VOL. 10, NO. 11, JUNE 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

ANOPTIMIZED ARCHITECTURE FOR ADAPTIVE DIGITAL FILTER

Britto Pari J. and Joy Vasantha Rani S. P.
Department of Electronics Engineering, MIT campus, Anna University, Chennai, India
E-Mail: brittopari@yahoo.co.in

ABSTRACT

In this paper, we propose an efficient adaptive FIR filter architecture using a single multiplier and adder irrespective of number taps using the concept of time sharing multiplier architecture. For efficient optimization of multiplier architectures, Output Product Coding and parallel pipelined multiplier are applied. The proposed Adaptive FIR filter architecture is implemented for 32-tap using Verilog and synthesized using XILINX VIRTEX-5 FPGA device. The results are validated using FPGA in Loop (FIL), where simulation is done using MATLAB/Simulink-xPC target tool box. This design provides substantial area reduction compared to the conventional Adaptive FIR filter architectures for the FPGA implementation. The proposed Adaptive FIR filter supports up to 323 MHz input sampling frequency for FPGA implementation.

Keywords: filter, look up table (LUT), odd multiple storage, output product coding (OPC), least mean square (LMS), FPGA in loop (FIL).

1. INTRODUCTION

The current research is highly focused in the area of adaptive signal processing applications such as channel equalization, acoustic echo cancellation, interference cancellation, system identification and so on. These applications require higher order Digital filters thereby making the hardware complex. Hence the area occupied to be large when implemented in FPGA chips. Therefore researchers have been continuously focusing their work on reduction in the hardware complexity. [1]- [3].

Finite Impulse Response (FIR) digital filter is widely used as a basic block in signal and image processing applications. Since the number of multiply-accumulate (MAC) operations required per filter output increases linearly with the filter order, the real-time implementation of the higher order filters is a challenging task. Several attempts havebeen made and continued to develop low-complexity dedicated VLSI systems for these filters [4]–[10].

There are two basic variants of memory-based techniques used for avoiding the usage of embedded multipliers They are based on Distributed Arithmetic (DA) for inner product computation [11]-[16] and computation of multiplication by look-up-table (LUT) [17]–[24]. In the LUT based approach, multiplications of input values with a fixed-coefficient are performed by an LUT consisting of all possible pre-computed product values corresponding to all possible values of input multiplicand while in the DA-based approach, an LUT is used to store all possible values of inner-products of a fixed-point vector. If the inner-products are implemented in a straight-forward way, the memory-size for implementation of LUT-multiplier increases exponentially with the word length of input values, whereas the memory size of the DA-based approach increases exponentially with the inner-product-length. It is observed that the reduction of memory-size achieved by such decompositions is accompanied by increase in latency as well as the number of adders and latches.

In Adaptive FIR filter architecture, the direct implementation of N-tap FIR filter requires N MAC operations, which are too expensive in hardware implementation due to its logic complexity and area constraint. Therefore an architecture has to be designed which overcomes the above constraints. In this paper, a filter is implemented with time sharing multiplier architecture across single MAC core irrespective number of taps by increasing the filter operating frequency. For optimizing the performance of the multiplier, Output Product Coding (OPC) scheme [5] and parallel pipelined multiplier architectures are implemented. The main advantages of these two architecture schemes are that they reduce complexity and increase the speed of the architecture in the design by manipulating the odd multiples of The main advantages of OPC architecture scheme is that it reduces complexity and increase the speed of the architecture in the design by manipulating the odd multiples of the fixed coefficient Where as in pipelined parallel multiplier the speed of the architecture is improved by changing the combinational logic into sequential logic with inserting pipelined registers. The performances of the proposed architectures are analysed in terms of area and time. The results are validated using FPGA in the Loop (FIL) technique.

The rest of the paper is organized as follows. In Section II describes about efficient Multiplier structures for Adaptive FIR filter. Section III describes the details of architecture design of Adaptive FIR Filter. The performance of the design are analyzed and discussed in Section IV. Finally, Section V concludes the paper in brief.

