A DA Serial Multiplier Technique based on 32-Tap FIR Filter for Audio Application

K Balraj[1],
Department of ECE
Dr. B.R. Ambedkar National Institute Of Technology
(NIT) Jalandhar, Punjab, India
balraj224@gmail.com

Ashish Raman[2],
Department of ECE
Dr. B.R. Ambedkar National Institute of Technology
(NIT) Jalandhar, Punjab, India
ramana@nitj.ac.in

Dinesh Chand Gupta[3]
Department of ECE
Dr. B.R. Ambedkar National Institute of Technology
(NIT) Jalandhar, Punjab, India
dinesh_chand_gupta@yahoo.com

***Abstract*--This paper presents the design and implementation of a high speed 32-tap finite impulse response (FIR) filter that employs the Distributive Arithmetic (DA) technique for the complex computation of Audio Coefficients and Multipliers. Distributive Arithmetic (DA) has been used to implement a bit-serial scheme of a general symmetric version of FIR filter due to its high stability and linearity by taking optimal advantage of the look-up table (LUT), base structure of Field Programmable Gate Array (FPGA). Distributive Arithmetic algorithm (DA-FIR) technique is employed for reducing the complex computations, thereby increasing the speed and it also reduces the area and power consumption. The design is modeled using Verilog HDL and implemented on Virtex II Pro FPGA that consumes 39% resources of FPGA and shows the clock latency of 34 cycles at 192.45 MHz clock frequency.**

Keywords- FIR filter, MATLAB, Distributive Arithmetic (DA), FPGA, Cadence RTL compiler.

I. Introduction

The most common approaches to the implementation of digital filtering algorithm are general purpose digital signal processing chips for audio application, or special purpose digital filtering chips and application-specific integrated circuits (ASICs) for higher rates[9].This approach to the implementation of Distributive Arithmetic algorithm on FPGA. Technological advancements by Xilinx in Field Programmable Gate Arrays (FPGAs) in the past 10 years have opened new paths for DSP design engineers. The FPGA maintains the advantages of the high specificity of the ASIC while avoiding the high development costs and inability to make design modifications after production. The FPGA also adds design flexibility and adaptability with optimal device utilization conserving both board space and system power which is often not the case with DSP chips. When the design is demanding more than 100 MIPS, time-to-market critical or design adaptability is crucial the Xilinx FPGA is a great solution. When designing a DSP system in an FPGA the data can be processed taking advantage of single chip paralleled structures and arithmetic algorithms to exceed the performance of a single general-purpose DSP device. Distributed Arithmetic [2] used for Baugh-Wooley multiplication is just one of the approaches used to increase data bandwidth and the throughput as much as several orders of magnitudes greater in FPGAs than that possible in off-the-shelf DSP solutions. One example is a 16-Tap, 8-Bit Finite Impulse Response (FIR) filters [3]. It can support more than 780 MIPS at 5 million samples per second while occupying less than 2,100-gates or 70% of the 100 CLBs in an XC4003 device. Another example is a 32-Tap, 10-Bit FIR filter. It can support more than 3 GOPS at 5 million samples per second. When increasing the number of Taps has no significant impact on the sample rate when Distributed Arithmetic is used; The FPGA design cycle requires less hardware specific knowledge that required by most DSP chips or ASIC design solutions. Smaller design groups with less experienced engineers can be given the opportunity and trusted to design larger, more complex DSP systems in a shorter amount of time than that of larger design groups with more experienced engineers who are required to have knowledge of device specific programming languages. The FPGA design team can have a complex DSP system designed, tested, verified.

The paper is organized in the following sections: In section II FPGA Implementation of FIR filter using DA Technique to compression are discussed. Section III MATLAB designing of FIR filter explained in this section. The performance parameters of the system are discussed in section IV. Finally conclusions have been drawn in section V.

