A HIGHLY-SCALEABLE FIR USING THE RADIX-4 BOOTH ALGORITHM

Oscal T.-C. Chen, Wei-Lung Liu

Signal and Media Laboratories,
Department of Electrical Engineering,
National Chung Cheng University,
Chia-Yi, Taiwan, R.O.C.

Hsun-Chang Hsieh, Jeng-Yih Wang

Digital Signal Processing Dept.,
Computer & Communication Research Lab.,
Industrial Technology Research Institute,
Hsinchu, Taiwan, R.O.C.

ABSTRACT

A highly-scaleable FIR architecture based on the radix-4 Booth algorithm has been designed with scaleable dynamic ranges of input data and filter coefficients. The radix-4 Booth algorithm is demonstrated to have a lower hardware complexity and a fair throughput rate than the other radix approaches. In order to achieve scaleability, the configurable-connection function between latches of input data, and filter taps has been explored. The precision of filter coefficients is adjustable by using a path-control function. Especially, the proposed architecture only employs data-path controls to realize the scaleable issue without changing the word lengths and components of input latches and filter taps. The pre-processing unit for manipulating input data and post-processing unit for computing accumulation results have been realized to support scaleable operations. Based on our architecture in a chip design, the cascaded configuration between chips is also easily accomplished for many industrial applications.

1. INTRODUCTION

Filters of finite impulse response applied in real-world applications have many advantages of easy implementation, noise immunity, sharp cut-off frequencies, high stability and so on [1]. The major operation of a FIR filter is the convolution that is realized by using adders, multipliers, and delay elements. However, a multiplier takes a lot of computational time to perform its function. In order to reduce the complexities, high-speed FIR filters without using multipliers have been proposed by many researchers [2-4]. These multiplierless filters can be classified by a memory-based approach, a canonical signed-digit (CSD) approach, and a Booth-algorithm approach.

The simplified FIR design in the above three approaches allows easy incorporation of programmability. However, scaleable dynamic ranges of input data and filter coefficients are not straight-forward achievable. In the memory-based FIR design, the word length of input data and precision of filter coefficients are usually fixed for one memory configuration. In order to achieve scaleability, we have to configure the memory cells and rearrange connections among taps. Due to high cost of the original architecture for a large dynamic data range, the memory-based FIR may not be a good candidate for scaleable design. In the CSD FIR design, filter coefficients are easily scaleable but functional units in each tap require the maximum word-length design. All CSD taps are directly addressed by every input datum using the fixed word-length hardware. When considering a large dynamic range of input data, we need partition input data to be a sub-datum sequence. Hence, there is a need of the complicated tap design to support this data flow. The scaleable CSD FIR cannot be realized at a low cost.

In the Booth-algorithm FIR design, bit-level input data can be easily scaled for different dynamic ranges, and precision of filter coefficients can be adjustable due to the regular structure of each tap. Here, we propose a new architecture of scaleable FIR based on the Booth algorithm. This architecture provides users more flexibly to manipulate input data and filter coefficients. Especially, the proposed architecture only employs data-path controls to accomplish the scaleable issue. In addition, the pre-processing unit for manipulating input data and post-processing unit for computing accumulation results have been designed to support scaleable operations. When considering the proposed architecture in a chip design, cascading capability between chips for a FIR with a large number of taps is also addressed for many industrial applications.

2. MODIFIED BOOTH ALGORITHM

In 1951, Andrew Booth presented the radix-2 algorithm to multiply two signed 2's complement numbers, which was a simplified high-speed multiplication operation. Since then, other researchers have proposed many modified Booth's algorithms in high radices to improve computational efficiency [5,6]. By using a high-radix Booth algorithm, fewer partial products are required to accomplish a final result but generation of these partial products is more complicated. Therefore, it is very critical how to incorporate an adequate radix number of Booth algorithm in a FIR design with low-cost and high-speed performances.

The radix-2 multiplication is realized by using decoding, addition, subtraction, and shifting operations. Based on every neighboring two bits of input data, the decoding operation determines the intermediate products. The shifting operation scales these intermediate products to be correct forms for addition or subtraction. As compared to the conventional multiplication, this approach shows very efficient and easy in hardware implementation. However, the radix-2 Booth algorithm requires W additions per multiplication where W is the word length of input data.

