Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 08, August-2016

# Optimized Design Implementation of Direct Memory-Based Hardware for Efficient Resource-Constraint Digital Signal Processing Systems

Nandita Jaiswal
M.Tech (VLSI),
Department of Electronics & Communication Engineering
Hindustan College of Science & Technology
Farah, Mathura, India

Soumitra Sarkar
M.Tech Project Coordinator,
Department of Electronics & Communication Engineering
Hindustan College of Science & Technology
Farah, Mathura, India

***Abstract*— An advance approach for Direct-Memory-Based hardware for area-delay-power efficient systems for commonly encountered computation-intensive cores of digital signal processing (DSP) systems is presented by combining three techniques, where the memory-size is reduced to one-eighth at the cost of some increase in combinational circuit complexity for signed magnitude numbers. Each of these techniques results in the reduction of the memory size by a factor of two. It is shown that by efficiently combining sign-bit exclusion technique, a different form of anti-symmetric product coding (APC) and a modified odd-multiple-storage (OMS) scheme, we get an optimized direct-memory-based multiplication hardware for resource-constraint DSP systems which provides a reduction in memory size to one-eighth over conventional direct-memory-based hardware, at the cost of a marginal area overhead. The proposed design for small input sizes can be used for efficient implementation of high-precision multiplication by input operand decomposition. The proposed optimized design also offers almost 87.5% and 85% reductions in direct- memory size for L=5 bits and L=6 bits signed-magnitude numbers respectively, over conventional direct-memory size.**

***Keywords*— Digital signal processing (DSP), direct-memory-based computing, very large scale integration (VLSI).***

## I. INTRODUCTION

Rapid advancement in very large scale integration (VLSI) technology and hardware performance of digital devices have paved way to efficient memory-based computing systems as alternative to the conventional logic-only computing in order to meet the stringent constraint and growing requirements of the digital signal processing (DSP) systems in different application environments. Since DSP is considered as the major component of the digital revolution that is currently taking place around the world, it is therefore, important to design dedicated VLSI chips for fast and efficient computation of the DSP applications. It is observed that algorithms optimized for software-implementation, in general, are not well-suited for dedicated hardware-implementation. Appropriate algorithm design has a major role on developing a hardware entity that can meet the system requirements and specification. Not only it should necessarily lead to reduction of computational complexity, but also should facilitate maximization of concurrency by exploiting the possible parallelism to achieve high-throughput performance. Moreover, the architecture should be developed synergetic with the underlying algorithms to derive a cost effective and area-time-power efficient optimal VLSI. Memory-based designs consequently are gaining substantial popularity in the DSP application space. DSP algorithms involve multipliers that not only consume most of the resources of the system but also involve most of the computation-time. Significant researches have, therefore, been made in the past two decades for efficient multiplier less implementation of DSP systems.

Most of the DSP algorithms involve repetitive multiply accumulate operations and inner-product computation. Besides, very often the multiplying coefficients (e.g., filter coefficients or transform kernel coefficients) remain constant during the DSP operations. This behavior of DSP algorithms is utilized to realize the memory-based computing systems. There are two basic variants of memory-based computing techniques found to be popularly used. One of the techniques is the direct memory-based implementation of multiplications [1], while the second is based on distributed arithmetic (DA) [2]. The DA principle is used primarily to compute the inner-products by repeated shift-add operations of partial products corresponding to the successive bit-vectors of one of the input vectors. Whereas in the direct-memory-based implementations, the multiplications of input values with the fixed coefficients are performed by a look-up-table (LUT), where each of the LUTs contains the pre-computed product values for all possible values of input samples.

Apart from that, memory-based computing structures are more regular than the multiply–accumulate structures and offer many other advantages, e.g., greater potential for high-throughput and low-latency implementation and less dynamic power consumption [11]. However, we find that now increasing efforts are made to carry any sort of significant work on optimization for memory-based multiplication.

The rest of the paper follows as: in the next section, we have discussed comparison study of techniques used in direct-memory-based computation. In Section III, the proposed optimized technique used in direct-memory-based multiplication are discussed which are sign-bit exclusion, the APC and the modified OMS technique. The Section IV explains to use optimized direct-memory-based multiplication for signed and unsigned operands. The algorithmic design for proposed techniques is given in Section V and the optimized hardware implementation of direct-memory-based multiplier is presented in Section VI. The results of the proposed multiplier along with the conclusion are presented in Section VII.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 08, August-2016

## II. COMPARISON STUDY OF TECHNIQUES USED IN DIRECT-MEMORY-BASED COMPUTATION

Direct-Memory-based computing is well suited for many DSP algorithms, which involve multiplication with a fixed set of coefficients. In the conventional direct-memory-based implementations, the multiplications of input value B with the fixed coefficients A are performed by a look-up-table (LUT) of size $2^L$, (L is the word-length of B) where each value of the LUT contains the pre-computed product values P=A·B for all possible values of input samples. The product word $A \cdot B_i$ is stored at the location $B_i$ for $0 \leq B_i \leq 2^L - 1$, such that if an $L$-bit binary value of $B_i$ is used as the address for the LUT, then the corresponding product value is available as its output. Several architectures have been reported in the literature for memory-based implementation of DSP algorithms involving orthogonal transforms and digital filters [1]–[8].

In [9], odd multiples of the fixed coefficient is required to be stored, which is referred to as the odd-multiple-storage (OMS) scheme. Using OMS approach, one can reduce the LUT size to half, but it has significant combinational overhead since it requires a barrel-shifter along with a control-circuit to generate the control-bits for producing a maximum of (L − 1) left-shifts, and an encoder to map the L-bit input word to (L − 1)- bit LUT address.

In [10], there is *anti symmetric product coding* (APC) approach, in this the LUT size can also be reduced to half, where the product words are recoded as anti symmetric pairs. The APC approach, although providing a reduction in LUT size by a factor of two, incorporates substantial overhead of area and time to perform the two's complement operation of LUT output for sign modification and that of the input operand for input mapping.

However, we find that when the APC approach is combined with the OMS technique, the two's complement operations could be very much simplified since the input address and LUT output could always be transformed into odd integers. However, the OMS technique in [9] cannot be combined with the APC scheme in [10], since the APC words generated according to [10] are odd numbers. Moreover, the OMS scheme in [9] does not provide an efficient implementation when combined with the APC technique. In [11], a combined APC-OMS scheme reduces the LUT size to one-fourth. But further optimization can also be achieved, with our proposed scheme that combines three techniques which reduces the LUT size to one-eight. We therefore present three schemes for optimization of LUT with lower area-and time-overhead. One of the proposed optimization is based on exclusion of sign-bit from the LUT address, and the other two optimization is based on a coding of stored product word, where a different form of APC and combined that with a modified form of the OMS scheme for efficient memory based multiplication are presented.

