International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 22 (2018) pp. 15679-15687
© Research India Publications. http://www.ripublication.com

A High-speed Inner Product Computation Architecture based on DA-OBC

Zainul Abdin Jaffery, Shaheen Khan*

Departement of Electrical Engineering, Jamia Millia Islamia,

Jamia Nagar, Delhi 110025, India.

Abstract

In this paper, the proposed method use an optimized architecture based on Distributed Arithmetic-Offset Binary Code (DA-OBC) technique. In this architecture, combination of both Look-up Tables (LUTs) and multiplexers (MUXs) are used for computational purposes. The validity of the proposed method is demonstrated by implementing a 16th-order inner product in the form of Finite Impulse Response (FIR) filter on different FPGAs. The performance results for number of LUTs, maximum frequency and dynamic power consumption are consistent across different FPGA devices. It is observed that the proposed architecture has dynamic power savings of 7.51% and 13.50% compared to existing With LUT and Without LUT architectures respectively on Spartan 6. While on Spartan 3A, the proposed architecture has dynamic power savings of 8.83% and 15.28% compared to With LUT and Without LUT architectures respectively. Moreover, the proposed architecture has appreciable resource savings advantage compared to the existing architectures.

Keyword: DSP, DA-OBC, OBC-LUT, FPGA, Inner Product Computation

1. INTRODUCTION

The inner product forms the basis for many signal processing tools such as convolution, correlation, Discrete Fourier Transform and filters. Digital filters are considered one of the main components of Digital Signal Processing (DSP) [1]. Earlier, 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=order of inner products) [2]. Distributed Arithmetic (DA) is another popular technique that avoids 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, for higher order inner products, the DA based technique is more efficient than MAC [2], [3].

In literature, several architectures have been proposed for inner product computation using Field Programmable Gate Arrays (FPGAs) [4-14]. The DA based technique use look-up tables (LUTs) for the storage of pre-computed values. These pre-computed values can be accessed quickly according to the input data and hence, DA based algorithm produces faster results. Moreover, FPGA realization is easy since LUT forms the basic components of FPGA logic resources [5]. However, the size of LUT increases exponentially with the order of inner products [2], [6]. To overcome this problem, two most common techniques are used: LUT partitioning [6] and Offset Binary Coding (OBC) [4], [11], 15]. In LUT partitioning, the partial tables are used and their results are added. In its basic form, DA in association with OBC technique reduces the LUT size by factor of two where the symbols '+1' and '-1' are used instead of binary '1' and '0' [4]. 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 [15, 16].

In this paper, a new architecture based on Distributed Arithmetic Offset Binary Code (DA-OBC) has been proposed for the high-speed inner product computation. Existing DA based architecture for fixed coefficient FIR filter using OBC scheme given in [11] and another 'LUT-less DA-OBC' architecture given in [15] are considered for comparison analysis. These two comparative architectures use 'DA-OBC Address Encoding Logic' circuitry while the proposed architecture avoids it to provide improved performance. To observe the consistency in results, all the architectures have been synthesized and implemented on Spartan 6 and Spartan 3A FPGAs. The pipelining, symmetry and bit-parallel techniques are used to optimize all these architectures.

2. DAOBC AlGORITHM

In order to understand the DA-OBC concept, consider the Nth-order inner product expression:

$$y_n = \sum_{n=0}^{N-1} h_n x_n \tag{1}$$

Here y_n represents the output, h_n are known coefficients and x_n is the input variable.

Let the x_n be represented in 2's complement number having its magnitude less than 1:

$$x_n = -x_{0n} + \sum_{b=1}^{B-1} x_{bn}.2^{-b} \tag{2}$$

* ***Corresponding Author:*** *Shaheen Khan, Research Scholar, Departement of Electrical Engineering, Jamia Millia Islamia, Jamia Nagar, Delhi 110025, India. Email: shaheen.khan.2@gmail.com*

Where, $x_{bn} \epsilon \{0,1\}$, x_{0n} represents sign bit and B is the bit-length of x_n.

By using the value of x_n from Eq. (2) into Eq. (1), y_n is expressed as:

$$y_n = -\sum_{n=0}^{N-1} h_n x_{0n} + \sum_{b=1}^{B-1} (2^{-b}) \sum_{n=0}^{N-1} x_{bn} h_n \qquad (3)$$

Eq. (3) is used for signed DA realization. For this to realize, LUT may be prepared for the term $\sum_{n=0}^{N-1} x_{bn} h_n$ to reduce the computation complexity [4].

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) [4]. To understand this, x_n can be expressed as:

$$x_n = \frac{1}{2} \left(x_n - (-x_n)\right) \qquad (4)$$

Now, 2's complement representation of the $-x_n$ is given as:

$$-x_n = -\bar{x}_{0n} + \sum_{b=1}^{B-1} \bar{x}_{bn} 2^{-b} + 2^{-(B-1)} \qquad (5)$$

Where the bar symbol indicates the complement of a bit.

