A High-Speed Multiplier-Free Realization of IIR Filter Using ROM's

Thanyapat Sakunkonchak and Sawasd Tantaratana
Department of Electrical Engineering, Sirindhorn International Institute of Technology
Thammasat University, Rangsit Campus, Pathumthani, 12121, Thailand
E-mail: thong@siit.tu.ac.th, sawasd@siit.tu.ac.th

Abstract: In this paper, we propose a high-speed multiplier-free realization using ROM's to store the results of coefficient scalings in combination with higher signal rate and pipelined operations. We show that hardware multipliers are not needed. By varying some parameters, the proposed structure provides various combinations of hardware and clock speed (or throughput). An example is given comparing the proposed realization with the distributed arithmetic (DA) realization. Results show that with proper choices of the parameters the proposed structure achieves a faster processing speed with less hardware, as compared to the DA realization.

1. Introduction

Realization of digital filters has been investigated in a variety of ways during the past two decades. Some realizations attempt to use more efficient multipliers, while others avoid hardware multipliers.

Distributed arithmetic (DA) is an efficient way to compute a dot product using partial products, each of which can be obtained by table look-up. The first description of DA was presented by Peled and Liu [1],[2]. White [3] presented a review on DA and pointed out ways to reduce the hardware (hence, chip area) and/or increase the processing speed.

Although DA provides partial products efficiently; when the filter order becomes large, the VLSI silicon area and processing speed of DA become less attractive. The size of ROM increases exponentially as the filter order increases. The speed of a DA realization is limited by the time response of ROM and accumulation of partial products. However, an attempt to reduce the ROM size results in reduced speed. Speed can be increased at the expense of more memory [3]. In [4], we proposed a realization using table look-up in combination with higher signal rate and pipelined operation to obtain a multiplier-free realization. The idea was to break the input signal into subsignals, use ROM's to store the results of scaling by the filter coefficients, and then shift-and-accumulate to obtain the final output. Processing of the subsignals is done at an elevated clock speed. However, the simpler computation (shift and add) relieves the constraint of higher clock speed. In addition, operation at higher clock speed allows reuse of the same hardware, yielding hardware saving.

This work was supported by the National Electronic and Computer Technology Center (NECTEC), the National Science and Technology Development Agency (NSTDA), Thailand, through grant #07/2542

This paper extends the result in [4] by re-arranging the structure to improve the speed with a small increase in hardware. The DA realization and the proposed multiplier-free realization using ROM's are presented in Section 2 and 3, respectively. Pipelining the adder in order to increase the processing speed of the system is described in Section 4. Section 5 compares the area and speed of the proposed structure with those of DA realization. Finally, a conclusion is given in Section 6.

2. Distributed Arithmetic Realization

Consider an N^{th} order IIR filter with the output $y(n)$ given by

$$y(n) = \sum_{i=0}^{N} a_i x(n-i) + \sum_{i=1}^{N} b_i y(n-i) \qquad (1)$$

where $x(n)$ is the input of the filter. Suppose the signals $x(n)$ and $y(n)$ have wordlength of J bits in two's complement representation, i.e. $x(n) = -x_{(0)}(n) + \sum_{j=1}^{J-1} x_{(j)}(n)2^{-j}$, where $x_{(0)}(n)$ is the sign bit, and a similar form for $y(n)$. Then, (1) can be written as [3]

$$y(n) = \sum_{j=1}^{J-1} \mathbf{A}\mathbf{X}_{(\mathbf{j})}^{\mathbf{T}} \mathbf{2^{-j}} - \mathbf{A}\mathbf{X}_{(\mathbf{0})}^{\mathbf{T}} \qquad (2)$$

where $\mathbf{A} = [a_0\ a_1 \ldots a_N\ b_1 \ldots b_N]$ and $\mathbf{X}_{(\mathbf{j})} = [x_{(j)}(n)\ x_{(j)}(n-1) \ldots x_{(j)}(n-N)\ y_{(j)}(n-1) \ldots y_{(j)}(n-N)]$.