The multiplication interpreted in a high radix is a powerful method to achieve a high-speed performance. Here, FIR using the radix-4 Booth algorithm is introduced with input data of $\boldsymbol{X}$ and filter coefficients of $\boldsymbol{C}$. Each datum of $\boldsymbol{X_i}$ is partitioned into many 3-bit groups, triplets, each of which has one bit overlapped with the previous group. This triplet can be represented by the following,

$$X_{i,l} = \left\{ X_i^{2l+1}, X_i^{2l}, X_i^{2l-1} \right\} \quad (1)$$

where l=0, 1, …, $W/2-1$, X_i^j is the j[th] digit of X_i and $X_i^{-1} = 0 \cdot X_i^{2l-1}$ is overlapped with the previous triplet of $X_{i,l-1}$ such that the 2's complement of X_i can be represented by

$$X_i = -X_i^{W-1} \times 2^{W-1} + \sum_{j=0}^{W-2} X_i^j \times 2^j = \sum_{l=0}^{W/2-1} \left(-2X_i^{2l+1} + X_i^{2l} + X_i^{2l-1}\right) \times 2^{2l} \quad (2)$$

When considering C_j multiplied with X_i, the equation (2) is modified to

$$C_j \times X_i = \sum_{l=0}^{W/2-1} \left(-2X_i^{2l+1} + X_i^{2l} + X_i^{2l-1}\right) \times C_j \times 2^{2l} = \sum_{l=0}^{W/2-1} B\left(X_{i,l}, C_j\right) \times 2^{2l} \quad (3)$$

According to Eq. (3), $B\left(X_{i,l}, C_j\right)$ is the intermediate product which can be represented by 5 different values,

$$B\left(X_{i,l}, C_j\right) = \begin{cases} 0 & if \quad X_{i,l} = \{0,0,0\}, \{1,1,1\} \\ C_j & if \quad X_{i,l} = \{0,1,0\}, \{0,0,1\} \\ -C_j & if \quad X_{i,l} = \{1,1,0\}, \{1,0,1\} \\ 2C_j & if \quad X_{i,l} = \{0,1,1\} \\ -2C_j & if \quad X_{i,l} = \{1,0,0\} \end{cases} \quad (4)$$

From Eq. (3), we observe that convolution between the input data and filter coefficients requires $W/2$ summations based on the intermediate products of -2C$_j$, -C$_j$, 0, C$_j$, and 2C$_j$. In such an approach, the intermediate products are easily obtained by using 1-bit shifting, and/or setting the carry-in bit of an adder to logic-1 and inverting the input signal for -C$_j$, 2C$_j$ or -2C$_j$. Hence, the complexity of decoding function in the radix-4 approach is very low.

When considering a higher radix approach, radix-8 multiplication groups 4 bits of ***X*** at a time with one bit overlapped with the previous set. Its intermediate products consist of -4C$_j$, -3C$_j$, -2C$_j$, -C$_j$, 0, C$_j$, 2C$_j$, 3C$_j$, and 4C$_j$. The 3C$_j$ requires an additional hardware component to implement itself. Although the radix-8 approach reduces intermediate addition steps for a shorter latency, each step takes little more hardware complexities than that in the radix-4 approach. Furthermore, a radix-16 multiplication employs 5 bits of ***X*** at a time with one bit overlapped with previous set. The intermediate products include -8C$_j$, -7C$_j$, -6C$_j$, -5C$_j$, -4C$_j$, -3C$_j$, -2C$_j$, -C$_j$, 0, C$_j$, 2C$_j$, 3C$_j$, 4C$_j$, 5C$_j$, 6C$_j$, 7C$_j$, and 8C$_j$. The 3C$_j$, 5C$_j$, 6C$_j$, and 7C$_j$ needs additional hardware components to realize themselves.

In order to effectively analyze hardware complexity, we explored an N-tap FIR using the Booth algorithm, which is shown in Fig. 1. The requirement of data latches for storing input data in different radix approaches is considered. The 3C$_j$, 5C$_j$, 6C$_j$, and 7C$_j$ are simply implemented by using data latches. Table 1 lists the comparison of hardware complexities and throughput rates of radix-2, radix-4, radix-8, and radix-16 approaches. The hardware complexity of radix-4 approach is smaller than those of radix-8 and radix-16 approaches if $L \times N + \frac{3}{2}W \times (N-1)$ is smaller than $(2L+2) \times N + \frac{4}{3}W \times (N-1)$ and $(5L+11) \times N + \frac{5}{4}W \times (N-1)$. Here, *L* and *W* are the word lengths of filter coefficients and input data, respectively. Especially, we can summarize these two conditions to that of *6L+12* larger than $W \times (1 - \frac{1}{N})$, which is always true for reasonable dynamic ranges of input data and filter coefficients. When considering routing and multiplexing of decoding functions, the radix-4 approach always has a lower complexity than radix-8 and radix-16 approaches. In addition, the hardware complexity of radix-4 approach can be smaller than that of radix-2 approach due to less requirement of input data latches. On the other hand, the throughput rate of radix-4 approach is worse than those of radix-8 and radix-16 approaches, and better than that of radix-2 approach. Therefore, the radix-4 approach in the FIR design can have the least hardware complexity and a fair throughput rate.