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

$$x_n = \frac{1}{2} \left(-(x_{0n} - \bar{x}_{0n}) + \sum_{b=1}^{B-1} (x_{bn} - \bar{x}_{bn}) 2^{-b} - 2^{-(B-1)} \right) \qquad (6)$$

Now, the two new variables c_{0n} and c_{bn} are defined for simplification:

$$\left.\begin{aligned} &c_{0n} = -(x_{0n} - \bar{x}_{0n}) \qquad \text{for } b = 0 \\ &\text{And} \\ &c_{bn} = (x_{bn} - \bar{x}_{bn}) \qquad \text{for } b = 1 \text{ to } B-1 \end{aligned}\right\} \qquad (7)$$

Where, the possible values of c_{bn} are +1 or -1.

Using the substitution of c_{bn} from Eq. (7), Eq. (6) may be rewritten as:

$$x_n = \frac{1}{2} \left(\sum_{b=0}^{B-1} c_{bn} 2^{-b} - 2^{-(B-1)} \right) \qquad (8)$$

By substituting Eq. (8) into Eq. (1), following equation is obtained:

$$y_n = \frac{1}{2} \sum_{n=0}^{N-1} h_n \left(\sum_{b=0}^{B-1} c_{bn} 2^{-b} - 2^{-(B-1)} \right) \qquad (9)$$

$$y_n = \sum_{b=0}^{B-1} K_0(c_{bn}) 2^{-b} + K_1 2^{-(B-1)} \qquad (10)$$

Where

$$K_0(c_{bn}) = \frac{1}{2} \sum_{n=0}^{N-1} h_n c_{bn} \qquad (11)$$

And

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

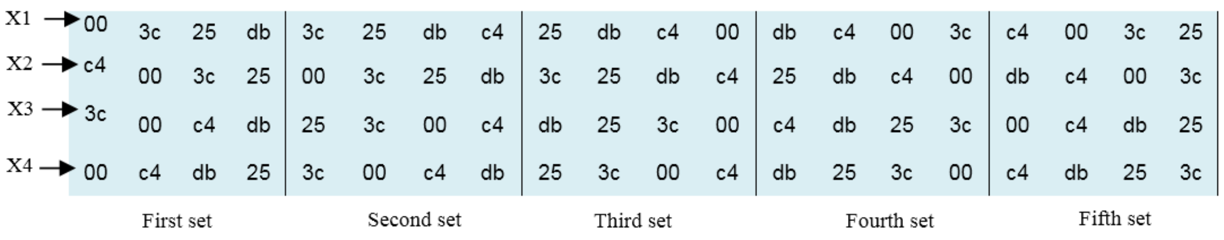
The term $K_0(c_{bn})$ as shown in Eq. (11), is implemented in different ways. H. Yoo and D. V. Anderson realized this term without LUT i.e. using adders/subtractors [15]. While in [11], the authors realized the same term using LUTs. In the proposed architecture, this computation has been performed using the combination of LUTs and MUXs. In the bit-parallel approach adopted in proposed architecture (discussed in sub-section 3.3.3), the $K_0(c_{bn})$ values from N LUTs are accessed simultaneously based on the DA addresses during each clock cycle and then scaled appropriately by 'power of two'. Also, as shown in Eq. (10), the scaled value of $K_0(c_{bn})$ is added to the one time scaled value of K_1 by $2^{-(B-1)}$ to obtain the final output, y_n.

3. METHODOLOGY

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

3.1. Input and Shift Register operation

Five unique discrete sample values (i.e. 0.0000, 0.4755, 0.2939, -0.2939 and -0.4755) of input series x1 are obtained from a continuous sinusoidal signal of 0.5 amplitude value with frequency 1.0 MHz. To obtain these sample values, this sinusoidal signal is sampled using 5.0 MHz sampling frequency. The sample values are represented in 2's complement using 8-bits (where Most Significant Bit (MSB) is the sign bit) which are further expressed in two hexadecimal digits: 00H, 3CH, 25H, DBH and C4H. Input schema in terms of input series x1, x2, x3 and x4 are shown in Fig. 1 and 30 sample values are considered for testing. For the architectures of N=16 (discussed in Section 4), the four consecutive input sample values of x1, x2, x3 and x4 are referred to as First set, Second set up to Fifth set values as shown in Fig. 1. These sets are used to generate the four address lines which further cover the four unique coefficient values of upper and lower half sections separately. To take the symmetry property of linear phase FIR filter into account, the First Input Sequence is obtained by adding respective values of series x1 and x4 and similarly the Second Input Sequence is obtained by addition of x2 and x3. A very small truncation error which appears in this addition may be neglected. The input processing is shown in Fig. 2 for the First Input Sequence (i.e. x1+x4) for Second set values of Fig. 1. The Shift Register is described by two-dimensional (size: 8x4) arrays in VHDL. It accepts the data from Input Register and shift the data to the right in each cycle; thereby results in generation of total eight 4-bit DA addresses after every four clock cycles (as shown in Fig. 2). These DA addresses are used to access the data from respective computational resources (i.e. LUTs and MUXs). The analogous parallel processing will follow for the Second Input Sequence (i.e. x2+x3).



