FIR Filter Design Using The Signed-Digit Number System and Carry Save Adders – A Comparison

Hesham Altwaijry
Computer Engineering Department,
King Saud University
PO Box 51178, Riyadh 11543, Saudi Arabia

Yasser Mohammad Seddiq
Computer Engineering Department,
King Saud University
PO Box 51178, Riyadh 11543, Saudi Arabia

Abstract— **This work looks at optimizing finite impulse response (FIR) filters from an arithmetic perspective. Since the main two arithmetic operations in the convolution equations are addition and multiplication, they are the targets of the optimization. Therefore, considering carry-propagate-free addition techniques should enhance the addition operation of the filter. The signed-digit number system is utilized to speedup addition in the filter. An alternative carry propagate free fast adder, carry-save adder, is also used here to compare its performance to the signed-digit adder. For multiplication, Booth encoding is used to reduce the number of partial products. The two filters are modeled in VHDL, synthesized and place-and-routed. The filters are deployed on a development board to filter digital images. The resultant hardware is analyzed for speed and logic utilization**

Keywords— FIR Filters – Signed Digit – Carry-Save – FPGA

I. Introduction

Digital signal processing (DSP) systems employ computer systems to digitally process input signals. An example where computer arithmetic is a key factor in optimizing the design is digital filters especially convolution-based ones. The hardware complexity and processing delay of these digital filters are proportional to a parameter called the filter order, which is highly desired to be large [1]. These filters have been built using FPGA's [2] [3] [4]

A fundamental principle in computer arithmetic upon which all the succeeding aspects are based is how values are to be represented. As all the computing platforms that are used today for digital signal processing are based on digital electronics, the arithmetic operations they perform should be handled in a way that is suitable to the nature of the electronics that build these platforms. The way a value is represented is called a number system. Computers were initially developed to use the binary number system (radix-2). Although, computers use radix-2, there have been few number systems discussed in the computer arithmetic literature that are unconventional in terms of representation and operations. Such number systems are used in computers for some special applications.

A. The Binary Number System

Binary number systems are called positional number systems [5]. A general expression for the value of an *n*-digit number *A* consisting of digits a_{n-1}, a_{n-2}, … , a_0, in radix-*r* number system is as follows:

$$A = \sum_{i=0}^{n-1} a_i \times r^i \tag{1}$$

In computers, the choice of *r* is 2 due to electronic circuit limitations. When *r* equals a constant value as in the decimal and the binary systems, this is called a fixed-radix number system. An observation on the conventional fixed-radix positional representation is that special representations are required for signed number and that carry propagation in addition, which increases the delay of operations, limits system scalability and adds more complexity to algorithm implementation.

B. Unconventional Number Systems

A common feature of the unconventional number systems is redundancy; a positional number system is redundant when the number of elements in its digit set is greater than *r*, where *r* is the radix. In a redundant number system, an algebraic value can have more than one representation. Redundant number systems can improve system reliability, increase speed of operations, and provide structural flexibility**.** [6]

Signed digit number systems are a positional number representation with a constant radix $r \geq 3$. Each digit of a signed-digit number can have one value of the set $\{-a, -a+1, \ldots, 0, \ldots, a-1, a\}$ [7] [8]. The maximum possible magnitude, *a*, is set as follows

$$\frac{r_o + 1}{2} \leq a \leq r_o - 1 \quad \text{for odd radices } r_o \geq 3 \tag{2}$$

$$\frac{r_e}{2} + 1 \leq a \leq r_e - 1 \quad \text{for odd radices } r_e \geq 4 \tag{3}$$

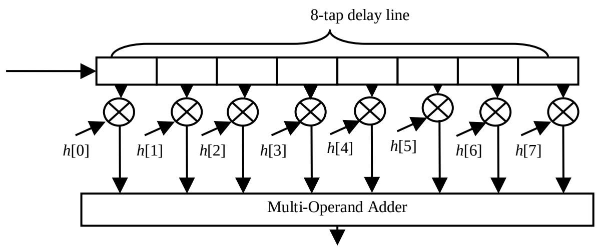


Fig. 1. Basic FIR Filter

When a value is represented with n binary bits, then it will be represented with $k = \lceil n / \lceil \log_2 r \rceil \rceil$ signed digits [9]

Signed-digit systems have the advantage that the addition time of a multi-operand adder that is built by cascading identical digit adders is constant.

