International Journal of Advance Research and Innovation
Vol. 8(2), Apr-Jun 2019, pp. 91-96
Doi: 10.51976/ijari.822016

Article Info

Received: 17 Jan 2020 | Revised Submission: 20 May 2020 | Accepted: 28 May 2020 | Available Online: 15 Jun 2020

Low Latency and Efficient LUT Based Multiplier for DSP Applications

Merugumalli Rama Krishna, K. Sri Lakshmi**, S. Lalitha* and C. Amulya**

ABSTRACT

In digital signal processing memory based computation plays a vital role for DSP applications, which has multiplication with a fixed set of coefficient. LUT optimization for memory based multiplication can be done with these three computational techniques like Anti symmetry product coding (APC) and Odd multiple storage (OMS), combined APC-OMS. OMS technique with the modified APC-OMS based LUT multiplier can be discussed in terms of area and delay. These techniques are coded in VHDL language and synthesized in Xilinx ISE design suite 14.7. Thus, this proposes the APC-OMS based LUT multiplier can optimize the LUT size and consumes less area and obtains high speed when compared to other techniques. The proposed LUT based multiplier requires less significant area and less multiplication time than the canonical-signed-digit based multiplier.

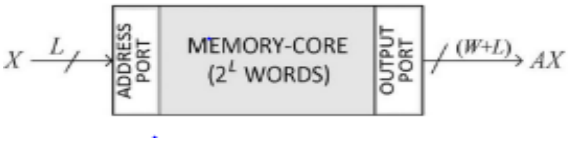
Keywords: *Look up table; Anti symmetric coding; Odd multiple Storage; Canonical-signed-digit; Memory based computations.*

1.0 Introduction

Along with the device scaling over the years, semiconductor memory has become cheaper, faster and more power-efficient. Moreover, according to the projections of the ITRS embedded memories will have dominating presence in the SoC, which may exceed 90% of total SoC content. It has also been found that the transistor packing density of memory components is not only high but also increasing much faster than the transistor density of logic components. Apart from that, the 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 [7]. Memory-based computing is well-suited for many DSP algorithms, which involve multiplication with fixed set of coefficients [8].

The basic functional model of memory-based multiplier is shown in Fig.1. Let A be a fixed coefficient and X be an input word to be multiplied with A. Assuming X to be a positive binary number with word-length L, there can be 2L possible values of X, and accordingly, there can be 2L possible values of product C = A·X. Therefore, for memory based multiplication, an LUT of 2L words consisting of precomputed product values corresponding to all possible values of X is conventionally used.

Figure 1: Conventional LUT



The product-word A·Xiis stored at the location whose address is the same as Xi for 0 ≤ 2L−1, such that if L-bit binary value of Xi is used as address for the LUT, then the corresponding product value A·Xiis 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 but do not find any significant work on LUT optimization for memory based multiplication [1].

**Department of Electronics and Communication Engineering, Andhra Loyola Institute of Engineering and Technology Vijayawada, Andhra Pradesh, India*

***Corresponding author; Department of Electronics and Communication Engineering, Andhra Loyola Institute of Engineering and Technology Vijayawada, Andhra Pradesh, India (E-mail: Srilakshmikadiyala99@gmail.com)*

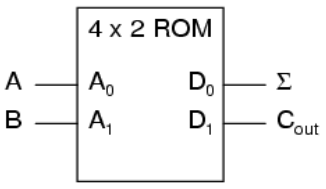
A new approach to LUT design for memory-based multiplication, which could be used to reduce the memory-size by half for small input-widths. It is shown that although 2L possible values of X correspond to 2L possible values of C = A.X, only (2L/2) words corresponding to the odd multiples of A may only be stored in the LUT while the even multiples of A could be derived by left-shift operations of one of those odd multiples[1]. 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 this project, two new schemes are proposed 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 optimization is based on a recording of stored product word [7].

This paper has been organized as follows. Section I describes motivation and introduction of the work. Section II describes related work. Section III describes Proposed System. Section IV describes Simulation results. Section V describes conclusion and future scope.