	First set				Second set				Third set				Fourth set				Fifth set			
X1 →	00	3c	25	db	3c	25	db	c4	25	db	c4	00	db	c4	00	3c	c4	00	3c	25
X2 →	c4	00	3c	25	00	3c	25	db	3c	25	db	c4	25	db	c4	00	db	c4	00	3c
X3 →	3c	00	c4	db	25	3c	00	c4	db	25	3c	00	c4	db	25	3c	00	c4	db	25
X4 →	00	c4	db	25	3c	00	c4	db	25	3c	00	c4	db	25	3c	00	c4	db	25	3c

Fig. 1. Input sets

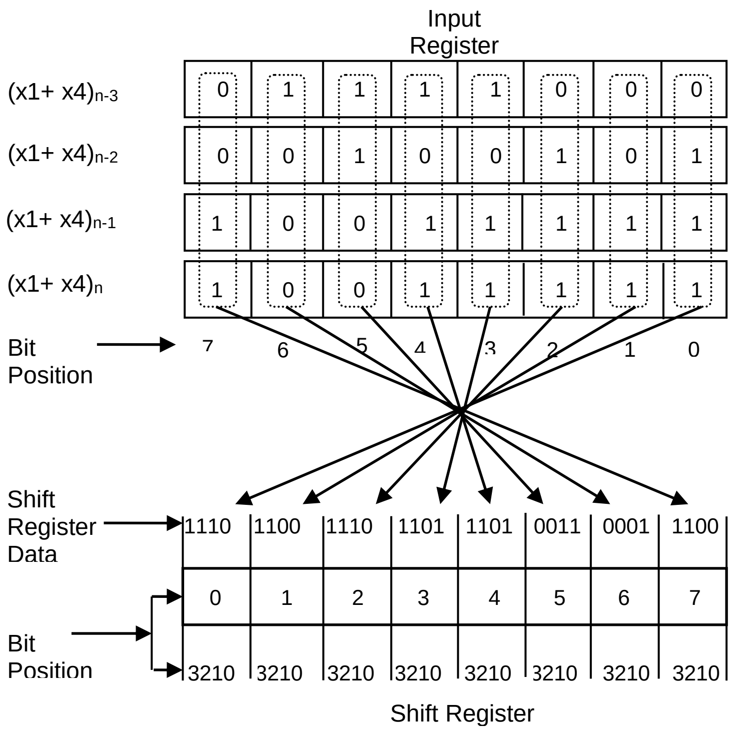


Fig. 2. Input processing

3.2. LUT Reduction

In [15], 2:1 MUXs are repeatedly used to reduce the size of each LUT by 'power of two'. However, in the proposed design for N=16, the size of each LUT is reduced to one fourth by the use of single 4:1 MUX. This is achieved by considering only the first four rows referred to as Basic OBC LUT (BOL) of Table 1. The upper two bits of DA address, x_{b3} and x_{b2} are used to select the coefficient values [i.e. 0, h_2, h_3 and (h_2+h_3)] through 4:1 MUX. As shown in Table 1, the second four rows may be realized by adding coefficient value h_2 to each contents of the BOL.

Similarly, the third and last four rows may be realized from the BOL by adding h_3 and (h_2+h_3) values respectively to its contents. It is evident that the size of BOL is $2^2 \times 24$ (where 24=bit-length of the coefficients). The lower two bits of DA address, x_{b1} and x_{b0} are used to address the contents of BOL. The basic 4-input LUT using OBC is given in Table 1 which may be divided into four parts and realized as addition of the results of BOL and 4:1 MUX. The same parallel processing will occur for all the DA addresses.

Table 1. Basic 4-input LUT using OBC

x_{b3}	x_{b2}	x_{b1}	x_{b0}	OBC LUT contents	
0	0	0	0	$-1/2(+h_3+h_2+h_1+h_0)$	Basic OBC LUT (BOL)
0	0	0	1	$-1/2(+h_3+h_2+h_1-h_0)$	
0	0	1	0	$-1/2(+h_3+h_2-h_1+h_0)$	
0	0	1	1	$-1/2(+h_3+h_2-h_1-h_0)$	
0	1	0	0	$-1/2(+h_3-h_2+h_1+h_0)$	BOL+h_2
0	1	0	1	$-1/2(+h_3-h_2+h_1-h_0)$	
0	1	1	0	$-1/2(+h_3-h_2-h_1+h_0)$	
0	1	1	1	$-1/2(+h_3-h_2-h_1-h_0)$	
1	0	0	0	$-1/2(-h_3+h_2+h_1+h_0)$	BOL+h_3
1	0	0	1	$-1/2(-h_3+h_2+h_1-h_0)$	
1	0	1	0	$-1/2(-h_3+h_2-h_1+h_0)$	
1	0	1	1	$-1/2(-h_3+h_2-h_1-h_0)$	
1	1	0	0	$-1/2(-h_3-h_2+h_1+h_0)$	BOL+(h_2+h_3)
1	1	0	1	$-1/2(-h_3-h_2+h_1-h_0)$	
1	1	1	0	$-1/2(-h_3-h_2-h_1+h_0)$	
1	1	1	1	$-1/2(-h_3-h_2-h_1-h_0)$	

3.3. Optimization

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

3.3.1. Symmetry

For the 16th-order inner product (i.e. N=16), the coefficient values have been extracted using MATLAB ver. 2013 for the linear phase FIR filter. Hence, the coefficients obey the following symmetry:

$$h_n = h_{N-n-1} \tag{13}$$

By using Eq. (13), y_n is represented using equation:

$$y_n = \sum_{n=0}^{\frac{N}{2}-1} h_n(x_n + x_{N-n-1}) \tag{14}$$

Eq. (14) represents the reduction of the computational resources of proposed architecture by a factor of two. This also explains the reason why the First Input Sequence is obtained by the addition of two separate series x1 and x4. Similarly, the Second Input Sequence is obtained by the addition of x2 and x3. This evidently results in the improvement of resource usage and power dissipation.