A special case in signed-digit representation is for the radix $r = 2$ and the digit set is $\{\bar{1}, 0, 1\}$ where $\bar{1}$ represents –1. In this case, the representation is called canonic signed-digit (CSD). Three main properties of the CSD are that it is irredundant, the number of nonzero digits is minimal and multiplying any two adjacent digits will produce zero. In the applications that involve multiple constant multiplications (MCM) as in the FIR filters, using CSD guarantees the minimal number of adders.

II. Digital Filters

Filters are signal processing components that are used to process interfered and corrupted signals. They can be classified to two main categories: analog and digital filters. Filters in these two categories are different in terms of cost, speed, accuracy, power consumption and implementation, but they are similar in the sense that they are both used to filter signals.

A commonly used method of implementing digital filters is by considering a subset of the filter’s impulse response. Filter designed this way are called finite impulse response (FIR) filters. The mathematical process used to get the output of a linear system according to its impulse response is the convolution. When a digital signal $x[n]$ is to be processed by a system of impulse response $h[n]$, the output is the result of the following equation [1]:

$$y[n] = \sum_{k=0}^{N-1} h[k]x[n-k] \tag{4}$$

The above equation describes how each sample of the output signal is calculated. This is an application of the widely used mathematical operation of the dot product, which consists purely of multiplication and addition. Optimizing the dot product does not only serve the FIR filter application, but also some other applications that are similarly described such as radar processing, signal correlation and matrix multiplication.

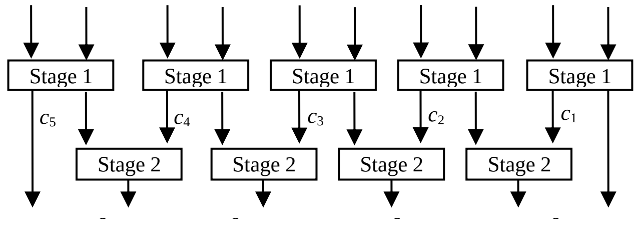


Fig. 2. Signed Digit Addition

A general block diagram of the convolution process as it is implemented in hardware is shown in Fig. 1 The delay line represents the inversion and shift in the input $x[n]$. The taps of the delay line are multiplied by the constant values of $h[n]$. The wider the delay line, the more accurate the results of the FIR filter are. Of course, this is on expense of more hardware resources, higher power consumption and higher cost.

In order to make the FIR filter performs addition faster, breaking the carry propagation chain in its adders is essential. The two most common techniques to achieve this are signed-digit addition and carry-save addition.

A. *Signed-Digit Addition*

Signed-digit number system can perform addition and subtraction with a limited propagation of carry. This feature makes the adder's delay independent of the operand length which implies less delay. The carry propagation can go as far as one position to the left. The sign of the number is implicitly expressed in the digits and no special representation is needed for this purpose. A block diagram of a signed digit adder is depicted in Fig. 2. The values shown in the figure are determined in [10]

B. *Carry-Save Addition*

Carry-save addition is one of the carry-propagate free methods of addition [11]. Carry-save adders (CSA) are mainly used when adding three operands or more. CSAs are built using (3, 2) counters in a manner that prevents carry propagation. The term (3, 2) counter is an alternative name for a full adder because it receives three bits of the same weight and outputs two bits representing the number of ones in the three-bit input. In the ordinary carry-propagate adders, the least significant bit of the output of the (3, 2) counter is the sum while the most significant bit is the carry that propagates to the left. Therefore, adding multi-operands using a CSA will result in a vector of the sum bits and another vector of carry bits. This two-vector result invites the need for a carry-propagate adder to add these two vectors to get a single result in the normal binary representation.

III. Implementing Signed Digit Filter

For the purpose of implementing a high-speed FIR filter, the arithmetic advantages of the signed-digit number system have been exploited to enhance the filter performance. This section is an elaboration to our work in [12]. It discusses the adder and the multiplier design and implementation.