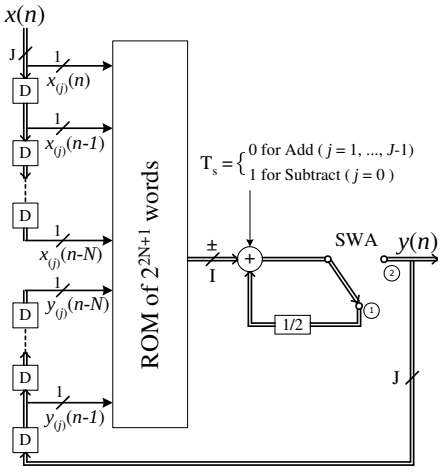


Fig. 1. A DA realization for an N^{th} order IIR filter.

We can see from (2) that the output $y(n)$ is obtained by adding partial results for $j = 1$ to $J-1$ and subtracting the partial results for $j = 0$. Since the coefficient $\mathbf{A}$ is fixed, and $x_{(j)}(n)$ takes value of 0 or 1, the

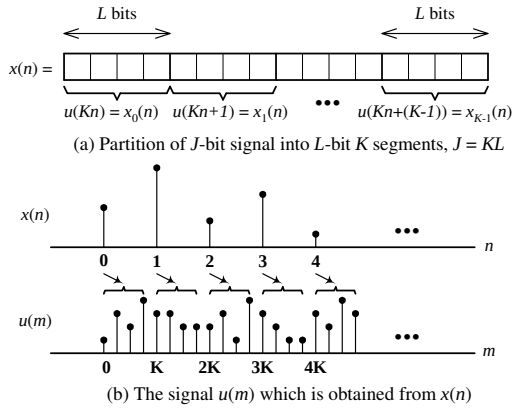
value of $\mathbf{A}\mathbf{X}_{(\mathbf{j})}^{\mathbf{T}}$ can be pre-computed for all 2^{2N+1} possible values of vector $\mathbf{X}_{(\mathbf{j})}$ and stored in a ROM of 2^{2N+1} words each of I bits. A block diagram of the computation is depicted in Figure 1. The signal T_s dictated whether the partial result is added (for $j = 1, \ldots, J-1$) or subtracted (for $j = 0$) from the accumulated result. By some re-arrangement, the ROM size may be reduced by a factor of 2 to 2^{2N}. Note that the ROM size grows exponentially as N increases.

The speed of DA may be increased in two ways: one at the expense of linearly increased memory plus more arithmetic operations, the other at the expense of exponentially increased memory, see [3] for detail.

3. The Proposed Multiplier-Free Realization

In this section, we describe the proposed realization which uses ROM's in combination with elevated signal rate to obtain multiplier-free pipelined operation.

We assume that the input signal $x(n)$ has a wordlength of J bits. Consider a partitioning of $x(n)$ into K segments (subwords) of L bits each, with $J = KL$, as shown in Figure 2(a). Note that $x_0(n)$, $x_1(n)$, ..., $x_{K-1}(n)$ are the subwords. Define $u(m)$ as the signal obtained from time-multiplexing the K subwords, as shown in Figure 2(b). Hence, $u(m)$ is a signal of wordlength L bits and a signal rate which is K times the signal rate of $x(n)$. Note that we will use n and m for the time index of signals at the rate of $x(n)$ and $u(m)$, respectively. Furthermore, in a block diagram, signals with the same rate as $x(n)$ will be represented by thick lines, and signals with the higher rate by thin lines. As seen from Figure 2, we can write $x(n)$ and



