M. Mounika Varalakshmi, G.Sravya / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 2, Issue 2,Mar-Apr 2012, pp1472-1476

Innovation on a Memory Based Multiplier

M. Mounika Varalakshmi
Dept. of ECE,
Nova College of Engg. & Tech.,Jangareddygudem, A.P. India

G.Sravya
E.C.E Dept.,
Nova College of Engg. & Tech,
Associate Prof.,
Jangareddygudem, India

ABSTRACT

Several architectures have been reported in the literature for memory-based implementation of DSP algorithms involving orthogonal transforms and digital filters. A common way of implementing constant multiplication is by a series of shift and adds operations. If the multiplier is represented in Canonical Signed Digit (CSD) form, then the number of additions (or subtractions) used will be a minimum by sharing the two most common sub expressions which can be expected to lead to a 33% saving of the number of additions. The multiplier uses look-up-table (LUT) as memory for their computations. However, we do not find any significant work on LUT optimization for memory-based multiplication. A new approach to LUT design was presented, where only the odd multiples of the fixed coefficient are required to be stored which we have referred to as the odd-multiple storage (OMS) scheme. In addition to that the anti symmetric product coding (APC) approach, the LUT size is reduced to half and provides a reduction. When APC approach is combined with the OMS technique, the two's complement operations could be simplified since the input address and LUT output could always be transformed into odd integers, and thus reduces the LUT size to one fourth of the conventional LUT.

The proposed LUT multipliers for word size L = W = 5 and 6 bits are coded in VHDL and synthesized in Xilinx ISE 12.2i. The proposed LUT design for small input sizes can be used for efficient implementation of high-precision multiplication by input operand decomposition. Memory-based computing is well suited for many DSP algorithms, which involve multiplication with a fixed set of coefficients. 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 CSD based multipliers. For 16-bit and 32-bit word sizes, it offers more than 30% and 50% of saving in area–delay product over the corresponding CSD multipliers.

I. INTRODUCTION

Along with the progressive device scaling, semiconductor memory has become cheaper, faster, and more power efficient. Moreover, according to the projections of the international technology road map for semiconductors, embedded memories will have dominating presence in the system-on—chips, which may exceed 90% of the total Soc content It has also been found that the transistor packing density of memory components is not only higher but also increasing much faster than those of logic components.

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. Memory based computing is well suited for many digital signal processing (DSP) algorithms, which involve multiplication with a fixed set of coefficients.

A conventional lookup-table (LUT) based multiplier is shown in Fig. 1, where A is a fixed coefficient, and X is an input word to be multiplied with A. Assuming X to be a positive binary number of word length L, there can be 2^L possible values of X, and accordingly, there can be 2^L possible values of product C = A · X. Therefore, for memory-based, multiplication, an LUT of 2^L words, consisting of pre-computed product values corresponding to all possible values of X, is conventionally used.

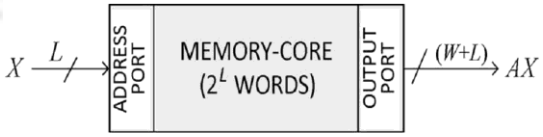


Fig.1.Conventional LUT-based multiplier.

Several architectures have been reported in the literature for memory-based implementation

of DSP algorithms involving orthogonal transforms and digital filters [2]-[8]. However, we do not find any significant work on LUT optimization for memory-based multiplication. I have presented a new approach to LUT design, where only the odd multiples of the fixed coefficient are required to be stored[9],, which we have referred to as the odd-multiple storage (OMS) scheme in this brief. In addition, we have shown that, by the antisymmetric product coding (APC) approach, the LUT size can also be reduced to half.

II. PROPOSED LUT OPTIMIZATIONS

We discuss here the proposed APC technique and its further optimization by combining it with a modified form of OMS.

A. APC for LUT Optimization

For simplicity of presentation, we assume 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 I 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. 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 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, we can have u = 16A − [(v-u)/2] v = 16A + [(v-u)/2].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 desired product could be obtained by Product word = 16A + (sign value) × (APC word) The product value for X = (10000) corresponds to APC value "zero," which could be derived by resetting the LUT output, instead of storing that in the LUT.

B. Modified OMS for LUT Optimization

For [9] the multiplications of any binary word X of size L, with a fixed coefficient A, instead of storing all the 2^L possible values of C=A.X, 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. 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,

- A memory unit of $[(2^L/2) + 1]$ words of (W +L)-bit width is used to store the product values, where the first $(2^L/2)$ words are odd multiples of multiples of A, and the last word is zero.
- A barrel shifter for producing a maximum of (L − 1) left shifts is used to derive all the even multiples of A.
- 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.

The even multiples 2A, 4A, and 8A are derived by left- shift operations of A. Similarly, 6A and 12A are derived by left shifting 3A, while 10A and 14A are derived by left shifting 5A and 7A, respectively. A barrel shifter for producing a maximum of three left shifts could be used to derive all the even multiples of A.

