4. DESIGN A 4 BIT FULL ADDER AND SUBTRACTOR AND SIMULATE THE SAME USING BASIC GATES.

AIM: Design a 4 bit full adder and subtractor and simulate the same using basic gates.

THEORY: In Digital Circuits, A **Binary Adder-Subtractor** is capable of both the addition and subtraction of binary numbers in one circuit itself. The operation is performed depending on the binary value the control signal holds. It is one of the components of the ALU (Arithmetic Logic Unit). This Circuit requires prerequisite knowledge of Ex-or Gate, Binary Addition and Subtraction, and Full Adder. Let's consider two 4-bit binary numbers A and B as inputs to the Digital Circuit for the operation with digits.

```
A0 A1 A2 A3 for A
B0 B1 B2 B3 for B
```

The circuit consists of 4 full adders since we are performing operations on 4-bit numbers. There is a control line K that holds a binary value of either 0 or 1 which determines that the operation is carried out is addition or subtraction.

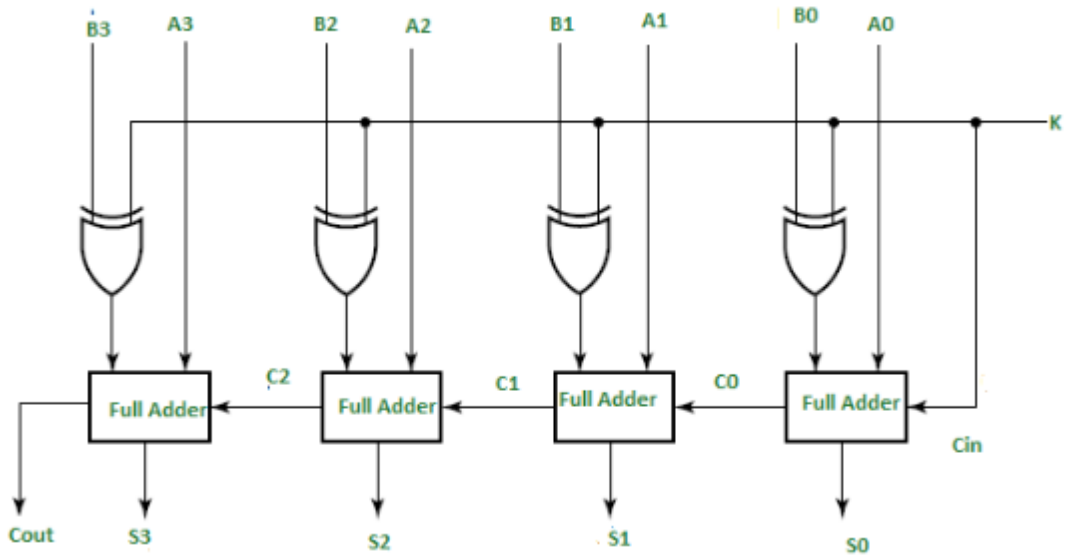


Fig 4.1 Full adder and Subtractor circuit

As shown in the figure, the first full adder has a control line directly as its input (input carry Cin), The input A0 (The least significant bit of A) is directly input in the full adder. The third input is the ex-or of B0 and K.The two outputs produced are Sum/Difference (S0) and Carry (C0). If the value of K (Control line) is 1, the output of B0(ex-or)K=B0′(Complement B0). Thus

the operation would be A+(B0′). Now 2's complement subtraction for two numbers A and B is given by A+B'+Cin. This suggests that when K=1, the operation being performed on the four-bit numbers is subtraction. Similarly If the Value of K=0, B0 (ex-or) K=B0. The operation is A+B which is simple binary addition. This suggests that When K=0, the operation is performed on the four-bit numbers in addition. Then C0 is serially passed to the second full adder as one of its outputs. The sum/difference S0 is recorded as the least significant bit of the sum/difference. A1, A2, A3 are direct inputs to the second, third and fourth full adders. Then the third input is the B1, B2, B3 EX-ORed with K to the second, third and fourth full adder respectively. Carry outputs of each full adder are connected as input to the successive full adder and the last output carry is taken as the cout.