(a) Partition of J-bit signal into L-bit K segments, $J = KL$

(b) The signal $u(m)$ which is obtained from $x(n)$

Fig. 2. Partition of $x(n)$ to yield $u(m)$

$ax(n)$ as

$$x(n) = \sum_{k=0}^{K-1} x_k(n)2^{-kL} = \sum_{k=0}^{K-1} u(Kn+k)2^{-kL}$$

$$ax(n) = \sum_{k=0}^{K-1} au(Kn+k)2^{-kL} \quad (3)$$

The partial result $au(Kn+k)$, k=0,...,K-1, can be obtained by table look-up of a ROM which has a content of 2^L words. The partial results are then scaled by 2^{-kL} and accumulated. As we shall see, the scalings and accumulations for all the filter coefficients are performed by only one scale and accumulate unit, providing hardware reduction.

Before we can obtain the proposed pipelined structure, we have to transform (1) so that $y(n)$ does not depend on $y(n-1)$ because the latency of the resulting realization prevents $y(n-1)$ to be available in time. To this end, we write from (1)

$$y(n-1) = \sum_{i=1}^{N+1} a_{i-1}x(n-i) + \sum_{i=2}^{N+1} b_{i-1}y(n-i) \quad (4)$$

Substituting this in (1) yields

$$\begin{aligned} y(n) &= a_0x(n) + a_Nb_1x(n-(N+1)) + b_1b_Ny(n-(N+1)) \\ &\quad + \sum_{i=1}^{N}(a_i + a_{i-1}b_1)x(n-i) + \sum_{i=2}^{N}(b_i + b_1b_{i-1})y(n-i) \\ &\triangleq \sum_{i=0}^{N+1} c_ix(n-i) + \sum_{i=2}^{N+1} d_iy(n-i) \end{aligned} \quad (5)$$

where $c_0 = a_0$, $c_i = a_i + b_1a_{i-1}$, $i = 1, \ldots, N$, and $c_{N+1} = a_Nb_1$; and similarly for d_i.

Based on the above idea in (3), we can realize the IIR filter (5) as depicted in Figure 3. Each ROM contains 2^L word storing the results of $c_iu(\cdot)$ or $d_iv(\cdot)$, and accessed by $u(\cdot)$ or $v(\cdot)$. Note that the scalings by 2^{-kL} for all the coefficients are performed by a single unit inside the dotted box in Figure 3, providing hardware saving. The scaled and accumulated result is delayed and downsampled by a factor of K to obtain $y'(n)$ at the original signal rate.

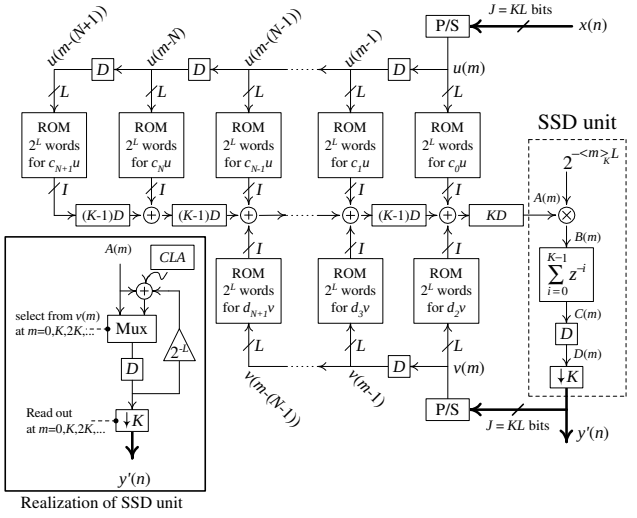


Fig. 3. The proposed realization of N^{th} order IIR filter

To derive the input/output relation of Figure 3, we note that

$$u(m) = x_{<m>_K}\left(\left\lfloor \frac{m}{K} \right\rfloor\right), \quad v(m) = y'_{<m>_K}\left(\left\lfloor \frac{m}{K} \right\rfloor\right) \quad (6)$$

where $\lfloor\gamma\rfloor$ denotes the integer part of γ, and $< m >_K$ is $m \mod K$. The signal before scaling is

$$A(m) = \sum_{i=0}^{N+1} c_i u(m-(i+1)K) + \sum_{i=2}^{N+1} d_i v(m-(i-1)K) \quad (7)$$

