Hardware Efficient Fast FIR Filter Based on Karatsuba Algorithm

Evangelos Kyritsis
Dept. of Electrical and Computer Engineering
National Technical University of Athens
15780 Athens, Greece
evkiritsis@gmail.com

Kiamal Pekmestzi
Dept. of Electrical and Computer Engineering
National Technical University of Athens
15780 Athens, Greece
pekmes@cs.ntua.gr

***Abstract*—In this work, an efficient implementation of a programmable Finite Impulse Response (FIR) filter based on the use of the Karatsuba Multiplication Algorithm (KMA) is presented. In this FIR filter circuit, a parallel, Modified Booth (MB) pre-encoded, Carry-Save (CS) Wallace tree multiplier is used as a building block. The KMA is a fast divide and conquer algorithm for the multiplication of large numbers. As a result, the proposed circuit is highly efficient in terms of speed, area and power in comparison with the Conventional FIR filter architecture. Simulations of FIR filters in transposed form made over standard-cell implementation based on a Faraday 90nm technology show an average reduction of about 15% in the delay, 9% in area and 17% in power.**

Keywords—Finite impulse response (FIR) filter; Karatsuba algorithm; VLSI Design

I. INTRODUCTION

The FIR filter is one of the most important digital signal processing (DSP) applications. Various techniques have been used in order to improve the efficiency of these circuits.

The Karatsuba Multiplication Algorithm (KMA) [1-2] has been applied in the implementation of FIR filters in [3]. The Karatsuba algorithm reduces a $2N\times2N$ bit multiplication to a set of $(N+1)\times(N+1)$ and $N\times N$ bit multiplications and additions by splitting the $2N$ bit input operands and generating some auxiliary variables. When the Karatsuba algorithm is applied to the FIR filter equation, a parallel architecture is obtained giving speed and area savings especially in case of high dynamic range filters. In the implementation of FIR filters in [3] all the intermediate results are in two's complement form and it does not utilize the Carry-Save (CS) form. In this paper, we propose an architecture based on a modified formulation of KMA. Also in the proposed implementation CS arithmetic is used in order to decrease the critical path and to speed-up the calculations.

The paper is organized as follows: The multiplication unit, the Conventional FIR filter and the Karatsuba FIR filter architecture are presented in Section II, III and IV respectively. The implementation results are presented in Section V, while concluding remarks are discussed in section VI.

II. MULTIPLICATION UNIT (MU)

Multiplication is a very significant arithmetic operation for FIR filters. In our designs a parallel, Modified Booth (MB) CS Wallace tree multiplier has been designed and used in the FIR filter circuits [4]. In order to increase the efficiency we consider that the coefficients are pre-encoded and stored in MB form. The architecture of the Multiplication Unit (MU) used in implementation of the FIR filter is shown in Figure 1. The MU contains a Partial Product Generator unit (PPG), and a Wallace tree for adding the partial products. For the Wallace tree, the Synopsys *DesignWare* IP (DW02_tree) has been used [5]. The standard MB encoder hasn't been used in the specific implementation because operand B is pre-stored in MB encoding.

