

AREA EFFICIENT AND LOW LATENCY FINITE IMPULSE RESPONSE FILTER DESIGN

1. GALAM KRISHNAIAH, 2.ALURI BULLI BABU

1. M.Tech, Dept. Of ECE, Nova College of Engineering and Technology, Ibrahimpatnam, A.P

2. Guide, Dept. Of ECE, Nova College of Engineering and Technology, Ibrahimpatnam, A.P

ABSTRACT:

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. Multipliers are key components of many high performance systems such as FIR filters, microprocessors, digital signal processors, etc. A system's performance is generally determined by the performance of the multiplier because the multiplier is generally the slowest element in the system. Furthermore, it is generally the most area consuming. Hence, optimizing the speed and area of the multiplier is a major design issue. FIR filter multipliers are extensively characterized with power simulations, providing a methodology for the perturbation of the coefficients of baseline filters at the algorithm level to trade-off reduced power consumption for filter quality. The proposed optimization technique does not require any hardware overhead and it enables the possibility of scaling the power consumption of the filter at runtime, while ensuring the full baseline performance of any programmed filter whenever it is required.

INTRODUCTION:

FIR DIGITAL filters find extensive applications in mobile communication systems for applications such as channelization, channel equalization, matched filtering, and pulse shaping, due to their absolute stability and linear phase properties. The filters employed in mobile systems must be realized to consume less power and operate at high speed. Recently, with the advent of software defined radio (SDR) technology, finite impulse response (FIR) filter research has been focused on reconfigurable realizations. The fundamental idea of an SDR is to replace most of the analog signal processing in the transceivers with digital signal processing in order to provide the advantage of flexibility through reconfiguration. This will enable different air-interfaces to be implemented on a single generic hardware platform to support multistandard wireless communications [1]. Wideband receivers in SDR must be realized to meet the stringent specifications of low power consumption and high speed. Reconfigurability of the receiver to work with different wireless communication standards is another key requirement in an SDR. The most computationally intensive part of an SDR receiver is the channelizer since it operates at the highest sampling rate [2]. It

extracts multiple narrowband channels from a wideband signal using a bank of FIR filters, called channel filters. Using polyphase filter structure, decimation can be done prior to channel filtering so that the channel filters need to operate only at relatively low sampling rates. This can relax the speed of operation of the filters to a good extent [2]. However due to the stringent adjacent channel attenuation specifications of wireless communication standards, higher order filters are required for channelization and consequently the complexity and power consumption of the receiver will be high. As the ultimate aim of the future multi-standard wireless communication receiver is to realize its functionalities in mobile handsets, where its full utilization is possible, low power and low area implementation of FIR channel filters is inevitable.

In [3], the filter multiplications are done via state machines in an iterative shift and add component and as a result of this there is huge savings in area. For lower order filters, the approach in [3] offers good trade-off between speed and area. But in general, the channel filters in wireless communication receivers need to be of high order to achieve sharp transition band and low adjacent channel attenuation

requirements. For such applications, the approach in [3]

FILTERS: 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. This chapter gives a brief about the types of filters.

ANALOG FILTERS: 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. In the low pass case, a potential divider is formed from a series resistor followed by a shunt capacitor, as illustrated in Figure. The filter input is at one end of the resistor and the output is at the point where the resistor and capacitor join. The RC filter works because the capacitor reactance reduces as the frequency increases. It should be remembered that the reactance is 90° out of phase with resistance.

DIGITAL FILTERS: 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

FINITE IMPULSE RESPONSE: In signal processing, a finite impulse response (FIR) filter is a filter whose impulse response (or response to any

finite length input) is of finite duration, because it settles to zero in finite time. This is in contrast to infinite impulse response (IIR) filters, which may have internal feedback and may continue to respond indefinitely (usually decaying).

The impulse response (that is, the output in response to a Kronecker delta input) of an Nth-order discrete-time FIR filter lasts exactly $N + 1$ samples (from first nonzero element through last nonzero element) before it then settles to zero.

