# Design and Implementation of an 8-Tap FIR Filter

Debopama Basu (M.Tech SoCD) - 152402006

**Abstract**

Finite Impulse Response (FIR) filters are widely used in digital signal processing applications due to their inherent stability and linear phase characteristics. This report presents the design, implementation, and verification of an 8-tap FIR filter. The filter coefficients are provided in Q4.12 fixed-point format and the design includes a hardware architecture diagram, Verilog implementation, and simulation verification using input samples. The input and output samples are plotted and analyzed to validate the performance of the FIR filter.

## 1 Introduction

Digital filters play a crucial role in processing discrete-time signals for a variety of applications such as audio processing, communication systems, and biomedical signal analysis. Among digital filters, FIR filters are particularly valued for their stability and ease of implementation. FIR filters are non-recursive filters whose output depends only on the current and a finite number of past input samples.

In this experiment, we focus on the design and simulation of an 8-tap FIR filter. The filter utilizes 16-bit input and output samples represented in Q4.12 fixed-point format. The main objectives are:

- To design a hardware architecture for an 8-tap FIR filter.
- To implement the architecture in Verilog.
- To simulate the design using input data from a file and verify the output using MATLAB or Python.

## 2 8-Tap FIR Filter Design

An FIR filter of length $N$ has the general difference equation:

$$y(n) = \sum_{i=0}^{N-1} a_i \cdot x(n-i) \tag{1}$$

For this experiment, we have $N = 8$, and the filter equation becomes:

$$y(n) = a_0 \cdot x(n) + a_1 \cdot x(n-1) + a_2 \cdot x(n-2) + \cdots + a_7 \cdot x(n-7) \tag{2}$$

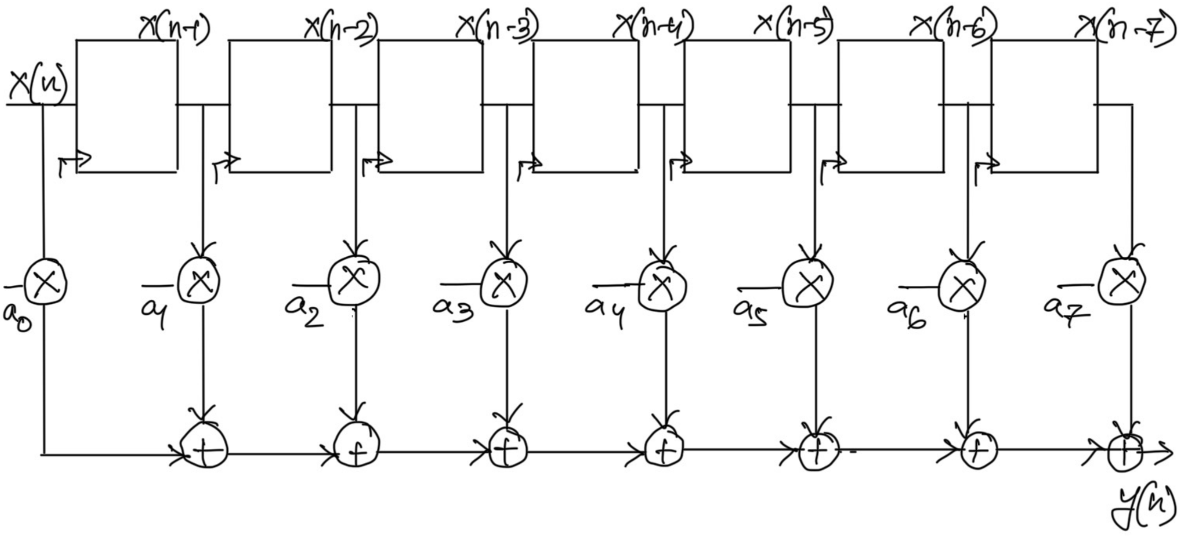


Figure 1: Block Diagram of an 8-tap FIR filter

### 2.1 Filter Coefficients

The filter coefficients are provided as follows:

$$a_0 = -0.0841, \quad a_1 = -0.0567, \quad a_2 = 0.1826, \quad a_3 = 0.4086,$$
$$a_4 = 0.4086, \quad a_5 = 0.1826, \quad a_6 = -0.0567, \quad a_7 = -0.0841$$

These coefficients are symmetric, indicating that the FIR filter has a linear phase response, which is beneficial in many signal processing applications.

### 2.2 Fixed-Point Representation

The filter operates on 16-bit input and output samples using the Q4.12 fixed-point format. In this format, 4 bits are used for the integer part (including the sign bit) and 12 bits for the fractional part. This representation allows for efficient implementation of arithmetic operations in hardware while maintaining sufficient precision.