Inside the dotted box, we have

$$B(m) = A(m)2^{-<m>_K L}, \quad C(m) = \sum_{k=0}^{K-1} B(m-k)$$
$$D(m) = C(m-1), \quad y'(n) = D(nK) \quad (8)$$

Hence, the output is

$$y'(n) = \sum_{k=0}^{K-1}\sum_{i=0}^{N+1} c_i u((n-i-1)K-1-k)2^{-<nK-1-k>_K L} + \sum_{k=0}^{K-1}\sum_{i=2}^{N+1} d_i v((n-i+1)K-1-k)2^{-<nK-1-k>_K L} \quad (9)$$

Substituting the value of $u(m)$ and $v(m)$ into (9), after a few steps of manipulation we have

$$y'(n) = \sum_{i=0}^{N+1} c_i \sum_{l=0}^{K-1} x_l(n-2-i)2^{-lL} + \sum_{i=2}^{N+1} d_i \sum_{l=0}^{K-1} y'_l(n-i)2^{-lL} = \sum_{i=0}^{N+1} c_i x(n-2-i) + \sum_{i=2}^{N+1} d_i y'(n-i) \quad (10)$$

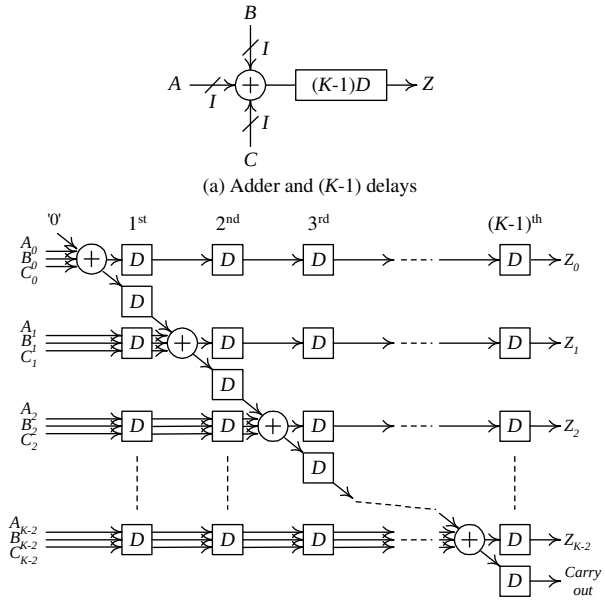
Comparing this with (5), we see that $y'(n) = y(n-2)$. Hence, Figure 3 realizes the IIR filter in (1) with the output delayed by two units. Note that $2N+2$ ROM's of 2^L words are needed for storing the partial results. Note also that accumulation of partial results of all coefficients is performed by only one summer of $H(z) = \sum_{i=0}^{K-1} z^{-i}$, resulting in hardware saving. The Scale-Sum-Downsample (SSD) unit in Figure 3 which consists of a scaling by $2^{-<m>_K L}$, a summer, a delay, and a downsampler can be realized by a simple structure, as shown in Figure 3.

The ROM size can be adjusted by changing the value of L, with $J = KL$. When L is large, the ROM size is larger, increasing the ROM accessing time and hardware. As L decreases, ROM size is smaller, but K will be larger, which implies that more values need to be accumulated in the SAD unit, reducing processing speed. Therefore, as K and L change we have various combinations of speed and hardware.

Although the proposed structure has many delay units, they are only L-bit wide. Comparing to the J-bit delay unit of conventional DA structure, K delays of L-bit wide require similar silicon area as one delay in Figure 1. The advantage of the proposed structure over DA in Figure 1 is that it requires less silicon area when L is not too large. In addition, the processing speed is higher than that of DA realization, since Figure 3 needs to accumulate K values to produce one output sample, while Figure 1 accumulates J values to produce each output sample ($J = KL$).