1.1 General LUT design

It is possible to store binary data within solid-state devices. Those storage "cells" within solid-state memory devices are easily addressed by driving the "address" lines of the device with the proper binary values. A ROM memory circuit[5] written, or programmed, with certain data, such that the address lines of the ROM served as inputs and the data lines of the ROM served as outputs, generating the characteristic response of a particular logic function. Theoretically, could ROM chip can program to emulate whatever logic function, wanted without having to alter any wire connections or gates. Consider the following example of a 4 x 2 bit ROM memory programmed with the functionality of a half adder:

Figure 2: 4X2 ROM



Address		Data	
A	B	C_{out}	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

If this ROM has been written with the above data representing a half-adder's truth table, driving the A and B address inputs will cause the respective memory cells in the ROM chip to be enabled, thus outputting the corresponding data as the Σ (Sum) and C out bits. Unlike the half-adder circuit built of gates or relays, this device can be set up to perform any logic function at all with two inputs and two outputs, not just the half-adder function. To change the logic function, all we would need to do is write a different table of data to another ROM chip. EPROM chip can also program which could be re-written at will, giving the ultimate flexibility in function. It is vitally important to recognize the significance of this principle as applied to digital circuitry. Whereas the half-adder built from gates or relays processes the input bits to arrive at a specific output, the ROM simply remembers what the outputs should be for any given combination of inputs. This is not much different from the "times tables" memorized in grade school: rather than having to calculate the product of 5 times 6 (5 + 5 + 5 + 5 + 5 + 5 = 30), school-children are taught to remember that 5 x 6 = 30, and then expected to recall this product from memory as needed. Likewise, rather than the logic function depending on the functional arrangement of hard-wired gates or relays (hardware), it depends solely on the data written into the memory (software).

1.2 LUT for multipliers

Multiplications can be computationally expensive in most hardware and software implementations. Various approaches in literature have been proposed to alleviate this overhead, usually at the cost of multiplication accuracy. One such example is the conversion of multiplication coefficients to dyadic fractions [2], which can be computed with a minimal sequence of bit shifts and additions. However, such approaches have proved to be limiting, requiring a lot of hand-tweaking to simultaneously minimize the complexity of the calculation as well as the deviation from the desired result. Instead, a table-based lookup scheme to implement the multiplication steps is proposed. Whenever a multiplication result is needed, the system can simply look up the correct result on a precomputed table [1], without needing any computation whatsoever. This greatly simplifies the transform and inverse calculations. Table lookup can replace any coefficient multiplication or unary

operation [3]. Although table lookup is often simpler than the actual calculation, the table size grows exponentially with the input signal range. However, for image and video applications [5], most signals are unsigned 8 bit values, which require only 256 possible cases, so the table based approach can be implemented with a reasonable cost. To implement coefficient multiplication, where the coefficient is 0.6834. To avoid using a multiplier, traditional lossless transforms approximate the given coefficient with a dyadic fraction (for example, to ¾). Then the coefficient multiplication can be implemented using shifts and additions as shown. Table lookup is also depicted .Unlike in the dyadic fraction case [6], table based multiplication yields a much more accurate approximation of the original coefficient.

2.0 Related Work

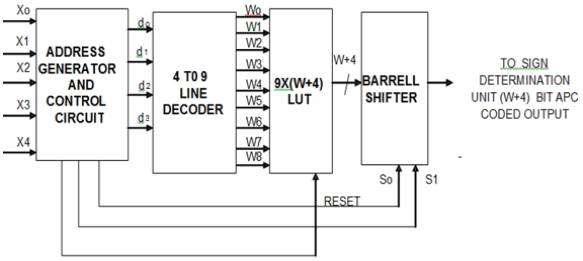
Guo et .al[1]Two new techniques are presented, for the reduction of look-up-table size of memory-based multipliers to be used in digital signal processing applications. It is shown that by simple sign-bit exclusion, the LUT size is reduced by half at the cost of a marginal area overhead. Moreover, a novel anti-symmetric product coding scheme is proposed to reduce the LUT size by further half, where the LUT output is added with or subtracted from a fixed value. It is shown that the optimized LUTs for small input width could be used for efficient implementation of high-precision LUT-multipliers, where the total contribution of all such fixed offsets could be added to the final result or could be initialized for successive accumulations. The proposed LUT-multiplier and the existing ones are coded in VHDL and synthesized by Synopsys Design Compiler using TSMC 90 nanometre library. The proposed optimized LUT-multiplier is found to involve less area and less multiplication time than the existing LUT-multipliers. The conventional LUT-multiplier and the odd-multiple storage LUT of involve nearly 56.8% and 26.2% more area-delay product, in average, for 16-bit input than the proposed LUT design.

