Design and Implementation of Re-configurable Digital Filter on FPGA.

Anisha Sharma
School of Computing and Electrical Engineering,IIT Mandi,
Himachal Pradesh, India.
b20029@students.iitmandi.ac.in

***Abstract*—A digital filter is a system that performs mathematical operations on a discrete and sampled time signal, so as to enhance or reduce certain aspects of that particular signal as may be. necessary. It is largely used in signal processing and differs from an analog.We have to design the Digital Filter on FPGA and then implementing the same. We have given two filters FIR (finite impulse response) as well as IIR (infinite impulse response) . Both IIR and FIR have been discussed in the paper .**
Index Terms = IIR , FIR ,Digital Filter , FPGA.

I. Introduction

The type of Digital filter used is like IIR and FIR . A discrete signal x[n] that can be processed by FIR as well as IIR filters to deliver y[n] discrete output signal and transfer function of these filters is given .
FIR(finite impulse response) is a filter whose impulse response (or response to any finite length input) is of finite duration, because it settles to zero in finite time in signal processing .FIR filters requires no feedback , this means that any rounding errors are not compounded by summed iterations. The same relative error occurs in each calculation. This also makes implementation simpler and are inherently stable, since the output is a sum of a finite number of finite multiples of the input values and no output value is involved . It can easily be designed to be linear phase by making the coefficient sequence symmetric.
The Transfer Function of FIR is given:-
HF (z) = b0 + b1 × z^{-1} + b2 × z^{-2} + b3 × z^{-3}
and this can be written as
y[n]/x[n] = b0 + b1 × z^{-1} + b2 × z^{-2} + b3 × z^{-3}
After solving this ,
y[n] = = b0x[n] + b1x[n 1] + b2x[n 2] + b3x[n 3]

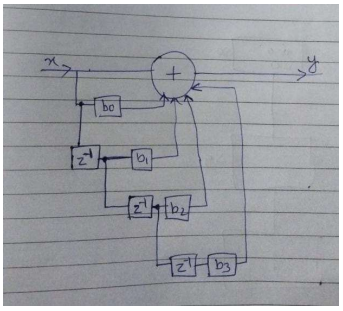

Fig. 1: FIR Filter

IIR (The infinite impulse response) filter is a recursive filter in that the output from the filter is computed by using the current and previous inputs and previous outputs. Because the filter uses previous values of the output, there is feedback of the output in the filter structure. The advantage of this approach is that the implementation lends itself to reduced memory and processing when implemented in software. However, these filters have nonlinear phase behavior and because of this characteristic, phase response must be of little concern to the DSP programmer.
The Transfer Function of FIR is given:-
HF (z) = b0 + b1 × z^{-1} + b2 × z^{-2} / (1+ a1 × z^{-1}+ a2 × z^{-2})
and this can be written as
y[n]/x[n] = b0 + b1 × z^{-1} + b2 × z^{-2} / (1+ a1 × z^{-1}+ a2 × z^{-2})
After solving this ,y[n] = b0x[n]+b1x[n1]+b2x[n2]a1y[n1]a2y[n2]

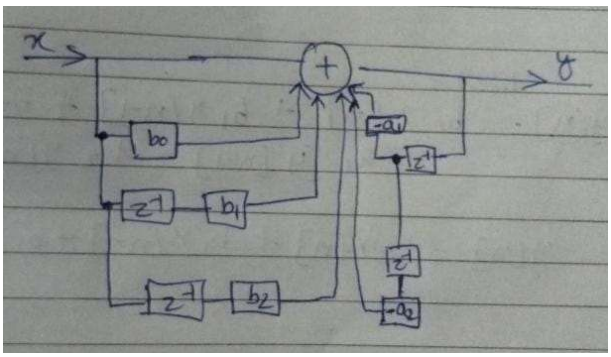

Fig. 2: IIR Filter

II. Implemented Hardware Architecture and Discription

As we have seen from the discussion that FIR filter does not have feedback (do not depend on the output value) but IIR Filter have feedback which means depend on the output value .

