Design and Implementation of High Speed IIR and FIR Filter using Pipelining

Ravinder Kaur, Ashish Raman, *Member, IACSIT*, Hardev Singh and Jagjit Malhotra

Abstract—**The FIR & IIR Filters are being designed using HDL languages since speed is among the chief interest in this era; the main objective is to enhance the speed of the system. In the whole system if the speed of the individual block is enhanced the overall speed of the system is enhanced digital computer arithmetic is an aspect of logic design with the objective of developing appropriate algorithms in order to attain an effective utilization of the available hardware. Since ultimately, speed, power and chip area are the most often used measures of the efficiency of an algorithm, there has a strong link between the algorithms and technology applied for its implementation. Here it is done by applying the technique pipelining. The comparative analysis of pipelined & non-pipelined FIR and IIR filters is performed by using different FPGA's. The results reveal that the implemented filters turn in a consistent quality of output.**

Index Terms— **Infinite impulse response (IIR), Finite impulse response (FIR), Pipelining, Field programmable gate arrays.**

I. Introduction

High-performance digital filters are all important to the execution of digital signal processing systems. The speed of a filter realization counts not alone on the potentialities of the hardware platform employed, but as well on the computational structure of the code. [1] In pipeline processing, any operation on a long critical path is broken into levels of smaller, quicker operations, with registers between levels, so as to get a smaller critical path delay. The result is a higher operating frequency and a higher throughput. In a feedback system, viz an IIR filter, the registers introduced in a feedback loop will alter the loop delay, leading in a modified transfer function. Hence, in order to pipeline an IIR filter while conserving its original transfer function, the computations must first be redeveloped into what is called a look-ahead filter form [2]. FIR filter is the key functional block in the field of digital signal processing. A count of implementations can be ascertained in the public literatures, either by software or hardware solutions. [3] The proposed design in this paper is an attempt to optimize the system speed with minimal cost of hardware and software. The central design concept is to build filters with minimal delay, without sacrificing the performance of original filters. FIR filters feature the advantage of linear phase, stability, fewer finite precision errors, and efficient implementation. In contrast, they have a major disfavor of high order need (many coefficients) than IIR counterpart with comparable performance. The high order demand enforces additional hardware demands, arithmetic operations, area usage, and power consumption as designing and fabricating the filter. Consequently, minimizing or reducing these parameters, is a major aim in digital filter design task. [4] This paper discusses the design and implementation of a non pipelined and pipelined IIR and FIR filter to accelerate processing while conserving the dynamics of the filters.

Manuscript received July 14, 2010

Ravinder Kaur is with DAV Institute of Engineering & Technology, Jalandhar, India.
Ashish Raman is Assistant Prof with Dr. B.R. Ambedkar National Institute of Technology, Jalandhar , India
Hardev Singh is with Dr. B.R. Ambedkar National Institute of Technology, Jalandhar , India
Jagjit Malhotra is Assistant Prof with DAV Institute of Engineering & Technology, Jalandhar, India.

II. Finite Impulse Response Filter

The difference equation for FIR which defines the relation of the input signal to the output signal is given as

$$y[n] = b_0 x[n] + b_1 x[n-1] + \dots\dots + b_N x[n-N] \qquad \text{(i)}$$

where $x[n]$ is the input signal, $y[n]$ is the output signal and b_i are the filter coefficients. N is known as the filter order; an Nth-order filter has (N + 1) terms on the right-hand side; these are commonly referred to as taps. The equation (i) can be given as a convolution of filter coefficients and the input signal.

$$y[n] = \sum_{i=0}^{N} b_i x[n-i] \qquad \text{(ii)}$$

III. Infinite Impulse Response Filter

The difference equation for IIR that defines how the output signal is related to the input signal is given as

$$y[n] = \frac{1}{a_0}(b_0x[n] + b_1x[n-1] + \cdots + b_px[n-P] - a_1y[n-1] - a_2y[n-2] - \cdots - q_Qy[n-Q]) \qquad \text{(iii)}$$

where P is the feedforward filter order, b_i are the feedforward filter coefficients is the feedback filter order,a_i are the feedback filter coefficients[n] is the input signal ,y[n] is the output signal.

