

# Assignment No. 1

HSCD

Submitted by:

**Name: Mukund Vishwas Chavan**

**Student ID: 01011**

**Student E-mail: [2024ht01011@wilp.bits-pilani.ac.in](mailto:2024ht01011@wilp.bits-pilani.ac.in)**

Design and implement a simple feedforward neural network using Verilog HDL on an FPGA. The network will have one hidden layer and will perform a basic classification task.

### 1. Network Architecture:

- Input Layer:

Inputs: Four 8-bit signed inputs (in1, in2, in3, in4) represent the features of the system.

- Hidden Layer:

Neurons: Three neurons calculate weighted sums of the inputs. Each neuron takes four weighted inputs and a bias term, then applies a ReLU activation:

$$\text{hidden\_out} = \text{ReLU}(w_1 \times \text{in1} + w_2 \times \text{in2} + w_3 \times \text{in3} + w_4 \times \text{in4} + \text{bias})$$

Activation Function: Implemented as:

$$\text{ReLU}(x) = \max(0, x)$$

- Output Layer:

Neurons: Two neurons calculate weighted sums of the hidden layer outputs and apply ReLU:

$$\text{output} = \text{ReLU}(w_1 \times \text{hidden\_out1} + w_2 \times \text{hidden\_out2} + w_3 \times \text{hidden\_out3} + \text{bias})$$

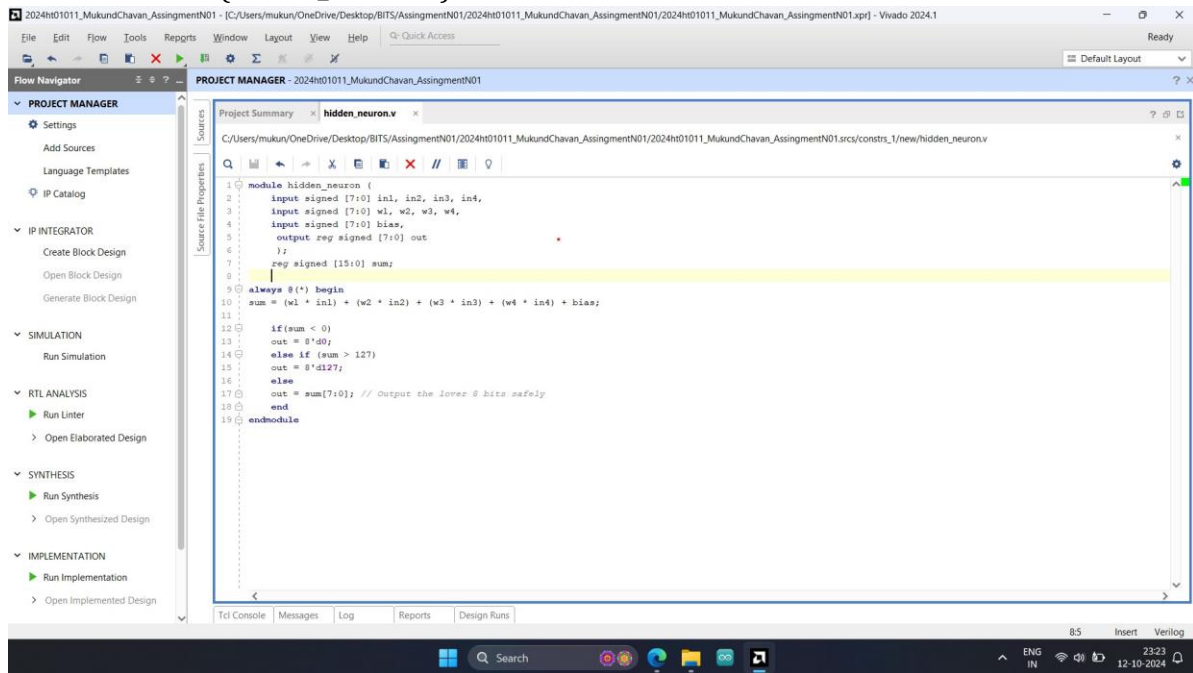
### 2. Weights and Biases:

- For simplicity, the neural network uses 8-bit signed integers for weights and biases. This choice allows easy representation and computation while ensuring that the values fit within the operational range of the model.

