B.M.S. College of Engineering

(Autonomous Institution affiliated to VTU, Belagavi)

Department of Computer Science and Engineering

AAT
Verilog Laboratory
Report

19CS3PCLOD

(Autonomous Scheme 2020)

Submitted By:

Name: **Ananya Setty B.A**
USN: **1BM21CS**

B.M.S. College of Engineering
Department of Computer Science and Engineering

Laboratory Certificate

This is to certify that **ANANYA SETTY B.A** has satisfactorily completed the course of Experiments in Practical Logic Design (Verilog) prescribed by the Department during the odd semester 2021-22.

Name of the Candidate: **ANANYA SETTY B.A**

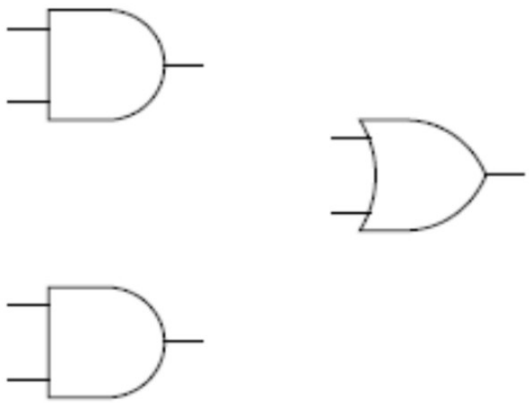
USN No.: **1BM21CS** Semester: **III** Section: **A**

Marks	
Max. Marks	Obtained
10	
Marks in Words	

Signature of the staff in-charge **Head of the Department**

Date:

Verilog Program List
19CS3PCLOD
Laboratory Experiments

Serial No.	Title
	CYCLE I **Structural Modeling**
•	Write HDL implementation for the following Logic • AND/OR/NOT Simulate the same using structural model and depict the timing diagram for valid inputs.
•	Write HDL implementation for the following Logic • NAND/NOR Simulate the same using structural model and depict the timing diagram for valid inputs.
•	Write HDL implementation for the following Logic  Simulate the same using structural model and depict the timing diagram for valid inputs.
•	Write HDL implementation for a 4:1 Multiplexer. Simulate the same using structural model and depict the timing diagram for valid inputs.
•	Write HDL implementation for a 2-to-4 decoder. Simulate the same using structural model and depict the timing diagram for valid inputs.
•	Write HDL implementation for a 4-to-2 encoder. Simulate the same using structural model and depict the timing diagram for valid inputs.

	CYCLE II **Behavior Modeling**
•	Write HDL implementation for a RS flip-flop using behavioral model. Simulate the same and depict the timing diagram for valid inputs.
•	Write HDL implementation for a JK flip-flop using behavioral model. Simulate the same and depict the timing diagram for valid inputs.
•	Write HDL implementation for a 4-bit right shift register using behavioral model. Simulate the same and depict the timing diagram for valid inputs.
•	Write HDL implementation for a 3-bit up-counter using behavioral model. Simulate the same and depict the timing diagram for valid inputs.
	CYCLE III **Dataflow Modeling**
•	Write HDL implementation for AND/OR/NOT gates using data flow model. Simulate the same and depict the timing diagram for valid inputs.
•	Write HDL implementation for a 3-bit full adder using data flow model. Simulate the same and depict the timing diagram for valid inputs.

Verilog Program List-19CS3PCLOD

SCHEME OF CONDUCT AND EVALUATION

CLASS: III SEMESTER
YEAR: 21-22

EVALUATION SCHEME Tutorial Test: 1 hour

Expt. No.	TITLE	Max. Marks	Marks Obtained	Signature
•	AND/OR/NOT	5		
•	NAND/NOR			
•	Logic diagram			
•	Multiplexer			
•	Decoder			
•	Encoder			
•	RS			
•	JK			
•	Shift right			
•	Counter			
•	AND/OR/NOT – data flow			
•	3-bit full adder			
	Test: Viva – 2 Marks + Write-up – 1 Mark + Execution – 2 Marks	5		
	TOTAL MARKS	10		

Icarus Verilog Installation (Windows)

- Download and install iverilog from here: http://bleyer.org/icarus/
- During installation, check the checkbox so that iverilog is added to the System PATH.
- If this is not done, one will need to manually locate their iverilog installation directory and copy the path to the bin folder situated within it.
- To do this, we will need to navigate to the control panel and access the environment variables section of the computer. Here, we add a new variable and set it to `%PATH%;c:\iverilog\bin` assuming that is the installation directory for iverilog.
- Open command prompt or any other preferred terminal and type `iverilog` and press enter. Trace back the steps for mistakes if version information is not displayed and re-install if necessary.

Compilation and viewing waveforms

- With iverilog, a third party text editor is necessary. Any editor would work (notepad, gedit, or vim, for example) but a modern text editor such as SublimeText 3 or Visual Studio Code is preferred as plug-ins can be installed to provide syntax highlighting for the Verilog HDL code being written.
- While writing the code, two lines must be added in the test bench module.

```
module testSR;
    reg [1:0]A;
    reg c;
    wire q,qb;
    SR_FF srff(A,c,q,qb);
    initial c = 1'b0;
    always #5 c = ~c;
    initial
        begin
            $dumpfile("test.vcd");
            $dumpvars(0,testSR);
```

  The $dumpfile() and $dumpvars() functions should be passed, with the former containing the name of a file ending with .vcd and the latter containing the name of the testbench module itself. This will dump a vcd file that will allow us to view the waveform. This should always be right after the initial and begin statements.
- After this file is written and saved under a .v extension (example `VHDLcode.v`), the terminal must be opened and the user must navigate to the directory in which the file is saved.
- In this directory, running `iverilog -o outputFile VHDLcode.v` will output a .vvp file with the name `outputFile.vvp`.
- After generating the vvp file, run the `vvp outputFile` command to get the waveform dump with name of the given string under $dumpfile().

- The next step is to run `gtkwave` in the command line and open File > Open New Tab > select the generated .vcd file. Afterward, click on the added element on the viewer and insert it. This will display the waveform.

