IIR filter design in FPGA -Simple example

Marko Katic
University of Zagreb
24 PUBLICATIONS **39** CITATIONS
SEE PROFILE

Radovan Stojanovic
University of Montenegro
154 PUBLICATIONS **763** CITATIONS
SEE PROFILE

University of Montenegro
Faculty of Electrical Engineering

IIR filter design in FPGA – Simple example

Milojka Ratković, Marko Katić, Časlav Labudović, Mihailo Kopitović and Radovan Stojanović

Date and place: January 2020, Podgorica

Contents

Summary

This is a final exam project, made in ADEG electronics lab at the University of Montenegro, with mentoring of prof. dr Radovan Stojanovic. In further text, the problem is explained in details and offered an elegant solution. In addition to that, we have enclosed high level design and description of our solution alongside with software structure.

Abstract

The goal of this assignment is to demonstrate the work method of a simple first order IIR (Infinite Impulse Response) filter by showing the results in a Vector Waveform simulation. Infinite Impulse Response (IIR) filter is a digital filter.Time delay gives the information of speed. IIR filter which has minimum time delay gives better response than others. Area utilization and time delay are two important factors to design any filter. The important advantage of IIR filter on FIR filter is itsimplementation efficiency. IIR filters require less number oforders as comparison to meet same specification. FPGA provides more logic flexibility and the power consumption is low. FPGA is a semiconductor device containing programmable logic components and programmable interconnects. The task has been presented in the further text.

Problem description

Our task is to show a design of a first order IIR filter and present the results of the simulations. If we compare the percentage of area utilization with different order IIR filter, it can be proven that the IIR filter with minimum filter order has minimum area utilization. The low order of IIR filter has minimum time delay and better time response. Alsothis type of IIR filter has better speed as well as minimum area utilization. For minimum area consumption and better time response the order of IIR filter should be as minimum as possible.

Hardware/software solution and simulation

As a software solution we used Altera Quartus 9.1 which is approved and tested for all FPGA and CPDL based systems. Source code was written in VHDL and the simulations were made and presented in Vector WaveForm file.

Theoretical background

IIR filters have an infinite impulse response because the recursive part is included. They do not have linear phase characteristics because the impulse response cannot be symmetrical. IIR filter is characterized by the following equation:

$$y(n) = \sum_{k=0}^{\infty} a_k * x(n - k) - \sum_{k=1}^{\infty} b_k * y(n - k) \qquad (1)$$

Where a_k and $b_k$are the coefficients of the filter and *x(n)* and *y(n)* are the input and output to the filter. The output sample *y(n)* is a function past outputs *y(n-k)*, as well as present and past input samples *x(n)* and *x(n-k)* that is the IIR filter is a feedback system.

Problems with the operation of this filter can be manifested through stability or variable group delay. The condition of stability is that all the poles are inside a unit circle.

Bilinear transformation allows one analog filter (usually Butterworth) to be mapped to a digital IIR filter.

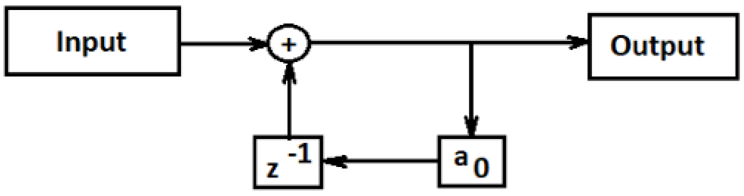


Fig. 1.0. First order IIR filter (low pass)

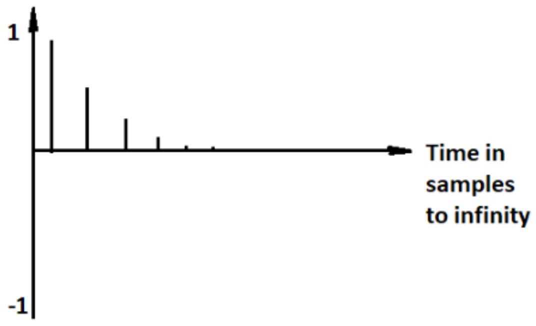


Fig. 1.1 – Impulse response

As above explained, the output in this moment depends of the output and input signals in previous moments.
Coefficients multiplying the input or output signals are determined by vector of values which we put in.