## III. PROPOSED OPTIMIZED TECHNIQUE EMPLOYED IN DIRECT- MEMORY-BASED MULTIPLICATION

The optimization of product values stored could easily be performed for unsigned as well as signed magnitudes numbers. Besides, numbers could be fractions or integers in fixed-point format. But, for simplicity we assume here the multiplicand B to be an integer in sign-magnitude representation, while the constant A is assumed to be either in sign-magnitude or in 2's complements representation. We present here the proposed sign-bit exclusion scheme, the APC technique and its further optimization by combining it with a modified form of OMS.

### *A. Direct-Memory-Based Sign-Bit Exclusion Technique:*

As the name suggest in this technique the sign-bit of multiplier and multiplicand is excluded, and the product values stored is P=|A|·|B|, where |A| is magnitude-part of A and |B| is magnitude-part of B. The signed-bit which is the most significant bit (MSB) of A and B are XOR operated and the result of XOR operation is concatenated with P to get the true result of multiplication. Since |B| is an (L − 1)-bit binary number, all possible product values of |A|·|B| can be stored as $2^{(L-1)}$ LUT words which reduces its size to half.

The product words required to be stored for different values of B for direct-memory-based multiplication for L = 6 is shown in Table I. The product word corresponding to B = (1 $b_4$ $b_3$ $b_2$ $b_1$ $b_0$) is negative of that for B = (0 $b_4$ $b_3$ $b_2$ $b_1$ $b_0$) for any given value of |B| = ($b_4$ $b_3$ $b_2$ $b_1$ $b_0$). Therefore, product words on the fourth column can be derived by negating the product word stored at the second column on the same row. Therefore, instead of 64 product words only 32 values of |A| · |B| for all possible values of |B| are required to be stored, as shown in the sixth column. The technique requires only one additional XOR gate to determine the sign of product word P.

TABLE I. DIRECT-MEMORY-BASED SIGN-BIT EXCLUSION TECHNIQUE FOR L=6

| Input, B address, \|B\| | product values | Input, B address, \|B\| | product values | stored words | |
|---|---|---|---|---|---|
| | | | | *2's comp* | *sign-magnitude* |
| 0 0 0 0 0 0 | 0 | 1 0 0 0 0 0 | 0 | 0 | 0 |
| 0 0 0 0 0 1 | *A* | 1 0 0 0 0 1 | −*A* | *A* | \|*A*\| |
| 0 0 0 0 1 0 | 2*A* | 1 0 0 0 1 0 | −2*A* | 2*A* | 2\|*A*\| |
| 0 0 0 0 1 1 | 3*A* | 1 0 0 0 1 1 | −3*A* | 3*A* | 3\|*A*\| |
| 0 0 0 1 0 0 | 4*A* | 1 0 0 1 0 0 | −4*A* | 4*A* | 4\|*A*\| |
| 0 0 0 1 0 1 | 5*A* | 1 0 0 1 0 1 | −5*A* | 5*A* | 5\|*A*\| |
| 0 0 0 1 1 0 | 6*A* | 1 0 0 1 1 0 | −6*A* | 6*A* | 6\|*A*\| |
| 0 0 0 1 1 1 | 7*A* | 1 0 0 1 1 1 | −7*A* | 7*A* | 7\|*A*\| |
| 0 0 1 0 0 0 | 8*A* | 1 0 1 0 0 0 | −8*A* | 8*A* | 8\|*A*\| |
| 0 0 1 0 0 1 | 9*A* | 1 0 1 0 0 1 | −9*A* | 9*A* | 9\|*A*\| |
| 0 0 1 0 1 0 | 10*A* | 1 0 1 0 1 0 | −10*A* | 10*A* | 10\|*A*\| |
| 0 0 1 0 1 1 | 11*A* | 1 0 1 0 1 1 | −11*A* | 11*A* | 11\|*A*\| |
| 0 0 1 1 0 0 | 12*A* | 1 0 1 1 0 0 | −12*A* | 12*A* | 12\|*A*\| |
| 0 0 1 1 0 1 | 13*A* | 1 0 1 1 0 1 | −13*A* | 13*A* | 13\|*A*\| |
| 0 0 1 1 1 0 | 14*A* | 1 0 1 1 1 0 | −14*A* | 14*A* | 14\|*A*\| |
| 0 0 1 1 1 1 | 15*A* | 1 0 1 1 1 1 | −15*A* | 15*A* | 15\|*A*\| |
| 0 1 0 0 0 0 | *16A* | 1 1 0 0 0 0 | −16*A* | 16*A* | 16\|*A*\| |
| 0 1 0 0 0 1 | 17*A* | 1 1 0 0 0 1 | −17*A* | 17*A* | 17\|*A*\| |
| 0 1 0 0 1 0 | 18*A* | 1 1 0 0 1 0 | −18*A* | 18*A* | 18\|*A*\| |
| 0 1 0 0 1 1 | 19*A* | 1 1 0 0 1 1 | −19*A* | 19*A* | 19\|*A*\| |
| 0 1 0 1 0 0 | 20*A* | 1 1 0 1 0 0 | −20*A* | 20*A* | 20\|*A*\| |
| 0 1 0 1 0 1 | 21*A* | 1 1 0 1 0 1 | −21*A* | 21*A* | 21\|*A*\| |
| 0 1 0 1 1 0 | 22*A* | 1 1 0 1 1 0 | −22*A* | 22*A* | 22\|*A*\| |
| 0 1 0 1 1 1 | 23*A* | 1 1 0 1 1 1 | −23*A* | 23*A* | 23\|*A*\| |
| 0 1 1 0 0 0 | 24*A* | 1 1 1 0 0 0 | −24*A* | 24*A* | 24\|*A*\| |
| 0 1 1 0 0 1 | 25*A* | 1 1 1 0 0 1 | −25*A* | 25*A* | 25\|*A*\| |
| 0 1 1 0 1 0 | 26*A* | 1 1 1 0 1 0 | −26*A* | 26*A* | 26\|*A*\| |
| 0 1 1 0 1 1 | 27*A* | 1 1 1 0 1 1 | −27*A* | 27*A* | 27\|*A*\| |
| 0 1 1 1 0 0 | 28*A* | 1 1 1 1 0 0 | −28*A* | 28*A* | 28\|*A*\| |
| 0 1 1 1 0 1 | 29*A* | 1 1 1 1 0 1 | −29*A* | 29*A* | 29\|*A*\| |
| 0 1 1 1 1 0 | 30*A* | 1 1 1 1 1 0 | −30*A* | 30*A* | 30\|*A*\| |
| 0 1 1 1 1 1 | 31*A* | 1 1 1 1 1 1 | −31*A* | 31*A* | 31\|*A*\| |

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 08, August-2016

The sign-bit exclusion technique can also be applied for 2's complement representation of coefficient A, which stores the product words in 2's complement representation, and requires a 2's complement unit along with a 2:1 MUX to change the sign of LUT output for negative values of B. If the sign bit is 0 the result is product value stored in LUT but if sign bit is 1 the true result is 2's complement of product value stored in LUT. The address of stored product word is the same as the magnitude bit of input.

