ASSIGNMENT 1
CS666: HARDWARE SECURITY OF INTERNET OF THINGS

Group G2

August 2023

1 GROUP MEMBERS

1. VIKRANT CHAUHAN : MS CYBERSERCURITY : 231110407
2. SOUVIK MUKHERJEE : MS CYBERSECURITY : 231110405
3. VALETI LOKESH : MS CYBERSECURITY : 231110406
4. VISHAL KUMAR : MTECH CYBERSECURITY : 231110058
5. M DHILIPKUMAR : MTECH CYBERSECURITY : 231110026

All group members have equally worked in this project and contributed equally in this project.

2 Helping instruction for running the code files

For question Number 1:-
Step 1: cd ques1
Step 2: iverilog -o full_adder.vvp full_adder_test_test.v
Step 3: vvp full_adder.vvp
Step 4: See output
Step 5: iverilog -o eight_bit_full_adder.vvp eight_bit_full_adder_test.v
Step 6:vvp eight_bit_full_adder.vvp
Step 7: See Output

For question Number 2:-
Step 1: cd ques2
Step 2: iverilog -o multiplier.vvp multiplier_test.v
Step 3: vvp multiplier.vvp

For question Number 3:-
Step 1: cd ques3
Step 2: iverilog -o d_ff.vvp d_ff_test.v
Step 3: vvp d_ff.vvp
Step 4: See output
Step 5: iverilog -o john.vvp john_test.v
Step 6:vvp john.vvp
Step 7: See Output

3 Question 1

3.1 Question

Design a Verilog module for the 1-bit full adder that should have three inputs: A, B, and Cin (carry-in), and two outputs: Sum and Cout (carry-out). Now, using instantiation of the 1-bit full adder module, design a 8-bit full adder that should have three inputs: A (8-bit input), B (8-bit input), and Cin (1-bit carry-in), and two outputs: Sum (8-bit output) and Cout (1-bit carry-out)

3.2 Solution Methodology

A 1-bit Full Adder is used to add 2 one bit numbers.It take the two numbers as input as well as a cin(carry in) as input and then gives Sum and Cout(carry out as output).

For creating one bit adder we make use of the following formulaes: Sum=A XOR B XOR Cin Cout=A.B + B.CIN + A.CIN

3.2.1 Steps to create a One Bit Full Adder

To Get Sum:-
Firstly we will pass the input numbers A and B and Cin in an XOR gate. The result obtained is our sum.
To get Cout:-
For this we we first calculate A AND B, A AND Cin and B AND Cin. The obtained result is passed through OR gates to get our final output of Cout.