FIR filters can be discrete-time or continuous-time, and digital or analog.

For a causal discrete-time FIR filter of order N, each value of the output sequence is a weighted sum of the most recent input values:

$$y[n] = b_0x[n] + b_1x[n-1] + \dots + b_Nx[n-N]$$

$$= \sum_{i=0}^N b_i \cdot x[n-i],$$

where:

- $x[n]$ is the input signal,
- $y[n]$ is the output signal,
- N is the filter order; an N th-order filter has $(N + 1)$ terms on the right-hand side
- b_i is the value of the impulse response at the i 'th instant for $0 \leq i \leq N$ of an N th-order FIR filter. If the filter is a direct form FIR filter then b_i is also a coefficient of the filter.

This computation is also known as discrete convolution.

The $x[n-i]$ in these terms are commonly referred to as taps, based on the structure of a tapped delay line that in many implementations or block diagrams provides the delayed inputs to the multiplication operations.

FIR FILTER USING BAUGHWOOLEY AND BOOTH ENCODING:

Digital multipliers can be implemented choosing from a wide range of topologies based on the desired number representation, as well as on design requirements, such as area and speed [21]. Two of the most common topologies have been considered as an example for the analysis of the internal switching activity, which is strongly related to the dynamic power consumption: the radix-2 Baugh-Wooley (BW2) multiplier [22] that presents a simple and straightforward implementation, and the radix-4 Booth-recoded (BR4) multiplier, known for its more

complex structure and high-speed performance. Both topologies have been implemented to perform signed multiplication, using fixed-point two's complement as number representation. While these multipliers differ in the partial-products generator (PPG), they implement the same partial-products reducer (PPR) and the same vector-merging adder (VMA), here implemented as a carry-save adder with $(m,2)$ compressors [22] and a carry-propagate adder, respectively. The RTL representations of both the BW2 multiplier and of the carry-save adder with $(m,2)$ compressors have been taken from the VHDL Library of Arithmetic Units proposed in [22]. The considered topologies are reviewed with more details in the following subsections.

The BW2 multiplier is a simple structure which can achieve medium operating speed with moderate silicon area. Fig. illustrates the operating concept of this topology.

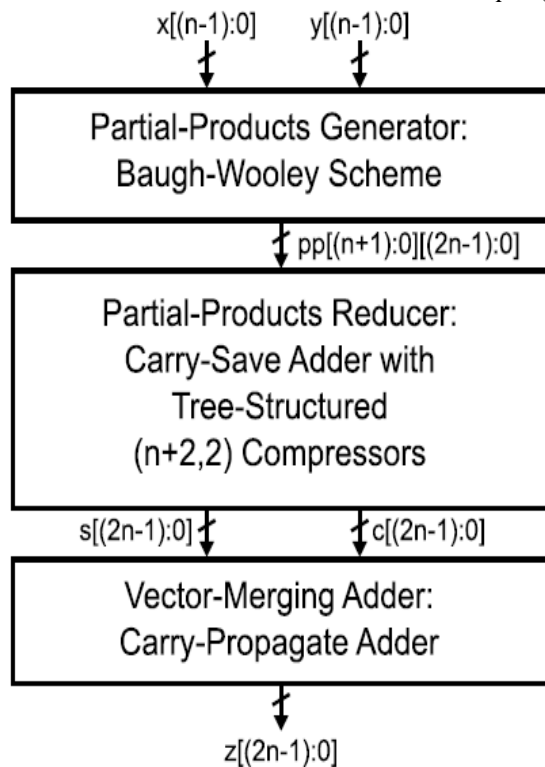


Fig. Structure of a signed $n \times n$ -bit radix-2 Baugh-Wooley multiplier

input operands are passed to the PPG that implements the Baugh-Wooley scheme [21], [22], which feeds the PPR implemented as a carry-save adder with $(m,2)$ compressors [22] as shown in Fig. For the considered implementation, the half adders (HAs) and full adders (FAs) instantiated inside each of the compressors are connected in a tree structure to reduce the critical path of the PPR. The sums and carries of the PPR are then passed to the VMA

implemented as a carry-propagate adder that provides the final result of the multiplication. The PPR is usually the most complex structure of the multiplier and therefore consumes the largest amount of power. For this reason, in order to reduce the power consumption of the BW2 multiplier, we first analyze the switching activity of the implemented PPR. The primary operation of the PPR is to shift and add the partial products, and therefore it is generally implemented with HAs and FAs as the main building blocks. The switching activity of these gates can be reduced by increasing the probability of having stable logic-zeros at their inputs, which corresponds to forcing more partial products to be equal to zero.