II. FPGA Implementation of FIR Digital Filter Using DA Technique and Baugh-Wooley Multiplier

This paper focus on the implementation of a specific fixed coefficient FIR filters by using the Xilinx tools. The specific filter is designed with 10-bits input and 32-bits coefficients, which are optimized by using our new public-domain software. Our objective is to compare the new software techniques with the technique used in the Xilinx audio application. The Audio application use to basic building blocks to build an 32-tap low pass FR filter with 10 bits input and 32 bits signed coefficients. The basic building blocks include constant coefficient Multiplier (KCM), Adders, registers and a delay locked loop. In fully parallel FIR filter, the inputs to the multiplier are the tap data and the constant coefficient. These multipliers are called KCM since one of the inputs in a constant. In Xilinx, KCMs are implemented using the Hybrid Technique [q. An example structure similar to [6] is presented in Figure 1. Efficient implementation of the KCM is achieved by the authors by storing the pre-computed partial products of the fixed

coefficients. These partial products are stored in ROMs using distributed memory in Xilinx FPGAs. In our new public domain software, the multipliers are constructed using shifting and adding arithmetic operation. The performance of the filter generated by the software is then comparing to the results, which appear in the Audio application. Also, note that the Audio application uses a low pass FIR filter. To be consistent with the Audio application, look up table (LUT) and the number of slice registers of the Xilinx technology are the main basis for comparison. However, the Audio application structure has a latency of four clock cycles while our structure outputs data at the end of first clock cycle itself.

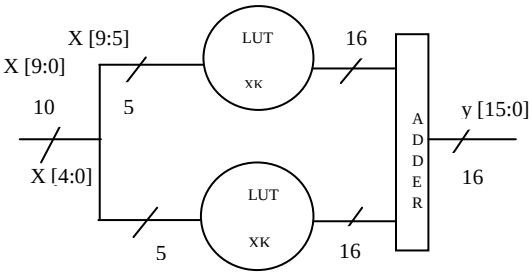

Figure1. *Look Up Table Architecture*

A. FIR FILTER IMPLEMENTATION

Finite impulse response (FIR) filtering is one of the most widely used operations in Digital Signal Processing (DSP) devices [2]. The basic equation of the FIR filter is given as

$$H[z] = \sum_{n=0}^{N-1} h[n] Z^{-n} \qquad (1)$$

N is the length of h (n), that FIR filter tap number. Because it is z^{-1} of the (N-1) polynomial, which has (N-1) a zero in the z plane, the original z= 0 is a (N-1) re-order pole. The FIR filter direct-type structure as shown in Figure2, the output can be expressed as:

$$y[n] = \sum_{i=0}^{N-1} h[i]x[n-i] \qquad (2)$$

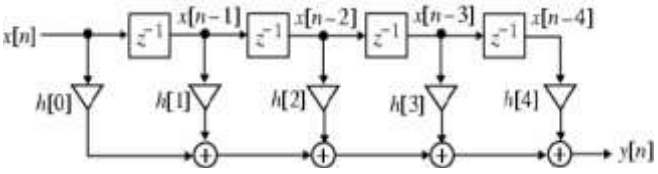

Figure2. *Direct-type FIR filter*

It is not difficult to achieve this structure of FIR filter with adder and the multiplier but the direct implementation of the FIR filter regardless of speed or in power consumption is failed.

Using FIR filter in the actual system, multiple applications of the FIR filter linear-phase characteristics, according to the any coefficients used to design. Specification of the given FIR filter is: 32taps, 10-bits data and 32-bits coefficient width and single MAC implementation. Basic block diagram of the direct form programmable FIR Filter is shown in the Fig.3.

