

Feedforward Neural Network Design

(HSCD Assignment 1)

Name : Ashutosh Rajendra Karve
Roll no : 01021
Email : 2024ht01021@wilp.bits-pilani.ac.in
Contact : +91 9765541324
Date : 10/10/2024
Place : Pune, Maharashtra

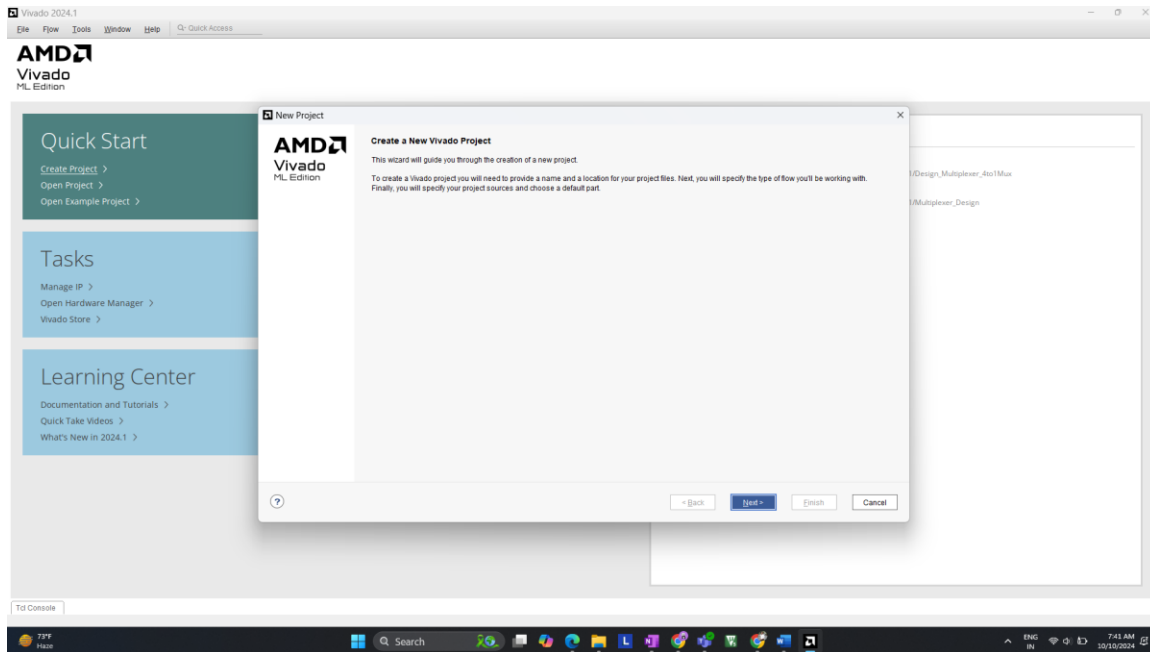
1. Introduction:

In this assignment, a **Feedforward Neural Network** (FNN) with one hidden layer was implemented using Verilog HDL. The network architecture consists of four inputs, a hidden layer with three neurons, and an output layer with two neurons. Each neuron uses a **ReLU (Rectified Linear Unit)** activation function, which ensures that the output of the neurons is non-negative.

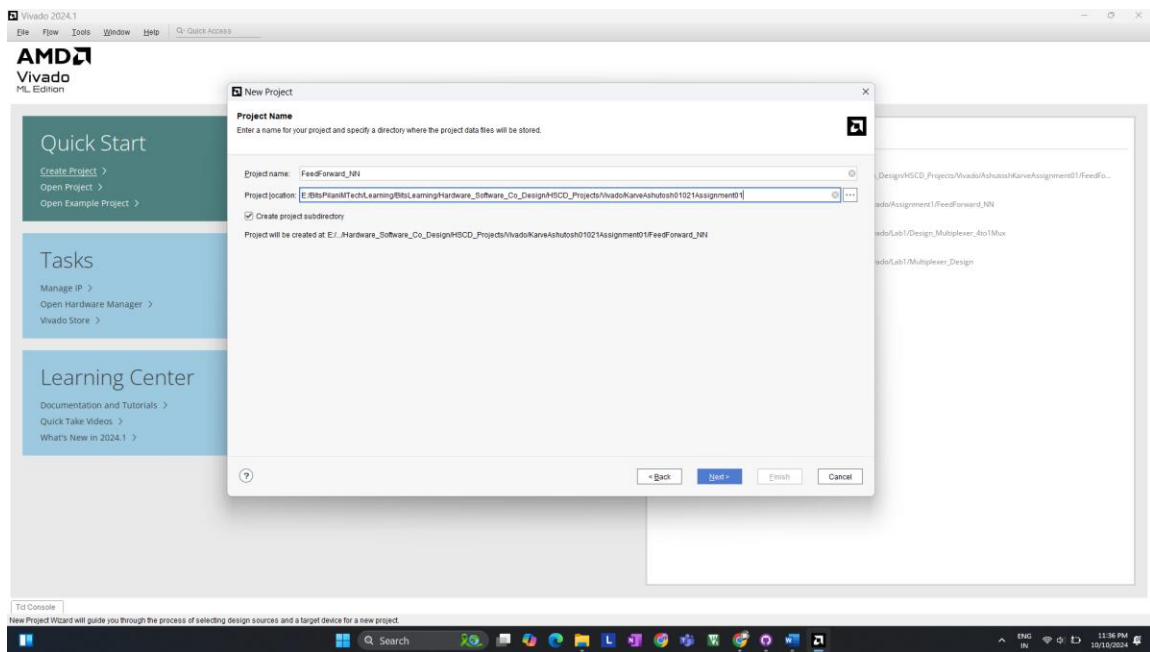
2. Vivado Project Setup

2.1 Project Creation

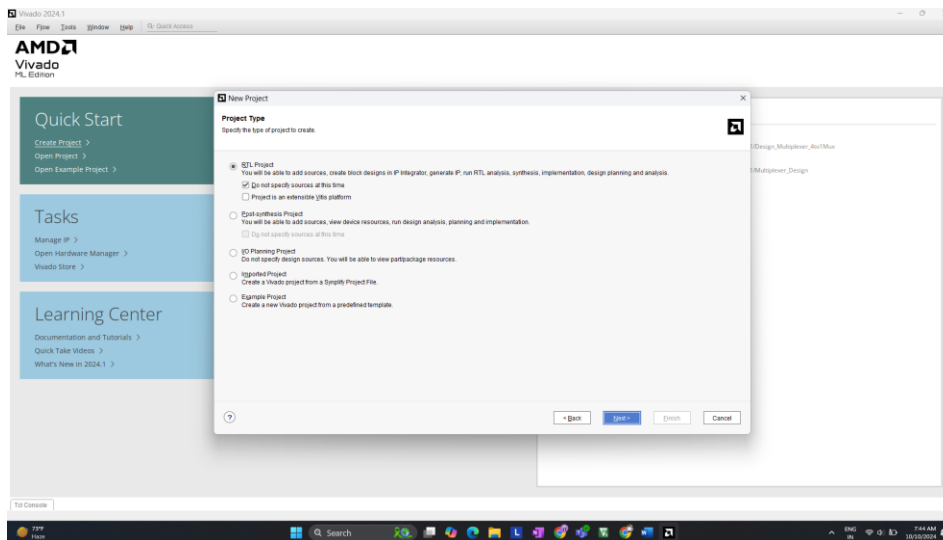
The project was created in Vivado, targeting the ZedBoard (Zynq-7000). The RTL project was set up for Verilog coding.