2. EFFICIENT MULTIPLIER STRUCTURES FOR ADAPTIVE FIR FILTER

In the filter the MAC structure and delay blocks are the main building blocks. The performance of the DSP algorithms entirely depends upon the multipliers in terms

VOL. 10, NO. 11, JUNE 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

of critical path. The two schemes discussed here are OPC and pipelined parallel multiplier architectures.

a) OPC scheme

When the both negative and positive sample of the signal exist in case of bipolar signals, odd multiple scheme method [5] fails. This disadvantage can be overcome by Output Product Coding proposed by [5]. Consider the input and the co-efficient to be in L-bit sign magnitude format. The most significant bit of both input and co-efficient corresponds to the sign value while the remaining (L-1) bits correspond to the magnitude. In order to obtain the complete product value in the sign magnitude representation, the magnitude and the sign bits are processed separately and finally appended. Memory optimization is also achieved by applying the OPC technique in LUT.

From Table-1 it is observed that the address value of ith row is the two's complement of (128+2-i)th row for $2 \leq i \leq 64$. The sum of product values on these two rows is 128A. Let the product values on the i^{th} and the $(128+2-i)^{th}$ be u and v, respectively. The relation between 'u' and 'v' is given as

$$u = \left[\frac{u+v}{2} + \frac{v-u}{2}\right] \text{and } v = \left[\frac{u+v}{2} + \frac{v-u}{2}\right] \quad (1)$$

where (u+v) is 128A. From Table-1 it is observed that there are three considerations namely, a constant value, value accessed from the look up table and also whether an addition or a subtraction is to be performed. The constant value is nothing but the value obtained after the computation of (u+v) which is given by 128A. The value accessed from the memory array depends upon the magnitude portion of the input vector. i.e., for an input of L bits, only (L-2) bits are considered for memory access.

The block diagram shown in Figure-1 corresponds to the architecture for the computation of the product value using OPC method for an input of 8-bits (inclusive of the sign bit). It has an encoder that converts an input of 7 bits (magnitude alone) to a 6 bit address value. The conversion is controlled by logical relations (1a) to (1g).

$$D_5 = \overline{X_6}.X_5 \quad (1.1a)$$
$$D_4 = \overline{X_6}.X_4 \quad (1.1b)$$
$$D_3 = \overline{X_6}.X_3 \quad (1.1c)$$
$$D_2 = \overline{X_6}.X_2 \quad (1.1d)$$
$$D_1 = \overline{X_6}.X_1 \quad (1.1e)$$
$$D_0 = \overline{X_6}.X_0 \quad (1.1f)$$

The control circuit decides the operation of two's complement to be performed on (L-2) bit. If L-2 bit is '1' two's complement address is considered and the normal address for '0' which becomes the final address to access the value from the memory array consisting of stored value in the LUT as tabulated in Table-1. The next unit to be considered is the add/subtract unit which decides whether to add or subtract the value accessed from the memory array with the constant value of $\frac{u+v}{2}$. Here the value of $\frac{u+v}{2}$ is 64A. The add/subtract unit makes a decision on this and performs the operation accordingly. The sign value of the output is given by the logical relation $s = x_7 \oplus A_7$. This sign bit is concatenated with the magnitude in the append block for a complete product. Therefore, for an input of N-bits, this method makes use of 2L-2memory locations thereby reducing the number of memory locations further.

Table-1. Look up table for OPC scheme.

Address $X_6X_5X_4X_3X_2X_1X_0$	**Product value**	**Product can be written as**	**Stored value in LUT**
0	0	64A-64A	64A
1	A	64A-63A	63A
10	2A	64A-62A	62A
11	3A	64A-61A	61A
100	4A	64A-60A	60A
. . .	. . .	. . .	. . .
111100	60A	64A-4A	4A
111101	61A	64A-3A	3A
111110	62A	64A-2A	2A
111111	63A	64A-A	A
1000000	64A	64A-0	-
1000001	65A	64A+A	-
1000010	66A	64A+2A	-
. . .	. . .	. . .	. . .
1111100	124A	64A+60A	-
1111101	125A	64A+61A	-
1111110	126A	64A+62A	-
1111111	127A	64A+63A	-

b) Parallel pipeline multiplier

The parallel multipliers are commonly used in high performance digital signal processors. They require more hardware compared to the serial multiplier in order to provide performance improvements. There are various ways to implement the parallel multipliers [11]. The simplest way of implementing n x n1 bit parallel multiplier is to generate all partial products and reduce them to rows of carry and sum signals.

