21BDS0340

Abhinav Dinesh Srivatsa

Digital Systems Design Lab

Task 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S. No. | Components | Page No. | Student Check Mark | RA Check Mark |
| 1 | Aim | 2 | ✓ |  |
| 2 | Components required and tools required | 2 | ✓ |  |
| 3 | Half adder diagram with truth table | 2 | ✓ |  |
| 4 | Half adder with gate level in Verilog | 3 | ✓ |  |
| 5 | Half adder with dataflow in Verilog | 3 | ✓ |  |
| 6 | Half adder with behavioural in Verilog | 3 | ✓ |  |
| 7 | Test bench for half adder and snip of output waveform | 4 | ✓ |  |
| 8 | Full adder diagram with truth table | 5 | ✓ |  |
| 9 | Full adder with gate level in Verilog | 5 | ✓ |  |
| 10 | Full adder with dataflow in Verilog | 6 | ✓ |  |
| 11 | Full adder with behavioural in Verilog | 6 | ✓ |  |
| 12 | Test Bench for full adder and snip of output waveform | 6 | ✓ |  |
| 13 | Full subtractor diagram with two half subtractors | 7 | ✓ |  |
| 14 | Full subtractor with structural modelling in Verilog | 8 | ✓ |  |
| 15 | Test bench for full subtractor and snip of output waveform | 8 | ✓ |  |

**Aim**

1. Design half adder and implement in gate-level, dataflow, behavioural, testbench using Verilog.
2. Design full adder with basic gates and implement in gate-level, dataflow, behavioural, testbench using Verilog.
3. Design full subtractor with two half subtractors and implement in any one of the levels, testbench using Verilog.

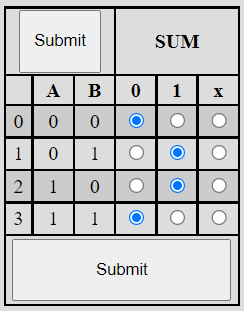
**Components Required**

1. AND, OR, NOT and XOR gates
2. 5V voltage source
3. Led indicator

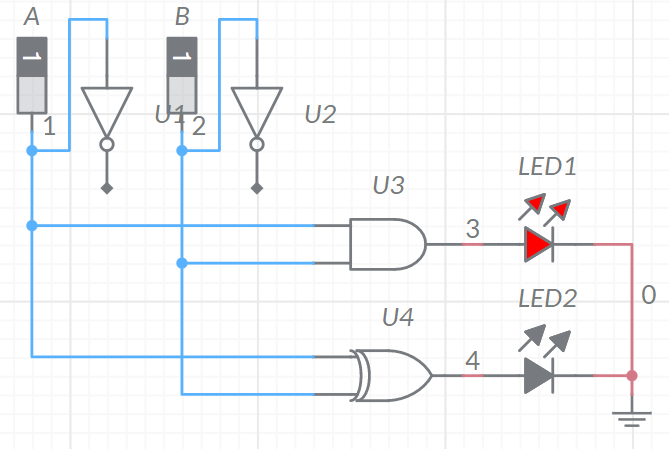
**Tools Required**

1. Multisim simulator
2. Truth table display
   * <http://www.32x8.com/>
3. Modelsim

**Half Adder Diagram with Truth Table**

**A screenshot of a computer

Description automatically generated with low confidence**



**Half Adder with Gate Level in Verilog**

// Gate Level

module halfadder(a, b, sum, carry);

    input a, b;

    output sum, carry;

    xor g1(sum, a, b);

    and g2(carry, a, b);

endmodule

**Half Adder with Dataflow in Verilog**

// Dataflow

module halfadder(a, b, sum, carry);

    input a, b;

    output sum, carry;

    assign sum = a ^ b;

    assign carry = a & b;

endmodule

**Half Adder with Behavioural in Verilog**

// Behavioural

module halfadder(a, b, sum, carry);

    input a, b;

    output reg sum, carry;

    always @(\*);

    begin

    sum = a ^ b;

    carry = a & b;

    end

endmodule

**Test Bench for Half Adder and Snip of Output Waveform**

module test;

    reg a, b;

    wire sum, carry;

    halfadder ha1(a, b, sum, carry);

    initial

    begin

        a = 0; b = 0;