3. PROPOSED FIR ARCHITECTURE

The VLSI implementation plays a key role in developing high-speed, low-cost, light-weight, and low-power applications. Based on pipeline, parallel, or programmable schemes, various FIR architectures have been proposed by many researchers to pursue high-throughput and cost-effective designs. The main challenge would be to optimize flexible architectures for various FIR applications at a low cost. In this paper, we propose a programmable FIR architecture with scaleable dynamic ranges of input data and filter coefficients. This design provides users more flexibly to handle the dynamic data ranges as well as programmable coefficients.

A previous example of the *N*-tap FIR with output data of ***Y*** is illustrated further. The relationship between input signals and output signals can be described as follows,

$$Y_n = \sum_{i=0}^{N-1} C_i \times X_{n-i} \quad (5)$$

The multiplication between X_{n-i} and C_i can be accomplished by the radix-4 Booth algorithm. By applying Eq. (3) to Eq. (5) , we can obtain

$$Y_n = \sum_{i=0}^{N-1} C_i \times X_{n-i} = \sum_{i=0}^{N-1} \left[\sum_{l=0}^{W/2-1} B\left(X_{n-i,l}, C_i\right) \times 2^{2l} \right] \quad (6)$$

According to Eq. (6), we can construct the FIR architecture requiring accumulations in each tap to sum up intermediate products. In such a design, each tap consists of a coefficient latch, a Booth decoder, an adder, a 2-to-1 multiplexor, and an accumulation latch, as shown in Fig. 1. In order to improve this architecture, Lee et al. rearranged Eq. (6) to obtain the following equation [2],

$$Y_n = \sum_{l=0}^{W/2-1} \left[\sum_{i=0}^{N-1} B\left(X_{n-i,l}, C_i\right) \right] \times 2^{2l} \quad (7)$$

Based on Eq. (7), the modified FIR architecture is shown in Fig. 2. The accumulation in each tap is moved to the post-processing unit

such that the word length and hardware components of each tap are optimized. In order to achieve scaleable dynamic ranges of input data and filter coefficients, Eq. (7) is modified by the following way,

$$Y_n = \sum_{l=0}^{W/2-1} \sum_{j=0}^{S} \left[\sum_{i=0}^{N-1} B\left(X_{n-i,l}, C_{i,j}\right) \right] \times 2^{2l} \qquad (8)$$

where $C_{i,j}$ is the j^{th} sub-precision component of C_i, and S is the number of sub-coefficients. According to Eq. (8), scaleability can be realized in the control of W and S. Since input data are recoded in the radix-4 format, scaleable data ranges can be pursued by configuring the connections between input data latches and filter taps. In such a design, various dynamic ranges can be achieved when the pre-processing unit provides the correct triplet sequence and the post-processing unit provides an enough data bandwidth for accumulation of intermediate results. On the other hand, precision of filter coefficients can be scaled by configuring the connections among taps while the post processing unit can sum the sub-precision values to yield a correct result. Figure 3 shows the proposed scaleable architecture in the FIR design. Especially, our architecture only employs data-path controls to realize the scaleable issue, which can make the FIR design more regular and low-cost.

In the implementation of our scaleable FIR, it is very important how to efficiently and effectively realize the configurable connections. For designing scaleable dynamic ranges of input data, simple multiplexors can be utilized to configure the connection topology as shown in Fig. 4. The paths of each filter tap connected to several input data latches are controlled by a multiplexor for different scaleable ranges. Only one path is enabled to link a filter tap to the corresponding latch of input data. The control unit interprets scaleable ranges and generates control signals for multiplexors to determine the correct paths.

Precision of filter coefficients can be adjusted by controlling data paths among taps. Since the word length of latches for storing filter coefficients is fixed, we can arrange a filter coefficient stored in several latches. According to Eq. (8), each filter coefficient is partitioned into several non-overlapped sub-coefficients which are individually utilized in the decoding function of the Booth algorithm. However, the sign bits of non-overlapped sub-coefficients require additional consideration as follows,

$$\begin{aligned} C_i &= -c_i^{P-1} \times 2^{P-1} + \sum_{j=0}^{P-2} c_i^j \times 2^j \\ &= \sum_{k=0;\, k=k+\frac{P}{S}}^{t<P-1} \left(-c_i^{P-1} \times 2^{t+\frac{P}{S}} + \sum_{j=k}^{k+\frac{P}{S}-1} c_i^j \times 2^j + c_i^{P-1} \times 2^t \right) - c_i^{P-1} \times 2^0 \end{aligned} \qquad (9)$$

where c_i^k is the k^{th} digit of C_i, and P is the coefficient precision. In other words, the sign bit of a filter coefficient becomes the sign bits of its sub-coefficients. In order to compensate the additional negative values, the least significant bit of all sub-coefficients except the lowest-precision one is added by this sign bit.