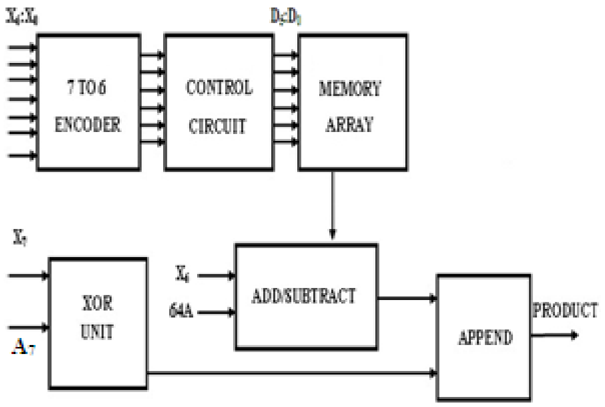

Figure-1. Proposed architecture for OPC scheme.

The final step is the addition of the generated carry and sum signals. The major disadvantage of this process is that speed is decreased due to the bit wise multiplier operation. In a parallel pipelined multiplier, the first step performed is the formation of a bit-product matrix. A bit-product matrix is simply an array of bit-products formed by multiplication of the individual bits of the two numbers being multiplied, a multiplicand and a multiplier. The generated partial products and their summed resultants are stored in the pipelined registers. The pipelined registers are inserted to increase the speed of the operation by changing the combinational logic into sequential logic. The principle is illustrated in the Figure-2.

3. EFFICIENT ADAPTIVE FIR FILTER ARCHITECTURE USING LMS ALGORITHM

Filters with fixed coefficients are suitable at the condition of known characteristics of signal and noise. However, they become inapplicable when the characteristics are not known or in other words when the input signal is not stationary. Therefore, an adaptive filter is used whose coefficients vary depending upon the input. One of the basic and simple adaptive algorithms is the gradient based algorithm. The LMS (least mean square) algorithm is one of the adaptive algorithms based on the method of steepest descent. This algorithm aims at reducing the gradient of the squared error value. The adaptive filter also has the same kind of implementation as that of a fixed filter with the usage of delay elements, adder, and multiplier.

This magnitude or the difference value is given as a feedback input to the system. Adaptive algorithm makes use of the input and the error magnitude in order to vary the weight values such that the obtained actual output is very much near to the desired output i.e., e(n) value is minimizedor ideally zero. Therefore, this process of varying the weights continues in a recursive manner until the lowest possible value is reached. The minimum value of e(n) can be obtained only if d(n) the desired output and the actual output Y(n) are equal.

However, in practical situations the ideal value of the error value being zero cannot be obtained but it can be done to get the lowest possible e(n) value. The most basic and the familiar adaptive algorithm is the least mean square algorithm (LMS).The equation governing the LMS algorithm is given as shown in equation (2).