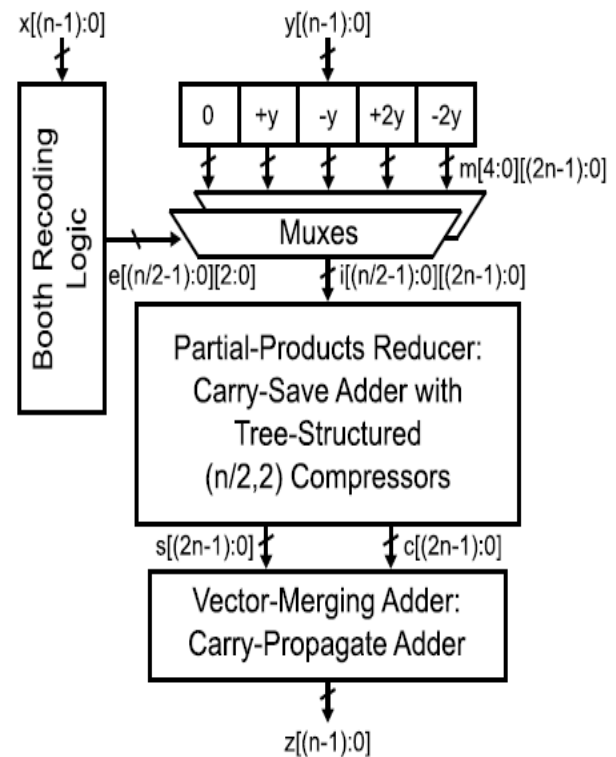


Fig. Structure of a signed $n \times n$ -bit radix-4 Booth-recoded multiplier.

Since radix-2 Baugh-Wooley multipliers are rather slow, we also study a fast multiplier that uses Booth recoding. A BR4 multiplier, shown in Fig., has been considered, where the PPR is fed with less than half of the partial products of those in the BW2 multiplier, thereby providing a much shorter critical path. As opposed to the symmetric BW2 multiplier, in the BR4 topology, the two input operands are processed differently, since x is passed to the recoding logic that decides which multiples of y should be fed to the PPR. For the considered implementation, a carry-save adder with $(m,2)$ compressors [22] is used for the PPR and a carry-propagate adder is used for the VMA, as in the BW2 multiplier. The non-zero partial product per

input operand analysis on the PPG, previously shown for the BW2 multiplier, was also applied to the 8×8 -bit BR4 multiplier.

In order to characterize the power consumption of a multiplier implementation, a constant value was applied to the coefficient operand input, while a sequence of 1000 independent uniformly distributed random values were assigned to the data operand. All $2n$ coefficient values were considered for each $n \times n$ -bit multiplier, and the dynamic power consumption was extracted for each coefficient operand value. The presented power analyses consider 8×8 -bit multipliers; however, larger bit-widths were also tested and provided similar results. Due to the symmetric structure of the BW2 multiplier, the power characterization is almost identical for both ports used for the coefficient operand. Hence, only results for x as the coefficient operand are reported for this topology. In particular, some recoded values of x can cause a large number of the operands summed in the PPR to be equal to zero, which significantly reduces the power consumption since the PPR is one of the main contributors to the power consumed by the entire multiplier. Furthermore, if x is constant, the Booth recoding logic does not present any switching activity. On the other hand, when y is kept constant, little power variation is observed.

PROPOSED ARCHITECTURE:

Modified Booth Algorithm Encoder

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 while doing research on crystallography at Birkbeck college in Bloomsbury, London. 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.