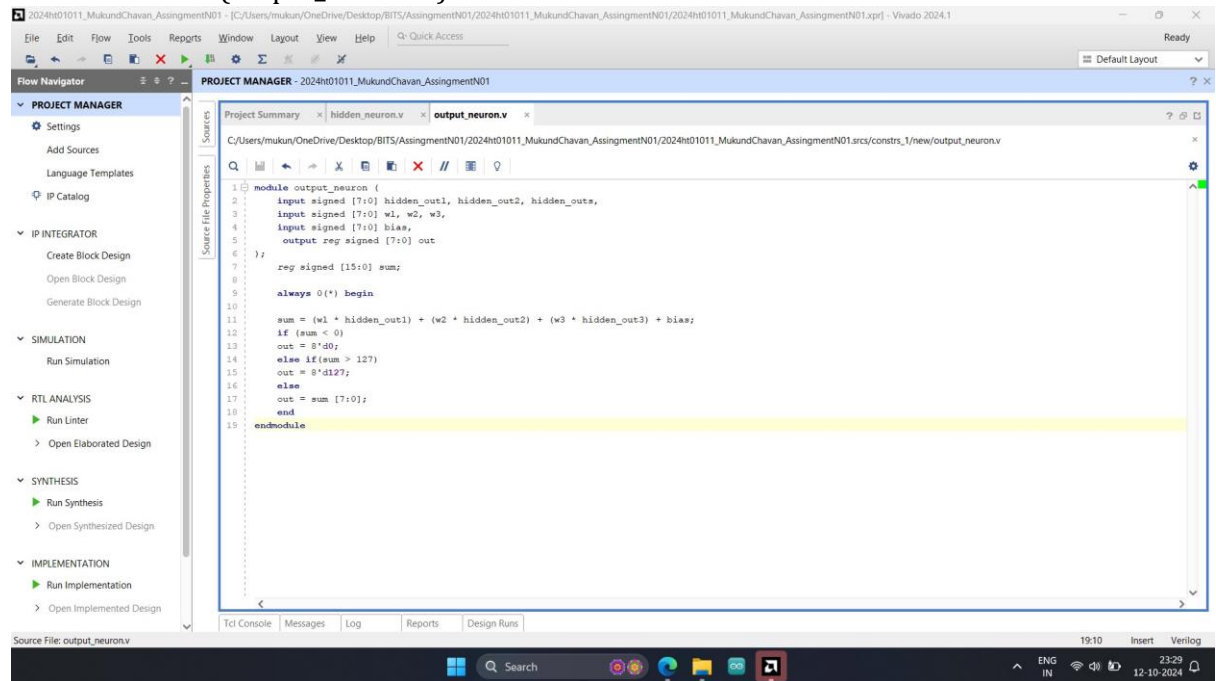
### 3. Design Implementation and Verilog Code Overview

- a) feedforward\_nn: The top-level module that integrates the input layer, hidden layer, and output layer.
- b) hidden\_neuron: A module representing each neuron in the hidden layer, implementing the weighted sum and ReLU activation.
- c) output\_neuron: A module for output neurons, similar to hidden neurons but specifically for combining hidden layer outputs.

