RTL CHALLENGE

DAY-5:- Verilog Code for Implementation of FULL ADDER.

Software used:- Xilinx Vivado(2023.1)

Theory:

Full Adder:-

Full Adder is the adder that adds three inputs and produces two outputs. The first two inputs are A and B and the third input is an input carry as C-IN. The output carry is designated as C-OUT and the normal output is designated as S which is SUM. The C-OUT is also known as the majority 1’s detector, whose output goes high when more than one input is high. A full adder logic is designed in such a manner that can take eight inputs together to create a byte-wide adder and cascade the carry bit from one adder to another. we use a full adder because when a carry-in bit is available, another 1-bit adder must be used since a 1-bit half-adder does not take a carry-in bit. A 1-bit full adder adds three operands and generates 2-bit results.

Full adder circuit diagram:-

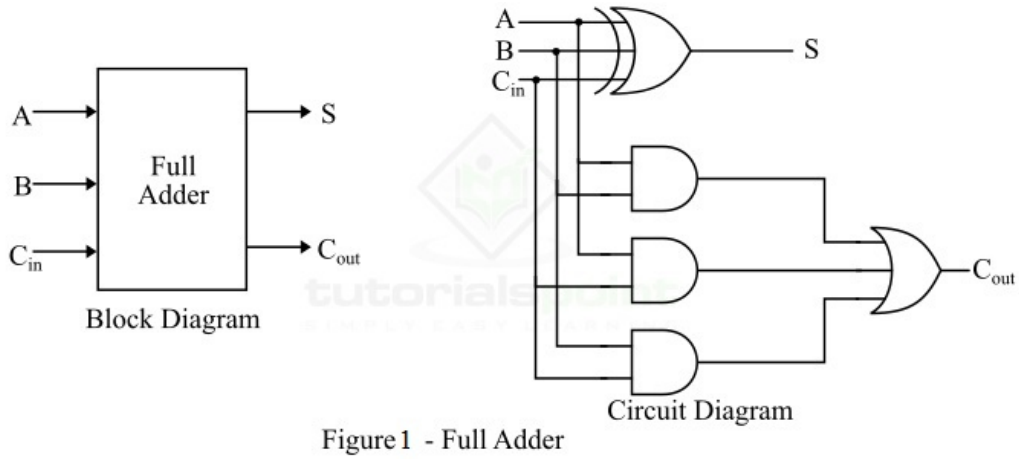


Figure 1 - Full Adder

TRUTH TABLE:-

Inputs			Outputs	
A	B	C – IN	Sum	C - Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

4 Bit Full Adder :-

A four-bit full adder is a crucial component in digital circuitry, facilitating the addition of four pairs of binary digits along with an initial carry-in bit. Typically constructed by cascading four individual full adders together, it processes inputs representing the binary numbers to be added and produces outputs consisting of four sum bits and a final carry-out bit. Each full adder stage handles one pair of binary digits and passes any carry to the subsequent stage. Implemented using combinations of basic logic gates such as AND, OR, and XOR gates, the four-bit full adder plays a pivotal role in arithmetic operations within CPU's and other digital systems. Its ability to efficiently add larger binary numbers makes it indispensable for various computational tasks in digital design and engineering.

16 Bit Full Adder:-

A 16-bit full adder is a vital component within digital systems, facilitating the addition of two 16-bit binary numbers along with an initial carry-in bit. Constructed by cascading 16 individual full adders together, it processes inputs representing the binary numbers to be added and produces outputs comprising 16 sum bits and a final carry-out bit. Each full adder stage manages one pair of corresponding bits from the input numbers, passing any generated carry to the subsequent stage. Utilizing combinations of basic logic gates like AND, OR, and XOR gates, the 16-bit full adder is integral to arithmetic operations in CPU’s, ALU's , and various digital systems where handling larger binary numbers is essential. Its scalability and efficiency.