IIR filter VHDL code

```
library IEEE;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity iir is
port(
  clk           : in  std_logic;
  in_data    : in  std_logic_vector(15 downto 0);
  out_data   : buffer std_logic_vector(15 downto 0));
end iir;

architecture arch_iir of iir is
signal temp : unsigned(15 downto 0) := (others => '0');
begin

process (clk)
begin
  if rising_edge(clk) then
      out_data <= std_logic_vector(unsigned(in_data) + temp);
  end if;
  if falling_edge(clk) then
      temp <= unsigned(out_data)/2;
  end if;
end process;

end arch_iir;
```

In our particular case we have output signal: $y(n) = k_1*y(n-1) + p_1*x(n)$.

This filter is stable as long as the gain $k_1 < 1$. This is known as a stability condition. Every IIR filter, no matter how simple or complicated the arrangement, will have stability conditions. We must remember to define and set these limits and exceptions in code.

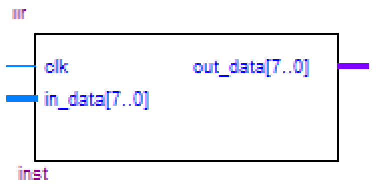
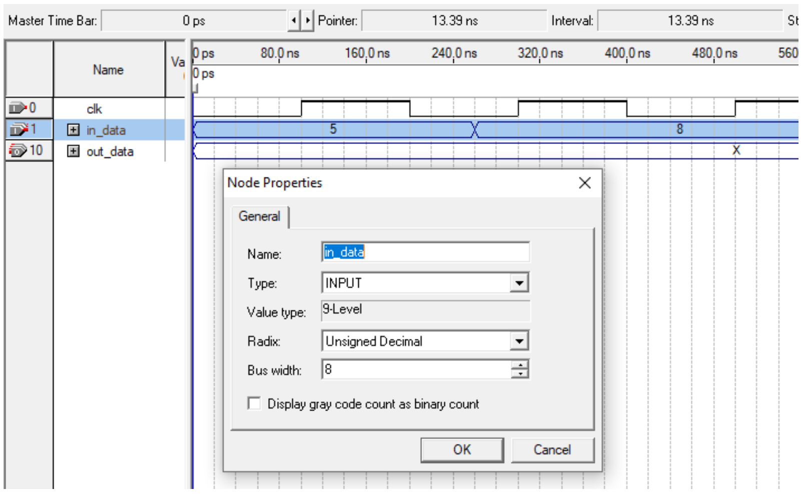


Fig. 2.0 Block scheme of IIR filter

Simulation process and results

To determine the input values in vector waveform file, we must select the *in_data signal/Right Button Mouse/Properties/Radix: Unsigned Decimals.*



