International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 11 (2015) pp. 27477-27486

http://www.ripublication.com

Implementation Of Fir Filter Using Booth Multiplier And Carry Skip Adder

Kondeti Lakshmi and K.Chitambara Rao

M.Tech (VLSI SD), AITAM, Tekkali
laxmi.kondeti@gmail.com
Assoc Prof. (PhD) ECE, AITAM, Tekkali
rao.chiddubabu@gmail.com

Abstract

Digital filters are used extensively in all areas of electronic industry in which FIR filters are most widely used. This paper presents a design of FIR filter by using BOOTH multiplier and CARRY SKIP adder. Optimizing the speed and area of a multiplier is a major design issue. The area and speed are the conflicting constraints because the faster speed results in the larger area. The faster execution speed and smaller area are the important factors in designing the DSP systems. In this paper the multiplier considered is the Booth Multiplier. And the adder is the carry skip adder. So mentioned multiplier is combined with the adders in the design of FIR filter so that they occupy less amount of space when compared with the normal multiplier. This criterion is very important in the fabrication of the chips and the high performance system requires components which are as small as possible. Finally the simulation and synthesis results obtained are compared with the results of previous methods and to say which one is the best.

Keywords - BOOTH MULTIPLIER, CARRY SKIP ADDER, XILINX ISE, VERILOG

I. INTRODUCTION

Filter is a frequency selective network. It passes a band of frequencies while attenuating the others. Filters are classified as analog and digital depending on nature of inputs and outputs. Filters are further classified as finite impulse response and infinite impulse response filters depending on impulse response. Analog filters can be passive or active. Passive filters use only resistors, capacitors, and inductors. Passive designs tend to be used where there is a requirement to pass significant direct current

(about 1mA) through low pass or band stop filters. They are also used more in specialized applications, such as in high-frequency filters or where a large dynamic range is needed. (Dynamic range is the difference between the background noise floor and the maximum signal level.) Also, passive filters do not consume any power, which is an advantage in some low-power systems. The main disadvantage of using passive filters containing inductors is that they tend to be bulky. This is particularly true when they are designed to pass high currents, because large diameter wire has to be used for the windings and the core has to have sufficient volume to cope with the magnetic flux. Very simple analog low pass or high pass filters can be constructed from resistor and capacitor (RC) networks. The RC filter works because the capacitor reactance reduces as the frequency increases. At low frequencies the reactance of the capacitor is very high and the output voltage is almost equal to the input, with virtually no phase difference. At the cut off frequency, the resistance and the capacitive reactance are equal and the filter's output is l / f i of the input voltage, or -3 dB. At this frequency the output will not be in phase with the input , it will lag by 45" due to the influence of the capacitive reactance. At frequencies above the 3 dB attenuation point, the output voltage will reduce further. The rate of attenuation will be 6 dB per doubling of frequency (per octave). As the frequency rises, the capacitive reactance falls and the phase shift lag approaches 90".

Digital filters are used extensively in all areas of electronic industry. This is because digital filters have the potential to attain much better signal to noise ratios than analog filters and at each intermediate stage the analog filter adds more noise to the signal, the digital filter performs noiseless mathematical operations at each intermediate step in the transform. The digital filters have emerged as a strong option for removing noise, shaping spectrum, and minimizing inter-symbol interference in communication architectures. These filters have become popular because their precise reproducibility allows design engineers to achieve performance levels that are difficult to obtain with analog filters. Digital Filters can be constructed from 3 fundamental mathematical operations which are Addition (or subtraction), Multiplication (normally of a signal by a constant), Time Delay i.e: delaying a digital signal by one or more sample periods.