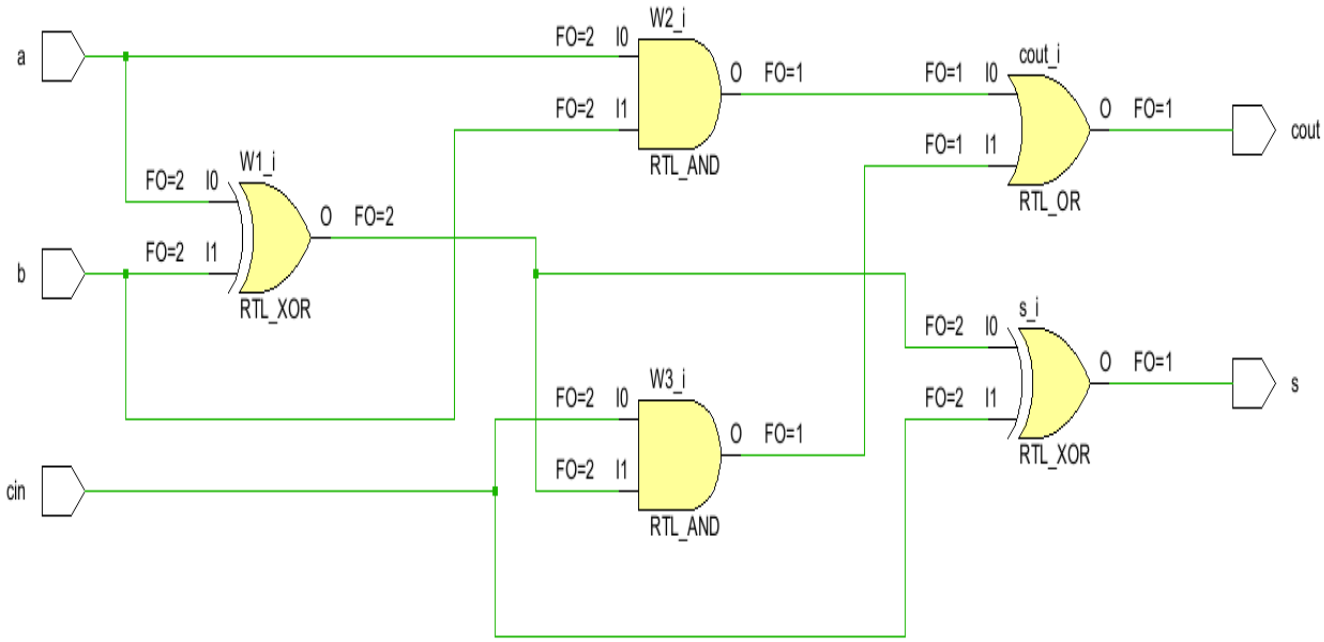
Code for Full Adder using Gates:-

```
// Design Name: Gatelevel
// Module Name: FullAdderusingGates
// Project Name: RTL Challenge

module FullAdderusingGates(
    input a,b,cin,
    output s,cout
    );
    wire W1,W2,W3;
    xor(W1,a,b);
    and(W2,a,b);
    xor(s,W1,cin);
    and(W3,cin,W1);
    or(cout,W2,W3);

endmodule
```

