Modified Reconfigurable Fir Filter Design Using Look up Table

R. Dhayabarani, Assistant Professor.
M. Poovitha, PG scholar,

V.S.B Engineering College, Karur, Tamil Nadu.

Abstract **-** Memory based structures are used in many kind of digital signal processing (DSP) applications, such as which involve in multiplication with a fixed set of coefficients. Memory-based structures are better performance in area minimization compare with multiply-accumulate structures and have many other advantages like reduced latency since the memory-access-time is much shorter than the usual multiplication-time compared to the conventional multipliers. The multiplier uses LUT's as memory for their computations. The anti-symmetric product coding (APC) and odd-multiple-storage (OMS) techniques were proposed for look-up-table (LUT) design. Memory-based structure such as APC and OMS techniques are used for efficient Multiplication. Hence, the combination of these two techniques provides reduction in LUT size to one fourth of the conventional Look up Table (LUT). The proposed LUT multiplier is designed based on Xilinx 9.2 synthesis tool and the result has shown as less area and reduced-latency implementation (less number of gates and less combinational delay) compared to conventional LUT multiplier.

Keywords **– Digital Signal Processing (DSP), Look up Table (LUT), Anti-Symmetric Product Coding (APC), Odd Multiple Storage (OMS), Xilinx 9.2 synthesis tool.**

I.INTRODUCTION

Finite-Impulse Response (FIR) digital filter is widely used as a basic tool in various signal processing and image processing applications [1]. The order of an FIR filter primarily determines the width of the transition-band, such that the higher the filter order, the sharper is the transition between a pass-band and adjacent stop-band. Many applications in digital communication (channel equalization, frequency channelization), speech processing (adaptive noise cancelation), seismic signal processing (noise elimination), and several other areas of signal processing require large order FIR filters [2], [3]. Since the number of multiply-accumulate (MAC) operations required per filter output increases linearly with the filter order, real-time implementation of these filters of large orders is a challenging task. Several attempts have, therefore, been made and continued to develop low-complexity dedicated VLSI systems for these filters [4]–[7].

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 $2L$ possible values of X, and accordingly, there can be $2L$ possible values of product $C = A \cdot X$. Therefore, for memory-based multiplication, an LUT of 2^L words, consisting of precomputed product values corresponding

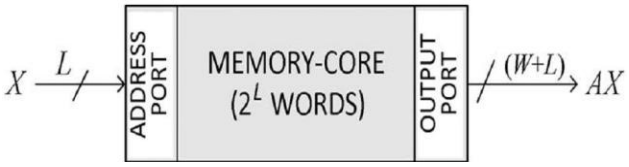

Fig.1.Conventional LUT-based multiplier

to all possible values of X, is conventionally used. The product word $A \cdot Xi$ is stored at the location Xi for $0 \leq Xi \leq 2L - 1$, such that if an L-bit binary value of Xi is used as the address for the LUT, then the corresponding product value $A \cdot Xi$ 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 [8]–[14]. However, we do not find any significant work on LUT optimization for memory-based multiplication. Recently, we have presented a new approach to LUT design, where only the odd multiples of the fixed coefficient are required to be stored [15], which we have referred to as the *odd-multiple-storage* (OMS) scheme in this brief. In addition,

we have shown that, by the *anti-symmetric product coding* (APC) approach, the LUT size can also be reduced to half, where the product words are recoded as anti-symmetric pairs [14].

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.1 However, the OMS technique in [15] cannot be combined with the APC scheme in [14], since the APC words generated according to [14] are odd numbers. Moreover, the OMS scheme in [15] does not provide an efficient implementation when combined with the APC technique. In this brief, we therefore present a different form of APC and combined that with a modified form of the OMS scheme for efficient memory based multiplication.