### 2.3 Hardware Architecture

The hardware architecture of the FIR filter consists of:

- Shift registers to store the current and past seven input samples.
- Multipliers to multiply each input sample with the corresponding coefficient.
- Adders to accumulate the products and compute the final output sample.

The architecture can be implemented in a pipelined manner to improve throughput or in a sequential manner to save hardware resources.

## 3 Verilog Code to implement the 8-tap FIR filter

```

module fir_filter (
    input clk,
    input rst,
    input signed [15:0] x_in,
    output reg signed [15:0] y_out
);
    reg signed [15:0] x_reg[0:7];
    reg signed [27:0] acc;
    integer i;

    parameter signed [15:0] COEFF[0:7] = {
        -345,    // -0.0841
        -232,    // -0.0567
         748,    // 0.1826
        1674,    // 0.4086
        1674,    // 0.4086
         748,    // 0.1826
        -232,    // -0.0567
        -345     // -0.0841
    };

    always @(posedge clk or posedge rst) begin
        if (rst) begin
            for (i = 0; i < 8; i = i + 1)
                x_reg[i] <= 0;
            y_out <= 0;
            acc <= 0;
        end else begin
            for (i = 7; i > 0; i = i - 1)
                x_reg[i] <= x_reg[i - 1];
            x_reg[0] <= x_in;

            acc = 0;
            for (i = 0; i < 8; i = i + 1)
                acc = acc + x_reg[i] * COEFF[i];

            y_out <= acc[27:12];  // Scale back to Q4.12
        end
    end
endmodule
```

Listing 1: Top Module

## 4 Testbench Code with a Golden reference for comparing the output values of the Verilog Code.

```
module fir_tb;
    reg clk = 0;
    reg rst = 1;
    reg signed [15:0] x_in;
    wire signed [15:0] y_out;
```

```
    integer input_file, output_file, i;
    reg [15:0] hex_in;
    real float_in, float_out, golden_out, prev_golden_out, error;

    real coeffs[0:7];
    real samples[0:7];

    fir_filter uut (
        .clk(clk),
        .rst(rst),
        .x_in(x_in),
        .y_out(y_out)
    );

    always #5 clk = ~clk;

    initial begin
        coeffs[0] = -0.0841; coeffs[1] = -0.0567; coeffs[2] = 0.1826; coeffs[3] = 0.4086;
        coeffs[4] = 0.4086;  coeffs[5] = 0.1826;  coeffs[6] = -0.0567; coeffs[7] = -0.0841;

        for (i = 0; i < 8; i = i + 1)
            samples[i] = 0.0;

        input_file = $fopen("Input_Data.txt", "r");
        output_file = $fopen("Output_Data.txt", "w");

        #10 rst = 0;
        prev_golden_out = 0.0;
        for (i = 0; i < 256; i = i + 1) begin
            $fscanf(input_file, "%h", hex_in);
            x_in = hex_in;
            float_in = $itor($signed(hex_in)) / 4096.0;

            samples[7] = samples[6];
            samples[6] = samples[5];
            samples[5] = samples[4];
            samples[4] = samples[3];
            samples[3] = samples[2];
            samples[2] = samples[1];
            samples[1] = samples[0];
            samples[0] = float_in;

            golden_out = 0.0;
            for (int j = 0; j < 8; j = j + 1)
                golden_out += coeffs[j] * samples[j];

            #10;

            float_out = $itor(y_out) / 4096.0;
            error = float_out - prev_golden_out;

            $fwrite(output_file, "%f\n", float_out);
            $display("Sample %0d: DUT = %f, Golden = %f, Error = %f", i, float_out, prev_golden_out, error);

```

```
            prev_golden_out = golden_out;
        end

        $fclose(input_file);
        $fclose(output_file);
        $stop;
    end
endmodule
```