An IIR filter is a recursive filter where the current output depends on previous outputs [5].The condensed form of the difference equation (iii) is

$$y[n] = \frac{1}{a_0}\left(\sum_{i=0}^{P} b_i x[n-i] - \sum_{j=1}^{Q} a_j y[n-j]\right) \qquad \text{(iv)}$$

IV. Pipelining

Pipelining is an implementation technique in which multiple instructions are overlapped in execution. Today, Pipelining is a key to making processors fast. The total execution time for each individual instruction is not altered by pipelining. Pipelining does not accelerate instruction execution time, but it does accelerate program execution time by increasing the number of instructions finished per unit time.

V. Pipelining of Multiplications

In a filter the multiplication of a signal by a constant filter coefficient is the most time-consuming operation. By revising the conditions of shifts and additions the pipelining of the multiplications is achieved. The constant is constituted in canonical signed digit (CSD) format as to minimize the amount of shift-and-add operations for a constant multiplication [6].Like an binary format the CSD is represented with the difference that each digit might have an value of 0,1, or -1 (represented here as 1). Representing a constant filter coefficient in CSD significantly scales down the number of shift-and-add operations demanded to perform the multiplication by that coefficient. For example, the binary value as "11111010" stood for the decimal value 250 and in CSD as "100001010". Since the binary representation has six non- zero digits, multiplication by 250 demands the addition of six terms when a binary representation is applied:

$$u \times 250 = u \times 2^1 + u \times 2^3 + u \times 2^4 + u \times 2^5 + u \times 2^6 + u \times 2^7 \quad \text{(v)}$$

The CSD representation, with only three non-zero digits, calls for the addition (or subtraction) of only three terms:

$$u \times 250 = u \times 2^1 - u \times 2^3 + u \times 2^8 \quad \text{(vi)}$$

In a binary tree of ripple carry adders the additions themselves are coordinated. As a whole, if A denotes the number of non-zero digits used to constitute the constant, the number of levels in the binary adder tree is given by M=[log $_2$A]. For our example, the CSD format demands only two levels where as the binary representation leads in a multiplier with three levels. The multiplier coefficient represented by using a value of A has a two-fold effect on the filter implementation. 1st, it checks the number of adders required for the multiplier itself. 2nd it checks the number of levels M in the consequent tree of adders, which can cause an effect on the structure of the filter. Since lower values of M, a given system throughput perhaps attained with less pipelining. This successively means that fewer registers are demanded in the reformulated system, and a lower-order filter, may be implemented. Hence applying the CSD representation to minimize A not just brings down the number of adders in each multiplier, but as well reduces the amount of multipliers required to implement the filter.

VI. Look-Ahead Filter Forms

The reformulation of the filter in a look-ahead filter form demands the pipelining of the feedback loop in an IIR filter. Here it is exemplify by the process upon a second-order digital filter constituted by the transfer function

$$G(z) = \frac{az^2 + bz + c}{z^2 + dz + e} \quad (vii)$$

and represents the difference equation

$$y(k) = au(k) + bu(k-1) + cu(k-2) - dy(k-1) - ey(k-2) \quad (viii)$$

In an look-ahead form of the difference equation (viii), the term $y(k-1)$ is rewritten in terms by older values of y, such $y(k-2)$ and $y(k-3)$ which are variable earliest. As, for example, $y(k-2)$ is available one clock cycle earlier than $y(k-1)$, the reformulation allows for the insertion of one another level of pipelining in the feedback computation. Not every look-ahead forms preserve the stability of the original filter [7]; one that make so is the Scattered Look-Ahead (SLA) form [8]. Beginning from the transfer function of the original filter in (vii), the transfer function of the SLA filter comprises

