International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-4, Issue-8)

Research Article | **August 2015**

IIR Lattice FilterStructure Design in FPGA

Jadhav Narendra, Bharade Prasad
Department of ETC Engineering, SGGSIE & T,
Nanded, India

Abstract:

The IIR filter implementation in FPGA, utilizing the dedicated hardware resources can effectively achieve application-specific integrated circuit (ASIC)-like performance while reducing development time cost and risks. In this paper, IIR filter is implemented on FPGA. Lattice filter approach in realizing a digital filter is considered. This approach gives a better performance than the common filter structures in terms of speed of operation, cost, and power consumption in real-time.The adder, multiplier and delay blocks are the three basic building blocks in the design of digital filter structure. The IIR filter is implemented in Virtex-5 FPGA XC5VLX50T device and simulated with the help of Xilinx ISE (Integrated Software Environment). Software WEBPACK project navigator 14.2version was used for synthesizing and simulation the code. For floating point IIR filter structure have been realized. Work in real time.Modules such as multiplier,adder, RAM and two's compliment, right shifter, were used.

Keywords: VHDL, Xilinx, LOP, IIR, Floating point, FPGA

I. INTRODUCTION

This paper presents designs for floating-point add and multiply, including the first published details of how to optimize each for FPGAs. Whereas most published work in this area focuses on automatically generating floating-point units with a variety of bit widths, this article focuses specifically on optimized, hand-placed implementations of the IEEE 754 floating-point standard, a requirement in many scientific computing domains. The units are optimized to achieve the highest possible clock rate without sacrificing area, because the performance of a floating-point application for FPGAs is dependent on both rea of the unit and the clock rate of the unit. Unlike ASICs or microprocessors, however,FPGAs can customize the floating-point unit on a per-application basis. Thus, the units do support a parameterizable pipeline depth to accommodate applications that do not tolerate high-latency floating-point units.

There has been extensive research into floating-point arithmetic [1] on FPGAs. Several efforts [Shirazi et al. 1995; Belanovic and Leeser 2002; Dido et al. 2002; Gaar et al. 2002a; Liang et al. 2003] have investigated the use of custom floating-point formats [2] in FPGAs. Some have even considered automatically optimizing the bit widths of floating-point formats [Gaar et al. 2002b] when using these custom formats. Compared to IEEE standards [IEEE Standards Board 1985], these formats require significantly less area and run significantly faster; however, the differences in format are typically designed to eliminate much of the complexity in the formats. Therefore, the optimizations are not directly comparableThere is also extensive work in floating-point optimizations [3] for use in microprocessors. Examining work in that field, however, highlights the dramatic differences between designs for FPGAs and designs for microproces- sors. When discussing support for rounding modes [5] [Even and Seidel 2000], for example, there is a significant focus on how to correctly support all of the IEEE rounding modes. In contrast, FPGAs are not fixed function [4] and only need to implement one roundingmode per design (though that rounding mode may change between designs). Other optimizations that are common for microprocessors[Lang and Bruguera 2004; Naini and Dhablania 2001; Schwarz et al. 2005; Seidel and Even 2001] are inappropriate in an FPGA environment where dedicated carry chains [6] and embeddedmultipliers are available. Indeed, ignoring such resources would dramatically increase both the latency of the design and the area; however, because such resources are relatively inflexible, they preclude many known techniques to optimize floating-point units in VLSI technology.

The organization of the paper is as follows. In Section II, shows floating point adder algorithm. In Section III, floating point multiplier algorithm. In Section IV, IIR lattice filter structure explained. In Section V, VHDL simulation result. Finally, Section VI concludes the paper.