Listing 2: Testbench Code

# Python Script for FIR Filter Output Analysis

```python
import re
import numpy as np
import matplotlib.pyplot as plt

data = """
Sample 0: DUT = 0.000000, Golden = 0.000000, Error = 0.000000
Sample 1: DUT = -0.161377, Golden = -0.161075, Error = -0.000302
Sample 2: DUT = -0.030762, Golden = -0.030903, Error = 0.000141
Sample 3: DUT = 0.290527, Golden = 0.290786, Error = -0.000259
Sample 4: DUT = 0.446045, Golden = 0.446156, Error = -0.000111
Sample 5: DUT = 0.469238, Golden = 0.469334, Error = -0.000096
Sample 6: DUT = 0.632080, Golden = 0.632027, Error = 0.000053
...
Sample 207: DUT = 0.042725, Golden = 0.042885, Error = -0.000160
"""

pattern = r"DUT = ([\-\d\.]+), Golden = ([\-\d\.]+), Error = ([\-\d\.]+)"
matches = re.findall(pattern, data)

dut_values = np.array([float(m[0]) for m in matches])
golden_values = np.array([float(m[1]) for m in matches])
error_values = np.array([float(m[2]) for m in matches])
sample_indices = np.arange(len(dut_values))

# Validate error calculation
calculated_errors = dut_values - golden_values
if not np.allclose(calculated_errors, error_values, atol=1e-6):
    print("Warning: Parsed errors do not match calculated errors.")

# Compute error metrics
mean_error = np.mean(error_values)
max_error = np.max(np.abs(error_values))
rmse = np.sqrt(np.mean(error_values**2))

print(f"Mean Error: {mean_error:.6f}")
print(f"Max Error: {max_error:.6f}")
print(f"RMSE: {rmse:.6f}")

# Plot 1: DUT vs Golden outputs
plt.figure(figsize=(12, 6))
plt.plot(sample_indices, dut_values, label='DUT Output', marker='o', linestyle='-')
```

```
plt.plot(sample_indices, golden_values, label='Golden Output', marker='
    x', linestyle='--')
plt.title('DUT vs Golden Outputs')
plt.xlabel('Sample Index')
plt.ylabel('Output Value')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# Plot 2: Error plot
plt.figure(figsize=(12, 4))
plt.plot(sample_indices, error_values, color='red', label='Error',
    marker='s')
plt.title('Error per Sample')
plt.xlabel('Sample Index')
plt.ylabel('Error Value')
plt.axhline(0, color='black', linewidth=0.8, linestyle='--')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Listing 3: Python Code to Parse and Plot FIR Filter Output

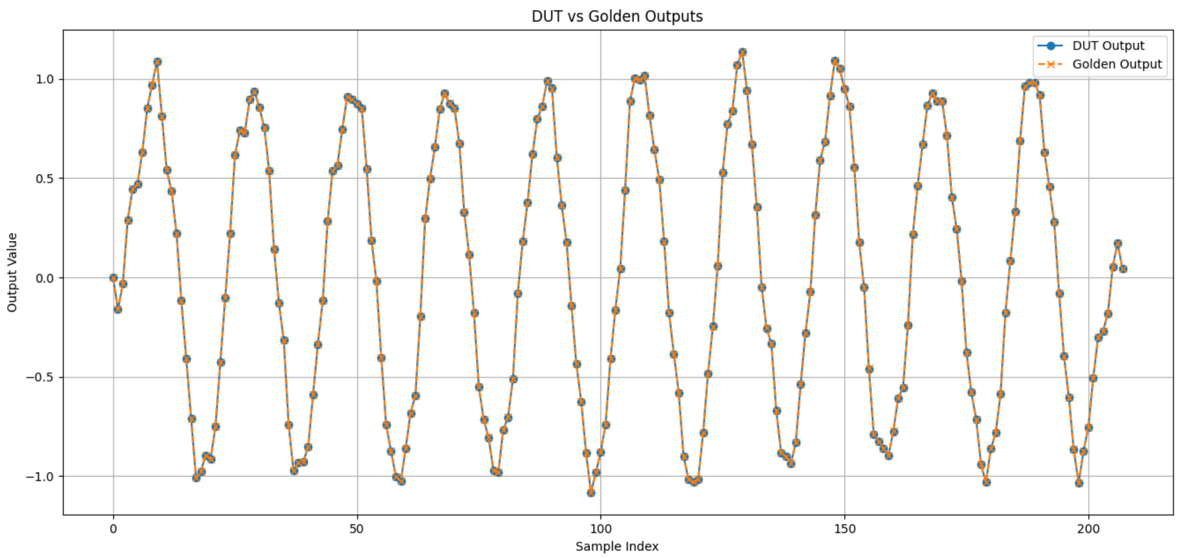


Figure 2: Plot comparing the FIR filter outputs against the golden reference