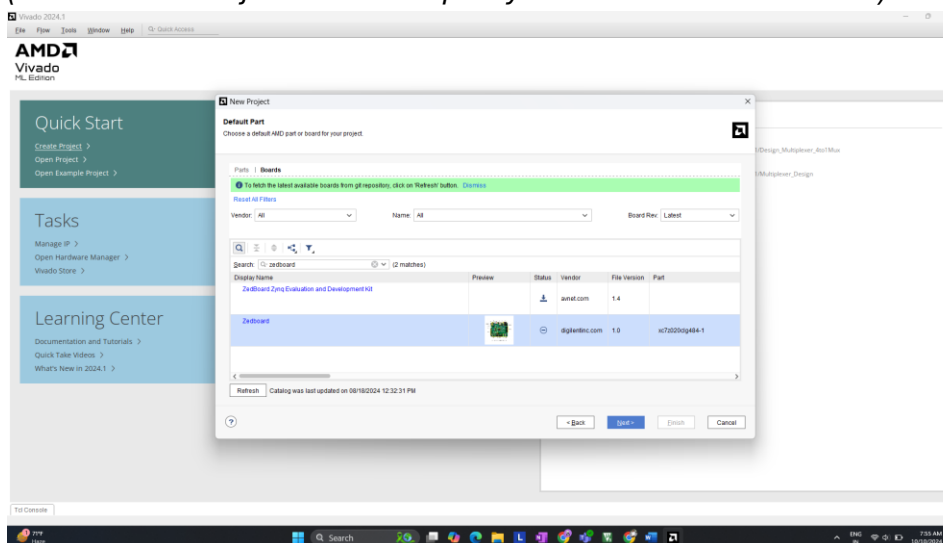
(Create Project >> Next)



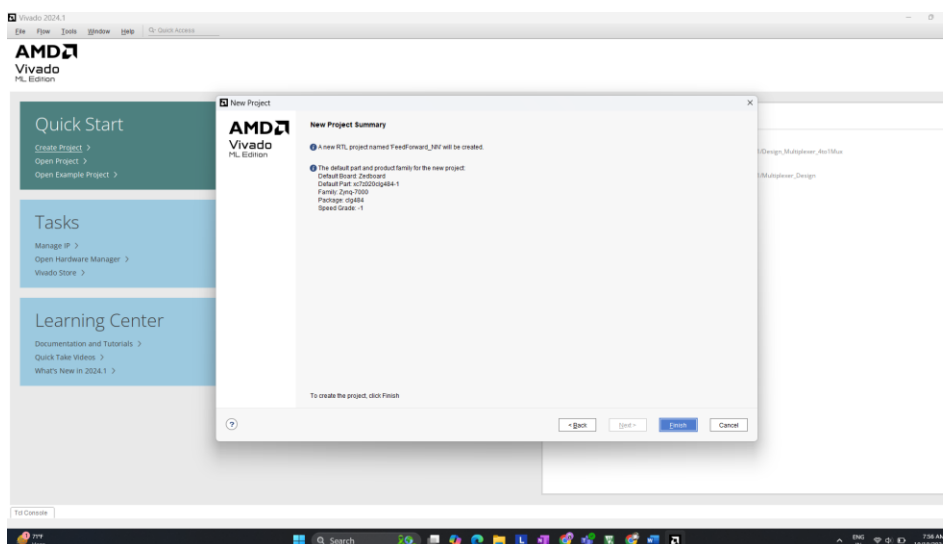
(Project Name (FeedForward_NN) >> Next)



(Select >> RTL Project >> Do not specify sources at this time >> Next)

