International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 17 (2017) pp. 6462-6471

Optimized Inner Product Computation Architectures using DAOBC

Zainul Abdin Jaffery and Shaheen Khan[2],*

Department of Electrical Engineering, Jamia Millia Islamia, New Delhi, India.

[2]Orcid: 0000-0001-6139-3256

Abstract

This paper presents novel methods to design and implement a high performance inner product computation. The proposed methods use optimized architectures based on Distributed Arithmetic-Offset Binary Code (DAOBC) technique. In these architectures, the optimization is performed using pipelining, symmetry and bit-parallel techniques. In this brief, the 8^{th}-order fixed coefficient Finite Impulse Response (FIR) filters are implemented on Spartan-6 (xc6slx16-2csg324) Field Programmable Gate Array (FPGA) and their performance is measured in terms of dynamic power dissipation, number of Look-up Tables (LUTs) and maximum frequency of the circuit. The architectures: namely, 'With LUT' and 'Without LUT' are implemented in the optimized as well as in their un-optimized form for comparison purpose. Result analysis shows that the actual output values for the input sets are almost same as the simulated output values. It is observed that the optimized 'With LUT' architecture has power saving of 45.94% compared to its un-optimized case. For optimized 'Without LUT' architecture, the power saving is 42.76% compared to its un-optimized case.

Keywords: Distributed Arithmetic-Offset Binary Coding, Offset Binary Coding Look-up Table, Finite Impulse Response filter, Inner Product Computation, Field Programmable Gate Arrays

INTRODUCTION

The inner product forms the basis for many signal processing tools such as convolution, correlation, Discrete Fourier Transform and filters. Multiply and Accumulate (MAC) based technique is generally used for inner product computation. In this technique, the output is available after N clock cycles (where N is the order of inner product) [1]. Distributed Arithmetic (DA) is another widely used technique that does not require multiplication. Moreover, compared to MAC approach the DA technique requires the number of clock cycles to complete the inner product proportional to bit-length of input data instead of number of coefficients. Hence, the DA based technique is more efficient than MAC based technique for higher order of inner products [2].

In literature, several techniques have been developed for inner product computation using Field Programmable Gate Array (FPGA) technology [3]-[8], [10]-[14]. The most fitting technique is the Distributed Arithmetic (DA). The DA based technique uses Lookup Table (LUT) for the storage of pre-computed values. These pre-computed values can be accessed quickly according to the input data and hence, DA intensive algorithm avoids direct multiplication. This makes DA-based computation fast as well as most suitable for FPGA realization, as LUT also forms the basic components of FPGA logic resources [1], [4]. However, the size of LUT increases by 'power of two' with the filter order. Therefore, for the higher order filter, it becomes a very tedious job to design the filter using single large size LUT and accordingly the number of cycles to process the input will also increase. The two most common techniques to reduce the LUT size are: LUT partitioning (i.e. by using separate small LUT tables and combining their results) [1] and Offset Binary Coding (OBC) [3]. In OBC technique, the symbols '+1' and '-1' are used compared to '1' and '0' in binary representation. Moreover, this scheme can easily be implemented for further reduction of LUT size with suitable use of control circuitry and even LUT-less architecture may be obtained [5], [11].

Various methods are available in open literature for the design of FIR filters based on DA [3]-[7], [9]-[13]. The design optimization of FIR filter by systolic decomposition of DA was presented in [6]. DA has also been used for adaptive FIR filter architectures for high throughput [2], [7]. Using arithmetic representation, Residue Number System (RNS) is used to improve the performance in terms of speed and

resource occupation of FIR filters [8]. Recently in [9], the RNS based DA approach has been developed for simplification of the scaling accumulator using thermometer and one hot code representations. At architecture level, various architectures of FIR filters based on DA-OBC has been implemented; especially for the fixed coefficients [3], [5], [11], [13]. It is also observed from [4], [6] that optimization in architecture plays a key role for the performance of the filter.

In this paper, two optimized architectures based on DAOBC have been proposed for the inner product computation on Spartan-6 FPGA. The pipelining, symmetry and bit-parallel techniques are used to optimize the architectures. For comparative analysis, corresponding to the two existing architectures of fixed coefficient FIR filter using DAOBC scheme given in [7] and [5] have been considered.