```
PS C:\Users\USER\OneDrive\Desktop\FIR_VLSI> python plot.py
Mean Error: -0.000130
Max Error: 0.000782
RMSE: 0.000304
```

Figure 3: Error Details from Python Code

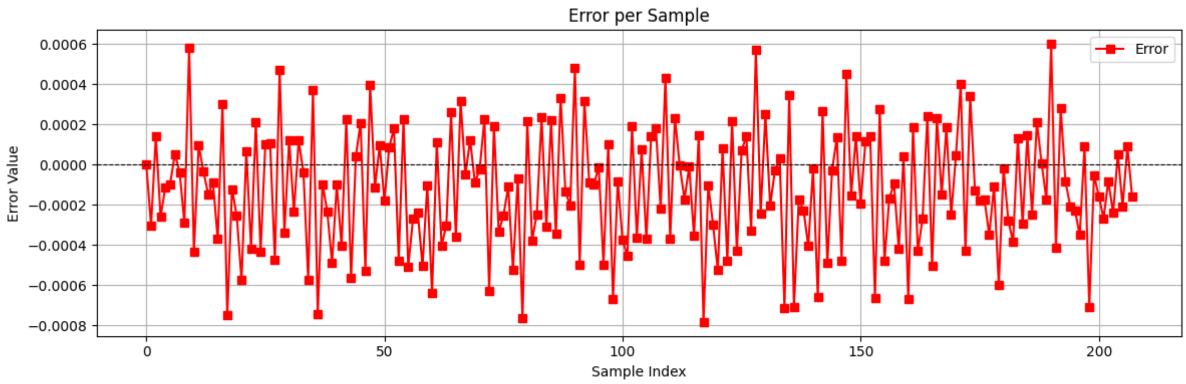


Figure 4: Error Plot

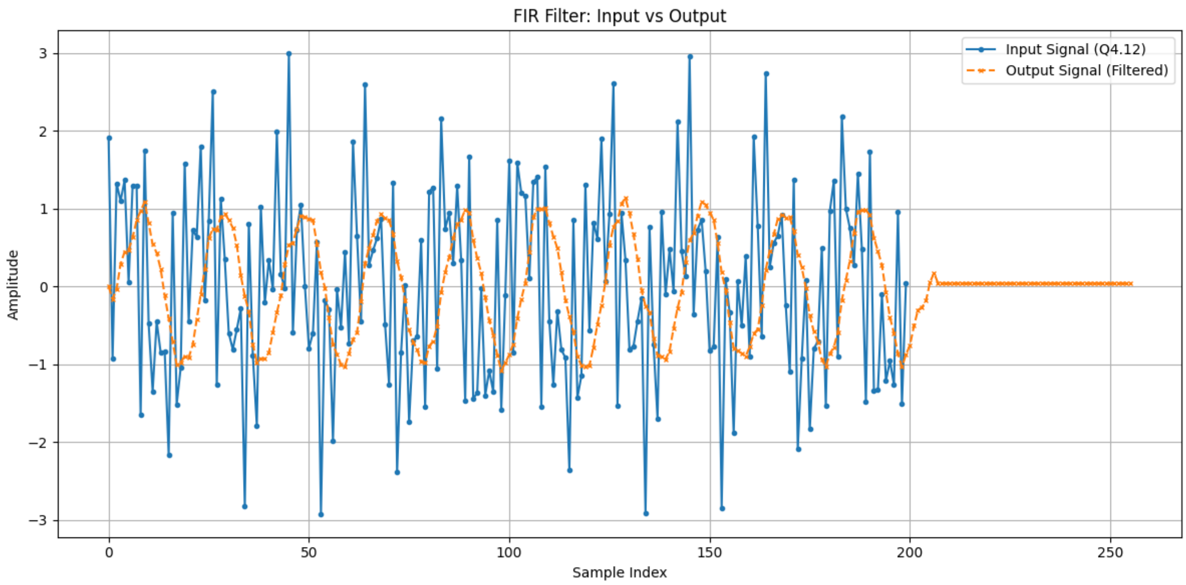


Figure 5: Input v/s output plot of FIR filter

# Verilog Simulation Output

```
Sample 0: DUT = 0.000000, Golden = 0.000000, Error = 0.000000
Sample 1: DUT = -0.161377, Golden = -0.161075, Error = -0.000302
Sample 2: DUT = -0.030762, Golden = -0.030903, Error = 0.000141
Sample 3: DUT = 0.290527, Golden = 0.290786, Error = -0.000259
Sample 4: DUT = 0.446045, Golden = 0.446156, Error = -0.000111
Sample 5: DUT = 0.469238, Golden = 0.469334, Error = -0.000096
Sample 6: DUT = 0.632080, Golden = 0.632027, Error = 0.000053
Sample 7: DUT = 0.851318, Golden = 0.851355, Error = -0.000037
Sample 8: DUT = 0.969482, Golden = 0.969772, Error = -0.000290
Sample 9: DUT = 1.088623, Golden = 1.088042, Error = 0.000581
Sample 10: DUT = 0.812256, Golden = 0.812689, Error = -0.000433
Sample 11: DUT = 0.539795, Golden = 0.539698, Error = 0.000097
Sample 12: DUT = 0.433105, Golden = 0.433140, Error = -0.000034
Sample 13: DUT = 0.223633, Golden = 0.223779, Error = -0.000147
Sample 14: DUT = -0.116211, Golden = -0.116123, Error = -0.000088
Sample 15: DUT = -0.408203, Golden = -0.407834, Error = -0.000369
```