Master Time Bar: 0 ps Pointer: 13.39 ns Interval: 13.39 ns

	Name	Value
0	clk	
1	in_data	5, 8
10	out_data	X

Node Properties

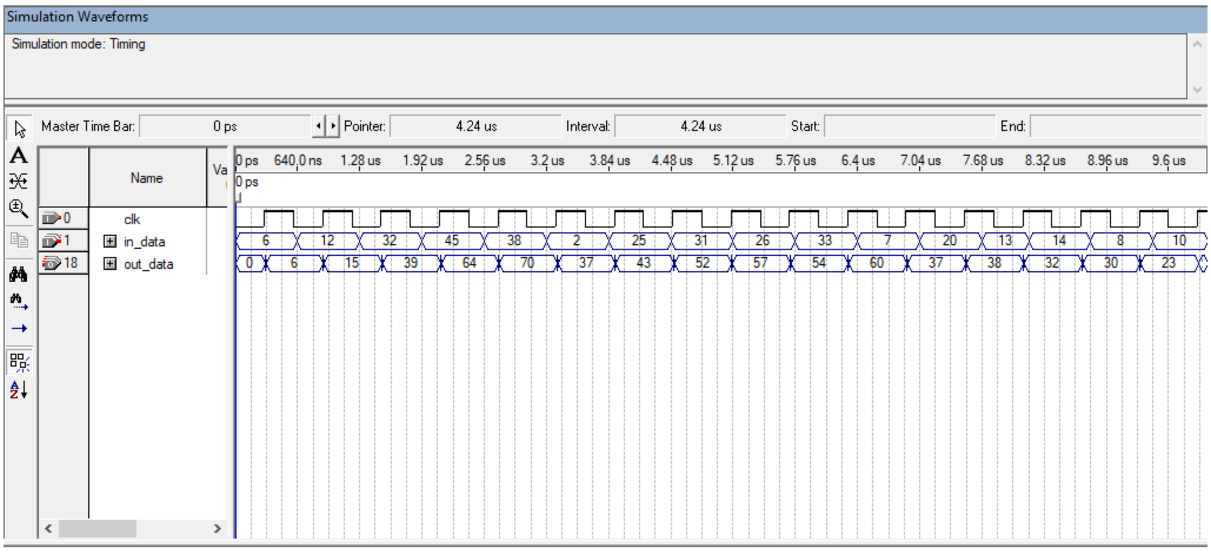
General

Name: in_data
Type: INPUT
Value type: 9-Level
Radix: Unsigned Decimal
Bus width: 8
☐ Display gray code count as binary count

OK Cancel

Fig. 3.0 – Node Properties

The values are taken only on rising edges.



Simulation Waveforms

Simulation mode: Timing

Master Time Bar: 0 ps Pointer: 4.24 us Interval: 4.24 us Start: End:

	Name	Values
0	clk	
1	in_data	6, 12, 32, 45, 38, 2, 25, 31, 26, 33, 7, 20, 13, 14, 8, 10
18	out_data	0, 6, 15, 39, 64, 70, 37, 43, 52, 57, 54, 60, 37, 38, 32, 30, 23

Fig. 4.0 – Results of the simulation in Vector Waveform file

As a proof that the presented simulation results are valid, we implemented a code of 16 input samples for the same first order IIR filter in Matlab to compare the results.

Samples	n=1	n=2	n=3	n=4	n=5	n=6	n=7	n=8	n=9	n=10	n=11	n=12	n=13	n=14	n=15	n=16
In_data	6	12	32	45	38	2	25	31	26	33	7	20	13	14	8	10

Table 1.0. – Input samples

Matlab solution and comparing

The code written in VHDL is now transferred in Matlab.

```
in_data=[6, 12, 32, 45, 38, 2, 25, 31, 26, 33, 7, 20, 13, 14, 8, 10];
out_data=[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
temp=0.0;
for i=1:16
    out_data(i)=in_data(i)+0.5*temp;
    temp=out_data(i);
end
```

The input signal in_data is plotted:

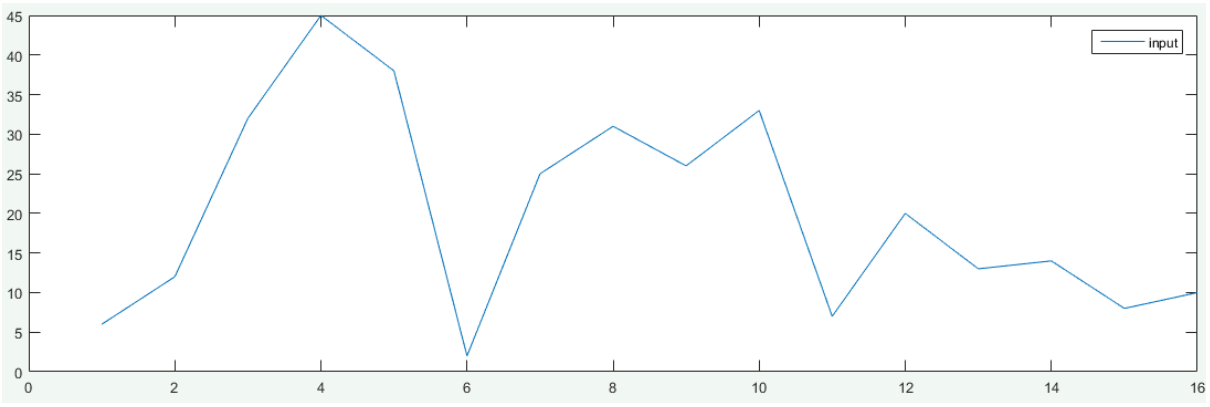


Fig. 5.0. – Input signal (in_data)