Jen et.al [4]Efficient memory-based VLSI arrays and a new design approach for the discrete Fourier transform and discrete cosine transform are presented. The DFT and DCT are formulated as cyclic convolution forms and mapped into linear arrays which characterize small numbers of I/O channels and low I/O bandwidth. Since the multipliers consume much hardware area, the designs utilize small ROMs and adders to implement the multiplications. Moreover, the ROM size can be reduced effectively by arranging the data in the designs appropriately. The arrays outperform others in the architectural topology, computing speeds, hardware complexity, the number of I/O channels, and I/O bandwidth. They benefit from the advantages of both systolic array and the memory-based architectures.

3.0 Proposed System

A new approach to LUT design is presented, where only the odd multiples of the fixed coefficient are required to be stored, which is referred to as the odd-multiple-storage scheme in this brief. In addition, we have shown that, by the anti-symmetric product coding approach [7], the LUT size can also be reduced to half, where the product words are recoded as Anti-symmetric pairs.

Figure 3: Proposed LUT Multiplier



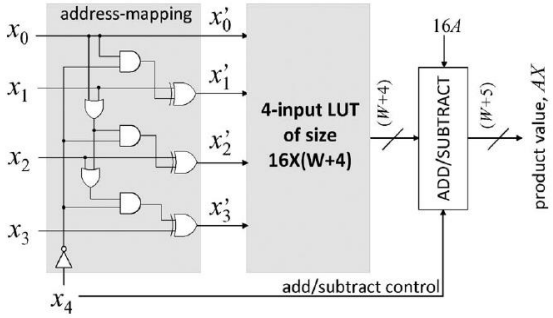
If the input bit size= 5 then the memory stored is of 2^5/2 = 15 locations which results in a reduction in LUT size by factor of 2.

3.1 Proposed LUT APC part

The structure and function of the LUT-based multiplier for L = 5 using the APC technique [6] is shown in Figure It consists of a four-input LUT of 16 words to store the APC values of product words as given in the sixth column of Table I, except on the last row, where 2A is stored for input X = (00000) instead of storing a "0" for input X = (10000). Besides, it consists of an address-mapping circuit and an add/subtract circuit. The address-mapping circuit generates the desired address (x3', x2', x1', x0'). A straightforward implementation of address mapping can be done by $X'L$ using $x4$ as the control bit. The

address-mapping circuit, however, can be optimized to be realized by three XOR gates, three AND gates, two OR gates, and a NOT gate, as shown in Figure. Note that the RESET can be generated by a control circuit (not shown in this figure) .The output of the LUT is added with or subtracted from 16*A*, for *x*4 = 1 or 0, respectively, by the add/subtract cell. Hence, *x*4 is used as the control for the add/subtract cell.

Figure 4: Proposed APC Part



For simplicity of presentation, it is assumed both X and A to be positive integers. The product words for different values of X for L = 5 are shown in Table I. It may be observed in this table that the input word X on the first column of each row is the two's complement of that on the third column of the same row. In addition, the sum of product values corresponding to these two input values on the same row is 32A. LUT based multiplier for L=5 using the APC technique

W = Width of A

L = Width of X

16 x (W+4) → 16 Locations and each location having (W+4) bits.

Let the product values on the second and fourth columns of a row be u and v, respectively. Since one can write

u = [(u + v)/2 − (v − u)/2] and

v = [(u + v)/2 + (v − u)/2], for

(u + v) = 32A,

U=16A+[(V-U)/2]

V=16A-[(V-U)/2]

The product values on the second and fourth columns of Table I therefore have a negative mirror symmetry [9]. This behaviour of the product words can be used to reduce the LUT size, where, instead of storing u and v, only [(v − u)/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.