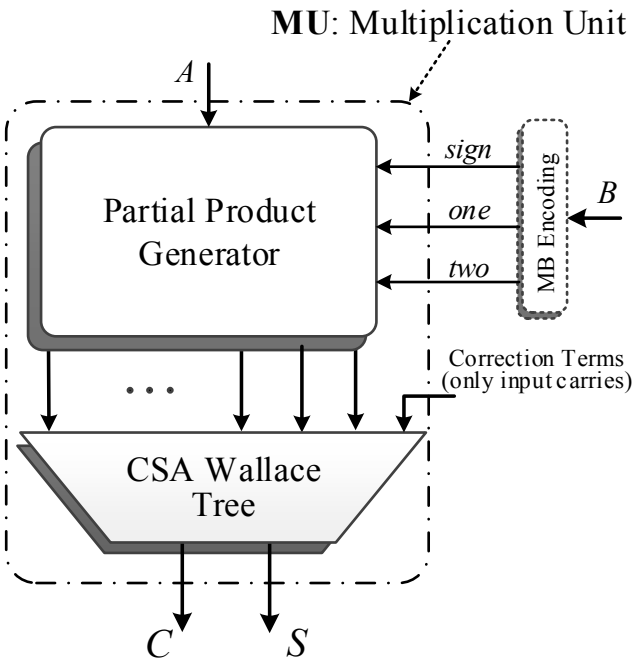

Fig. 1: Multiplication Unit

978-1-4673-9680-6/16/$31.00 ©2016 IEEE

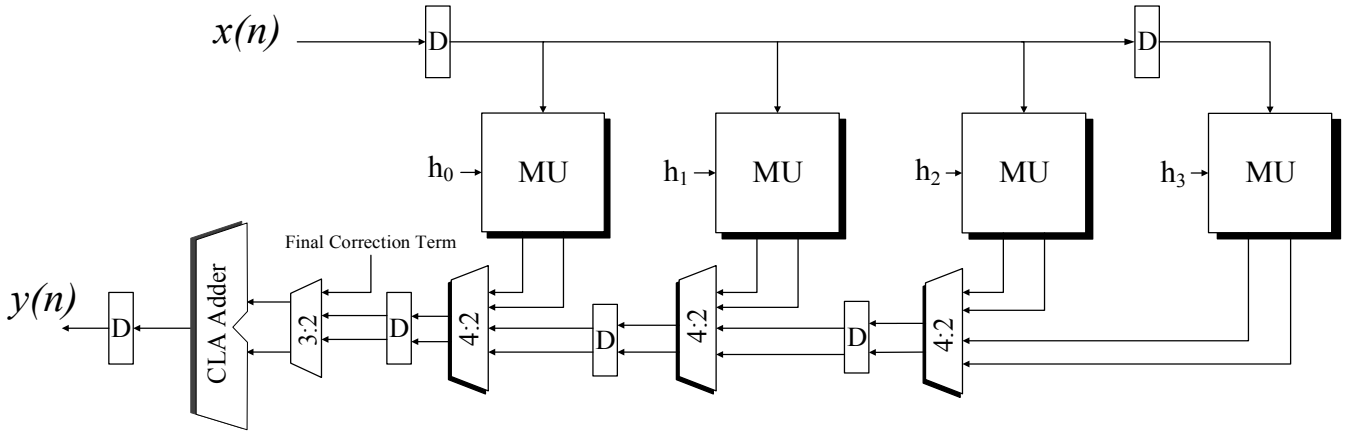

Fig. 2: Conventional FIR filter structure in transposed form

The addition of a correction term is needed in order to obtain the right result of the multiplication. The necessity of the correction term arises for two reasons. Firstly, the necessary sign extension of all partial products up to the word-length of the result *(4N)*. Secondly, the input carries which are needed for the conversion of the partial products from one's complement to two's complement representation, when the according MB digit *sign* is one (*sign[i]=1)*.

The input carries are added separately into every MU of the filter. On the other hand, the part of the correction term concerning the sign extension has a predetermined form and can be isolated in order to form a final term for all MUs of the filter. Consequently, the addition of only one final correction term is needed instead of T separate terms, giving some area savings.

III. CONVENTIONAL FIR FILTER

The transfer function for a FIR filter in time domain is shown in (1).

$$y(n)=\sum_{k=0}^{T-1}h(k)\cdot x(n-k) \tag{1}$$

Variable *T, h(k), x(n)* and *y(n)* in (1) represent the filter order, the filter coefficients, input and output respectively.

The structure of the Conventional FIR filter in transposed form for *T*=4, is shown in Figure 2. This architecture is identical when extended for more taps. We have designed the circuit using basic delay units (D), the multiplier shown in Figure 1 (MU), the Synopsys *DesignWare* IP DW01_add (CLA) and DW02_tree (4:2 and 3:2 units) [5]. The final correction term contains the ones from the sign extension of all partial products.

