ENGINEERING IN ADVANCED RESEARCH SCIENCE AND TECHNOLOGY

ISSN 2278-2566
Vol.01, Issue.03
April -2019
Pages: -165-172

APC AND OMS BASED LUT DESIGN FOR MULTIPLIER APPLICATIONS

1. NUKATHOTI GOPI, 2. JANGILI PRASAD
1. M. Tech Student, Dept. of ECE, ABR College of Engineering and Technology, Kanigiri, A.P
2. Assistant Professor, Dept. of ECE, ABR College of Engineering and Technology, Kanigiri, A.P

ABSTRACT:
Recently, we have proposed the antisymmetric product coding (APC) and odd-multiple-storage (OMS) techniques for lookup-table (LUT) design for memory-based multipliers tobe used in digital signal processing applications. Each of these techniques results in the reduction of the LUT size by a factorof two. In this brief, we present a different form of APC and a modified OMS scheme, in order to combine them for efficient memory-based multiplication. The proposed combined approachprovides a reduction in LUT size to one-fourth of the conventional LUT. We have also suggested a simple technique for selective sign reversal to be used in the proposed design. It is shown that theproposed LUT design for small input sizes can be used for efficient implementation of high-precision multiplication by input operand decomposition. 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.

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. Memory-based computing is well-suited for many DSP algorithms, which involve multiplication with fixed set of coefficients.

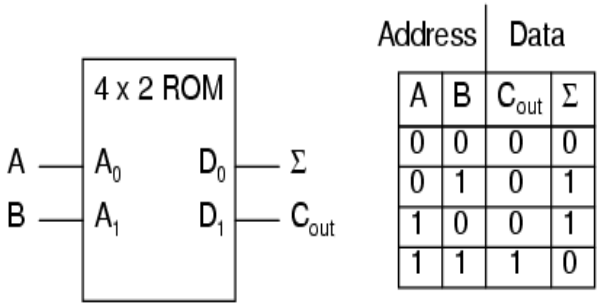
The basic functional model of memory-based multiplier is shown in Fig.1.1. Let A be a fixed coefficient and X be an input word to be multiplied with A. Assuming X to be a positive binary number with word-length L, there can be 2L possible values of X, and accordingly, there can be 2L possible values 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. The product-word A·Xi is stored at the location whose address is the same as Xi for $0 \leq 2L-1$, such that if L-bit binary value of Xi is used as address for the LUT, then the corresponding product value A·Xi 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 but do not find any significant work on LUT optimization for memory based multiplication. 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. Using this 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 this project, two new schemes are proposed for optimization of LUT with lower area- and time-

Copyright @ 2019 **ijearst. All rights reserved.**

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 recoding of stored product word.

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 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:



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

Figure 4X2 ROM

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).

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, 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, without needing any computation whatsoever. This greatly simplifies the transform and inverse calculations. Table lookup can replace any coefficient multiplication or unary operation. 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, 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, table based multiplication yields a much more accurate approximation of the original coefficient.

Proposed System Architecture

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, the LUT size can also be reduced to half, where the product words are recoded as Anti-symmetric pairs.

Copyright @ 2019 **ijearst. All rights reserved.**

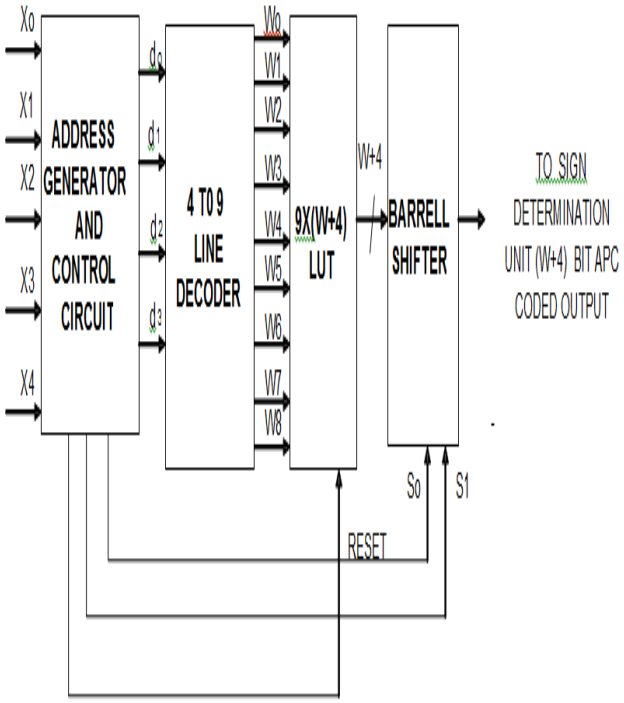


Fig. Proposed LUT Multiplier

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.

