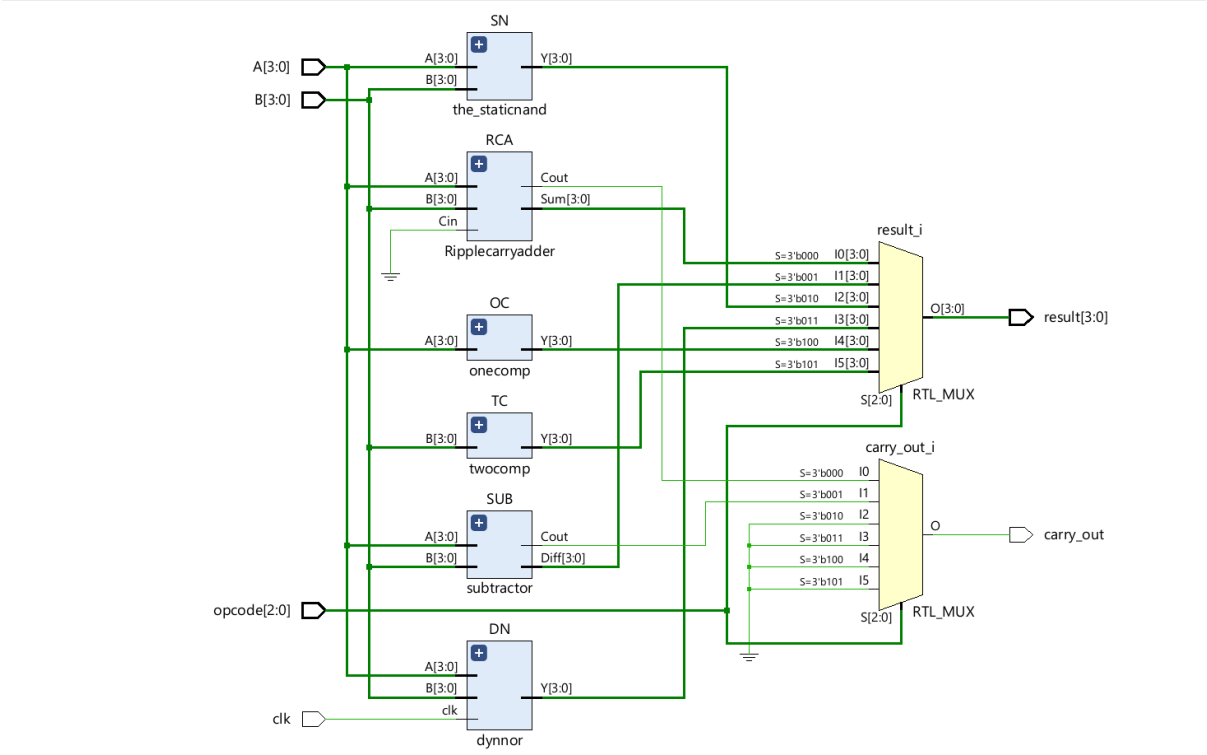
3) 4 input Nand operation using static Nand gate Opcode: 010

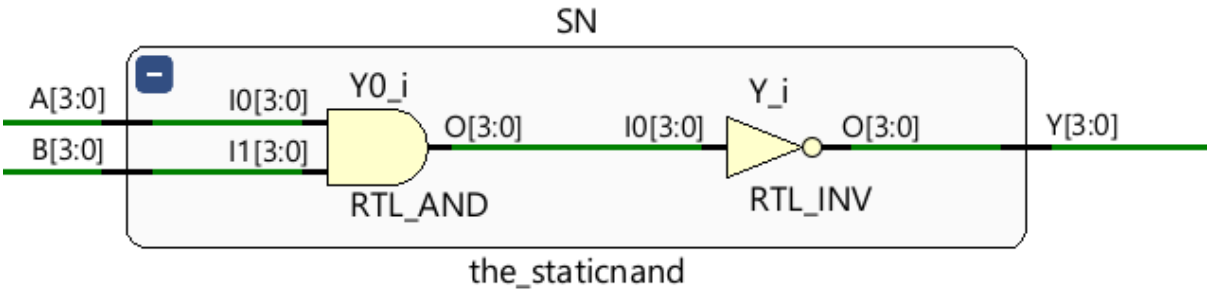
4) 4 input Nor operation using dynamic Nor gate Opcode: 011