A. *IIR Filter:-*

The equation showing the IIR filter's output demonstrates that the current input, two prior inputs, and the two previous

outputs must be multiplied by the corresponding coefficients and output for the current time is obtained by subtracting the sum of the outputs from the sum of the inputs. While the outputs are multiplied by coefficients a1 and a2, the inputs are multiplied by coefficients b0, b1, and b2. Using the adders, each of them is added together to produce the final result.

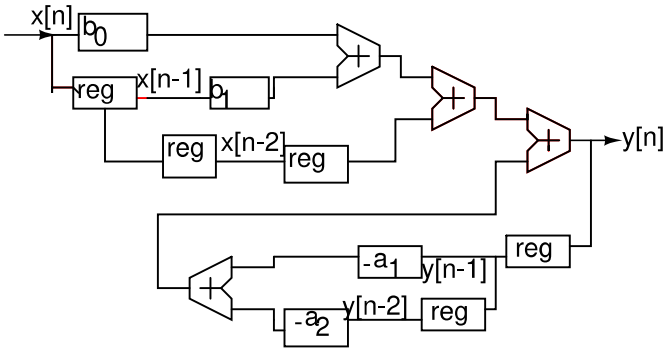

Fig. 3: Architecture of IIR Filter

B. *FIR Filter:-*

According to the equation showing the FIR filter's output, the current input and the previous three inputs are individually multiplied by their corresponding coefficients before being added to produce the output. The input is therefore multiplied by the factor b0 before being delivered separately through an edge-triggered register. From there, it travels along two paths, passing through the next register on one path and being multiplied by coefficient b1 on the other.

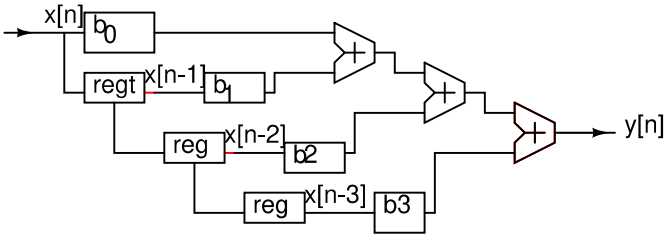

Fig. 4: Architecture of FIR Filter

C. *Reconfigurable IIR and FIR Filter:-*

The multiplexer driven by the signal iir_fir and the general architecture of the reconfigurable IIR-FIR filter are shown in fig. 5. And also we have seen that in both eqation of IIR and FIR ,b0x[n]+b1x[n1]+b2x[n2] this part is common . In FIR b3x[n-3] is added and in IIR a1y[n1]a2y[n2] is added , So we can select these two parts that which is to be added with the help of multiplexer and then add to the common part .

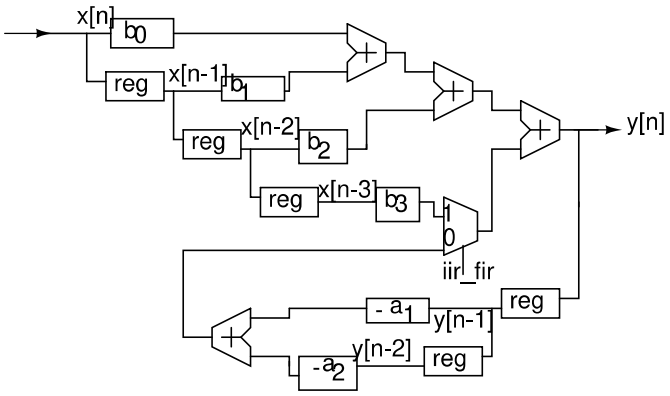

Fig. 5: Architecture of Combined IIR and FIR Filter

III. Design Flow Adopted in the Experiment:-

A. *Discription of the Process :-*

We concluded from the Architecture design that in IIR Filter and FIR filter , if we use the multiplexar we can get the Reconfigurable Filter , it depends on the select line iir_fir that which filter is selected and which operation is performed . For IIR filter , iir_fir is 0 and for FIR it is 1 .
Input Sequence:-

Clock Cycle	$x[n]$	$x[n-1]$	$x[n-2]$	$x[n-3]$
1	0	X	X	X
2	0	0	X	X
3	0	0	0	X
4	0	0	0	0
5	0	0	0	0
6	1	0	0	0
7	2	1	0	0
8	6	2	1	0
9	5	6	2	1
10	15	5	6	2

B. *Verilog Code of the Components of Filter:-*

5 modules is used with the testbench .