This paper is organized as follows: Section 2 describes the Distributed Arithmetic Offset Binary Code (DAOBC) Algorithm. Section 3 describes the methodology for the proposed designs. Section 4 explains the proposed architectures for the 8th-order Low Pass FIR filter based on DAOBC algorithm. The simulation results and performance comparison on Spartan 6 and Application Specific Integrated Chip (ASIC) 28/32 nm library are given in Section 5. Finally, the conclusion and future scope of work is presented in Section 6.

DAOBC ALGORITHM

Consider the N[th] order 'sum of products' expression:

$$y(n) = \sum_{n=0}^{N-1} h(n)x(n) \quad (1)$$

Where h(n) are the known coefficient values, x(n) represent the corresponding input value and y(n) is the system output.

Let x(n) be represented in two's complement binary number having its magnitude less than 1, then x(n) is expressed as:

$$x(n) = -x_0(n) + \sum_{b=1}^{B-1} x_b(n).2^{-b} \quad (2)$$

Where, $x_b(n) \in \{0,1\}$, $x_0(n)$ represents sign bit and B is the word-length of x(n).

By using the value of x(n) from Equation (2) into Equation (1), the output y(n) can be expressed as:

$$y(n) = -\sum_{n=0}^{N-1} h(n)\,x_0(n) + \sum_{b=1}^{B-1}\left(2^{-b}\right)\sum_{n=0}^{N-1} x_b(n)h(n) \quad (3)$$

Equation (3) is used for signed DA realization. For this to realize, LUT may be prepared for the term $x_b(n)h(n)$ to reduce the computation complexity [1].

The LUT size can further be reduced by half using Offset Binary Code (OBC). In OBC, the input is not taken in straight binary code (i.e. 0 and 1) rather it is taken in Offset Binary Code (i.e. -1 and +1) [3]. In order to understand this, consider input x(n), which can be expressed as:

$$x(n) = \frac{1}{2}\left[x(n) - \left(-x(n)\right)\right] \quad (4)$$

Now, two's complement representation of the negative of x(n) is written as:

$$-x(n) = -x_0'(n) + \sum_{b=1}^{B-1} x_b'(n)2^{-b} + 2^{-(B-1)} \quad (5)$$

Where, the " ′ " symbol indicates the complement of a bit.

Using Equation (2) and Equation (5), Equation (4) may be rewritten as:

$$x(n) = \frac{1}{2} - \left[\left(x_0(n) - x_0'(n)\right) + \sum_{b=1}^{B-1}\left(x_b(n) - x_b'(n)\right)2^{-b} - 2^{-(B-1)}\right] \quad (6)$$

Now, following new variables are defined to simplify Equation (6):

$$\text{And} \quad \left.\begin{array}{ll} c_0(n) = -(x_0(n) - x_0'(n)) & \text{for} \quad b = 0 \\ c_b(n) = x_b(n) - x_b'(n) & \text{for} \quad b = 1 \text{ to } B-1 \end{array}\right\} \quad (7)$$

Where the possible values of $c_b(n)$ are +1 or -1.

Using the above substitution of $c_b(n)$ from Equation (7), Equation (6) may be rewritten as:

$$x(n) = \frac{1}{2}\left[\sum_{b=0}^{B-1} c_b(n)2^{-b} - 2^{-(B-1)}\right] \quad (8)$$

By substituting Equation (8) in to Equation (1), following equation is obtained:

$$y(n) = \frac{1}{2}\sum_{n=0}^{N-1} h(n)\left[\sum_{b=0}^{B-1} c_b(n)\,2^{-b} - 2^{-(B-1)}\right] \quad (9)$$

$$y(n) = \sum_{b=0}^{B-1} K_0(c_b(n))\,2^{-b} + K_1\,2^{-(B-1)} \quad (10)$$

Where

$$K_0\big(c_b(n)\big) = \frac{1}{2}\sum_{n=0}^{N-1} h(n)c_b(n) \quad (11)$$

And

$$K_1 = -\frac{1}{2}\sum_{n=0}^{N-1} h(n) \quad (12)$$

The implementation of the 'sum of products' term, $K_0(c_b(n))$ may be obtained in number of ways. H. Yoo and David V. Anderson [5] realized this term with and without the use of LUTs whereas Xie et al. [4] obtained this computation term realized with the use of ROMs. In our proposed architectures, the computation has been performed using the LUTs as well as without using LUTs. The 2^{-7} scaled value of constant K_1 (shifting K_1 to right by 7 position as in our case B=8) is then added to the appropriately scaled values (as per Equation (10)) of $K_0(c_b(n))$ in order to obtain the final output value 'y'.