Table 1: Stored APC Words

APC WORDS FOR DIFFERENT INPUT VALUES FOR $L = 5$

Input, X	product values	Input, X	product values	address $x_3'x_2'x_1'x_0'$	APC words
0 0 0 0 1	A	1 1 1 1 1	$31A$	1 1 1 1	$15A$
0 0 0 1 0	$2A$	1 1 1 1 0	$30A$	1 1 1 0	$14A$
0 0 0 1 1	$3A$	1 1 1 0 1	$29A$	1 1 0 1	$13A$
0 0 1 0 0	$4A$	1 1 1 0 0	$28A$	1 1 0 0	$12A$
0 0 1 0 1	$5A$	1 1 0 1 1	$27A$	1 0 1 1	$11A$
0 0 1 1 0	$6A$	1 1 0 1 0	$26A$	1 0 1 0	$10A$
0 0 1 1 1	$7A$	1 1 0 0 1	$25A$	1 0 0 1	$9A$
0 1 0 0 0	$8A$	1 1 0 0 0	$24A$	1 0 0 0	$8A$
0 1 0 0 1	$9A$	1 0 1 1 1	$23A$	0 1 1 1	$7A$
0 1 0 1 0	$10A$	1 0 1 1 0	$22A$	0 1 1 0	$6A$
0 1 0 1 1	$11A$	1 0 1 0 1	$21A$	0 1 0 1	$5A$
0 1 1 0 0	$12A$	1 0 1 0 0	$20A$	0 1 0 0	$4A$
0 1 1 0 1	$13A$	1 0 0 1 1	$19A$	0 0 1 1	$3A$
0 1 1 1 0	$14A$	1 0 0 1 0	$18A$	0 0 1 0	$2A$
0 1 1 1 1	$15A$	1 0 0 0 1	$17A$	0 0 0 1	A
1 0 0 0 0	$16A$	1 0 0 0 0	$16A$	0 0 0 0	0

For $X = (0\ 0\ 0\ 0\ 0)$, the encoded word to be stored is $16A$.

Since the representation of the product is derived from the anti symmetric behaviour of the products [8], we can name it as anti symmetric product code. The 4-bitaddress X_ = (x_3, x_2, x_1, x_0) of the APC word is given by

X' { **XL,** if x4=1; **X′L**, if x4=0

3.2 Proposed APC-OMS part

For the multiplication of any binary word X of size L, with a fixed coefficient A, instead of storing all the 2L possible values of C = A · X, only (2L/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 [10] of one of those odd multiples. Based on the above assumptions, the LUT for the multiplication of an L-bit input with a W-bit coefficient could be designed by the following strategy:

1 A memory unit of [(2L/2) + 1] words of (W + L)-bit width is used to store the product values, where the first (2L/2) words are odd multiples of A, and the last word is zero.

2 A barrel shifter for producing a maximum of (L − 1) left shifts is used to derive all the even multiples of A.

3 The L-bit input word is mapped to the (L − 1)-bit address of the LUT by an address encoder, and control bits for the barrel shifter [10] are derived by a control circuit.

Table 2: Stored APC-OMS Words

input X' $x'_3x'_2x'_1x'_0$	product value	# of shifts	shifted input, X''	stored APC word	address $d_3d_2d_1d_0$
0 0 0 1	A	0	0 0 0 1	$P0 = A$	0 0 0 0
0 0 1 0	$2 \times A$	1			
0 1 0 0	$4 \times A$	2			
1 0 0 0	$8 \times A$	3			
0 0 1 1	$3A$	0	0 0 1 1	$P1 = 3A$	0 0 0 1
0 1 1 0	$2 \times 3A$	1			
1 1 0 0	$4 \times 3A$	2			
0 1 0 1	$5A$	0	0 1 0 1	$P2 = 5A$	0 0 1 0
1 0 1 0	$2 \times 5A$	1			
0 1 1 1	$7A$	0	0 1 1 1	$P3 = 7A$	0 0 1 1
1 1 1 0	$2 \times 7A$	1			
1 0 0 1	$9A$	0	1 0 0 1	$P4 = 9A$	0 1 0 0
1 0 1 1	$11A$	0	1 0 1 1	$P5 = 11A$	0 1 0 1
1 1 0 1	$13A$	0	1 1 0 1	$P6 = 13A$	0 1 1 0
1 1 1 1	$15A$	0	1 1 1 1	$P7 = 15A$	0 1 1 1