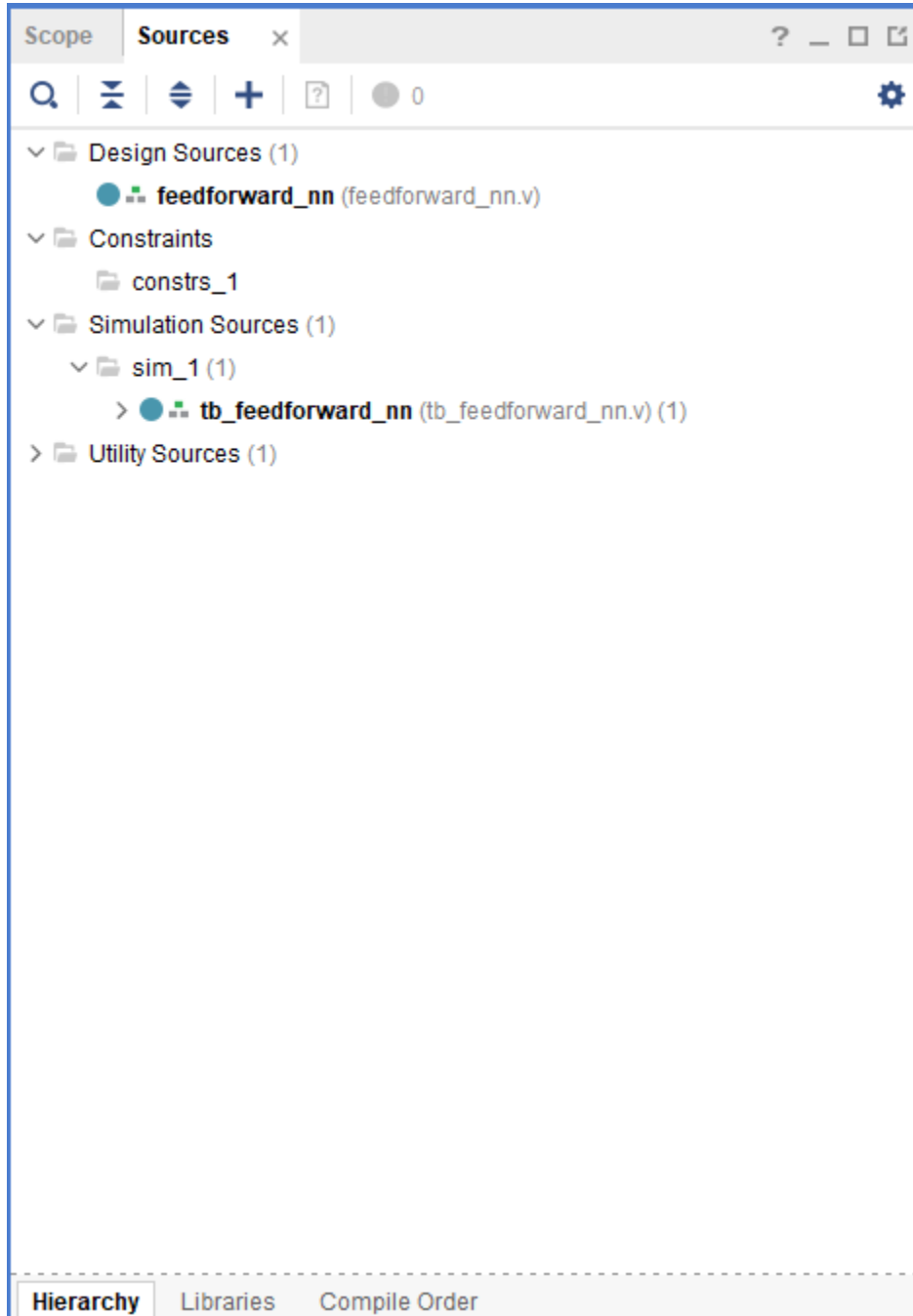


(Select >> Zedboard >> Next)

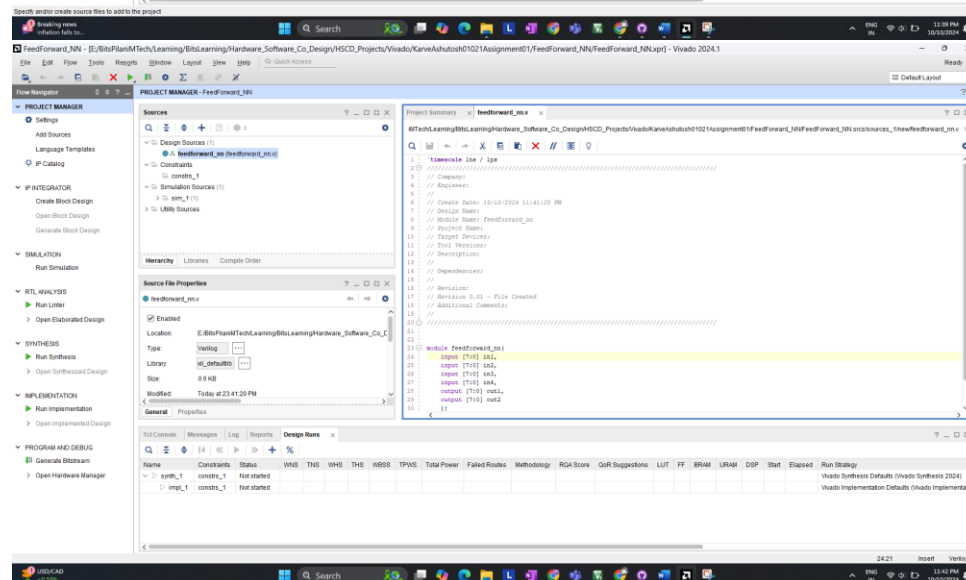
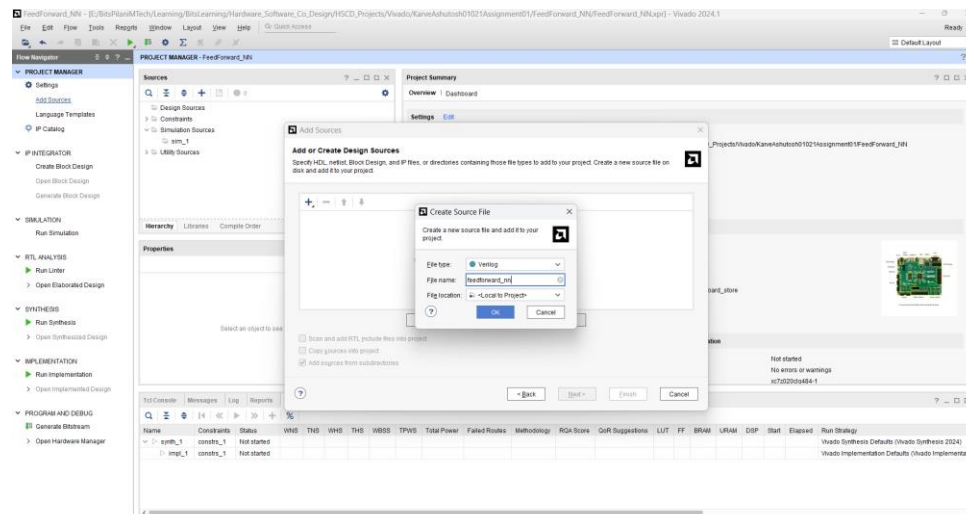
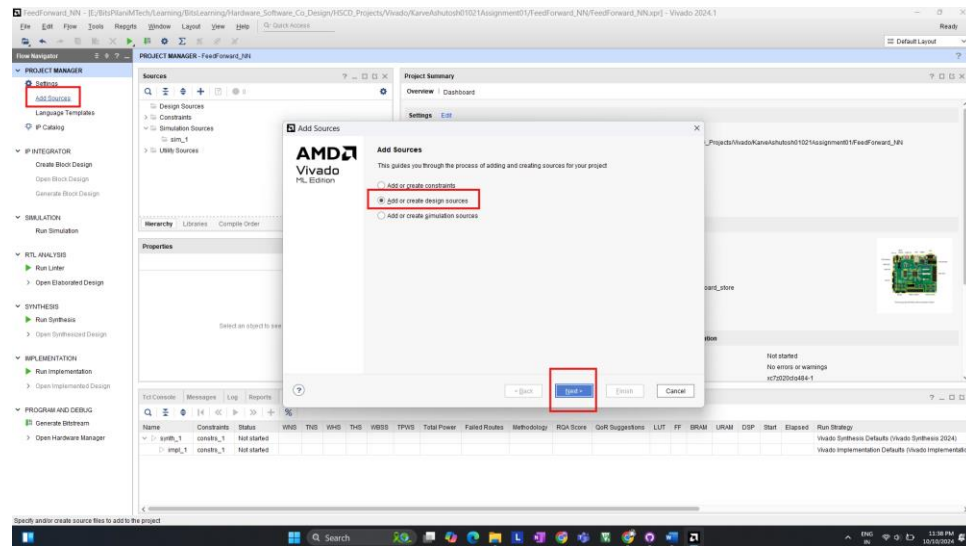


2.2 Design Sources Overview

The design consists of the **feedforward_nn.v** file for the network and the **tb_feedforward_nn.v** file for the testbench. Both files were added as design and simulation sources, respectively.



Example: how to add files >> (feedforward_nn.v) (design source)



3. Network Architecture

The network was implemented with the following layers:

- **Input Layer:** 4 inputs, each 8-bit signed.
- **Hidden Layer:** 3 neurons with ReLU activation.
- **Output Layer:** 2 outputs, each 8-bit signed.