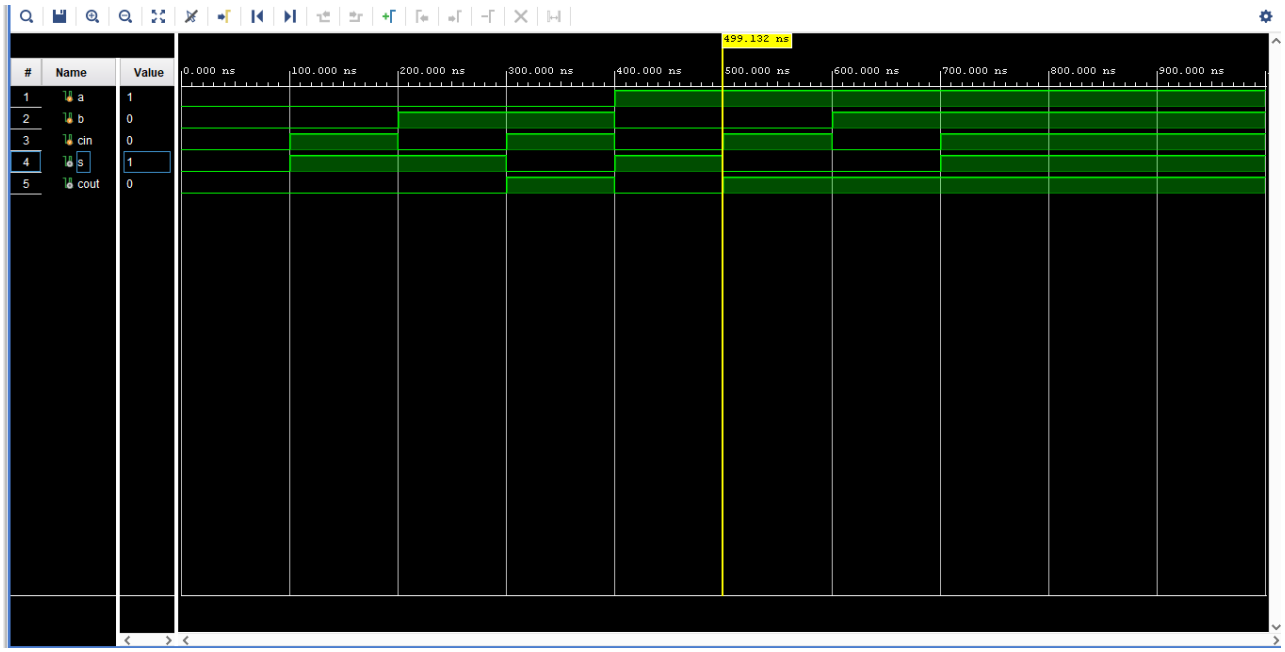
Test Bench for Full Adder using Gates:-

```
// Design Name: TestBench
// Module Name: FullAdderusingGates_tb
// Project Name: RTL Challenge

module FullAdderusingGates_tb();
reg a;
reg b;
reg cin;
wire s,cout;
FullAdderusingGates uut(.a(a),.b(b),.cin(cin),.cout(cout),.s(s));
initial
    begin
        a=0;b=0;cin=0;#100;
        a=0;b=0;cin=1;#100;
        a=0;b=1;cin=0;#100;
        a=0;b=1;cin=1;#100;
        a=1;b=0;cin=0;#100;
        a=1;b=0;cin=1;#100;
        a=1;b=1;cin=0;#100;
        a=1;b=1;cin=1;#100;

    end
endmodule
```

Schematic diagram:-



Output Wave Forms:-



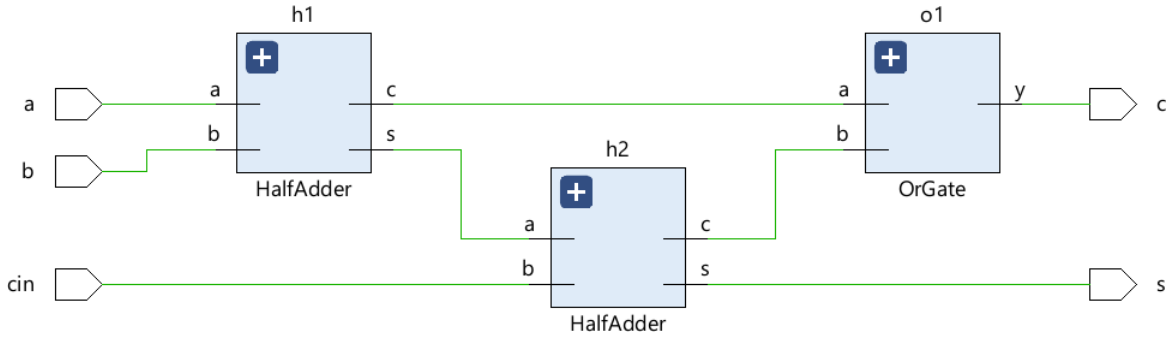
Code for Full Adder using Half Adder:-

```
module FullAdderUsingHalfAdder(
    input a,b,cin,
    output s,c
    );
    wire w1,w2,w3;
    HalfAdder h1 (.a(a),.b(b),.s(w1),.c(w2));
    HalfAdder h2 (.a(w1),.b(cin),.s(s),.c(w3));
    OrGate o1(w2,w3,c);
endmodule
```