3.3.2. Pipelining

In the proposed architecture, this technique is used to divide large arithmetic operation into smaller parts. This feature is incorporated in the ‘Pipeline registers and right shift-adder tree’ block of the proposed architecture. It results in power reduction and the maximum frequency of the circuit is also increased appreciably [4], [5], [6].

3.3.3. Bit-Parallel

In the proposed architecture, all the bits (i.e. here both the input sequences) are processed simultaneously in each clock cycle. This is depicted through Fig. 2 and this technique is favored for fast processing of inputs (i.e. high-speed operation) [5], [19].

4. PROPOSED ARCHITECTURE

The proposed architecture is shown in Fig. 3. Here, the upper half section uses the First Input Sequence and it is designed for the first four coefficient values i.e. h_3, h_2, h_1 and h_0. In this section, for the bit-parallel operation discussed above, input bit-length=8, the corresponding eight Basic OBC LUTs (BOL1s) and eight 4:1 MUXs are used as computational resources. Each BOL1 of size $2^2 \times 24$ are designed as per the concept depicted in Table 1. The 4:1 MUXs are used for selecting one out of the four coefficient values from 0, h_2, h_3 and (h_2+h_3). Likewise, for the lower half section in Fig. 3, the Second Input Sequence is used and it is designed for the other four coefficient values i.e. h_7, h_6, h_5 and h_4. Here also, for input bit-length of 8, the eight BOL2s and eight 4:1 MUXs are used. The 4:1 MUXs selects one of the four coefficient values i.e. 0, h_6, h_7 and (h_6+h_7).

Therefore, the proposed architecture uses total computation hardware as sixteen BOLs (size: $2^2 \times 24$) and sixteen 4:1 MUXs. Based on the address generated from Right Shift Register, in each clock cycle, the pre-computed values will be accessed from the computational resources. The impact of the respective basic 4-input LUT based on DA-OBC (shown in Table 1) is obtained by adding the results of BOLs and MUXs in the ‘Pipeline registers and right shift-adder tree’ blocks separately. In the same blocks, the pipelining feature and corresponding one time ‘power of two’ scaled value of K_1 is also incorporated. The purpose of the Adder is to add the results of the upper and lower half sections, which ultimately provides the final output ‘y’.

To make a justified comparison, the two existing architectures have also been implemented for the same order (i.e. N=16) of FIR filter using the same design and device considerations with exactly same level of optimization as the proposed architecture. First existing architecture for fixed coefficients using DA-OBC scheme given in [11] has been referred here as With LUT architecture. As it uses LUTs, it is named as With LUT architecture as shown in Fig. 4. Depending on its 8-bits input, eight OBC LUT1s (2^3x24) [11] are generated to store pre-computed values for one set of four coefficients (i.e. h_3, h_2, h_1 and h_0). An analogous parallel lower half-section of Fig. 4 has been designed for the next four coefficient values (i.e. h_7, h_6, h_5 and h_4) in form of OBC LUT2s. The DA address from the Right Shift Register is further customized to get DA-OBC address from the DA-OBC Address Encoding Logic. It is realized using the three XOR gates and a NOT gate [11]. In each clock cycle, the DA-OBC Addresses are used to access the pre-computed values stored in OBC LUTs.