This modified booth multiplier is used to perform high-speed multiplications using modified booth algorithm. This modified booth multiplier's computation time and the logarithm of the word length of operands are proportional to each other. We can reduce half the number of partial product. Radix-4 booth algorithm used here increases the speed of multiplier and reduces the area of multiplier circuit. In this algorithm, every second column is taken and multiplied by 0 or +1 or +2 or -1 or -2 instead of multiplying with 0 or 1 after shifting and adding of every column of the booth multiplier.

The modified stand-alone multiplier consists of a modified recorder (MBR). MBR has two parts, i.e., Booth Encoder (BE) and Booth Selector (BS). The

operation of BE is to decode the multiplier signal, and the output is used by BS to produce the partial product. Then, the partial products are added to the Wallace tree adders, similar to the carry-save-adder approach. The last transfer and sum output line are added by a carry look-ahead adder, the carry being stretched to the left by positioning.

Table Quartet coded signed-digit table

Quartet value	Signed-digit value
0000	0
0001	+1
0010	+1
0011	+2
0100	+2
0101	+3
0110	+3
0111	+4
1000	-4
1001	-3
1010	-3
1011	-2
1100	-2
1101	-1
1110	-1
1111	0

Here we have a multiplication multiplier, $3Y$, which is not immediately available. To Generate it, we must run the previous addition operation: $2Y + Y = 3Y$. But we are designing a multiplier for specific purposes and then the multiplier belongs to a set of previously known numbers stored in a memory chip. We have tried to take advantage of this fact, to relieve the radix-8 bottleneck, that is, $3Y$ generation. In this way, we try to obtain a better overall multiplication time or at least comparable to the time, we can obtain using a radix-4 architecture (with the added benefit of using fewer transistors). To generate $3Y$ with 21-bit words you just have to add $2Y + Y$, ie add the number with the same number moved to a left position.

A product formed by multiplying it with a multiplier digit when the multiplier has many digits. Partial products are calculated as intermediate steps in the calculation of larger products.

The partial product generator is designed to produce the product multiplying by multiplying A by 0, 1, -1, 2, -2, -3, -4, 3, 4. Multiply by zero implies that the product is "0". Multiply by "1" means that the product remains the same as the multiplier. Multiply by "-1" means that the product is the complementary form of the number of two. Multiplying with "-2" is to move left one as this rest as per table.

SIGN EXTENSION CORRECTOR:

The Sign Extension Corrector is designed to increase the Booth multiplier capacity by multiplying not only the unsigned number but also the signed number.

The principle of the sign extension that converts the signed multiplier not signed as follows. When unsigned is signalled $s_u = 0$, it indicates the multiplication of the unsigned number and when $s_u = 1$, it shows the multiplication of the signed number. When a bit signal is called unsigned bit (s_u), it is indicated whether the multiplication operation is an unsigned number or number.

Table.Sign extension corrector

Sign-unsigned	Type of operation
0	Unsigned multiplication
1	Signed multiplication

Example:

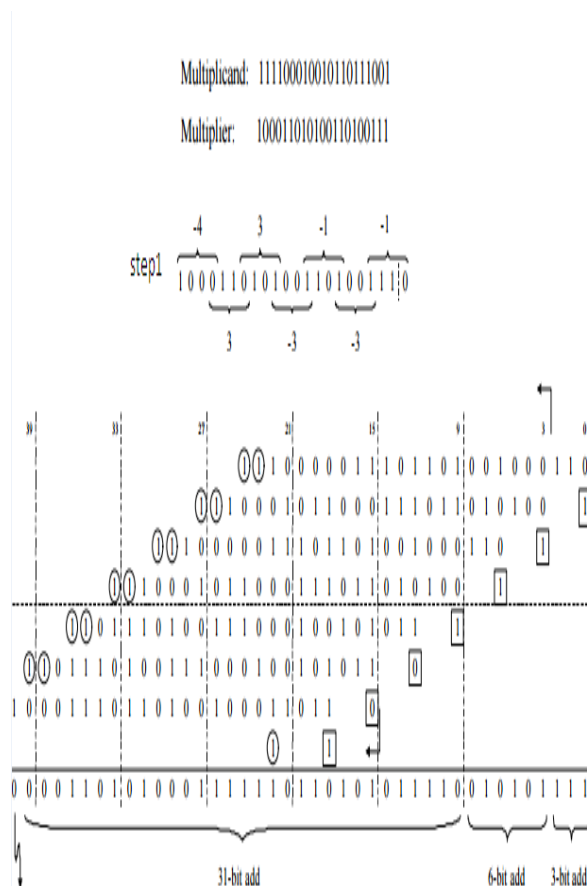
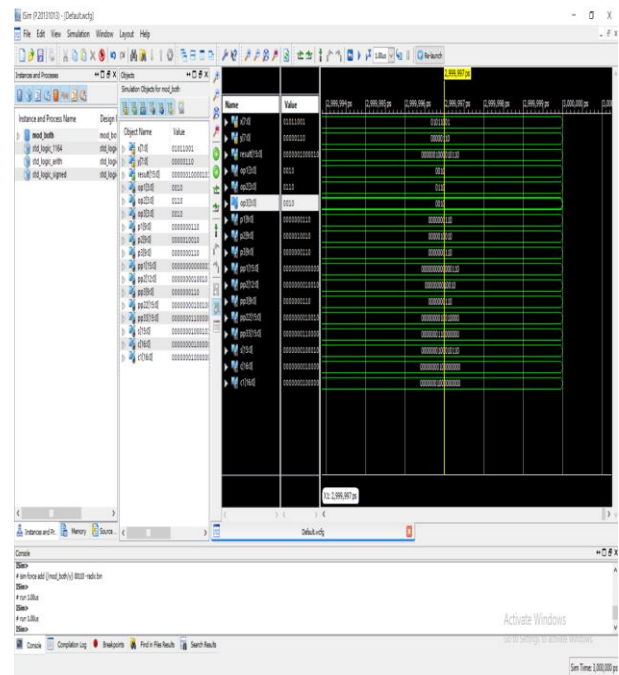


Fig. Example of modified booth algorithm

RESULT:

SIMULATION:



SYNTHESIS:

PARAMETER	EXISTING	PROPOSED
POWER (mW)	393	80
TIME (ns)	23.2	15.74
Density	573	80

CONCLUSION:

The implemented techniques can effectively be applied, as an example, to reconfigurable FIR accelerators. A simple greedy algorithm is used to modify the coefficients of a baseline filter to derive a new set of coefficients that are optimized for low power consumption while allowing for some degradation of the filtering quality. By exploiting the flexibility on the algorithm level, the proposed approximate computing technique does not require any design overhead for a programmable accelerator. At the same time, it ensures the quality of the baseline filter whenever it is required, while it offers also the possibility of scaling the power consumption at runtime when energy is short and reduced accuracy is tolerated.