$$G_s(z) = \frac{az^2 + bz + c}{z^2 + dz + e} \cdot \frac{z^2 - dz + e}{z^2 - dz + e} = \frac{\hat{a}z^4 + \hat{b}z^3 + \hat{c}z^2 + \hat{f}z + \hat{g}}{z^4 + \hat{d}z^2 + \hat{e}} \quad (ix)$$

where

$\hat{a} = a, \hat{b} = b - ad, \hat{c} = ae + c - db, \hat{d} = 2e - d^2,$
$\hat{e} = e^2, \hat{f} = be - cd, and\ \hat{g} = ce.$

Two newer poles have been acquainted; these poles feature the same magnitudes as the original poles, and alter by them alone in their angles. This implies that the SLA form is stable when the original system was stable (all poles inside the unit circle). The transfer function in (ix) equates to the difference equation

$$y(k) = \hat{a}u(k) + \hat{b}u(k-1) + \hat{c}u(k-2) + \hat{f}u(k-3) + \hat{g}u(k-4) - \hat{d}y(k-2) - \hat{e}y(k-4) \quad \text{(x)}$$

This is to be noted that in the difference equation (x), $y(k-1)$ has been eliminated, and $y(k)$ is calculated from $y(k-2)$ and $y(k-4)$. It is likewise possible to eliminate $y(k-2)$, and so forth, whenever more levels of pipelining are demanded.

VII. Results & Conclusion

The design and implementation of non pipelined and pipelined IIR and FIR filters was carried out. Simulation and synthesis for FPGAs has been accomplished on Spartan 3 series FPGA, target device (XC3S500E) (Speed Grade -4) and Virtex 2P series FPGA, target device (XC2VB50) (Speed Grade -6) from Xilinx. Simulation results obtained for IIR and FIR Filters have been successfully implemented on FPGA. The synthesis report results are tabulated in Table 1.The simulation results obtained on synthesis device, FPGA SPARTAN 3E shows that by using pipelined technique the delay for an IIR Filter reduced to 4.534ns from 4.903ns as obtained for non pipelined technique. Significant decline in delay from 15.458ns to 8.631ns was observed in case of FIR filters with the implementation of pipelined technique. The advantage of pipelining was also verified by using VIRTEX 2P. Similar results were observed wherein the delay for IIR and FIR filters reduced by 0.575ns and 5.654ns with the implementation of pipelining technique.

The above results demonstrate that the pipelined technique reduces delay and enhances speed as compared to non pipelined technique. However, the impact of pipelining is more significant on FIR filters as compared to IIR filters.

TABLE 1: SHOWS THE SIMULATION RESULTS OF THE DIGITAL FILTERS ON DIFFERENT FPGA'S

SYNTHESIS DEVICE	FILTER	NON PIPELINED		PIPELINED	
		Delay	Frequency	Delay	Frequency
FPGA (SPARTAN 3E)	IIR	4.903ns	203.957MHz	4.534ns	220.556MHz
	FIR	15.458ns	64.691MHz	8.631ns	115.861MHz
FPGA (VIRTEX 2P)	IIR	3.665ns	272.851MHz	3.09ns	322.997MHz
	FIR	12.29ns	81.327MHz	6.636ns	150.693MHz

The graphs below fig 1(a) & fig 1(b) shows the delays(ns) for IIR and FIR Filters on Spartan 3E and Virtex 2P FPGA's which clearly shows that the delay in the pipelined filter is far less than non pipelined filter. It can be clearly seen that results for Virtex 2P for pipelining is more significant.