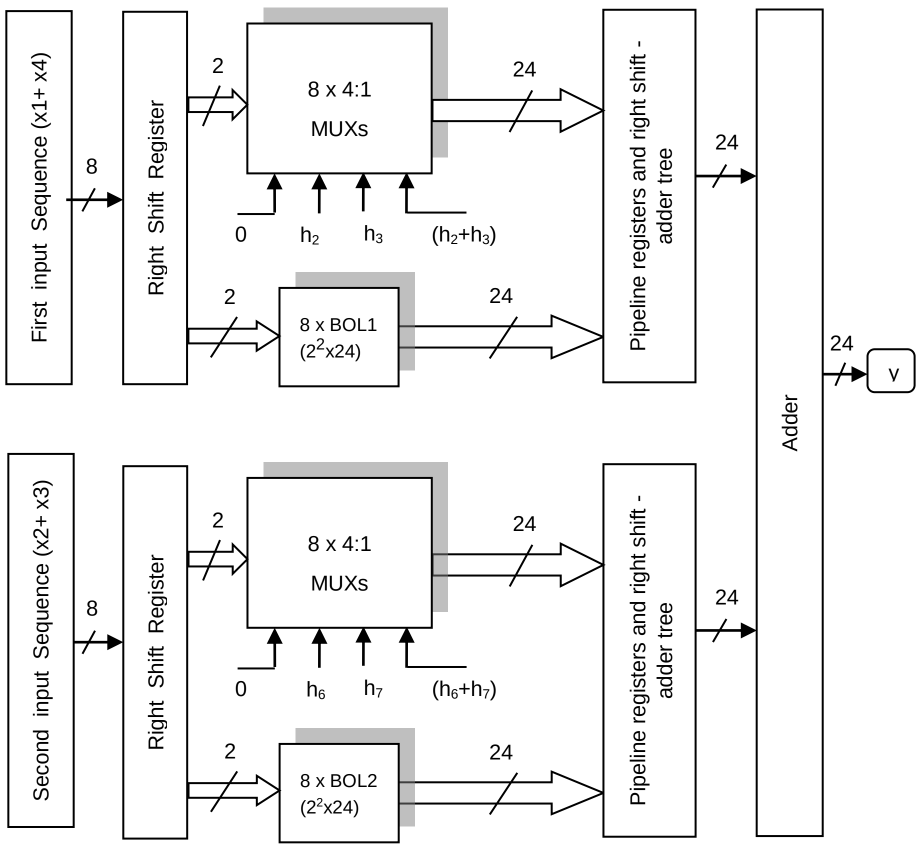

First input Sequence (x1+ x4)
8
Right Shift Register
2
8 x 4:1
MUXs
0
h2
h3
(h2+h3)
24
2
8 x BOL1
(2^2x24)
24
Pipeline registers and right shift - adder tree
24
Adder
24
y
Second input Sequence (x2+ x3)
8
Right Shift Register
2
8 x 4:1
MUXs
0
h6
h7
(h6+h7)
24
2
8 x BOL2
(2^2x24)
24
Pipeline registers and right shift - adder tree
24

Fig. 3. The proposed high-speed architecture for N=16

Second existing LUT-less DA-OBC architecture given in [15] has been referred here as Without LUT architecture is shown in Fig. 5. In this case, the computational resources involve adders/subtractors without using LUTs. Here, the DA-OBC Address Encoding Logic is designed using three XOR gates only [15]. Rest of the design and functionality is the same as that of With LUT architecture.

5. SIMULATION AND RESULTS

The coefficients for the designs are obtained by FDA Tool of MATLAB ver. 2013 using the specifications: Window method: Rectangular, Order of the filter: 16, Sampling frequency (f_s): 6.0 MHz, Cutoff frequency (f_c): 1.75 MHz, Coefficient bit-length: 24 bits. For all the architectures discussed in Section 4, the designs are represented using respective VHDL source files. These design entries are synthesized using XST tool and simulations of the designs are performed using ISIM tool. Both of these tools are integrated in ISE Design Suite version 14.5. The First and Second Input Sequences are applied using a test bench created for VHDL code of the architecture. The clock frequency used for all the architecture designs is 5.0 MHz and the supply voltage is 1.2 V for both the FPGA devices. It is clear from the timing simulation diagram shown in Fig. 6 that first correct and stable output appears at the 6th clock cycle and thereafter sequence of output is obtained based on the various sets of inputs described in Figure 1. The reset signal is used to remove the intermediate unintended output values. The architectures meet the design goal as for a particular set, the calculated output given in Table 2 matches with its simulated value shown in Figure 6 with very small error.

Total On-chip power dissipation 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 and verified from Tables 3a and b for all the architectures on a particular FPGA device. Dynamic power depends on the architecture design and is influenced by a number of factors such as selection of FPGA device, clock frequency used, switching activity rates, interconnections of FPGA, resource usage, power supply voltage level used for the device [5], [13]. The power is calculated using XPower Analyzer tool inbuilt in ISE Design Suite.

All the three architectures 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 [20]. From the Spartan 6 performance comparison of Table 3a, it is observed that the proposed architecture has power savings of 7.51% compared to With LUT architecture and 13.50% compared to Without LUT architecture.

