COAL Lab # 01

Verilog (HDL) Programing Fundamentals

Name: Muhammad Usman

Class: BSCS 3C1

Reg No: cs211208

<u>Lab Exercises:</u>

Task 01

Literature Review:

A combinational circuit is the digital logic circuit in which the output depends on the combination of inputs at that point of time with total disregard to the past state of the inputs. The digital logic gate is the building block of combinational circuits. a combinational circuit could be used to add any number of inputs, or to subtract them, or to perform other mathematical operations.

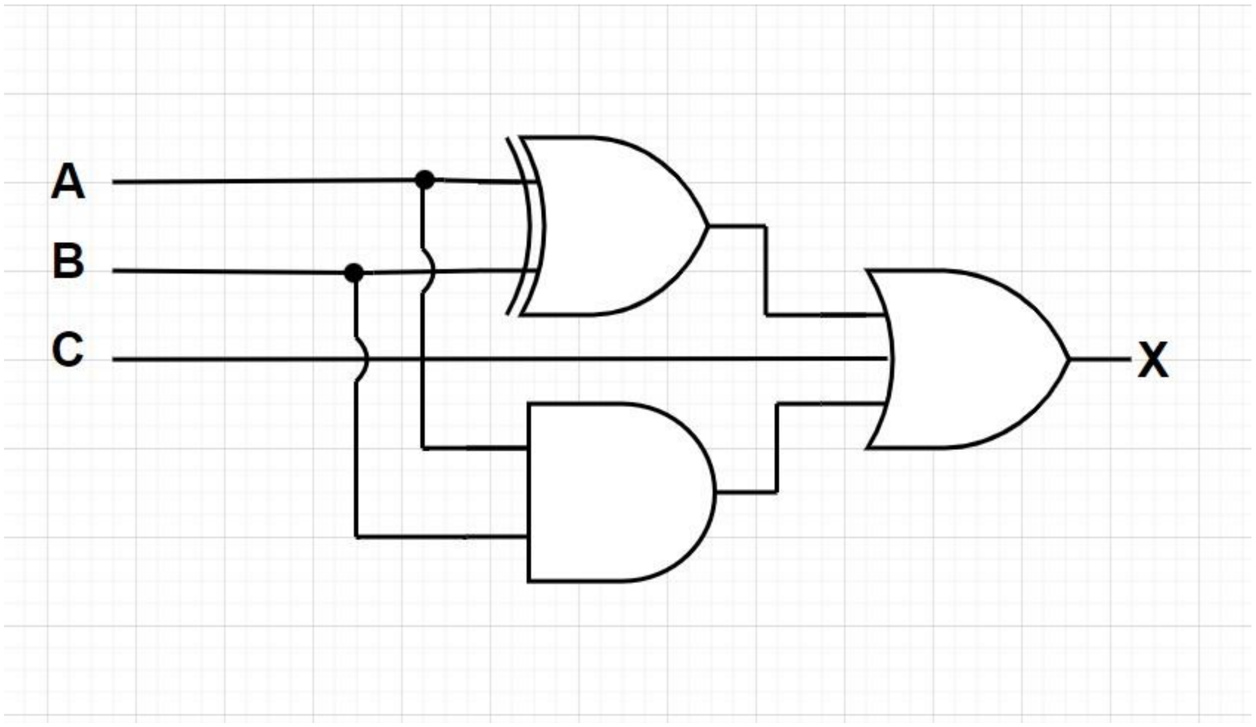
Task:

Implement combinational circuit using Verilog.

Truth Table:

A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Block Diagram:



Boolean Equation:

$$Q = (A \oplus B) + (A.B) + C$$

Verilog Code:

design.v

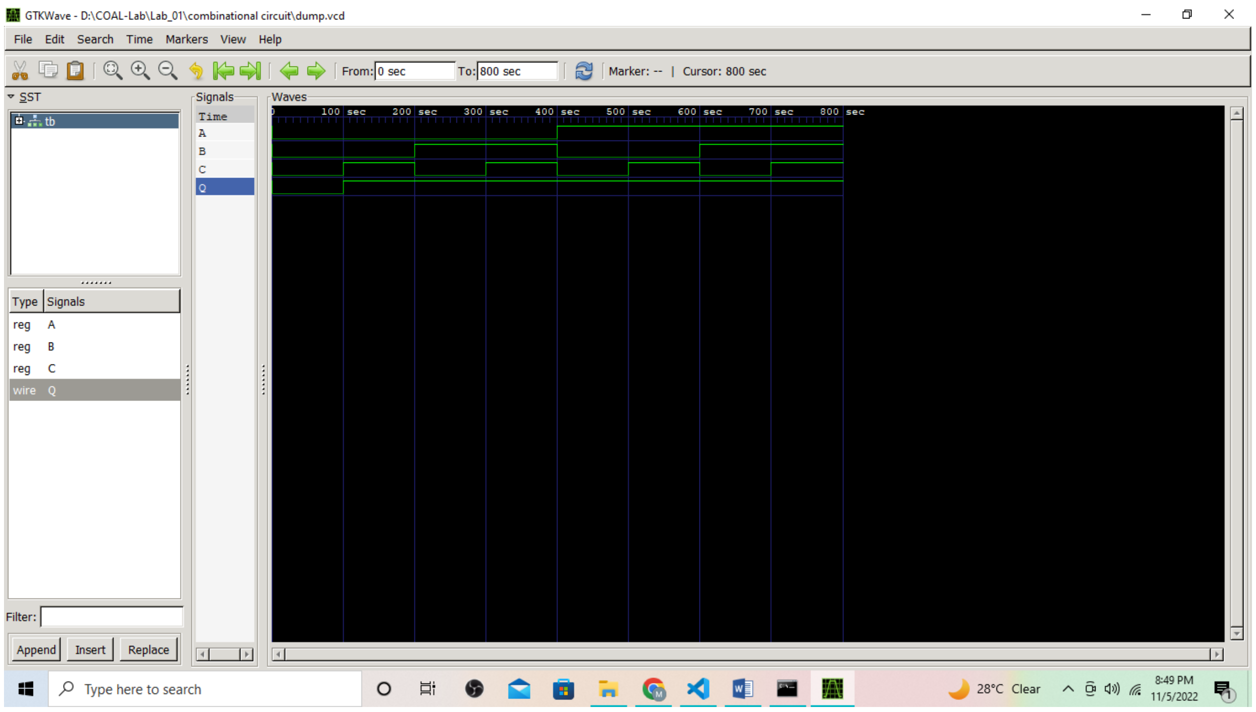
```
module combinational_circuit(A,B,C,Q);
```

```
    input A,B,C;
    output Q;

    // interim signals
    wire A_xor_B,A_and_B;

    // logic designing
    assign A_xor_B = A ^ B;
    assign A_and_B = A & B;
    assign Q = C | A_and_B | A_xor_B;

endmodule
```

testbench.v

```
module tb();

    reg A,B,C;
    wire Q;
    // module declare
    combinational_circuit dut (.A(A),.B(B),.C(C),.Q(Q));

    initial begin
      A <= 1'b0;
      B <= 1'b0;
      C <= 1'b0;
      #100

      A <= 1'b0;
      B <= 1'b0;
      C <= 1'b1;
      #100

      A <= 1'b0;
      B <= 1'b1;
      C <= 1'b0;
      #100

      A <= 1'b0;
      B <= 1'b1;
      C <= 1'b1;
      #100
      A <= 1'b1;
      B <= 1'b0;
      C <= 1'b0;
      #100

      A <= 1'b1;
      B <= 1'b0;
      C <= 1'b1;
      #100

      A <= 1'b1;
      B <= 1'b1;
      C <= 1'b0;
      #100

      A <= 1'b1;
      B <= 1'b1;
      C <= 1'b1;
      #100
    end

    initial begin
        $dumpfile("dump.vcd");
        $dumpvars(0);
    end

endmodule
```

Waveforms:


GTKWave - D:\COAL-Lab\Lab_01\combinational circuit\dump.vcd
File Edit Search Time Markers View Help
From: 0 sec To: 800 sec Marker: -- | Cursor: 800 sec
SST
tb
Signals
Time
A
B
C
Q
Waves
100 sec 200 sec 300 sec 400 sec 500 sec 600 sec 700 sec 800 sec
Type Signals
reg A
reg B
reg C
wire Q
Filter:
Append Insert Replace
Type here to search
28°C Clear
8:49 PM
11/5/2022

Task 02

Literature Review:

The sequential circuit is a special type of circuit that has a series of inputs and outputs. The outputs of the sequential circuits depend on both the combination of present inputs and previous outputs. The previous output is treated as the present state. They possess the ability to store the output. They also include a time sequence of inputs and internal states. The famously known sequential elements are D-Flip Flops, J-K Flip Flop, S-R Latch , etc.

Task:

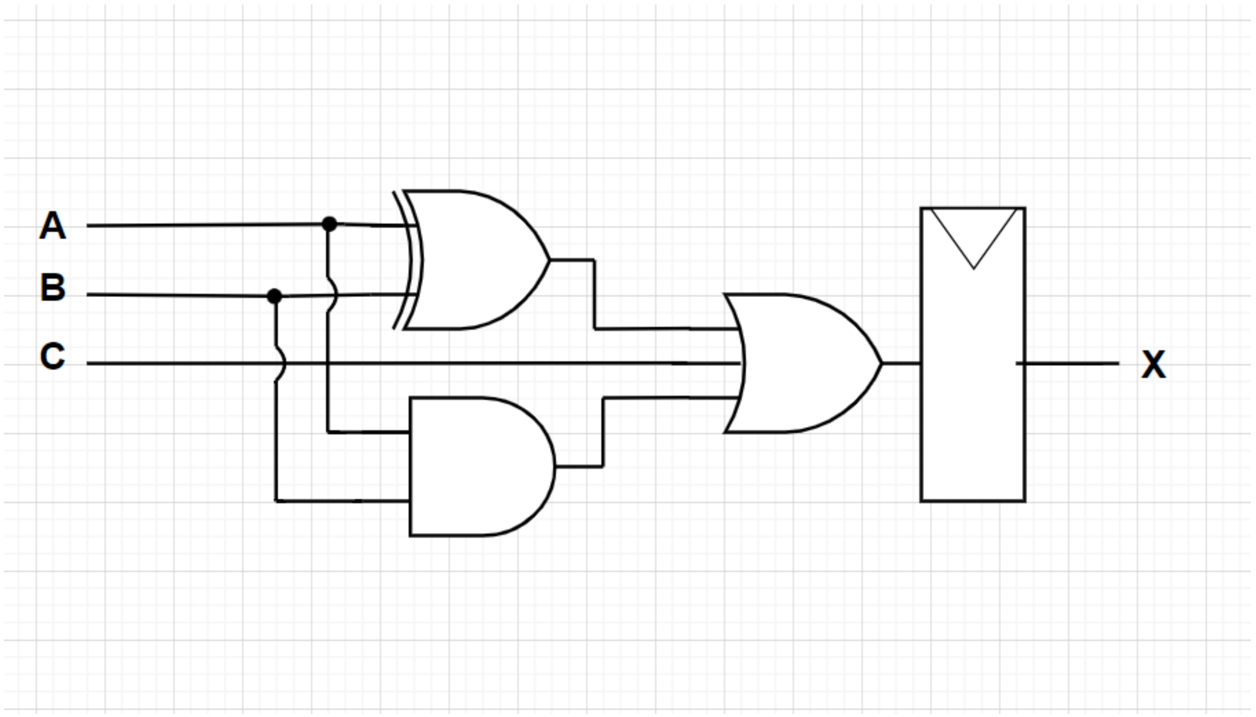
Implement Sequential circuit using Verilog.

Truth Table:

A	B	C	X
0	0	0	0

0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Block Diagram:



Boolean Equation:

$$Q = (A \oplus B) + (A.B) + C$$

Verilog Code:

design.v

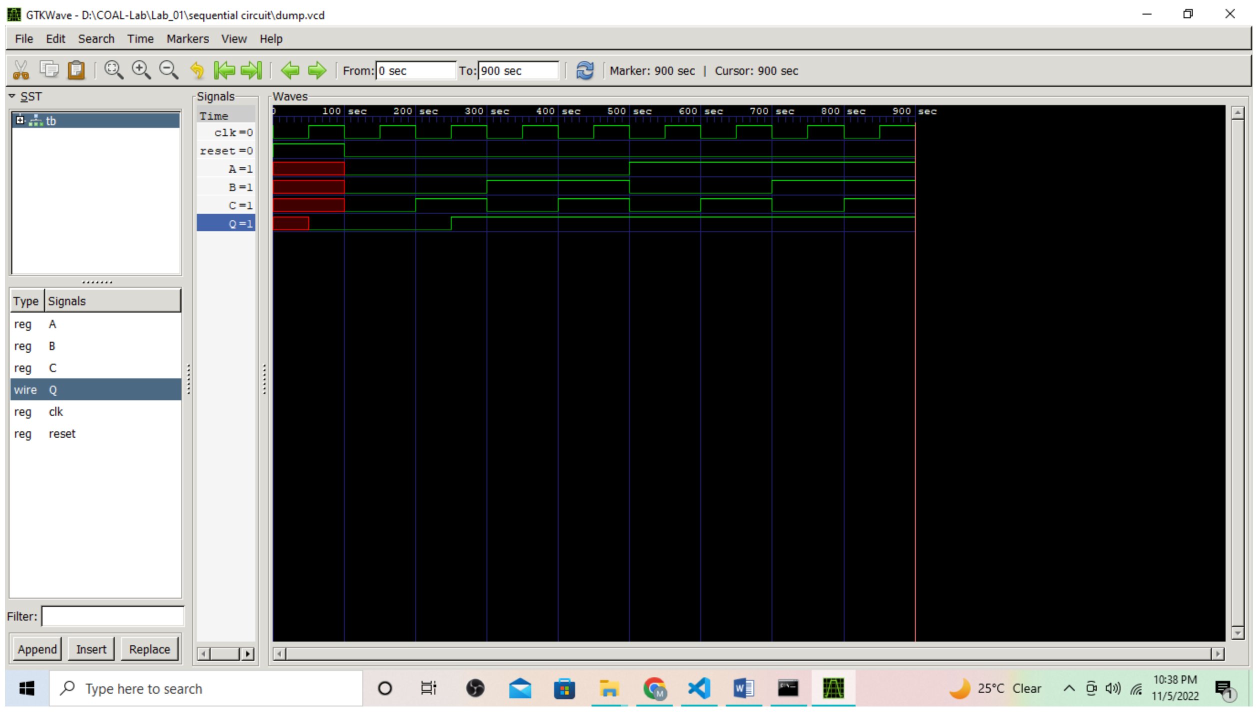
```verilog
module sequential_circuit(A,B,C,clk,reset,Q);

    input A,B,C,clk,reset;
    output Q;

    // interim signals
    wire A_xor_B , A_and_B , A_or_B_C;
    reg flop;

    // logic designing
    assign A_xor_B = A ^ B;
    assign A_and_B = A & B;
    assign A_or_B_C = C | A_and_B | A_xor_B;

    always @(posedge clk) begin
        if(reset == 1'b1)
        begin
            flop <= 1'b0;
        end
        else begin
            flop <= A_or_B_C;
        end
    end

    assign Q = flop;

endmodule
```