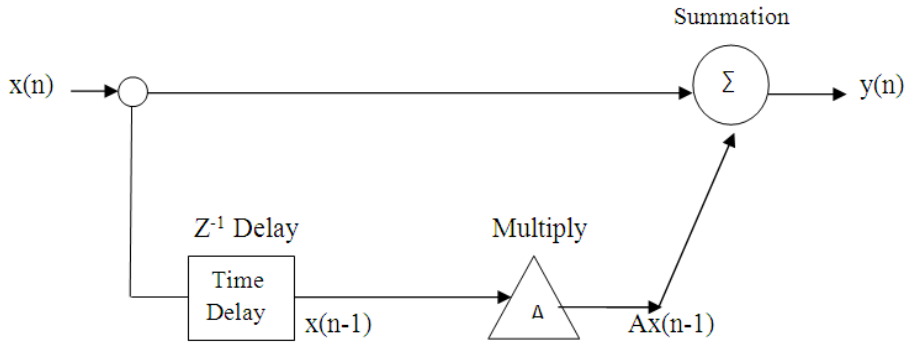

Figure 1.1 Block diagram of a Simple Digital Filter

The Impulse Response of a digital filter, *h(n)* is the response of the filter to an input consisting of the unit impulse function, $\delta(n)$. If the impulse response of a system is known, it is possible to calculate the system response for any input sequence $x(n)$. By definition, the unit impulse is applied to a system at sample index n=0. So, the impulse response is non-zero only for values of n greater than or equal to zero i.e *h(n)* is zero for n<0. This impulse response is said to be causal otherwise the system would be producing a response before an input has been applied. It is known from the time-invariance property of a Linear Time Invariant System that the response of a system to a delayed unit impulse $\delta(n-k)$ will be a delayed version of the unit impulse, i.e $h(n-k)$. It is also known from the linearity property that the response of a system to a weighted sum of inputs will be a weighted sum of responses of the system to each of the individual inputs. Therefore, the response of a system to an arbitrary input $x(n)$ can be written as follows:

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k)$$

Finite Impulse Response (FIR) filters are one of two primary types of filters used in DSP, the other type being Infinite Impulse Response Filters (IIR) filters. The impulse response of an FIR filter is "finite" because there is no feedback in the filter.

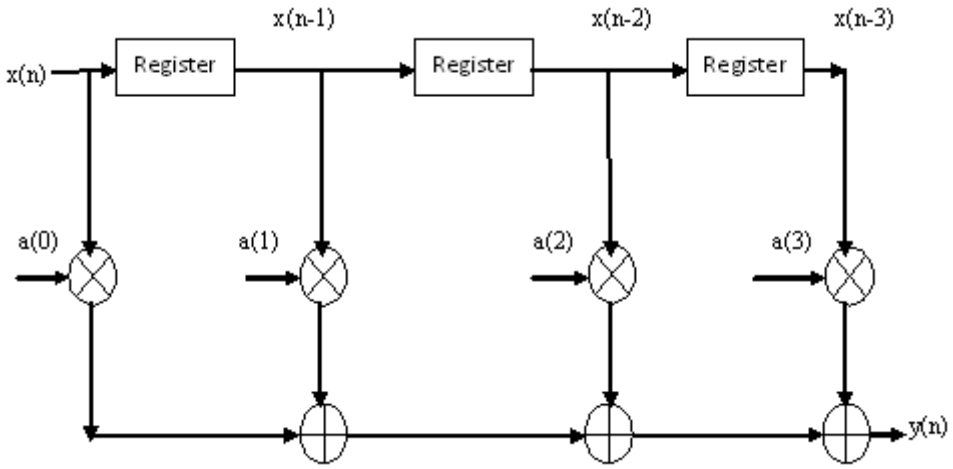

Figure 1.2 FIR Filter Architecture

II. BOOTH MULTIPLIER

Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation. The algorithm was invented by Andrew Donald Booth in 1950. Booth used desk calculators that were faster at shifting than adding and created the algorithm to increase their speed. Booth's algorithm is of interest in the study of computer architecture. Booth's algorithm examines adjacent pairs of bits of the N-bit multiplier Y in signed two's complement

representation, including an implicit bit below the least significant bit, $y_{-1} = 0$. For each bit y_i, for i running from 0 to N-1, the bits y_i and y_{i-1} are considered. Where these two bits are equal, the product accumulator P remains unchanged. Where $y_i = 0$ and $y_{i-1} = 1$, the multiplicand times 2^i is added to P; and where $y_i = 1$ and $y_{i-1} = 0$, the multiplicand times 2^i is subtracted from P. The final value of P is the signed product.

The representation of the multiplicand and product are not specified; typically, these are both also in two's complement representation, like the multiplier, but any number system that supports addition and subtraction will work as well. Here, the order of the steps is not determined. Typically, it proceeds from LSB to MSB, starting at $i = 0$; the multiplication by 2^i is then typically replaced by incremental shifting of the P accumulator to the right between steps; low bits can be shifted out, and subsequent additions and subtractions can then be done just on the highest N bits of P. There are many variations and optimizations on these details. The algorithm is often described as converting strings of 1's in the multiplier to a high-order +1 and a low-order –1 at the ends of the string. When a string runs through the MSB, there is no high-order +1, and the net effect is interpretation as a negative of the appropriate value.