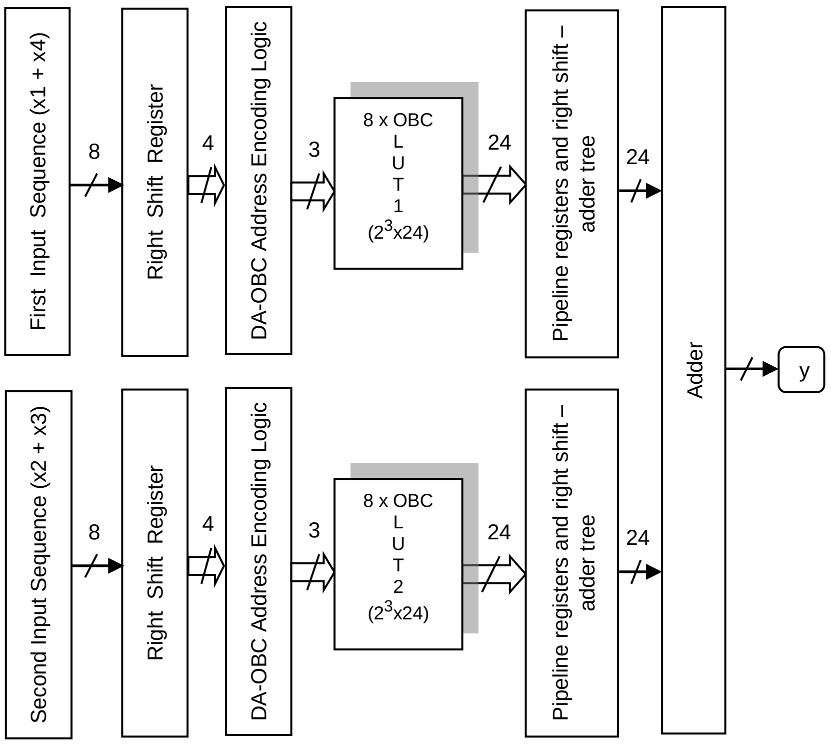

First Input Sequence (x1 + x4)
8
Right Shift Register
4
DA-OBC Address Encoding Logic
3
8 x OBC LUT 1 (2^3x24)
24
Pipeline registers and right shift – adder tree
24
Second Input Sequence (x2 + x3)
8
Right Shift Register
4
DA-OBC Address Encoding Logic
3
8 x OBC LUT 2 (2^3x24)
24
Pipeline registers and right shift – adder tree
24
Adder
y

Fig. 4. With LUT Architecture

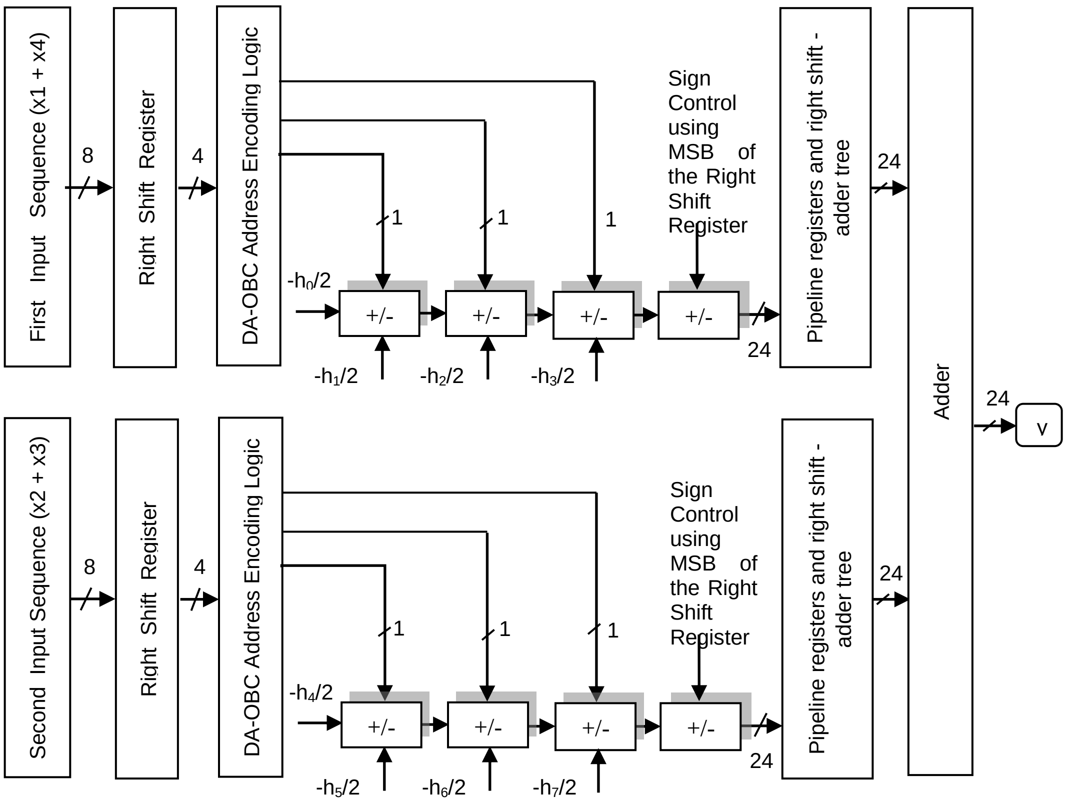

First Input Sequence (x1 + x4)
8
Right Shift Register
4
DA-OBC Address Encoding Logic
1
1
1
Sign Control using MSB of the Right Shift Register
-h0/2
+/-
+/-
+/-
+/-
-h1/2
-h2/2
-h3/2
24
Pipeline registers and right shift - adder tree
24
Second Input Sequence (x2 + x3)
8
Right Shift Register
4
DA-OBC Address Encoding Logic
1
1
1
Sign Control using MSB of the Right Shift Register
-h4/2
+/-
+/-
+/-
+/-
-h5/2
-h6/2
-h7/2
24
Pipeline registers and right shift - adder tree
24
Adder
24
v