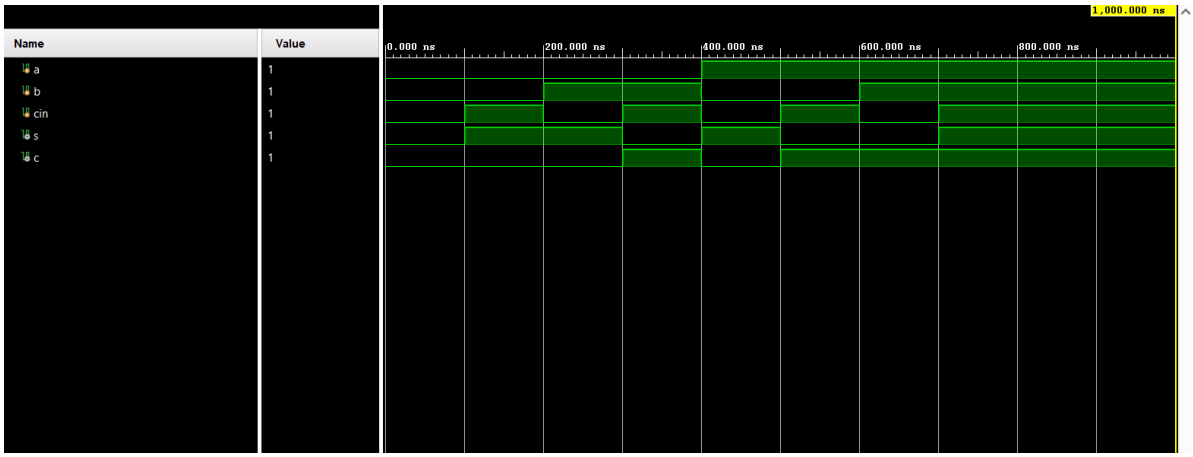
Test Bench :-

```
module FullAdderUsingHalfAdder_tb;
reg a,b,cin;
wire s,c;
FullAdderUsingHalfAdder uut(.a(a),.b(b),.cin(cin),.s(s),.c(c));
initial
    begin
        a=0;b=0;cin=0;#100;
        a=0;b=0;cin=1;#100;
        a=0;b=1;cin=0;#100;
        a=0;b=1;cin=1;#100;
        a=1;b=0;cin=0;#100;
        a=1;b=0;cin=1;#100;
        a=1;b=1;cin=0;#100;
        a=1;b=1;cin=1;#100;

    end
endmodule
```

Schematic:-



Output Wave Forms:-



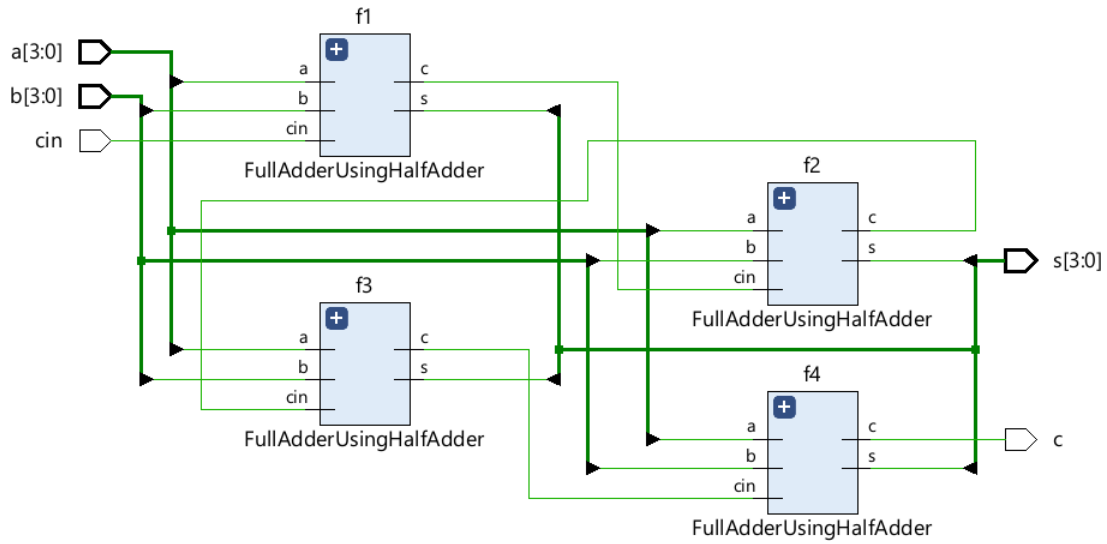
Code for Four Bit Full Adder :-

```
module FourBitFullAdder(
    input [3:0] a,b,
    input cin,
    output [3:0] s,
    output c
    );
    wire w1,w2,w3;
    FullAdderUsingHalfAdder f1(.a(a[0]),.b(b[0]),.cin(cin),.s(s[0]),.c(w1));
    FullAdderUsingHalfAdder f2(.a(a[1]),.b(b[1]),.cin(w1),.s(s[1]),.c(w2));
    FullAdderUsingHalfAdder f3(.a(a[2]),.b(b[2]),.cin(w2),.s(s[2]),.c(w3));
    FullAdderUsingHalfAdder f4(.a(a[3]),.b(b[3]),.cin(w3),.s(s[3]),.c(c));
endmodule
```