Reason for Choosing Booth Multiplier

The general multiplication process is done by a dot method. The figure below shows a dot method representation i.e. how the basic dot multiplication process proceeds.

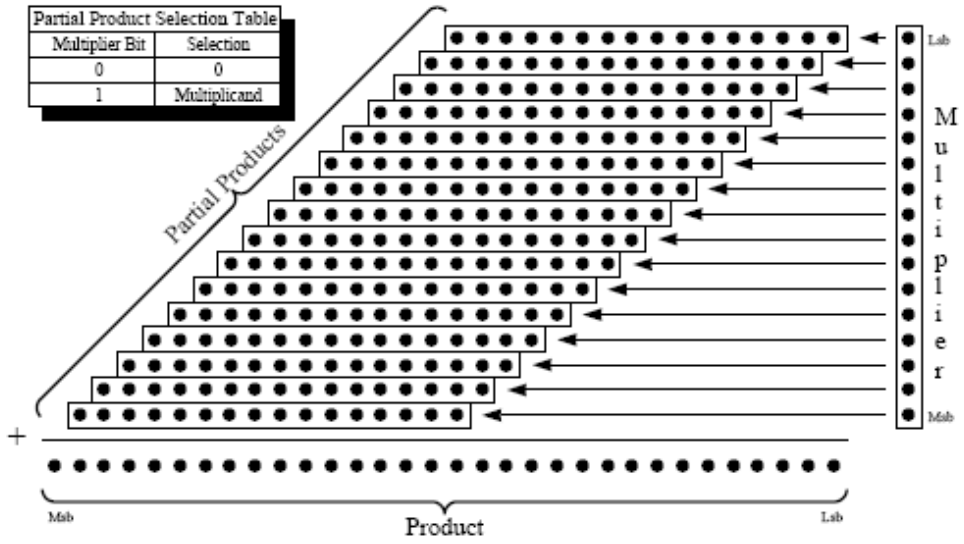

Figure 2.1 16 bit simple multiplication.

Each dot in the diagram is a place holder for a single bit which can be a zero or one. The partial products are represented by a horizontal row of dots, and the selection method used in producing each partial product is shown by the table in the upper left corner. The partial products are shifted to account for the differing

arithmetic weight of the bits in the multiplier, aligning dots of the same arithmetic weight vertically. The final product is represented by the double length row of dots at the bottom. This dot multiplication process has more complexity when compared with the Booth multiplier.

In Booth multiplier a generator that creates a smaller number of partial products will allow the partial product summation to be faster and use less hardware. The simple multiplication generator can be extended to reduce the number of partial products by grouping the bits of the multiplier into pairs, and selecting the partial products from the set {0, M, 2M, 3M}, where M is the multiplicand. This reduces the number of partial products by half, but requires a carry propagate add to produce the 3M multiple, before any partial products can be generated. The implementation of the Booth Multiplier has been discussed in the next section.

Carry skip Adder

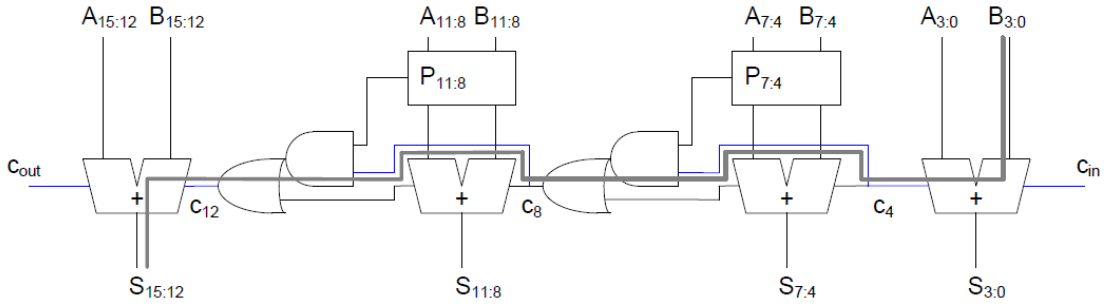

Figure 2.2 Block diagram of a carry skip adder

III. BOOTH MULTIPLICATION ALGORITHM

Booth's algorithm can be implemented by repeatedly adding (with ordinary unsigned binary addition) one of two predetermined values *A* and *S* to a product *P*, then performing a rightward arithmetic shift on *P*. Let m and r be the multiplicand and multiplier, respectively and let *x* and *y* represent the number of bits in m and r.