Now we can see the output difference between the two software solutions:

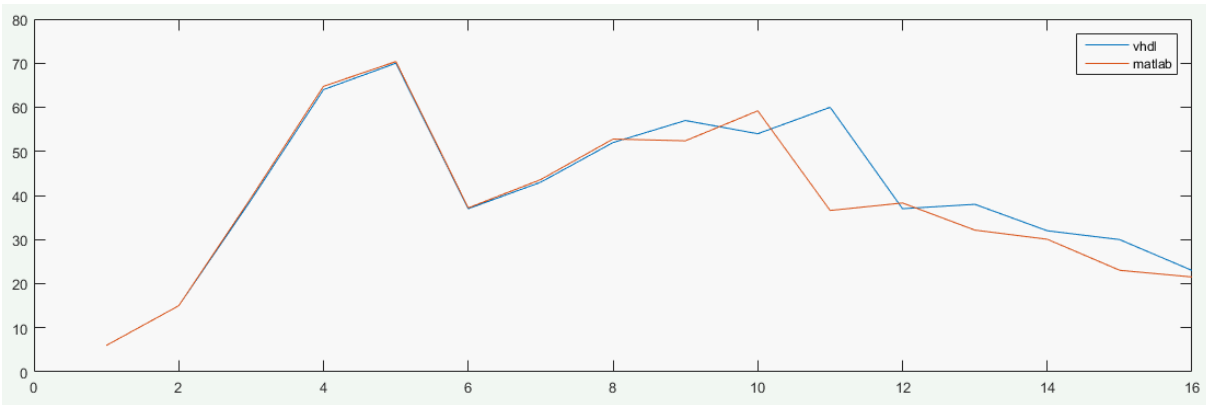


Fig. 5.1. – Output signals of VHDL solution(blue) and Matlab solution(red)

As one can observe Matlab output signal faithfully follows the VHDL output signal to a certain point where there is a deviation from the value of the VHDL output signal. In table 2.0 we can see the exact values of the outputs:

Samples	VHDL out_data	Matlab out_data
n=1	6,000	6,000
n=2	15,000	15,000
n=3	39,000	39,500
n=4	64,000	64,750
n=5	70,000	70,375
n=6	37,000	37,187
n=7	43,000	43,594
n=8	52,000	52,797
n=9	57,000	52,398
n=10	54,000	59,199
n=11	60,000	36,599
n=12	37,000	38,199
n=13	38,000	32,149
n=14	32,000	30,075
n=15	30,000	23,037
n=16	23,000	21,519

Table 2.0. – Comparrison of output results

As can be seen, matlab has finer results while in VHDL they are rounded to the full value.

The percentage error is calculated using Matlab commands:

```
error=100*(vhdl_out-matlab_out)./matlab_out;

average_error=sum(error)/16
```

The difference between VHDL and Matlab solution for our example is 7.2445%.

Verification on board

Verification was successfully done in a way described in previous paragraph. As proof we add to this statement a video, which was made in ADEG electronics lab.

Link for the video

https://youtu.be/Ym2UcUgJNSo

Literature

1. Radovan D. Stojanovic AUTOMATIZOVANO PROJEKTOVANJE DIGITALNIH SISTEMA (VHDL i FPGA)
2. DE2-70 User manual version 1.08
3. Anurag Yadav, Rajesh Mehra, "*FPGA Based IIR Filter Design Analysis for Different Orders*", Journal of Basic and Applied Engineering Research, Print ISSN: 2350-0077; Online ISSN: 2350-0255; Volume 1, Number 12; October-December 2014 pp. 106-109
4. https://theaudioprogrammer.com/digital-filter-design-fir-vs-iir-filters/