3.2.2 Circuit Diagrams and Terminal snippets For One Bit Full Adder

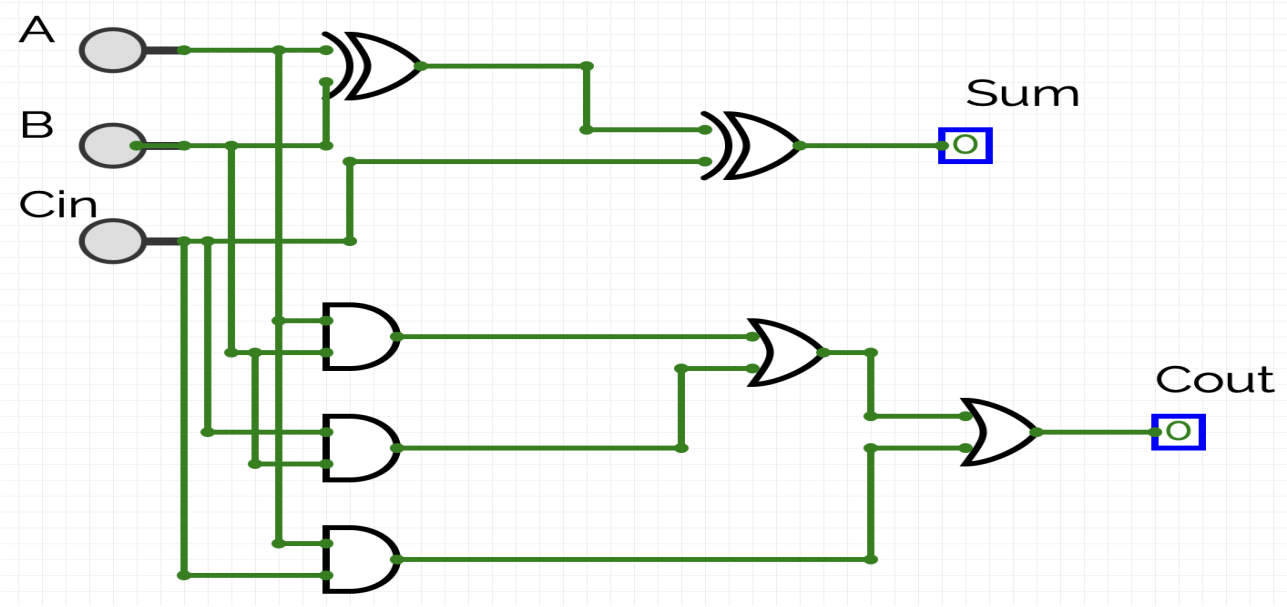

Figure 1: Circuit Diagram Of One Bit Full Adder

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

Figure 2: Truth Table Of Full Adder

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

vikrantchauhan@Vikrants-MacBook-Air ques1 % iverilog -o full_adder.v full_adder_test.v
vikrantchauhan@Vikrants-MacBook-Air ques1 % vvp full_adder.vvp
VCD info: dumpfile full_adder.vcd opened for output.
                   0 A=x,B=x,Cin=x,Sum=x,Cout=x
                   1 A=0,B=0,Cin=0,Sum=0,Cout=0
                   2 A=0,B=0,Cin=1,Sum=1,Cout=0
                   7 A=0,B=1,Cin=0,Sum=1,Cout=0
                  12 A=0,B=1,Cin=1,Sum=0,Cout=1
                  17 A=1,B=0,Cin=0,Sum=1,Cout=0
                  22 A=1,B=0,Cin=1,Sum=0,Cout=1
                  27 A=1,B=1,Cin=0,Sum=0,Cout=1
                  32 A=1,B=1,Cin=1,Sum=1,Cout=1
vikrantchauhan@Vikrants-MacBook-Air ques1 %
```

Figure 3: Output Of One Bit Full Adder

3.2.3 Steps to create a 8 Bit Full Adder