IV. KARATSUBA FIR FILTER

Let us consider two numbers: *a* and *b* both of 2*N* bits. Each number can be divided to two subwords of *N* bits, as follows:

$$a=a_H\cdot 2^N+a_L \quad \text{and} \quad b=b_H\cdot 2^N+b_L$$

The product *P* of these numbers is given by the next relation:

$$P=a\cdot b=a_H\cdot b_H\cdot 2^{2N}+\left(a_L\cdot b_H+a_H\cdot b_L\right)\cdot 2^N+a_L\cdot b_L \tag{2}$$

Where:

$$P_H=a_H\cdot b_H$$

$$P_M=a_L\cdot b_H+a_H\cdot b_L$$

$$P_L=a_L\cdot b_L$$

We compute also the next quantity:

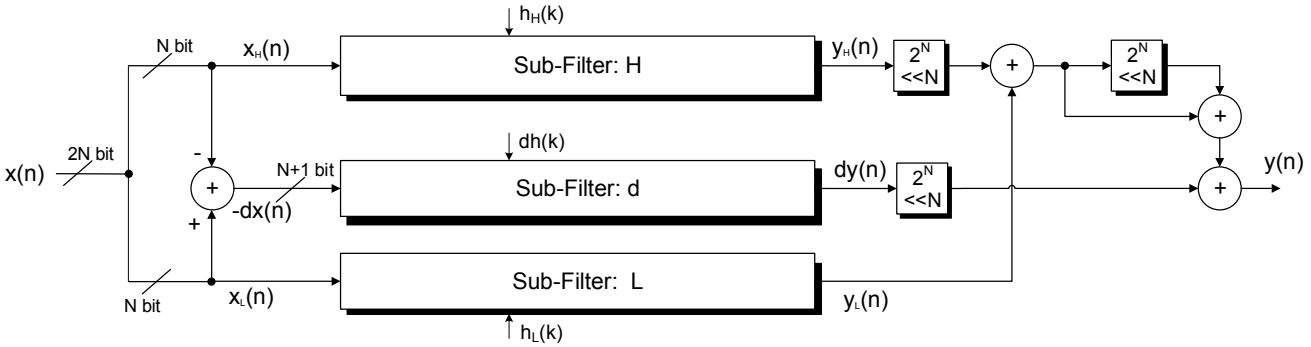

Fig. 3: Karatsuba FIR filter

$$dP = da \cdot db = (a_H - a_L) \cdot (b_H - b_L) = P_H + P_L - P_M \quad (3)$$

Consequently the term P_M is given by the following relation:

$$P_M = P_H + P_L - dP \quad (4)$$

According to (2) and (4) for the product P holds that:

$$P = (P_H \cdot 2^N + P_L) \cdot (2^N + 1) - dP \cdot 2^N \quad (5)$$

According to (5), the computation of the product *P* can be simplified in the calculation of three smaller products plus some operations of shifts, additions and substitutions. We applied this formula in order to split the original filter into three filters of reduced dynamic range, working in parallel. The relations which describe these sub-filters are the following:

$$y_H(n) = \sum_{k=0}^{T-1} h_H(k) \cdot x_H(n-k) \quad (6)$$

$$y_L(n) = \sum_{k=0}^{T-1} h_L(k) \cdot x_L(n-k) \quad (7)$$

$$dy(n) = \sum_{k=0}^{T-1} dh(k) \cdot dx(n-k) \quad (8)$$

The filter output becomes:

$$y(n) = (y_H(n) \cdot 2^N + y_L(n)) \cdot (2^N + 1) - dy(n) \cdot 2^N \quad (9)$$

Figure 3 shows the architecture of the Karatsuba FIR filter. The architecture of the three sub-filters is identical with that shown in Figure 2, except the output which is in CS representation. The Synopsys *DesignWare* IP DW01_sub [5] has been used for the calculation of the quantity $-dx = x_L - x_H$ as needed. For the output conversion three hardwired shifters and two CLA (DW01_add) have been used.