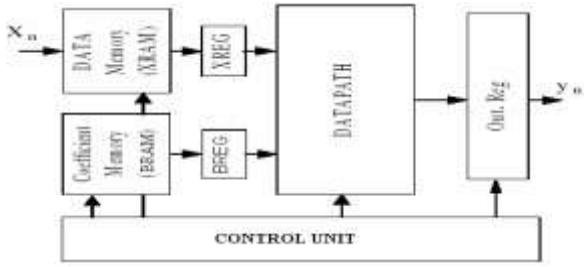

Figure3. Basic Block diagram of FIR Filter

Data width of all components are 10-bit, XRAM and BRAM both have size 10x64[2]. XRAM stores the input data while coefficients are already stored in the BRAM. Controller is responsible for ordering and control of each logic function. It generates the addresses, read, and write signals for both memories. Data path is responsible for performing data manipulations. As far as operation is concerned, the present and N-1 previous data samples of input x (n) comes to the data path through XREG & are multiplied by corresponding N-tap coefficients one by one at each clock through BREG. Summation is also done at each clock by adding current and previous results of multiplications to form the filter output y (n) after N cycles. Data path consists of a single 10-bit MAC unit and a round off module to get a 16-bit output of the filter output after rounding the 32-bit output of MAC unit. Direct form FIR filter implementation is run time programmable and uses generic multiplier. Upper limit for the number of taps is 32, 10-bits data and 32-bits coefficient. Power, area and speed results of this FIR filter are considered as a referenced to study this effect of partitioned multiplier on the performance of Direct Form FIR Filter.

B. Distributed Arithmetic (DA)

Distributed arithmetic (DA) is an important FPGA technology. It is extensively used in computing the sum of products,

$$y[n] = < h, x > = \sum_{n=0}^{N-1} h[n]x[n] \qquad (3)$$

DA system, assumes that the variable x [n] is represented by

$$x[n] = \sum_{b=0}^{B-1} X_b[n]2^b, x_b[n] \in [0,1] \qquad (4)$$

If h [n] is the known coefficients of the FIR filter, then output of FIR filter in bit level from is:

$$y[n] = \sum_{n=0}^{N-1} h[n] X \sum_{b=0}^{B-1} x_b[n] X 2^b \qquad (5)$$

In distributed arithmetic from is given below eq.

$$y[n] = \sum_{b=0}^{B-1} 2^b X \sum_{n=0}^{N-1} f(h[n], x_b[n]) \qquad (6)$$

In Eq. (6) second summation term realizing as one LUT. The use of this LUT or ROM eliminates the multipliers [10]. Distributed Arithmetic (DA) algorithm as a very efficient solution especially suited for LUT- based FPGA architecture. Using the Baugh-Wooley multiplier architecture is used on an efficient partition of the function in partial terms using 2's complement binary data representation. The partial bit terms can be stored in LUTs [8], observed that the requirement of ROM/LUT capacity increases exponentially with the order of the filter and taps of the filter.

C. *Baugh-Wooley multiplier*

The Baugh Wooley (BW) multiplier is an efficient way to handle the sign bits. This technique has been developed in order to design regular multipliers, suited for 2's-complement numbers [8]. Gebali has extended this basic idea and developed efficient fastest product processors capable of performing multiply-accumulate operations without the speed penalty [7]. Let us consider two n-bit numbers, A and B to be multiplied. A and B can be represented as.

$$A = -a_{n-1}2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i \qquad (7)$$

$$B = -b_{n-1}2^{n-1} + \sum_{j=0}^{n-2} b_j 2^j \qquad (8)$$

Where the a_i and b_i are the bits in A and B respectively, and a_{n-1} and b_{n-1} are sign bits. The product of A*B=P is given below

$$P = A*B \qquad (9)$$

$$P = (-a_{n-1}2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i) \times (-b_{n-1}2^{n-1} + \sum_{j=0}^{n-2} b_j 2^j) \qquad (10)$$

The final product is obtained by subtracting the last two positive terms from the first two terms

$$P = a_{n-1} b_{n-1} 2^{2n-2} + \sum_{i=0}^{n-2} \sum_{j=0}^{n-2} a_i b_j 2^{i+j}$$

$$-2^{n-1} \sum_{i=0}^{n-2} a_i b_{n-1} 2^i - 2^{n-1} \sum_{j=0}^{n-2} a_{n-i} b_j 2^j \qquad (11)$$

D. *Baugh- Wooley 8- Bit Multiplier*

Multipliers are complex adder arrays. Figure 5 shows the 8-bit BW multiplier. A BW multiplier is a regular multiplier that is suited for 2's – complement numbers. Multiplication is done two steps. This BW multiplier each box corresponded to an AND gate, full- adder shown in figure [4].