1. Determine the values of A and S and the initial value of *P*. All of these numbers should have a length equal to (*x* + *y* + 1).
1. A: Fill the most significant (leftmost) bits with the value of m. Fill the remaining (*y* + 1) bits with zeros.
2. S: Fill the most significant bits with the value of (−m) in two's complement notation. Fill the remaining (*y* + 1) bits with zeros.
3. P: Fill the most significant *x* bits with zeros. To the right of this, append the value of **r**. Fill the least significant (rightmost) bit with a zeros.
2. Determine the two least significant (rightmost) bits of *P*.
1. If they are 01, find the value of *P* + *A*. Ignore any overflow.
2. If they are 10, find the value of *P* + *S*. Ignore any overflow.
3. If they are 00, do nothing. Use *P* directly in the next step.
4. If they are 11, do nothing. Use *P* directly in the next step.

3. Arithmetically shift the value obtained in the 2nd step by a single place to the right. Let *P* now equal this new value.
4. Repeat steps 2 and 3 until they have been done *y* times.
5. Drop the least significant (rightmost) bit from *P*. This is the product of m and r.

Example
Find 3 × (−4), with m = 3 and r = −4, and *x* = 4 and *y* = 4
m = 0011, -m = 1101, r = 1100
A = 0011 00000
S = 1101 00000
P = 0000 11000

Perform the loop four times
1. P = 0000 1100 0. The last two bits are 00
   P = 0000 0110 0. Arithmetic right shift

2. P = 0000 0110 0. The last two bits are 00
   P = 0000 0011 0. Arithmetic right shift

3. P = 0000 0011 0. The last two bits are 10
   P = 1101 0011 0. P = P + S
   P = 1110 1001 1. Arithmetic right shift

4. P = 1110 1001 1. The last two bits are 11
   P = 1111 0100 1. Arithmetic right shift

   The product is 1111 0100, which is −12.

IV. SIMULATION RESULTS

The simulation is performed using XILINX ISE 13.1 version. The block schematic of FIR filter, RTL schematic and the waveforms of FIR filter are shown in the figure below. These results are obtained by considering the eight bit input data. The synthesis is also performed and the results obtained are compared with the previous methods used for the design of FIR filter. By observing the comparison results, the delay becomes reduced as well as the power also. Therefore the speed performance of the filter is increased.