8x8 Multiplier

It multiplies 8 bits by 8 bits and generates a 16-bit output,

```
Prod. module mul(a,b,prod);
input [7:0] a,b;
output [15:0] prod;

  assign prod = a*b;
endmodule
```

16 x16 Adder

It produces a 16-bit sum after performing a 16-bit addition. Assign statements are used to create the carry and sum bits, and a for loop is used to conduct a carry lookahead addition.

```
  module adder16(
input [15:0] in1,
input [15:0] in2,
```

output [15:0] sum
);

wire [16:0] c;
genvar i;

assign c[0],sum[0] = in1[0]+in2[1];
for (i=1;i¡16;i = i+1) begin
assign c[i],sum[i] = in1[i]+in2[i]+c[i-1];
end
endmodule

2x1Multiplexer

It generates a 16-bit output from two 16-bit inputs and a 1-bit choose signal. The output is chosen depending on the select signal using an assign statement. module mux_2by1(
output [15:0] in1,
output [15:0] in2,
input sel,
output [15:0] out);

assign out = sel?in1:in2;
endmodule

Buffer/Register

This module records the current input, the previous three inputs, and the output, together with the previous two outputs, in three 16-bit registers and four 8-bit registers. It is started by the clock signal's negative edge. module buff_7x8(
input clk,
input [7:0] pres_x,
input [15:0] pres_y,
output reg [7:0] x_1,
output reg [7:0] x_2,
output reg [7:0] x_3,
output [15:0] y_1,
output [15:0] y_2);
reg [7:0] buff0 [1:3];
reg [15:0] buff1 [1:2];
assign y_1 = buff1[1] ;
assign y_2 = buff1[2] ;

always @(negedge clk) begin
buff0[1] <= pres_x;
buff1[1] <= pres_y;
buff0[3] <= buff0[2];
buff0[2] <= buff0[1];
buff1[2] <= buff1[1];
x_1 <= buff0 [1];
x_2 <= buff0 [2];
x_3 <= buff0[3];
end
endmodule

Top Module of the Filter

This module assembles the whole top-level circuit by connecting the multiplexer and the necessary multipliers and adders. module top(input [7:0] x_n,
input clk, iir_fir,
output [15:0] y_n);

wire [7:0] x_n_1, x_n_2, x_n_3;
wire [15:0] y_n_1, y_n_2;
wire [15:0] p0, p1, p2, s2_1, p4, p5, s0, s1, s2, s2_2;

buff_7x8 buff(clk, x_n, y_n, x_n_1, x_n_2, x_n_3, y_n_1, y_n_2);
mul m0(x_n, 8'b00000001, p0);
mul m1(x_n_1, 8'b00000010, p1);
mul m2(x_n_2, 8'b00000011, p2);
mul m3(x_n_3, 8'b00000100, s2_1);
mul m4(y_n_1, -16'd5, p4);
mul m5(y_n_2, -16'd6, p5);
adder16 a0(p0,p1,s0);
adder16 a1(s0,p2,s1);
adder16 a2(p4,p5,s2_2);
mux_2by1 mux0(s2_1,s2_2,iir_fir,s2);
adder16 a3(s1,s2,y_n);
endmodule

Testbench of Top Module of the Filter

This testbench code is for IIR filter , after 5 clock cycles when it take the input value , iir_fir signal becomes zero . module top_tb();
reg clk,iir_fir;
reg [7:0] in_n;
wire [15:0] out_n;
top uut(in_n,clk,iir_fir,out_n);
always #20 clk <= c̃lk;
initial begin
clk <= 0;
iir_fir <= 0;
#40;
in_n <= 8'b00000000;
iir_fir <= 1;
#200;
in_n ¡<= 8'b00000001;
iir_fir <= 0;
#40;
in_n <= 8'b00000010;
iir_fir <= 0;
#40; in_n <= 8'b00000110;
iir_fir <= 0;
#40;
in_n <= 8'b00001111;
iir_fir <= 0;
#40;
in_n <= 8'b00000101;
iir_fir <= 0;
#40;
end
endmodule