```
Sample 16: DUT = -0.708740, Golden = -0.709041, Error = 0.000301
Sample 17: DUT = -1.007812, Golden = -1.007063, Error = -0.000749
Sample 18: DUT = -0.975586, Golden = -0.975465, Error = -0.000121
Sample 19: DUT = -0.895020, Golden = -0.894765, Error = -0.000255
Sample 20: DUT = -0.913818, Golden = -0.913244, Error = -0.000575
Sample 21: DUT = -0.749268, Golden = -0.749334, Error = 0.000067
Sample 22: DUT = -0.424561, Golden = -0.424145, Error = -0.000416
Sample 23: DUT = -0.103271, Golden = -0.103480, Error = 0.000209
Sample 24: DUT = 0.222900, Golden = 0.223333, Error = -0.000432
Sample 25: DUT = 0.616943, Golden = 0.616843, Error = 0.000100
Sample 26: DUT = 0.739990, Golden = 0.739885, Error = 0.000106
Sample 27: DUT = 0.729004, Golden = 0.729477, Error = -0.000473
Sample 28: DUT = 0.894775, Golden = 0.894306, Error = 0.000469
Sample 29: DUT = 0.936768, Golden = 0.937105, Error = -0.000337
Sample 30: DUT = 0.854736, Golden = 0.854615, Error = 0.000122
Sample 31: DUT = 0.755859, Golden = 0.756095, Error = -0.000235
Sample 32: DUT = 0.535645, Golden = 0.535525, Error = 0.000119
Sample 33: DUT = 0.143555, Golden = 0.143592, Error = -0.000037
Sample 34: DUT = -0.127930, Golden = -0.127358, Error = -0.000572
Sample 35: DUT = -0.316406, Golden = -0.316778, Error = 0.000372
Sample 36: DUT = -0.740723, Golden = -0.739982, Error = -0.000741
Sample 37: DUT = -0.972900, Golden = -0.972804, Error = -0.000096
Sample 38: DUT = -0.929932, Golden = -0.929699, Error = -0.000233
Sample 39: DUT = -0.927002, Golden = -0.926514, Error = -0.000488
Sample 40: DUT = -0.853516, Golden = -0.853415, Error = -0.000100
Sample 41: DUT = -0.590088, Golden = -0.589683, Error = -0.000404
Sample 42: DUT = -0.334717, Golden = -0.334943, Error = 0.000226
Sample 43: DUT = -0.114746, Golden = -0.114181, Error = -0.000565
Sample 44: DUT = 0.283447, Golden = 0.283405, Error = 0.000042
Sample 45: DUT = 0.535645, Golden = 0.535437, Error = 0.000208
Sample 46: DUT = 0.564941, Golden = 0.565469, Error = -0.000527
Sample 47: DUT = 0.745605, Golden = 0.745209, Error = 0.000397
Sample 48: DUT = 0.910400, Golden = 0.910513, Error = -0.000112
Sample 49: DUT = 0.895264, Golden = 0.895168, Error = 0.000095
Sample 50: DUT = 0.874756, Golden = 0.874935, Error = -0.000179
Sample 51: DUT = 0.850098, Golden = 0.850012, Error = 0.000086
Sample 52: DUT = 0.547363, Golden = 0.547183, Error = 0.000180
Sample 53: DUT = 0.185303, Golden = 0.185778, Error = -0.000476
Sample 54: DUT = -0.018311, Golden = -0.018535, Error = 0.000224
Sample 55: DUT = -0.403320, Golden = -0.402813, Error = -0.000507
Sample 56: DUT = -0.742676, Golden = -0.742408, Error = -0.000268
Sample 57: DUT = -0.875000, Golden = -0.874762, Error = -0.000238
Sample 58: DUT = -1.001465, Golden = -1.000961, Error = -0.000504
Sample 59: DUT = -1.025146, Golden = -1.025043, Error = -0.000103
Sample 60: DUT = -0.860107, Golden = -0.859470, Error = -0.000638
Sample 61: DUT = -0.681396, Golden = -0.681508, Error = 0.000112
Sample 62: DUT = -0.593018, Golden = -0.592616, Error = -0.000402
Sample 63: DUT = -0.196289, Golden = -0.195988, Error = -0.000301
Sample 64: DUT = 0.295898, Golden = 0.295640, Error = 0.000259
Sample 65: DUT = 0.499023, Golden = 0.499380, Error = -0.000357
Sample 66: DUT = 0.658203, Golden = 0.657888, Error = 0.000315
```