METHODOLOGY

In the proposed methodology, some of the important parts of the designs are discussed in below given sub-sections.

Input and Shift Register operation

A continuous sinusoidal signal of amplitude value of 0.5 with frequency 1.0 MHz has been used to obtain five unique discrete sample values (i.e. 0.0000, 0.4755, 0.2939, -0.2939 and -0.4755). In order to obtain these sample values, the specified input signal is sampled using 5.0 MHz sampling frequency. These are represented in 2's complement representation using 8-bits (where Most Significant Bit (MSB) is the sign bit and rest seven bits to the right side of decimal are considered for magnitude). These sample values are further expressed in two hexadecimal digits: 00H, 3CH, 25H, DBH and C4H. The matrix representation of in terms of input series x1 and x2 are shown in Figure 1 and these are considered for 30 sample values for testing. For the architectures, discussed in Section 4, the four consecutive sample values of inputs shown in Figure 1 are used to generate the 4-bit DA addresses for each input series which further cover the four unique coefficient values. The schema depicted through Figure 1 is shown for FIR filter of N=8, that is why it will need two input series x1 (for one set of four coefficients) and x2 (for another set of four coefficients) to cover the total eight coefficient values.

The proposed methods utilize the symmetry property of linear phase FIR filter. To take this property in to account the Input Sequence is obtained by adding the two separate series x1 and x2. This Input Sequence is fed to the eight bit input register. This processing is depicted in Figure 2 for the First Set of Input Sequence of Figure 1. The Shift Register accepts the data in each clock cycle and shifts it to one position right results in generation of the four bit data after four cycles as shown in Figure 2. This Shift Register Data forms the Distributed Arithmetic (DA) address. In this way, for the eight bits of input, after four cycles, total eight 4-bit DA addresses get generated simultaneously which forms the basis for computation.

$$X1=\begin{bmatrix}00 & 3C & 25 & DB\\ 25 & 3C & 00 & C4\end{bmatrix}\begin{bmatrix}3C & 25 & DB & C4\\ DB & 25 & 3C & 00\end{bmatrix}\begin{bmatrix}25 & DB & C4 & 00\\ C4 & DB & 25 & 3C\end{bmatrix}$$
$$X2=\quad \text{First Set} \qquad \text{Second Set} \qquad \text{Third Set}$$

$$X1=\begin{bmatrix}DB & C4 & 00 & 3C\\ 00 & C4 & DB & 25\end{bmatrix}\begin{bmatrix}C4 & 00 & 3C & 25\\ 3C & 00 & C4 & DB\end{bmatrix}$$
$$X2=\quad \text{Fourth Set} \qquad \text{Fifth Set} \quad (13)$$

Figure 1: Matrix representation of the Input Series x1 and x2

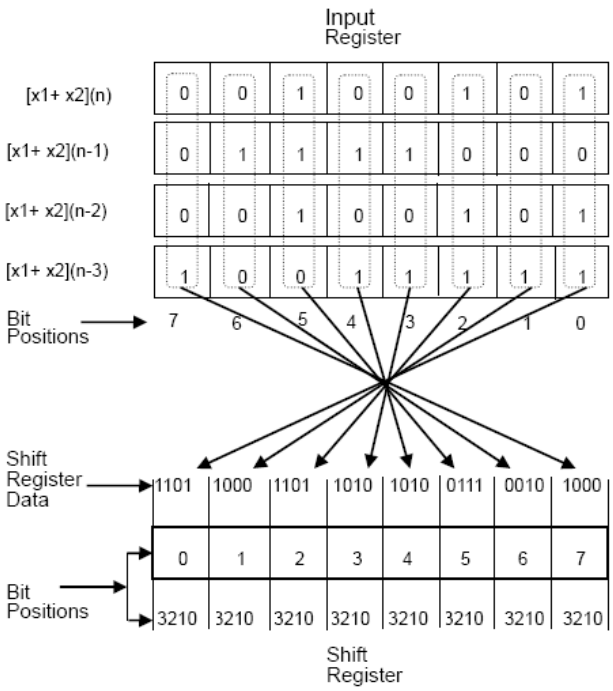

Figure 2: Input registers and the Shift Register

LUT Reduction

Using the DA-OBC algorithm mentioned in Section 2, the size of the LUT for the proposed 'With LUT' architecture is reduced to half as given in Table 1[7]. The 3-bit DA-OBC addresses (i.e. $x_b(2)$, $x_b(1)$ and $x_b(0)$) which are used to access the OBC LUT Contents are obtained from 4-bit DA addresses.