testbench.v

```verilog
module tb();

    reg A,B,C,clk,reset;
    wire Q;
    // module declare
    sequential_circuit dut (.A(A),.B(B),.C(C),.clk(clk),.reset(reset),.Q(Q));

    always begin
      clk <= 1'b0;
      #50;
      clk <= 1'b1;
      #50;
    end

    initial begin
      reset <= 1'b1;
      #100;
      reset <= 1'b0;
      A <= 1'b0;
      B <= 1'b0;
      C <= 1'b0;
      #100;

      A <= 1'b0;
      B <= 1'b0;
      C <= 1'b1;
      #100;

      A <= 1'b0;
      B <= 1'b1;
      C <= 1'b0;
      #100;

      A <= 1'b0;
      B <= 1'b1;
      C <= 1'b1;
      #100;

      A <= 1'b1;
      B <= 1'b0;
      C <= 1'b0;
      #100;

      A <= 1'b1;
      B <= 1'b0;
      C <= 1'b1;
      #100;

      A <= 1'b1;
      B <= 1'b1;
      C <= 1'b0;
      #100;

      A <= 1'b1;
```

```
        B <= 1'b1;
        C <= 1'b1;
        #100;
        $finish;
    end

    initial begin
        $dumpfile("dump.vcd");
        $dumpvars(0);
    end

endmodule
```

Waveforms:



GTKWave - D:\COAL-Lab\Lab_01\sequential circuit\dump.vcd
File Edit Search Time Markers View Help
From: 0 sec To: 900 sec
Marker: 900 sec | Cursor: 900 sec
SST
tb
Type Signals
reg A
reg B
reg C
wire Q
reg clk
reg reset
Filter:
Append Insert Replace
Signals
Time
clk =0
reset =0
A =1
B =1
C =1
Q =1
Waves
0 100 sec 200 sec 300 sec 400 sec 500 sec 600 sec 700 sec 800 sec 900 sec
Type here to search
25°C Clear
10:38 PM
11/5/2022

In Lab Task:

Task 03

Literature Review:

Half adder is a combinational arithmetic circuit that adds two numbers and produces a sum bit (s) and carry bit (c) both as output. The addition of 2 bits is done using a combination circuit called a Half adder. The input variables are augend and addend bits and output variables are sum & carry bits.

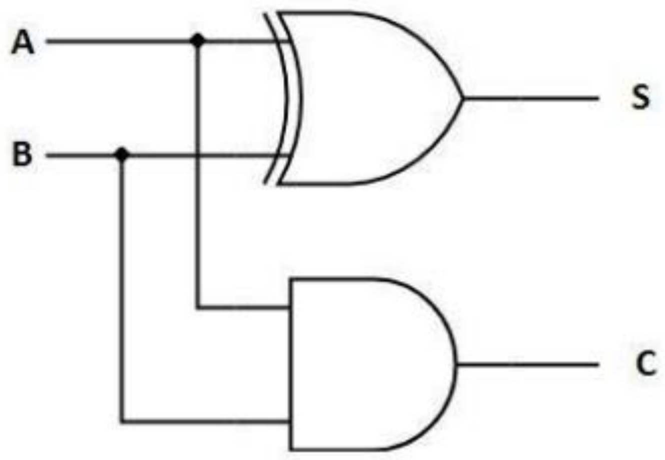
Task:

Implement Half adder using Verilog.