V. IMPLEMENTATION RESULTS

Both Karatsuba and Conventional FIR filters were implemented with $T \in \{16,32\}$ and $2N \in \{16, 32\}$ for the representation of $h(k)$ and $x(n)$ in order to be compared in terms of delay, area and power.

The FIR filters have been described with Verilog. For the functional simulation we used *ModelSim*. Then, the designs were implemented based on *Faraday* 90nm (FSD0A_A_GENERIC_CORE_TT1V25C) technology using Synopsys tools. The synthesis has been made with *Design Compiler*. The synthesis constraints have been set for optimal results, without keeping the hierarchy of the designs. For the *Static Time Analysis* (STA) we used *PrimeTime*. The post-synthesis simulation has been made again in *ModelSim*. Finally, the mean power consumption has been obtained using *PrimePower*.

TABLE I: Delay of Conventional and Karatsuba architecture

Delay Conventional (nsec)		
#bits (2N)	**#taps (T)**	
	16	**32**
16	0.98	1.07
32	1.28	1.55
Delay Karatsuba (nsec)		
#bits (2N)	**#taps (T)**	
	16	**32**
16	0.82	0.90
32	1.08	1.37

TABLE II: Area of Conventional and Karatsuba architecture for $T_{CONV,min}$

Area Conventional (mm²)		
#bits (2N)	**#taps (T)**	
	16	**32**
16	0.164	0.301
32	0.460	0.924
Area Karatsuba (mm²)		
#bits (2N)	**#taps (T)**	
	16	**32**
16	0.145	0.283
32	0.409	0.860

TABLE III: Power of Conventional and Karatsuba architecture for $T_{CONV,min}$

Power Conventional (mW)		
#bits (2N)	**#taps (T)**	
	16	**32**
16	132.9	222.1
32	264.6	659.6
Power Karatsuba (mW)		
#bits (2N)	**#taps (T)**	
	16	**32**
16	106.2	197.1
32	213.7	550.2

The delays for the two implementations are shown in Table I while in Table IV the delay savings are shown. Let T_{MU} be the propagation delay of the multiplication unit and T_{CSA} the propagation delay of the CSA 4:2 adder. The critical path of the Conventional implementation is $T_{MU}(2N)+T_{CSA}(4N+\log_2 T)$, while for the Karatsuba implementation is $T_{MU}(N+1)+T_{CSA}(2N+2+\log_2 T)$. The critical path of Karatsuba implementation corresponds to the sub-filter d.

The implementations have been compared in terms of area and power for a clock period equal to the critical delay of the Conventional implementation shown in Table I. Table II and Table V show the area of the two implementations and area savings respectively. For T=32 the area savings are lower than for T=16 due to the large number of used registers for the Karatsuba implementation. In particular, the number of registers used for the Karatsuba implementation is $2T(6N+2+3\log_2 T)$, while for the Conventional implementation is $2T(4N+\log_2 T)$. The number of registers is doubled because CS arithmetic is used in both FIR filter implementations.

The total power consumption results for the two implementations are shown in Table III and the total power savings in Table VI. Full timing simulations have shown that

the Karatsuba implementation is more efficient than the Conventional implementation in terms of power, mainly due to the reduced net switching power as well as its lower leakage power as shown in Table VIII and Table IX. Table VII show the cell internal power savings which are the lowest among the three components of the total power consumption.