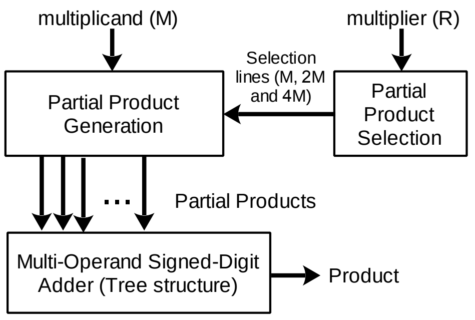

multiplicand (M)
multiplier (R)
Selection lines (M, 2M and 4M)
Partial Product Generation
Partial Product Selection
Partial Products
Multi-Operand Signed-Digit Adder (Tree structure)
Product

Fig. 3. Signed Digit Booth-3 Multiplier

A. Digit Set and Encoding

The signed-digit number system used in this work is in radix-2. Therefore, the allowed set of digits is { $\bar{1}$, 0, 1} where $\bar{1}$ represents –1. As there are three possible digits in this set, two bits are needed to encode each digit.

There are two commonly used encoding methods used to encode the digits of a number in the signed-digit representation [10] [13]. In the first method, each digit is assigned a 2's complement number representing the algebraic value of that digit. In the second encoding method, each digit of the number in signed-digit representation is assigned a 2-bit code x such that the sum of the two bits is equal to the value of that digit. If the high bit holds a negative sign, the low bit holds a positive sign and vice versa. That is, the value can be determined either as $x^+ + x^-$ or as $x^- + x^+$.

The second encoding method makes converting signed-digit numbers to 2's complement number easier [10]. Additionally, sign inversion of a digit is simply swapping the high and the low bits. Therefore, in this work the second encoding method with the value determined as $x_h - x_l$ is used.

A Signed digit adder can be implemented as a straight forward implementation into an FPGA by means of using lookup tables (LUT). However, the synthesis of this implementation results in a very poor utilization for the FPGA logic elements. Alternatively, the adder can be implemented using logic gates based upon equations presented by [14].

B. Signed Digit Partial Product Generation

FIR filters involve multiplying the input samples with the filter kernel coefficients. Thus, improving the filter multipliers will significantly improve the filter performance. Multiplication is done two steps:

1. Generating the partial products.
2. Accumulating the partial products.

The number of partial products needed for the multiplication can be reduced by using Booth's algorithm [15], however this encoding is performed serially, it can be done in parallel using the modified Booth encoding [16]. Booth-2 encoding is the most commonly used method. However, Booth-3 provides more reduction for the non-zero digits but the existence of the hard multiple 3M forms an obstacle when applying Booth-3 encoding. An efficient solution for the hard multiple 3M was proposed in [13] by exploiting the advantages of the signed-digit number system. The multiplier proposed accepts two operands in 2's complement representation and gives their product in signed-digit representation. The digit coding method used in that work is the sum-of-bits in the form $x^+ + x^-$. The signed-digit encoding method is utilized to determine the hard multiple 3M as 4M – M.

Finally, the partial products are added up to get the final product. This step requires a signed-digit multi-operand adder. Binary tree architecture is used to build the multi-operand adder using two-operand signed-digit adders. The block diagram of the Booth-3 multiplier that is designed in this work is shown in Fig. 3

C. Signed Digit Filter

After the multiplier and the multi-operand adder are already implemented, the FIR filter just needs to be assembled. The delay line has been implemented as an array of registers. Since the output of the final adder is still in signed-digit representation, a converter had to be added to convert the result to 2's complement format. In this work, the converter is simply a carry lookahead adder that adds the positive and the negative parts of the signed-digit number. Converting an n-digit signed-digit number X is performed as follows:

$X = (x^+, x^-)_{n-1}\,(x^+, x^-)_{n-2} \ldots (x^+, x^-)_2\,(x^+, x^-)_1\,(x^+, x^-)_0.$
The positive part is $XP = x^+_{n-1}\,x^+_{n-2} \ldots x^+_2\,x^+_1\,x^+_0.$
The negative part is $XN = x^-_{n-1}\,x^-_{n-2} \ldots x^-_2\,x^-_1\,x^-_0$
The 2's complement representation is $Y = XP + XN$.

A scalable and parameterized design has been highly considered. Thus, when the FIR filter is assembled, the filter order and the sample width are defined as design generics in the VHDL code such that the generated filter architecture meets the intended filter parameter. The block diagram of the signed-digit FIR filter implemented in this work is depicted in Fig. 4. If the sample width is W and the filter order is N, the filter output sample will be of width $2W + \lceil \log_2 N \rceil$.