Table 1: Reduced LUT using OBC for N=4

$x_b(2)$	$x_b(1)$	$x_b(0)$	OBC LUT Contents
0	0	0	1/2[h(3)-h(2)-h(1)-h(0)]
0	0	1	1/2[h(3)-h(2)-h(1)+h(0)]
0	1	0	1/2[h(3)-h(2)+h(1)-h(0)]
0	1	1	1/2[h(3)-h(2)+h(1)+h(0)]
1	0	0	1/2[h(3)+h(2)-h(1)-h(0)]
1	0	1	1/2[h(3)+h(2)-h(1)+h(0)]
1	1	0	1/2[h(3)+h(2)+h(1)-h(0)]
1	1	1	1/2[h(3)+h(2)+h(1)+h(0)]

Optimization

The proposed architecture has been optimized for performance in terms of key parameters like resource usage, power dissipation and speed. The optimization strategies have been described in the following sub-sections.

Symmetry

For the case of linear phase FIR filter, the coefficients have been extracted from MATLAB version 2013 using the specification:

Window method: Rectangular,

Order of the filter: 8,

Sampling frequency (f_s): 6.0 MHz,

Cutoff frequency (f_c): 1.75 MHz,

Coefficient word-length: 24 bits (where MSB is the sign bit and rest 23 bits to the right side of decimal are considered for magnitude).

The coefficients obeys the following symmetry:

$$h(n) = h(N - n - 1) \tag{14}$$

In our case, N=8 which may be represented as:

$$y(n) = \sum_{n=0}^{\frac{N}{2}-1} h(n)[x(n) + x(N - n - 1)] \tag{15}$$

Equation (15) represents the reduction of the computational resources of proposed architecture by a factor of two. This also explains the reason why the Input Sequence is obtained by the addition of two separate series x1 and x2. This evidently results in the improvement of resource usage and power dissipation.

Pipelining

In the proposed architecture, this technique is used to divide large arithmetic operation into small arithmetic operations. This feature is incorporated in the 'Pipeline register and right shift-adder tree' block of the proposed architecture. This results in power reduction and the speed of the circuit is also increased appreciably [4], [12].

Bit-Parallel

In the proposed architectures, all the bits of the Input Sequence are processed in each clock cycle. This is depicted through Figure 2 and this technique is favored for fast processing of inputs [1].

PROPOSED ARCHITECTURES

In this section, the two proposed architectures based on DAOBC are presented. Figure 3 shows the block diagram of proposed 8th-order 'With LUT' architecture which uses LUTs for inner product computation. As discussed in sub-section 3.1, after four clock cycles, the 8-bit Input Sequence is applied to Right Shift Register and it generates eight 4-bit DA addresses. These eight 4-bit DA addresses from the Right Shift Register are further customized to get eight 3-bit DA-OBC addresses from the respective eight 'DA-OBC Address Encoding Logic' circuits. Here, each of the 'DA-OBC Address Encoding Logic' circuit is designed using the three XOR gates and a NOT gate [7]. These eight DA-OBC addresses are further used to access the corresponding pre-computed values (i.e. $K0(c_b(n))$) from respective eight OBC LUTs. As per the algorithm discussed in Section 2, the values of the OBC LUTs are accessed without sign change if the MSB of the DA addresses is '1' and these values are considered as negative if the MSB of the DA addresses is '0'. In each clock cycle, these accessed values will be fully pipelined, appropriately shifted and accumulated in the 'Pipeline Register and right shift-adder tree' block. Further, in the same block, the corresponding 2^{-7} (i.e. seven times right shifted) scaled value of K1 is also added to get the final output value 'y'.