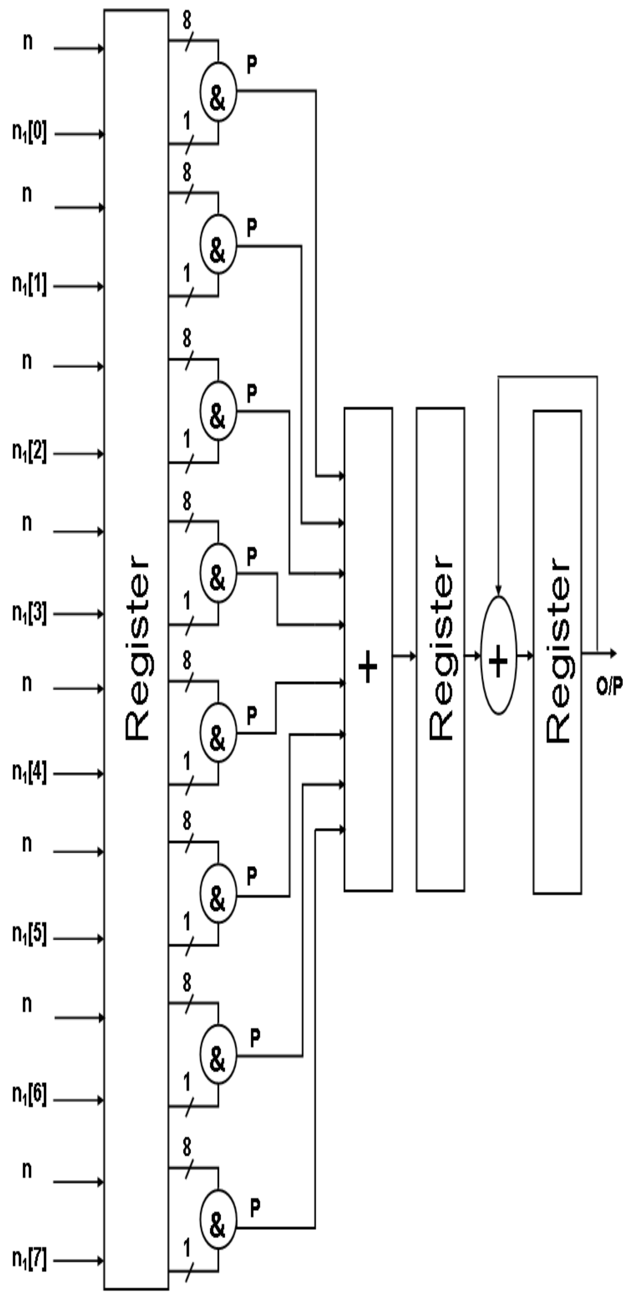

Figure-2. Parallel pipeline multiplier.

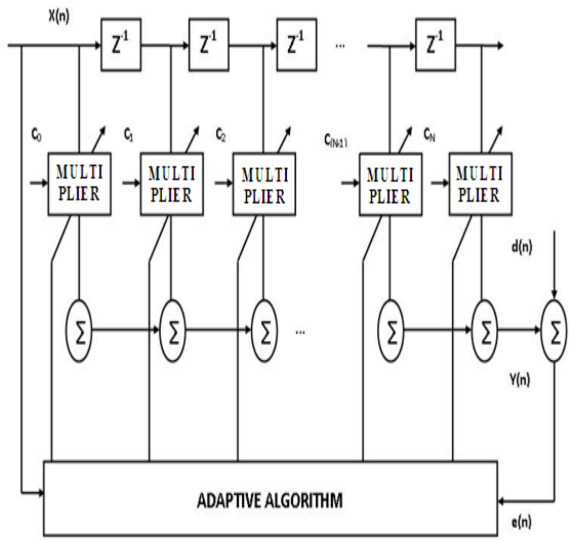

Figure-3. Conventional Adaptive architecture.

$$w(n+1) = w(n) - \mu\Delta\varepsilon[n] \tag{2}$$

The equation (2) is called as the weight update equation. The parameter μ is called as the step size and the value ε[n] represents the mean square error value. Figure-3 shows the conventional architecture scheme implemented in Adaptive FIR filter. In these Adaptive FIR filter architecture, by increasing the number of taps, number of multipliers is also increased linearly. In order to overcome these difficulties, the efficient architecture is proposed with time sharing multiplier architecture across the single MAC core by increasing the output filter frequency. Suppose for 2 tap FIR filter, the input sampling rate is 1Mega samples per second(MSPS), by increasing the sampling rate of output filter to '2x' MSPS, for each clock cycle, data is injected to the multiplier to do filter operation and the output is obtained within two clock cycles.

The proposed adaptive FIR architecture consists of following elements as shown in Figure-4 Multiplexer is used to select the data across the registers and perform the multiplier operation with the filter coefficients are stored in registers. Accumulator block is used to add the previous data value to the present data value and is set to zero after 2 clock cycles. Multiplexer Select lines and an accumulator operation are selected by only one generic counter. For sixteen tap filter, 16 clock cycles are required. Similarly implementation for any number of taps can be done by using single multiplier and adder by increasing operating frequency of FIR filter. The error signal is the difference between desired signal 'din' and filter output 'Y_out' and is fed back to the input. The error signal is multiplied with a step index of μ =0.06, the resultant is multiplied to x_in to simultaneously update the filter coefficients c0_reg and c1_reg.