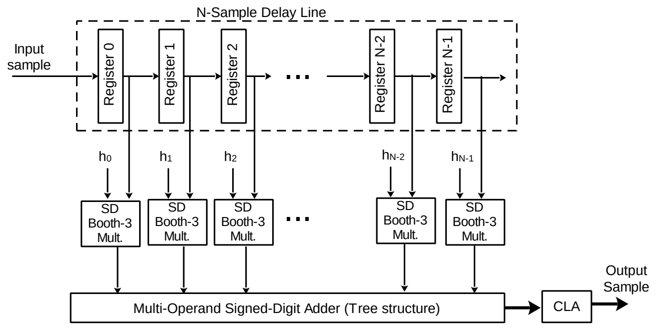


Fig. 4. Signed Digit FIR Filter [12]

IV. Implementing Carry Save Filter

In this section, another method of breaking the carry propagation chain is reported. That is by using carry-save addition (CSA) [11]. CSAs are built using (3, 2) counters in a manner that prevents carry propagation. The term (3, 2) counter is an alternative name for the full adder because it receives three bits of the same weight and outputs two bits representing the number of ones in the three-bit input. In CSA instead of propagating the carry bits to a higher position, these carry bits are kept and added using later stages of the CSA. Carry-save adders are used in FIR filters to add the partial products of the multipliers and to calculate the final result of the filter.

A. Carry-Save Addition

An efficient way of designing a carry-save adder to achieve fast performance is by designing it based on a 3-operand carry-save adder. A k-operand CSA adder (where k > 3), is constructed out of several blocks of 3-operand CSAs [17] [18]. This k-operand CSA could be implemented in two common ways: cascade or tree. The cascade structure accepts one new operand at each level except at the first level where three new operands are accepted. The number of levels in this structure is more than the number of levels in the tree structure, which implies more delay. However, the cascade structure remains a preferred option sometimes due to its regular layout, which implies more simplicity in the VLSI design.

On the other hand, the tree structure, which is known as the Wallace tree [11], accepts as many operands as possible at the first level. The following levels are used to add the sum and the carry vectors in addition to the operands remaining from the first level, which must be at most two remaining operas. When using the tree structure to build a CSA, the number of levels will be less than the cascade structure. [19]

B. Carry-Save Filter

The CSA and Booth-2 multiplier are the fundamental blocks of the carry save FIR filter. As in the signed-digit case, the FIR filter is built by assembling these blocks with some extra logic for the delay line, which is an array of registers, and the converter, which is a carry lookahead adder. The conversion from the carry-save to the 2’s complement format is performed by adding the sum and carry vectors. The filter is illustrated in Fig. 5

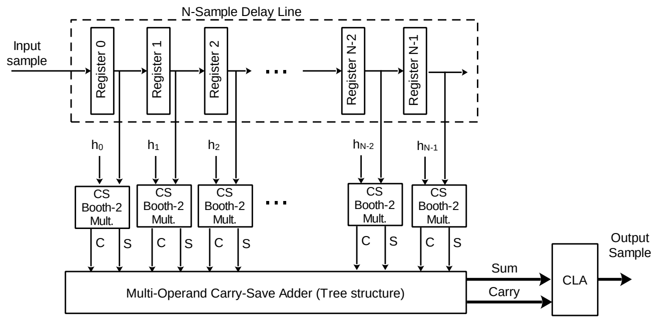


Fig. 5. Carry-Save FIR Filter

V. TESTING AND VERIFICATION

The implementation of the different configurations of the FIR filters have been tested functionally using test benches written in VHDL. Those test benches covered the top-level architectures along with the sub-components throughout the design hierarchy. Moreover, the filters have been synthesized and mapped into an FPGA in order to verify their functionality on real hardware for image processing. This experiment is reported briefly in [12] while described in more details in the following. Such applications are challenging in the sense that two dimensional (2D) FIR filters are needed instead of the one dimensional (1D) filters available in hand. A 2D FIR filter applies the 2D convolution equation:

$$y[m,n] = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} x[i,j]h[m-i,n-j] \tag{5}$$