4 Bit Full Adder and Subtractor Truth Table

CONTROL		INPUT 4 Bit Full Adder												
Cin	K	A3	A2	A1	A0	B3	B2	B2	B1	S3	S2	S1	S0	Cout
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	1	0	0	1	0	0
0	0	0	0	1	0	0	0	1	0	0	1	0	0	0
0	0	0	0	1	1	0	0	1	1	0	1	1	0	0
0	0	0	1	0	0	0	1	0	0	1	0	0	0	0
0	0	0	1	0	1	0	1	0	1	1	0	1	0	0
0	0	0	1	1	0	0	1	1	0	1	1	0	0	0
0	0	0	1	1	1	0	1	1	1	1	1	1	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0	1
0	0	1	0	0	1	1	0	0	1	0	0	1	0	1
0	0	1	0	1	0	1	0	1	0	0	1	0	0	1
0	0	1	0	1	1	1	0	1	1	0	1	1	0	1
0	0	1	1	0	0	1	1	0	0	1	0	0	0	1
0	0	1	1	0	1	1	1	0	1	1	0	1	0	1
0	0	1	1	1	0	1	1	1	0	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1

Fig.4.2 Truth table of parallel adder and subtractor

Procedure: 4 bit parallel adder/ subtractor is simulated in NI multisim software with the utilization of IC 7483 along with XOR gate 7486 according to the circuit diagram as shown in the figure 4.1.

4 bit adder/sub tractor circuit:

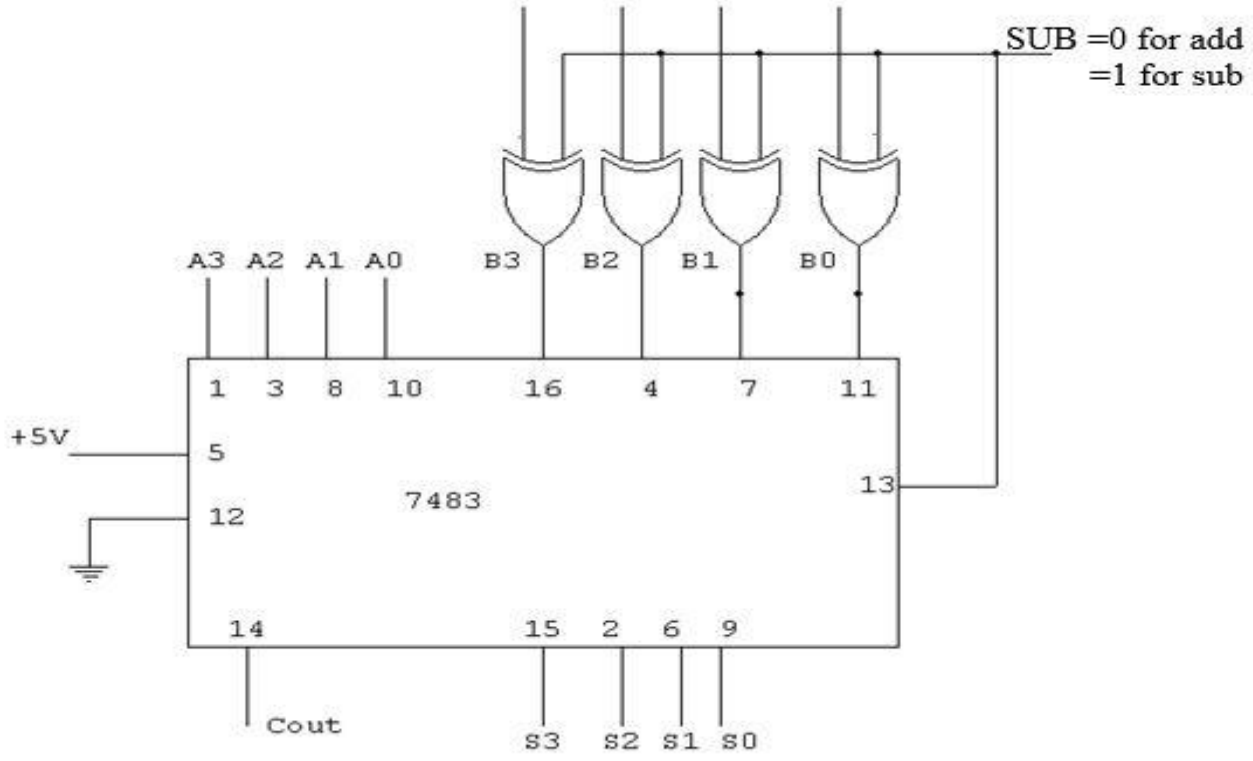


Fig 4.3 4 bit adder/subtractor circuit using 7483 and XOR gate 7486

- Drop the IC 7483 from the source panel of NI multisim platform once you open your project layout; drop the indicators, interactive digital constant, ground for signal indicator, power source, for completion of the circuit.
- Take IC7486 XOR gate from the panel and make the circuit arrangement as per the diagram shown in fig 4.1.
- Take the numbers X and Y(from the observation book)
- Observe the output from the indicators after the specified two 4 bit numbers are considered at the input.

Output from the simulation:

Summary and Conclusion:

Fig 2.1 Block Diagram of Half Adder

Truth Table for Half Adder

Inputs		Outputs	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

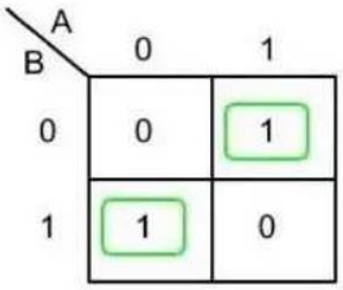


Fig 2.2 K-MAP for SUM

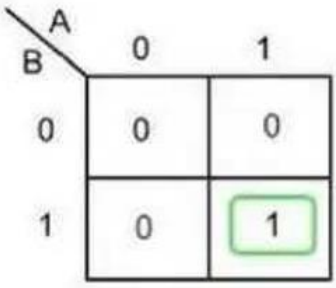


Fig 2.3 K-MAP for CARRY

The simplified expressions for K-map are

1) Sum =A'B+AB'
2) Carry=AB

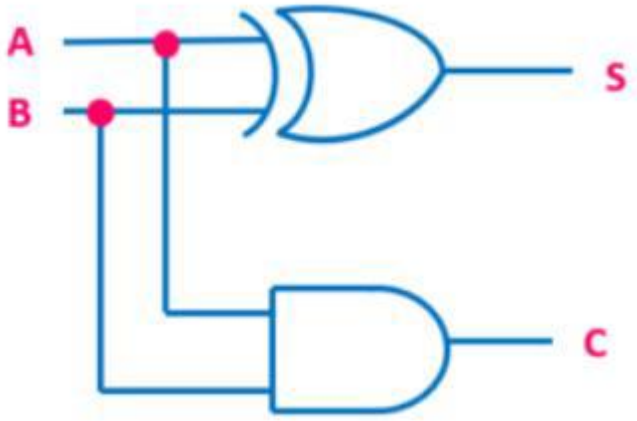


Fig 2.4 Logic Circuit for Half Adder

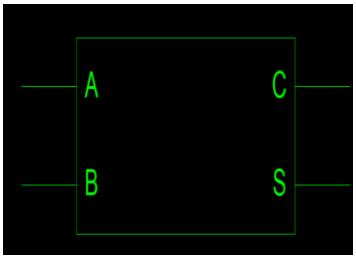
The half adder circuit is suitable for the addition of two bits at the LSB (Least Significant Bit) Position. Because during the addition of two bits at the LSB position, there is no incoming carry. But whenever there is an incoming carry (Cin) along with the two bits, then a full adder circuit can be used.

Procedure

1. Write a Verilog Code using appropriate modeling style (Structural, data flow, or behavioral) to describe the desired logic. Ensure Your Verilog code includes input and output ports, logical operations and any necessary components. Obtain RTL Schematic for Half adder.