        #100

        a = 0; b = 1;

        #100

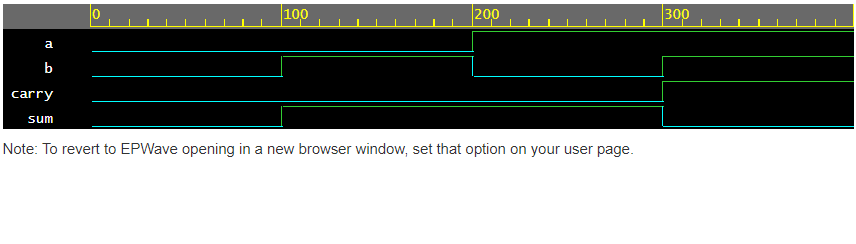
        a = 1; b = 0;

        #100

        a = 1; b = 1;

    end

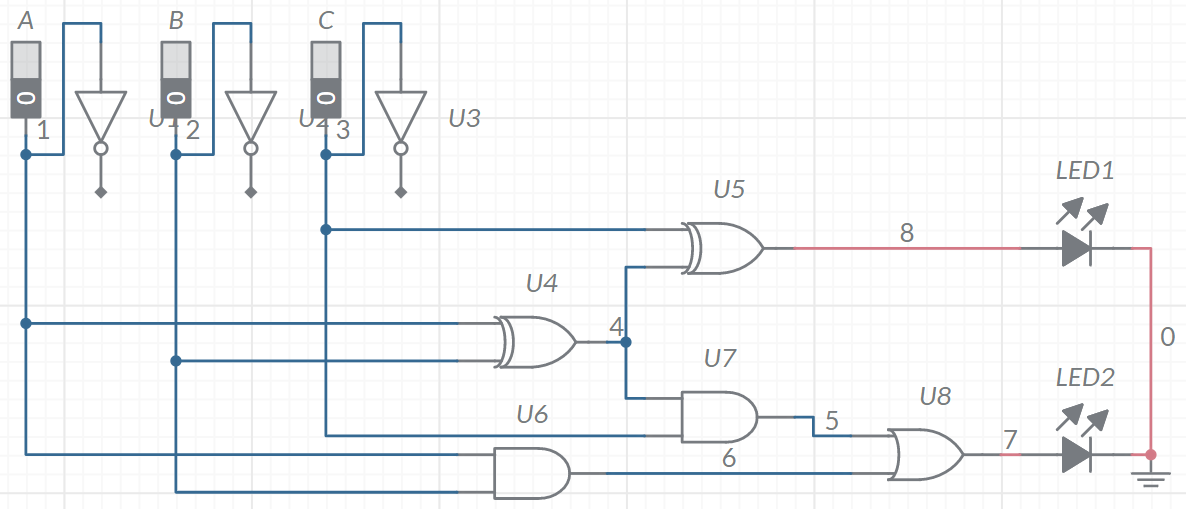
endmodule



**Text

Description automatically generated**Text

Description automatically generated with medium confidence**Full Adder Diagram with Truth Table**



**Full Adder with Gate Level in Verilog**

// Gate Level

module fulladder(a, b, c, sum, carry);

    input a, b, c;

    output sum, carry;

    xor(xor1, a, b);

    and(prod1, a, b);

    and(prod2, xor1, c);

    or(carry, prod1, prod2);

    xor(sum, xor1, c);

endmodule

**Full Adder with Dataflow in Verilog**

// Dataflow

module fulladder(a, b, c, sum, carry);

    input a, b, c;

    output sum, carry;

    assign xor1 = a ^ b;

    assign prod1 = a & b;

    assign prod2 = xor1 & c;

    assign carry = prod1 & prod2;

    assign sum = xor1 ^ c;

endmodule

**Full Adder with Behavioural in Verilog**

// Dataflow

module fulladder(a, b, c, sum, carry);

    input a, b, c;

    output reg sum, carry;

    always @(\*);

    begin

    xor1 = a ^ b;

    prod1 = a & b;

    prod2 = xor1 & c;

    carry = prod1 & prod2;

    sum = xor1 ^ c;

    end

endmodule

**Test Bench for Full Adder and Snip of Output Waveform**

module test;

reg a, b, c;

wire sum, carry;

fulladder fa1(a, b, c, sum, carry);

initial

begin

a = 0; b = 0; c = 0;