A sample of an image signal is in indexed by two values: *m* and *n* indicating the row and column position of that sample respectively. Since the filters implemented so far in this work are 1D, they are instantiated in parallel to build a 2D FIR filter of order M-by-N such that each one of the M 1D filters performs convolution operation of order N as illustrated in Fig.6. The kernel h of each filter is assigned as one row of the 2D kernel. The 2D filter of order 11-by-11 that is designed in this work is intended to smooth out the sharp edges of an input image by averaging out each pixel with its neighbors such that the output image is a blurred version of the original one. So, this is a moving average filter, which is a low-pass filter. After blurring the image, the result is subtracted from the original image in order to extract the image edges and cancel out the constant regions. In this work the filter size is 11-by-11 and the filter kernel is

$$h = \begin{bmatrix} \frac{1}{11^2} & \frac{1}{11^2} & \frac{1}{11^2} & \cdots & \frac{1}{11^2} \\ \frac{1}{11^2} & \frac{1}{11^2} & \frac{1}{11^2} & \cdots & \frac{1}{11^2} \\ \frac{1}{11^2} & \frac{1}{11^2} & \frac{1}{11^2} & \cdots & \frac{1}{11^2} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{1}{11^2} & \frac{1}{11^2} & \frac{1}{11^2} & \cdots & \frac{1}{11^2} \end{bmatrix} \tag{6}$$

The above described process is implemented on Altera Cyclone II Starter Development Board. The image source is a normal personal computer. The interface between the FPGA and the computer is the serial port. The results are depicted in Fig. 7

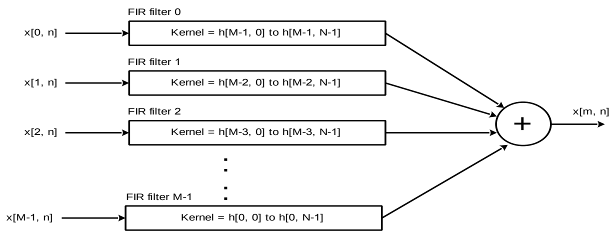
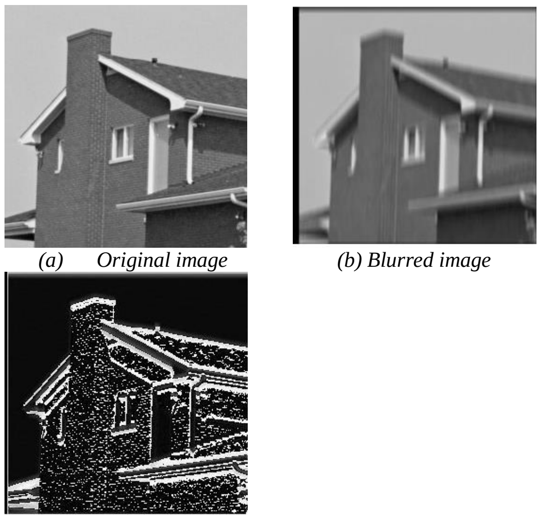

FIR filter 0
x[0, n]
Kernel = h[M-1, 0] to h[M-1, N-1]
FIR filter 1
x[1, n]
Kernel = h[M-2, 0] to h[M-2, N-1]
FIR filter 2
x[2, n]
Kernel = h[M-3, 0] to h[M-3, N-1]
FIR filter M-1
x[M-1, n]
Kernel = h[0, 0] to h[0, N-1]
+
x[m, n]

Fig. 6. 2-D FIR filter [12]



(a) Original image *(b) Blurred image*

(c) Subtracted image

Fig. 7. Edge Extraction using FIR filter [12]

VI. Synthesis Results

The two FIR filter designs have been implemented in a scalable and parameterized manner by exploiting the generality features of VHDL. The sample width W and the filter order N are the two generics of the two filters. Recalling that the major problem that is handled in this work is the carry propagation delay, different values for W, which is directly affecting the carry chain, should be examined. Since we are talking about FIR filters, the order N is also worth examining.

Three precision levels of W are selected: low precision, medium precision and high precision where W is equal to 12, 24 and 48 bits respectively. These values of W are selected because when Booth-3 is applied on them, the number of generated partial products is a power of 2, which reduces the complexity of the multi-operand adder when built as a binary tree. In fact, there is also some attractive and practical advantage for choosing sample widths of 12 and 24 bits. That is, most of the commercial analog-to-digital converters (ADCs) that are used today electronic systems are 12-bit wide and most of the audio codec components used in media systems are 24-bit wide.