### B. *Direct-Memory-Based Anti-Symmetric Product Coding (Apc) Technique:*

In Anti-Symmetric Product Coding we arrange the inputs in such a way that we can utilize the anti-symmetric property to get the output. Here for convince we assume both numbers to be positive, the product words for different input values of B for L = 5 are shown in Table II. The table is so arranged that the input word on the first column of each row is the 2's complement of that on the third column of the same row. Also, the sum of product values corresponding to these two input values on the same row is 32A. Let the product values on the second and fourth columns of a row be j and k, respectively.

Since one can write $j = [(j + k)/2 - (k - j)/2]$ and

$$k = [(j + k)/2 + (k - j)/2].$$

For $(j + k) = 32A$, we can have

$$j = 16A - [(k - j)/2] \quad (1a)$$

$$k = 16A + [(k - j)/2] \quad (1b)$$

The product values on the second and fourth columns therefore have negative mirror symmetry. This behavior of the product words can be used to reduce the LUT size, where, instead of storing j and k, only [(k - j)/2] is stored for a pair of input on a given row. The 4-bit LUT addresses and corresponding coded words are listed on the fifth and sixth columns of the table, respectively. Since the arrangement of products is done by the anti-symmetric behavior of the products, we called it anti-symmetric product coding.

To evaluate the address of APC words if MSB of the input is 1 then address is rest of the least significant bits (LSB) but if MSB of the input is 0 then address is the 2's complement of rest of the LSBs. Therefore, 4-bit address $B' = (b'_3\, b'_2\, b'_1\, b'_0)$ of the APC word is given by:

$$B' = \begin{cases} B_L & ,if\ b_4 = 1 \\ B'_L & ,if\ b_4 = 0 \end{cases} \quad (2)$$

Where $B_L = (b_3\, b_2\, b_1\, b_0)$ is the four less significant bits of B, and $B'_L$ is the 2's complement of $B_L$. The desired product could be obtained by adding or subtracting the stored value (k−j)/2 to or from the fixed value 16*A* when $b_4$ is 1or 0, respectively, i.e.

$$\text{Product word} = 16A + (\text{sign value}) \times (\text{APC word}) \quad (3)$$

Where (APC word) = (k-j)/2, sign value =+1 for $b_4 = 1$ and sign value = −1 for $b_4 = 0$. The product value for *B* = (10000) corresponds to APC value "zero," which could be derived by resetting the LUT output, instead of storing that in the LUT.

TABLE II. DIRECT-MEMORY-BASED APC TECHNIQUE FOR L=5

| Input, B | Product values | Input, B | Product values | Address B′, $b'_3\, b'_2\, b'_1\, b'_0$ | APC words |
|---|---|---|---|---|---|
| 0 0 0 0 1 | *A* | 1 1 1 1 1 | *31A* | 1 1 1 1 | *15A* |
| 0 0 0 1 0 | 2*A* | 1 1 1 1 0 | *30A* | 1 1 1 0 | 14*A* |
| 0 0 0 1 1 | 3*A* | 1 1 1 0 1 | *29A* | 1 1 0 1 | 13*A* |
| 0 0 1 0 0 | 4*A* | 1 1 1 0 0 | *28A* | 1 1 0 0 | 12*A* |
| 0 0 1 0 1 | 5*A* | 1 1 0 1 1 | *27A* | 1 0 1 1 | 11*A* |
| 0 0 1 1 0 | 6*A* | 1 1 0 1 0 | *26A* | 1 0 1 0 | 10*A* |
| 0 0 1 1 1 | 7*A* | 1 1 0 0 1 | *25A* | 1 0 0 1 | 9*A* |
| 0 1 0 0 0 | 8*A* | 1 1 0 0 0 | *24A* | 1 0 0 0 | 8*A* |
| 0 1 0 0 1 | 9*A* | 1 0 1 1 1 | *23A* | 0 1 1 1 | 7*A* |
| 0 1 0 1 0 | 10*A* | 1 0 1 1 0 | *22A* | 0 1 1 0 | 6*A* |
| 0 1 0 1 1 | 11*A* | 1 0 1 0 1 | *21A* | 0 1 0 1 | 5*A* |
| 0 1 1 0 0 | 12*A* | 1 0 1 0 0 | *20A* | 0 1 0 0 | 4*A* |
| 0 1 1 0 1 | 13*A* | 1 0 0 1 1 | 19*A* | 0 0 1 1 | 3*A* |
| 0 1 1 1 0 | 14*A* | 1 0 0 1 0 | 18*A* | 0 0 1 0 | 2*A* |
| 0 1 1 1 1 | 15*A* | 1 0 0 0 1 | 17*A* | 0 0 0 1 | *A* |
| 1 0 0 0 0 | *16A* | 1 0 0 0 0 | 16*A* | 0 0 0 0 | *0* |

[a] For B= (0 0 0 0 0), the encoded word to be stored is 16 A.

### C. *Direct-Memory-Based Modified Odd Multiple Storage (Oms) Technique:*

In Odd Multiple Storage technique instead of storing all the $2^L$ possible values of product P = A · B as in conventional, here only $(2^L/2)$ words corresponding to the odd multiples of A may be stored in the LUT, while all the even multiples of A could be derived by left-shift operations of one of those odd multiples. In Table III, we have shown that, the even multiples 2*A*, 4*A*, and 8*A* are derived by left-shift operations of *A*. Similarly, 6*A* and 12*A* are derived by left shifting 3*A*, while 10*A* and 14*A* are derived by left shifting 5*A* and 7*A*, respectively. A barrel shifter for producing a maximum of three left shifts could be used to derive all the even multiples of *A*.

In Modified OMS technique the address of the APC stored words becomes the input B of OMS such that when we combine APC-OMS technique we get the reduction in LUT size by one-fourth over conventional. At the eight memory locations the eight odd multiples of product words are stored by relation Pi = A × (2i + 1) for i =0, 1, 2 . . . . 7. As required by (3), the word to be stored for B = (00000) is not 0 but 16A, which we can obtain from A by four left shifts using a barrel shifter. However, if 16A is not derived from A, only a maximum of three left shifts is required to obtain all other even multiples of A. A maximum of three bit shifts can be implemented by a two-stage logarithmic barrel shifter, but the implementation of four shifts requires a three-stage barrel shifter. Therefore, it would be a more efficient strategy to store 2A for input B = (00000), so that the product 16A can be derived by three arithmetic left shifts. The product values and encoded words for input words B= (00000) and (10000) are separately shown in Table IV. For B= (00000), the desired encoded word 16A is derived by 3-bit left shifts of 2A [stored at address (1000)]. For B = (10000), the APC word "0" is derived by resetting the LUT output, by an active-high RESET signal given by:

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 08, August-2016

$$RESET = \left(\overline{b_0 + b_1 + b_2 + b_3}\right) \cdot b_4 \quad (4)$$

TABLE III. DIRECT-MEMORY-BASED OMS TECHNIQUE OF APC WORDS FOR L=5

| Input, B′ $b'_3 b'_2 b'_1 b'_0$ | Product value | No. of shifts | Control $S_1 S_0$ | Shifted Input, B″ | Stored odd APC words | address, $d_3 d_2 d_1 d_0$ |
|---|---|---|---|---|---|---|
| 0 0 0 1 | *A* | 0 | 0 0 | 0 0 0 1 | P0=A | 0 0 0 0 |
| 0 0 1 0 | 2× *A* | 1 | 0 1 | | | |
| 0 1 0 0 | 4× *A* | 2 | 1 0 | | | |
| 1 0 0 0 | 8× *A* | 3 | 1 1 | | | |
| 0 0 1 1 | 3*A* | 0 | 0 0 | 0 0 1 1 | P1=3A | 0 0 0 1 |
| 0 1 1 0 | 2× 3*A* | 1 | 0 1 | | | |
| 1 1 0 0 | 4× 3*A* | 2 | 1 0 | | | |
| 0 1 0 1 | 5*A* | 0 | 0 0 | 0 1 0 1 | P2=5A | 0 0 1 0 |
| 1 0 1 0 | 2× 5*A* | 1 | 1 0 | | | |
| 0 1 1 1 | 7*A* | 0 | 0 0 | 0 1 1 1 | P3=7A | 0 0 1 1 |
| 1 1 1 0 | 2× 7*A* | 1 | 0 1 | | | |
| 1 0 0 1 | 9*A* | 0 | 0 0 | 1 0 0 1 | P4=9A | 0 1 0 0 |
| 1 0 1 1 | 11*A* | 0 | 0 0 | 1 0 1 1 | P5=11A | 0 1 0 1 |
| 1 1 0 1 | 13*A* | 0 | 0 0 | 1 1 0 1 | P6=13A | 0 1 1 0 |
| 1 1 1 1 | 15*A* | 0 | 0 0 | 1 1 1 1 | P7=15A | 0 1 1 1 |

TABLE IV. PRODUCTS AND ENCODED WORDS FOR B=(00000) AND B=(10000)

| Input, B $b_4 b_3 b_2 b_1 b_0$ | Product value | Encoded word | Stored values | No. of shifts | address $d_3 d_2 d_1 d_0$ | Control $S_1 S_0$ |
|---|---|---|---|---|---|---|
| 1 0 0 0 0 | *16A* | 0 | - - - | - - | - - - - | - - |
| 0 0 0 0 0 | *0* | 16A | 2A | 3 | 1 0 0 0 | 11 |

It may be seen from Tables III and IV that the 5-bit input word *B* can be mapped into a 4-bit LUT address ($d_3 d_2 d_1 d_0$), by a simple set of mapping relation

$$d_i = b''_{i+1}, \quad \text{for i = 0, 1, 2} \quad \text{and} \quad d_3 = \overline{b''_0} \quad (5)$$

Where B″ = ($b''_3\ b''_2\ b''_1\ b''_0$) is generated by shifting-out all the leading zeros of *B′* by an arithmetic right shift followed by address mapping, i.e.

$$B'' = \begin{cases} Y_L & ,if\ b_4 = 1 \\ Y'_L & ,if\ b_4 = 0 \end{cases} \quad (6)$$

Where $Y_L$ and $Y'_L$ are derived by circularly shifting-out all the leading zeros of $B_L$ and $B'_L$, respectively. The RESET signal can alternatively be generated as ($d_3$ AND $b_4$).

$$RESET = d_3 \cdot b_4 \quad (7)$$

The control bits s0 and s1 to be used by the barrel shifter to produce the desired number of shifts of the LUT output are generated by the control circuit, according to the relation

$$s_0 = \overline{b_0 + \overline{(b_1 + \overline{b_2})}} \quad (7a)$$

$$s_1 = \overline{(b_0 + b_1)} \quad (7b)$$

Note that ($s_1\ s_0$) is a 2-bit binary equivalent of the required number of shifts specified in Tables III and IV.

## IV. OPTIMIZED DIRECT-MEMORY-BASED MULTIPLICATION FOR SIGNED AND UNSIGNED OPERANDS

In this section, we discuss that the direct-memory-based multiplication of input B with fixed coefficient A could be easily carried out for any combination of signed and unsigned magnitude number by just modifying the design.

### *A. Both Operands in Signed-magnitude form :*

The APC–OMS combined optimization of the LUT can be performed for signed values of A and B with the help of sign-bit exclusion technique. All the three technique are well utilized when both operands are in sign-magnitude form, the multiples of magnitude of the fixed coefficient are to be stored in the LUT, and the sign of the product could be obtained by the XOR operation of sign bits of both multiplicands. When both operands are in two's complement forms, a two's complement operation of the output of the LUT is required to be performed for MSB equal to 1.

### *B. Both Operands in Unsigned-Magnitude form :*

When both the operands A and B are in unsigned-magnitude form then there is no need for the sign-bit exclusion technique for the optimization. Here only the APC-OMS combined optimization technique is used for reduction in LUT size.

### *C. Input is Unsigned-Magnitude and fixed coefficient is Signed Magnitude form:*

For the multiplication of unsigned input B with signed coefficient A, the products could be stored in two's complement representation, and the sign-modification circuit checks the MSB of the output to give a 2's complement as true output for MSB equal to 1 and as it is otherwise. A straight forward implementation of the sign-modification circuit involves multiplexing of the LUT output and its two's complement, to reduce the area–time complexity.

### *D. Input is Signed-Magnitude and fixed coefficient is Unsigned Magnitude form:*

For the multiplication of signed input B with unsigned coefficient A, as the sign-bit of input is excluded by the sign-bit exclusion technique the products could be stored as it is, and the sign-modification circuit checks the MSB of the input to give a 2's complement as true output for MSB equal to 1 and as it is otherwise. Here also all the three techniques are well utilized with no modification in proposed combined design.

## V. ALGORITHMIC DESIGN FOR PROPOSED TECHNIQUES USED IN OPTIMIZED DIRECT-EMORY-BASED MULTIPLICATION

This section presents the design algorithms of techniques used in optimized direct-memory-based multiplication.

### *A. Algoritmic Design for Direct-Memory-Based Sign-Bit Exclusion technique:*

The design of this technique for L= 6 bits, consists of a memory array of 32 word size which stores the pre-computed product values and an address generating circuit which is in form of a 5To 32 line decoder for address mapping the input to a particular memory location. It also consist of a 2's

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 08, August-2016

complement unit which generates the product words in 2's complement representation along with a 2:1 MUX to change the sign of LUT output for negative values of input.

ALGORITHM

Step1: Let fixed coefficient input A be 10 bit word.

Step2: Multiplicand input B ($b_5$ $b_4$ $b_3$ $b_2$ $b_1$ $b_0$) is 6 bit word.

Step3: Product output P is 16 bit word.

Step4: Memory component has pre-computed 32 Stored

Product Words SPW of 16 bit size.

Output SPW = A·|B| corresponding to input |B|.

Step5: Signal is declared for address D ($d_4$ $d_3$ $d_2$ $d_1$ $d_0$) of 5 bits.

Step6: Signal is declared for SPW of 16 bits.