References

- Zucker, M. (2019). *E15 - Installing and testing Icarus Verilog*. [online] Swarthmore.edu. Available at: https://www.swarthmore.edu/NatSci/mzucker1/e15_f2014/iverilog.html
- KONSTADELIAS, I. (n.d.). *Icarus Verilog + GTKWave Guide*. [ebook] http://inf-server.inf.uth.gr/~konstadel/resources/Icarus_Verilog_GTKWave_guide.pdf

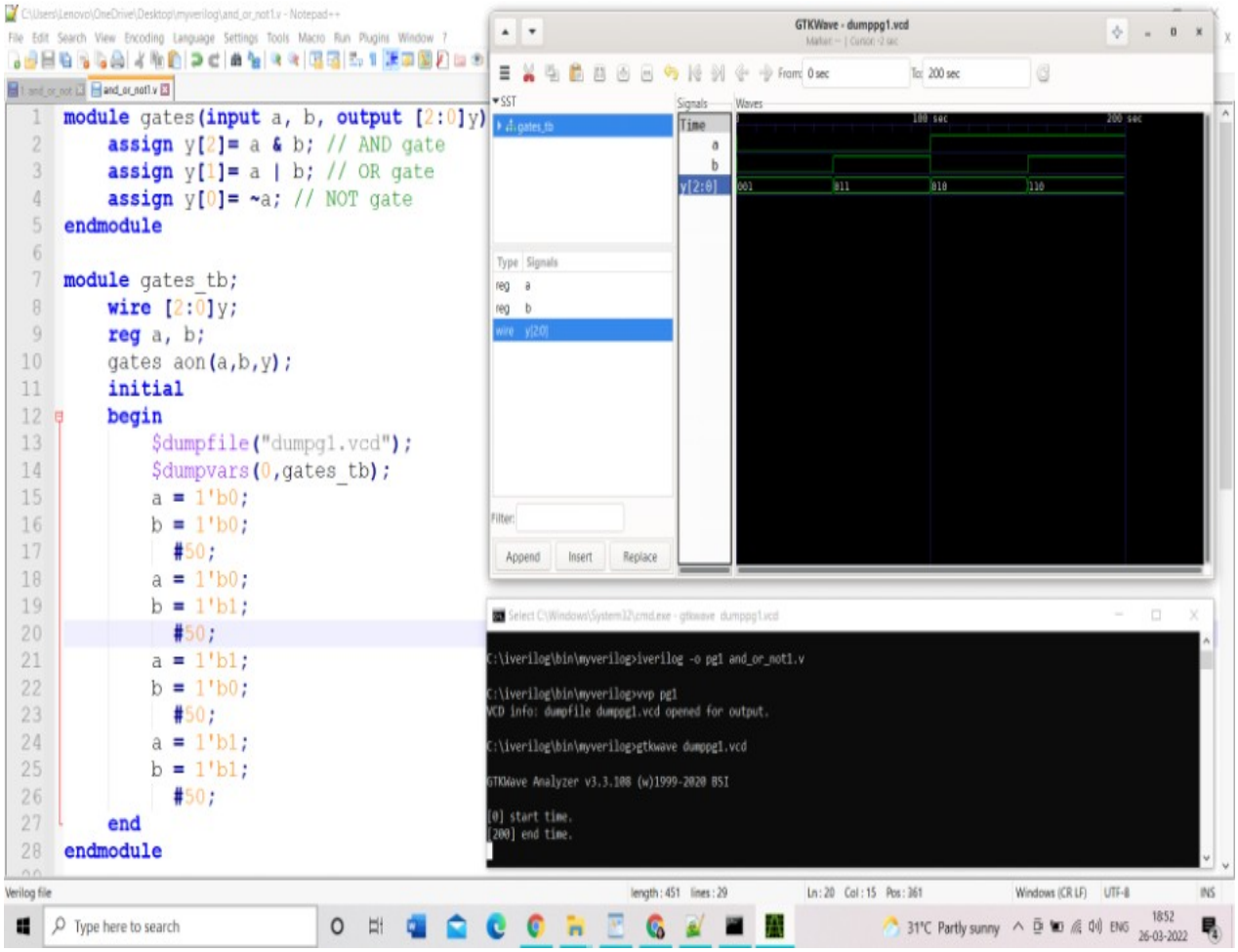
Cycle 1: Structural Modelling

Experiment 1: <u>Source Code</u>

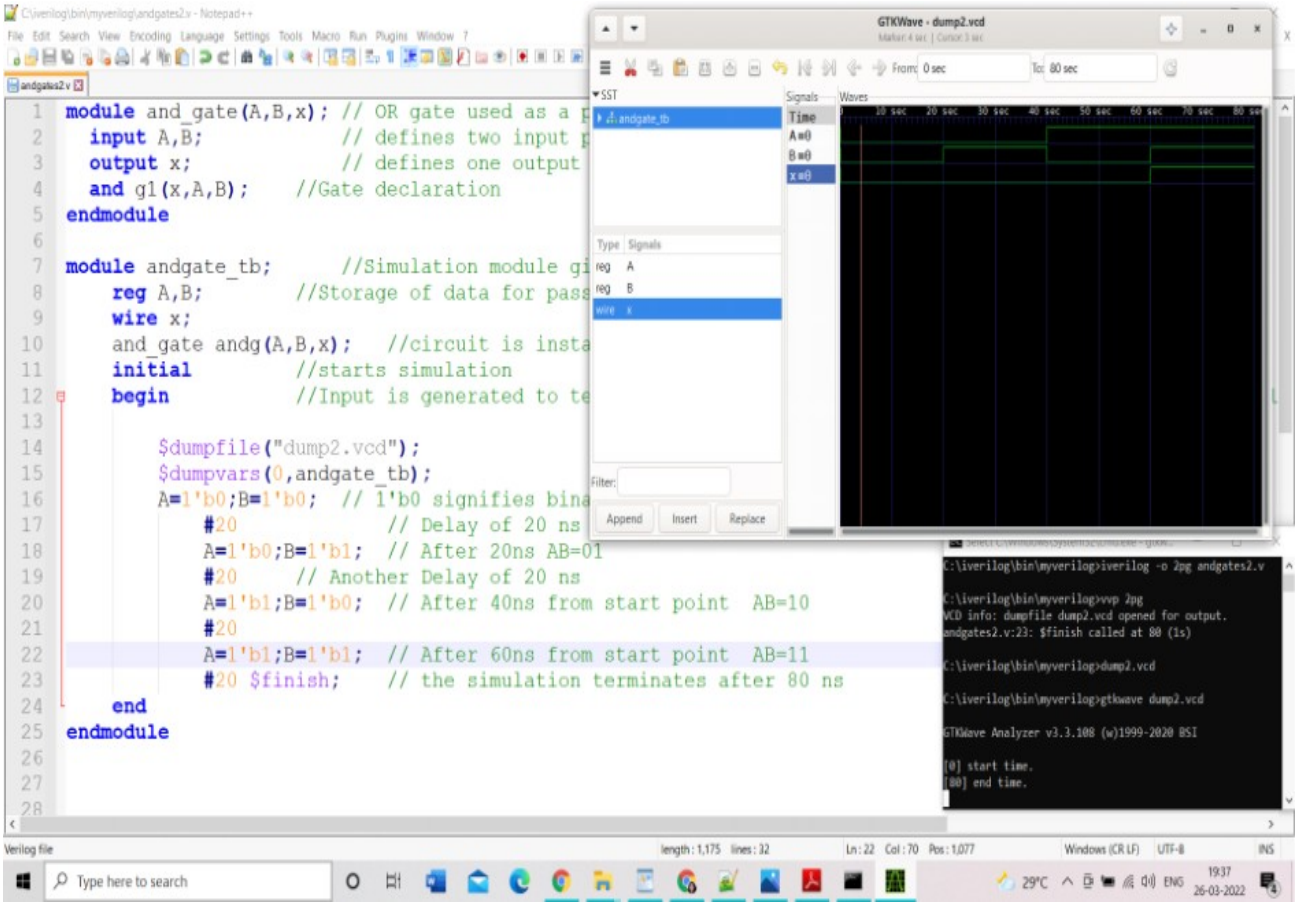
```
module gates(input a, b, output [2:0]y);
        assign y[2]= a & b; // AND gate
        assign y[1]= a | b; // OR gate
        assign y[0]= ~a; // NOT gate