For the values of N, arbitrarily chosen values of 16, 32 and 64 samples have been considered. With two filter types, three precision levels and three filter orders, there are 18 different FIR filters to be synthesized. The 18 filters have been synthesized and place-and-routed using Quartus II software. The target FPGA is Altera Stratix III EP3SL340 [20]. The timing and hardware results of the place-and-route process are targeted for analysis. To get accurate timing results, the delay from the primary input to the primary output of the filter should be measured. In fact, some software design tools measure the delay starting from the FPGA pin to which the primary input is assigned and ending at the pin to which the primary output pin is assigned. This method of measuring delay is not accurate when comparing two or more designs because there will be some extra routing delay between the pins of the primary ports. This routing delay is dependent on where the tool places and routes the design and hence is not regular. To avoid this problem in this work, the primary input and the primary output of the filter are latched. Once the tool figures out that there is some logic between two registers, it will calculate the maximum clock frequency fmax allowed for this design. The reciprocal of the frequency (1/ fmax) is the propagation delay of the logic between the two registers, plus some sequencing overhead which is common for all designs. Thus, the presence of this extra delay in the comparison is fair. The place-and-route results for delay and hardware are collected and analyzed. These results are discussed in the following figures.

The delay data collected for the 18 filters are summarized in Fig.8. An observation on the chart is that the latency in both filters is proportional to W and N. The justification of this observation is that the sample width W directly affects the number of partial products that is generated in the Booth multipliers, which in turn increases the number of levels in the multi-operand adder inside the multiplier. Likewise, the value of N affects the number of levels in the multi-operand adder that generates the final filter result. This increase in the adder levels, which is proportional to W and N, results in a larger latency. It is interesting to notice that the delay of the two filters is not equally proportional to W and N; it is highly proportional to W while slightly proportional to N. This is, in fact, due to the delay introduced by the carry lookahead adder that acts as a converter at the last stage in both filters. A CLA is a carry propagate adder and it is expected to be highly tied to W. When this CLA has been separately analyzed, it has been found that it is responsible for about 28 % of the overall filter delay. This explains why W has more influence on the filter delay than N.

Another observation is that the signed-digit filter is always faster than the carry-save filter. This improvement in filter performance needs more analysis in order to see how the improvement behaves with respect to the design parameters and how significant it is. Fig. 9 depicts the ratio of the signed-digit filter delay over the carry-save filter delay. Clearly, the delay improvement is very slight since the chart indicates that signed-digit filter performs between 1.1 and 1.2 times faster than the carry-save filter. The speedup is almost constant regardless of the values of W and N.