II. FLOATING POINT ADDER ALGORITHM

A. Floating Point Adder Algorithm Using LOP [7-8-9]:

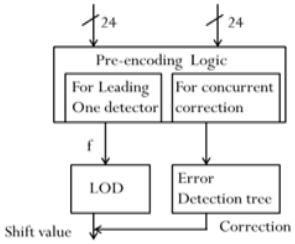

Figure 1: Structure of LOP

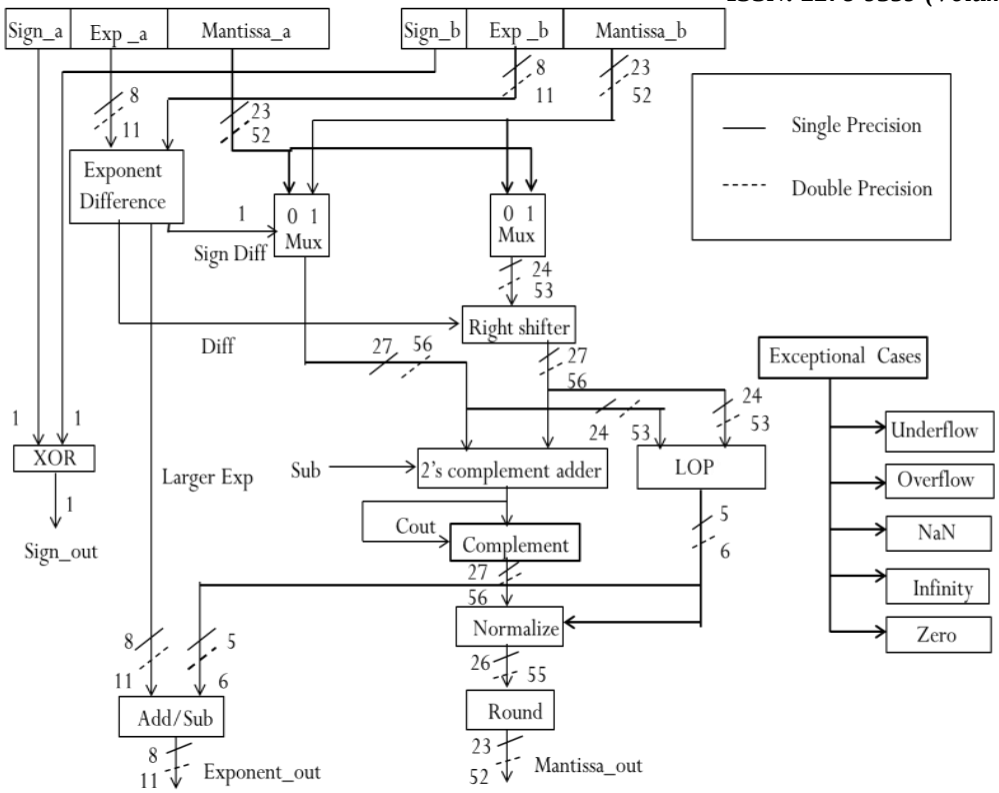

Figure 2: floating point adder using LOP

III. FLOATINGPOINT MULTIPLIER ALGORITHM

The multiplier operation consist of four steps [7],

1. Add exponents and multiply the significands
2. Normalization of M_z^* and adjust exponent:

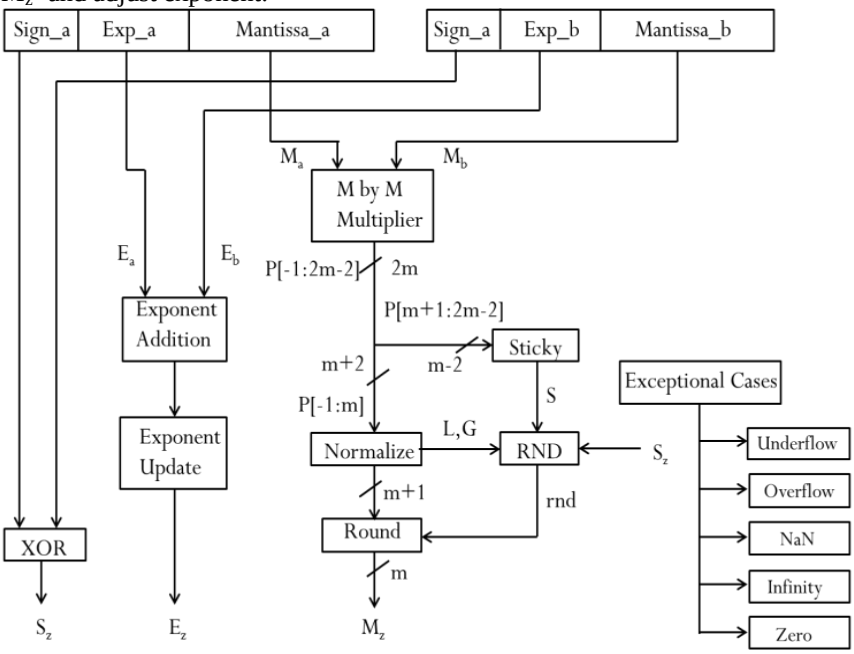

Figure 3: floating point multiplier

3. Rounding using round towards even
4. Register flags and special values.

IV. IIR LATTICE FILTER STRUCTURE

The IIR lattice filter structure [10-11] for an all pole system is given by following set of recursive equation,

$f_N(n)=x(n)$ (1)

$f_{m-1}(n)=f_m(n)-K_m g_{m-1}(n-1)$; m=N,N-1,……1 (2)