Truth Table:

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Block Diagram:



Boolean Equation:

$$S = A \oplus B$$

$$C = A.B$$

Verilog Code:

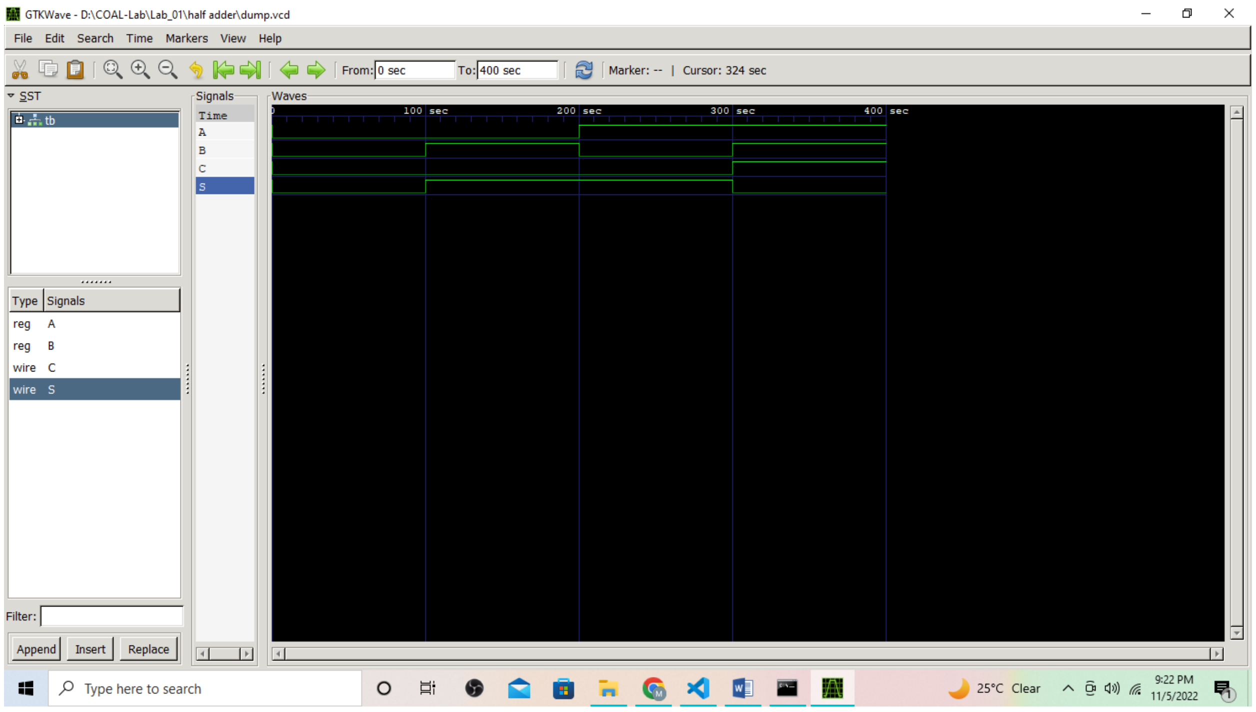
design.v

```
module half_adder(A,B,S,C);

    input A,B;
    output S,C;

    assign S = A ^ B;
    assign C = A & B;


endmodule
```

testbench.v

```verilog
module tb();

    reg A,B;
    wire S,C;
    // module declare
    half_adder dut (.A(A),.B(B),.S(S),.C(C));

    initial begin
      A <= 1'b0;
      B <= 1'b0;
      #100;

      A <= 1'b0;
      B <= 1'b1;
      #100;

      A <= 1'b1;
      B <= 1'b0;
      #100;

      A <= 1'b1;
      B <= 1'b1;
      #100;

    end

    initial begin
        $dumpfile("dump.vcd");
        $dumpvars(0);
    end

endmodule
```

Waveforms:



Task 04

Literature Review:

2 bit counter is a sequential circuit which stores the previous output in memory element then use it to increment the count. On every clock cycle, under defined conditions, one will add in the input and it will be stored in memory element and then for next cycle it will repeat the process until it reaches the limit of bits storage.

Task:

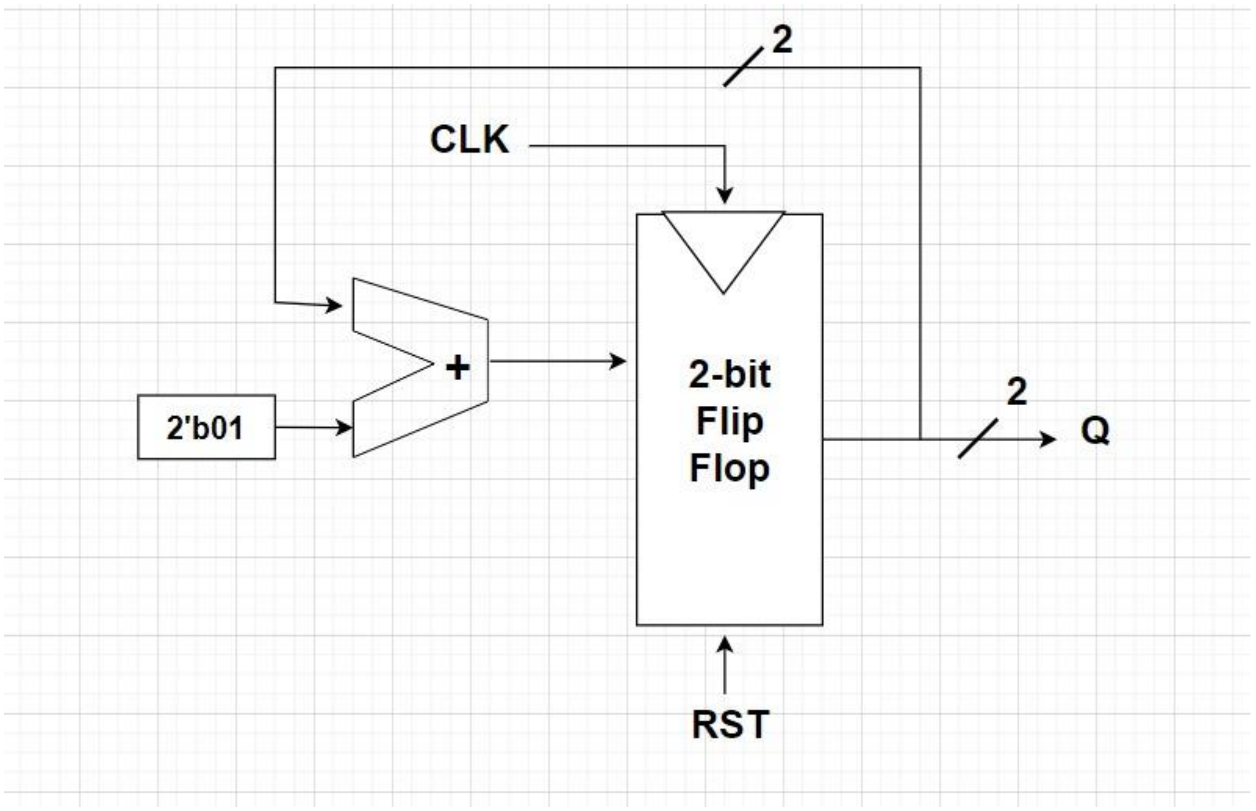
Implement 2 bit counter using Verilog.

Truth Table:

Clock	Q1	Q2
initial	0	0

1	0	1
2	1	0
3	1	1
4	0	0

Block Diagram:



Verilog Code:

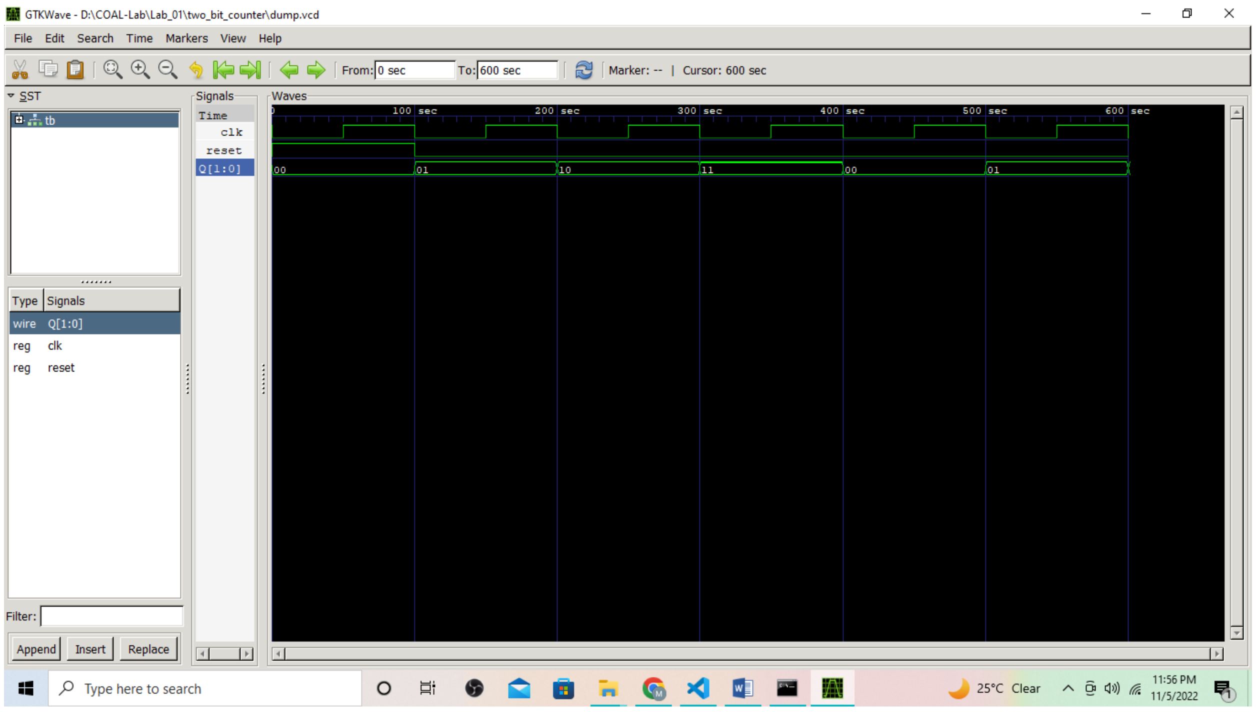
design.v

```
module two_bit_adder(input clk,reset,output [1:0]Q);


    reg [1:0] flop;

    always @(negedge clk) begin
        if(reset == 1'b1)
        begin
            flop <= 2'b00;
        end
        else begin
            flop <= flop + 2'b01;
        end
    end
    assign Q = flop;

endmodule
```