Step7: Begin for finding address D value:

Address D = |B| (i.e. $b_4$ $b_3$ $b_2$ $b_1$ $b_0$);

Step8: Begin Process 1 for finding SPW for given address D:

Case Address D is

(Here a list of address value D and corresponding to it SPW value is given according to table I ).

end Case;

End Process 1;

Step9: Begin Process 2 for finding true product P:

If sign-bit $b_5 = 0$ (i.e. for positive number) then

Product P = SPW;

else sign-bit $b_5 = 1$ (i.e. for negative number) then

Product P = 2's complement of SPW;

End Process 2;

Step10: End

*B. Algoritmic Design for Direct-Memory-Based Anti-Symmetric Product Coding (APC) Technique:*

In this technique for L=5 bit, taking only the magnitude part of signed 6bit numbers or unsigned 5bit numbers, here the design consist of a four-input memory array of 16 words to store the APC values of product words as given in the sixth column of Table II, except on the last row, where 2*A* is stored for input *B*= (00000) instead of storing a "0" for input *B* = (10000). Besides, it consists of an address-mapping circuit and an add/subtract circuit. The address-mapping circuit generates the desired address B′= ($b'_3$ $b'_2$ $b'_1$ $b'_0$) according to (2). A straightforward implementation of address mapping can be done by multiplexing $B_L$ and $B'_L$ using $b_4$ as the control bit. The output of the memory table is added with or subtracted from 16*A*, for $b_4 = 1$ or 0, respectively, according to (3) by the add/subtract cell. Hence, $b_4$ is used as the control for the add/subtract cell.

ALGORITHM

Step1: Let fixed coefficient input A be 10 bit word.

Step2: Multiplicand input B ($b_4$ $b_3$ $b_2$ $b_1$ $b_0$) is 5 bit unsigned number or is |B| for signed 6 bit number.

Step3: Product output P is 16 bit word.

Step4: Memory component has pre-computed 16 APC words.

Step5: Signal is declared for address B′ ($b'_3$ $b'_2$ $b'_1$ $b'_0$) of 4 bit.

Step6: Signal is declared for APC word of 16 bits.

Step7: Begin Process 1 for finding address B′ :

If $b_4 = 1$ then

Address B′ = ($b_3$ $b_2$ $b_1$ $b_0$);

else $b_4 = 0$ then

Address B′ =2's complement of ($b_3$ $b_2$ $b_1$ $b_0$);

End Process 1;

Step8: Begin Process 2 for finding APC for given address B′:

Case Address B′ is

(Here a list of address value B′and corresponding to it APC value is given according to table II).

end Case;

End Process 2;

Step9: Begin Process 3 for finding true product P:

If b4 = 1 then

Product P = 16A + APC;

else b4 = 0 then

Product P = 16A - APC;

End Process 3;

Step10: End

*C. Algoritmic Design for Direct-Memory-Based Modified Odd Multiple Storage (OMS) Technique:*

In this technique for L=4 bits taking unsigned 4bit number or APC words of L=5 bit and for any coefficient width W, here the design consists of a memory array of nine words of ($W$ + 4)-bit width, a four-to-nine-line address decoder, a barrel shifter, an address generation circuit, and a control circuit for generating the RESET signal and control word ($s_1s_0$) for the barrel shifter. The pre-computed values of $A \times (2_i + 1)$ are stored as $P_i$, for $i$ = 0, 1, 2, . . . , 7, at the eight consecutive locations of the memory array, as specified in Table III, while 2*A* is stored for input *B* = (00000) at memory address "1000," as specified in Table IV. The decoder takes the 4-bit address from the address generator and generates nine word-select signals, i.e., {$v_i$, for $0 \leq i \leq 8$}, to select the referenced word from the memory. The control bits $s_0$ and $s_1$ to be used by the barrel shifter to produce the desired number of shifts of the memory output are generated by the control circuit, according to (7a) and (7b). The RESET signal can be generated by (7).

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 08, August-2016

The address-generator circuit receives the input operand B and maps that onto the 4-bit address word ($d_3d_2d_1d_0$), according to (5) and (6).

ALGORITHM

Step1: Let fixed coefficient input A be 10 bit word.

Step2: Multiplicand input B ($b_3$ $b_2$ $b_1$ $b_0$) is 4 bit unsigned number or is APC word of L=5 bit.

Step3: Product output P is 16 bit word.

Step4: Memory component has pre-computed 9 OMS words.

Step5: Signal is declared for address D ($d_3$ $d_2$ $d_1$ $d_0$) of 4 bits.

Step6: Signal is declared for OMS word of 16 bits.

Step7: Signal is declared for control S ($s_1$ $s_0$) of 2 bits.

Step8: Signal is declared for RESET of 1 bit.

Step9: Begin Process 1for finding address and control signal:

Case Input B is

(Here a list of input values B and corresponding to it Control signal value S and address location D is given according to table III and IV).

end Case;

End Process 1;

Step10: Begin Process 2 for finding OMS for given address D:

Case Address D is

(Here a list of address value D and corresponding to it OMS value is given according to table III and IV).

end Case;

End Process 2;

Step11: Begin Process 3 for finding true product value.

RESET = $d_3$. $b_4$

If RESET = 1 then

Product P= 0

else RESET=0 then

Product P= OMS << S (i.e. OMS is left shift by S)

End Process 3;

Step12: End

*D. Algorithmic Design Of Combined Technique Used In Proposed Optimized Direct-Memory-Based Multiplication:*

The algorithmic design principle here is to utilize all the three technique Sign-Bit Exclusion, APC and Modified OMS efficiently in an optimized manner to get our proposed design which reduce the LUT memory size to one-eighth of the conventional LUT. By efficiently combine all the technique we get optimized direct-memory-based multiplication hardware. It consists of an address generator and control circuit, 4-To-9 address-line decoder, 9×(W+6) LUT memory units, barrel shifter, add/subtract unit and 2's complement/sign-modification unit.

ALGORITHM

Step1: Let fixed coefficient input A be 10 bit word.

Step2: Multiplicand input B (b5 b4 b3 b2 b1 b0) is 6 bit signed number.

Step3: Product output P is 16 bit word.

Step4: Memory component has pre-computed 9 OMS words.

Step5: Signal is declared for address D($d_3$ $d_2$ $d_1$ $d_0$) of 4 bits.

Step6: Signal is declared for input B′($b'_3$ $b'_2$ $b'_1$ $b'_0$) of OMS technique of 4 bits.

Step7: Signal is declared for OMS word of 16 bits.

Step8: Signal is declared for APC word of 16 bits.

Step9: Signal is declared for control S($s_1$ $s_0$) of 2 bits.

Step10: Signal is declared for RESET of 1 bit.