Using the one bit adder which we had made in the previous we make the 8 bit full adder.
For making the 8 bit full adder the steps are:-
Step 1:- we create 8 instances of one bit full adder.Now to find the sum of eight bit numbers we pass each bit of both the numbers to one-one one bit full adders.
Step 2:- Now in the first one bit full adder we pass Cin as 0. For the other 1 bit full adders we pass the Cin as the Cout of their previous one bit full adder.
Step 3:- In this way we obtain the desired result of making a 8 bit full adder

3.2.4 Circuit Diagrams and Terminal snippets For Eight Bit Full Adder

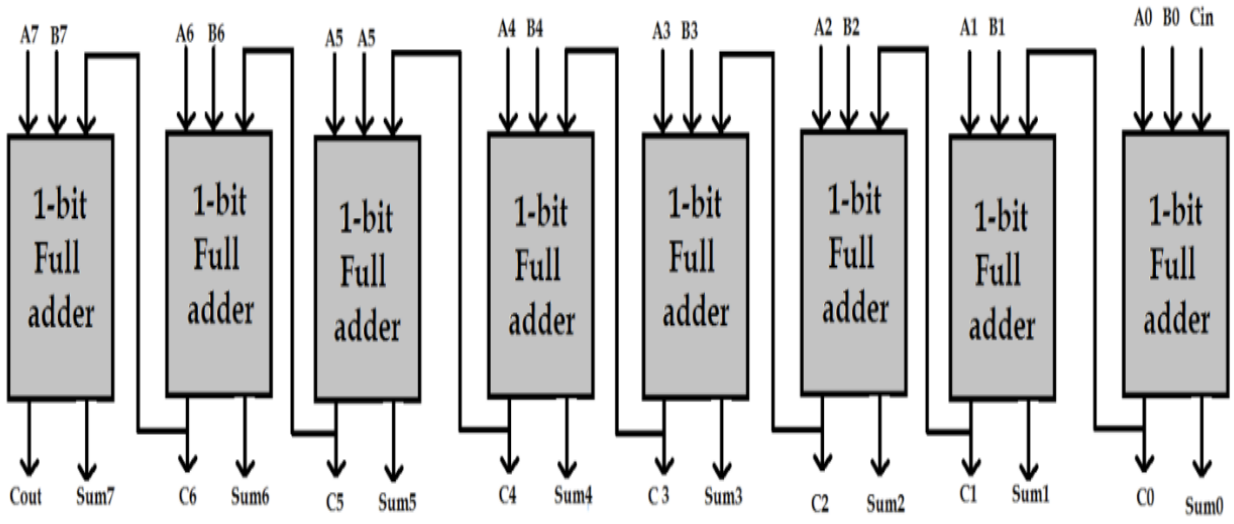

Figure 4: Circuit Diagram Of Eight Bit Full Adder

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

vikrantchauhan@Vikrants-MacBook-Air ques1 % iverilog -o eight_bit_full_adder.vvp eight_
vikrantchauhan@Vikrants-MacBook-Air ques1 % vvp eight_bit_full_adder.vvp
VCD info: dumpfile eight_full_adder.vcd opened for output.
                  0 A=xxxxxxxx,B=xxxxxxxx,Sum=xxxxxxxx,Cout=x
                  5 A=00000010,B=00000011,Sum=00000101,Cout=0
vikrantchauhan@Vikrants-MacBook-Air ques1 %
```