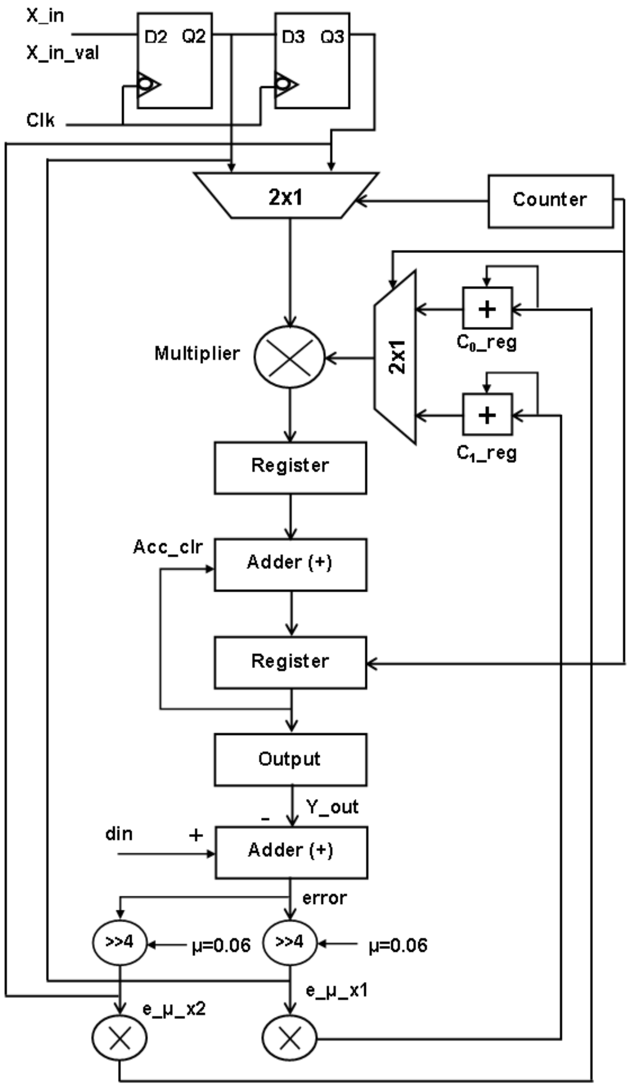

Figure-4. Proposed Adaptive FIR filter architecture.

The performance of the filter further enhanced by the introduction of two architecture schemes which reduces the complexity and critical path. Therefore, Parallel pipeline multipliers are used in systems where increased performance is required. In these multiplier, the pipelined registers are inserted to increase the speed of the operation by changing the combinational logic into sequential. By means of OPC based LUT multiplier, the number of memory locations needed to store partial products is reduced from 2L to 2L-2. Hence the memory size is greatly reduced as compared to odd schematic based multiplication, which leads to reduction in complexity.

4. RESULTS AND DISCUSSIONS

The proposed Adaptive FIR Filter architectures are implemented using two methods: (i) Output product coding and (ii) Parallel Pipelined Multiplier, and both are synthesized using XILINX VIRTEX-5(XC5VSX95T-1FF1136.

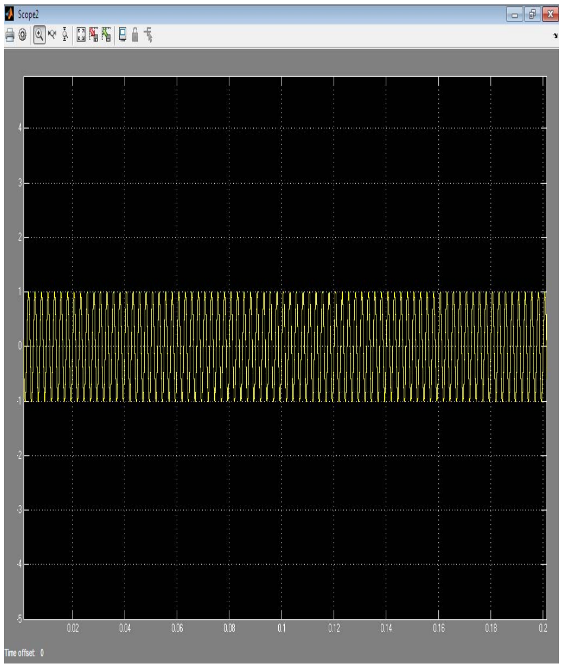

Figure-5. Input signal of Adaptive FIR filter (400HZ).