## ➤ Hidden Neuron Module (hidden\_neuron.v)



## ➤ Output Neuron Module (output\_neuron.v)



## ➤ Top module (feedforward\_nm.V)

The screenshot shows the Vivado IDE interface. The Project Manager on the left lists the project files: Settings, Add Sources, 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), SYNTHESIS (Run Synthesis, Open Synthesized Design), IMPLEMENTATION (Run Implementation, Open Implemented Design). The main editor displays the code for the feedforward\_nm.v module. The code defines a module with inputs in1, in2, in3, in4, out1, out2, h1\_w1, h1\_w2, h1\_w3, h1\_w4, h2\_w1, h2\_w2, h2\_w3, h2\_w4, h3\_w1, h3\_w2, h3\_w3, h3\_w4, o1\_w1, o1\_w2, o1\_w3, o2\_w1, o2\_w2, o2\_w3, bias1, bias2, bias3, out\_bias1, and out\_bias2. It instantiates three hidden\_neuron blocks (h1, h2, h3) and calculates the output out1 based on the hidden outputs and weights. The output out1 is then compared to a threshold (127) to produce the final output out2.

```
1 module feedforward_nm (
2     input signed [7:0] in1, in2, in3, in4,
3     output reg signed [7:0] out1, out2,
4     input signed [7:0] h1_w1, h1_w2, h1_w3, h1_w4,
5     input signed [7:0] h2_w1, h2_w2, h2_w3, h2_w4,
6     input signed [7:0] h3_w1, h3_w2, h3_w3, h3_w4,
7     input signed [7:0] o1_w1, o1_w2, o1_w3,
8     input signed [7:0] o2_w1, o2_w2, o2_w3,
9     input signed [7:0] bias1, bias2, bias3,
10    input signed [7:0] out_bias1, out_bias2
11);
12    wire signed [7:0] hidden_out1, hidden_out2, hidden_out3;
13
14    hidden_neuron h1(.in1(in1), .in2(in2), .in3(in3), .in4(in4),
15    .w1(h1_w1), .w2(h1_w2), .w3(h1_w3), .w4(h1_w4),
16    .bias(bias1), .out(hidden_out1));
17
18    hidden_neuron h2(.in1(in1), .in2(in2), .in3(in3), .in4(in4),
19    .w1(h2_w1), .w2(h2_w2), .w3(h2_w3), .w4(h2_w4),
20    .bias(bias1), .out(hidden_out2));
21
22    hidden_neuron h3(.in1(in1), .in2(in2), .in3(in3), .in4(in4),
23    .w1(h3_w1), .w2(h3_w2), .w3(h3_w3), .w4(h3_w4),
24    .bias(bias1), .out(hidden_out3));
25
26    always @(*) begin
27        out1 = (o1_w1 * hidden_out1) + (o1_w2 * hidden_out2) + (o1_w3 * hidden_out3) + out_bias1;
28        if (out1 < 0)
29            out1 = 0'd0;
30        else if (out1 > 127)
31            out1 = 0'd127;
32    end
33    out2 = out1[7:0];
34endmodule
```

The screenshot shows the Vivado IDE interface. The Project Manager on the left lists the project files: Settings, Add Sources, 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), SYNTHESIS (Run Synthesis, Open Synthesized Design), IMPLEMENTATION (Run Implementation, Open Implemented Design). The main editor displays the code for the feedforward\_nm.v module. The code defines a module with inputs in1, in2, in3, in4, out1, out2, h1\_w1, h1\_w2, h1\_w3, h1\_w4, h2\_w1, h2\_w2, h2\_w3, h2\_w4, h3\_w1, h3\_w2, h3\_w3, h3\_w4, o1\_w1, o1\_w2, o1\_w3, o2\_w1, o2\_w2, o2\_w3, bias1, bias2, bias3, out\_bias1, and out\_bias2. It instantiates three hidden\_neuron blocks (h1, h2, h3) and calculates the output out1 based on the hidden outputs and weights. The output out1 is then compared to a threshold (127) to produce the final output out2.

```
13 hidden_neuron h1(.in1(in1), .in2(in2), .in3(in3), .in4(in4),
14 .w1(h1_w1), .w2(h1_w2), .w3(h1_w3), .w4(h1_w4),
15 .bias(bias1), .out(hidden_out1));
16
17 hidden_neuron h2(.in1(in1), .in2(in2), .in3(in3), .in4(in4),
18 .w1(h2_w1), .w2(h2_w2), .w3(h2_w3), .w4(h2_w4),
19 .bias(bias1), .out(hidden_out2));
20
21 hidden_neuron h3(.in1(in1), .in2(in2), .in3(in3), .in4(in4),
22 .w1(h3_w1), .w2(h3_w2), .w3(h3_w3), .w4(h3_w4),
23 .bias(bias1), .out(hidden_out3));
24
25 always @(*) begin
26     out1 = (o1_w1 * hidden_out1) + (o1_w2 * hidden_out2) + (o1_w3 * hidden_out3) + out_bias1;
27     if (out1 < 0)
28         out1 = 0'd0;
29     else if (out1 > 127)
30         out1 = 0'd127;
31     else
32         out1 = out1[7:0];
33 end
34
35 out2 = (o2_w1 * hidden_out1) + (o2_w2 * hidden_out2) + (o2_w3 * hidden_out3) + out_bias2;
36
37 if (out1 < 0)
38     out2 = 0'd0;
39 else if (out2 > 127)
40     out2 = 0'd127;
41
42 end
43 endmodule
```