In this architecture, the OBC LUTs are designed for size 2^3x24 for the corresponding 3-bit DA-OBC addresses for four coefficient values i.e. h(3), h(2), h(1) and h(0) as per Table 1. Therefore, in our case, for the 8-bit input, the proposed architecture uses total eight OBC LUTs for computation.

For its corresponding un-optimized architecture [7], the input series x1 and x2 are separately processed to obtain the eight 3-bit DA-OBC addresses from the respective eight 'DA-OBC Address Encoding Logic' circuits. Based on the eight bits of x1, separate eight OBC LUTs are required for the coefficient values (h(3), h(2), h(1) and h(0)). While for x2, another eight OBC

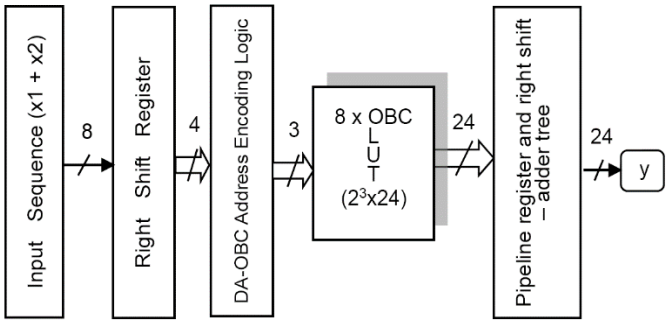

Figure 3: The proposed optimized 8th-order 'With LUT' architecture

LUTs are used for another coefficient values (h(7), h(6), h(5) and h(4)). Hence, its un-optimized architecture uses double the architectural resources compared to its optimized case.

To consolidate our findings, another architecture for the same order of FIR filter using the same parameters and optimization has been implemented on the same FPGA device. Rest all the design parameters like coefficient values, device parameters with Placement and Route settings are considered. This is referred to as 'Without LUT' (as shown in Figure 4) architecture because it does not use LUTs for computation. Here, each of the eight 'DA-OBC Address Encoding Logic' circuits use three xor gates [5] and it is used to encode the 4-bit DA address into 3-bit DA-OBC addresses. Based on these DA-OBC addresses, the computation is performed using the circuits that use additions/subtractions as explained in Figure 4. The division of coefficient values by two is performed by shifting them right by two. The Intermediate computed values based on DA-OBC addresses are used without sign change if the MSB of the DA addresses is '0' and these values are considered as negative if the MSB of the DA addresses is '1'. Rest of the functionality is the same as that of architecture discussed in Figure 3.

For its corresponding un-optimized architecture [5], the input series x1 and x2 are separately processed the same way as in earlier un-optimized case. Based on the eight bits of x1, separate eight computation circuits using adders/subtractors are required for the coefficient values (h(3), h(2), h(1) and h(0)). While for x2, separate computation circuits are used for another coefficient values (h(7), h(6), h(5) and h(4)). Hence, in this case also, the un-optimized architecture uses double the architectural resources compared to its optimized case.