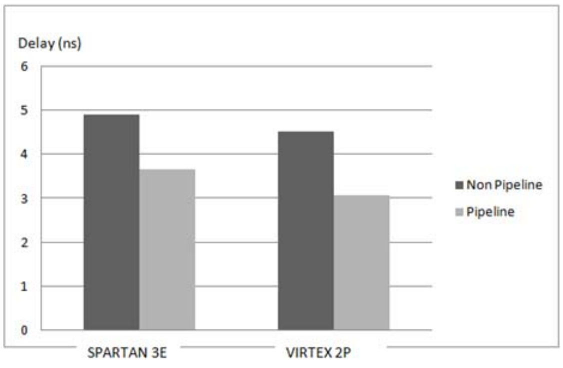

Fig 1(a) Graph for IIR Filter showing delay

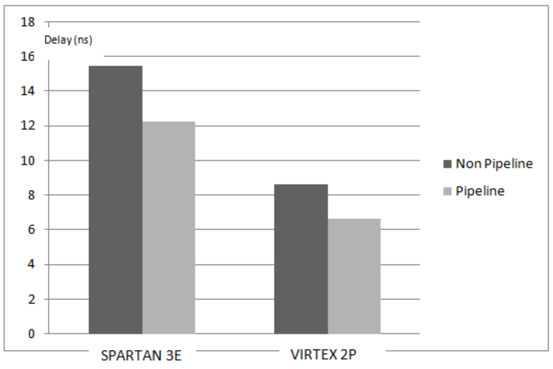

Fig 1(b) Graph for FIR Filter showing delay

The screen shots showing the synthesis report for IIR & FIR filters (pipelined) during the synthesis taken are shown below from fig 2(a) to fig 2(d)

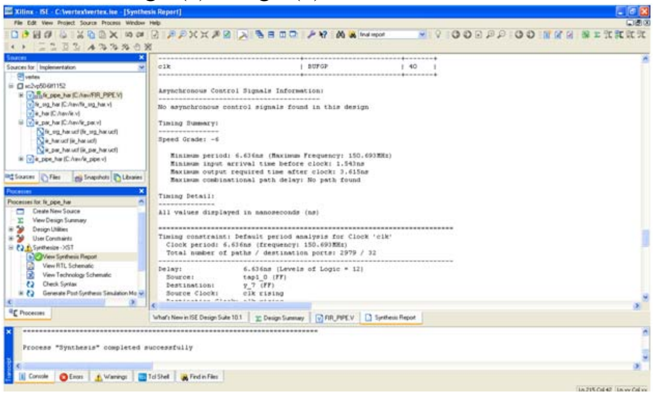
```
clk                              | BUFGP                  | 40    |
---------------------------------+------------------------+-------+

Asynchronous Control Signals Information:
----------------------------------------
No asynchronous control signals found in this design

Timing Summary:
---------------
Speed Grade: -6

   Minimum period: 6.636ns (Maximum Frequency: 150.693MHz)
   Minimum input arrival time before clock: 1.543ns
   Maximum output required time after clock: 3.615ns
   Maximum combinational path delay: No path found

Timing Detail:
--------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default period analysis for Clock 'clk'
  Clock period: 6.636ns (frequency: 150.693MHz)
  Total number of paths / destination ports: 2979 / 32
-------------------------------------------------------------------------
Delay:               6.636ns (Levels of Logic = 12)
  Source:            tap1_0 (FF)
  Destination:       y_7 (FF)
  Source Clock:      clk rising
```
```
Process "Synthesis" completed successfully
```