In the design of the path-control function, the main idea is to arrange the accumulation relationship among the decoded values of sub-coefficients and input data. In order to effectively to realize the path-control function, a latch is required in neighboring taps. The post-processing unit is utilized to sum the partial results for generating a correct-format value. Figure 4 shows the path-control function in the FIR design, which consists of N multiplexors addressed by the control unit.

The pre-processing unit for manipulating input data and the post-processing unit for generating output results are designed to support the scaleable FIR computing. The pre-processing unit, as shown in Fig. 5, consists of an input buffer, a data latch, a comparator, a ripple counter, and a multiplexor. The input buffer is used to store various dynamic ranges of input data. The data latch records the maximum value of the counter for the currently-used dynamic data range. The comparator is to compare the output value of the counter with that of the data latch. If they are the same, the counter is cleared to zero. The ripple counter is utilized to generate the control signals of the multiplexor for selecting the three-bit data in a correct sequence. On the other hand, the post-processing unit performs accumulation for final results. First, the selector of sub-precision results is used to determine the effective data according to the operation mode of coefficient precision. These effective results with sign extension are summed to become a correct output result by using the adder tree. The output results are accumulated at $W/2$ times for a final convolution value. Figure 6 shows the post-processing unit of which major components are adders, shifters, latches, and multiplexors for supporting various dynamic ranges of input data and filter coefficients.

If the tap number of FIR cannot be realized in one chip, the last latch of input data is connected to the output pins for cascading next chip. However, the latches of input data may be not completely used due to the configurable connections for various dynamic data ranges. The bypass design can be utilized to get rid of the serial pipelined flow. The last datum for filter operations in the first chip can be easily transmitted to the input latch of the second chip without delay. On the other hand, the accumulated result in the first chip is also transmitted to the post processing unit of the second chip. In such a design as shown in Fig. 7, the chip cascading can be easily accomplished without additional off-chip logic functions.

4. REFERENCES

[1] A. Oppenheim, R. Schafer, *Discrete-Time Signal Processing*, Prentice Hall, New Jersey, 1989.

[2] H. Lee, C. Jen, C. Liu, "A new hardware-efficient architecture for programmable FIR filters," *IEEE Trans. on Circuits and Systems - II: Analog and Digital Signal Proc.*, pp. 637-644, Sept. 1996.

[3] K. Khoo, A. Kwentus, A. Willson, "A programmable FIR digital filter using CSD coefficients," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 6, pp. 869-874, June 1996.

[4] C. Golla et al., "A 30M samples/s programmable filter processor," *Digest of IEEE Int'l Solid-State Circuits Conf.*, pp. 116-117, 276, Feb. 1990.

[5] R. Katti, "A modified Booth algorithm for high radix fixed-point multiplication," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 2, no. 4, pp. 522-524, Dec. 1994.

[6] P. Madrid, B. Millar, E. Swartzlander, "Modified Booth algorithm for high radix fixed-point multiplication," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 1, no. 2, pp. 164-167, June 1993.

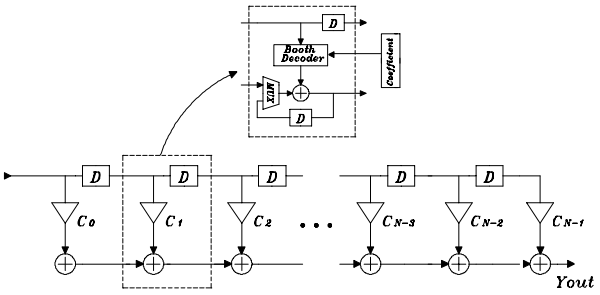

Fig. 1 The N-tap FIR using the Booth algorithm

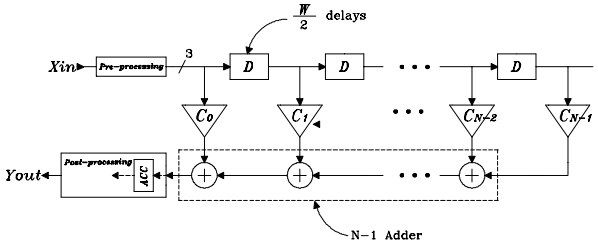

Fig. 2 The modified FIR with accumulation-free taps.