IV. Experimental Results and Discussions

Output Sequence :-

Cycle	$y[n]$ (FIR)
1	X
2	X
3	X
4	0
5	0
6	1
7	5
8	13
9	38

A. *Simulations of the Filter:-*

IIR Filter

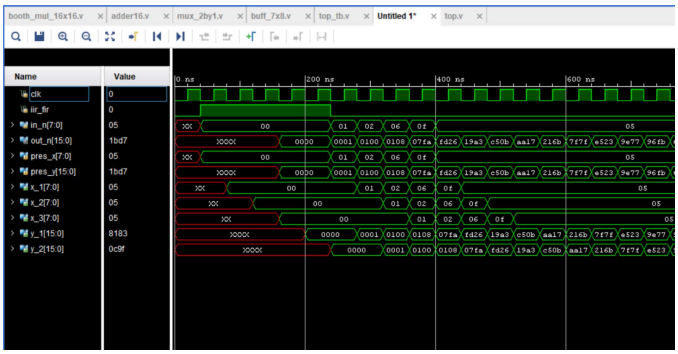

Fig. 6: Simulations of the IIR Filter

FIR Filter

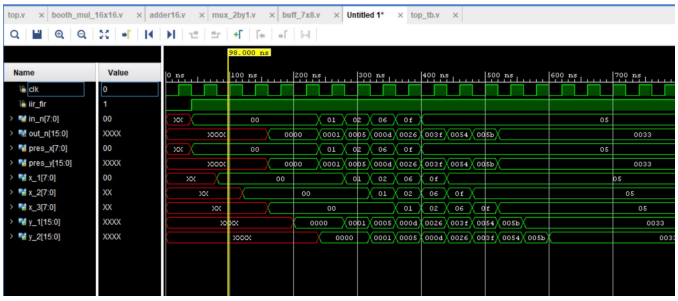

Fig. 7: Simulations of the FIR Filter

B. *Timing Analysis:-*

Setup Time

Total Delay for Setup is 8.975 .

Logic Delay = 4.394

Net Delay = 4.581

Hold Time

Total Delay for Hold is 0.287 .

Logic Delay = 0.146

Net Delay = 0.141

C. *RTL Schemetic:-*

We can see the schematic of the design similar to our architecture .