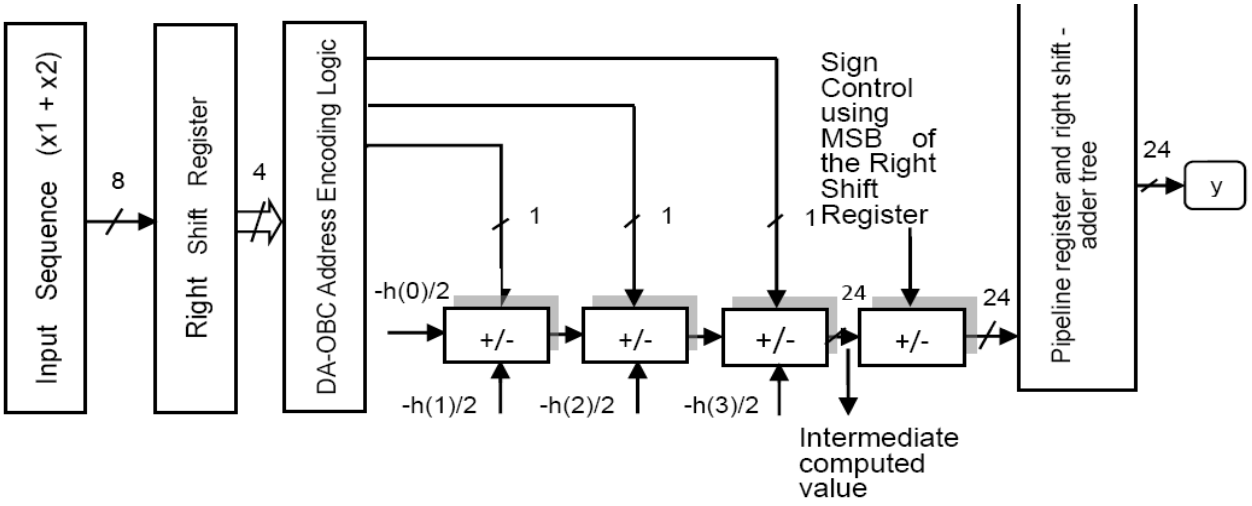

Figure 4: The proposed optimized 8th -order 'Without LUT' architecture

SIMULATION AND RESULTS

The input in the form of Input Sets (as explained in Sub-section 3.1) are applied using a test bench created corresponding to the VHDL code of the filter. The clock frequency used for the design is 5.0 MHz.The first correct output appears at the 6th clock cycle and thereafter sequences of outputs are obtained based on the values of Input Sequence applied. The designed filter meets the design goal with respect to the specified sinusoidal signal.

For all the architectures discussed in Section 4, the designs are represented using respective VHDL source files. The source files are synthesized using XST and simulation of design is performed using ISIM which is shown in Figure 5. It may be observed that simulation output values of Figure 5 are consistent with the actual output values given in Table 2.

Table 2: Comparison between actual versus simulated output for the proposed architectures

Input Sequence	Actual Output	Simulated output	Error (%)
First Input	0.00000	0.00000	0.0000
Second Input	-0.47592	-0.47417	0.3677
Third Input	-0.29413	-0.29299	0.3875
Fourth Input	0.29413	0.29299	0.3875
Fifth Input	0.47592	0.47417	0.3677

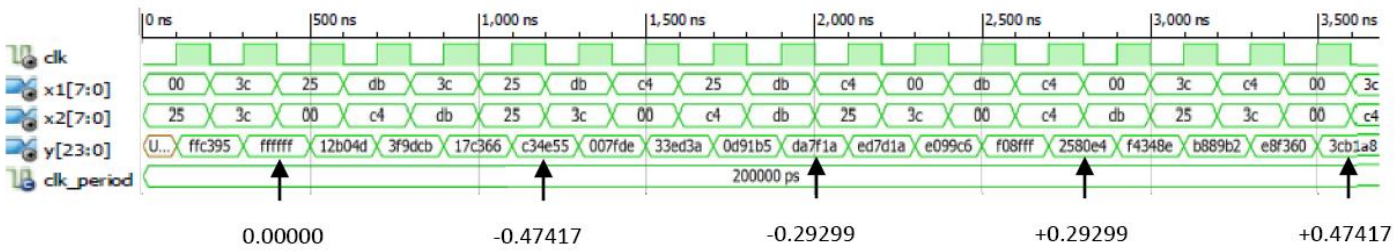

Figure 5: Simulation timing diagram

Total On-chip power of the circuit is comprised of Static and Dynamic power. Our primary focus is on the Dynamic power of the circuit as static power is almost constant for all the architectures for a particular order on a FPGA device. The dynamic on-chip power can further be subdivided in to Clock, Logic, Signals and I/Os (Input/Outputs) power. Dynamic power depends on the architecture design and is influenced by a number of factors such as selection of FPGA device, clock frequency of implementation, switching activity rates, interconnections of FPGA, resource usage, power supply voltage level used for specific FPGA device [4], [6]. The power is calculated using XPower Analyzer tool of ISE Design Suite version 14.5.

All the architectures have been designed for 8^{th}-order Low Pass FIR filters and these have been implemented on Spartan 6 (device: xc6slx16-2csg324) FPGA. Spartan 6 provides leading system integration capabilities with the lowest total cost for high-volume applications. It is built on a mature 45 nm low power copper process technology that delivers the optimal balance of cost, power, and performance. It incorporates efficient 6-input LUTs to minimize power and improve performance. This variant of Spartan 6 FPGA has 14,579 logic cells, 2,278 total slices, 9,112 Number of 6-input LUTs, 136 Kb Max Distributed RAM and 232 number of I/Os [15].