3.1 ReLU Activation Function

The ReLU function ensures that the output of a neuron is non-negative, and is defined as follows:

```
feedforward_nn.v  x  tb_feedforward_nn.v  x  Untitled 2  x
IMTechLearningBtlLearningHardware_Software_Co_DesignHSCD_Projects/VivadoKaveAshutosh01021Assignment01FeedForward_NN/FeedForward_NN1.srcs/sources_1/newfeedforwa

42 // Initialize weights and biases
43 initial begin
44 // Initialize hidden layer weights and biases
45 w_hidden[0][0] = 0'sd1; w_hidden[0][1] = 0'sd2; w_hidden[0][2] = -0'sd1; w_hidden[0][3] = 0'sd1;
46 w_hidden[1][0] = 0'sd0; w_hidden[1][1] = 0'sd1; w_hidden[1][2] = -0'sd2; w_hidden[1][3] = 0'sd2;
47 w_hidden[2][0] = 0'sd3; w_hidden[2][1] = -0'sd1; w_hidden[2][2] = 0'sd2; w_hidden[2][3] = -0'sd1;
48
49 b_hidden[0] = 0'sd1; b_hidden[1] = -0'sd1; b_hidden[2] = 0'sd0;
50
51 // Initialize output layer weights and biases
52 w_output[0][0] = 0'sd2; w_output[0][1] = -0'sd3; w_output[0][2] = 0'sd1;
53 w_output[1][0] = 0'sd1; w_output[1][1] = 0'sd2; w_output[1][2] = -0'sd1;
54
55 b_output[0] = 0'sd1; b_output[1] = -0'sd2;
56 end
57
58 // ReLU activation function
59 function signed [7:0] ReLU(input signed [7:0] x);
60 begin
61 if (x > 0'sd0)
62 ReLU = x;
63 else
64 ReLU = 0'sd0;
65 end
66 endfunction
67
68 // Compute hidden layer outputs
69 always @(*) begin
70 hidden_output[0] = ReLU(in1 * w_hidden[0][0] + in2 * w_hidden[0][1] + in3 * w_hidden[0][2] + in4 * w_hidden[0][3] + b_hidden[0]);
71 hidden_output[1] = ReLU(in1 * w_hidden[1][0] + in2 * w_hidden[1][1] + in3 * w_hidden[1][2] + in4 * w_hidden[1][3] + b_hidden[1]);
72 hidden_output[2] = ReLU(in1 * w_hidden[2][0] + in2 * w_hidden[2][1] + in3 * w_hidden[2][2] + in4 * w_hidden[2][3] + b_hidden[2]);
73 end
74
75 // Compute output layer values and apply ReLU
76 always @(*) begin
77 out1 = ReLU(hidden_output[0] * w_output[0][0] + hidden_output[1] * w_output[0][1] + hidden_output[2] * w_output[0][2] + b_output[0]);