endmodule

module gates_tb;
        wire [2:0]y;
        reg a, b;
        gates aon(a,b,y);
        initial
        begin
                $dumpfile("dumpg1.vcd");
                $dumpvars(0,gates_tb);
                a = 1'b0;
                b = 1'b0;
                  #50;
                a = 1'b0;
                b = 1'b1;
                  #50;
                a = 1'b1;
                b = 1'b0;
                  #50;
                a = 1'b1;
                b = 1'b1;
                  #50;
        end
endmodule
```

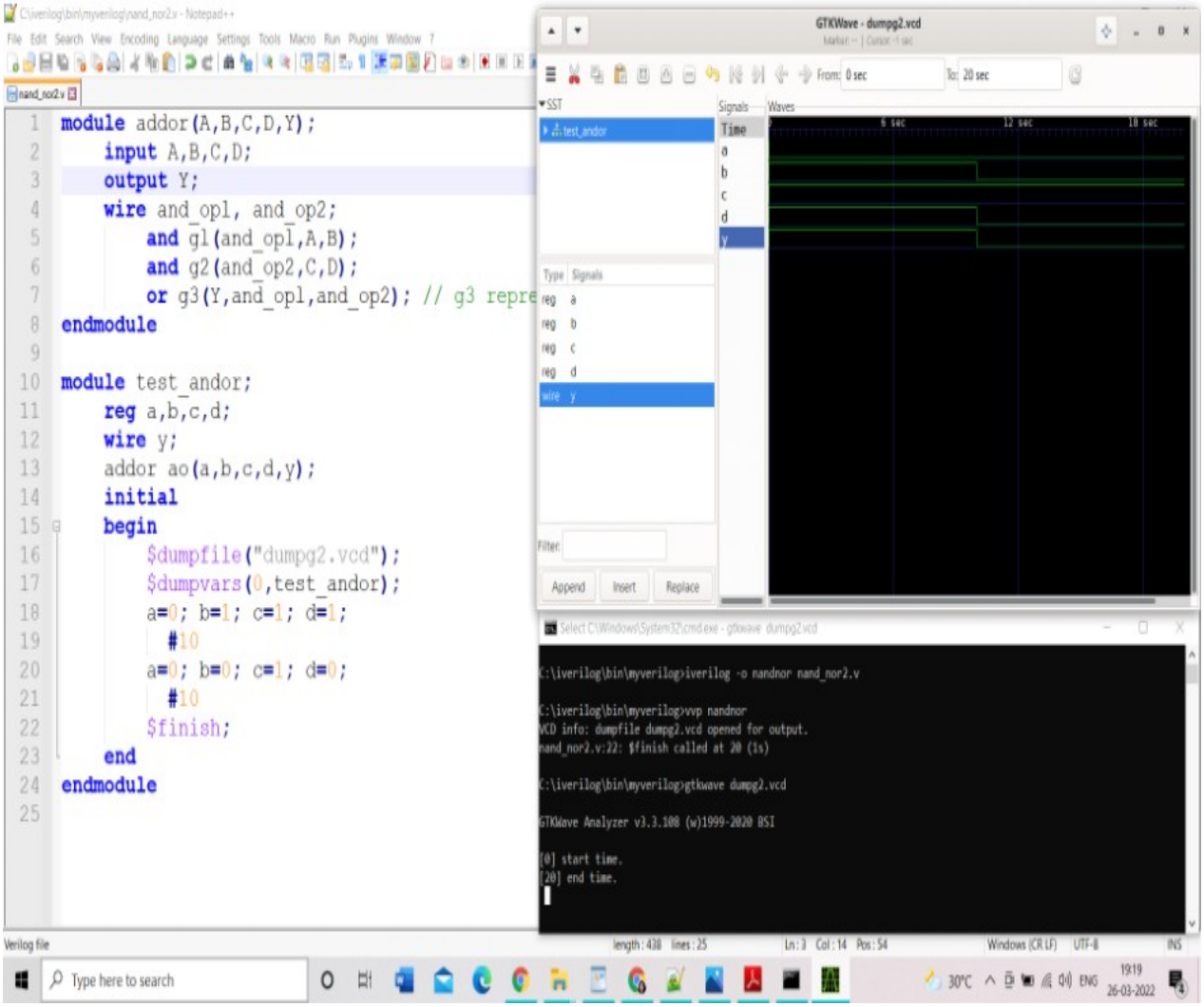
Compilation, Execution and Result of Simulation:



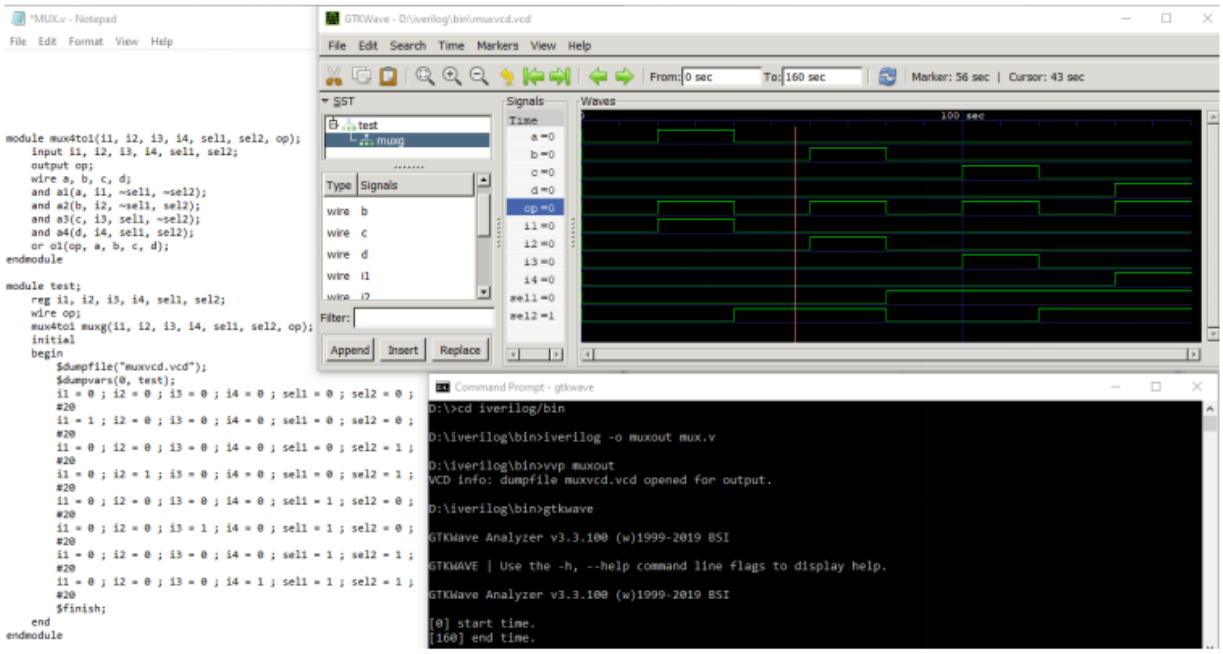
Experiment 2 :<u>Compilation, Execution and Result of Simulation</u>



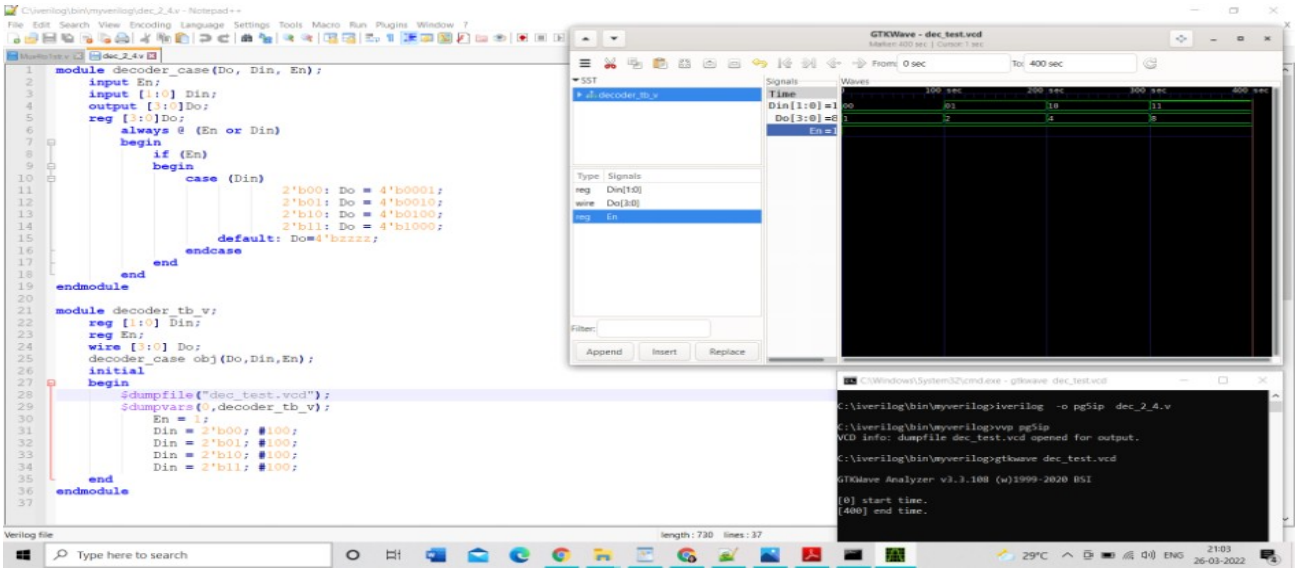
Experiment 3 :<u>Compilation, Execution and Result of Simulation</u>