Fig. 5. Without LUT architecture

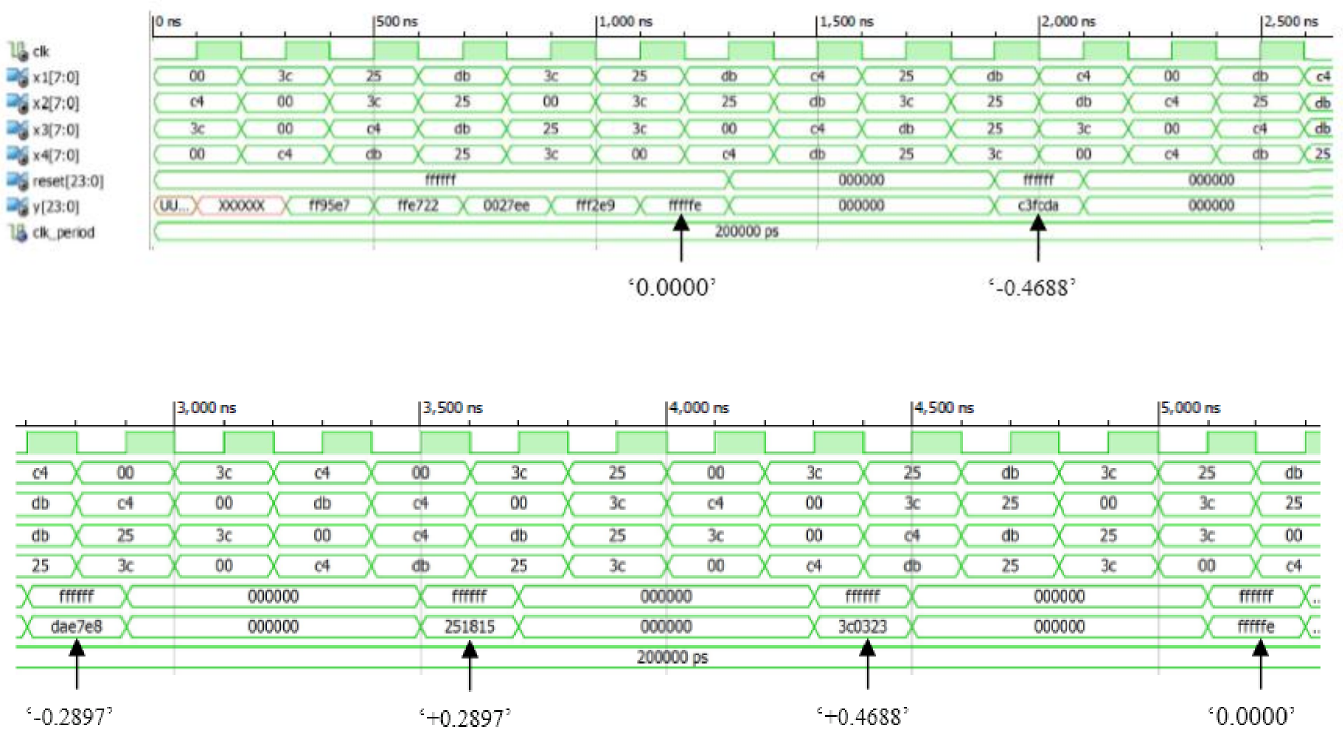


Fig. 6. Timing Simulation Diagram

Table 2 Comparison between Calculated versus simulated output for the proposed architecture

Input	Calculated	Simulated	Error (%)
First set	0	0	0
Second set	-0.47592	-0.4688	1.496
Third set	-0.29413	-0.2897	1.506
Fourth set	+0.29413	+0.2897	1.506
Fifth set	+0.47592	+0.4688	1.496

Next, all the three architectures have also been implemented on Automotive Spartan 3A (device: xa3s200a-4ftg256) FPGA. The Xilinx Automotive (XA) Spartan-3A families of FPGAs have been designed to meet the challenges of high-volume, cost-sensitive and I/O-intensive automotive electronics applications [21]. From Spartan 3A performance comparison given in Table 3b, it is observed that the proposed architecture has power savings of 8.83% compared to With LUT architecture while 15.28% compared to Without LUT architecture. Graph of Fig. 6 illustrates the percentage of the resource savings for proposed method compared to the existing methods on different FPGAs.

The proposed architecture has resource savings of 5.99% and 45.92% compared to With LUT and Without LUT architectures respectively on Spartan 6. While on Spartan 3A, the resource savings of proposed architecture compared to With LUT and Without LUT architectures are 12.14% and 45.33% respectively.

It is observed from Tables 3a, b and Fig. 7 that the proposed architecture gives better performance than the two existing architectures for all the parameters and the results are consistent for both the FPGA devices. The Without LUT architecture outperforms With LUT architecture only in term of maximum frequency otherwise for the remaining two parameters the With LUT architecture performs better than the Without LUT architecture. The proposed architecture outperforms the existing architectures because it does not require the DA-OBC Address Encoding Logic which avoids logic complexity and requires lesser resources that lead to better performance in terms of all the key parameters.

Table 3a. Performance comparison of various architectures on Spartan 6

Spartan 6 (xc6slx16-2csg324) FPGA

Parameter		Proposed	With LUT [11]	Without LUT [15]
Number of 6-input LUTs		643	684	1189
Maximum Frequency (MHz)		96.984	88.613	93.449
Power consumption (mW)	Static	20	20.01	20.02
	Dynamic	10.96	11.85	12.67

Table 3b. Performance comparison of various architectures on Spartan 3A

Spartan 3A (xa3s200a-4ftg256) FPGA

Parameter		Proposed	With LUT [11]	Without LUT [15]
Number of 4-input LUTs		861	980	1575
Maximum Frequency (MHz)		59.474	57.244	57.428
Power consumption (mW)	Static	16.82	16.83	16.84
	Dynamic	23.84	26.15	28.14

