B. TECH (CSE) - III SEM

UE23CS251A – DIGITAL DESIGN & COMPUTER ORGANIZATION

PROJECT DOCUMENTATION ON

Design and implement a circuit that counts the number of ones in a given input data.

SEC: C

Team Members:

S.No	SRN	NAME
1.	*PES1UG23CS157*	*Chaitanya M*
2.	*PES1UG23CS166*	*Chinmay Shivanand Muragod*
3.	*PES1UG23CS167*	*Chirag K M*
4.	*PES1UG23CS173*	*Darshith M S*

AUGUST – DECEMBER 2024

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

RR CAMPUS (MAIN CAMPUS)

BENGALURU – 560085, KARNATAKA

TABLE OF CONTENTS

GITHUB LINK :
ilb225112/DDCO_mini_proj: 3rd Sem mini_proj for Digital Design and Computer Organization

HOW CAN WE COUNT NUMBER OF ONE'S IN A GIVEN DATA?

Basic idea:

*The addition of **Single binary bits** in a row can also be understood in terms of the number of one's present*:

for example,

Binary addition	*SUM*	*No. of ones*
0 + 0	0	*Nil (0)*
0 + 1	1	*One (1)*
1 + 1	10 -> 2	*Two (2)*
1 + 1 + 1	11 -> 3	*Three (3)*

. . .

. . .

. . .

0+0=0 indicates that no ones are present,

0+1=1 signifies that only one 1 is present, and

1+1=10 shows that there are two one's present.

1+1+1=11 shows that there are 3 one's present and so on…

So provided with any binary number adding all the individual bits would give us the Total no. of one's present in it.

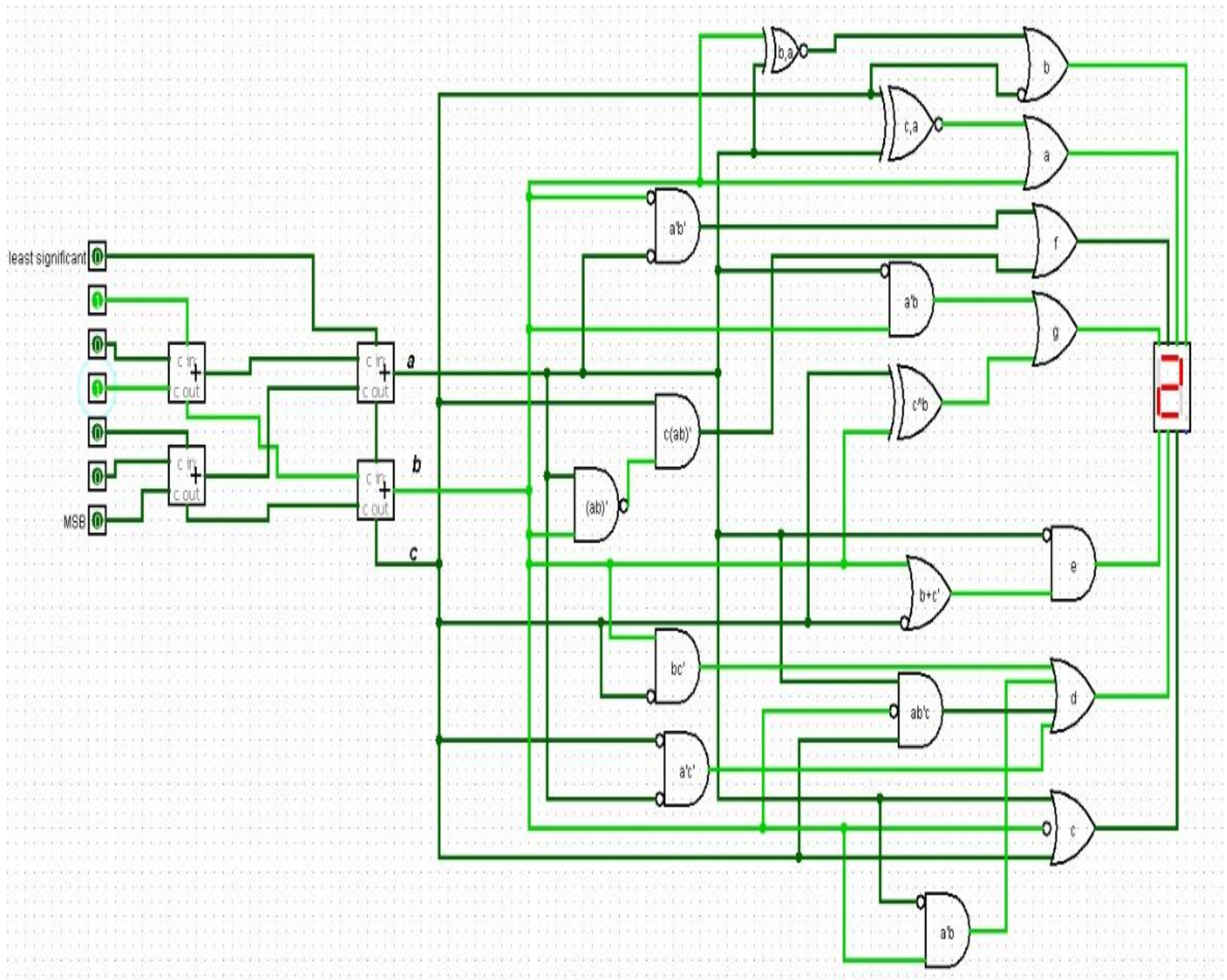
Note: The reason for choosing 7 bits

Initially on choosing 4 bits, the no. of bits required to represent output is 3, Because max number of 1's is 4->100 since 5,6,7 could also be represented using 3 bit we just chose 7 bit. Also, standard being 8 bits one of them would be sign bit, so we go for 7.