78 out2 = ReLU(hidden_output[0] * w_output[1][0] + hidden_output[1] * w_output[1][1] + hidden_output[2] * w_output[1][2] + b_output[1]);
79 end
```

3.2 Weights and Bias Initialization

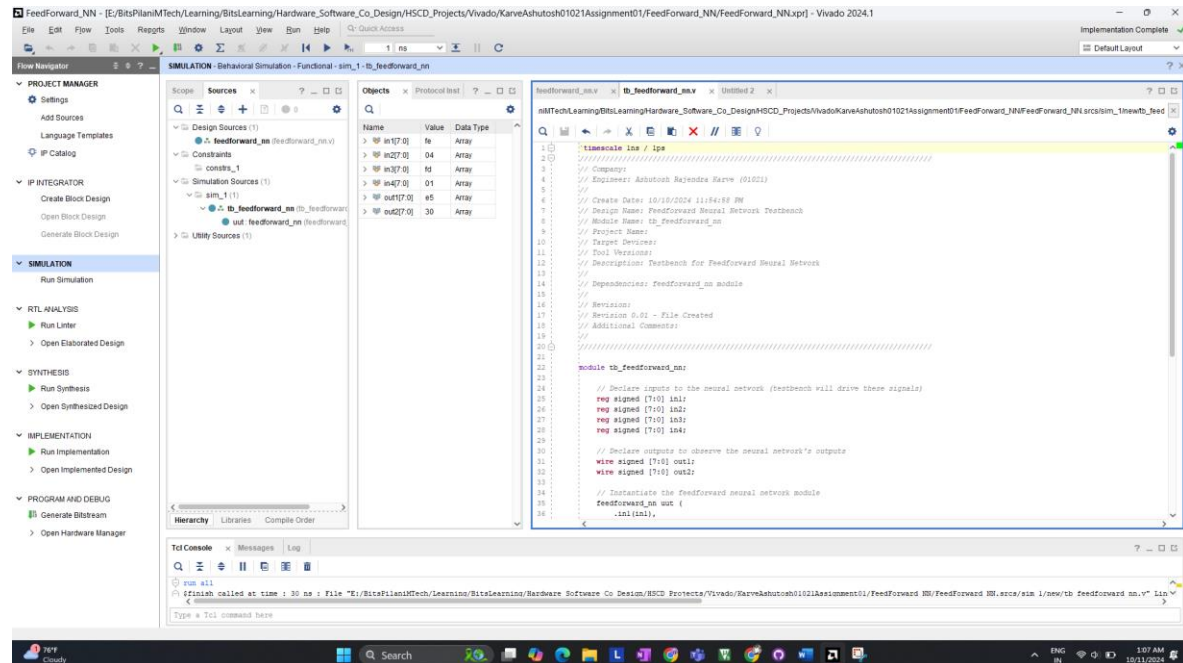
The weights and biases for both the hidden layer and the output layer were initialized as signed 8-bit values.

```
feedforward_nn.v  x  tb_feedforward_nn.v  x  Untitled 2  x
IMTechLearningBtlLearningHardware_Software_Co_DesignHSCD_Projects/VivadoKaveAshutosh01021Assignment01FeedForward_NN/FeedForward_NN1.srcs/sources_1/newfeedforwa

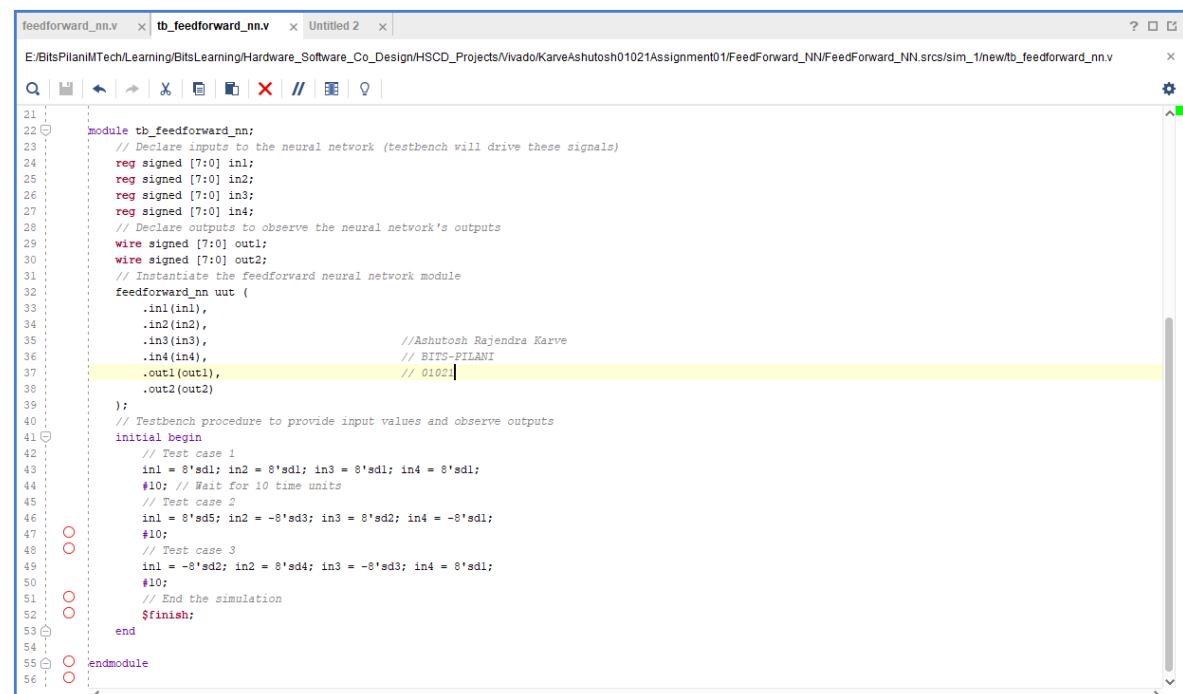
33 // Internal signals for hidden layer outputs
34 reg signed [7:0] hidden_output[0:2]; // 3 neurons in the hidden layer
35
36 // Weights and biases for hidden and output layers
37 reg signed [7:0] w_hidden[0:2][0:3]; // 3 hidden neurons, each with 4 inputs
38 reg signed [7:0] b_hidden[0:2]; // Biases for hidden neurons
39 reg signed [7:0] w_output[0:1][0:2]; // 2 output neurons, each with 3 inputs (from hidden layer)
40 reg signed [7:0] b_output[0:1]; // Biases for output neurons
41
42 // Initialize weights and biases
43 initial begin
44 // Initialize hidden layer weights and biases
45 w_hidden[0][0] = 0'sd1; w_hidden[0][1] = 0'sd2; w_hidden[0][2] = -0'sd1; w_hidden[0][3] = 0'sd1;
46 w_hidden[1][0] = 0'sd0; w_hidden[1][1] = 0'sd1; w_hidden[1][2] = -0'sd2; w_hidden[1][3] = 0'sd2;
47 w_hidden[2][0] = 0'sd3; w_hidden[2][1] = -0'sd1; w_hidden[2][2] = 0'sd2; w_hidden[2][3] = -0'sd1;
48
49 b_hidden[0] = 0'sd1; b_hidden[1] = -0'sd1; b_hidden[2] = 0'sd0;
50
51 // Initialize output layer weights and biases
52 w_output[0][0] = 0'sd2; w_output[0][1] = -0'sd3; w_output[0][2] = 0'sd1;
53 w_output[1][0] = 0'sd1; w_output[1][1] = 0'sd2; w_output[1][2] = -0'sd1;
54
55 b_output[0] = 0'sd1; b_output[1] = -0'sd2;
56 end
57
58 // ReLU activation function
59 function signed [7:0] ReLU(input signed [7:0] x);
60 begin
61 if (x > 0'sd0)
62 ReLU = x;
63 else
64 ReLU = 0'sd0;
65 end
66 endfunction
67
68 // Compute hidden layer outputs
```