Verilog HDL for Half Adder using Data Flow

< write the verilog code for Half adder in Data Flow from the observation >



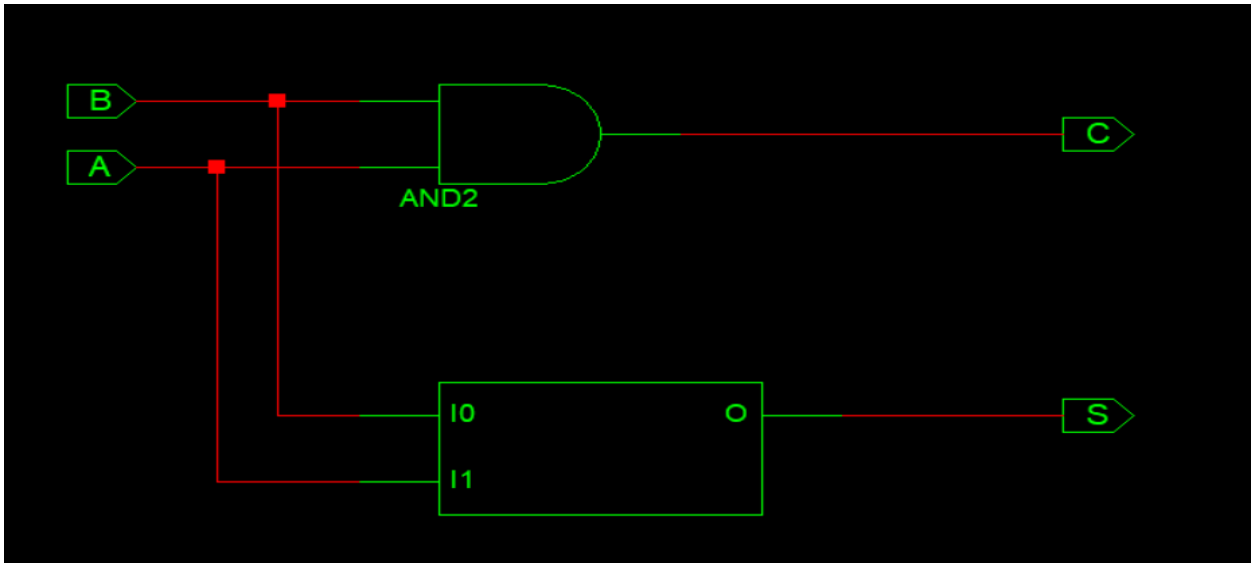


Fig 2.5 Simple Circuit
Fig 2.6 RTL Schematic for Half Adder

1.a) Create a test bench module that instantiates your Verilog module, defines test inputs and applies stimulus and generate the simulation waveform.

```
`timescale 1ns/1ps

module test;
   reg a = 1'b0;
   reg b = 1'b0;
   wire s;
   wire c;

da UUT (
       .a(a),
       .b(b),
       .s(s),
```

```
    .c(c));

integer TX_FILE = 0;
integer TX_ERROR = 0;

initial begin  // Open the results file...
    TX_FILE = $fopen("results.txt");
    #2000 // Final time:  2000 ns
    if (TX_ERROR == 0) begin
        $display("No errors or warnings.");
        $fdisplay(TX_FILE, "No errors or warnings.");
    end else begin
        $display("%d errors found in simulation.", TX_ERROR);
        $fdisplay(TX_FILE, "%d errors found in simulation.", TX_ERROR);
    end
    $fclose(TX_FILE);
   $stop;
end

initial begin
    // -------------  Current Time:  200ns
    #200;
    b = 1'b1;
    // -------------------------------------
    // -------------  Current Time:  300ns
    #100;
    a = 1'b1;
    b = 1'b0;
    // -------------------------------------
    // -------------  Current Time:  400ns
    #100;
    b = 1'b1;
    // -------------------------------------
    // -------------  Current Time:  2000ns
    #1600;
    a = 1'b0;
end

task CHECK_s;
    input NEXT_s;

    #0 begin
        if (NEXT_s !== s) begin
            $display("Error at time=%dns s=%b, expected=%b", $time, s, NEXT_s);
            $fdisplay(TX_FILE, "Error at time=%dns s=%b, expected=%b", $time, s, NEXT_s);
            $fflush(TX_FILE);
```

```
            TX_ERROR = TX_ERROR + 1;
        end
    end
endtask
task CHECK_c;
    input NEXT_c;

    #0 begin
        if (NEXT_c !== c) begin
            $display("Error at time=%dns c=%b, expected=%b", $time, c, NEXT_c);
            $fdisplay(TX_FILE, "Error at time=%dns c=%b, expected=%b", $time, c, NEXT_c);
            $fflush(TX_FILE);
            TX_ERROR = TX_ERROR + 1;
        end end endtask  endmodule
```

