**Parallel and Pipeline Implementation of FIR and IIR Filters on FPGA using MATLAB/SIMULINK**

**Objective:**

1. Design IIR or FIR filters on MATLAB/SIMULINK platform and generate HDL code.

2. Verify the HDL code in XILINX-VIVADO.

**Tools Used:**

MATLAB 2024B (HDL coder and Fixed-Point Tool) and VIVADO 2019.1

Design follows Fixed Point data representation and Two’s complement data representation.

**Example:**

The Design



1. Data stored in 1-D LUT and fetched by an HDL counter.
2. Convery block used for converting data from one type to another. Here, as all data types are Fixed Point thus this block is not required.
3. All block should have fixed point format data type. Here, 16 bits are used and 10 bits are used fraction and 6 bits are integer. Maximum value is 31 and minimum value is -31.

Input to the Look Up Table (1D)

1. Method 1: From workspace





1. Method 2: By Manually Putting the values.





Data that is stored in ROM.



Output



Vivado Output



**Code Generated from Simulink**

`timescale 1 ns / 1 ns

module DSP\_test

(clk,

reset,

clk\_enable,

ce\_out,

Out1);

input clk;

input reset;

input clk\_enable;

output ce\_out;

output signed [15:0] Out1; // sfix16\_En10

wire enb;

wire signed [15:0] Constant1\_out1; // sfix16\_En10

wire [3:0] count\_step; // ufix4

wire [3:0] count\_from; // ufix4

reg [3:0] HDL\_Counter\_out1; // ufix4

wire [3:0] count; // ufix4

wire need\_to\_wrap;

wire [3:0] count\_value; // ufix4

wire [7:0] prelookup\_comp\_in\_u; // ufix8

wire [3:0] prelookup\_idx; // ufix4

wire signed [15:0] alpha1\_D\_Lookup\_Table\_table\_data [0:15]; // sfix16\_En10 [16]

wire signed [15:0] alpha1\_D\_Lookup\_Table\_out1; // sfix16\_En10

wire signed [31:0] Product1\_mul\_temp; // sfix32\_En20

wire signed [15:0] Product1\_out1; // sfix16\_En10

wire signed [15:0] Constant\_out1; // sfix16\_En10

wire signed [31:0] Product\_mul\_temp; // sfix32\_En20

wire signed [15:0] Product\_out1; // sfix16\_En10

reg signed [15:0] Delay\_out1; // sfix16\_En10

wire signed [15:0] Add\_add\_temp; // sfix16\_En10

wire signed [15:0] Add\_out1; // int16

wire signed [15:0] Data\_Type\_Conversion1\_out1; // sfix16\_En10

assign Constant1\_out1 = 16'sb0000001000000000;

// Count limited, Unsigned Counter

// initial value = 0

// step value = 1

// count to value = 15

assign count\_step = 4'b0001;

assign count\_from = 4'b0000;

assign enb = clk\_enable;

assign count = HDL\_Counter\_out1 + count\_step;

assign need\_to\_wrap = HDL\_Counter\_out1 == 4'b1111;

assign count\_value = (need\_to\_wrap == 1'b0 ? count :

count\_from);

always @(posedge clk)

begin : HDL\_Counter\_process

if (reset == 1'b1) begin

HDL\_Counter\_out1 <= 4'b0000;

end

else begin

if (enb) begin

HDL\_Counter\_out1 <= count\_value;

end

end

end

assign prelookup\_comp\_in\_u = {4'b0, HDL\_Counter\_out1};

assign prelookup\_idx = (prelookup\_comp\_in\_u == 8'b00000000 ? 4'b0000 :

(prelookup\_comp\_in\_u >= 8'b00001111 ? 4'b1111 :

HDL\_Counter\_out1));

assign alpha1\_D\_Lookup\_Table\_table\_data[0] = 16'sb0000000000000000;

assign alpha1\_D\_Lookup\_Table\_table\_data[1] = 16'sb0000000100000000;

assign alpha1\_D\_Lookup\_Table\_table\_data[2] = 16'sb0000001000000000;

assign alpha1\_D\_Lookup\_Table\_table\_data[3] = 16'sb0000001100000000;

assign alpha1\_D\_Lookup\_Table\_table\_data[4] = 16'sb0000010000000000;

assign alpha1\_D\_Lookup\_Table\_table\_data[5] = 16'sb0000001100000000;

assign alpha1\_D\_Lookup\_Table\_table\_data[6] = 16'sb0000001000000000;

assign alpha1\_D\_Lookup\_Table\_table\_data[7] = 16'sb0000000100000000;

assign alpha1\_D\_Lookup\_Table\_table\_data[8] = 16'sb0000000000000000;

assign alpha1\_D\_Lookup\_Table\_table\_data[9] = 16'sb1111111100000000;

assign alpha1\_D\_Lookup\_Table\_table\_data[10] = 16'sb1111111000000000;

assign alpha1\_D\_Lookup\_Table\_table\_data[11] = 16'sb1111110100000000;

assign alpha1\_D\_Lookup\_Table\_table\_data[12] = 16'sb1111110000000000;

assign alpha1\_D\_Lookup\_Table\_table\_data[13] = 16'sb1111110100000000;

assign alpha1\_D\_Lookup\_Table\_table\_data[14] = 16'sb1111111000000000;

assign alpha1\_D\_Lookup\_Table\_table\_data[15] = 16'sb1111111100000000;

assign alpha1\_D\_Lookup\_Table\_out1 = alpha1\_D\_Lookup\_Table\_table\_data[prelookup\_idx];

assign Product1\_mul\_temp = Constant1\_out1 \* alpha1\_D\_Lookup\_Table\_out1;

assign Product1\_out1 = Product1\_mul\_temp[25:10];

assign Constant\_out1 = 16'sb0000010000000000;

assign Product\_mul\_temp = Constant\_out1 \* alpha1\_D\_Lookup\_Table\_out1;

assign Product\_out1 = Product\_mul\_temp[25:10];

always @(posedge clk)

begin : Delay\_process

if (reset == 1'b1) begin

Delay\_out1 <= 16'sb0000000000000000;

end

else begin

if (enb) begin

Delay\_out1 <= Product\_out1;

end

end

end

assign Add\_add\_temp = Product1\_out1 + Delay\_out1;

assign Add\_out1 = {{10{Add\_add\_temp[15]}}, Add\_add\_temp[15:10]};

assign Data\_Type\_Conversion1\_out1 = {Add\_out1[5:0], 10'b0000000000};

assign Out1 = Data\_Type\_Conversion1\_out1;

assign ce\_out = clk\_enable;

endmodule // DSP\_test

**Test bench used to Verify it**

module dsp\_test\_tb(

);

reg clk,reset,clk\_enable;

wire ce\_out;

wire [15:0] Out1;

DSP\_test uut

(clk,

reset,

clk\_enable,

ce\_out,

Out1);

always #5 clk = ~clk;

initial begin

clk = 0; clk\_enable = 1; reset = 1;

#100;

reset = 0;

end

endmodule