Proposed LUT APC Part

The structure and function of the LUT-based multiplier for $L = 5$ using the APC technique 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 $16A$, for $x4 = 1$ or 0, respectively, by the add/subtract cell. Hence, $x4$ is used as the control for the add/subtract cell.

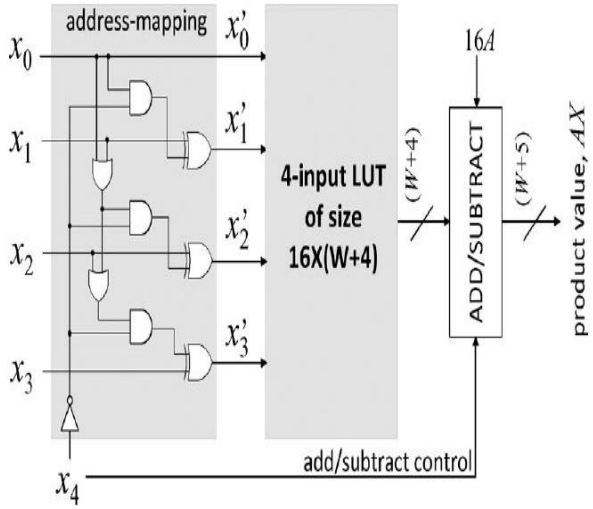


Figure Proposed APC Part

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

Table: 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 $16A$.

Copyright @ 2019 **ijearst. All rights reserved.**

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

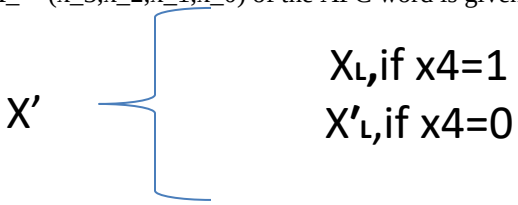
. Let the product values on the second and fourth columns of a row be u and v, respectively. Since one can write

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. This behavior 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. Since the representation of the product is derived from the antisymmetric behavior of the products, we can name it as antisymmetric product code. The 4-bit address X_ = (x_3,x_2,x_1,x_0) of the APC word is given by



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 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.

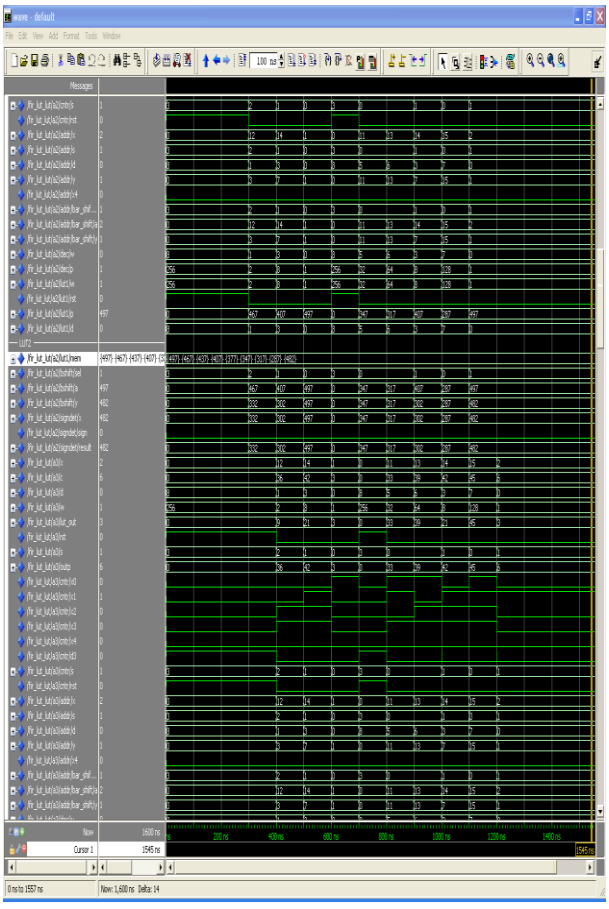
2) A barrel shifter for producing a maximum of (L − 1) left shifts is used to derive all the even multiples of A.

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 are derived by a control circuit.

Table: Stored APC-OMS Words

input $x'_3x'_2x'_1x'_0$	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

RESULT:



Copyright @ 2019 **ijearst. All rights reserved.**

CONCLUSION :

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.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

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

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

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

Copyright @ 2019 **ijearst. All rights reserved.**