Step11: Begin Process 1 for finding Input B′ of OMS value:

Case Input ($b_5$ $b_4$) is

[Value ($b_5$ b4) = "00"

Input of OMS B′ =2's complement of ($b_3$ $b_2$ $b_1$ $b_0$);

Value ($b_5$ $b_4$) = "01"

Input of OMS B′ = ($b_3$ $b_2$ $b_1$ $b_0$);

Value ($b_5$ $b_4$) = "10"

Input of OMS B′ =2's complement of ($b_3$ $b_2$ $b_1$ $b_0$);

Value ($b_5$ $b_4$) = "11"

Input of OMS B′ = ($b_3$ $b_2$ $b_1$ $b_0$);]

end Case;

End Process 1;

Step12: Begin Process 2 for finding address D and control signal S:

Case Input B' is

(Here a list of input values B' and corresponding to it Control signal value S and address location D is given according to tables III and IV).

end Case;

End Process 2;

Step13: Begin Process 3 for finding OMS for given address D:

Case Address D is

(Here a list of address D and corresponding to it OMS value is given according to tables III and IV).

end Case;

End Process 3;

Step14: Begin Process 4 for finding APC product value:

RESET = d3. b4;

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 08, August-2016

If RESET = 1 then

Product APC= 0;

else RESET=0 then

Product APC= OMS << S; (i.e. OMS is left shift by S)

End Process 4;

Step15: Begin Process 5 for finding true product P;

Case Input ($b_5$ $b_4$) is

[Value (b5 b4) = "00"

Product P= 16A – APC;

Value ($b_5$ $b_4$) = "01"

Product P= 16A + APC;

Value ($b_5$ $b_4$) = "10"

Product P=2's complement of (16A – APC);

Value ($b_5$ $b_4$) = "11"

Product P=2'scomplement of (16A + APC);]

end Case;

End Process 5;

Step16: End

## VI. HARDWARE IMPLEMENTATION OF DIRECT-MEMORY-BASED MULTIPLIER USING THE PROPOSED OPTIMIZATION TECHNIQUE

The hardware implementation of direct-memory-based multiplier for an L-bit input with a W-bit coefficient using the proposed optimization scheme is shown in fig.1. The multiplicand input ($b_5$ $b_4$ $b_3$ $b_2$ $b_1$ $b_0$) is applied to address generator and control circuit to generate the desired address location ($d_3d_2d_1d_0$), RESET and control signal ($s_1$ $s_0$), according to (5), (6), (7), (7a) and (7b) respectively. The function of a 4To9 address-line decoder is to take the 4-bit address from the address generator and generate nine word-select signals, i.e., {$v_i$, for $0 \leq i \leq 8$}, to select the referenced word from the memory unit. The memory unit consist of LUT of nine words of ($W$ + 6)-bit width, which stores the pre-computed values of $P_i = A \times (2_i + 1)$, for $i$ = 0, 1, 2, . . . , 7, at the eight consecutive locations of the LUT memory unit, as specified in Table III, while 2$A$ is stored for input $B$ = (00000) at LUT address "1000," as specified in Table IV. The control bits $s_0$ and $s_1$ is used by the barrel shifter to produce the desired number of shifts of the LUT memory output. The output of the barrel shifter is added with or subtracted from 16$A$, for $b_4$ = 1 or 0, respectively, according to (3) by the add/subtract unit. Hence, $b_4$ is used as the control for the add/subtract unit. At the 2's complement/sign modification unit, a two's complement operation of the output of add/subtract cell is required to be performed for $b_5$=1 and remain as it is for $b_5$=0, when both operands are in two's complement form. Here $b_5$ act as a sign control signal. The RTL schematic of Optimized Direct-Memory-Based Multiplier Design, Using Sign-Bit Exclusion, APC and OMS Technique is shown in Fig.2.

## VII. RESULT AND CONCLUSION

The proposed optimized memory-based multiplier is coded in VHDL and synthesized in Xilinx ISE 9.1i Project Navigator, for word size L = 5 and 6 bits for signed magnitude numbers and L= 5 bits unsigned magnitude numbers respectively. For unsigned numbers we use APC and modified OMS scheme whereas for signed-magnitude numbers the sign-bit exclusion

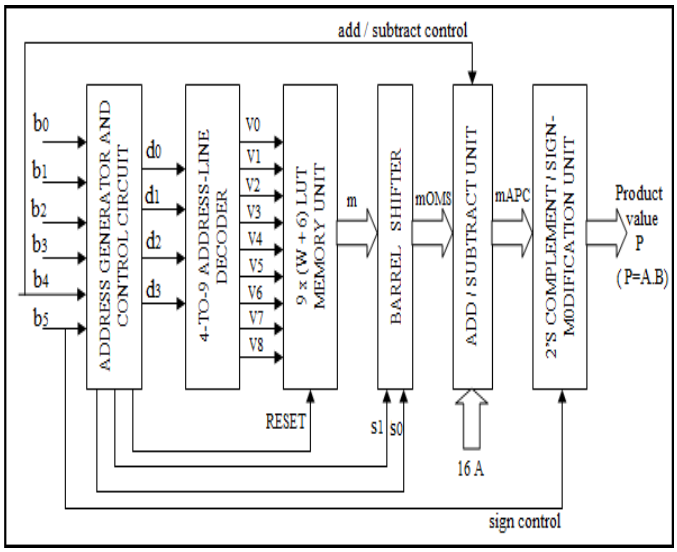


Fig. 1. Proposed Optimized Direct-Memory-Based Multiplier Design, Using Sign-Bit Exclusion, APC and OMS Technique.

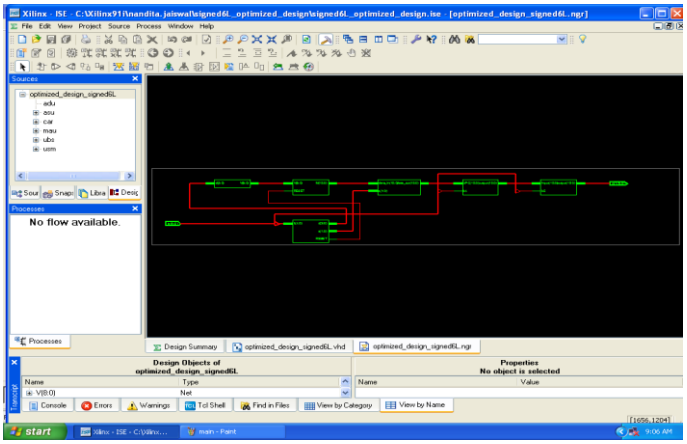


Fig. 2. RTL schematic of Optimized Direct-Memory-Based Multiplier Design, Using Sign-Bit Exclusion, APC and OMS Technique.

scheme is included in APC and modified OMS scheme to get an optimized LUT multiplier which reduces the memory size to one-eighth of conventional LUT. Simulation result, area utilization and timing analysis for 6 bit signed number, 5 bit unsigned number and 5 bit signed number are shown in fig.3, fig.4, fig.5, fig.6, fig.7, fig.8, fig.9, fig.10 and fig.11 respectively.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 08, August-2016