II. PROPOSED LUT OPTIMIZATIONS FOR MEMORY-BASED MULTIPLICATION

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.2 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 32*A*. Let the product values on the second and fourth columns of a row be *u* and *v*, respectively. Since one can write

$$U = \left[\frac{(U+V)}{2} - \frac{(V-U)}{2}\right] \text{ and } V = \left[\frac{(U+V)}{2} + \frac{(V-U)}{2}\right]$$

We can have

$$U = 16A - [(V-U)/2], V = 16A + \left[\frac{(V-U)}{2}\right]$$

The product values on the second and fourth columns of Table 1 therefore have 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 anti-symmetric behavior of the products, we can name it as anti-symmetric product code. The 4-bit address X'= (x'3x'2x'1x'0) of the APC word is given by

$$X = \begin{cases} X_L, if\ X_4 = 1 \\ X'_L, if X_4 = 0 \end{cases}$$

Where XL = (x3x2x1x0) is the four less significant bits of X and X'L is the two's complement of XL .The desired product could be obtained by adding or subtracting the stored value ($v - u$) to or from the fixed value 16*A* when x4 is 1 or 0, respectively, i.e.,
Product word = 16*A* + (sign value) × (APC word)

Where sign value = 1 for *x*4 = 1 and sign value = −1 for *x*4 = 0. 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 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.

In Table II, we have shown that, at eight memory locations, the eight odd multiples, $A \times (2i + 1)$ are stored as *Pi*, for $i = 0, 1, 2 \ldots 7$. 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*.

As required by the word to be stored for X = (00000) is not 0 but 16A, which we can obtain from A by four left shifts using a barrel shifter. However, if 16*A* 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

Input, X	product values	Input, X	product values	address $x'_3x'_2x'_1x'_0$	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 I
APC Words for different input values for $L = 5$

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

TABLE II

X = (00000), so that the product 16A can be derived by three arithmetic left shifts.

The product values and encoded words for input words X = (00000) and (10000) are separately shown in Table III For X = (00000), the desired encoded word 16A is derived by 3-bit left shifts of 2A [stored at address (1000)]. For X = (10000), the APC word "0" is derived by resetting the LUT output, by an active-high RESET signal given by

$$\text{RESET} = \overline{(x_0 + x_1 + x_2 + x_3)}.x_4$$

It may be seen from Tables II and III that the 5-bit input word X can be mapped into a 4-bit LUT address ($d3d2d1d0$), by a simple set of mapping relations

$$di = x''_{i+1} \text{ , for } i = 0, 1, 2 \text{ and } d3 = \overline{x''0} \qquad (5)$$

where X" =(x"3x"2x"1x"0) is generated by shifting-out all the leading zeros of $X\_$ by an arithmetic right shift followed by address mapping, i.e.,

$$x'' = \begin{cases} Y_L, if\ x4 = 1 \\ Y'_L, if\ x4 = 0 \end{cases}$$

where YL and Y_L are derived by circularly shifting-out all the leading zeros of XL and X_L, respectively.

TABLE II
OMS-Based design of the LUT of APC words for $L = 5$

input X $x_4x_3x_2x_1x_0$	product values	encoded word	stored values	# of shifts	address $d_3d_2d_1d_0$
1 0 0 0 0	$16A$	0	– – –	– –	– – –
0 0 0 0 0	0	$16A$	$2A$	3	1 0 0 0

TABLE III
Products and Encoded words for X = (00000) and (10000)

III. IMPLEMENTATION OF THE LUT-BASED MULTIPLIER USING THE PROPOSED LUT OPTIMIZATION SCHEME

In this section, 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 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 (x'3x'2x'1x'0). A straightforward implementation of

address mapping can be done by multiplexing XL and 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 Fig.2 Note that the RESET can be generated by a control circuit. The output of the LUT is added with or subtracted from16A, for x4= 1or 0, respectively, by the add/subtract cell. Hence, x4 is used as the control for the add/subtract cell.

