Experiment – 1

Aim: To design and simulation of half adder and full adder using different types of modeling.

Software Used: AMD Vivado 2023.2

Theory:

1. Half Adder: The half adder is a basic building block of adding two numbers as two inputs and produce out two outputs. The adder is used to perform OR operation of two single bit binary numbers. The **augent** and **addent** bits are two input states, and **'carry**' and **'sum** 'are two output states of the half adder.

Table 1.1 Truth Table of Half Adder

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

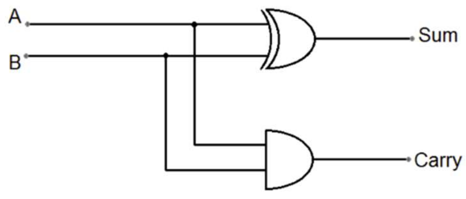

Fig 1.1 Circuit Diagram of Half Adder

In the above table, 'A' and' B' are the input states, and 'sum' and 'carry' are the output states. The carry output is 0 in case where both the inputs are not 1. The least significant bit of the sum is defined by the 'sum' bit. The design equations of the sum and carry are as follows:

Sum =A'B + AB' or A(xor)B

Carry = A.B

From this, a half adder circuit can be easily constructed using one XOR gate and AND gate.

2. Full Adder: The half adder is used to add only two numbers. To overcome this problem, the full adder was developed. The full adder is used to add three 1-bit binary numbers A, B, and C_{in}. The full adder has three input states and two output states i.e., sum and carry.

Table 1.2 Truth Table of Full Adder

A	B	C_{in}	Sum	Carry
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

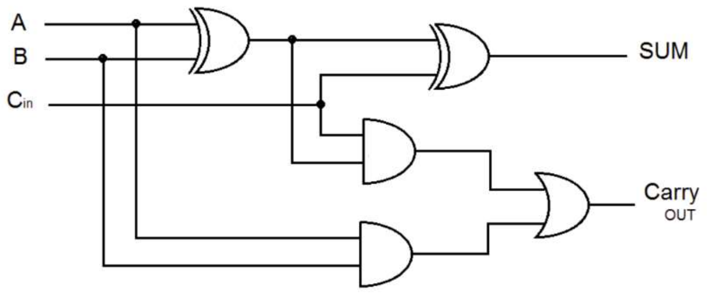

Fig 1.2 Circuit Diagram of Full Adder

In the above table, 'A', ' B' and 'C_{in}' are the input states, and 'sum' and 'carry' are the output states. The design equations of the sum and carry are as follows:

Sum =$A'B'C_{in} + A'B\, C_{in} + AB'\, C_{in}' + ABC_{in}$ or A(xor)B(xor) C_{in}

Carry = $AB + BC_{in} + C_{in}A$

Verilog Code:

1. Half Adder:

```
Gate Level Modeling:

module half_adder(input a, b, output sum, carry);

    xor gate_1(sum, a, b);

    and gate_2(carry, a, b);

endmodule


Data Flow Level Modeling:

module half_adder(input a, b, output sum, carry);

    assign sum = (~a&b)+(a&~b);

    assign carry = a&b;

endmodule


Behavioral Modeling:

module half_adder(input a, b, output reg sum, carry);

    always @(a, b)

    begin

        case({a, b})

        2'b00: begin sum = 0; carry = 0; end

        2'b01: begin sum = 1; carry = 0; end

        2'b10: begin sum = 1; carry = 0; end

        2'b11: begin sum = 0; carry = 1; end

        endcase

    end

endmodule
```

2. Full Adder:

Gate Level Modeling:

```
module full_adder(input a, b, carryIn, output sum, carryOut);
    wire wire_1, wire_2, wire_3;
    and gate_1(wire_1, a, b);
    and gate_2(wire_2, carryIn, b);
    and gate_3(wire_3, a, carryIn);
    or gate_4(carryOut, wire_1, wire_2, wire_3);
    xor gate_5(sum, a, b, carryIn);
endmodule
```

Data Flow Level Modeling:

```
module full_adder(input a, b, carryIn, output sum, carryOut);
    assign {carryOut, sum} = a+b+carryIn;
endmodule
```