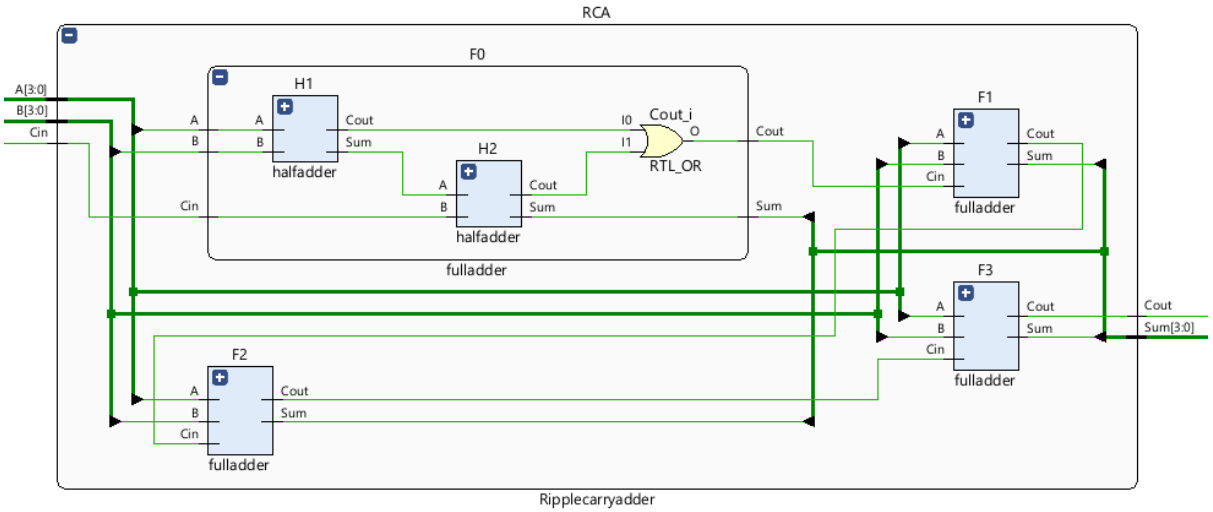
5)1's complement of A Opcode: 100

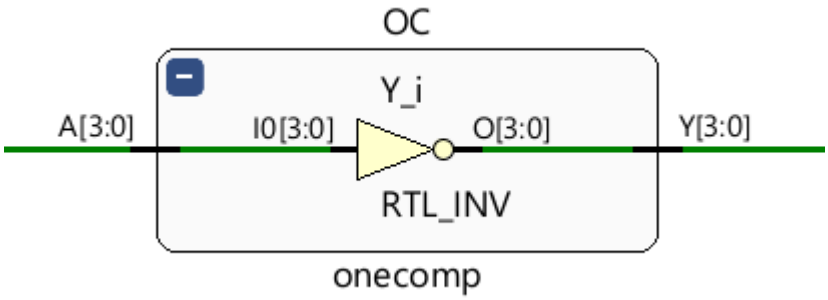
6)2's complement of B Opcode: 101

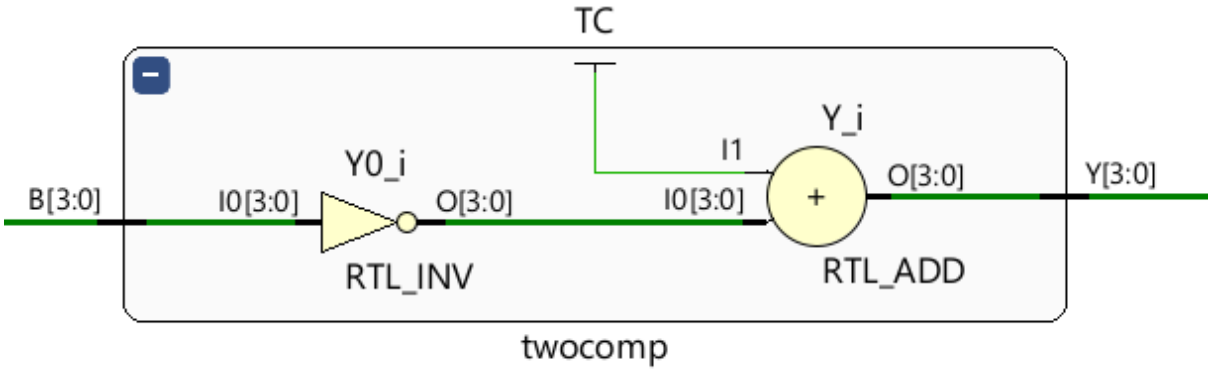
Provide schematic diagram and synthesis result.