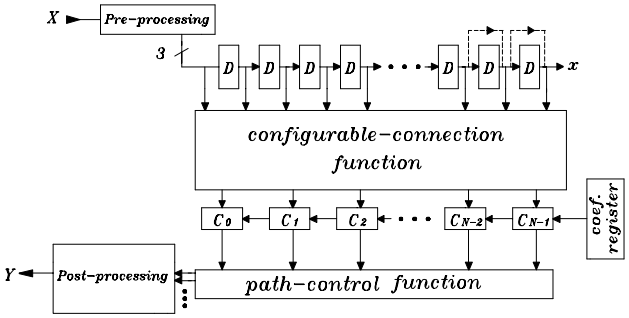

Fig. 3 The proposed scaleable FIR architecture.

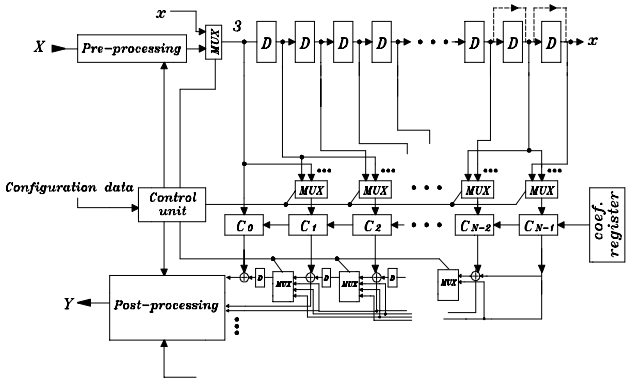

Fig. 4 The design of configurable-connection and path-control functions for the proposed FIR.

Table 1 The comparison of hardware complexities and throughput rates of radix-2, radix-4, radix-8, and radix-16 approaches.

Features \ Schemes		Radix 2	Radix 4	Radix 8	Radix 16
Latches (bits)	coefficients	L×N	L×N	(2L+2) ×N	(5L+11) ×N
	input data	2W(N-1)	$\frac{3W}{2}$ (N-1)	$\frac{4W}{3}$ (N-1)	$\frac{5W}{4}$ (N-1)
Throughput rates		$\frac{C}{W}$	$\frac{2C}{W}$	$\frac{3C}{W}$	$\frac{4C}{W}$
Bit number of the required latches: Example: L=8, W=8, N=64		1520	1268	1824	3894

(Note: N: number of FIR taps; W: word length of input data; L: word length of coefficients; C: clock rate.)

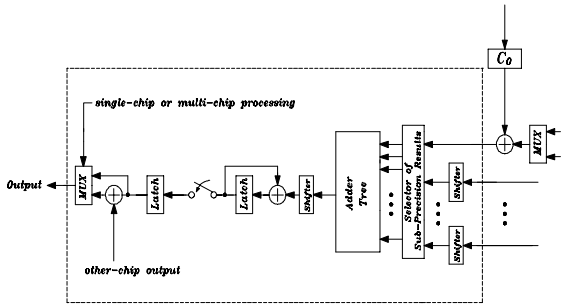

Fig. 5 The pre-processing unit.

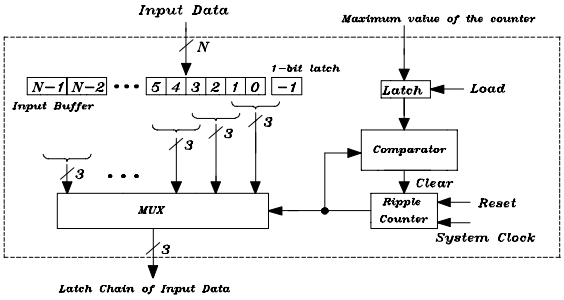

Fig. 6 The post-processing unit.

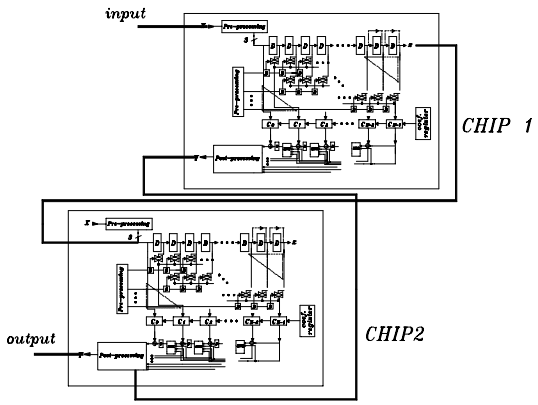

Fig. 7 Chip cascading of the proposed FIR.