Behavioral Modeling:

```
module full_adder(input a, b, carryIn, output reg sum, carryOut);
    always @(a, b, carryIn)
    begin
        case({a, b, carryIn})
        3'b000: begin sum = 0; carryOut = 0; end
        3'b001: begin sum = 1; carryOut = 0; end
        3'b010: begin sum = 1; carryOut = 0; end
        3'b011: begin sum = 0; carryOut = 1; end
        3'b100: begin sum = 1; carryOut = 0; end
        3'b101: begin sum = 0; carryOut = 1; end
        3'b110: begin sum = 0; carryOut = 1; end
        3'b111: begin sum = 1; carryOut = 1; end
        endcase
    end
endmodule
```

Test Bench Code:

1. Half Adder:

```
module half_adder_tb();
    reg A, B;
    wire SUM, CARRY;
    half_adder dut(.a(A), .b(B), .sum(SUM), .carry(CARRY));
    initial begin
    A = 0; B = 0; #200 A = 0; B = 1; #200
    A = 1; B = 0; #200 A = 1; B = 1; #200
    A = 0; B = 0;
    end
endmodule
```

2. Full Adder:

```
module full_adder_tb();
    reg A, B, CARRY_IN;
    wire SUM, CARRY_OUT;
    full_adder dut(.a(A), .b(B), .carryIn(CARRY_IN), .sum(SUM), .carryOut(CARRY_OUT));
    initial begin
    A = 0; B = 0; CARRY_IN = 0; #111 A = 0; B = 0; CARRY_IN = 1; #111
    A = 0; B = 1; CARRY_IN = 0; #111 A = 0; B = 1; CARRY_IN = 1; #111
    A = 1; B = 0; CARRY_IN = 0; #111 A = 1; B = 0; CARRY_IN = 1; #111
    A = 1; B = 1; CARRY_IN = 0; #111 A = 1; B = 1; CARRY_IN = 1; #111
    A = 0; B = 0; CARRY_IN = 0;
    end
endmodule
```

RTL Schematic:

1. Half Adder:

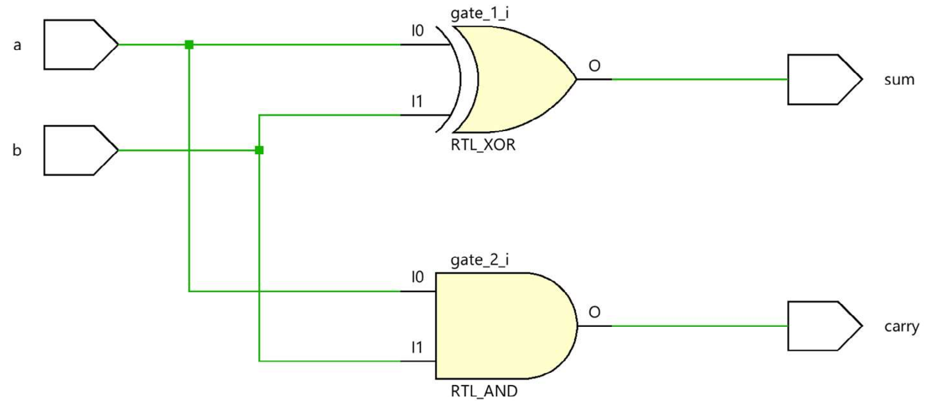

Fig 1.3: Schematic Diagram of Half Adder - Gate Level Modeling

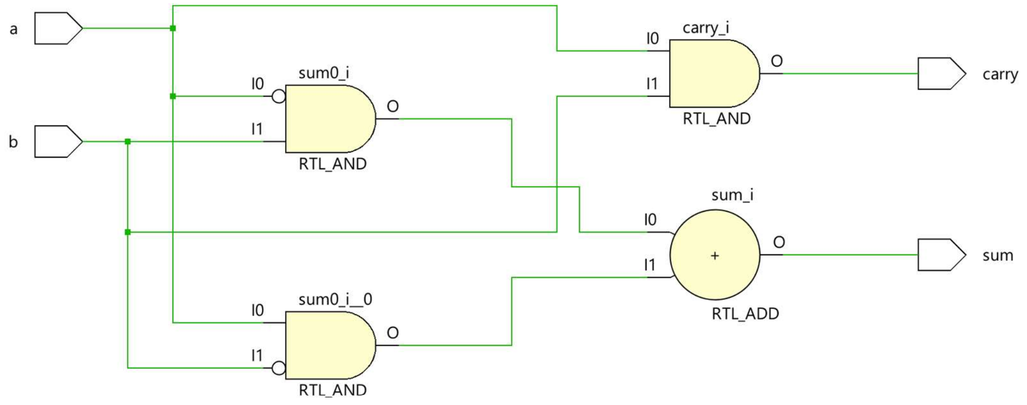

Fig 1.4: Schematic Diagram of Half Adder - Data Flow Level Modeling

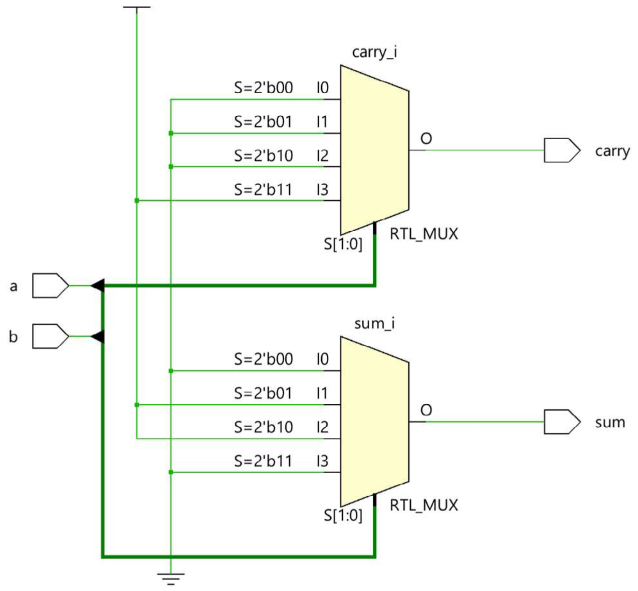

Fig 1.5: Schematic Diagram of Half Adder - Behavioral Modeling

2. Full Adder:

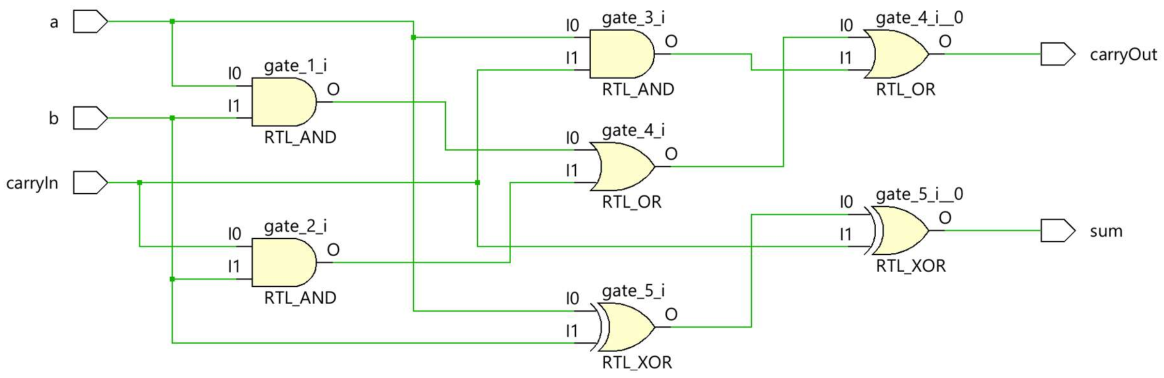

Fig 1.6: Schematic Diagram of Full Adder - Gate Level Modeling

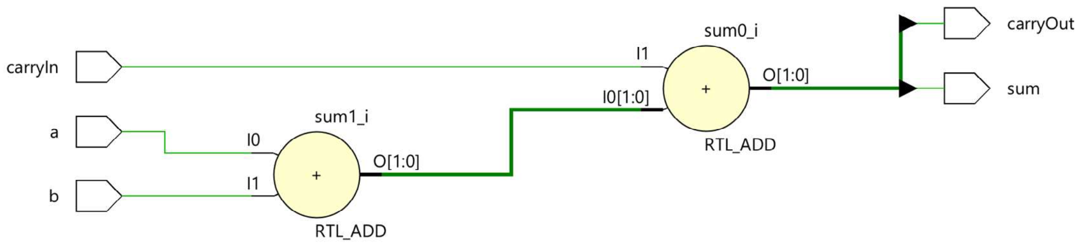

Fig 1.7: Schematic Diagram of Full Adder - Data Flow Level Modeling