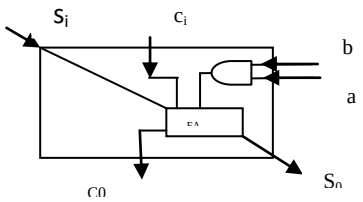

Figure4. *BW multiplier cell construction*

Figure [4] shows internal structure of cell construction in Baugh-Wooley multiplier. The main aim is power consumption by using Baugh-Wooley multiplier. This multiplier high speed and less area required.

Figure5. *16, 8-bit scalable BW Multiplier [5]*

Figure [5] shows 16 bit multiplication has an output of 32 bits, when doing a multiplication of 8-bit precision; only the most significant 8 bits need to go through the final stage of adders. The results of the products in the less significant 8 bits can be directly routed to the output, 8 additional multiplexers after the last row of adder will control the output bits 31 to 16, based on the information whether the circuit is performing a multiplication of either 8-bits or 16-bits. During the 8-bit precision multiplication, there is switching activity only in the lower, the output further lower power consumption.

III. *MATLAB Design of 32-tap FIR filter for Audio Application*

IV.

A. *Design performance requirements*

Design problem of FIR digital filter is determine the transforming the required sequence or the impulse response of the constant problems design a system function H (z) to be approaching the technical use of the finite impulse algorithm.

Design methods are mainly low pass filter function method, frequency sampling method, time domain method, This Filter design methodology used to FIR low-pass filter, sampling frequency of 40000Hz; Pass-band frequency of 22000Hz; stop-band frequency of 1200Hz.

B. *Audio Application specific design method of FIR filter*

The MATLAB TOOL is used to design the audio recording specific signal spectrum. We get some of the impulse response.

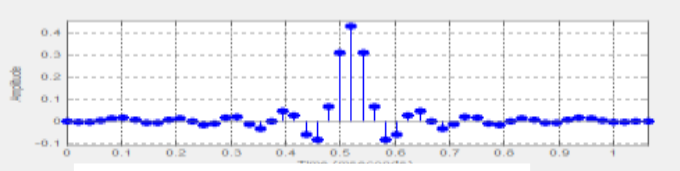

Figure6. *Impulse response of FIR filter*

Generate the original speech signal in MATLAB after add the noise speech signal and finally we get the reconstructed noisy speech wave forms shown in below figures.