III. Implementation of LUT Using The Optimization Scheme

TABLE I
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 $16A$.

TABLE II
OMS-BASED DESIGN OF THE LUT OF APC WORDS FOR $L = 5$

input X' $x_3'x_2'x_1'x_0'$	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

Here, we discuss the implementation of the LUT-based Multiplier using the proposed scheme, where the LUT is Optimized by a combination of the proposed APC scheme and a modified OMS technique.

A. Implementation of the LUT Multiplier Using APC for L=5

The structure and function of the LUT based multiplier for L = 5 using the APC technique is shown in Fig. 2. It consists of a four input LUT of 16 words to store the APC values of product words. Besides, it consists of an address mapping circuit and an add/subtract circuit. The address mapping circuit generates the desired address and can be optimized to be realized by three OR gates, and a NOT gate.

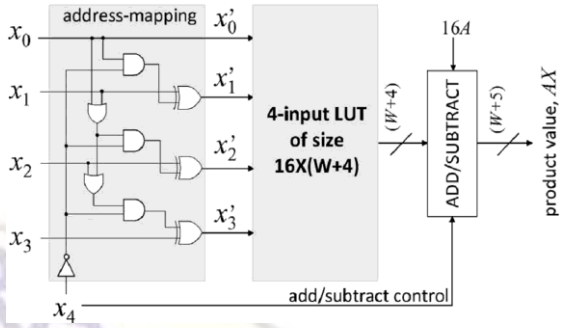


Fig. 2. LUT-based multiplier for L = 5 using the APC technique.

B. Implementation of the Optimized LUT Using Modified OMS

The proposed APC–OMS combined design of the LUT for L = 5 and for any coefficient width W is shown in Fig. 3. It consists of an LUT 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 (s1s0) for the barrel shifter. The pre-computed values of A × (2i + 1) are stored as Pi, for i = 0, 1, 2, . . . , 7, at the eight consecutive locations of the memory array as specified in TableII. The decoder takes the 4-bit address from the address generator and generates nine word-select signals, i.e., {wi, for $0 \le i \le 8$}, to select the referenced word from the LUT.

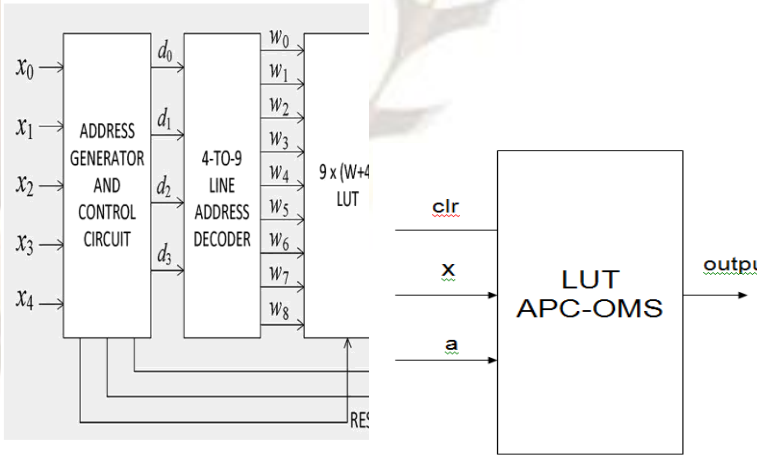


Fig.3.Proposed APC–OMS combined LUT design.

The 4-to-9-line decoder is a simple modification of 3-to-8- line decoder as shown in Fig.4(a). 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 corresponding relations. The RESET signal can alternatively be generated as (d3ANDx4). The control circuit to generate the control word and RESET is shown in Fig. 4(b). The address-generator circuit receives the 5-bit input operand X and maps that onto the 4 bit address word (d3d2d1d0).

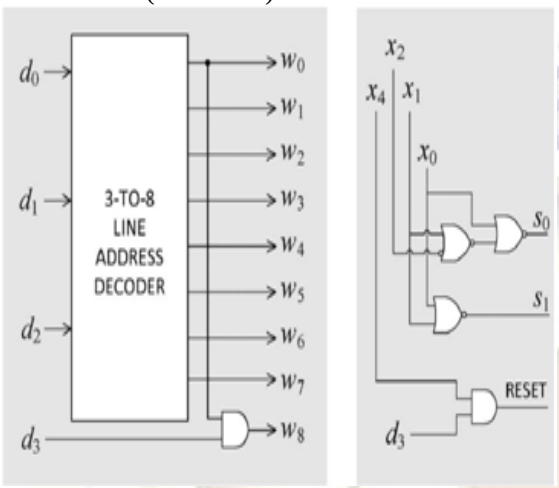


Fig. 4 (a) Four-to-nine-line address-decoder. (b) Control circuit for generation of s0, s1, and RESET.

IV. RESULTS

Top Module for LUT APC–OMS Optimization

Fig.5 LUT APC–OMS Optimization Top Module.