Simulation Output

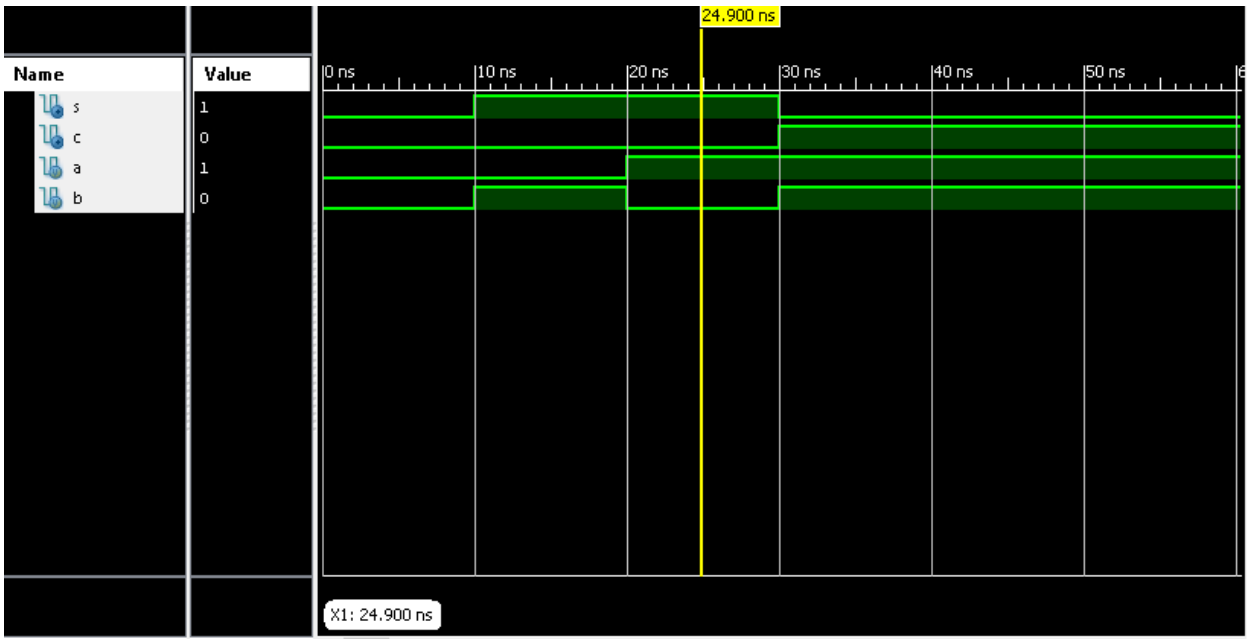


Fig 2.7 RTL Simulation Waveform for Half Adder.