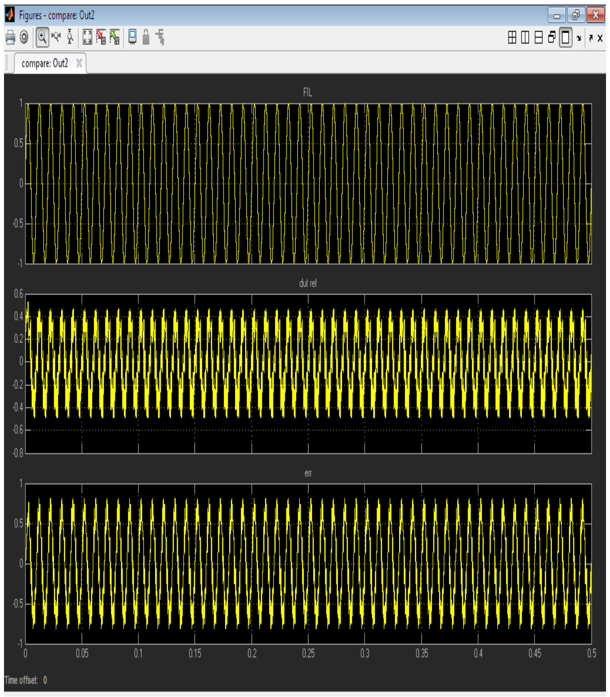

Figure-6. FPGA IN LOOP(FIL) simulation results obtained through Adaptive FIR filter implementation on Altera DE2 -115 board a) desired input(100HZ) b) error signal c) filter output.

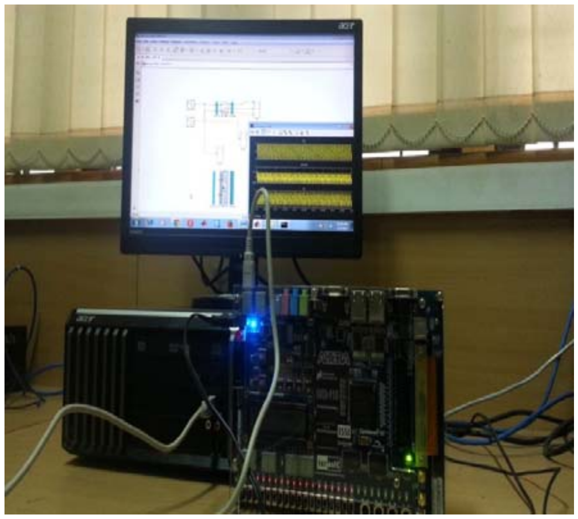

Figure-7. FPGA in loop (FIL) simulation setup.

FPGA In Loop (FIL) Simulation is done using MATLAB/Simulink-xPC target tool box which provides connection to Altera DE2 -115 FPGA kit. This helps in testing the models in real time. Figure-5 and Figure-6 shows the time domain input/output curve based on LMS algorithm, the results shown LMS algorithm can remove the noise signal, and output signals can converge to the input signal. The 400 HZ sine wave is used as the input.

The desired signal (dut_ref) 100HZ was generated by filtering the input with an FIR filter whose coefficients are evaluated by LMS algorithm. The step index taken as μ=0.06 and Figure-7 shows the FPGA in loop simulation setup. The application uses Simulink® and an FPGA development board to verify the HDL implementation of LMS Adaptive FIR filter.