Figure 5: Output Of Eight Bit Full Adder

4 Question 2

4.1 Question

Write a Verilog module for 4-bit multiplication? The module should have two 4-bit inputs, A and B, and produce an 8-bit product, P. The computational workflow is shown in Figure 1. Firstly, generate a partial product matrix with dimensions of 4x8. Then, compress the matrix to a 2x8 matrix using a 1-bit full adder module that was designed earlier. Finally, generate the final result using an 8-bit adder module that was also designed earlier.

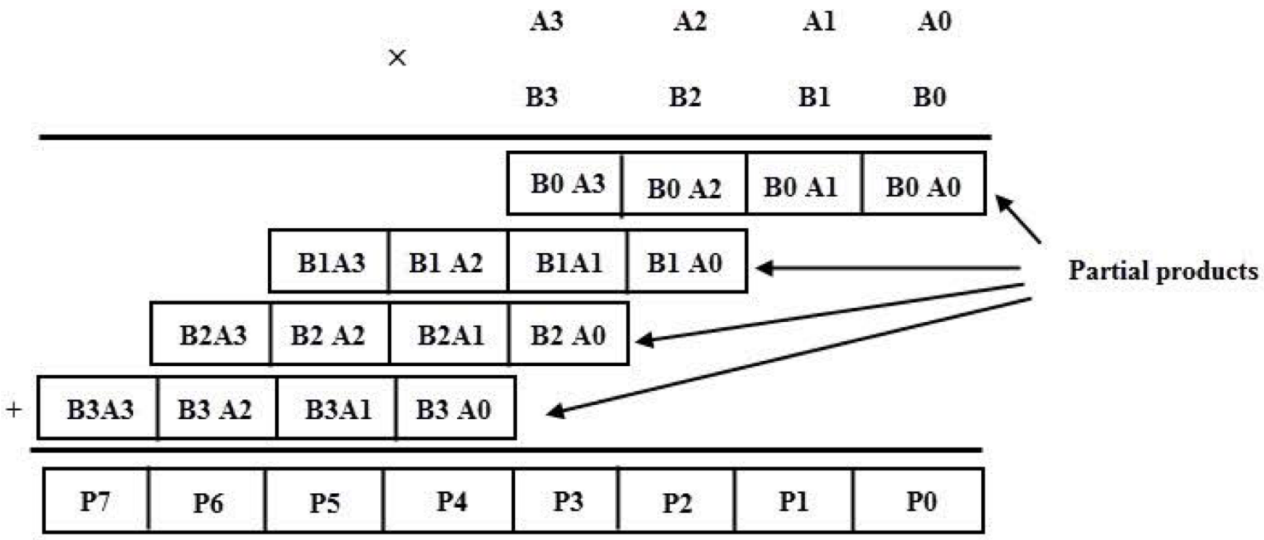

Figure 6: Diagram For Question 2

4.2 Solution Methodology

Our aim of this verilog program is to make a multiplier for multiplication of 4 bit numbers.For doing four bit multiplication the steps are:-
Step 1:- Input two 4 bit numbers A and B in a 4 bit register.
Step 2:- Now we take each bit of B and do AND operation with all the bits of number A.This AND operation is actually used to multiply the 2 bits. We store this result in a 4X8 Partial Product array.
Step 3:- Now we sum the first two rows of the partial product array by passing each bit to a one bit adder and then store that result in a Sum1 array.This one bit adder is the same which we had made in question 1 of this assignment.
Step 4:- Similarly we pass the last two rows of the partial product array by passing each bit to a one bit adder and then

store that result in a Sum2 array.This one bit adder is the same which we had made in question 1 of this assignment. Step 5:- Now we pass Sum1 and Sum2 array to eight bit adder which we had made in the question 1 of this assignment by doing this we obtain obtain our result of the multiplication of two 4 bit numbers which is the product of multiplication of two 4 bit numbers.Hence our result is obtained.