2. Half adder using Structural Modelling

< write the Verilog code and Test bench>

2.a) Test bench for Half Adder using structural modeling

```
`timescale 1ns/1ps

module test;

    reg a = 1'b0;
```

```
   reg b = 1'b0;
   wire s;
   wire c;
s UUT (
       .a(a),
       .b(b),
       .s(s),
.c(c));
integer TX_FILE = 0;
   integer TX_ERROR = 0;
    initial begin  // Open the results file...
       TX_FILE = $fopen("results.txt");
       #2000 // Final time:  2000 ns
       if (TX_ERROR == 0) begin
          $display("No errors or warnings.");
          $fdisplay(TX_FILE, "No errors or warnings.");
       end else begin
          $display("%d errors found in simulation.", TX_ERROR);
          $fdisplay(TX_FILE, "%d errors found in simulation.", TX_ERROR);
       end
       $fclose(TX_FILE);
       $stop;
   end
initial begin
       // -------------  Current Time:  200ns
```

```
        #200;
        b = 1'b1;
        // -------------------------------------
        // -------------  Current Time:  300ns
        #100;
        a = 1'b1;
        b = 1'b0;
        // -------------------------------------
        // -------------  Current Time:  400ns
        #100;
        a = 1'b0;
        b = 1'b1;
        // -------------------------------------
        // -------------  Current Time:  500ns
        #100;
        a = 1'b1;
    end
task CHECK_s;
        input NEXT_s;
#0 begin
            if (NEXT_s !== s) begin
                $display("Error at time=%dns s=%b, expected=%b", $time, s, NEXT_s);
                $fdisplay(TX_FILE, "Error at time=%dns s=%b, expected=%b", $time, s, NEXT_s);
                $fflush(TX_FILE);
                TX_ERROR = TX_ERROR + 1;
```

```
        end
      end
  endtask
  task CHECK_c;
    input NEXT_c;
#0 begin
        if (NEXT_c !== c) begin
          $display("Error at time=%dns c=%b, expected=%b", $time, c, NEXT_c);
          $fdisplay(TX_FILE, "Error at time=%dns c=%b, expected=%b", $time, c, NEXT_c);
          $fflush(TX_FILE);
          TX_ERROR = TX_ERROR + 1;
        end
      end
  endtask
endmodule
```

3. Half Adder using Behavioural Modelling

<follow the same procedure>

3. a) Test bench for Half Adder using Behavioral Model

```
`timescale 1ns/1ps
module test;
  reg a = 1'b0;
  reg b = 1'b0;
  wire s;
```

```
    wire c;
 beh UUT (
        .a(a),
        .b(b),
        .s(s),
        .c(c));
integer TX_FILE = 0;
    integer TX_ERROR = 0;
    initial begin  // Open the results file...
        TX_FILE = $fopen("results.txt");
        #2000 // Final time:  2000 ns
        if (TX_ERROR == 0) begin
            $display("No errors or warnings.");
            $fdisplay(TX_FILE, "No errors or warnings.");
        end else begin
            $display("%d errors found in simulation.", TX_ERROR);
            $fdisplay(TX_FILE, "%d errors found in simulation.", TX_ERROR);
        end
        $fclose(TX_FILE);
        $stop;
    end
 initial begin
        // ----------  Current Time:  200ns
        #200;
        b = 1'b1;
```

```
        // ------------------------------------
        // -------------  Current Time:  300ns
        #100;
        a = 1'b1;
        b = 1'b0;
        // ------------------------------------
        // -------------  Current Time:  400ns
        #100;
        b = 1'b1;
        // ------------------------------------
        // -------------  Current Time:  2000ns
        #1600;
        a = 1'b0;
    end
task CHECK_s;
        input NEXT_s;
 #0 begin
            if (NEXT_s !== s) begin
                $display("Error at time=%dns s=%b, expected=%b", $time, s, NEXT_s);
                $fdisplay(TX_FILE, "Error at time=%dns s=%b, expected=%b", $time, s, NEXT_s);
                $fflush(TX_FILE);
                TX_ERROR = TX_ERROR + 1;
            end
        end
    endtask
```

```
    task CHECK_c;
        input NEXT_c;
#0 begin
            if (NEXT_c !== c) begin
                $display("Error at time=%dns c=%b, expected=%b", $time, c, NEXT_c);
                $fdisplay(TX_FILE, "Error at time=%dns c=%b, expected=%b", $time, c, NEXT_c);
                $fflush(TX_FILE);
                TX_ERROR = TX_ERROR + 1;
            end
        end
    endtask
endmodule
```

SUMMARY AND CONCLUSION:

Half adder circuit design was executed successfully according to the design procedures and the circuit was implemented in Verilog HDL for the three models such as Structural, Dataflow and Behavioral Models.