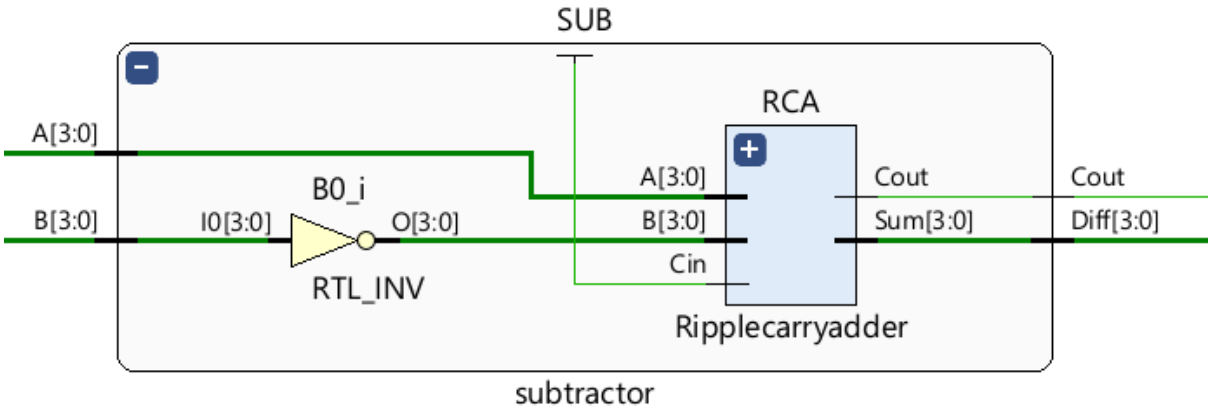


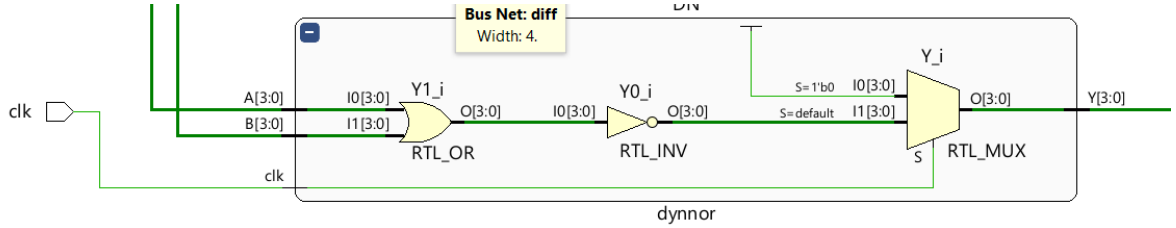



RCA
F0
H1
halfadder
H2
halfadder
fulladder
Cout_i
RTL_OR
F1
fulladder
F2
fulladder
F3
fulladder
A[3:0]
B[3:0]
Cin
Cout
Sum
Sum[3:0]
Ripplecarryadder