The performance evaluation of all the architectures are made based on the results obtained in terms of key performance metrics for resource usage in terms of Number of 6-input LUTs, Speed in terms of Maximum Frequency (MHz) and Power dissipation in terms of Dynamic Power (mW). Table 3 tabulates the performance comparison of the two proposed optimized architectures with their un-optimized cases.

From Table 3, it is observed that the proposed optimized 'With LUT' architecture has power saving of 45.94% compared to its un-optimized case. While the proposed optimized 'Without LUT' architecture has power saving of 42.76 % compared to its un-optimized case.

Figure 6 illustrates the percentage of the resource saving for proposed architecture compared to the existing architectures on different FPGAs. It is observed from Table 3 and Figure 6 that the proposed architecture for the specified filter gives better performance than their respective un-optimized architectures in terms of considered key parameters. The proposed architectures incorporate optimization and this is the reason why the proposed architectures outperform their respective un-optimized cases.

Table 3: Performance comparison on Spartan 6 FPGA

Parameter	With LUT		Without LUT	
	Un-Optimized [7]	Optimized	Un-Optimized [5]	Optimized
Number of 6-input LUTs	978	519	1368	710
Maximum Frequency (MHz)	76.799	117.8	75.301	121.315
Dynamic Power (mW)	15.8	8.54	16.18	9.26

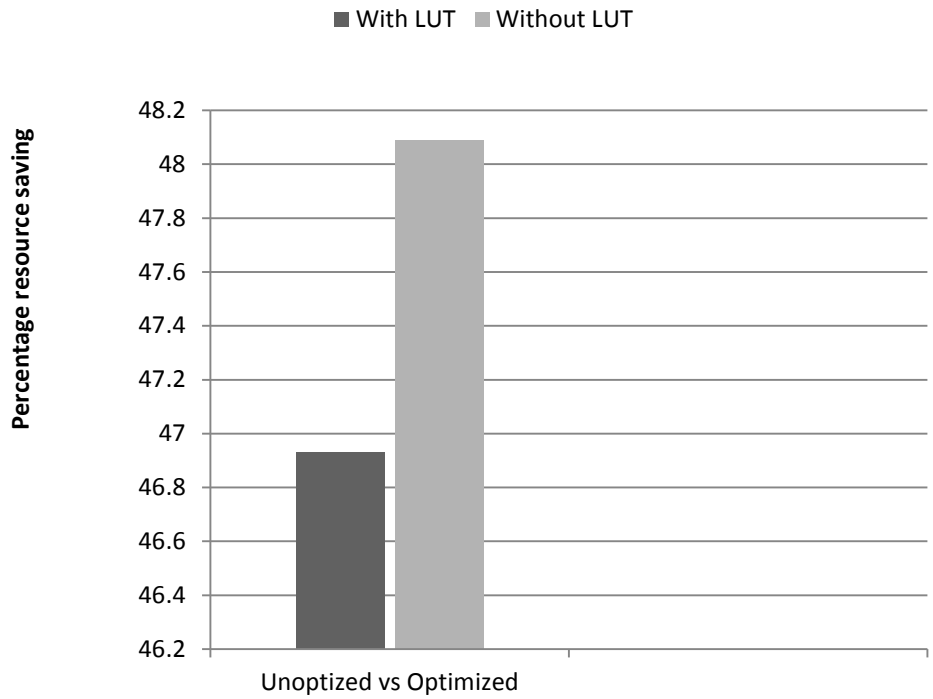

Figure 6: Percentage resource saving in terms of Number of 6-input LUTs for proposed optimized architecture compared to the un-optimized architectures

To validate the results on ASIC, all the architectures are also implemented on 28/32 nm library saed32hvt_ff0p85v25c. The results are obtained using Design Vision tool of Synopsys in terms of area, DAT (Data Arrival Time) and power dissipation using 10.00 MHz clock and 0.85 V operating voltage. Here the results are consistent apart from the optimized forms appears to be slower than their un-optimized forms amounting due to different technologies.