Digital Circuit Design

- The circuit design for counting the number of ones in a 7-bit binary input is illustrated in the attached image.
- The image shows how the full adders are interconnected to facilitate the counting process. Use of 7-segment display is being done.
- For the seven-segment display K-map and connections are attached as a GitHub link at last of this file.
- It uses 4-Full adders. The continued part after full adders is for lab implementation

Circuit Diagram:



Verilog Module (norm1.v)

Full Adder Module (fulladder)

This module implements a single full adder, which takes three input bits and produces a sum and a carry-out bit.

```
module fulladder(
        input wire a, b, cin,
        output wire sum, cout
        );

       assign sum = a ^ b ^ cin;
       assign cout = (a & b) | (cin & (a ^ b));
endmodule
```

Sum Module (output)

This module utilizes multiple instances of the full adder to sum the bits of the input data and produce the output.

```
module sum(
    input wire [6:0] inp,
    output wire [2:0] op
    );

    wire c1, c2, s1, s2, c3;

    fulladder a1 (.a(inp[3]), .b(inp[2]), .cin(inp[1]), .sum(s1), .cout(c1));
    fulladder a2 (.a(inp[6]), .b(inp[5]), .cin(inp[4]), .sum(s2), .cout(c2));
    fulladder a3 (.a(s1), .b(s2), .cin(inp[0]), .sum(op[0]), .cout(c3));
    fulladder a4 (.a(c1), .b(c2), .cin(c3), .sum(op[1]), .cout(op[2]));

endmodule
```

Testbench (norm1_tb.v)

Testbench Module:

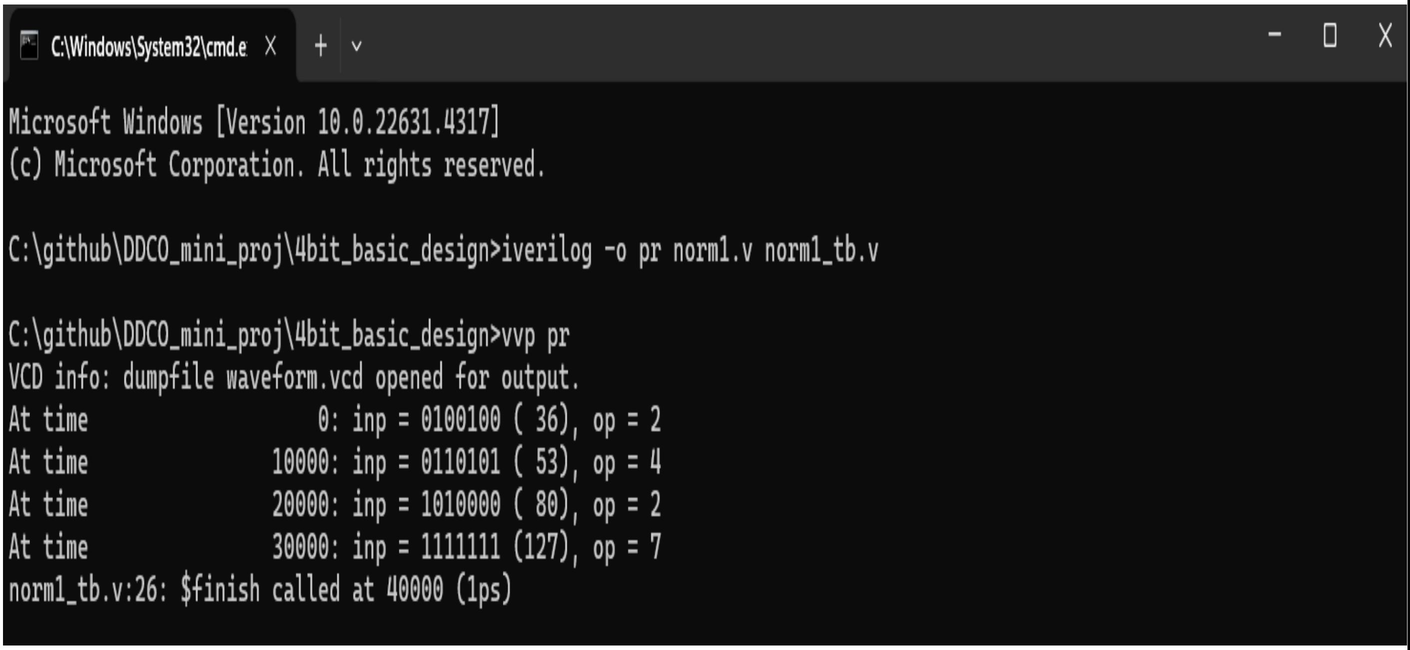
The testbench module simulates various 7-bit input values to validate the functionality of the sum module.

```
`timescale 1ns/1ps

module testbench;
    reg [6:0] inp;
    wire [2:0] op;

    sum uut (
        .inp(inp),
        .op(op)
    );


    initial begin
        inp = 7'b0100100;
        #10;
      inp = 7'b0110101;
        #10;

        inp = 7'b1010000;
        #10;

        inp = 7'b1111111;
        #10;

        $finish;
    end

    initial begin
        $monitor("At time %t: inp = %b (%d), op = %d", $time, inp,inp, op);
    end

    initial begin
        $dumpfile("waveform.vcd");
        $dumpvars(0, testbench);
    end
endmodule
```