$g_m(n)=K_m f_{m-1}(n)+g_{m-1}(n-1)$; m=N,N-1,…….1 (3)

$y(n)=f_0(n)=g_0(n)$ (4)

for all pole IIR system is,

$$H_a(z)=\frac{Y(z)}{X(z)}=\frac{F_0(z)}{F_m(z)}=\frac{1}{A_m(z)} \quad (5)$$

Similarly for all zero system,

$$H_b(z)=\frac{G_m(z)}{y(z)}=\frac{G_m(z)}{G_0(z)}=B_m(z)=z^{-m}A_m(z^{-1}) \quad (6)$$

The ladder part can be obtained by,

$$y(n)=\sum_{m=0}^{M} v_m g_m(n) \quad (7)$$

$$C_M(z)=\sum_{m=0}^{M} v_m B_m(z) \quad (8)$$

The coefficients of the numerator polynomial $C_M(z)$ determine the ladder parameters v_m and the coefficient in the denominator polynomial $A_N(z)$ will determine the lattice parameters K_m.

$$C_m(z)=\sum_{k=0}^{m-1} v_m B_k(z)+v_m B_m(z) \quad (9)$$

$$C_m(z)=C_{m-1}(z)+v_m B_m(z) \quad (10)$$

$$V_m=C_m(m) \quad (11)$$

$$C_{M-1}(z)=C_m(z)-v_m B_m(z) \quad (12)$$

IIR lattice filter structure design steps:

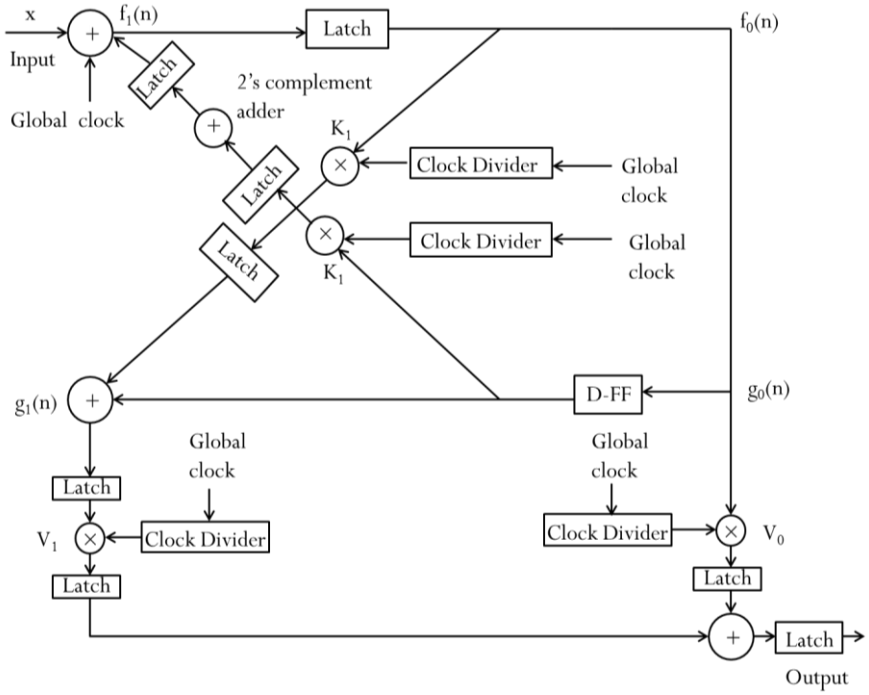

Figure 4: IIR lattice filter structure: single precision and double precision

V. RESULTS

1. Floating point adder:single precision format:

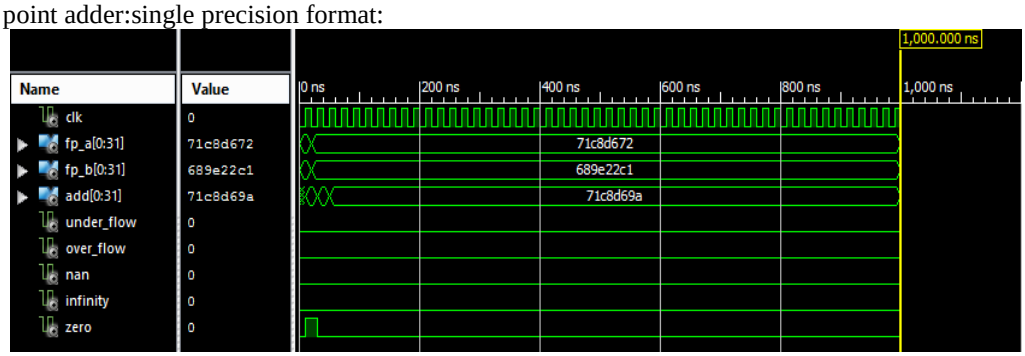

Figure 5:Floating point adder (single precision format)

2. Floating point adder: double precision format:

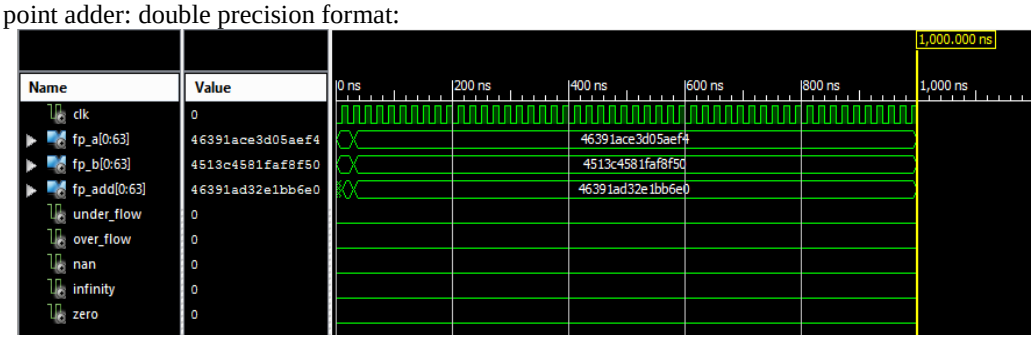

Figure 6: Floating point adder (double precision format)

3. Floating point multiplier: single precision format:

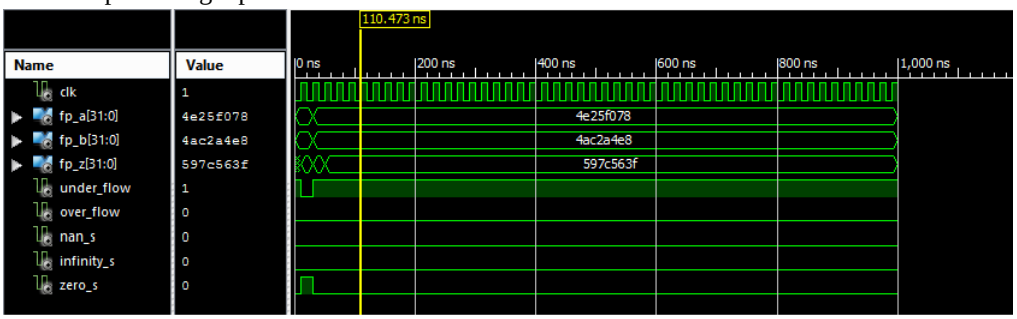

Figure 7:Floating point multiplier (single precision format)