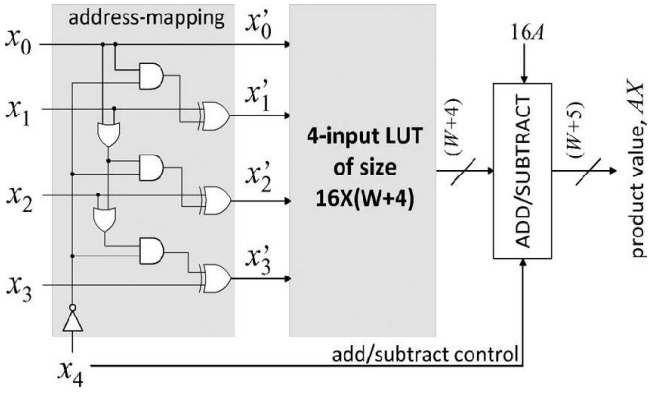

Fig. 2. LUT-based multiplier for $L = 5$ using the APC technique.

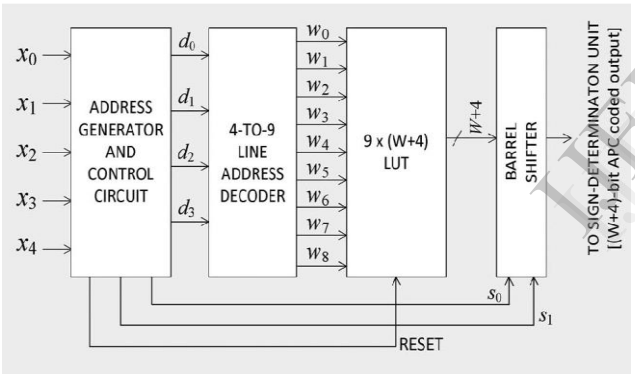

Fig. 3. Proposed APC–OMS combined LUT design for the multiplication of W -bit fixed coefficient A with 5-bit input X.

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 (s_1s_0) for the barrel shifter.

The precomputed values of $A \times (2i + 1)$ are stored as P_i, for $i = 0, 1, 2, \ldots, 7$, at the eight consecutive locations of the memory array, as specified in Table II, while $2A$ is stored for input $X = (00000)$ at LUT address "1000," as specified in Table III. The decoder takes the 4-bit address from the address generator and generates nine word-select signals, i.e., $\{w_i$, for $0 \le i \le 8\}$, to select the referenced word from the LUT. 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 s_0 and s_1 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 relations

$$S_0 = \overline{X_0 + \overline{X_1 + \overline{X_2}}}$$
$$S_1 = \overline{X_0 + X_1}$$

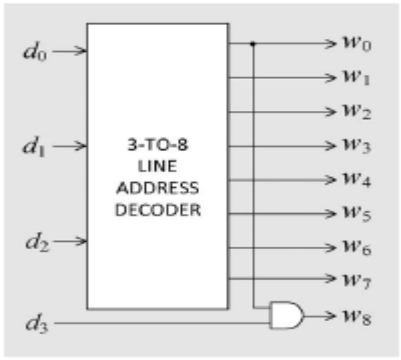

Fig.4 (a) Four- to-nine line address-decoder

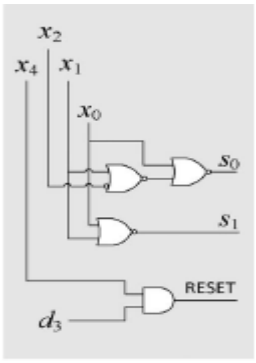

Fig.4 (b) Control signal generation

Note that (s_1s_0) is a 2-bit binary equivalent of the required number of shifts specified in Tables II and III. The RESET sig-nal given by (4) can alternatively be generated as (d_3 AND x_4). 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 $(d_3d_2d_1d_0)$, according to (5) and (6). A simplified address generator is presented later in this section.