Table 3: Performance comparison on ASIC 28/32 nm library

	With LUT		Without LUT	
Parameter	Un-Optimized [7]	Optimized	Un-Optimized [5]	Optimized
Area (μm^2)	9795.42	5407.36	14441.04	7011.43
DAT (ns)	2.32	2.67	2.32	2.61
Power Dissipation (μW)	255.45	139.37	325.24	160.16

CONCLUSION

In this paper, two optimized architectures based on DAOBC has been presented for the design and implementation of 8^{th}-order low pass FIR filter on spartan 6 (xc6slx16-2csg324) FPGA using ISE Design Suite 14.5. These two architectures are optimized for various performance parameters i.e. power dissipation, speed and resource usage using pipelining, symmetry and bit-parallel techniques. The two respective existing un-optimized architectures have also been used for the performance comparison. Apart from substantial speed and resource usage advantages for both the optimized architectures, the proposed 'With LUT' architecture has power saving of 45.94% compared to its un-optimized case. While the proposed 'Without LUT' architecture has power saving of 42.76 % compared to its un-optimized case. In practical situations, as per different requirements, the proposed architecture can further be conveniently used for implementation of even higher order inner product computation for any other type of filters on different FPGAs and ASICs.

REFERENCES

[1] Baese, U. M., 2007, "Digital Signal processing with Field Programmable Gate Arrays," 3rd Ed. New Delhi, India: Springer-Verlag Berlin Heidelberg, Springer (India) Pvt. Ltd.

[2] Allred, D. J., Yoo, H., Krishnan, V., Huang, and Anderson, D. V., 2005, "LMS adaptive filters using distributed arithmetic for high throughput," IEEE Transactions on Circuits and Systems-I," 52(7), pp. 1327–1337.

[3] White, S. A., 1989, "Applications of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review, " IEEE ASSP Magazine, 6(3), pp. 4-19.

[4] Xie, J., He, J., and Tan, G., 2010, "FPGA realization of FIR filters for high-speed and medium-speed by using modified distributed arithmetic architectures," Microelectronics Journal (Elsevier), 41, pp. 365-370.

[5] Yoo, H., and Anderson, D. V., 2005, "Hardware-efficient Distributed Arithmetic architecture for high-order digital filters," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), p. v/125 - v/128.

[6] Meher, P. K., Chandrasekaran, S., and Amira, A., 2008, "FPGA Realization of FIR Filters by Efficient and Flexible Systolization Using Distributed Arithmetic," IEEE Transactions on Signal Processing, 56(7), pp. 3009-3017.

[7] Prakash, M. S., and Shaik, R. A., 2013,"Low-Area and High-Throughput Architecture for an Adaptive Filter Using Distributed Arithmetic," IEEE Transactions on Circuits and Systems-II, 60(11), pp. 781-785.

[8] Pontarelli, S., Cardarilli, G. C., Re, M., and Salsano, A., 2012,"Optimized Implementation of RNS FIR Filters Based on FPGAs," J Sign Process Syst (Springer), 67, pp. 201–212.

[9] Vun, C. H., Premkumar, and A. B., Zhang, W, 2013,"A New RNS Based DA Approach for Inner Product Computation," IEEE Transactions on Circuits and Systems-I, 60(8), pp. 2139-2152.

[10] Eshtawie, M. A. M., and Othman, M., 2006,"On-Line DA-LUT Architecture for High-Speed High-Order Digital FIR Filters," Proceedings of the IEEE International Conference on Communication Systems (ICCS), Singapore, pp. 1-5.

[11] Choi, J., Shin, S., and Chung, J., 2000,"Efficient ROM size reduction for distributed arithmetic," In Proceedings of the IEEE ISCAS, Geneva, Switzerland, 2, pp. 61–64.

[12] Khan, S., and Jaffery, Z.A., 2015, "Low Power FIR Filter Implementation on FPGA using Parallel Distributed Arithmetic," IEEE International Conference, IDICON, N. Delhi, India, pp.1-5.

[13] Hong, B., Yin, H., Wang, X., and Xiao, Y., 2010,"Implementation of FIR Filter on FPGA Using DAOBC Algorithm," Information Science and Engineering, ICISE, pp. 3761-3764.

[14] Aktan, M., Yurdakul, A., and Dundar, G., 2008,"An algorithm for the design of low-power hardware-efficient FIR filters," IEEE Trans. Circuits Syst- I, 55, pp. 1536–1545.

[15] Spartan-6 FPGA Configurable Logic Block user guide [Internet]. Xilinx, Inc., 2100 Logic Drive, San Jose, CA 95124-3400. Available from: http://www.xilinx.com.