## 4) Testbench and Verification.

### ➤ Testbench

The screenshot displays the Vivado 2024.1 IDE during a behavioral simulation. The **Project Manager** on the left shows the project structure with **SIMULATION** selected. The **Scope** table lists the testbench and its components:

Name	Design
testbench	testbe
nn	feedfc
gbl	gbl

The **Objects** table shows the current state of the design:

Name	Value	Data
in1[7:0]	0a	Array
in2[7:0]	0f	Array
in3[7:0]	14	Array
in4[7:0]	ec	Array
out1[7:0]	00	Array
out2[7:0]	00	Array
h1_w1[7:0]	01	Array
h1_w2[7:0]	02	Array
h1_w3[7:0]	03	Array
h1_w4[7:0]	04	Array
h2_w1[7:0]	ff	Array
h2_w2[7:0]	02	Array
h2_w3[7:0]	fd	Array
h2_w4[7:0]	04	Array
h3_w1[7:0]	02	Array

The **Verilog Code Editor** shows the **testbench.v** file with the following code:

```
19 //////////////////////////////////////////////////
20 module testbench();
21 // Input signals
22 reg signed [7:0] in1, in2, in3, in4;
23
24 // Output signals
25 wire signed [7:0] out1, out2;
26
27 // Weights and biases for the hidden layer
28 reg signed [7:0] h1_w1, h1_w2, h1_w3, h1_w4;
29 reg signed [7:0] h2_w1, h2_w2, h2_w3, h2_w4;
30 reg signed [7:0] h3_w1, h3_w2, h3_w3, h3_w4;
31 reg signed [7:0] bias1, bias2, bias3;
32
33 // Weights and biases for the output layer
34 reg signed [7:0] o1_w1, o1_w2, o1_w3;
35 reg signed [7:0] o2_w1, o2_w2, o2_w3;
36 reg signed [7:0] out_bias1, out_bias2;
37
38 // Instantiate the neural network
```

The **Tcl Console** shows the simulation results:

```
# }
# }
# run 1000ns
Time: 0, Inputs: ( 10, 20, 30, 40), Outputs: ( 0, 100)
Time: 10, Inputs: (-15, 20, -25, 30), Outputs: ( 70, 55)
```

The screenshot displays the Vivado 2024.1 IDE during a behavioral simulation. The **Project Manager** on the left shows the project structure with **SIMULATION** selected. The **Scope** table lists the testbench and its components:

Name	Design
testbench	testbe
nn	feedfc
gbl	gbl

The **Objects** table shows the current state of the design:

Name	Value	Data
in1[7:0]	0a	Array
in2[7:0]	0f	Array
in3[7:0]	14	Array
in4[7:0]	ec	Array
out1[7:0]	00	Array
out2[7:0]	00	Array
h1_w1[7:0]	01	Array
h1_w2[7:0]	02	Array
h1_w3[7:0]	03	Array
h1_w4[7:0]	04	Array
h2_w1[7:0]	ff	Array
h2_w2[7:0]	02	Array
h2_w3[7:0]	fd	Array
h2_w4[7:0]	04	Array
h3_w1[7:0]	02	Array

The **Verilog Code Editor** shows the **testbench.v** file with the following code:

```
43 .h2_w1(h2_w1), .h2_w2(h2_w2), .h2_w3(h2_w3), .h2_w4(h2_w4),
44 .h3_w1(h3_w1), .h3_w2(h3_w2), .h3_w3(h3_w3), .h3_w4(h3_w4),
45 .bias1(bias1), .bias2(bias2), .bias3(bias3),
46 .o1_w1(o1_w1), .o1_w2(o1_w2), .o1_w3(o1_w3),
47 .o2_w1(o2_w1), .o2_w2(o2_w2), .o2_w3(o2_w3),
48 .out_bias1(out_bias1), .out_bias2(out_bias2)
49 );
50
51 initial begin
52 // Initialize weights and biases for the hidden and output layers
53 h1_w1 = 8'd1; h1_w2 = 8'd2; h1_w3 = 8'd3; h1_w4 = 8'd4;
54 h2_w1 = -8'd1; h2_w2 = 8'd2; h2_w3 = -8'd1; h2_w4 = 8'd4;
55 h3_w1 = 8'd1; h3_w2 = 8'd1; h3_w3 = 8'd2; h3_w4 = 8'd1;
56 bias1 = 8'd1; bias2 = -8'd1; bias3 = 8'd1;
57
58 o1_w1 = 8'd1; o1_w2 = 8'd2; o1_w3 = 8'd3;
59 o2_w1 = -8'd1; o2_w2 = 8'd1; o2_w3 = -8'd1;
60 out_bias1 = 8'd1; out_bias2 = -8'd1;
61
62
```

The **Tcl Console** shows the simulation results:

```
# }
# }
# run 1000ns
Time: 0, Inputs: ( 10, 20, 30, 40), Outputs: ( 0, 100)
Time: 10, Inputs: (-15, 20, -25, 30), Outputs: ( 70, 55)
```

2024ht01011\_MukundChavan - [C:/Users/mukun/OneDrive/Desktop/BITS/AssingmentN01/2024ht01011\_MukundChavan\_AssingmentN01/2024ht01011\_MukundChavan/2024ht01011\_MukundChavan.xpr] - Vivado 2024.1

File Edit Flow Tools Reports Window Layout View Run Help Quick Access

Synthesis Out-of-date details

Default Layout

Flow Navigator

PROJECT MANAGER

- Settings
- Add Sources
- Language Templates
- IP Catalog
- IP INTEGRATOR
  - Create Block Design
  - Open Block Design
  - Generate Block Design
- SIMULATION**
  - Run Simulation
- RTL ANALYSIS
- SYNTHESIS
  - Run Synthesis
  - Open Synthesized Design
- IMPLEMENTATION
  - Run Implementation
  - Open Implemented Design
- PROGRAM AND DEBUG
  - Generate Bitstream

Scope Sources

Name	Design
testbench	testbe
nn	feedfc
gibl	gibl

Objects Protocol

Name	Value	Data
in1[7:0]	0a	Array
in2[7:0]	0f	Array
in3[7:0]	14	Array
in4[7:0]	ec	Array
out1[7:0]	00	Array
out2[7:0]	00	Array
h1_w1[7:0]	01	Array
h1_w2[7:0]	02	Array
h1_w3[7:0]	03	Array
h1_w4[7:0]	04	Array
h2_w1[7:0]	ff	Array
h2_w2[7:0]	02	Array
h2_w3[7:0]	fd	Array
h2_w4[7:0]	04	Array
h3_w1[7:0]	02	Array