TABLE IV: Delay savings

Delay [KA/CONV-1] (%)		
#bits (2N)	**#taps (T)**	
	16	**32**
16	-16.33	-15.89
32	-15.63	-11.61

TABLE V: Area savings

Area [KA/CONV-1] (%)		
#bits (2N)	**#taps (T)**	
	16	**32**
16	-11.59	-5.98
32	-11.09	-6.93

TABLE VI: Power savings

Power [KA/CONV-1] (%)		
#bits (2N)	**#taps (T)**	
	16	**32**
16	-20.09	-11.26
32	-19.24	-16.59

TABLE VII: Cell internal power savings

Cell Internal Power [KA/CONV-1] (%)		
#bits (2N)	**#taps (T)**	
	16	**32**
16	-8.40	+2.4
32	-9.95	-12.73

TABLE VIII: Net switching power savings

Net Switching Power [KA/CONV-1] (%)		
#bits (2N)	**#taps (T)**	
	16	**32**
16	-40.63	-30.33
32	-30.62	-19.26

TABLE IX: Cell leakage power savings

Cell Leakage Power [KA/CONV-1] (%)		
#bits (2N)	**#taps (T)**	
	16	**32**
16	-34.67	-30.72
32	-29.22	-29.86

As can be seen from the above tables there are significant savings of the proposed Karatsuba based FIR filters in terms of *Area*, *Delay* and *Power* compared to the conventional ones. Specifically an average reduction of about 15% in the delay, 9% in area and 17% in power have been obtained.

The proposed design is also compared with [3] in Table X. Booth designs are based on Karatsuba Algorithm. The delay and area of the proposed design are significantly lower than [3]. However our design is more energy consuming which is based on the following reasons:

- The design in [3] is based on a different Standard Cell Library with low power features.
- The proposed design is based on Carry-Save Arithmetic that duplicates the number of registers and increases the switching activity of the circuit.

In the case of high speed applications the proposed design is almost 40% faster than [3]. Also in implementations where area occupation is dominant factor (e.g. FPGAs)[6] the proposed design is 100% more efficient.

Considered that power is critical factor our future effort will focus on improving the Karatsuba based FIR filters architecture to share not only low area and high speed operation but also low energy consumption.

TABLE X: Comparison of the proposed design with [3]

16 bit		**Delay (ns)**	**Area (mm^2)**	**Power (mW)**
16 Taps	**Proposed**	0.82	0.145	106.2
	[3]	1.47	0.303	32
32 Taps	**Proposed**	0.90	0.283	197.1
	[3]	1.50	0.584	59

VI. CONCLUSIONS

We have presented an efficient FIR filter based on the Karatsuba formula. The architecture we have designed is composed by three sub-filters of reduced dynamic range. A parallel, MB pre-encoded, CS Wallace tree multiplier was designed and used as a building block. We choose to use CS arithmetic in order to enhance delay savings. The proposed Karatsuba design show an improved performance, a smaller circuit area and lower power consumption, compared with the Conventional transposed FIR filter.

REFERENCES

[1] C. Eyupoglu, "Performance Analysis of Karatsuba Multiplication Algorithm for Different Bit Lenghts," Procedia – Social and Behavioral Sciences, vol. 195, pp. 1860-1864, Jul. 2015.

[2] A. Karatsuba and Y. Ofman, "Multiplication of multidigit numbers on automata," Doklady Akademii Nauk SSSR, vol. 145, no. 2, pp. 293-294, 1962.

[3] P. Albicocco, G. C. Cardarilli, S. Pontarelli, M. Re, "Karatsuba Implementation of FIR filters," in ASILOMAR, Pacific Groove, CA, 2012, pp.1111-1114.

[4] W.-C. Yeh and C.-W. Jen, "High-speed Booth encoded parallel multiplier design," IEEE Trans. Comput., vol. 49, no. 7, pp. 692-701, Jul. 2000.

[5] DesignWare Building Blocks Data sheet, available at: www.synopsys.com/dw/doc.php/doc/dwf/datasheets/dw02 mult.pdf

[6] Florent de Dinechin, Bogdan Pasca, Large Multipliers, "With Fewer DSP Blocks," International Conference on Field Programmable Logic and Applications, 2009.