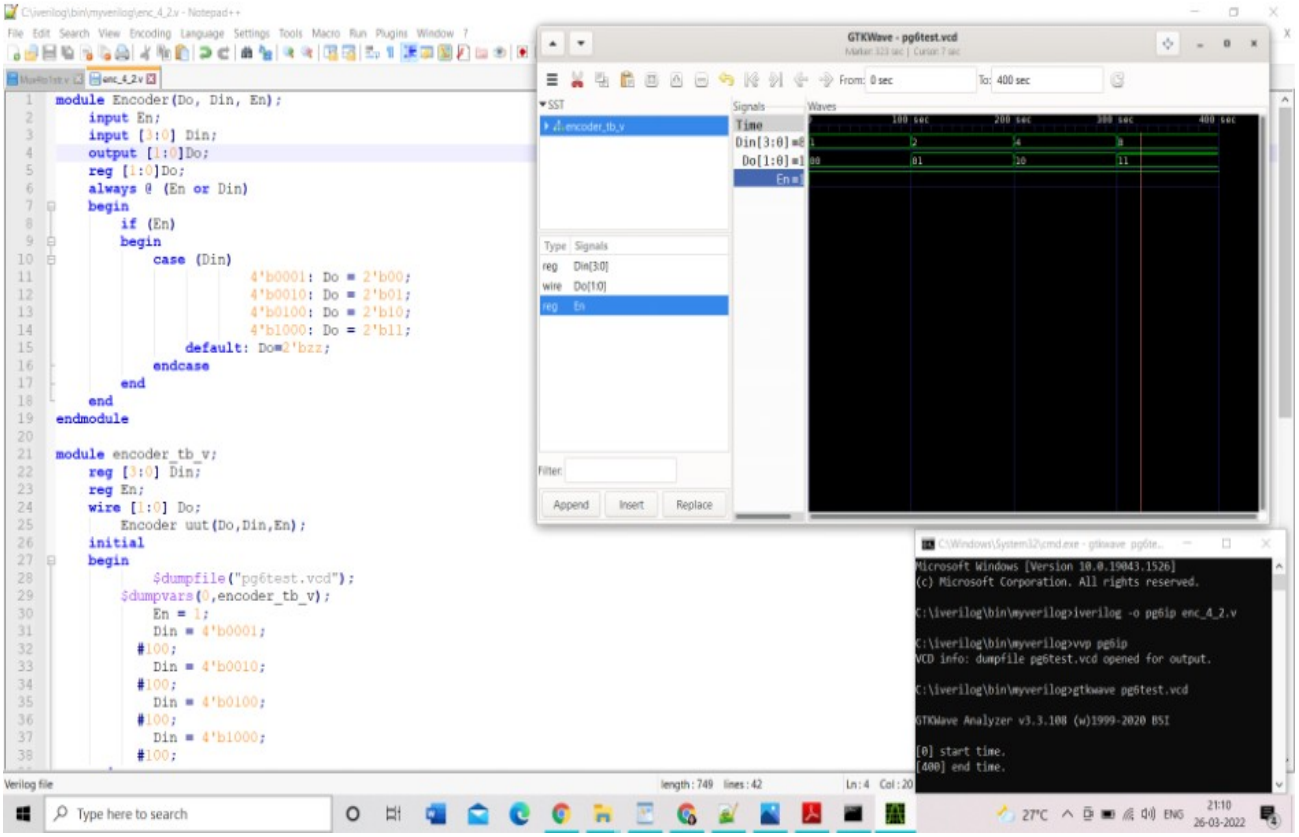
Experiment 4 :Compilation, Execution and Result of Simulation