OC
A[3:0]
I0[3:0]
Y_i
O[3:0]
Y[3:0]
RTL_INV
onecomp


TC
B[3:0]
I0[3:0]
Y0_i
O[3:0]
RTL_INV
I1
Y_i
O[3:0]
Y[3:0]
RTL_ADD
twocomp


SUB
A[3:0]
B[3:0]
I0[3:0]
B0_i
O[3:0]
RTL_INV
RCA
A[3:0]
B[3:0]
Cin
Cout
Sum[3:0]
Ripplecarryadder
Cout
Diff[3:0]
subtractor




```verilog
module halfadder(A,B,Sum,Cout);
    input A, B;
    output Sum, Cout;
    assign Sum = A ^ B;
    assign Cout = A & B;
endmodule
```

```verilog
module fulladder(A,B,Cin,Sum,Cout);
    input A, B, Cin;
    output Sum, Cout;
    wire S1, C1, C2;
    halfadder H1 (A, B, S1, C1);
    halfadder H2 (S1, Cin, Sum, C2);
    assign Cout = C1 | C2;
endmodule
```

```verilog
module Ripplecarryadder(A,B,Cin,Sum,Cout);
    input [3:0] A, B;
    input Cin;
    output [3:0] Sum;
    output Cout;
    wire C1, C2, C3;
    fulladder F0(A[0], B[0], Cin, Sum[0], C1);
    fulladder F1(A[1], B[1], C1, Sum[1], C2);
    fulladder F2(A[2], B[2], C2, Sum[2], C3);
    fulladder F3(A[3], B[3], C3, Sum[3], Cout);
endmodule
```

```
22
23  module subtractor(A,B,Diff,Cout);
24      input [3:0] A,B;
25      output [3:0] Diff;
26      output Cout;
27      wire [3:0] B_neg;
28      Ripplecarryadder RCA(A, ~B, 1'b1, Diff, Cout);
29  endmodule
30
```

```
22
23  module the_staticnand(A,B,Y);
24      input [3:0] A, B;
25      output [3:0] Y;
26      assign Y = ~(A & B);
27  endmodule
28
```

```
22
23  module dynnor(A,B,clk,Y);
24      input [3:0] A, B;
25      input clk;  // Clock controls dynamic phases
26      output [3:0] Y;
27      // clk=0 -> precharge phase
28      // clk=1 -> evaluate phase
29      assign Y = (clk == 1'b0) ? 4'b1111 : ~(A | B);
30  endmodule
31
32
```

```
23  module onecomp(A,Y);
24      input [3:0] A;
25      output [3:0] Y;
26      assign Y = ~A;
27  endmodule
28
```

```
22  module twocomp (B,Y);
23      input [3:0] B;
24      output [3:0] Y;
25      assign Y = ~B + 1;
26  endmodule
27
```

```
23  module ALU_4bit (A,B,opcode,clk,result,carry_out);
24      input [3:0] A, B;
25      input [2:0] opcode;
26      input clk;              // Clock-> dynamic nor
27      output reg [3:0] result;
28      output reg carry_out;
29
30      wire [3:0] sum, diff, nand_out, nor_out, ones_comp, twos_comp;
31      wire sum_cout, diff_cout;
32
33      Ripplecarryadder RCA (.A(A), .B(B), .Cin(1'b0), .Sum(sum), .Cout(sum_cout));
34      subtractor SUB (.A(A), .B(B), .Diff(diff), .Cout(diff_cout));
35      the_staticnand SN (.A(A), .B(B), .Y(nand_out));
36      dynnor DN (.A(A), .B(B), .clk(clk), .Y(nor_out));
37      onecomp OC (.A(A), .Y(ones_comp));
38      twocomp TC (.B(B), .Y(twos_comp));
39
40      always @(*) begin
41          case (opcode)
42              3'b000: begin
43                  result = sum;
44                  carry_out = sum_cout;
45              end
46              3'b001: begin
47                  result = diff;
48                  carry_out = diff_cout;
49              end
50              3'b010: begin
51                  result = nand_out;
52                  carry_out = 0;
53              end
54              3'b011: begin
55                  result = nor_out;
56                  carry_out = 0;
57              end
58              3'b100: begin
59                  result = ones_comp;
60                  carry_out = 0;
61              end
62              3'b101: begin
```

```
            end