4.2.1 Terminal snippets For Four Bit Multipliers

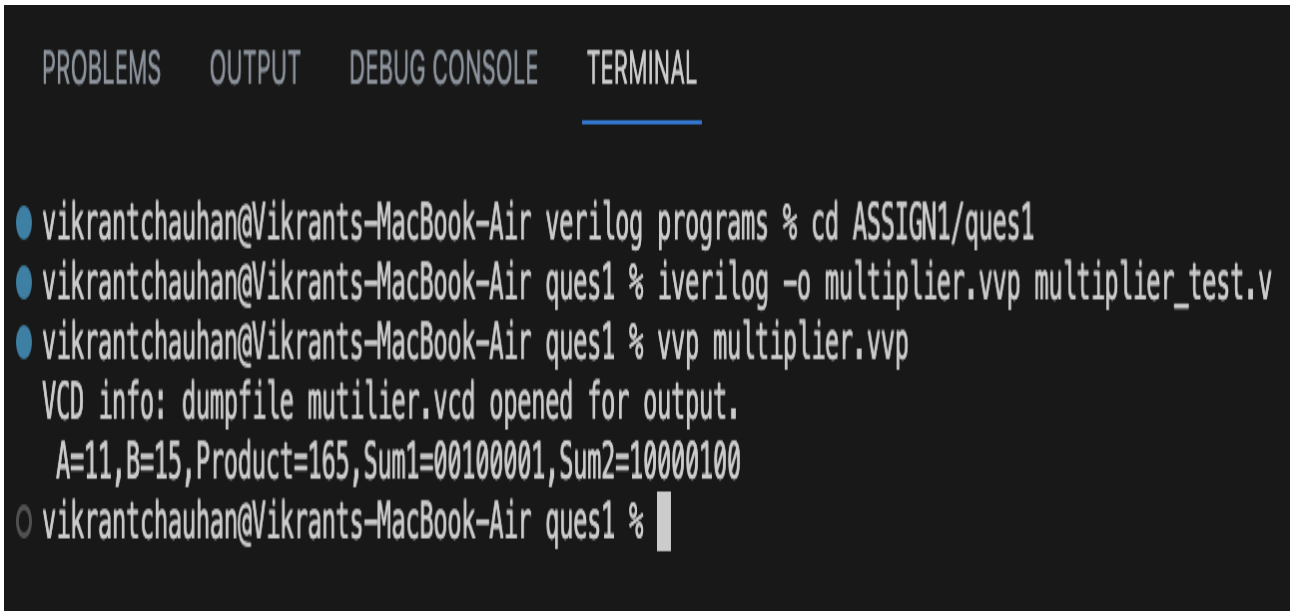

Figure 7: Output Of 4 bit multiplier

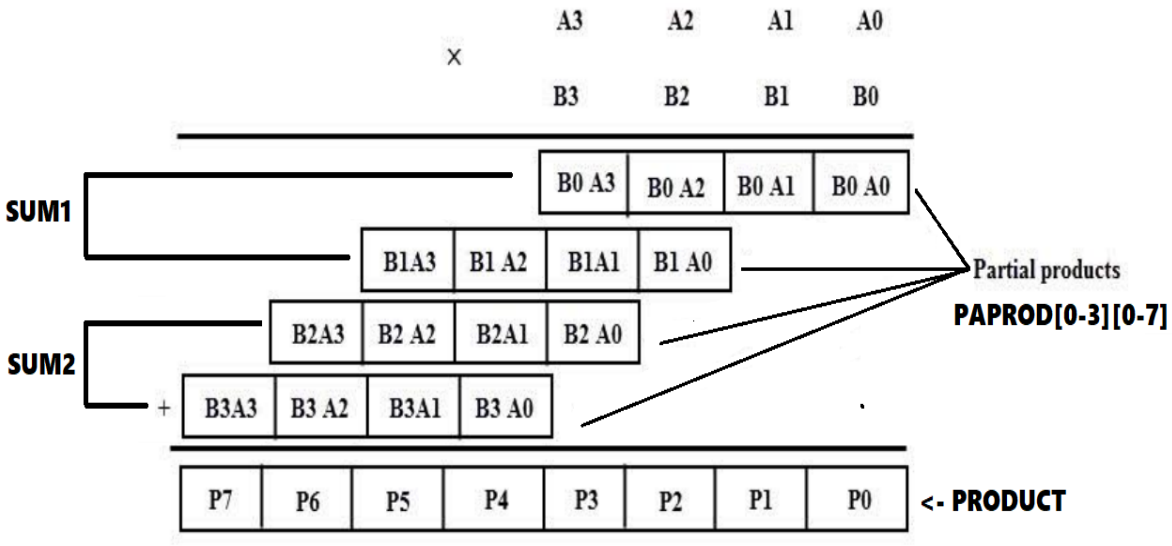

Figure 8: Diagram for 4 bit Multiplier

5 Question 3

5.1 Question

Write a Verilog code for a 4-bit forward counting synchronous Johnson counter with a one-bit reset signal. The Johnson counter is a type of shift register that cycles through a sequence of 2n states. In this case, the 4-bit counter will produce 16 states, each represented by a unique 4-bit binary pattern. Additionally, the counter should be able to reset to its initial state when a one-bit reset signal is asserted. The input consists of two signals 1. clock; 2. reset signal; and the output is a 4-bit value representing the state of the counter at a particular state.