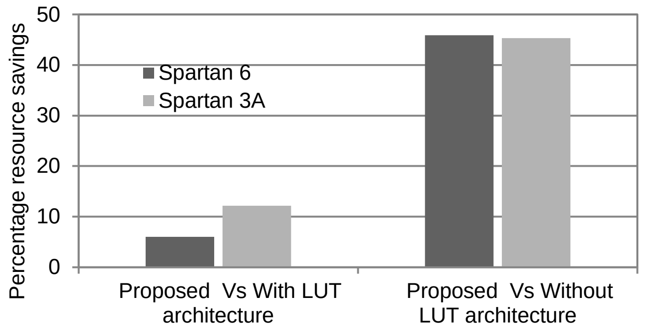


Fig. 7. Percentage resource savings of proposed versus existing architectures

6. CONCLUSION

In this paper, a proposed architecture based on DA-OBC is presented for the design and implementation of 16th-order inner product in the form of FIR filter on different FPGAs. In the proposed architecture, combination of both Look-up Tables (LUTs) and Multiplexers (MUXs) are used for computational purpose and it avoids the use of DA-OBC Address Encoding Logic circuit. The two existing architectures based on DA-OBC using the same parameters and optimizations; are also presented for performance comparison. It is observed that the proposed architecture not only has appreciable power and maximum frequency advantages but also has considerable resource savings compared to With LUT and Without LUT architectures on Spartan 6 and Spartan 3A FPGAs and the results are consistent across FPGA devices. In practical situations, as per different requirements, the proposed architecture can further be conveniently used for implementation of even higher order FIR filters and other type of FIR filters.

CONCLUSION

[1] K. Francis Akingbade, I. A. jewale Alimi, “Separation of Digital Audio Signals using Least-Mean-Square (LMS) Adaptive Algorithm”, *International Journal of Electrical and Computer Engineering*, vol. 4, no. 4, pp. 557–560, 2014.

[2] D. J. Allred et al. “LMS adaptive filters using distributed arithmetic for high throughput”, *IEEE Transactions on Circuits and Systems-I*, vol. 52, no. 7, pp. 1327–1337, 2005.

[3] Chan Hua Vun, Annamalai Benjamin Premkumar, Wei Zhang, "A New RNS Based DA Approach for Inner Product Computation", *IEEE Transactions on Circuits and Systems-I*, vol. 60, no. 8, pp. 2139-2152, 2013.

[4] S. A. White, "Applications of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review", *IEEE ASSP Magazine*, vol. 6, no. 3, pp. 4-19, 1989.

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

[6] Shaheen Khan, Z.A. Jaffery, "Low Power FIR Filter implementation on FPGA using Parallel Distributed Arithmetic", *IEEE International Conference, IDICON*, N. Delhi, pp. 1-5, 17-20 Dec, 2015.

[7] K.N. Macpherson, R.W. Stewart, "Area efficient FIR filters for high speed FPGA implementation", *IEE Proceedings-Vision, Image and Signal Processing*, vol. 153, no. 6, pp. 711-720, 2006.

[8] Gurpadam Singh, Neelam R. Prakash, "FPGA Implementation of Higher Order FIR Filter", accepted in *International Journal of Electrical and Computer Engineering*, vol. 7, no. 4, pp. 19-28, 2017.

[9] Y. C. Lim et al., "FPGA implementation of digital filters synthesized using the FRM technique", *Circuits Systems Signal Processing*, vol. 22, no. 2, pp. 211–218, 2003.

[10] W. Sen, T. Bin, Zhu Jun, "Distributed Arithmetic for FIR Filter Design on FPGA, *International Conference on Communications, Circuits and Systems, ICCCAS*, 2007, pp. 620–623, 11-13 July, 2007.

[11] M. Surya Prakash and Rafi Ahamed Shaik, "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, 2013.

[12] Bo Hong, Haibin Yin, Xiumin Wang, et al., "Implementation of FIR Filter on FPGA Using DAOBC Algorithm", *Information Science and Engineering, ICISE*, pp. 3761-3764, 2010.

[13] P. K. Meher, S. Chandrasekaran, A. Amira, "FPGA Realization of FIR Filters by Efficient and Flexible Systolization Using Distributed Arithmetic", *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3009-3017, 2008.

[14] P. K. Meher, "Hardware-efficient systolization of DA-based calculation of finite digital convolution of finite digital convolution", *IEEE Transactions on Circuit and Systems-II*, vol. 53, no. 8, pp. 707–711, 2006.

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

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

[17] Y. Jang, and S. Yang, "Low-power CSD linear phase FIR filter structure using vertical common sub-expression", *Electronic letters*, vol. 38, no. 15, pp. 777–779, 2002.

[18] T. D. Memon, Paul Beckett, A. Z. Sadik, "Power-Area-Performance Characteristics of FPGA-based Sigma-Delta FIR Filters' *Journal of Signal Processing Systems*, vol. 70, no. 3, pp. 275–288, 2013.

[19] K. K. Parhi, "VLSI Digital Signal Processing Systems: Design and Implementation", *John Wiley & Sons*, Inc, New York. 1999.

[20] "Spartan-6 FPGA Configurable Logic Block user guide", *http://www.xilinx.com* accessed 10 December 2016

[21] "XA Spartan-3A Automotive FPGA Family Data Sheet" *http://www.xilinx.com* accessed 24 December 2016