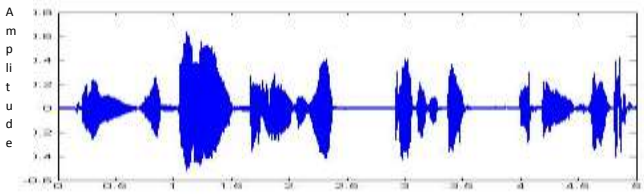

Figure7. *Original speech signal*

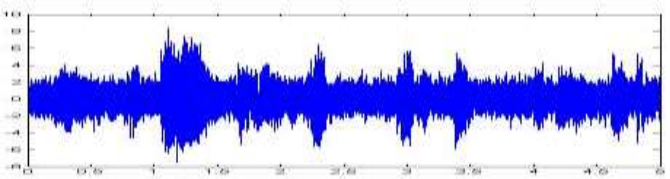

Figure8. *Noisy speech signal*

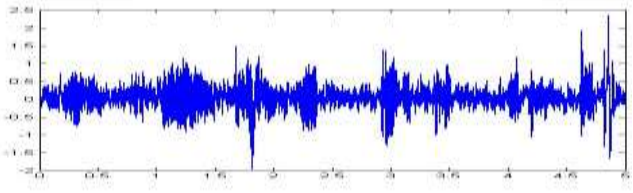

Figure9. *Reconstructed noisy speech signal*

IV. *Experimental Results of FIR filter*

A. *Determine the impulse response coefficients*

Using MATLAB software to find the filter coefficients, the 32-tap FIR filter using audio application unit impulse response are real numbers to meet the odd or even symmetry conditions. Unit impulse response coefficients obtained are as follows table1.

Table I. Impulse Response Coefficients

Coefficient number	Fir Digital Filter Coefficients
H[0]=H[31]	0.098
H[1]=H[30]	-0.811
H[2]=H[29]	1.097
H[3]=H[28]	0.353
H[4]=H[27]	0.0091
H[5]=H[26]	-0.6692
H[6]=H[25]	-0.0236
H[7]=H[24]	0.9774
H[8]=H[23]	0.0624
H[9]=H[22]	1.5443
H[10]=H[21]	1.2018
H[11]=H[20]	-1.05689

The most of FIR filter tap coefficients are the decimal and signed numbers but do not support floating numbers you need to convert tap coefficients into a binary numbers so the number of taps coefficient code is an issue that must be considered. This design uses SD code to minimize the code number of 1and -1 to achieve multiplier at a minimum cost. Filter coefficients of the SD code as follows, each tap coefficient of the first left seven (multiplied by 128) accurate to four decimal places. Some example is given here.

$$128 \bullet h(0) = h(31) = (1.1648)_{10} = (1.00101),$$
$$128 \bullet h(1) = h(30) = (1.5872)_{10} = (1.1001),$$

B. *Verilog simulation results for 32-tap FIR filter*

According to the characteristics of FIR filter coefficients even symmetric, adding up the output of the symmetric tap a coefficient then multiplied by the corresponding weighted of the binary and then adds the results. Top level of 32-tap filter synthesizes results shown in below.

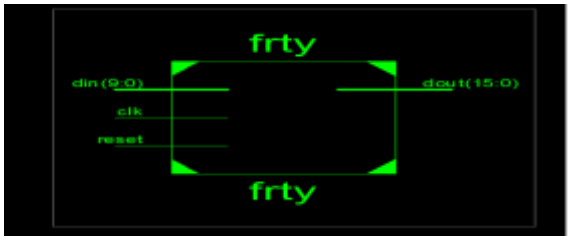

Figure10. *Top level of 32-tap FIR filter*

We Design the Net list of 32-tap FIR filter figure from Cadence RTL Compiler.

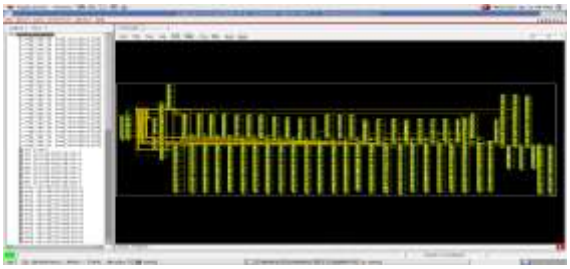

Figure11. *Net list of 32-tap FIR filter*

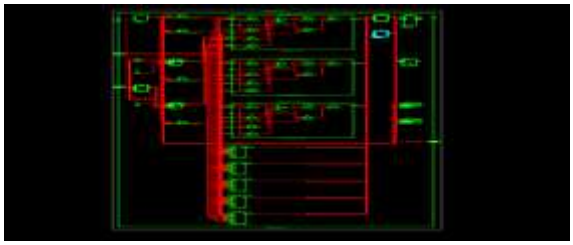

Figure12. *RTL Schematic of 32-tap FIR filter*

Figure13.*Simulation result of 32-tap FIR filter using audio application*

Simulation is as shown in Figure13 can be verified: the results of y (n) =h (n) *x for example, the corresponding pulse (00001111) derived from the output y (7) =h (n) *7=38.54*15=578.10 and y (7) = $(00000000000000000000001001000010)_2$= (578), comparison simulation results with the actual operational error =0.68, the results are very close to meet the design requirements.