#100

a = 0; b = 0; c = 1;

#100

a = 0; b = 1; c = 0;

#100

a = 0; b = 1; c = 1;

#100

a = 1; b = 0; c = 0;

#100

a = 1; b = 0; c = 1;

#100

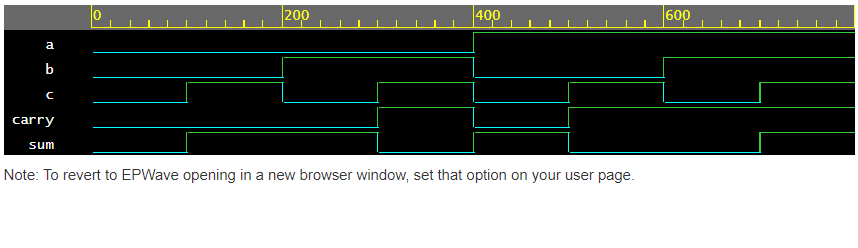
a = 1; b = 1; c = 0;

#100

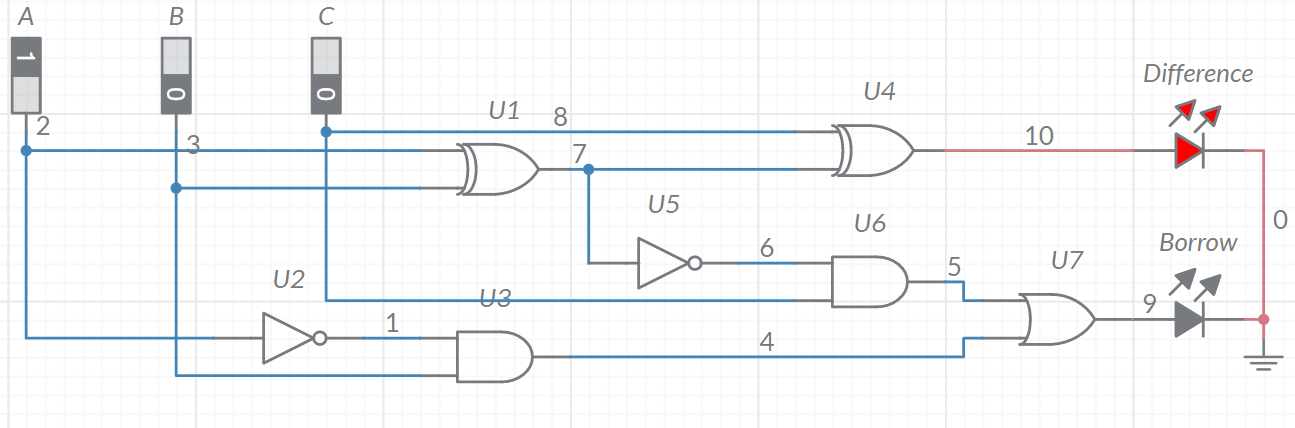
a = 1; b = 1; c = 1;

end

endmodule



**Full Subtractor Diagram with Two Half Subtractors**



**Full Subtractor with Structural Modelling in Verilog**

module halfsubtractor(x, y, b, d);

    input x, y;

    output b, d;

    xor(d, x, y);

    not(notx, x);

    and(b, notx, y);

endmodule

module fullsubtractor(x, y, z, b, d);

    input x, y, z;

    output b, d;

    wire b1, d1, b2;

    halfsubtractor hs1(x, y, b1, d1);

    halfsubtractor hs2(d1, z, b2, d);

    or(b, b1, b2);

endmodule

**Test Bench for Full Subtractor and Snip of Output Waveform**

module test;

    reg a, b, c;

    wire borrow, diff;

    fullsubtractor ha1(a, b, c, borrow, diff);

    initial

    begin

        a = 0; b = 0; c = 0;

        #100

        a = 0; b = 0; c = 1;

        #100

        a = 0; b = 1; c = 0;

        #100

        a = 0; b = 1; c = 1;

        #100

        a = 1; b = 0; c = 0;

        #100

        a = 1; b = 0; c = 1;

        #100

        a = 1; b = 1; c = 0;

        #100

        a = 1; b = 1; c = 1;

    end

endmodule