```
Sample 67: DUT = 0.847168, Golden = 0.847217, Error = -0.000049
Sample 68: DUT = 0.927002, Golden = 0.926879, Error = 0.000123
Sample 69: DUT = 0.876465, Golden = 0.876554, Error = -0.000090
Sample 70: DUT = 0.853027, Golden = 0.853050, Error = -0.000022
Sample 71: DUT = 0.676758, Golden = 0.676530, Error = 0.000228
Sample 72: DUT = 0.329834, Golden = 0.330459, Error = -0.000626
Sample 73: DUT = 0.116943, Golden = 0.116751, Error = 0.000192
Sample 74: DUT = -0.178711, Golden = -0.178377, Error = -0.000334
Sample 75: DUT = -0.550049, Golden = -0.549797, Error = -0.000252
Sample 76: DUT = -0.715332, Golden = -0.715222, Error = -0.000110
Sample 77: DUT = -0.804932, Golden = -0.804409, Error = -0.000523
Sample 78: DUT = -0.969238, Golden = -0.969173, Error = -0.000066
Sample 79: DUT = -0.979736, Golden = -0.978971, Error = -0.000765
Sample 80: DUT = -0.768066, Golden = -0.768284, Error = 0.000217
Sample 81: DUT = -0.703125, Golden = -0.702749, Error = -0.000376
Sample 82: DUT = -0.508301, Golden = -0.508053, Error = -0.000247
Sample 83: DUT = -0.079834, Golden = -0.080068, Error = 0.000234
Sample 84: DUT = 0.181396, Golden = 0.181707, Error = -0.000310
Sample 85: DUT = 0.378662, Golden = 0.378439, Error = 0.000223
Sample 86: DUT = 0.619629, Golden = 0.619974, Error = -0.000345
Sample 87: DUT = 0.797607, Golden = 0.797275, Error = 0.000333
Sample 88: DUT = 0.861328, Golden = 0.861460, Error = -0.000131
Sample 89: DUT = 0.988770, Golden = 0.988975, Error = -0.000205
Sample 90: DUT = 0.952148, Golden = 0.951667, Error = 0.000482
Sample 91: DUT = 0.603760, Golden = 0.604258, Error = -0.000498
Sample 92: DUT = 0.362793, Golden = 0.362476, Error = 0.000317
Sample 93: DUT = 0.176514, Golden = 0.176600, Error = -0.000086
Sample 94: DUT = -0.141357, Golden = -0.141260, Error = -0.000098
Sample 95: DUT = -0.435547, Golden = -0.435536, Error = -0.000011
Sample 96: DUT = -0.623779, Golden = -0.623283, Error = -0.000496
Sample 97: DUT = -0.883545, Golden = -0.883648, Error = 0.000103
Sample 98: DUT = -1.083740, Golden = -1.083074, Error = -0.000667
Sample 99: DUT = -0.982178, Golden = -0.982096, Error = -0.000082
Sample 100: DUT = -0.878662, Golden = -0.878288, Error = -0.000374
Sample 101: DUT = -0.739746, Golden = -0.739291, Error = -0.000455
Sample 102: DUT = -0.409424, Golden = -0.409613, Error = 0.000190
Sample 103: DUT = -0.162842, Golden = -0.162480, Error = -0.000361
Sample 104: DUT = 0.045410, Golden = 0.045336, Error = 0.000074
Sample 105: DUT = 0.440186, Golden = 0.440555, Error = -0.000369
Sample 106: DUT = 0.888428, Golden = 0.888289, Error = 0.000139
Sample 107: DUT = 1.001221, Golden = 1.001038, Error = 0.000183
Sample 108: DUT = 0.994629, Golden = 0.994849, Error = -0.000220
Sample 109: DUT = 1.014648, Golden = 1.014215, Error = 0.000433
Sample 110: DUT = 0.816162, Golden = 0.816532, Error = -0.000370
Sample 111: DUT = 0.642090, Golden = 0.641857, Error = 0.000233
Sample 112: DUT = 0.494141, Golden = 0.494146, Error = -0.000005
Sample 113: DUT = 0.181885, Golden = 0.182057, Error = -0.000172
Sample 114: DUT = -0.176270, Golden = -0.176260, Error = -0.000010
Sample 115: DUT = -0.384521, Golden = -0.384168, Error = -0.000354
Sample 116: DUT = -0.582031, Golden = -0.582177, Error = 0.000146
Sample 117: DUT = -0.902100, Golden = -0.901318, Error = -0.000782
```