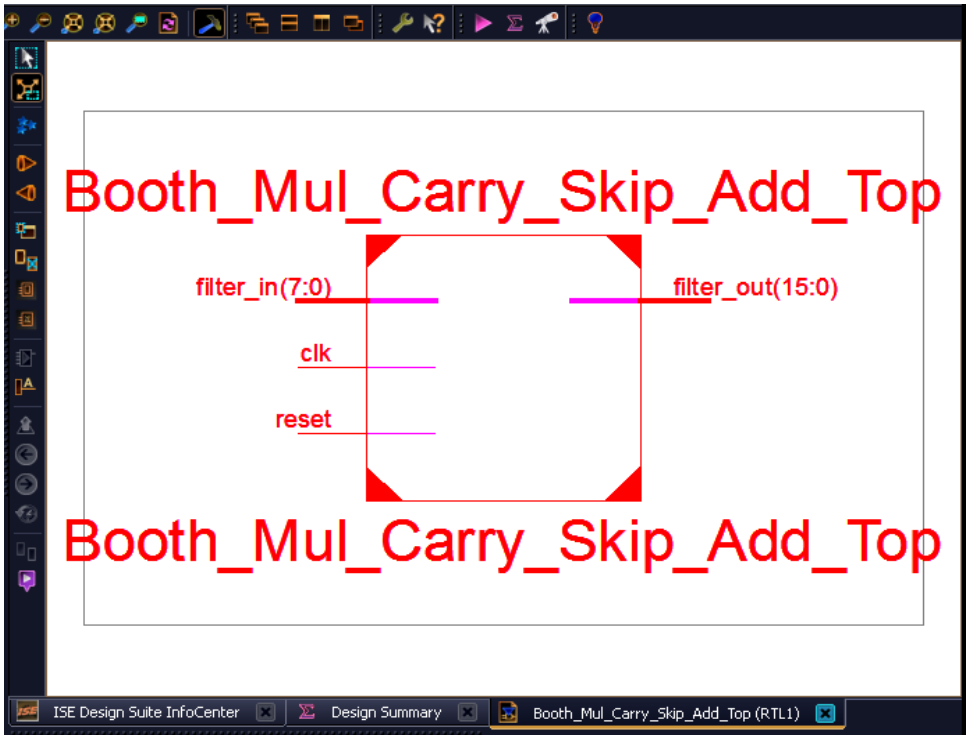

Figure 4.1 Schematic of FIR Filter

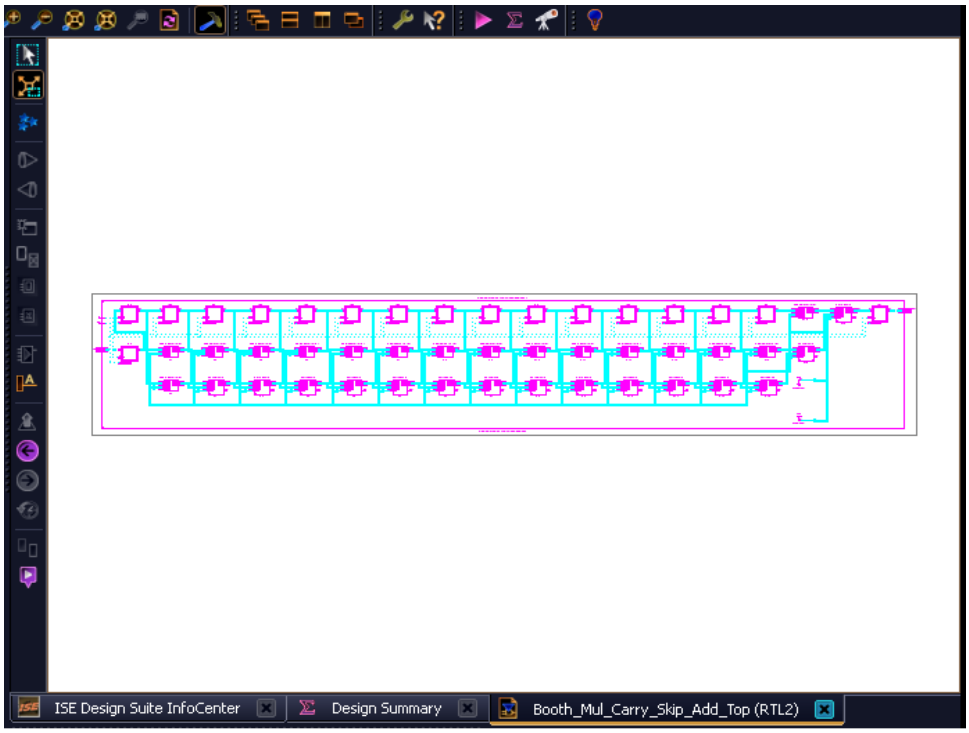

Figure 4.2 RTL Schematic of FIR filter

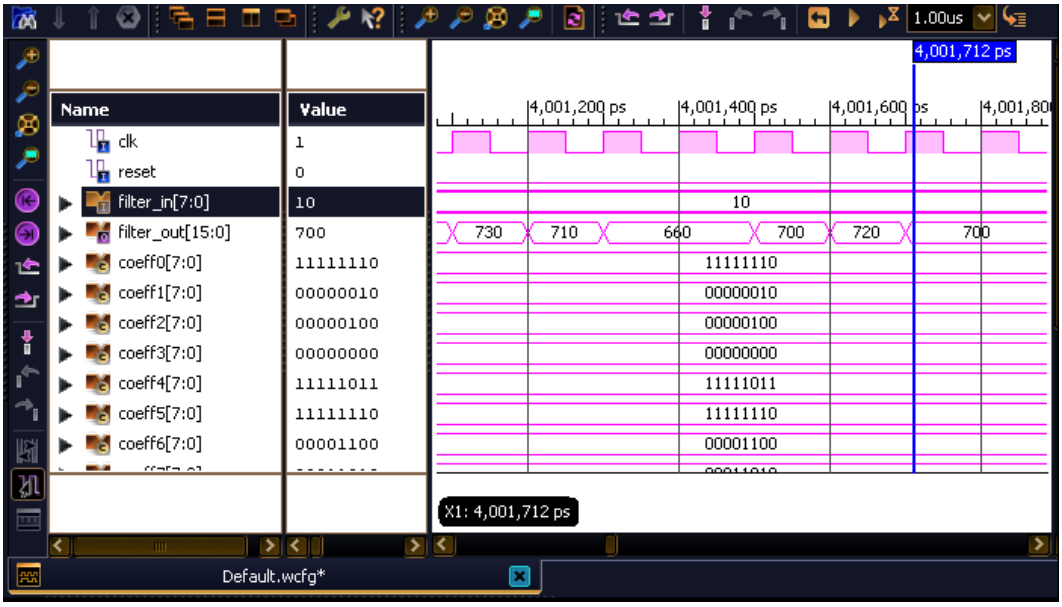

Figure 4.3 Waveform of the FIR filter

The design summary reports obtained by performing synthesis, includes area utilization, delay and power consumed by the device for the design of FIR filter is given in the figure below. And also the table shown in Table 4.2 presents the comparison of delay and power consumed values for the FIR filter design using different multipliers.

Logic Utilization	Used	Available	% Utilization
Number of Slice Flip Flops	151	4,896	3%
Number of 4 input LUTs	4,047	4,896	82%
Number of occupied Slices	2,274	2,448	92%
Number of Slices containing only related logic	2,274	2,274	100%
Number of Slices containing unrelated logic	0	2,274	0%
Total Number of 4 input LUTs	4,197	4,896	85%
Number of bonded IOBs	26	172	15%
Number of BUFGMUXs	1	24	4%

Table 4.1 Area Utilization of FIR filter

Method	Delay(ns)	Power(mW)
Shift Add multiplier	7.7	923
MCM Multiplier	12	1023
Booth Multiplier	3.42	52.27

Table 4.2 Comparison

V. CONCLUSION

In this project the Booth Multiplier and carry skip adder have been considered and are combined in the design of FIR filter. They occupy less amount of space when compared with the normal multiplier. This criterion is very important in the fabrication of the chips and the high performance system requires components which are as small as possible. Once all the combination of multiplier and adders has been done in the design of the FIR filter then a comparison is to be made to say which combination of multiplier and adders will be the best in terms of delay, power, area and memory. The synthesis and simulation is carried out using XILINX ISE 13.1 version software. The Total memory usage is 168036 kilobytes. The functionality is verified successfully.

VI. REFERENCES