4. Pipelined Adder

Let us focus on the adders in Figure 3. We can see that the speed of the filter is limited by the speed of the adders. To eliminate this problem, we consider a realization of such an adder followed by $(K-1)$ delay units, as shown in Figure 4(a). The adder has to accumulate all I-bit input signals, then pass the summing result through $(K-1)$ delays. We can gain advantage in speed with the same amount of hardware by distributing the $(K-1)$ delays into the adder as in Figure 4(b). Here, A_j, B_j, and C_j, $j = 0$ to $K-2$, are the $K-1$ subwords of the input signals A, B, and C, with each subword being $\lceil I/(K-1)\rceil$-bit long, where $\lceil\gamma\rceil$ denotes the smallest integer $\geq \gamma$. Hence, a pipelined operation is obtained. Since the adder is pipelined into a smaller-sized adders, the propagation delay of the adder is reduced, resulting in higher processing speed. The gain in a higher processing speed accrues with a small increase in hardware.



(a) Adder and (K-1) delays

(b) Distribution of (K-1) delays to obtain pipelined adder.

Fig. 4. Structure for pipelined adder

The attempt to reduce the critical path delay by distributing $(K-1)$ delays into the adder consequently shifts the bottleneck problem to the SSD unit. To solve this problem, the carry-lookahead adder (CLA) is used to realize the adder in the SSD unit. As long as the CLA inside SSD unit does not have the propagation delay higher than that of $T_{ROM} + T_{pipelined\ adder}$, the SSD unit will not slow down the system. Morinaka *et al.* [5], for example, presented a type of 64-bit CLA

Table 1. Area and Speed Comparison Between DA and the Proposed Realization for the 5^{th} Order IIR Filter

	Area		Speed	
DA	A_{ROM}	$2^{2N} = 2^{10}$-word ROM	T_{ROM}	14 ns
Wordlength = 32 bits		10,000 gates	$T_{2-input\ adder(32\ bits)}$	24 ns
	$A_{2-input\ adder}$	250 gates	Critical path delay	38 ns
	A_{delay}	2,250 gates	Clock cycles required/output	**32**
	TOTAL size	**12,500 gates**	Output signal rate	**822 ksamples/s**
DA	A_{ROM}	2×2^{2N}-word ROM	T_{ROM}	14 ns
Wordlength = 16 bits		20,000 gates	$T_{3-input\ adder(32\ bits)}$	40 ns
(2 ROM's + larger adder)	$A_{3-input\ adder}$	500 gates	Critical path delay	54 ns
	A_{delay}	2,250 gates	Clock cycles required/output	**16**
	TOTAL size	**22,750 gates**	Output signal rate	**1.2 Msamples/s**
Proposed realization	$A_{Logic\ gates}$	$[2N+2] \times 100$	$T_{Logic\ gates}$	1 ns
With $\mathbf{K = 16, L = 2}$		1,200 gates	$T_{3-input\ adder(3\ bits)}$	3.75 ns
	$A_{3-input\ adder}$	3,250 gates		
	A_{delay}	3,650 gates	Critical path delay	4.75 ns
	$A_{SSD\ unit}$	3,000 gates	Clock cycles required/output	**16**
	TOTAL size	**11,100 gates**	Output signal rate	**13.2 Msamples/s**
Proposed realization	A_{ROM}	$[2N+2] \times 300$	T_{ROM}	7 ns
With $\mathbf{K = 8, L = 4}$		3,600 gates	$T_{3-input\ adder(5\ bits)}$	6.25 ns
	$A_{3-input\ adder}$	3,250 gates		
	A_{delay}	3,500 gates	Critical path delay	13.25 ns
	$A_{SSD\ unit}$	3,000 gates	Clock cycles required/output	**8**
	TOTAL size	**13,350 gates**	Output signal rate	**9.4 Msamples/s**
Proposed realization	A_{ROM}	$[2N+2] \times 2,500$	T_{ROM}	9 ns
With $\mathbf{K = 4, L = 8}$		30,000 gates	$T_{3-input\ adder(11\ bits)}$	13.75 ns
	$A_{3-input\ adder}$	3,250 gates		
	A_{delay}	3,200 gates	Critical path delay	22.75 ns
	$A_{SSD\ unit}$	3,000 gates	Clock cycles required/output	**4**
	TOTAL size	**39,450 gates**	Output signal rate	**11 Msamples/s**
Proposed realization	A_{ROM}	$[2N+2] \times 560,000$	T_{ROM}	26 ns
With $\mathbf{K = 2, L = 16}$		6,720,000 gates	$T_{3-input\ adder(32\ bits)}$	40 ns
	$A_{3-input\ adder}$	3,250 gates		
	A_{delay}	2,500 gates	Critical path delay	66 ns
	$A_{SSD\ unit}$	3,000 gates	Clock cycles required/output	**2**
	TOTAL size	**6,728,750 gates**	Output signal rate	**7.6 Msamples/s**