```
Sample 118: DUT = -1.013916, Golden = -1.013815, Error = -0.000101
Sample 119: DUT = -1.028564, Golden = -1.028266, Error = -0.000298
Sample 120: DUT = -1.014893, Golden = -1.014370, Error = -0.000523
Sample 121: DUT = -0.781006, Golden = -0.781084, Error = 0.000079
Sample 122: DUT = -0.484619, Golden = -0.484140, Error = -0.000479
Sample 123: DUT = -0.245850, Golden = -0.246067, Error = 0.000217
Sample 124: DUT = 0.060059, Golden = 0.060489, Error = -0.000430
Sample 125: DUT = 0.529053, Golden = 0.528981, Error = 0.000072
Sample 126: DUT = 0.770508, Golden = 0.770368, Error = 0.000139
Sample 127: DUT = 0.837891, Golden = 0.838218, Error = -0.000328
Sample 128: DUT = 1.068359, Golden = 1.067789, Error = 0.000571
Sample 129: DUT = 1.136719, Golden = 1.136960, Error = -0.000241
Sample 130: DUT = 0.941895, Golden = 0.941646, Error = 0.000249
Sample 131: DUT = 0.672119, Golden = 0.672322, Error = -0.000203
Sample 132: DUT = 0.353516, Golden = 0.353542, Error = -0.000026
Sample 133: DUT = -0.047119, Golden = -0.047150, Error = 0.000031
Sample 134: DUT = -0.254883, Golden = -0.254169, Error = -0.000714
Sample 135: DUT = -0.333496, Golden = -0.333843, Error = 0.000347
Sample 136: DUT = -0.669678, Golden = -0.668971, Error = -0.000707
Sample 137: DUT = -0.882080, Golden = -0.881906, Error = -0.000174
Sample 138: DUT = -0.900635, Golden = -0.900409, Error = -0.000226
Sample 139: DUT = -0.937012, Golden = -0.936608, Error = -0.000404
Sample 140: DUT = -0.831055, Golden = -0.831035, Error = -0.000020
Sample 141: DUT = -0.537842, Golden = -0.537185, Error = -0.000657
Sample 142: DUT = -0.277100, Golden = -0.277367, Error = 0.000268
Sample 143: DUT = -0.072754, Golden = -0.072264, Error = -0.000490
Sample 144: DUT = 0.316406, Golden = 0.316433, Error = -0.000027
Sample 145: DUT = 0.591553, Golden = 0.591418, Error = 0.000135
Sample 146: DUT = 0.683105, Golden = 0.683581, Error = -0.000476
Sample 147: DUT = 0.912598, Golden = 0.912147, Error = 0.000451
Sample 148: DUT = 1.090088, Golden = 1.090241, Error = -0.000153
Sample 149: DUT = 1.050293, Golden = 1.050152, Error = 0.000141
Sample 150: DUT = 0.948242, Golden = 0.948436, Error = -0.000194
Sample 151: DUT = 0.861328, Golden = 0.861215, Error = 0.000114
Sample 152: DUT = 0.555176, Golden = 0.555035, Error = 0.000140
Sample 153: DUT = 0.178467, Golden = 0.179129, Error = -0.000662
Sample 154: DUT = -0.046875, Golden = -0.047153, Error = 0.000278
Sample 155: DUT = -0.458984, Golden = -0.458506, Error = -0.000479
Sample 156: DUT = -0.791260, Golden = -0.791092, Error = -0.000168
Sample 157: DUT = -0.824951, Golden = -0.824857, Error = -0.000095
Sample 158: DUT = -0.860352, Golden = -0.859936, Error = -0.000416
Sample 159: DUT = -0.897461, Golden = -0.897500, Error = 0.000039
Sample 160: DUT = -0.774414, Golden = -0.773748, Error = -0.000667
Sample 161: DUT = -0.607666, Golden = -0.607853, Error = 0.000187
Sample 162: DUT = -0.552979, Golden = -0.552549, Error = -0.000430
Sample 163: DUT = -0.239990, Golden = -0.239724, Error = -0.000266
Sample 164: DUT = 0.216064, Golden = 0.215823, Error = 0.000242
Sample 165: DUT = 0.461426, Golden = 0.461929, Error = -0.000504
Sample 166: DUT = 0.668457, Golden = 0.668224, Error = 0.000233
Sample 167: DUT = 0.865479, Golden = 0.865628, Error = -0.000150
Sample 168: DUT = 0.926025, Golden = 0.925840, Error = 0.000185
```