4.0 Simulation Results

Figure 5: Simulation Results for Word Size of L=5

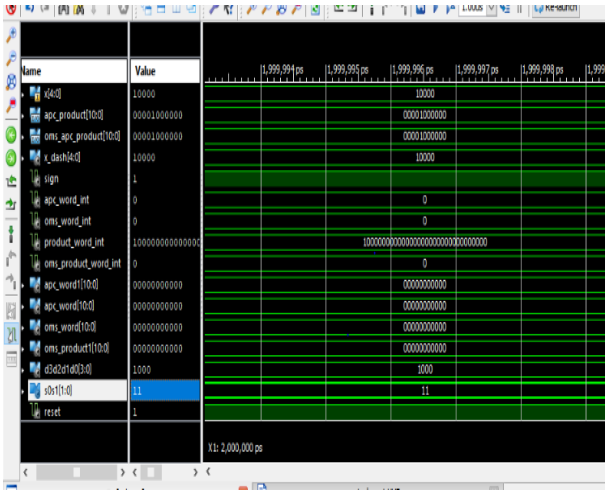


Table 3: Comparison Results of Existing and Proposed System

Design	Power (W)	Area (Gate Count)	Delay (ns)
Existing of word size L=5	0.790	92	6.49
Proposed of word size L=5	0.080	58	6.331

5.0 Conclusions

It is found that the proposed LUT-based multiplier involves comparable area and time complexity for a word size of 8 bits, but for higher word sizes, it involves significantly less area and less multiplication time than the canonical-signed-digit (CSD)-based multipliers. For 16 and 32 bit word sizes, respectively, it offers more than 30% and 50% of saving in area-delay product over the corresponding CSD multipliers. The proposed LUT multipliers for word size $L = W = 5$ bits are coded in VHDL and synthesized by XILINX using the TSMC 90-nm Library, where the LUTs are implemented as arrays of constants. The CSD-based multipliers having the same addition schemes are also synthesized with the same technology library. In future this LUT based multipliers can also be used in DSP systems for multiplications involving in FFT. It can be able to give the good results when compared to LUT multipliers using in FIR filters.

References

[1] PK Meher. New approach to LUT implementation and accumulation for memory-based multiplication, in Proc. IEEE ISCAS, May 2009, pp. 453–456.

[2] DF Chiper, MNS Swamy, MO Ahmad, T Stouraitis. A systolic array architecture for the discrete sine transform, IEEE Trans. Signal Process., 50(9), 2002, 2347–2354.

[3] HC Chen, JI Guo, TS Chang, CW Jen. A memory-efficient realization of cyclic convolution and its application to discrete cosine transform, IEEE Trans. Circuits Syst. Video Technol., 15(3), 2005, 445–453.

[4] JI Guo, CMLiu, CW Jen. The efficient memory-based VLSI array design for DFT and DCT, IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process, 39(10), 1992, 723–733.

[5] Q Znao, Y Tadokoro. A simple design of FIR filters with power-of-two coefficients, IEEE Trans. on Circuits and Systems, 35(5), 1998, 566–570.

[6] PK Meher. Memory-based Hardware for resource-constrained digital signal processing systems, in Proc.6th Int Conf. ICICS, 12, 2007, 1–4.

[7] PK Meher. LUT Optimization forMemory-Based Computation, IEEE Transactions oncircuits and systems-II: express briefs, 57(4), 2010.

[8] PK Meher. LUT optimization for memory-based computation, IEEE Transactions on Circuits and Systems II: Express Briefs, 57(4), 2010, 285– 289.

[9] PK Meher. New approach to look-up-table design and memory-based realization of FIR digital filter, IEEE Transactions on Circuits and Systems I: Regular Papers, 57(3), 2010, 592-603.

[10] JP Choi, SC Shin, JG Chung. Efficient ROM size reduction for distributed arithmetic, in IEEE International Symposium on Circuits and Systems. Emerging Technologies for the 21st Century.Proceedings (IEEE Cat No. 00CH36353), 2, 2000, 61–64.

[11] BK Mohanty, PK Mehar, SK Patel. LUT optimisation for distributed arithmetic based block LMS adaptive filter, IEEE Transactions on VLSI Systems, 24(5), 2016, 1926–1935.