4. Testbench Design

The testbench was created to validate the functionality of the feedforward neural network. The testbench provides input values to the network and captures the outputs. **Three test cases** were used to assess the behavior of the network.



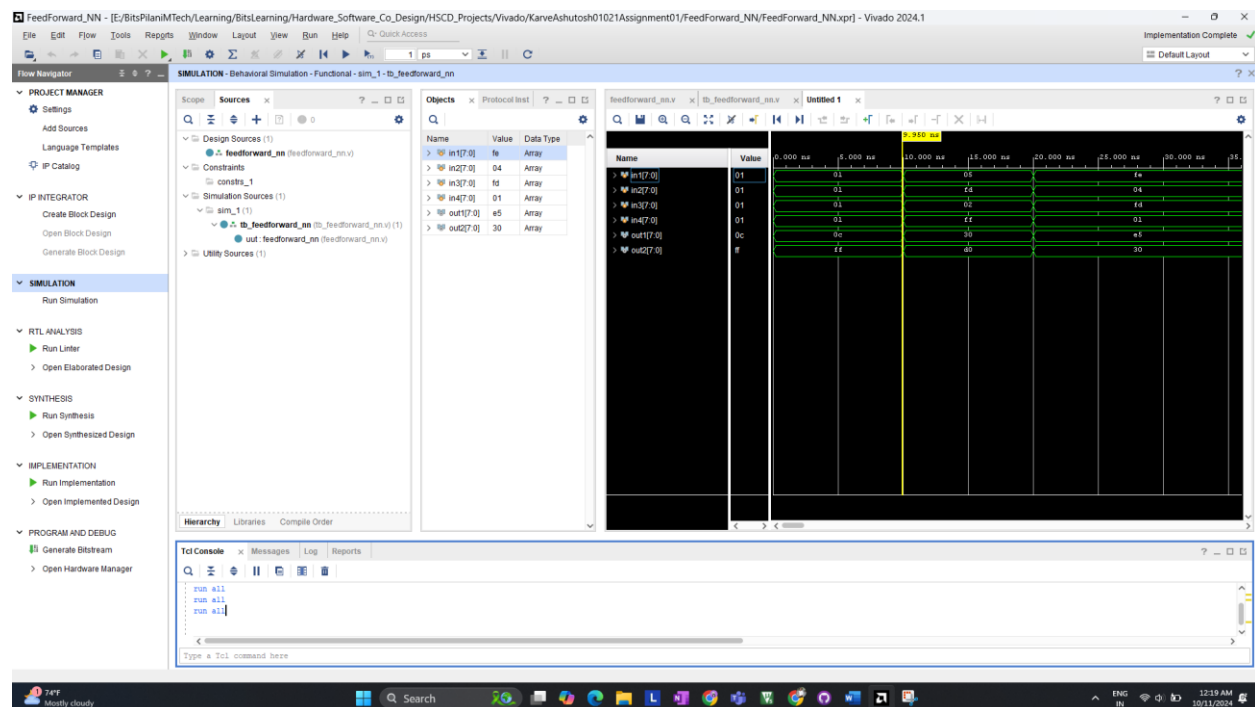
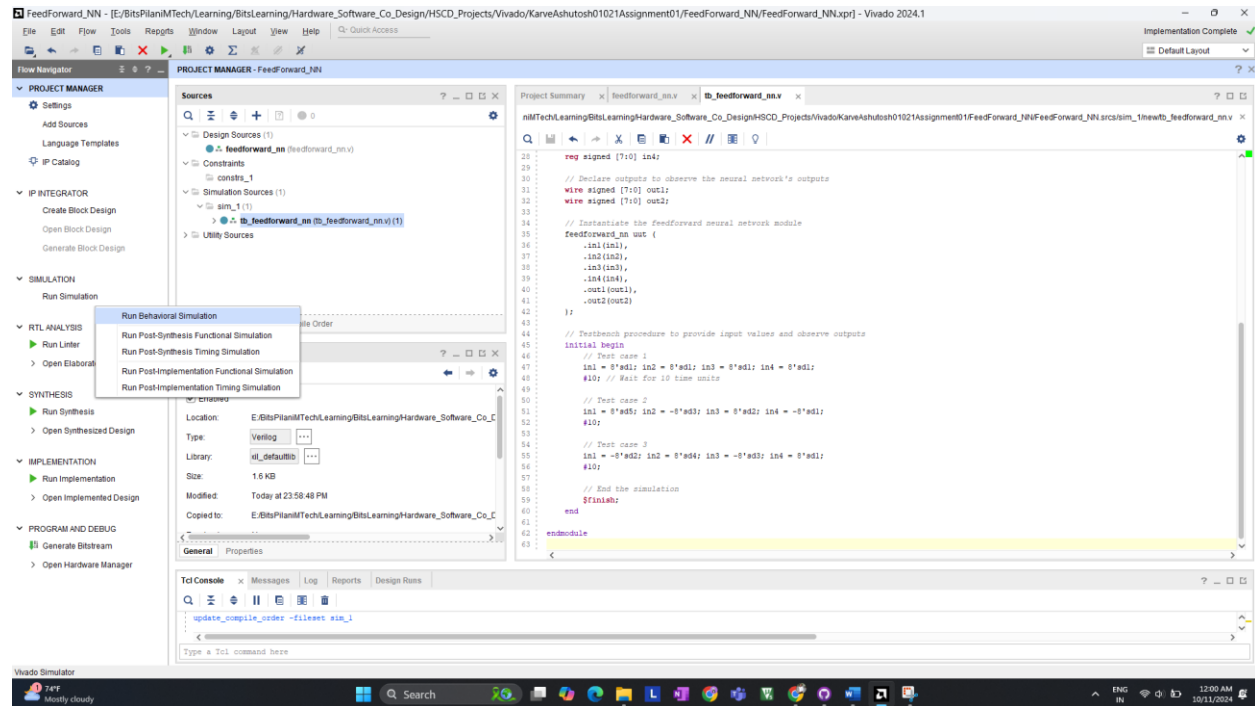
4.1 Testbench Code



5. Simulation Results

5.1 Running the Simulation

The testbench was run using Vivado's behavioral simulation. The waveform viewer shows the changes in inputs and corresponding outputs of the network.

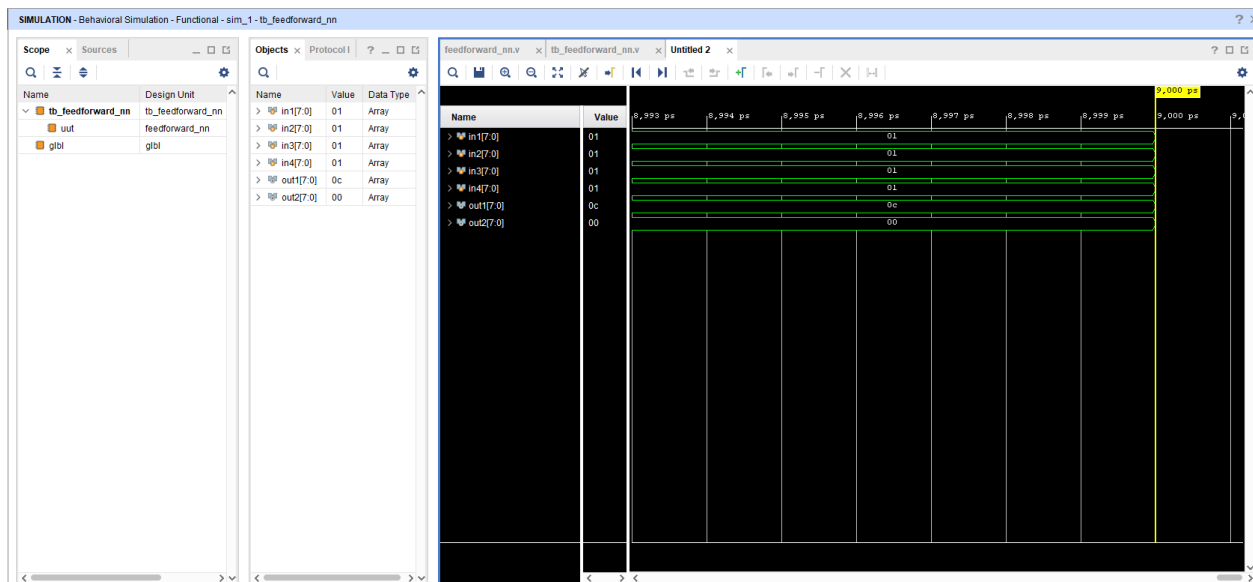


5.2 Test Case 1:

- Inputs:
 - in1 = 01, in2 = 01, in3 = 01, in4 = 01
- Outputs:
 - out1 = 0c (12 in decimal)
 - out2 = 00 (0 in decimal)

Explanation:

- out1 = 0c (12) indicates that the neural network computed a positive result for the first output neuron.
- out2 = 00 is expected, as the network output likely produced a negative result before ReLU was applied, which ReLU correctly converted to 0.