Commands to execute:

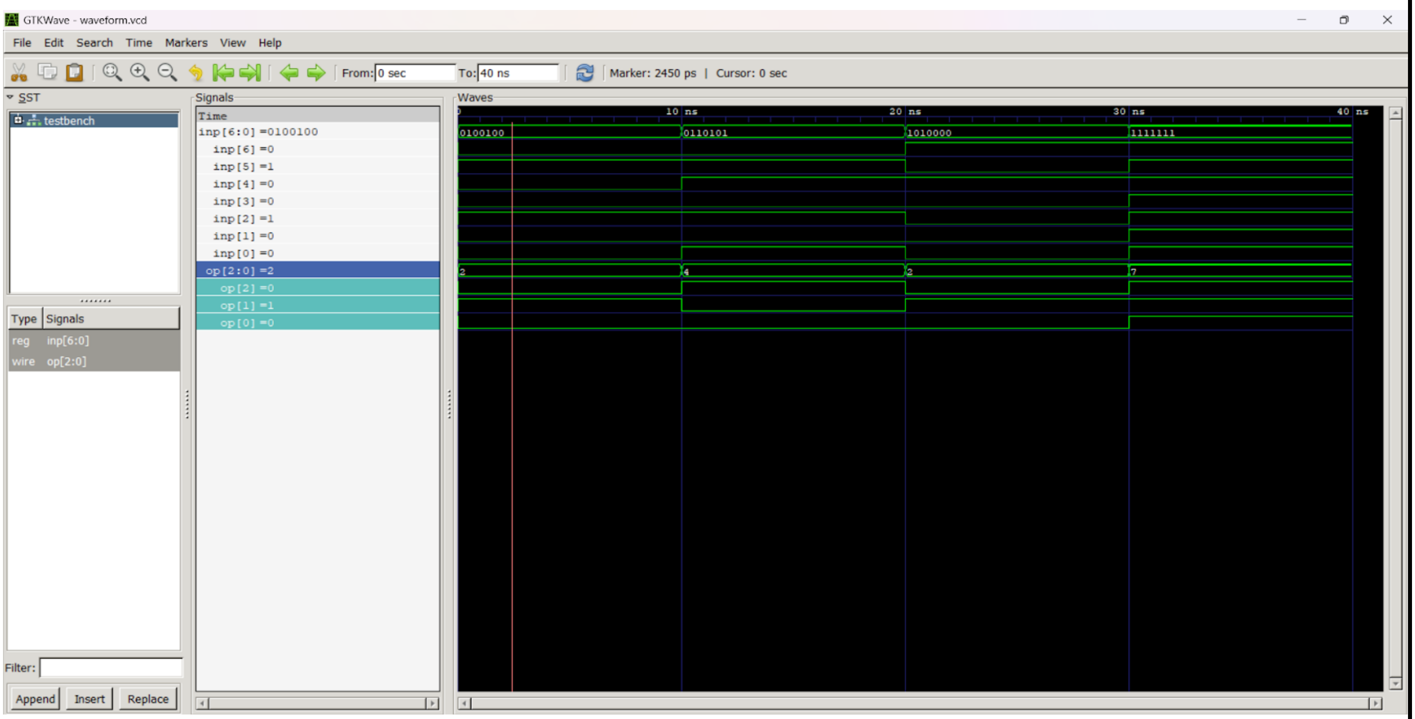
```
Iverilog -o pr filename.v  testbench.v
vvp pr
gtkwave  waveform.vcd
```

Compiler output:



```
C:\Windows\System32\cmd.e

Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\github\DDCO_mini_proj\4bit_basic_design>iverilog -o pr norm1.v norm1_tb.v

C:\github\DDCO_mini_proj\4bit_basic_design>vvp pr
VCD info: dumpfile waveform.vcd opened for output.
At time                    0: inp = 0100100 ( 36), op = 2
At time                10000: inp = 0110101 ( 53), op = 4
At time                20000: inp = 1010000 ( 80), op = 2
At time                30000: inp = 1111111 (127), op = 7
norm1_tb.v:26: $finish called at 40000 (1ps)
```

GTKWave Simulation Results



THANK YOU

GITHUB LINK :

[ilb225112/DDCO_mini_proj: 3rd Sem mini_proj for Digital Design and Computer Organization]