IV REALIZATION OF DIGITAL FIR FILTER USING PROPOSED LUT BASED MULTIPLIER

The Realization of digital FIR filter using proposed LUT multiplier is done by using direct form realization structure of digital FIR filter. This equation is applied to FIR filter design with output sequence y[n] in terms of its input sequence x[n]:

$$y(n) = \sum_{k=0}^{N-1} h(k).x(n-k)$$

Where x[n] is the input signal, y[n] is the output signal, h[k] is the coefficients of FIR filter frequency response, and N is the filter order. The direct form realization of digital FIR filter the input X is delayed and given to multiplier each multiplier gives products corresponding to different filter coefficients and all these products are accumulated and give fir filter output. The proposed LUT multiplier is used in the above Fig. 3 in which each multiplier is having fixed filter coefficients ,the inputs are delayed and given to this LUT multiplier .A memory-unit of (2L/2) words of (W+L) bit width is used to store all the odd multiples of filter coefficient. The L-bit input word is mapped to (L-1) -bit LUT address by an encoder. The barrel-shifter is derive all the even multiples of filter coefficient. The required control-bits for the barrel shifter are derived by Control-circuit to perform the necessary shifts of the LUT output. RESET signal is generated by the same control circuit to reset the LUT output when X = 0. There by corresponding products which are stored in the LUT of particular input given to LUT based multiplier based circuit in Fig. 3 are obtained. These products are finally accumulated and give as FIR filter output based on number of taps for a given filter. The FIR filter is realized using proposed LUT based multiplier is shown in Fig. 5.

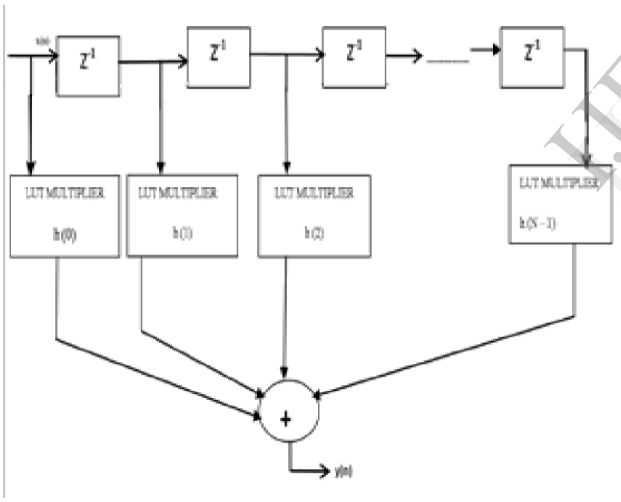

Fig.5 Realization of digital FIR filter using proposed LUT based multiplier

V. COMPARATIVE ANALYSES

When comparing the Conventional LUT multiplier with Proposed LUT-multiplier-based design by synthesizing using Xilinx and LEONARDO SPECTRUM tool given results that the memory based structure Proposed LUT based multiplier is having high-throughput, reduced-latency implementation, and occupying less area.

VI. RESULTS

Conventional LUT and Proposed LUT multipliers and respective filters are designed and synthesized using Xilinx gives that number of gates used and the combinational delays are less for the LUT memory based multiplier .Therefore this memory structure is having less area and better latency of implementation. The results are shown in table IV and simulation result for Proposed LUT multiplier.

Logic Utilization	**Conventional LUT**	**Proposed LUT**
Minimum Frequency	9.321ns	9.168ns
Maximum Frequency	107.290MHz	109.081MHz
output required time after clock	13.429ns	6.347ns
Number of Slice Flip Flop	43 out of 13,824 1%	37 out of 13,824 1%
Number of 4 input LUTs	37 out of 13,824 1%	18 out of 13,824 1%
Total Equivalent gate cont for design	806	665
Additional JTAG gate cont for JOBs	5,904	1,008
Number of bonded IOBs	122 out of 510 23%	20 out of 510 3%