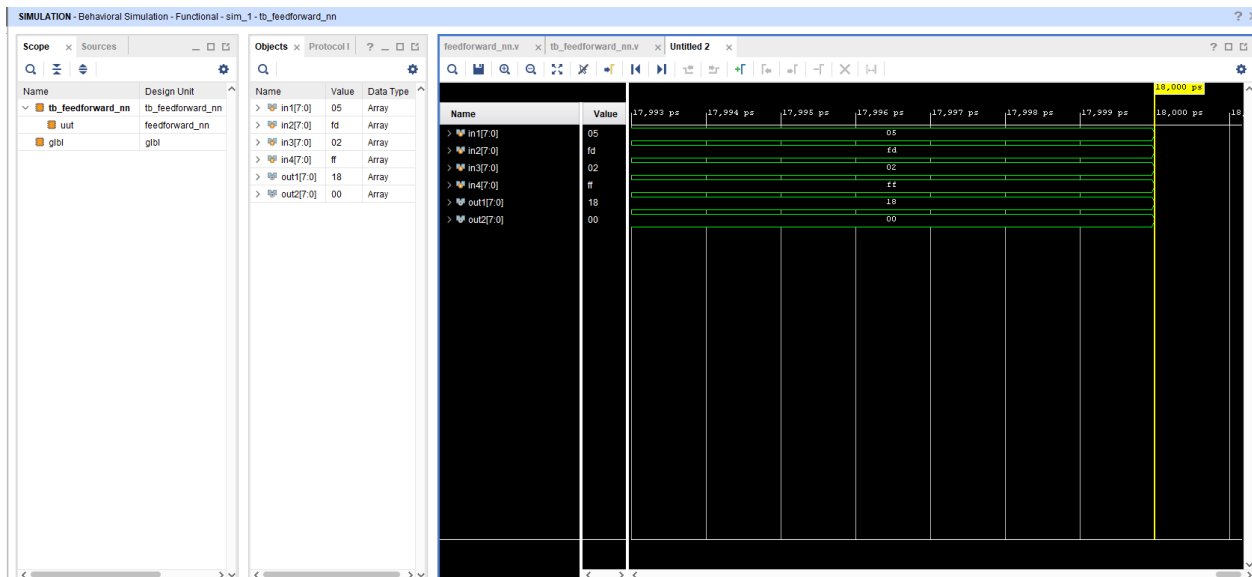


5.3 Test Case 2:

- Inputs:
 - in1 = 05, in2 = fd (-3 in decimal), in3 = 02, in4 = ff (-1 in decimal)
- Outputs:
 - out1 = 18 (24 in decimal)
 - out2 = 00 (0 in decimal)

Explanation:

- out1 = 18 (24) is a positive output, indicating that the input combination resulted in a valid, non-negative result after ReLU.
- out2 = 00 means that the network's second output neuron computed a negative value before ReLU was applied, and ReLU converted it to 0.

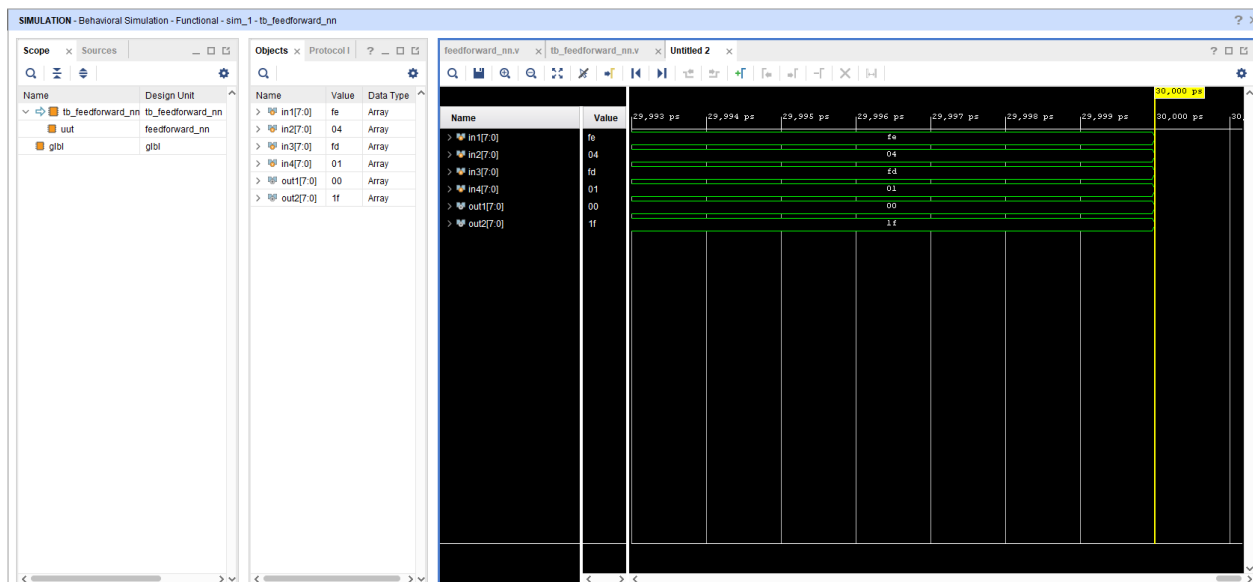


5.4 Test Case 3:

- Inputs:
 - in1 = fe (-2 in decimal), in2 = 04, in3 = fd (-3 in decimal), in4 = 01
- Outputs:
 - out1 = 00 (0 in decimal)
 - out2 = 1f (31 in decimal)

Explanation:

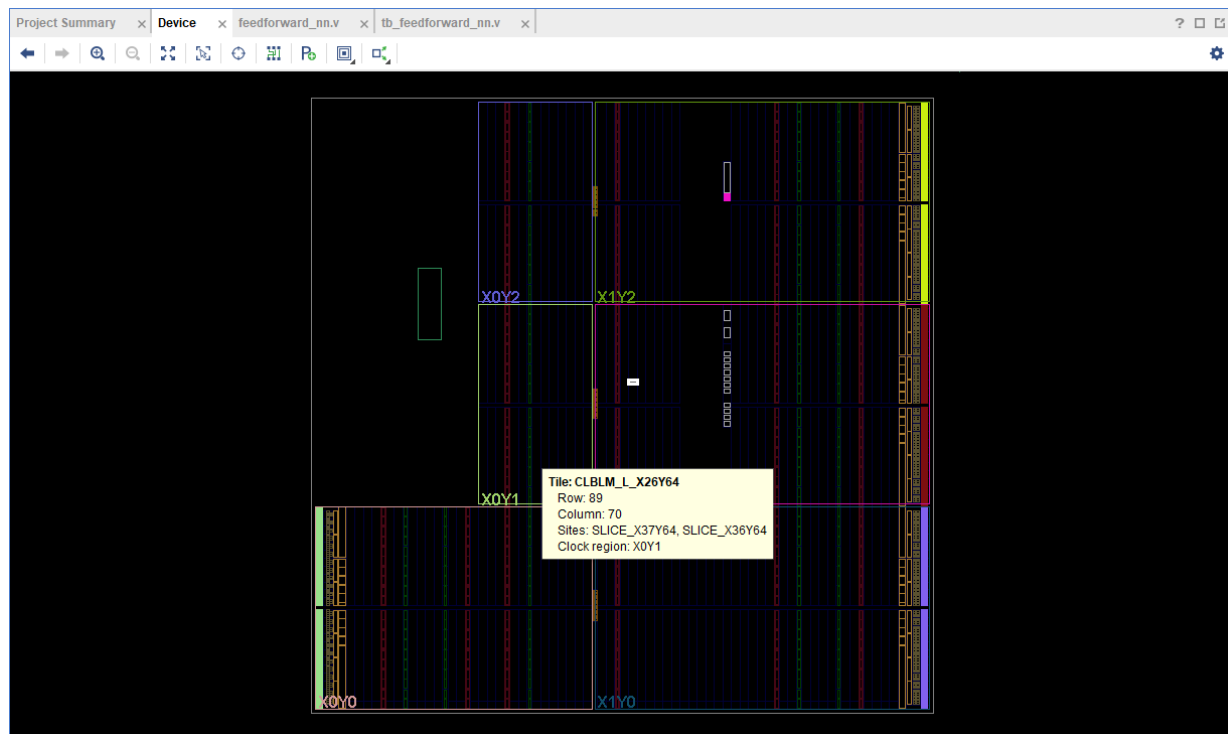
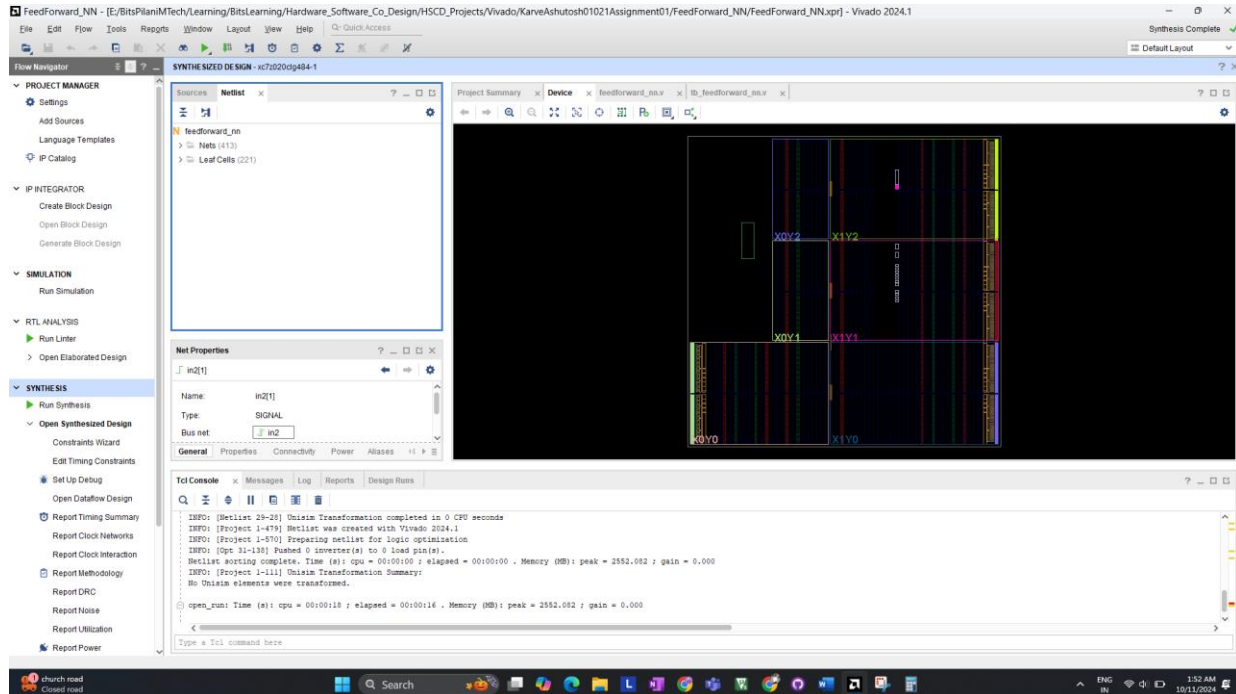
- out1 = 00 shows that the first output neuron computed a negative result before ReLU, which was correctly zeroed out.
- out2 = 1f (31) is a valid positive result from the second output neuron, which was not affected by ReLU since it was non-negative.



6. Synthesis and Elaboration

6.1 Synthesis Results

The design was successfully synthesized, and the resulting netlist was generated.



6.2 Elaboration Results

The elaboration process verified that the design was correctly structured.

FeedForward_NN - [E:/BitsPianiM/Tech/Learning/BitsLearning/Hardware_Software_Co_Design/HSCD_Projects/Vivado/KarveAshutosh01021Assignment01/FeedForward_NN/FeedForward_NN.xpr] - Vivado 2024.1

Implementation Complete ✓

Default Layout

Flow Navigator

- Language Templates
- IP Catalog
- IP INTEGRATOR
 - Create Block Design
 - Open Block Design
 - Generate Block Design
- SIMULATION
 - Run Simulation
- RTL ANALYSIS
 - Run Linter
 - Open Elaborated Design
 - Report Methodology
 - Report DRC
 - Report Noise
 - Schematic
- SYNTHESIS
 - Run Synthesis
 - Open Synthesized Design
 - Constraints Wizard
 - Edit Timing Constraints
 - Set Up Debug
 - Open Dataflow Design
 - Report Timing Summary
 - Report Clock Networks
 - Report Clock Interaction
 - Report Methodology
 - Report DRC
 - Report Noise
 - Report Utilization

ELABORATED DESIGN - xc7z020d484-1

Sources Netlist

Project Summary

Schematic

Properties

Select an object to see properties

Tcl Console

Messages

Log

Reports

Design Runs

Name	Constraints	Status	WNS	TNS	WHS	THS	WBSS	TPWS	Total Power	Failed Routes	Methodology	RQA Score	QoR Suggestions	LUT	FF	BRAM	URAM	DSP	Start	Elapsed	Run Strategy
✓ synth_1	constrs_1	synth_design Complete!	NA	NA	NA	NA	NA	NA	13.677	0				134	0	0	0	0	10/11/24, 9:11 AM	00:00:43	Vivado Synthesis Defaults (V)
✓ impl_1	constrs_1	route_design Complete!	NA	NA	NA	NA	NA	NA	13.677	0				134	0	0	0	0	10/11/24, 9:12 AM	00:01:13	Vivado Implementation Defaults (I)

Sim Time: 30 ns

ELABORATED DESIGN - xc7z020d484-1

Sources Netlist

Project Summary

Schematic

Properties

Select an object to see properties

Tcl Console

Messages

Log

Reports

Design Runs

Name	Constraints	Status	WNS	TNS	WHS	THS	WBSS	TPWS	Total Power	Failed Routes	Methodology	RQA Score	QoR Suggestions	LUT	FF	BRAM	URAM	DSP	Start	Elapsed	Run Strategy
✓ synth_1	constrs_1	synth_design Complete!	NA	NA	NA	NA	NA	NA	13.677	0				134	0	0	0	0	10/11/24, 9:11 AM	00:00:43	Vivado Synthesis Defaults (V)
✓ impl_1	constrs_1	route_design Complete!	NA	NA	NA	NA	NA	NA	13.677	0				134	0	0	0	0	10/11/24, 9:12 AM	00:01:13	Vivado Implementation Defaults (I)

Sim Time: 30 ns

7. Conclusion

This assignment involved the successful implementation and testing of a simple feedforward neural network with ReLU activation using Verilog HDL. The simulation results verified the functionality of the network, confirming that the design behaved as expected for each test case.