having only 2.6 ns delay time.

5. Comparison

As an example of comparing the DA realization with the proposed structure, we consider a 5^{th} order IIR filter with wordlength $J = 32$ bits. The area and speed of both realizations are shown in Table 1. We assume that carry-ripple adders are used in both realizations, and that the response time of a 1-bit 2-input and a 1-bit 3-input carry-ripple adders are 0.75 ns and 1.25 ns, respectively. With the CLA in [5], the critical path delay of the proposed realization is not in the SSD unit. Note that, for the case $L = 2$, ROM's are replaced with logic gates. In the table, we use the following notations:

A_ρ = Chip area for Component "ρ"
T_ρ = Response time for Component "ρ"

We can see that all four cases of the proposed realization are faster than the DA realization. However, as L increases, more hardware is required. The amount of hardware is more than that of the DA realization when L is too large.

The advantage of the proposed structure over the DA realization becomes greater as the filter order becomes large. An increased in filter order results in an exponential increase of the ROM size in Figure 1 and a linear increase in the number of ROM's in Figure 3. The processing speed of the DA realization is reduced by a larger ROM size, as the access time increases. However, the filter order does not affect the processing speed of the filter order since the ROM size remains unchanged.

6. Conclusion

We have presented a pipelined multiplier-free realization of IIR filters using ROM's and elevated signal rate. By proper selections of the parameters (K and L), the proposed realization gains advantage in both chip area and speed as compared to those of DA realization. When the filter order is larger, the advantage becomes greater.

References

[1] A. Peled and B. Liu, "A New Approach to the Realization of Nonrecursive Digital Filters," *IEEE Trans. on Audio and Electroacoustics,* vol.21, pp.477-485, Dec.1973.

[2] A. Peled and B. Liu, "A New Hardware Realization of Digital Filters," *IEEE Trans. on Acoustics, Speech, and Signal Proc.,* vol.22, pp.456-462, Dec. 1974.

[3] S.A. White, "Applications of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review," *IEEE ASSP Magazine,* vol. 6, pp.4-19, July 1989.

[4] T. Sakunkonchak and S. Tantaratana, "A Pipelined Multiplier-Free Realization of IIR Filter Using ROM's and Periodically Time-Varying Structure," In *Proc. IEEE ISPACS'99,* pp. 597-600, December 1999.

[5] H. Morinaka, H. Makino, Y. Nakase, H. Suzuki and K. Mashiko, "A 64bit Carry Look-ahead CMOS Adder using Modified Carry Select," In *Proc. IEEE 1995 Custom Integrated Circuits Conference,* pp. 585-588, 1995.