Fig 2(a) synthesis report for FIR pipelined implementation on Virtex 2p

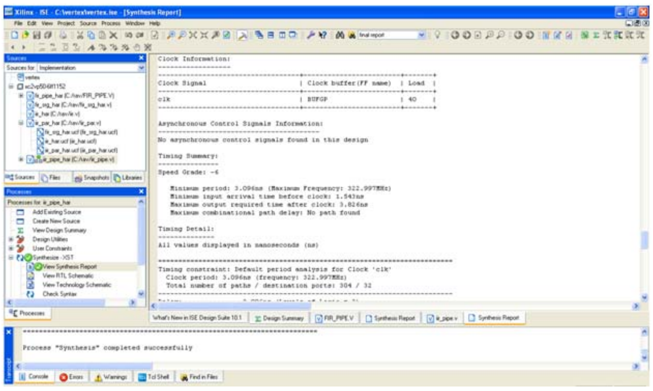
```
Clock Information:
------------------
---------------------------------+------------------------+-------+
Clock Signal                     | Clock buffer(FF name)  | Load  |
---------------------------------+------------------------+-------+
clk                              | BUFGP                  | 40    |
---------------------------------+------------------------+-------+

Asynchronous Control Signals Information:
----------------------------------------
No asynchronous control signals found in this design

Timing Summary:
---------------
Speed Grade: -6

   Minimum period: 3.096ns (Maximum Frequency: 322.997MHz)
   Minimum input arrival time before clock: 1.543ns
   Maximum output required time after clock: 3.826ns
   Maximum combinational path delay: No path found

Timing Detail:
--------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default period analysis for Clock 'clk'
  Clock period: 3.096ns (frequency: 322.997MHz)
  Total number of paths / destination ports: 304 / 32
-------------------------------------------------------------------------
```
```
Process "Synthesis" completed successfully
```

Fig 2(b) synthesis report for IIR pipelined implementation on Virtex 2p

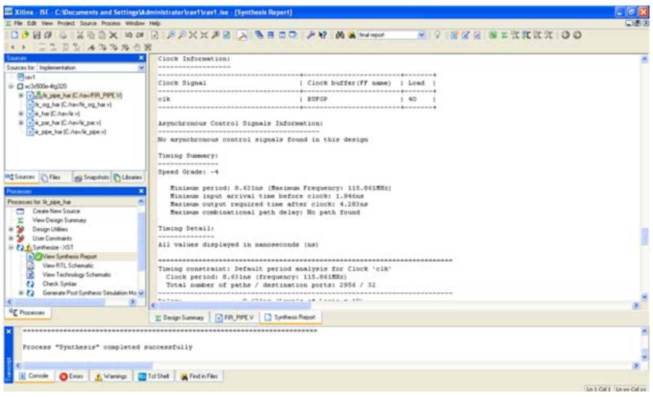
```
Clock Information:
------------------
---------------------------------+------------------------+-------+
Clock Signal                     | Clock buffer(FF name)  | Load  |
---------------------------------+------------------------+-------+
clk                              | BUFGP                  | 40    |
---------------------------------+------------------------+-------+

Asynchronous Control Signals Information:
----------------------------------------
No asynchronous control signals found in this design

Timing Summary:
---------------
Speed Grade: -4

   Minimum period: 8.631ns (Maximum Frequency: 115.861MHz)
   Minimum input arrival time before clock: 1.946ns
   Maximum output required time after clock: 4.283ns
   Maximum combinational path delay: No path found

Timing Detail:
--------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default period analysis for Clock 'clk'
  Clock period: 8.631ns (frequency: 115.861MHz)
  Total number of paths / destination ports: 2956 / 32
-------------------------------------------------------------------------
```
```
Process "Synthesis" completed successfully
```