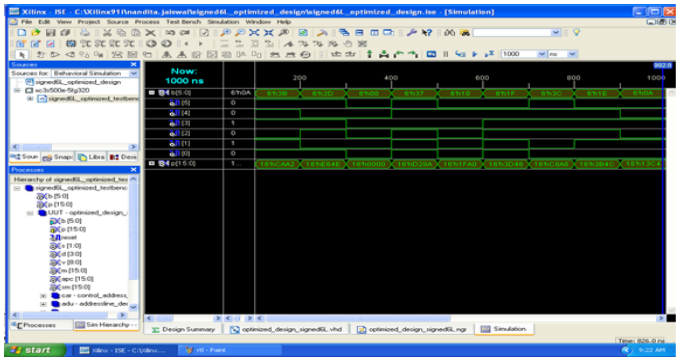

Fig. 3. Simulation Result of Proposed Optimized Design for Signed-Magnitude Numbers, L=6 Bits

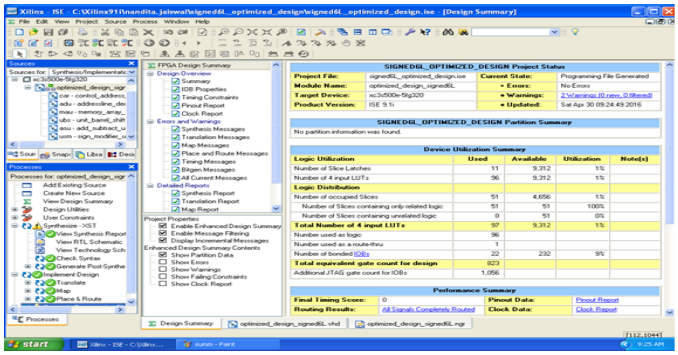

Fig. 4. Area Utilization of Proposed Optimized Design for Signed-Magnitude Numbers, L=6 Bits

Fig. 5.

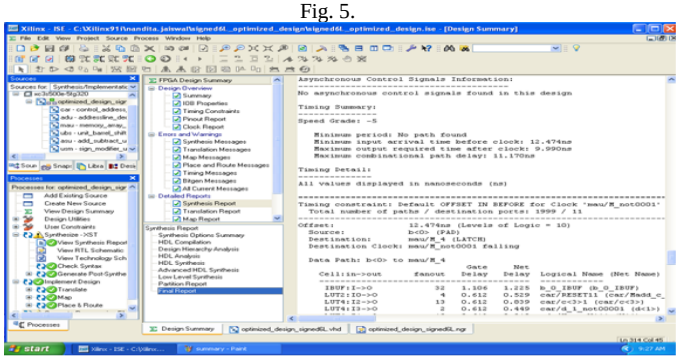

Fig. 6. Timing Report of Proposed Optimized Design for Signed-Magnitude Numbers, L=6 Bits

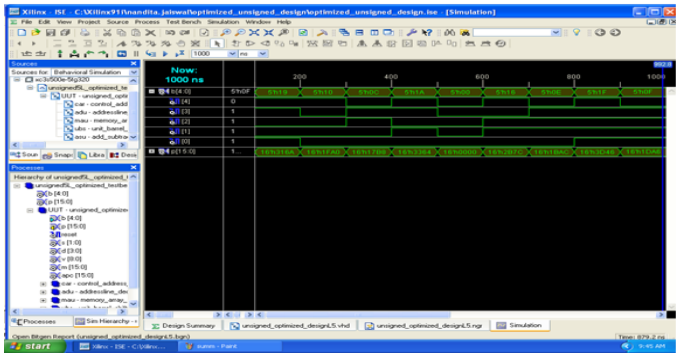

Fig. 7. Simulation Result of Optimized Design for Unsigned-Magnitude Numbers, L=5 Bits

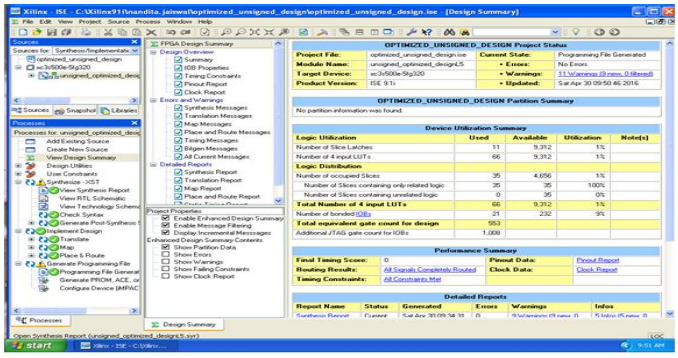

Fig. 8. Area Utilization of Optimized Design for Unsigned-Magnitude Numbers, L=5 Bits

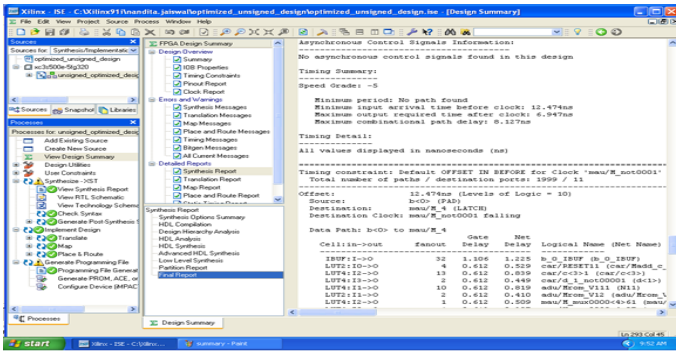

Fig. 9. Timing Report of Optimized Design for Unsigned-Magnitude Numbers, L=5 Bits

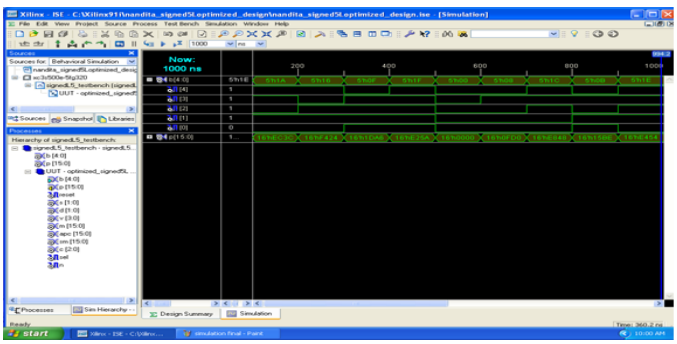

Fig. 10. Simulation Result of Proposed Optimized Design for Signed-Magnitude Numbers, L=5 Bits

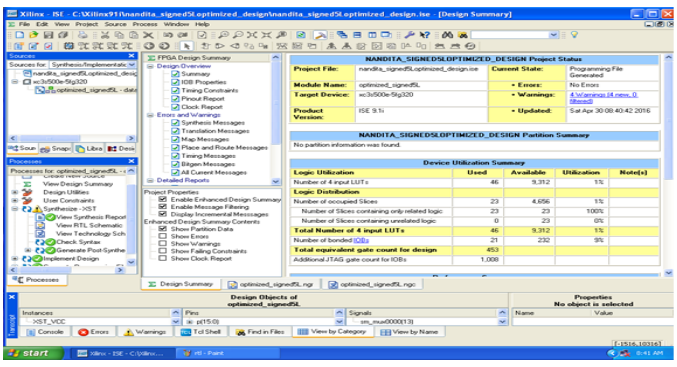

Fig. 11. Area Utilization of Proposed Optimized Design for Signed-Magnitude Numbers, L=5 Bits