Experiment 5 :Compilation, Execution and Result of Simulation

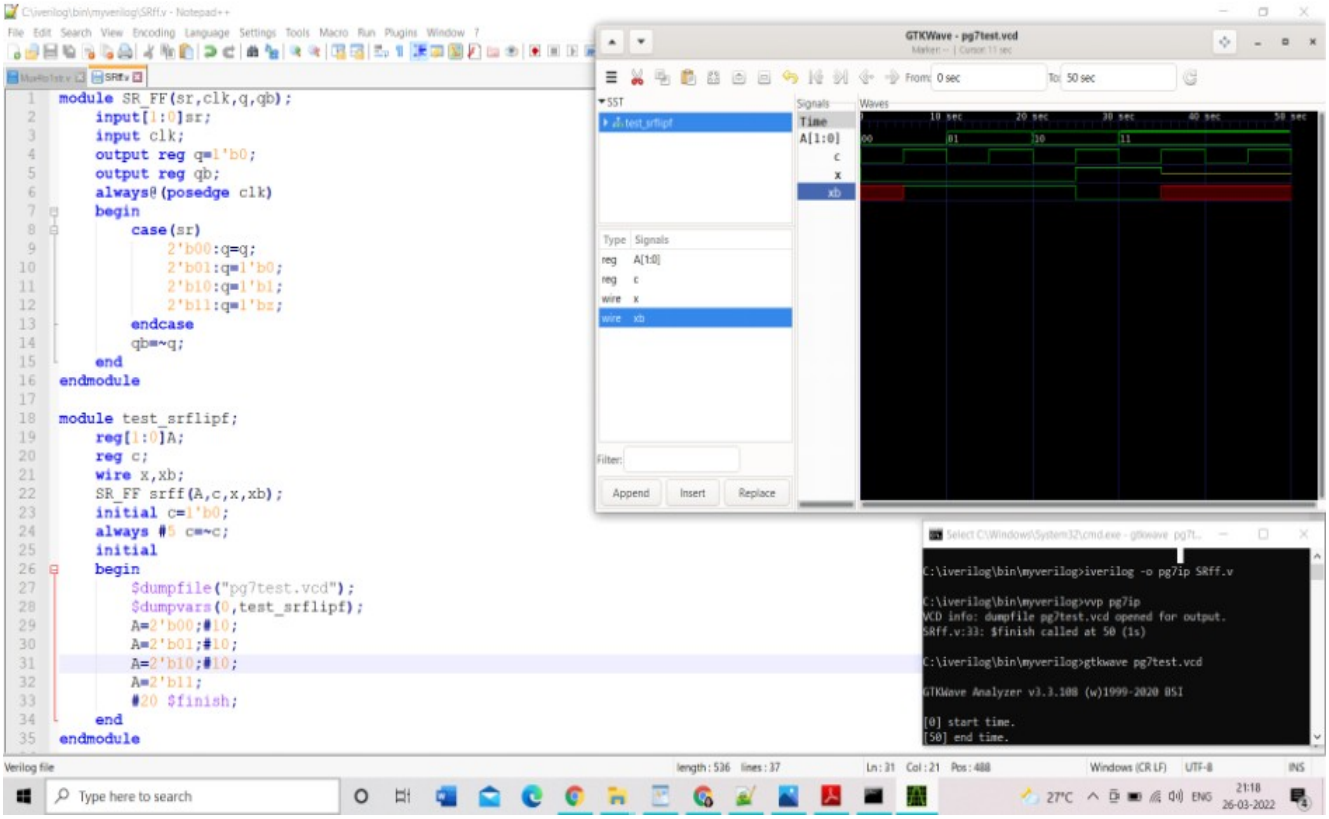


Experiment 6 :Compilation, Execution and Result of Simulation

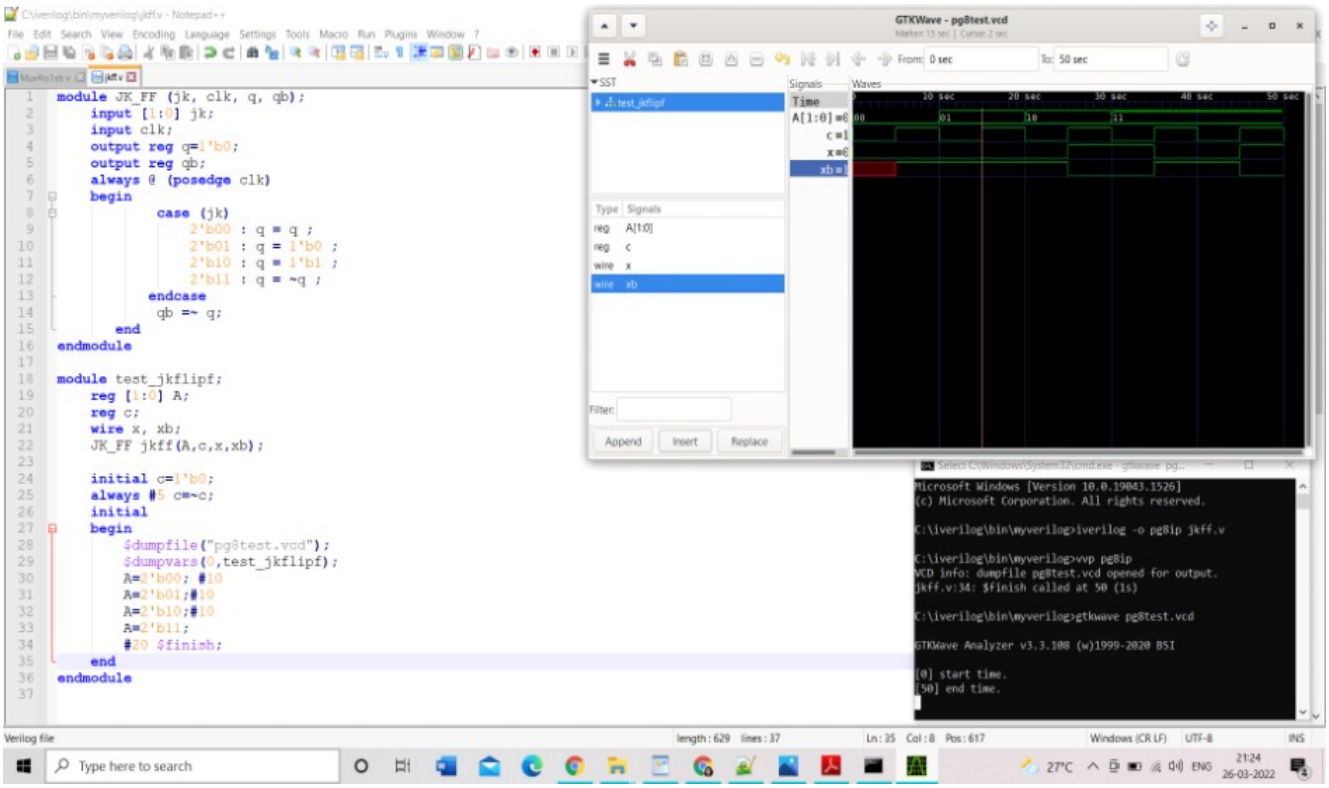


Cycle 2: Behavior Modelling

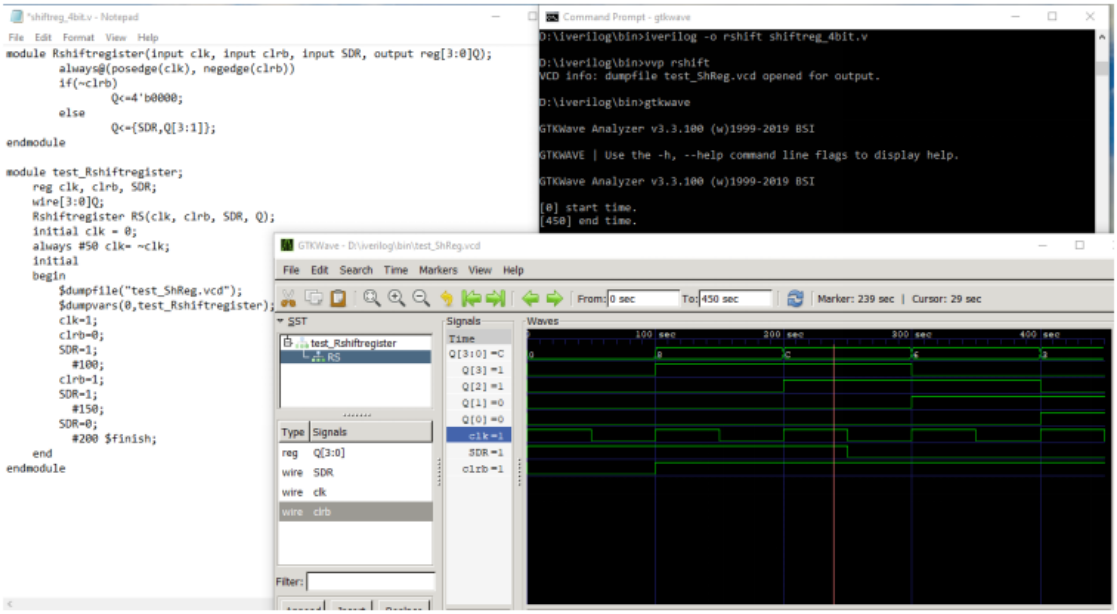
Experiment 7 :Compilation, Execution and Result of Simulation