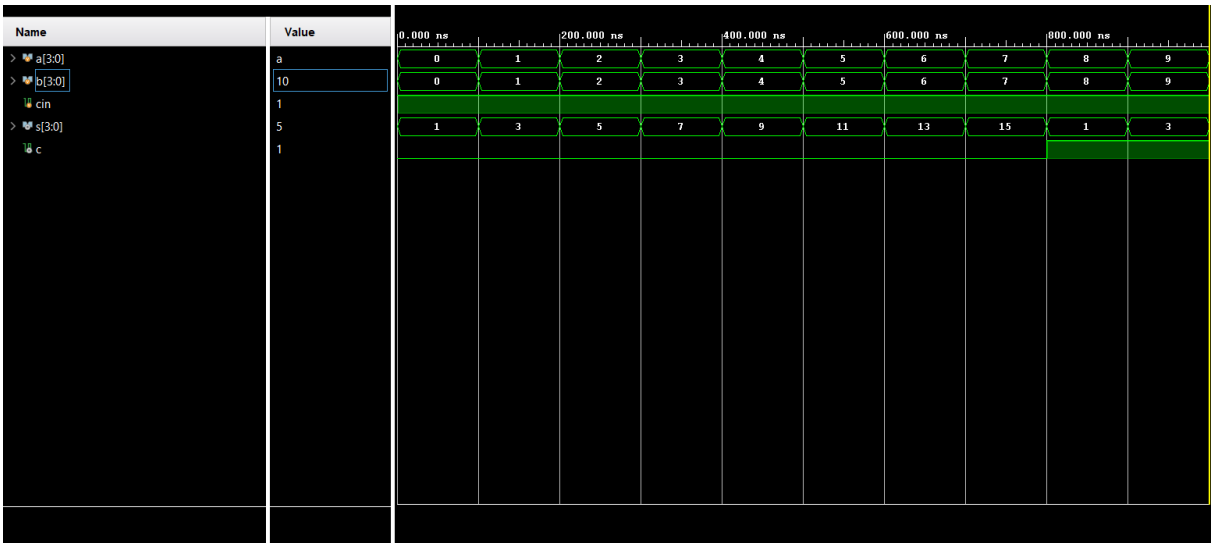
Test Bench:-

```
    module FourBitFullAdder_tb;
    reg [3:0]a;
    reg [3:0]b;
    reg cin;
    wire [3:0]s,c;
    FourBitFullAdder uut (.a(a),.b(b),.cin(cin),.s(s),.c(c));
    initial
        begin
            a=4'b0000;b=4'b0000;cin=1;#100;
            a=4'b0001;b=4'b0001;cin=1;#100;
            a=4'b0010;b=4'b0010;cin=1;#100;
            a=4'b0011;b=4'b0011;cin=1;#100;
            a=4'b0100;b=4'b0100;cin=1;#100;
            a=4'b0101;b=4'b0101;cin=1;#100;
            a=4'b0110;b=4'b0110;cin=1;#100;
            a=4'b0111;b=4'b0111;cin=1;#100;
            a=4'b1000;b=4'b1000;cin=1;#100;
            a=4'b1001;b=4'b1001;cin=1;#100;
            a=4'b1010;b=4'b1010;cin=1;#100;
            a=4'b1011;b=4'b1011;cin=1;#100;
            a=4'b1100;b=4'b1100;cin=1;#100;
            a=4'b1101;b=4'b1101;cin=1;#100;
            a=4'b1110;b=4'b1110;cin=1;#100;
            a=4'b1111;b=4'b1111;cin=1;#100;
        end
    endmodule
```