The performance results of Adaptive FIR filter architectures are analyzed in Table-2 and Table-3. From the Table-2 and 3, it is observed that time sharing multiplier architecture using Single MAC core achieved drastic area reduction. The speed of the architecture is further improved by pipelining registers across the multiplier and filter output. From Table-2 and 3, it can be analyzed that the design has drastic area reduction compared to the conventional Adaptive FIR filter architectures for the FPGA implementation.

5. CONCLUSIONS

The proposed method shown as efficient schemes for high-throughput single MAC based implementation of adaptive FIR digital filters. It is shown that the hardware cost could be substantially reduced by time sharing multiplier architecture across single MAC core. Thismethod resulted in substantial area reduction compared to the conventional Adaptive FIR filter architectures for the FPGA implementation. The proposed structure of 16-tap Adaptive FIR filter for FPGA implementation supports up to 323 MHz input sampling frequency. Thus the efficient Adaptive FIR filter architectures achieve high speed, low complexity and the flexibility of FPGA technology making the architectures a viable alternative to the development of reconfigurable hardware for real time signal processing applications.

Table-2. Performance results of Adaptive FIR filter using XILINX VIRTES-5 (XC5VSX95T-1FF1136).

Performance measures	**Conventional Adaptive FIR filter with OPC**				**Proposed Adaptive FIR filter with OPC**			
No. of taps	2-tap	8-tap	16-tap	32-t ap	2-tap	8-tap	16-tap	32-tap
Number of slices	117	411	808	1586	142	149	220	341
Number of slice registers	168	624	1232	2448	247	224	352	608
Number of slice Luts	4848	19454	38946	78012	1281	1360	1488	1664
Minimum sampling period(ns)	4.125	8.360	11.697	16.817	2.081	3.075	3.096	3.118
Maximum sampling frequency(MHz)	242.432	119.622	85.489	59.465	480.558	325.223	323.043	320.718

Table-3. Performance results of adaptive FIR filter using XILINX VIRTES-5 (XC5VSX95 T-1FF1136).

Performance measures	**Conventional Adaptive FIR filter with parallel pipeline multiplier**				**Proposed Adaptive FIR filter with parallel pipeline multiplier**			
No. of taps	2-tap	8-tap	16-tap	32-tap	2-tap	8-tap	16-tap	32-tap
Number of slices	42	121	240	412	70	119	164	280
Number of slice registers	112	400	784	1552	233	210	338	594
Number of slice Luts	593	2336	4179	9225	197	284	374	577
Minimum sampling period(ns)	12.859	15.525	19.129	23.684	6.418	6.686	7.092	7.481
Maximum sampling frequency(MHZ)	77.766	64.412	52.276	42.222	155.823	149.575	141.002	133.672

REFERENCES

[1] J. G. Proakis and D. G. Manolakis. 1996. Digital Signal Processing: Principles, Algorithms and applications. Upper Saddle River, NJ: Prentice-Hall.

[2] G. Mirchandani, R. L. Zinser Jr. and J. B. Evans. 1995. “A new adaptive noise cancellation scheme in the presence of crosstalk [speech signals],”IEEE Trans. Circuits Syst. II, Analog. Digit. Signal Process. Vol. 39, No. 10, pp. 681–694, October.

[3] D. J. Allred, H. Yoo, V. Krishnan, W. Huang and D. V. Anderson. 2005. “LMS adaptive filters using distributed arithmetic for high throughput,” IEEE Trans. Circuits Systems I, Reg. Papers, vol. 52, no.7, pp. 1327–1337, July.

[4] D. Xu and J. Chiu. 1999. “Design of a high-order FIR digital filtering and variable gain ranging seismic data acquisition system,” in: Proc. IEEE Southeastcon’93, Apr. 1993, p. 6. K. K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation. New York: Wiley.

[5] Pramod Kumar Meher. 2010. “New Approach to Look-Up-Table Design and Memory-Based Realization of FIR Digital Filter,” IEEE Transactions on circuits and systems—irregular papers, Vol. 57, No. 3, March.

[6] H. H. Dam, A. Cantoni, K. L. Teo and S. Nordholm. 2007. “FIR variable digital filter with signed power-of-two coefficients,” IEEE Trans. Circuits Syst. I, Reg. Papers, Vol. 54, No. 6, pp. 1348–1357, June.