5.2 Solution Steps

1. We have created two design files, one for the D-Flip Flop and imported it in the Johnson Counter Flip Flop file, and one testbench.
2. The D Flip Flop is created following the behavioral design method. In a positive edge triggering of clock, the 'd' value is passed to 'q'
3. We have used reset to initialize the initial value of 'q' to 0. Reset is initially set to high, and during this time, the 'q' gets assigned 0 value. When reset turns to 0, the flip flop becomes operational.
4. In every clock cycle (positive edge triggering), the compliment of q[3] comes to first DFF, and the q[0] is feeded as a input to the second DFF, and so on. Which generates the desired Johnson Flip Flop counting.
5. The clock is set to compliment itself after every 5-time unit. So, a cycle is achieved of 10-time units.
6. The testbench is scheduled to measure the counting after 1-time unit, as after 1-time unit, the reset is set to 0.
7. The testbench is scheduled to stop after 200-time units. We have created a vcd file inside the testbench, which dumps required data to plot the signals in GTK wave application.

5.3 Circuit Diagrams and Terminal snippets

Q0	Q1	Q2	Q3
0	0	0	0
1	0	0	0
1	1	0	0
1	1	1	0
1	1	1	1
0	1	1	1
0	0	1	1
0	0	0	1
repeat			

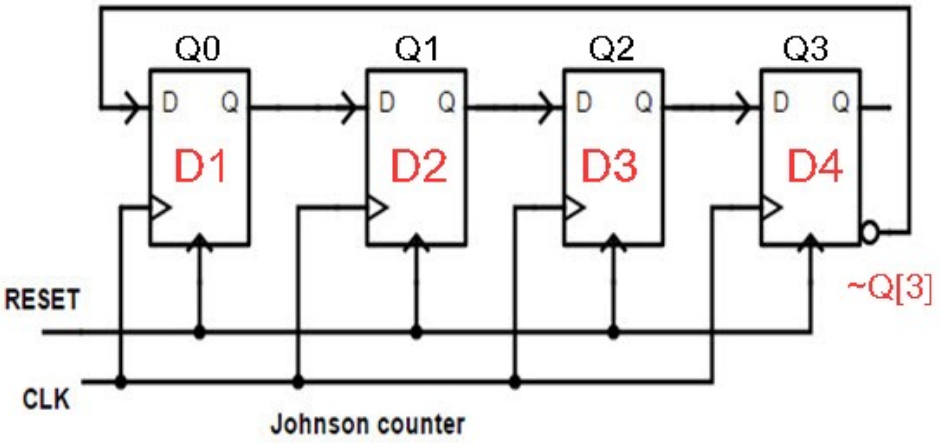

Figure 9: Circuit Diagram Of Johnson Counter

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\souvi\OneDrive\Desktop\q3> iverilog d_flipflop.v
PS C:\Users\souvi\OneDrive\Desktop\q3> iverilog john_count.v
PS C:\Users\souvi\OneDrive\Desktop\q3> iverilog john_test.v
PS C:\Users\souvi\OneDrive\Desktop\q3> iverilog -o new.vvp  john_test.v
PS C:\Users\souvi\OneDrive\Desktop\q3> vvp new.vvp
VCD info: dumpfile john_test.vcd opened for output.
                   1 0000
                  10 1000
                  20 1100
                  30 1110
                  40 1111
                  50 0111
                  60 0011
```

Figure 10: Terminal Snippet

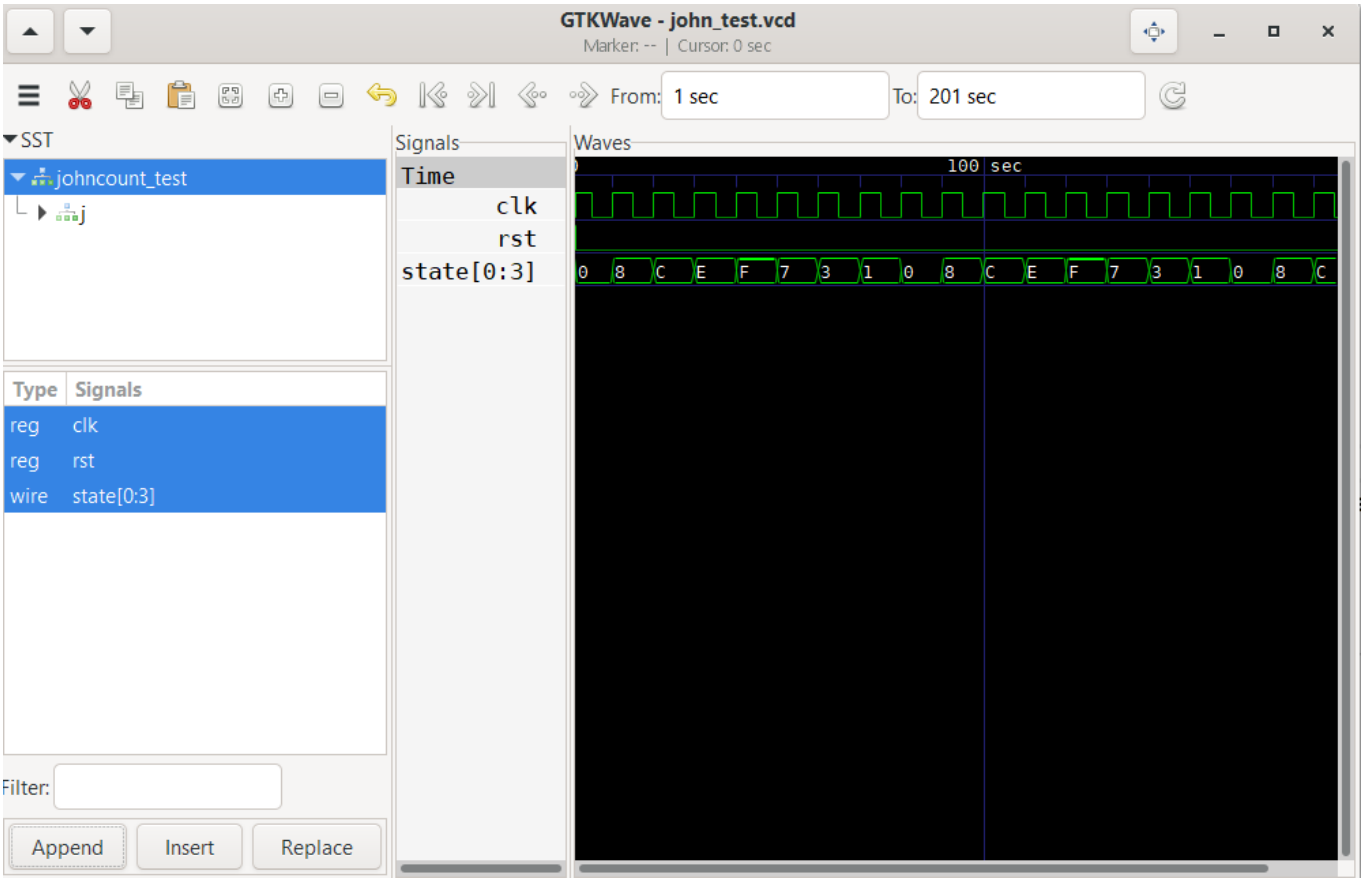

Figure 11: GTK Wave snippet

6 Group Members And Percentage of Contribution

1. VIKRANT CHAUHAN : MS CYBERSERCURITY : 231110407
2. SOUVIK MUKHERJEE : MS CYBERSECURITY : 231110405
3. VALETI LOKESH : MS CYBERSECURITY : 231110406
4. VISHAL KUMAR : MTECH CYBERSECURITY : 231110058
5. M DHILIPKUMAR : MTECH CYBERSECURITY : 231110026

All group members have equally worked in this project and contributed equally in this project.