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 08, August-2016

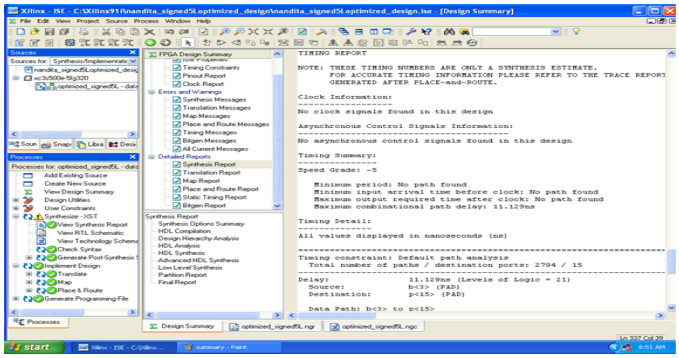

Fig. 12. Timing Report of Proposed Optimized Design for Signed-Magnitude Numbers, L=5 Bits

Comparison of memory block and LUT memory size for L= 6 bits by different techniques is shown in Table V. In Table VI we have compared the results derived from are design summary and synthesis report for area and timing parameters. It is found that for signed magnitude numbers of 5 bit word size it offers more saving in area-delay product as of 5 bit word size unsigned magnitude numbers. With our proposed optimized design LUT memory size for L= 6 bits is 14.065% of conventional LUT and for L= 5 bits is 12.5% of conventional LUT. Thus, the proposed optimized design also offers almost 85% and 87.5% reductions in LUT memory size for L=6 bits and L=5 bits respectively over conventional LUT. The design also saves a lot of multiplication computation power as it is direct-memory based and no product computation are carried out, hence with a very negligible trade off in delay a lot of area and power are saved.

The LUT multiplier could be used for memory-based implementation of linear and circular convolution, cosine and sinusoidal transforms, and inner-product computation. The performance of memory-based structures with different memory implementations could be studied in future for different DSP applications. Although the memory core of the direct-memory-based multiplier is reduced to nearly one-eighth by the proposed optimization technique, it is not efficient for operands of small widths, since it requires an adder to add the offset value. However, it could be used for multiplication with input of large word size by an input decomposition scheme. When the width of the input multiplicand is large, direct implementation of LUT multiplier involves a very large LUT. Therefore, the input word could be decomposed into a certain number of segments or sub-words, and the partial products pertaining to different sub-words could be shift added to obtain the desired product. In brief, we have shown the possibility of using direct-memory-based multipliers to implement the constant multiplication for DSP applications. The full advantages of proposed design, however, could be derived if the LUTs are implemented as NAND or NOR read-only memories and the arithmetic shifts for large operand word size are implemented by an array barrel shifter using metal–oxide–semiconductor transistors. Further work could still be done to derive Sign-Bit Exclusion, OMS–APC-based LUTs for higher input sizes with different forms of decompositions and parallel and pipelined addition schemes for suitable area–delay tradeoffs.

TABLE V. COMPARISON STUDY OF OPTIMIZED DESIGN FOR UNSIGNED- MAGNITUDE NUMBERS WITH PROPOSED OPTIMIZED DESIGN FOR SIGNED-MAGNITUDE NUMBERS

| PARAMETERS | UNSIGNED NUMBERS Using APC and OMS Technique L= 5 bit | SIGNED NUMBERS Using Sign-bit Exclusion, APC and OMS Technique L=5 bit | SIGNED NUMBERS Using Sign-bit Exclusion, APC and OMS Technique L=6 bit |
|---|---|---|---|
| Memory Block | 9 out of 32 | 4 out of 32 | 9 out of 64 |
| LUT size used in % | 28.125 | 12.5 | 14.0625 |
| Number of Slices used out of 4656 | 35 | 23 | 51 |
| Number of Slice Flip Flops used out of 9312 | 11 | ---- | 11 |
| Number of 4 input LUTs used out of 9312 | 66 | 46 | 97 |
| Number of bonded IOBs used out of 232 | 21 | 21 | 22 |
| Minimum input arrival time before clock (ns) | 12.474 | ---- | 12.474 |
| Maximum output required time after clock (ns) | 6.947 | ---- | 9.990 |
| Maximum combinational path delay (ns) | 8.127 | 11.129 | 11.170 |

TABLE VI. COMPARISON OF MEMORY BLOCKS AND LUT SIZE FOR L= 6

| METHOD | *CONVENTIONAL* | *SIGN-BIT EXCLUSION* | *APC* | *OMS* | *SIGN-BIT EXCLUSION + APC* | *APC+OMS* | *SIGN-BIT EXCLUSION + APC + OMS* |
|---|---|---|---|---|---|---|---|
| MEMORY BLOCK | 64 | 32 | 33 | 33 | 17 | 17 | 9 |
| LUT SIZE | 100% | 50% | 51.5625% | 51.5625% | 26.5625% | 26.5625% | 14.0625% |

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 08, August-2016

## REFERENCES

[1] International Technology Roadmap for Semiconductors. [Online]. Available: http://public.itrs.net/

[2] J.-I. Guo, C.-M. Liu, and C.-W. Jen, "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, Oct. 1992.

[3] H.-R. Lee, C.-W. Jen, and C.-M. Liu, "On the design automation of the memory-based VLSI architectures for FIR filters," *IEEE Trans. Consum. Electron.*, vol. 39, no. 3, pp. 619–629, Aug. 1993.

[4] D. F. Chiper, M. N. S. Swamy, M. O. Ahmad, and T. Stouraitis, "A systolic array architecture for the discrete sine transform," *IEEE Trans. Signal Process.*, vol. 50, no. 9, pp. 2347–2354, Sep. 2002.

[5] H.-C. Chen, J.-I. Guo, T.-S. Chang, and C.-W. Jen, "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, Mar. 2005.

[6] D. F. Chiper, M. N. S. Swamy, M. O. Ahmad, and T. Stouraitis, "Systolic algorithms and a memory-based design approach for a unified architecture for the computation of DCT/DST/IDCT/IDST," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 6, pp. 1125–1137, Jun. 2005.

[7] P. K. Meher, "Systolic designs for DCT using a low-complexity concurrent convolutional formulation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 9, pp. 1041–1050, Sep. 2006.

[8] P. K. Meher, "Memory-based hardware for resource-constrained digital signal processing systems," in *Proc. 6th Int. Conf. ICICS*, Dec. 2007, pp. 1–4.

[9] P. K. Meher, "New approach to LUT implementation and accumulation for memory-based multiplication," in *Proc. IEEE ISCAS*, May 2009, pp. 453–456.

[10] P. K. Meher, "New look-up-table optimizations for memory-based multiplication," in *Proc. ISIC*, Dec. 2009, pp. 663–666.

[11] P.K.Mehra, "LUT optimization for Memory-Based Computation," *IEEE Trans. Circuits and Syst.-II:Express Briefs, vol.*57,no.4,pp.285-289, Apr.2010.