[7] Pramod Kumar Meher. 2008. "FPGA Realization of FIR Filters by Efficient and Flexible Systolization Using Distributed Arithmetic", IEEE Transactions on signal processing, Vol. 56, No. 7, July.

[8] P. K. Meher and S. Y. Park. 2011. "High-throughput pipelined realization of adaptive FIR filter based on distributed arithmetic," in: Proc. IEEE/IFIP 19th Int. Conf. VLSI-SOC, October. pp. 428–433.

[9] Sang Yoon park and promodkumarmeher. 2014. "Efficient FPGA and ASIC Realizations of a DA-based Reconfigurable FIR Digital Filter", IEEE Transactions on Circuits and systems-II: Express Briefs, Vol.61, No.7, July.

[10] Logi CORE IP FIR Compiler v5.0, Xilinx, Inc., San Jose, CA, USA, 20103.

[11] C. R. Baugh and B. A. Wooley. 1973. "A two's complement parallel array multiplication algorithm," IEEE Trans. Comput., Vol. C- 22, pp. 1045–1047, December.

[12] Asgar Abbaszadeh and Khostov D. Sadeghipour. 2011."A New Hardware Efficient Reconfiqurable FIR Filter architecture suitable for FPGA applications" proc IEEE DSP.

[13] J. Park, *et al*. 2004. "Computation Sharing Programmable FIR Filter for Low-Power and High-Performance Applications", IEEE J. Solid stateCir. Sys., Vol.39, No.2, pp.348-357, February.

[14] R. Mahesh, and A. PVinod. 2010. "New Reconfigurable Architectures for implementing FIR Filter with low complexity," IEEE Transactions on computer aided design of integrated circuits and systems, Vol. 29, February.

[15] S. A. White. 1989. "Applications of the distributed arithmetic to digital signal processing: Atutorial review," IEEE ASSP Mag., Vol. 6, No. 3, pp. 5–19, July.

[16] M. Mehendale, S. D. Sherlekar and G. Venkatesh. 1997. "Area-delay tradeoff in distributed arithmetic based implementation of FIR filters," in: Proc.10th Int. Conf. VLSI Design, January. pp. 124–129.

[17] H. Yoo and D. V. Anderson. 2005. "Hardware-efficient distributed arithmetic architecture for high-order digital filters," in: Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing, (ICASSP'05), Mar. Vol.5, pp. v/125–v/128.

[18] S.-S. Jeng, H.-C. Lin and S.-M. Chang. 2006. "FPGA implementation of FIR filter using M-bit parallel distributed arithmetic," in: Proc. 2006 IEEE Intl. Symp. Circuits Syst. ISCAS, May.

[19] Y. H. Chan and W. C. Siu. 1992. "On the realization of discrete cosine transform using the distributed arithmetic," IEEE Trans. Circuits Syst. I, Fundam. Theory Appl., Vol. 39, No. 9, pp. 705–712, September.

[20] H.-C. Chen, J.-I. Guo, T.-S. Chang and C.-W. Jen. 2005. "A memory-efficient realization of cyclic convolution and its application to discrete cosine transform," IEEE Trans. Circuits Syst. Video Technol., Vol. 15, No. 3, pp. 445–453, March.

[21] Pramod Kumar Meher, S. Chandrasekaran and A. Amira. 2008. "FPGA realization of FIR filters by efficient and flexible systolization using distributed arithmetic," IEEE Trans. Signal Process, Vol. 56, No. 7, pp. 3009–3017, July.

[22] Pramod Kumar Meher. 2006. "Unified systolic-like architecture for DCT and DST using distributed arithmetic," IEEE Trans. Circuits Syst. I, Reg. Papers, Vol.53, No. 5, pp. 2656–2663, December.

[23] J.-I. Guo, C.-M. Liu and C.-W. Jen. 1992. "The efficient memory-based VLSI array design for DFT and DCT," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., Vol. 39, No. 10, pp. 723–733, October.

[24] D. F. Chiper. 1999. "A systolic array algorithm for an efficient unified memory-based implementation of the inverse discrete cosine transform," in: IEEE Conf. Image Process., October. pp. 764–768.