Fig 2(c) synthesis report for FIR pipelined implementation on Spartan 3E

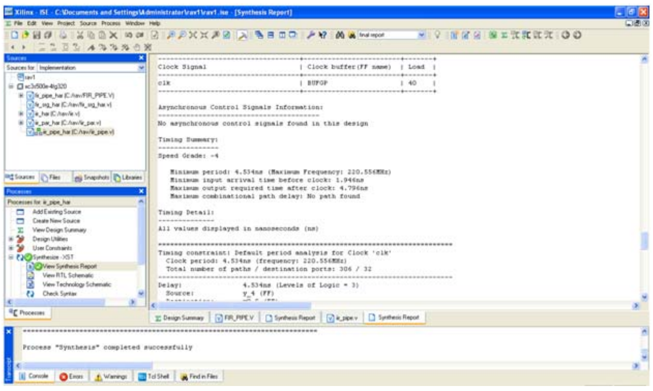
```
---------------------------------+------------------------+-------+
Clock Signal                     | Clock buffer(FF name)  | Load  |
---------------------------------+------------------------+-------+
clk                              | BUFGP                  | 40    |
---------------------------------+------------------------+-------+

Asynchronous Control Signals Information:
----------------------------------------
No asynchronous control signals found in this design

Timing Summary:
---------------
Speed Grade: -4

   Minimum period: 4.534ns (Maximum Frequency: 220.556MHz)
   Minimum input arrival time before clock: 1.946ns
   Maximum output required time after clock: 4.796ns
   Maximum combinational path delay: No path found

Timing Detail:
--------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default period analysis for Clock 'clk'
  Clock period: 4.534ns (frequency: 220.556MHz)
  Total number of paths / destination ports: 306 / 32
-------------------------------------------------------------------------
Delay:               4.534ns (Levels of Logic = 3)
  Source:            y_4 (FF)
```
```
Process "Synthesis" completed successfully
```

Fig 2(d) synthesis report for IIR implementation on Spartan 3E

The simutation waveform results are shown in fig 3(a) and 3(b) for IIR & FIR pipelined filters

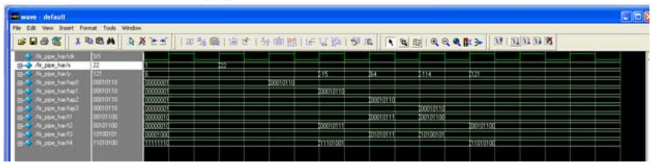

Fig 3(a) Simulation results for FIR Pipelined

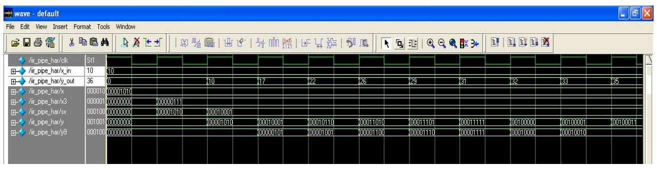

Fig 3(b) Simulation results for IIR Pipelined

REFERENCES

[1] Sami Khorbotly, Joan E. Carletta, Robert J. Veillette "A Methodology for Implementing Pipelined Fixed-Point Infinite Impulse Response Filters" 41st Southeastern Symposium on System Theory University of Tennessee Space Institute Tullahoma, TN, USA, March 15-17, 2009

[2] A. Shaw and M. Ahmed, "Pipelined recursive digital filters: a general look-ahead scheme and optimal approximation," *IEEE Trans. on Circuits and Systems II: Analog & Digital Signal Processing*, vol. 46, no. 11, pp. 1415– 1420, Nov. 1999.

[3] Chao-Huang Wei, Hsiang-Chieh Hsiao, Su-Wei Tsai "FPGA Implementation of FIR Filter with smallest Processor" IEEE, 2005

[4] Mohamed Al Mahdi Eshtawie, and Masuri Bin Othman " An Algorithm Proposed for FIR Filter Coefficients Representation"D. International Journal of Applied Mathematics and Computer Sciences 4;1 2008

[5] "Fpga Implementation Of Adaptive Iir Filters With Particle Swarm Optimization Algorithm" ZhenbinGao , XiangyeZeng , Jingyi Wang , Jianfei LiuSchool of Information Engineering, Hebei University of TechnologyTianjin 300401, P. R. China

[6] A. Avizienis, "Signed-digit number representation for fast parallel arithmetic," IRE Transactions on Electronic Computers, vol. 10, pp. 389–400, Sept.1961

[7] Y.C. Lim and B. Liu,"Pipelined recursive filter with minimum order augmentation," IEEE Transactions on Signal Processing, vol. 40, no. 7, pp.1643-1651, July 1992.

[8] K. Parhi & D. G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters I. Pipelining using scattered look-ahead and decomposition," IEEE Transactions on Acoustics Speech & Signal Processing, vol. 37, no. 7, pp. 1099-1117, July 1989.