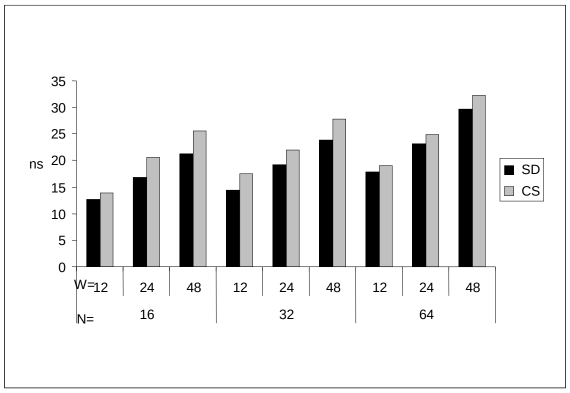


Fig. 8. FIR filter Delays

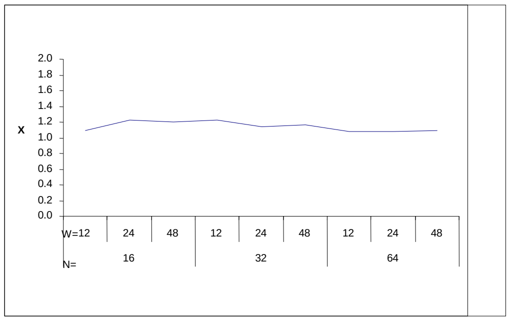


Fig. 9. Ratio of SD- to CSA FIR filter delays

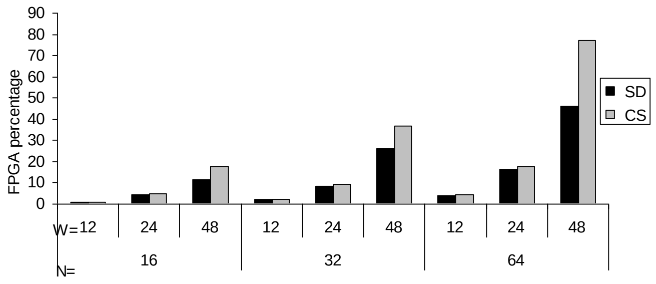


Fig. 10. Logic Utilization

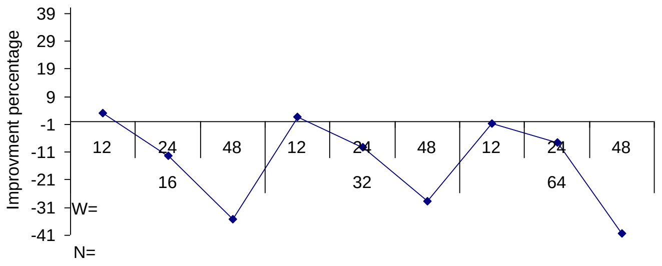


Fig. 11. Logic Utilization Percentage

The data of the logic utilized by the 18 filters is also collected and analyzed. The place-and-routing logic utilization results are summarized in Fig. 10. While Fig. 11 shows the percentage of hardware reduction in the signed-digit filter with respect to the carry-save filter. It is logical and expected to see that the hardware of the two filters grows as the values of W and N increase. It is noticeable that the hardware utilization when W = 12 bits, regardless of the value of N, is almost the same for the two types of filters. With higher values of W, the signed-digit filter has an advantage. The case of having the signed-digit filter being smaller than the carry-save filter despite that the added complexity of the signed-digit adder is may by the significance of Booth multipliers in the filters since most of the filter size is occupied by them. Booth multiplier efficiency in saving hardware and time becomes more significant and appreciated when the multiplier gets bigger. This is what makes the logic difference more notable when W is equal to 24 and 48 bits. From Fig. 11, the signed-digit filter is between 30 % and 40 % smaller than the carry-save filter for W = 48. This might seem counterintuitive. However, this could be justified by amount of multipliers used, and the fact that the reduction from Booth 2 to Booth 3 is the cause for this reduction in size.

VII. Conclusions

In this work, FIR filter design and implementation have been approached from arithmetic perspective. The signed-digit and the carry-save arithmetic techniques have been exploited to reduce addition time. Booth encoding was used to speedup multiplication. The authors designed, simulated and tested a high-speed FIR filter using the signed digit number system. The first part of work is the design and implementation of the FIR filter using the signed-digit number system and Booth-3 encoding to improve the filter adders and multipliers respectively. The implementation of the signed-digit two-operand and multi-operand adders has been discussed. In Booth-3 implementation, it has been shown how the signed-digit number system helps in generating the hard multiple 3M. The other part of this work is the design and implementation of an FIR filter using carry-save addition and Booth-2 encoding to improve the filter adders and multipliers respectively. The hierarchical design of CSAs of several sizes has been reported.

For the two types of filters in this work, nine different configurations have been considered for the sample width W (12, 24, 48 bits) and for filter order N (16, 32, 64) samples. A total of 18 filters of both types have been modeled and generated in VHDL. Then, these filters have been synthesized and place-and-routed. The data resulted from the place-and-rout process, which is related to system delay and logic size, has been collected and analyzed.

The results analysis have shown that the signed-digit FIR filters designed in this work are slightly faster than the carry-save FIR filters. The filter delay is slightly proportional to N, but highly proportional to W. The signed-digit filters are constantly about 1.1 times faster than the carry-save filters. Both types of filters have consumed almost the same amount of logic for low precision samples while they differ in logic utilization as the precision increases. The signed-digit filter reported better logic utilization especially for W = 48 bits where it becomes 30 % to 40 % smaller than the carry-save filter.