Schematic:-



Output Wave Forms:-



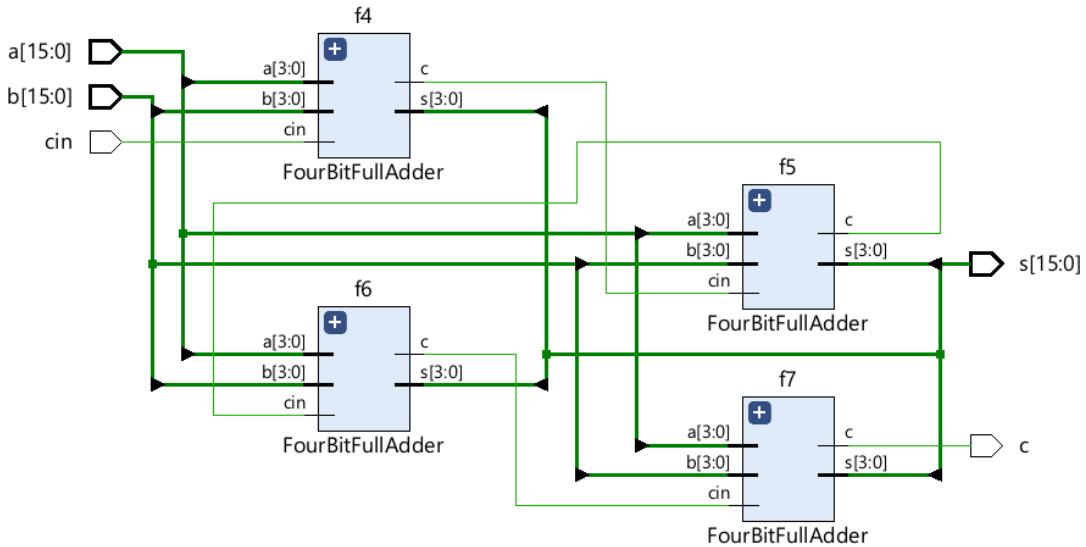
Code for 16 Bit Full Adder :-

```
module SixteenBitFullAdderUsing4BitFullAdder(
    input [15:0] a,b,
    input cin,
    output [15:0] s,
    output c
    );
    wire w1,w2,w3;
    FourBitFullAdder f4 (.a(a[3:0]),.b(b[3:0]),.cin(cin),.s(s[3:0]),.c(w1));
    FourBitFullAdder f5 (.a(a[7:4]),.b(b[7:4]),.cin(w1),.s(s[7:4]),.c(w2));
    FourBitFullAdder f6 (.a(a[11:8]),.b(b[11:8]),.cin(w2),.s(s[11:8]),.c(w3));
    FourBitFullAdder f7 (.a(a[15:12]),.b(b[15:12]),.cin(w3),.s(s[15:12]),.c(c));
endmodule
```

Test Bench:-

```
module SixteenBitFullAdderUsing4BitFullAdder_tb;
reg [15:0]a;
reg [15:0]b;
reg cin;
wire [15:0]s;
wire c;
SixteenBitFullAdderUsing4BitFullAdder uut (.a(a),.b(b),.cin(cin),.s(s),.c(c));
initial
    begin
        a=5;
        b=16'b1000110011110000;
        cin = 1;#100;

    end
endmodule
```

Schematic:-



Output Wave Forms:-