TABLE IV
Synthesis results of Proposed LUT based, Conventional Multipliers

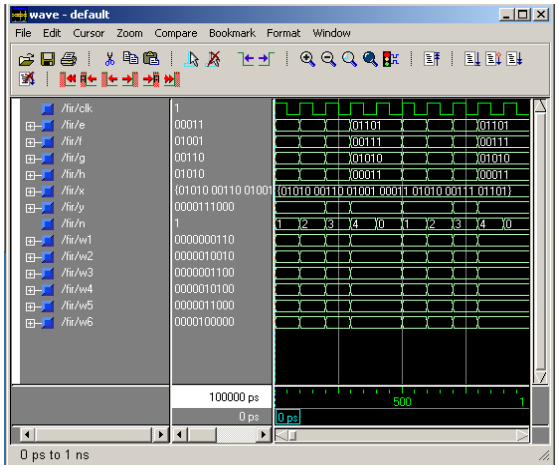

Fig.6 Wave form for Proposed LUT Multiplier using FIR Filter

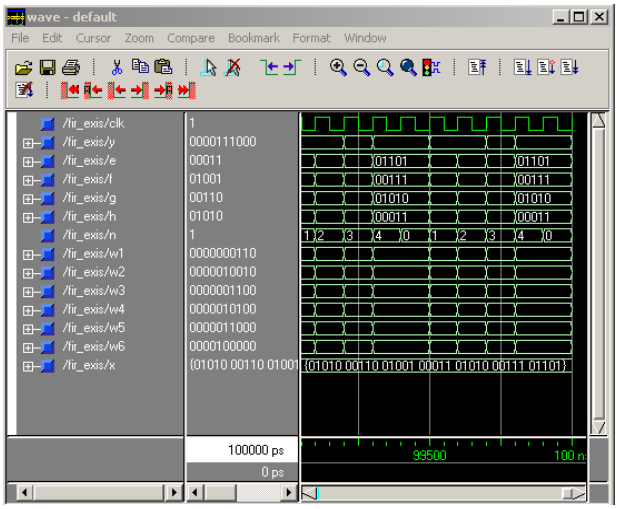

Fig.7 Waveform for Conventional LUT Multiplier using FIR Filter

VII. CONCLUSION

The proposed LUT-multiplier-based design of FIR filter is more efficient than the previous Conventional LUT based design of FIR filter in terms of area complexity for a given throughput and lower latency of implementation. Finally it is proved to be a low-complexity dedicated VLSI system for filters.

VII. FUTURE ENHANCEMENTS

In future CSE algorithm is used to improve the performance of APC-OMS LUT multiplier in terms of reduced area and latency is efficiency of the memory based LUT multiplier.

REFERENCES

[1] J. G. Proakis, D. G. Manolakis, "Digital Signal Processing:Principles, Algorithms and Applications". Upper SaddleRiver, NJ: Prentice- Hall, 1996.

[2] G. Mirchandani, R. L. Zinser Jr., and J. B. Evans, "A new adaptive noise cancellation scheme in the presence of crosstalk [speech signals]," IEEE Trans. Circuits Syst. II, Analog. Digit. Signal Process. vol. 39, no. 10, pp. 681–694, Oct. 1995.

[3] D. Xu and J. Chiu, "Design of a high-order FIR digital filtering and variable gain ranging seismic data acquisition system," in Proc. IEEE Southeastcon'93, Apr. 1993, p. 6.

[4] H. H. Dam, A. Cantoni, K. L. Teo, and S. Nordholm, "FIR variable digital filter with signed power-of-two coefficients," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 54, no. 6, pp. 1348–1357, Jun. 2007.

[5] R. Mahesh and A. P. Vinod, "A new common sub expression elimination algorithm for realizing low-complexity higher order digital filters," IEEE Trans. Computer-Aided Ded. Integr. Circuits Syst., vol. 27, no. 2, pp. 217–229, Feb. 2008.

[6] K. K. Parhi, VLSI Digital Signal Procesing Systems: Design and Implementation. New York: Wiley, 1999.

[7] H. H. Kha, H. D. Tuan, B.-N. Vo, and T. Q. Nguyen, "Symmetric orthogonal complex-valued filter bank design by semidefinite programming," IEEE Trans. Signal Process., vol. 55, no. 9, pp. 4405–4414, Sep. 2007.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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

[14] 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.

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