testbench.v

```
module tb();

    reg clk,reset;
    wire [1:0 ]Q;
    // module declare
   two_bit_adder dut (.clk(clk),.reset(reset),.Q(Q));

    always begin
      clk <= 1'b0;
      #50;
      clk <= 1'b1;
      #50;
    end

    initial begin
      reset <= 1'b1;
      #100;
      reset <= 1'b0;
      #500;
      $finish;
    end

    initial begin
        $dumpfile("dump.vcd");
        $dumpvars(0);
    end

endmodule
```

Waveforms:



<u>Post Lab Task:</u>

Task 05

Literature Review:

A combinational circuit is the digital logic circuit in which the output depends on the combination of inputs at that point of time with total disregard to the past state of the inputs. The digital logic gate is the building block of combinational circuits. a combinational circuit could be used to add any number of inputs, or to subtract them, or to perform other mathematical operations.

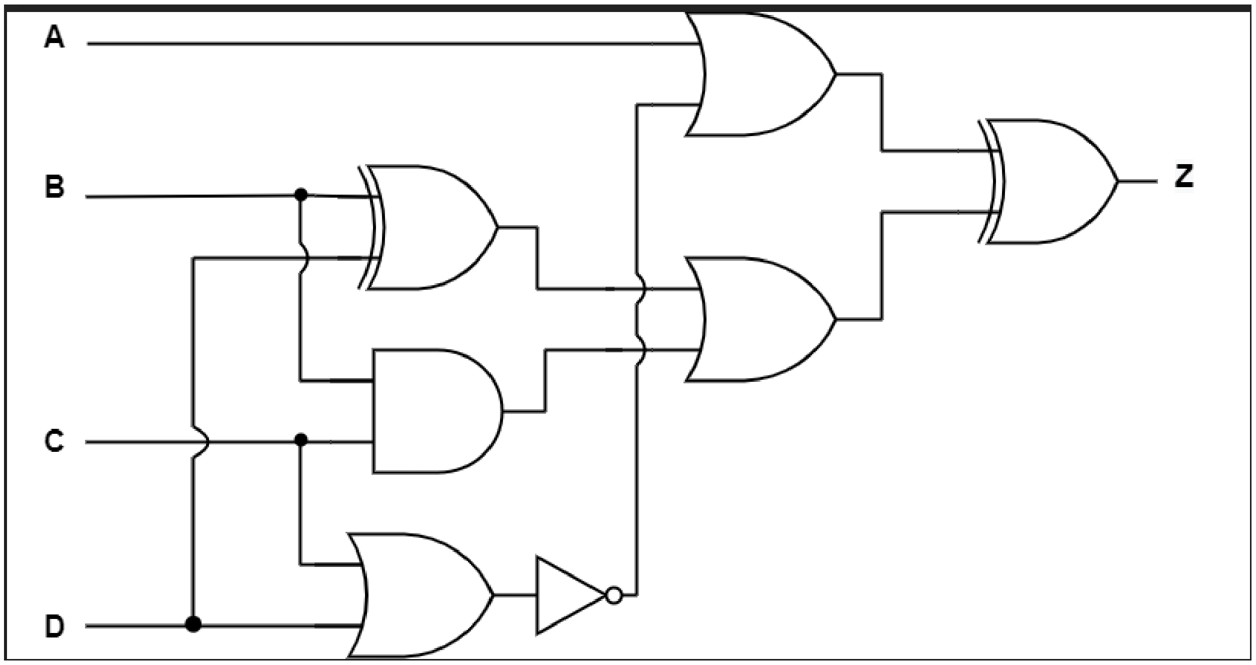
Task:

Implement combinational circuit using Verilog.

Truth Table:

A	B	C	D	X
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Block Diagram:



Boolean Equation:

$$Z = (A+(\overline{C+D})) \oplus ((B \oplus D)+B.C)$$

Verilog Code:

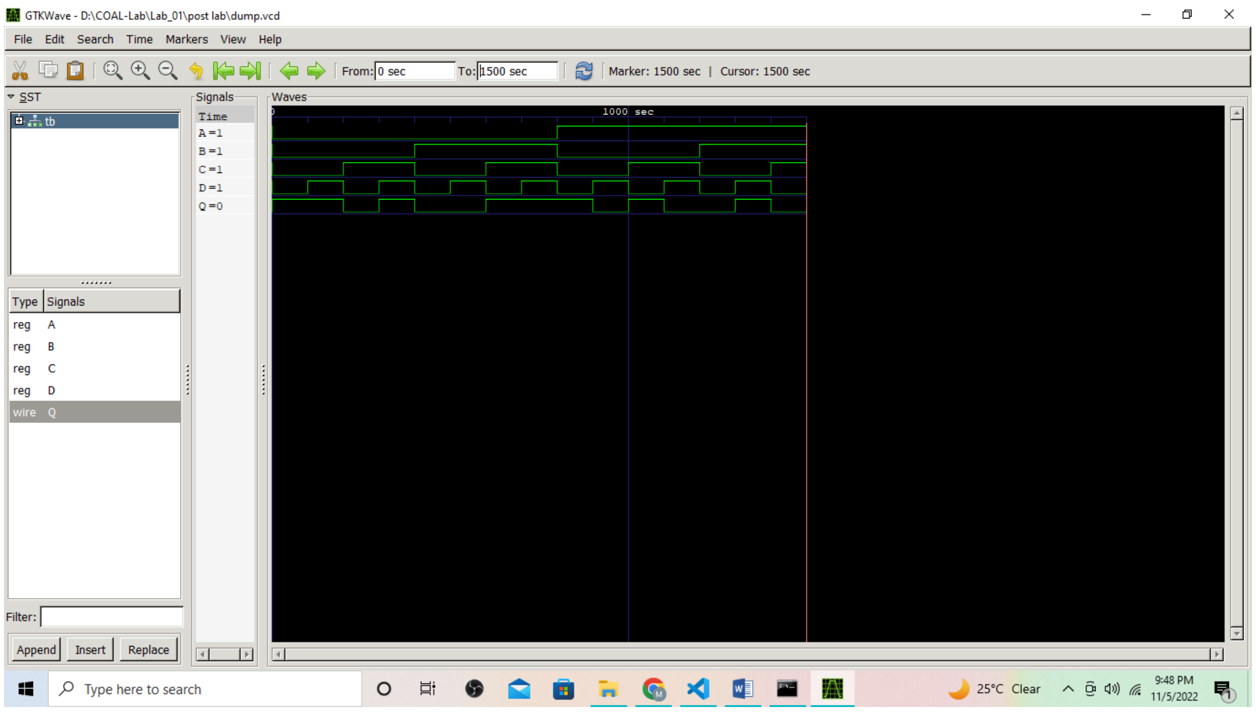
design.v

```
module post_lab(A,B,C,D,Q);

    input A,B,C,D;
    output Q;
    assign Q = (A|(!(C|D)))^((B^D)|(B&C));

endmodule
```

testbench.v

```
module tb();

    reg A,B,C,D;
    wire Q;

    post_lab dut(.A(A),.B(B),.C(C),.D(D),.Q(Q));

    initial begin
        A <= 1'b0;
        B <= 1'b0;
        C <= 1'b0;
        D <= 1'b0;
        #100;

        A <= 1'b0;
        B <= 1'b0;
        C <= 1'b0;
        D <= 1'b1;
        #100;

        A <= 1'b0;
        B <= 1'b0;
        C <= 1'b1;
        D <= 1'b0;
        #100;

        A <= 1'b0;
        B <= 1'b0;
        C <= 1'b1;
        D <= 1'b1;
        #100;

        A <= 1'b0;
        B <= 1'b1;
        C <= 1'b0;
        D <= 1'b0;
        #100;

        A <= 1'b0;
        B <= 1'b1;
        C <= 1'b0;
        D <= 1'b1;
        #100;
        A <= 1'b0;
        B <= 1'b1;
        C <= 1'b1;
        D <= 1'b0;
        #100;

        A <= 1'b0;
        B <= 1'b1;
        C <= 1'b1;
        D <= 1'b1;
        #100;

        A <= 1'b1;
        B <= 1'b0;
        C <= 1'b0;
        D <= 1'b0;
        #100;

        A <= 1'b1;
        B <= 1'b0;
        C <= 1'b0;
        D <= 1'b1;
        #100;

        A <= 1'b1;
        B <= 1'b0;
        C <= 1'b1;
        D <= 1'b0;
        #100;

        A <= 1'b1;
        B <= 1'b0;
        C <= 1'b1;
        D <= 1'b1;
        #100;

        A <= 1'b1;
        B <= 1'b1;
        C <= 1'b0;
        D <= 1'b0;
        #100;

        A <= 1'b1;
        B <= 1'b1;
        C <= 1'b0;
        D <= 1'b1;
        #100;

        A <= 1'b1;
        B <= 1'b1;
        C <= 1'b1;
        D <= 1'b0;
        #100;

        A <= 1'b1;
        B <= 1'b1;
        C <= 1'b1;
        D <= 1'b1;
        #100;
        $finish;
    end

    initial begin
        $dumpfile("dump.vcd");
        $dumpvars(0);
    end

endmodule
```

Waveforms:



Type	Signals
reg	A
reg	B
reg	C
reg	D
wire	Q