            3'b101: begin
                result = twos_comp;
                carry_out = 0;
            end
            default: begin
                result = 4'bxxxx;
                carry_out = 1'bx;
            end
        endcase
    end
endmodule
```

```
module ALU_4bit_tb;
    // Inputs
    reg [3:0] A;
    reg [3:0] B;
    reg [2:0] opcode;
    reg clk;

    // Outputs
    wire [3:0] result;
    wire carry_out;


    ALU_4bit uut (.A(A), .B(B), .opcode(opcode), .clk(clk), .result(result), .carry_out(carry_out));

    // (10 ns period)
    initial begin
        clk = 0;
        forever #5 clk = ~clk;
    end


    initial begin
        // Initial state
        A = 4'b0000; B = 4'b0000; opcode = 3'b000; #10;


        A = 4'b1001;
        B = 4'b0101;
        opcode = 3'b000;
        #10;

        A = 4'b1100;
        B = 4'b0111;
        opcode = 3'b001;
        #10;


        A = 4'b1010;
        B = 4'b1100;
        opcode = 3'b010;
```

```
44
45
46          A = 4'b1001;
47          B = 4'b0101;
48          opcode = 3'b000;
49          #10;
50
51          A = 4'b1100;
52          B = 4'b0111;
53          opcode = 3'b001;
54          #10;
55
56
57          A = 4'b1010;
58          B = 4'b1100;
59          opcode = 3'b010;
60          #10;
61
62          A = 4'b1001;
63          B = 4'b0110;
64          opcode = 3'b011;
65          #20;
66
67          A = 4'b1111;
68          opcode = 3'b100;
69          #10;
70
71          B = 4'b0010;
72          opcode = 3'b101;
73          #10;
74
75          $finish;
76      end
77
78      initial begin
79          $monitor("Time=%0t | clk=%b | opcode=%b | A=%b | B=%b | result=%b | carry_out=%b",
80                   $time, clk, opcode, A, B, result, carry_out);
81      end
82  endmodule
83
84
```

```
# run 1000ns
Time=0 | clk=0 | opcode=000 | A=0000 | B=0000 | result=0000 | carry_out=0
Time=5000 | clk=1 | opcode=000 | A=0000 | B=0000 | result=0000 | carry_out=0
Time=10000 | clk=0 | opcode=000 | A=1001 | B=0101 | result=1110 | carry_out=0
Time=15000 | clk=1 | opcode=000 | A=1001 | B=0101 | result=1110 | carry_out=0
Time=20000 | clk=0 | opcode=001 | A=1100 | B=0111 | result=0101 | carry_out=1
Time=25000 | clk=1 | opcode=001 | A=1100 | B=0111 | result=0101 | carry_out=1
Time=30000 | clk=0 | opcode=010 | A=1010 | B=1100 | result=0111 | carry_out=0
Time=35000 | clk=1 | opcode=010 | A=1010 | B=1100 | result=0111 | carry_out=0
Time=40000 | clk=0 | opcode=011 | A=1001 | B=0110 | result=1111 | carry_out=0
Time=45000 | clk=1 | opcode=011 | A=1001 | B=0110 | result=0000 | carry_out=0
Time=50000 | clk=0 | opcode=011 | A=1001 | B=0110 | result=1111 | carry_out=0
Time=55000 | clk=1 | opcode=011 | A=1001 | B=0110 | result=0000 | carry_out=0
Time=60000 | clk=0 | opcode=100 | A=1111 | B=0110 | result=0000 | carry_out=0
Time=65000 | clk=1 | opcode=100 | A=1111 | B=0110 | result=0000 | carry_out=0
Time=70000 | clk=0 | opcode=101 | A=1111 | B=0010 | result=1110 | carry_out=0
Time=75000 | clk=1 | opcode=101 | A=1111 | B=0010 | result=1110 | carry_out=0
$finish called at time : 80 ns : File "C:/Users/Anumita Samaddar/ALU4bit anumita/
```


Name
Value
A[3:0]
f
B[3:0]
2
opcode[2:0]
5
clk
1
result[3:0]
e
carry_out
0
0.000 ns
20.000 ns
40.000 ns
60.000 ns
80.000 ns
80.000 ns
0 9 c a 9 f
0 5 7 c 6 2
0 1 2 3 4 5
0 e 5 7 f 0 f 0 e