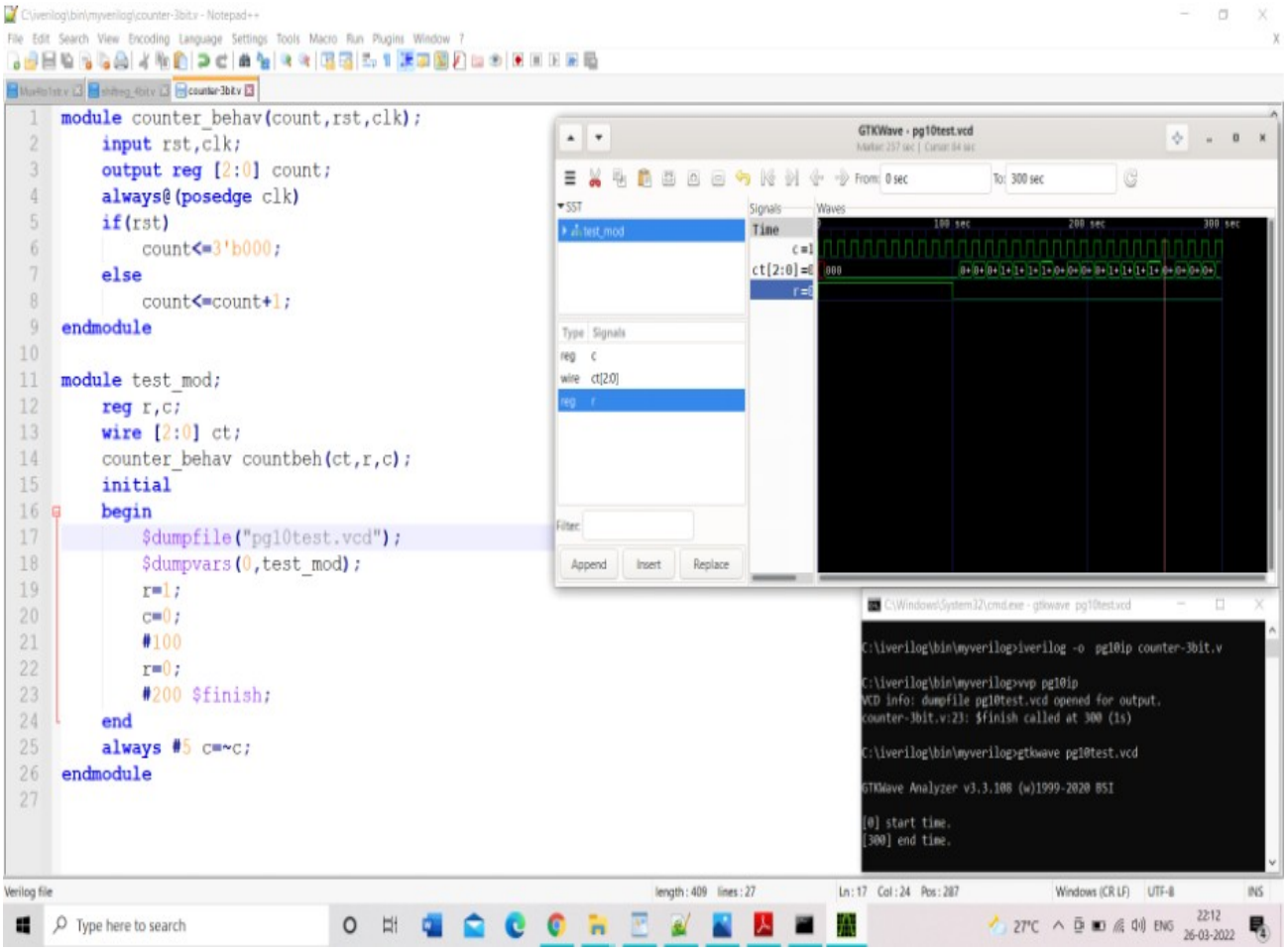
Experiment 8:Compilation, Execution and Result of Simulation