4. Floating point multiplier : double precision format

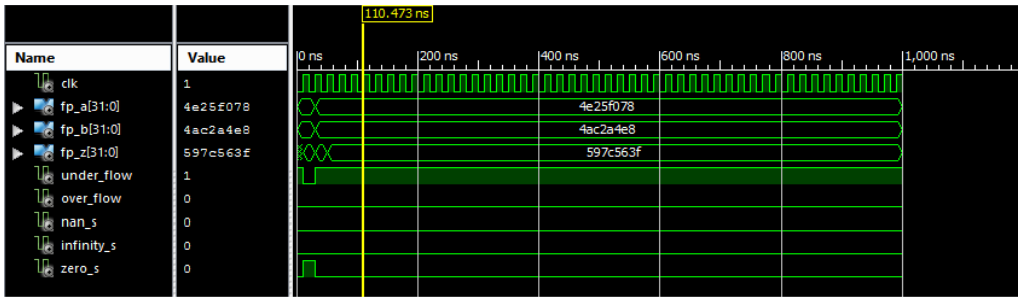

Figure 8: Floating point multiplier (double precision format)

VI. CONCLUSIONS

1. Floating point adder using LOP

FPGA implementation uses 454 slices in single precision and 658 slices in case of double precision format.

2. Floating point multiplier:

FPGA implementation uses95 slices in single precision and 196 slices in case of double precision format.

3. IIR lattice filter structure using LOP:

The IIR lattice filter structure implemented in single precision format using LOP has 1445 slices using Virtex-5 XC5VLX50T device. The IIR lattice filter implemented in double precision format using LOP has 3112 utilized slices.