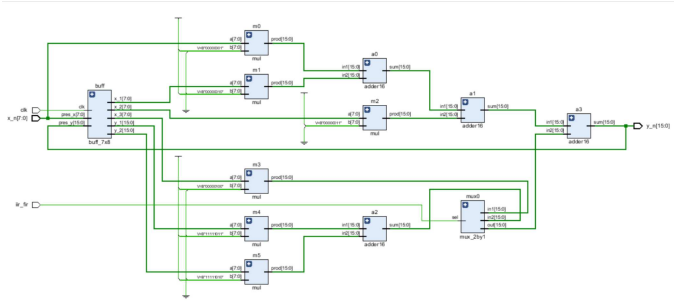

Fig. 8: Schematic of Reconfigurable Filter

D. *Synthesized Design:-*

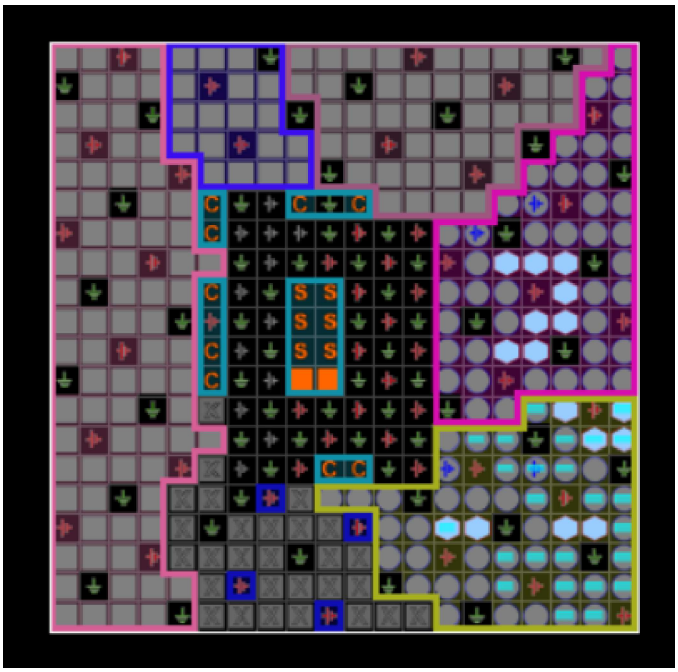

Fig. 9: Synthesized Design

E. *Power Report:-*

Power Report of Synthsis

Total On-chip power utilized is 10.79W.

Junction temperature was 125∘C.

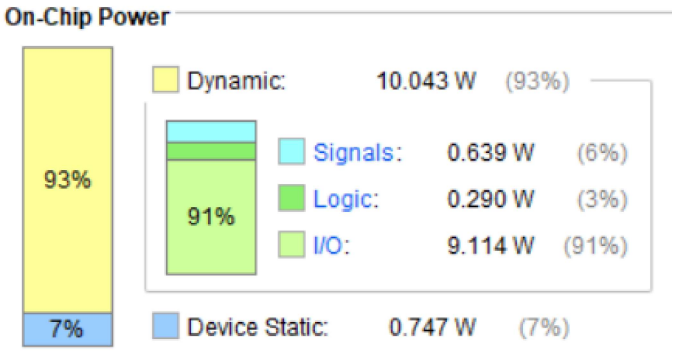

Fig. 10: Power Utilized in the Synthesis

Power Report of Implementation

Total On-chip power utilized after implemetation is 10.6 W. Junction temperature was 125◦C.

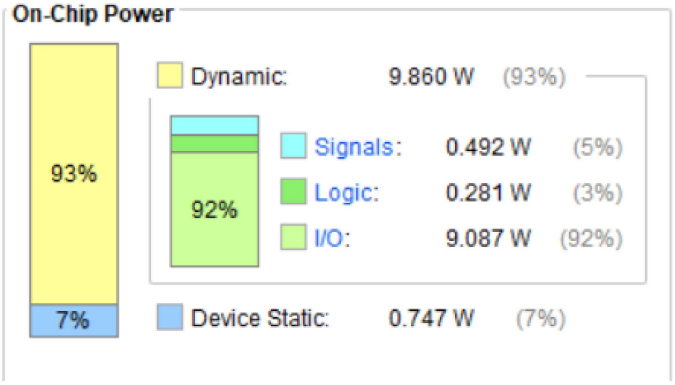

Fig. 11: Power Utilized in the Implementation

F. Resource Utilizaion:-

Resource Utilizaion of Synthsis

Resource	Utilization	Available	Utilization %
LUT	31	17600	0.18
FF	24	35200	0.07
IO	25	100	25.00

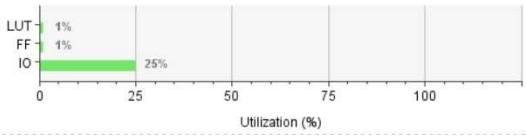

Fig. 12: Resource Utilized in the Synthesis

Resource Utilizaion of Implementation

Resource	Utilization	Available	Utilization %
LUT	31	17600	0.18
FF	24	35200	0.07
IO	25	100	25.00

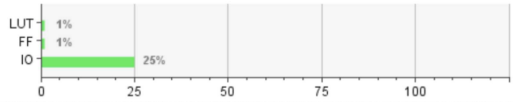

Fig. 13: Resource Utilized in the Implementation

V. Conclusions

1. By writing simple codes of multiplexer and register and adders , Multipler, we can write verilog code of FIR and IIR Filter by instanciating them .
2 . Schemetic Diagram of the Filter is aslo easily provided .
3 . Simulations of the Filter is also given by writing its testbench code and as we can see by simulations it is giving correct result.

VI. Refernces

1. Embedded System Design book by Frank Vahid and Tony Givargis