In conclusion, both the signed-digit and the carry-save filters are fast and efficient because of the carry-propagate-free addition they involve. The speedup that is gained in the FIR filter when signed-digit arithmetic is used is not so significant. Likewise, the filter size reduction for a sample width around 12 bits is almost negligible. However, the improvement in logic utilization for wider samples is strongly significant. Therefore, designing FIR filters using signed-digit number system becomes efficient and useful more than carry-save filters when the filter works for high precision samples. However, the signed-digit filter is superior over the carry-save filter in logic utilization more than speed.

REFERENCES

[1] S. Smith, The Scientist and Engineer's Guide to Digital Signal Processing, San Diego: California Technical Publishing, 1997.

[2] C.-J. Chou, S. Mohanakrishnan and J. B. Eva, "FPGA Implementation of Digital Filters," in International Conference of Signal Processing Applications and Technology ICSPAT '93, Santa Clara, CA, 1993.

[3] B. Parhami and D.-M. Kwai, "Parallel Architectures and Adaptation Algorithms for Programmable FIR Digital Filters with Fully Pipelined Data and Control Flows," Journal of Information Science and Engineering, no. 19, pp. 59-74, 2003.

[4] X. Jiang and Y. Bao, "FIR filter design based on FPGA," in International Conference on Computer Application and System Modeling (ICCASM) , Taiyuan, 2010.

[5] J. Deschamps and M. Davio, "Addition in Signed Digit Number System," in Proceedings of the eighth international symposium on Multiple-valued logic , Rosemont, Illinois, United States , 1978.

[6] D. Atkins, "Introduction to the Role of Redundancy in Computer Arithmetic," Computer, vol. 8, no. 6, pp. 74-77, June 1975.

[7] A. Avizieni, "Binary Compatible Signed Digit Arithmetic," AFIPS Conference Proceedings, vol. 26, no. 1, pp. 664-672, 1964.

[8] P. Ramamoorthy, B. Potu and G. Govind, "DSP System Architecture Using Signed-Digit Number Representations," ICASSP, vol. 3, pp. 1702-1705, April 1988.

[9] C. Nagendra, M. Irwin and R. M. Owens, "Area Time Power Tradeoffs in Parallel Adders," IEEE Transaction on Circuits and Systems, vol. 43, no. 10, pp. 689-702, October 1996.

[10] I. Koren, Computer Arithmetic Algorithms, Natick: A. K. Peters, 2002.

[11] S. Wallace, "A Suggestion for a Fast Multiplier," IEEE Transactions of Electronic Computers, pp. 14-17, Febuary 1964.

[12] Y. M. Seddiq and H. A. Altwaijry, "An Implementation of a 2D FIR Filter Using the Signed-Digit Number System," in *Saudi International Electronics, Communications and Photonics Conference (SIECPC2011)*, Riyadh, pp. 1-4. 2011.

[13] O. McSorley, "High Speed Arithmetic in Binary Computers," Proceedings of the IRE, vol. 49, no. 1, pp. 67-91, January 1961.

[14] H. Makino, Y. Nakase, H. Suzuki, H. Morinaka, H. Shinohara and K. Mashiko, "An 8.8ns 54 x 54 Bit Multiplier wuth High Speed Redundant Binary Architecture," IEEE Journal of Solid State Circuits, vol. 31, no. 6, pp. 773-783, June 1996 .

[15] J. Fadavi-Ardenkani, "M x N Booth Encoded Multiplier Generator Using Optimized Wallace Trees," IEEE Transaction on Very Large Scale Integration (VLSI) System, vol. 1, no. 2, pp. 120-125, June 1993.

[16] G. DeMicheli and P. Song, "Circuit and Architecture Tradeoffs for High Speed Multiplication," IEEE Journal of Solid State Circuits, vol. 26, no. 9, pp. 1184-1198, September 1991.

[17] L. Dadda, "Some Schemes for Parallel Multipliers," Alta Frequenza, vol. 34, pp. 349-356, 1965.

[18] D. Booth, "A Signed Binary Multiplication Technique," Quarterly Journal of Mechanics and Applied Mathematics, vol. 4, no. 2, pp. 236-240, 1951.

[19] N. Besli, A Novel Arithmetic Unit Using Redundant Binary Signed Digit Number System, Ph.D. Thesis, Florida Institute of Technology, 2004.

[20] "www.altera.com," [Online].