[1]. L.Wanhammar, *DSP Integrated Circuits*. New York: Academic, 1999.

[2]. C.Wallace, "A suggestion for a fast multiplier," *IEEE Trans. Electron Comput.*, vol. 13, no. 1, pp. 14–17, Feb. 1964.

[3]. W.Gallagher and E.Swartzlander, "High radix booth multipliers using reduced area adder trees," in *Proc. Asilomar Conf. Signals, Syst. Comput.*, vol. 1. Pacific Grove, CA, Oct.–Nov. 1994, pp. 545–549.

[4]. J.McClellan, T.Parks, and L. Rabiner, "A computer program for designing optimum FIR linear phase digital filters," *IEEE Trans. Audio Electroacoust.*, vol. 21, no.6, pp. 506–526, Dec. 1973.

[5]. H. Nguyen and A. Chatterjee, "Number-splitting with shift-and-add decomposition for power and hardware optimization in linear DSP synthesis," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 8, no. 4, pp. 419–424, Aug. 2000.

[6]. M. Ercegovac and T. Lang, *Digital Arithmetic*. San Mateo, CA: Morgan Kaufmann, 2003.

[7]. R. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 43, no. 10, pp. 677–688, Oct. 1996.

[8]. I.-C.Park and H.-J. Kang, "Digital filter synthesis based on minimal signed digit representation," in *Proc. DAC*, 2001, pp. 468–473.

[9]. L. Aksoy, E. Costa, P. Flores, and J. Monteiro, "Exact and approximate algorithms for the optimization of area and delay in multiple constant multiplications," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 6, pp. 1013–1026, Jun. 2008.

[10]. A. Dempster and M. Macleod, "Use of minimum-adder multiplier blocks in FIR digital filters," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 42, no. 9, pp. 569–577, Sep. 1995.

[11]. Y. Voronenko and M. Püschel, "Multiplierless multiple constant multiplication," *ACM Trans. Algor.*, vol. 3, no. 2, pp. 1–39, May 2007.

[12]. L. Aksoy, E. Gunes, and P. Flores, "Search algorithms for the multiple constant multiplications problem: Exact and approximate," *J. Microprocess. Microsyst.*, vol. 34, no. 5, pp. 151–162, Aug. 2010.