Simulation Results of Top Module:

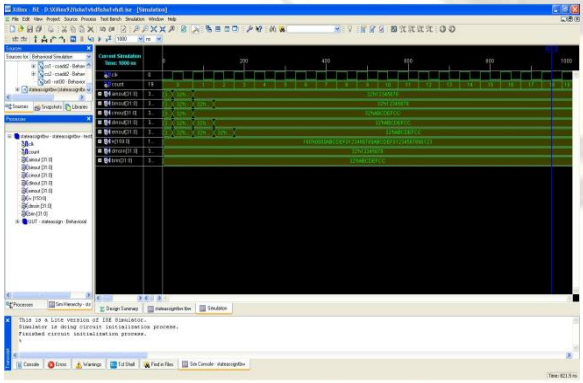


Fig. 6. Simulation Result for 32-bit.

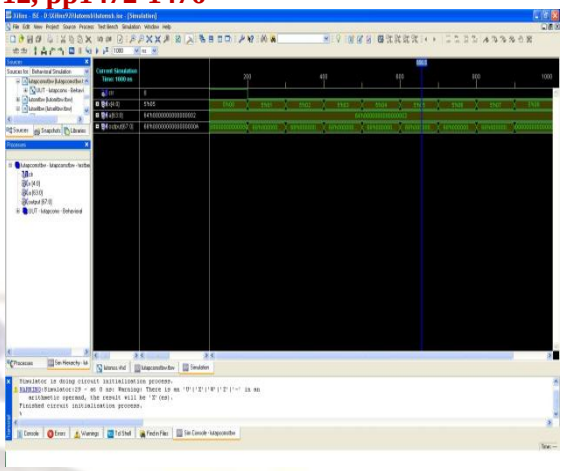
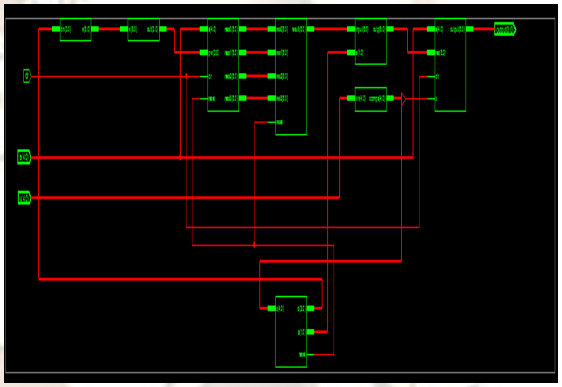


Fig. 7. Simulation Result for 64-bit.

RTL schematic of Top Module



SYNTHESIS REPORT

Source Parameters:

Input File Name : "lutapcoms.prj"
Input Format : mixed
Ignore Synthesis Constraint File : NO

Target Parameters:

Output File Name : "lutapcoms"
Output Format : NGC
Target Device : xc3s500e-5-fg320

Device utilization summary:

Selected Device : 3s500efg320-5
Number of Slices: 72 out of 4656 1%
Number of 4 input LUTs: 131 out of 9312 1%
Number of IOs: 20

Number of bonded IOBs: 20 out of 232 8%

Timing Detail:

All values displayed in nanoseconds (ns) Timing constraint: Default path analysis
Total number of paths / destination ports: 713114 / 9
Delay: 24.886ns (Levels of Logic = 25) Source: x<1> (PAD)
Destination: output<8> (PAD)
Data Path: x<1> to output<8> Gate Net
Cell: in->out fanout Delay Delay Logical Name (Net Name)
Total 24.886ns (16.759ns logic, 8.126ns route) (67.3% logic, 32.7% route)
CPU: 8.00 / 8.86 s | Elapsed : 8.00 / 8.00 s
Total memory usage is 152856 kilobytes

Number of errors:	0	(0 filtered)
Number of warnings:	23	(0 filtered)
Number of infos:	2	(0 filtered)

V. CONCLUSION

The proposed LUT multipliers for word size L = W = 5 and 6 bits are coded in VHDL and synthesized in Xilinx ISE 12.2i. Simulation Part is done in Modelsim 6.3c, where the LUTs are implemented as arrays of constants, and additions are implemented by the Wallace tree and ripple carry array. The CSD-based multipliers having the same addition schemes are also synthesized with the same technology library. we have shown the possibility of using LUT based multipliers to implement the constant multiplication for DSP applications.

VI. FUTURE SCOPE

The LUT multipliers for word size L = W = 8, 16, and 32 bits will be coded in VHDL and synthesizing using Xilinx ISE 12.2i. for the Simulation Part we are going to used Modelsim 6.3c for More Less Area and Less Multiplication Time than CSD.

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.,Oct. 1992.
[2] 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., Aug. 1993
[3] 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.,Sep. 2002.
[4] A.K.Sharma,Advanced Semiconductor Memories:Architectures,Designs, and Applications. Piscataway, NJ: IEEE Press, 2003.
[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., Mar.2005