REFERENCES

- [1] Yu Pan and Pramod Kumar Meher, Senior Member, IEEE, "Bit-Level Optimization of Adder-Trees for Multiple Constant Multiplications for Efficient FIR Filter Implementation," Transactions On Circuits And Systems—I: Regular Papers, Vol. 61, No. 2, February 2014.
- [2] D. R. Bull and D. H. Horrocks, "Primitive operator digital filter," IEEE Proceedings-G, vol. 138, no. 3, pp. 401–412, Jun. 1991.
- [3] A. G. Dempster and M. D. Macleod, "Use of minimum-adder multiplier blocks in FIR digital filters," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 42, no. 9, pp. 569–577, 1995.
- [4] S. D. S. M. Mehendale and G. Venkatesh, "Synthesis of multiplier-less FIR filters with minimum number of additions," in Proc. IEEE ICCAD, 1995.
- [5] I. C. Park and H. J. Kang, "Digital filter synthesis based on minimal signed digit representation," in Proc. Design Autom. Conf. (DAC), 2001.
- [6] Y. Voronenko and M. Püschel, "Multiplierless multiple constant multiplication," ACM Trans. Algorithms, vol. 3, no. 2, 2007.
- [7] P. K. Meher and Y. Pan, "Mcm-based implementation of block fir filters for high-speed and low-power applications," in Proc. VLSI and System-on-Chip (VLSI-SoC), 2011 IEEE/IFIP 19th Int. Conf., Oct. 2011, pp. 118–121.
- [8] L. Aksoy, C. Lazzari, E. Costa, P. Flores, and J. Monteiro, "Design of digit-serial FIR filters: Algorithms, architectures, and a CAD tool," IEEE Trans. Very Large Scale Integration (VLSI) Syst., vol. 21, no. 3, pp. 498–511, Mar. 2013.
- [9] M. B. Gately, M. B. Yeary, and C. Y. Tang, "Multiple real-constant multiplication with improved cost model and greedy and optimal searches," in Proc. IEEE ISCAS, May 2012, pp. 588–591.
- [10] M. Kumm, P. Zipf, M. Faust, and C.-H. Chang, "Pipelined adder graph optimization for high speed multiple constant multiplication," in Proc. IEEE ISCAS, May 2012, pp. 49–52.
- [11] R. Hartley and A. Casavant, "Tree-height minimization in pipelined architectures," in Proc. IEEE ICCAD, Nov. 1989.
- [12] J. Chen, C. H. Chang, F. Feng, W. Ding, and J. Ding, "Novel design algorithm for low complexity programmable FIR filters based on extended double base number system," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 62, no. 1, pp. 224–233, Jan. 2015.
- [13] S. Y. Park and P. K. Meher, "Efficient FPGA and ASIC realizations of a DA-based reconfigurable FIR digital filter," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 61, no. 7, pp. 511–515, Jul. 2014.
- [14] P. Kulkarni, P. Gupta, and M. Ercegovic, "Trading accuracy for power with an underdesigned multiplier architecture," in Proc. 24th Int. Conf. VLSI Design, Jan. 2011, pp. 346–351.
- [15] K. Bhardwaj, P. S. Mane, and J. Henkel, "Power- and area-efficient approximate wallace tree multiplier for error-resilient systems," in Proc. 15th Int. Symp. Quality Electron. Design (ISQED), Mar. 2014, pp. 263–269.
- [16] C.-H. Lin and I.-C. Lin, "High accuracy approximate multiplier with error correction," in Proc. IEEE 31st Int. Conf. Comput. Design (ICCD), Oct. 2013, pp. 33–38.
- [17] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE), Mar. 2014, pp. 1–4.
- [18] C. Neau, K. Muhammad, and K. Roy, "Low complexity FIR filters using factorization of perturbed coefficients," in Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE), 2001, pp. 268–272.
- [19] S. Hong, S. Kim, M. C. Papaefthymiou, and W. E. Stark, "Low power parallel multiplier design for DSP applications through coefficient optimization," in Proc. 12th Annu. IEEE Int. ASIC/SoC Conf., Sep. 1999, pp. 286–290.
- [20] M. Mehendale, S. D. Sherlekar, and G. Venkatesh, "Low-power realization of FIR filters on programmable DSPs," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 6, no. 4, pp. 546–553, Dec. 1998.
- [21] B. Parhami, Computer Arithmetic, vol. 20. Oxford, U.K.: Oxford Univ. Press, 1999.
- [22] R. Zimmermann, "VHDL library of arithmetic units," in Proc. 1st Int. Forum Design Lang. (FDL), Lausanne, Switzerland, 1998, pp. 267–272.
- [23] Y. Huang, A. Kapoor, R. Ruten, and J. P. de Gyvez, "A 13 bits 4.096GHz 45Nm CMOS digital decimation filter chain with carry-save format numbers," Microprocess. Microsyst., vol. 39, no. 8, pp. 869–878, Nov. 2015. [Online]. Available: <http://dx.doi.org/10.1016/j.micpro.2014.11.003>