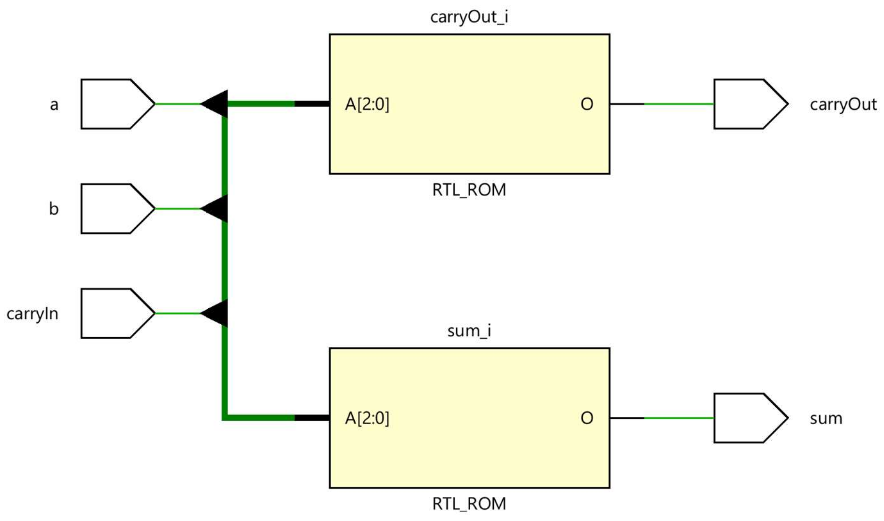

Fig 1.8: Schematic Diagram of Full Adder - Behavioral Modeling

Output Waveform:

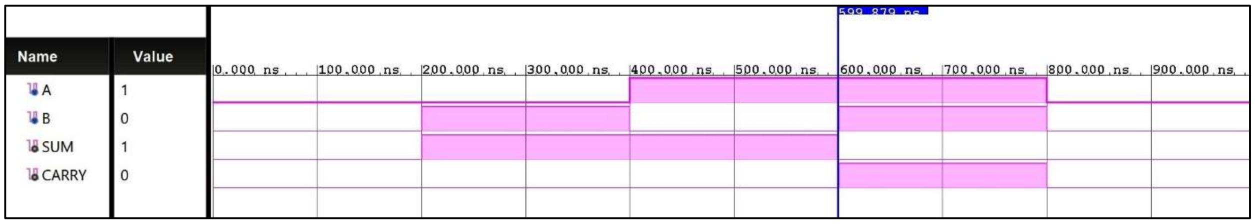

Fig 1.9: Half Adder – Output Waveform

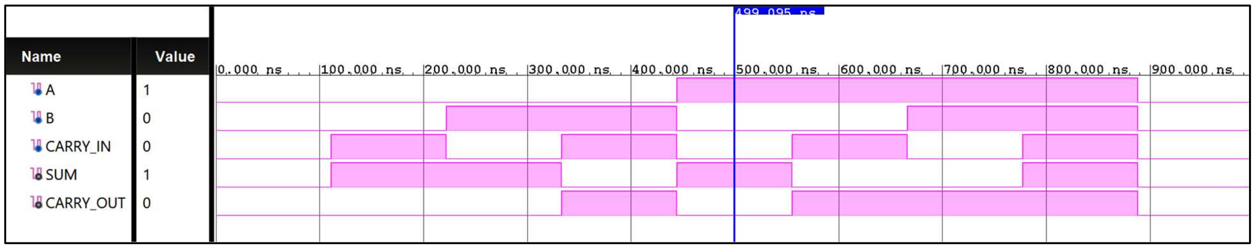

Fig 1.10: Full Adder – Output Waveform

Result: The half adder and full adder circuits has been designed and simulated using different types of modelling like gate level modeling, data flow modeling and behavioral level modeling and the waveforms are observed for the same.