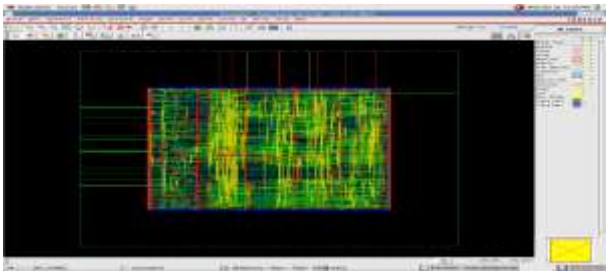

Figure14. *Layout of 32-tap FIR filter using Speech signal in SOC*

C. *Comparison of Power, Area and Timing Report*

Table2. Shows the comparison of power, area and timing report between different architectures of 32-tap FIR Filter.

Table 2**:** Comparison of Different Architectures Results

Types	Power (mW)	Area(μm^2)	Timing Report(Ps)
32-tap Decimation Filter using audio application	29.329	42843	3567
DA serial multiplier using audio application	22.568	29394	2987
This work 32-tap FIR filter using DA and BW algorithm for audio application	**19.755**	**19657**	**1818**

IV. *Conclusion and future work*

In this paper a 32-tap FIR filter is designed for Audio application. Xilinx 13.2 ISE is used for designing filter that uses DA and BW algorithm and both are implemented on FPGA vertex II pro. Simulation results shows that both DA and BW consumed less power and area as compare to decimation and DA serial multiplier for desired application. The DA and BW consume 39% resources of FPGA and show the clock latency of 34 cycles at 192.45 MHz clock frequency. Pipelining may be used for better results.

REFERENCES

[1] Y. Libo, "FPGA application development" from practice to improve Beijing: China Electric Power Press, 2007

[2] W.Jinmig, "VerilogHDL programming tutorial". Beijing: People's Posts & Telecom Press, 2004.1

[3] Z. ghaijun, "FPGA-based 16-order FIR filter design and Implementation". Journal of Anhui University: Neuroscience Edition, 2009.

[4] Z.B.cheng, "Dynamic Distributed Algorithm for FPGA-based research and application". Journal of Changchun University of Technology: Natural Science Edition, 2005.12.

[5] I. Koren, "Computer Arithmetic Algorithms", 2nd Edition, A K Peters, Ltd. 63 South Avenue Natick, MA 01760.

[6] K. Chapman, "Building High Performance FIR Filter Using KCM Xilinx" Ltd -UK, July 1996.

[7] S. Akhter, "VHDL Implementation of Fast NXN Multiplier Based on Vedic Mathematics", Jaypee Institute of Information Technology University, Noida, 201307 UP, INDIA, 2007 IEEE.

[8] P. Mehta and D. Gawali, "Conventional versus Vedic Mathematical Method for Hardware Implementation of a Multiplier," Proceeding International Conference on Advances in Computing, Control, & Telecommunication Technologies, 2009. ACT '09, pp-640-642, 2009.

[9] K.Y.Khoo, A. Kwentus, and A. N. Willson, Jr. "An efficient 175 MHz Programmable FIR digital filter". In IEEE Int.Symp. Circuits and Syst., pages 72-75, May 1993.

[10] T. Vigneswaran and P. S. Reddy "Design of Digital FIR Filter Based on DA algorithm" Journal of Applied Science, 2007

[11] J. S. Park, "Computation Sharing Programmable FIR Filter for Low-Power and High-Performance Applications", IEEE Journal of Solid-State Circuits, vol. 39, no. 2, February 2004.