4. IIR lattice filter structure : single precision and double precision floating point arithmetic

A)Single precision B) Double precision

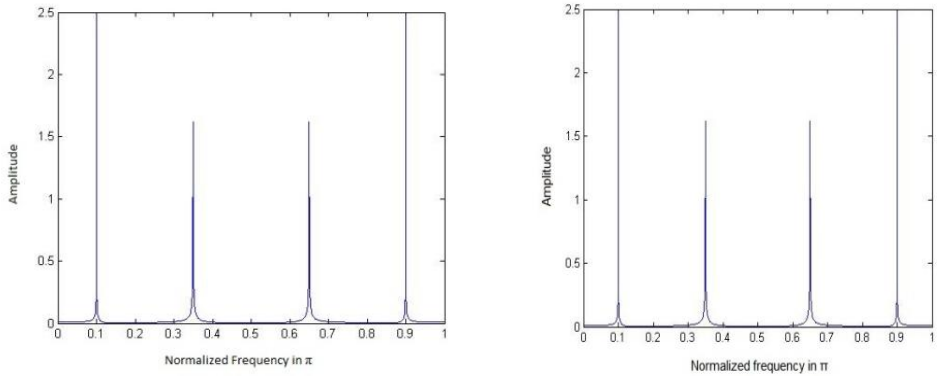

Figure 9: FFT of Input

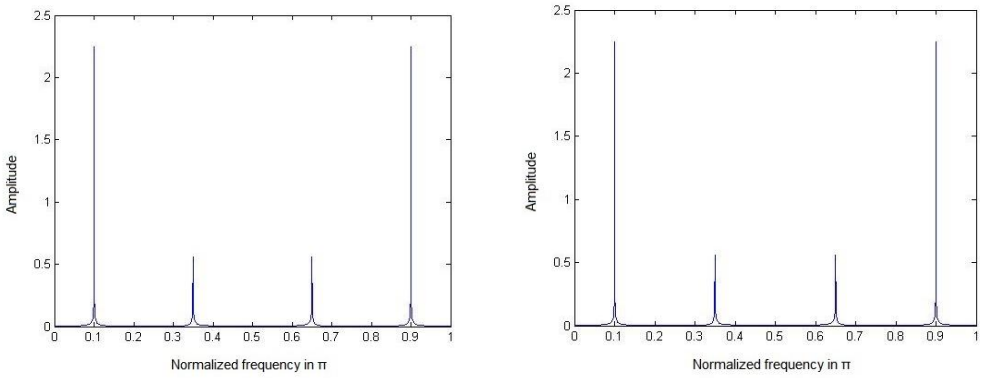

Figure 10: FFT of filtered output in MATLAB

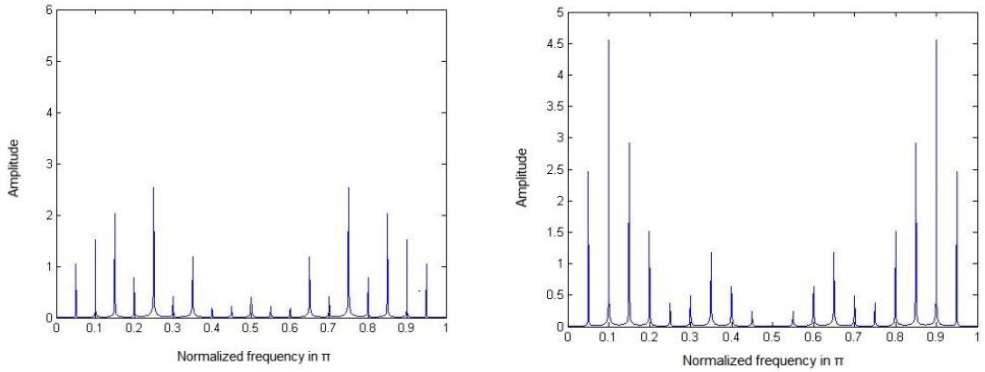

Figure 11:FFT of filtered output in VHDL

REFERENCES

[1] L. Louca, T. a. Cook, and W. H. Johnson, "Implementation of IEEE single precision floating point addition and multiplication on FPGAs," *FPGAs Cust. Comput. Mach. 1996. Proceedings. IEEE Symp.*, 1996.

[2] W. B. . I. Ligon, S. McMillan, G. Monn, K. Schoonover, F. Stivers, and K. D. Underwood, "A re-evaluation of the practicality of floating-point operations on FPGAs," *Proceedings. IEEE Symp. FPGAs Cust. Comput. Mach. (Cat. No.98TB100251)*, no. 0000, 1998.

[3] B. Roesler, Eric and Nelson, "Novel optimizations for hardware floating-point units in a modern FPGA architecture," in *Field-Programmable Logic and Applications: Reconfigurable Computing Is Going Mainstream*, Springer, 2002, pp. 637–646.

[4] O. Liang, Jian and Tessier, Russell and Mencer, "Floating point unit generation and evaluation for FPGAs," in *11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2003. FCCM 2003.*, IEEE, 2003, pp. 185–194.

[5] V. Govindu, Gokul and Zhuo, Ling and Choi, Seonil and Prasanna, "Analysis of high-performance floating-point arithmetic on FPGAs," in *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, IEEE, 2004, p. 149.

[6] "IEEE Standard for Binary Floating-Point Arithmetic," *IEEE Stand. Board ANSI*, vol. IEEE Std 7, 1985.

[7] M. D. Ercegovac, *Digital Arithmetic*. Elsevier, 2005.

[8] J. D. Bruguera and T. Lang, "Leading-one prediction with concurrent position correction," *IEEE Trans. Comput.*, vol. 48, no. 10, pp. 1083–1097, 1999.

[9] V. G. Oklobdzija, "An Algorithmic and Novel Design of a Leading Zero Detector Circuit: Comparison with Logic Synthesis," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. Vol. 2, pp. 124–128, 1994.

[10] S. K. Mitra, *Digital Siganl Processing-A computer based approach*, Fourth. Tata McGraw Hill, 2013.

[11] J. Proakis and M. Manolakis, *Digital Signal Processing*, Fourth. Pearson, 2007.

BIOGRAPHY

Narendra Jadhavwas born in Ahmedabad, India,in 1978. I received the B. Tech. and M. Tech. degree from the Dr. BATU, Lonere-Raigad, in 2000 and 2004 respectively.I am currently pursuing thePh.D. degree with the Department of ETC Engineering, SGGSIE&T, Nanded-India. My research interests include Biomedical Signal Processing and VLSI.

Prasad Bharadewas born in Sagroli, Nanded,India,in 1990. I received the M. Tech. degree from the SGGSIE&T, Nanded-India in 2015. My research interests include Digital Signal Processing and VLSI.