testbench.v

```
67 in1 = -8'd15; in2 = 8'd20; in3 = -8'd25; in4 = 8'd30; //Case 2
68 #10;
69 // Test Case 3
70 in1 = 8'd10; in2 = 8'd15; in3 = 8'd20; in4 = -8'd20; //Case 3
71 #10; *
72 $finish;
73 end
74 // Monitor output during simulation
75 initial begin
76 $monitor("Time: %d, Inputs: (%d, %d, %d, %d), Outputs: (%d, %d)",
77 $time, in1, in2, in3, in4, out1, out2);
78 end
79 endmodule
```

Tcl Console

```
# }
# }
# run 1000ns
Time: 0, Inputs: ( 10, 20, 30, 40), Outputs: ( 0, 100)
Time: 10, Inputs: (-15, 20, -25, 30), Outputs: ( 70, 55)
```

Type a Tcl command here

67-9 Insert Verilog

## ➤ Simulation Results

### □ All Three cases

The screenshot shows the Vivado IDE interface for a behavioral simulation. The left sidebar contains the Project Manager, IP Integrator, and Simulation sections. The main workspace displays the testbench.v file with three test cases. The Tcl Console shows the simulation results for the first two cases.

**testbench.v**

```
out_bias1 = 0'd1; out_bias2 = -0'd1;  
  
in1 = 0'd10; in2 = 0'd20; in3 = 0'd30; in4 = 0'd40; //Case 1  
#10;  
  
in1 = -0'd15; in2 = 0'd20; in3 = -0'd25; in4 = 0'd30; //Case 2  
#10;  
  
// Test Case 3  
in1 = 0'd10; in2 = 0'd15; in3 = 0'd20; in4 = -0'd20; //Case 3  
#10;  
  
$finish;  
end  
  
// Monitor output during simulation  
initial begin
```

**Tcl Console**

```
# }  
# run 1000ns  
Time: 0, Inputs: ( 10, 20, 30, 40), Outputs: ( 0, 100)  
Time: 10, Inputs: (-15, 20, -25, 30), Outputs: ( 70, 55)
```

The screenshot shows the Vivado IDE interface for a behavioral simulation. The left sidebar contains the Project Manager, IP Integrator, and Simulation sections. The main workspace displays the testbench.v file with three test cases. The Tcl Console shows the simulation results for the first two cases. The right pane shows a waveform viewer with signals for in1, in2, in3, in4, out1, out2, h1\_w1, h1\_w2, h1\_w3, h1\_w4, h2\_w1, h2\_w2, h2\_w3, h2\_w4, and h3\_w1.

**testbench.v**

```
out_bias1 = 0'd1; out_bias2 = -0'd1;  
  
in1 = 0'd10; in2 = 0'd20; in3 = 0'd30; in4 = 0'd40; //Case 1  
#10;  
  
in1 = -0'd15; in2 = 0'd20; in3 = -0'd25; in4 = 0'd30; //Case 2  
#10;  
  
// Test Case 3  
in1 = 0'd10; in2 = 0'd15; in3 = 0'd20; in4 = -0'd20; //Case 3  
#10;  
  
$finish;  
end  
  
// Monitor output during simulation  
initial begin
```

**Tcl Console**

```
# }  
# run 1000ns  
Time: 0, Inputs: ( 10, 20, 30, 40), Outputs: ( 0, 100)  
Time: 10, Inputs: (-15, 20, -25, 30), Outputs: ( 70, 55)
```

**Waveform Viewer**

Name	Value
in1[7:0]	0a
in2[7:0]	14
in3[7:0]	1e
in4[7:0]	28
out1[7:0]	00
out2[7:0]	64
h1_w1[7:0]	01
h1_w2[7:0]	02
h1_w3[7:0]	03
h1_w4[7:0]	04
h2_w1[7:0]	ff
h2_w2[7:0]	02