Experiment 9:<u>Compilation, Execution and Result of Simulation</u>

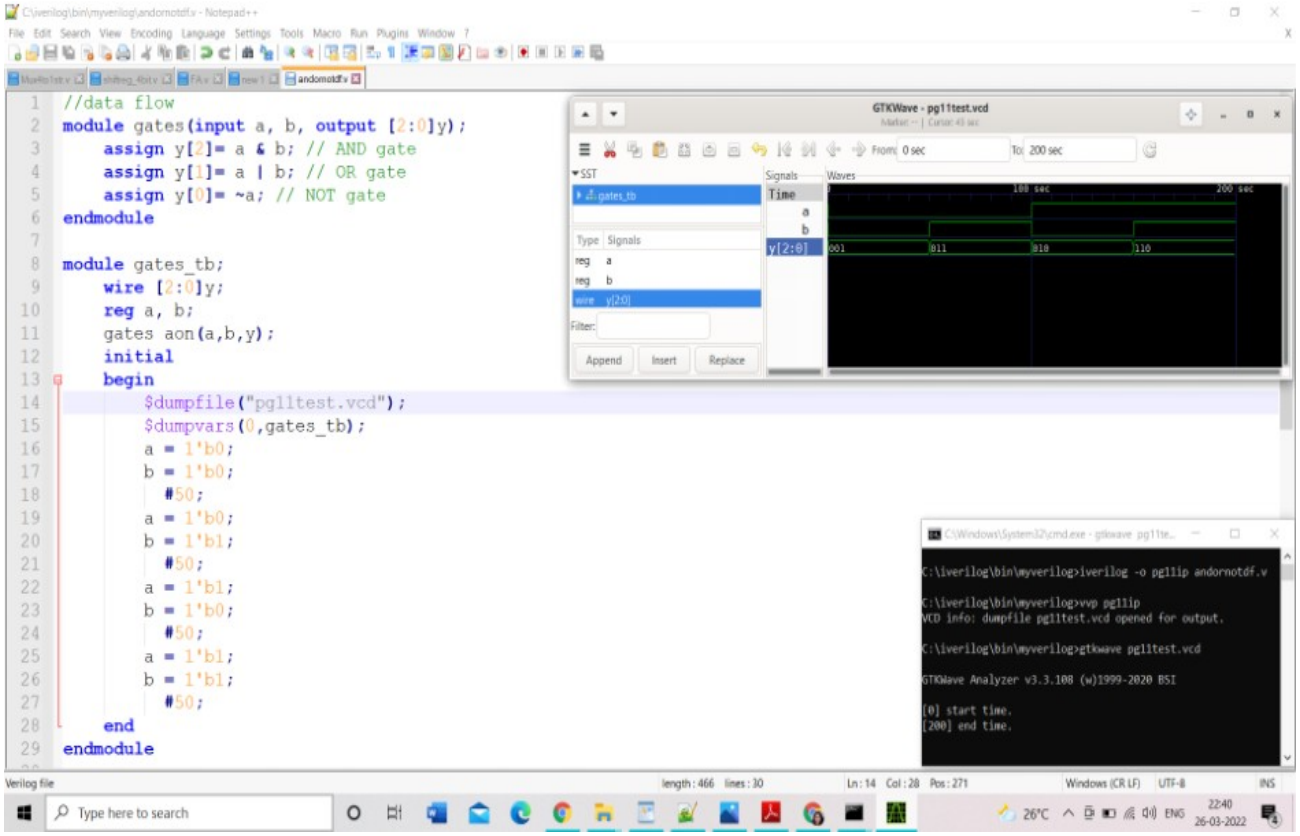


Experiment 10:<u>Compilation, Execution and Result of Simulation</u>

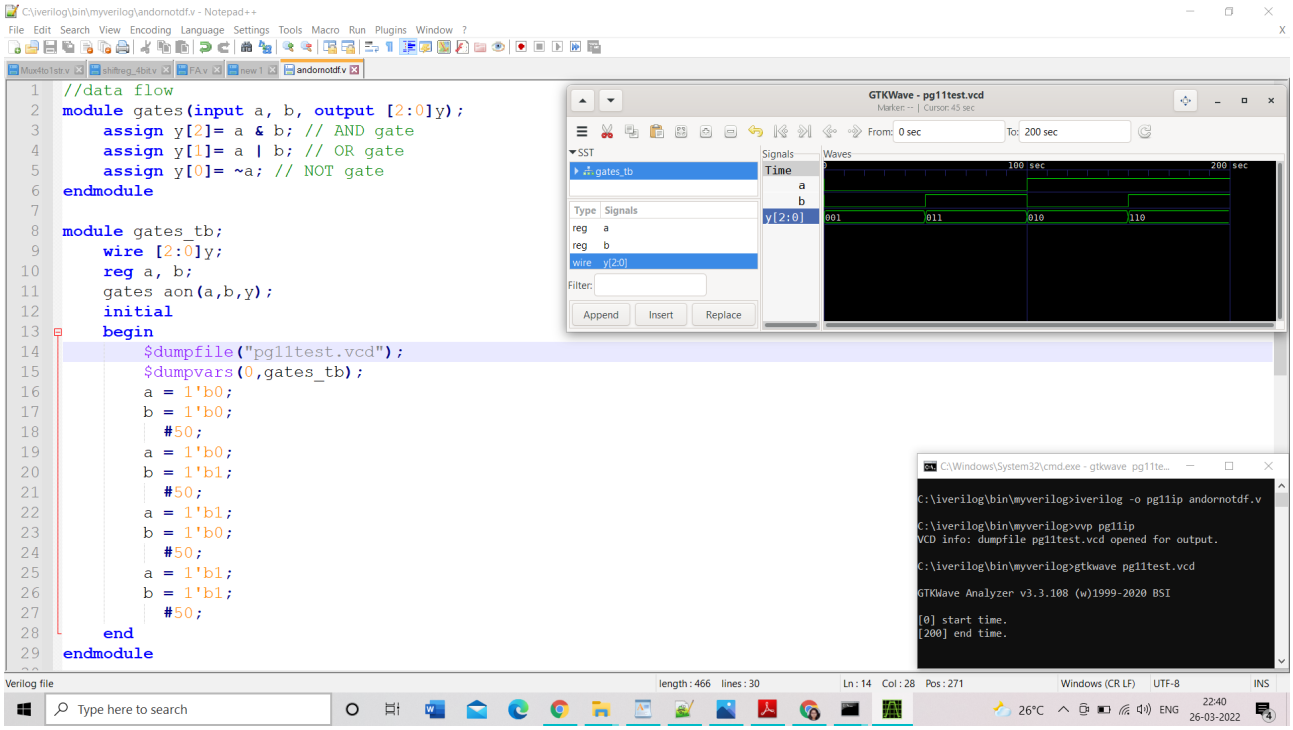


Cycle 3: Dataflow Modelling

Experiment 11:Compilation, Execution and Result of Simulation



Experiment 12:Compilation, Execution and Result of Simulation



Experiment 12:Compilation, Execution and Result of Simulation