```
Sample 169: DUT = 0.886963, Golden = 0.887213, Error = -0.000250
Sample 170: DUT = 0.889648, Golden = 0.889602, Error = 0.000046
Sample 171: DUT = 0.714111, Golden = 0.713713, Error = 0.000399
Sample 172: DUT = 0.404785, Golden = 0.405215, Error = -0.000430
Sample 173: DUT = 0.244629, Golden = 0.244289, Error = 0.000340
Sample 174: DUT = -0.016113, Golden = -0.015984, Error = -0.000129
Sample 175: DUT = -0.376221, Golden = -0.376042, Error = -0.000179
Sample 176: DUT = -0.577881, Golden = -0.577706, Error = -0.000175
Sample 177: DUT = -0.713623, Golden = -0.713275, Error = -0.000348
Sample 178: DUT = -0.942383, Golden = -0.942276, Error = -0.000107
Sample 179: DUT = -1.030762, Golden = -1.030166, Error = -0.000596
Sample 180: DUT = -0.862549, Golden = -0.862531, Error = -0.000017
Sample 181: DUT = -0.782715, Golden = -0.782438, Error = -0.000277
Sample 182: DUT = -0.585205, Golden = -0.584820, Error = -0.000385
Sample 183: DUT = -0.177246, Golden = -0.177376, Error = 0.000130
Sample 184: DUT = 0.083740, Golden = 0.084034, Error = -0.000294
Sample 185: DUT = 0.330811, Golden = 0.330665, Error = 0.000145
Sample 186: DUT = 0.687744, Golden = 0.687991, Error = -0.000246
Sample 187: DUT = 0.963623, Golden = 0.963410, Error = 0.000213
Sample 188: DUT = 0.978760, Golden = 0.978755, Error = 0.000005
Sample 189: DUT = 0.980469, Golden = 0.980642, Error = -0.000173
Sample 190: DUT = 0.918945, Golden = 0.918345, Error = 0.000600
Sample 191: DUT = 0.628662, Golden = 0.629075, Error = -0.000413
Sample 192: DUT = 0.455566, Golden = 0.455284, Error = 0.000282
Sample 193: DUT = 0.278564, Golden = 0.278648, Error = -0.000083
Sample 194: DUT = -0.077881, Golden = -0.077674, Error = -0.000207
Sample 195: DUT = -0.395264, Golden = -0.395034, Error = -0.000229
Sample 196: DUT = -0.601562, Golden = -0.601213, Error = -0.000350
Sample 197: DUT = -0.865967, Golden = -0.866057, Error = 0.000090
Sample 198: DUT = -1.034912, Golden = -1.034205, Error = -0.000707
Sample 199: DUT = -0.874512, Golden = -0.874459, Error = -0.000053
Sample 200: DUT = -0.754639, Golden = -0.754483, Error = -0.000156
Sample 201: DUT = -0.506592, Golden = -0.506326, Error = -0.000266
Sample 202: DUT = -0.300537, Golden = -0.300456, Error = -0.000081
Sample 203: DUT = -0.270264, Golden = -0.270024, Error = -0.000240
Sample 204: DUT = -0.182861, Golden = -0.182912, Error = 0.000051
Sample 205: DUT = 0.054932, Golden = 0.055137, Error = -0.000206
Sample 206: DUT = 0.174072, Golden = 0.173983, Error = 0.000089
Sample 207: DUT = 0.042725, Golden = 0.042885, Error = -0.000160
```

# Results and Conclusion

## Results

The comparison between the Device Under Test (DUT) output and the Golden output across 210 sample points shows that both outputs are almost perfectly aligned, indicating a high degree of accuracy in the DUT's performance. The outputs follow the same waveform pattern, with minimal deviation.

An error analysis, computed as the difference between DUT and Golden outputs, shows that the error values oscillate around zero with a maximum deviation of less than

$\pm 0.0008$. This small error range demonstrates the DUT's ability to accurately replicate the expected behavior.

## Conclusion

The analysis confirms that the DUT output closely matches the Golden output, with negligible error across all samples. The minimal error magnitude indicates functional correctness and high precision in the DUT's implementation. Therefore, the DUT is validated as a reliable and accurate system for the given task.