A Pipelined Multiplier-Free Realization of IIR Filter Using ROM's and Periodically Time-Varying Structure

Thanyapat Sakunkonchak and Sawasd Tantaratana
Department of Electrical Engineering, Sirindhorn International Institute of Technology
Thammasat University, Rangsit Campus, Patumtani, 12121, Thailand
Phone:(662)986-9009 ext. 3316, 1106, Fax:(662) 986-9112
E-mail: thong@siit.tu.ac.th, sawasd@siit.tu.ac.th

Abstract

We propose a pipelined multiplier-free realization which combines the ideas of distributed arithmetic (DA) and periodically time-varying (PTV) structure. The proposed realization uses ROM's to store the the results of coefficient scalings in combination with higher signal rate and pipelined operations. Hence, hardware multipliers are not needed. By varying some parameters, the proposed structure provides a tradeoff between hardware and clock speed (or throughput). An example of area-speed tradeoff between DA and the proposed realization is presented.

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 in a single direct step by table look-up [1]–[3]. 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.

Periodically time-varying (PTV) structures have been used to realize multiplier-free FIR filters [4], and IIR filters [5]. Bit-level architectures for VLSI implementation were presented.

Although DA provides a 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].

This paper presents a realization using table look-up in combination with higher signal rate and pipelined operation to obtain a multiplier-free realization. It combines the ideas of DA [3] and PTV structure [4], [5]. The DA realization is reviewed in Section II. In Section III, the proposed multiplier-free realization is presented. Section IV compares the area and speed tradeoff between DA and the proposed realization. Conclusion is given in Section V.

This work was supported by the National Electronics and Computer Technology Center (NECTEC), National Science and Technology Development Agency (NSTDA), Thailand, through grants #15/2540 and 07/2542.

2. Distributed Arithmetic Realization

Consider the IIR filter with input/output relation

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

Let the signals $x(n)$ and $y(n)$ have wordlength of J bits with two's complement representation, i.e.

$$\begin{aligned} x(n) &= -x_0(n) + \sum_{j=1}^{J-1} x_j(n) 2^{-j}, \\ y(n) &= -y_0(n) + \sum_{j=1}^{J-1} y_j(n) 2^{-j} \end{aligned} \tag{2}$$

where $x_j(n)$ and $y_j(n)$ are the j^{th} bits of $x(n)$ and $y(n)$, respectively, and $x_0(n)$ and $y_0(n)$ are the sign bits.

Substituting (2) into (1) and exchanging the order of summations yields

$$\begin{aligned} y(n) &= \sum_{j=1}^{J-1} \left[\sum_{i=0}^{N} a_i x_j(n-i) + \sum_{i=1}^{M} b_i y_j(n-i) \right] 2^{-j} \\ &\quad - \left[\sum_{i=0}^{N} a_i x_0(n-i) + \sum_{i=1}^{M} b_i y_0(n-i) \right] \\ &= \sum_{j=1}^{J-1} \mathbf{A} \mathbf{X}_\mathbf{j}^\mathbf{T} \mathbf{2}^{-\mathbf{j}} - \mathbf{A} \mathbf{X}_\mathbf{0}^\mathbf{T} \end{aligned} \tag{3}$$

where $\mathbf{A} = [a_0\ a_1 \dots a_N\ b_1 \dots b_M]$ and $\mathbf{X_j} = [x_j(n)\ x_j(n-1) \dots x_j(n-N)\ y_j(n-1) \dots y_j(n-M)]$. We can see from (3) 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^{N+M+1} possible values of vector $\mathbf{X_j}$ and stored in a ROM of 2^{N+M+1} words. A block diagram of the computation is depicted in Figure 1. The signal T_s dictated whether the partial result is added (for $j = 1, \dots, J-1$) or subtracted (for $j = 0$) from the accumulated result. The switch SWA is at position 1 for $J-1$ clock cycles and it is switched to position 2 for one clock cycle. Then, the result is dumped as the output $y(n)$.

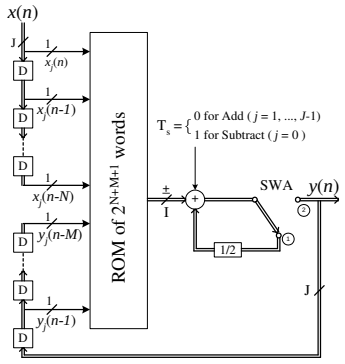

Fig. 1. A DA realization for an N^{th} order IIR filter, $N = 5$.

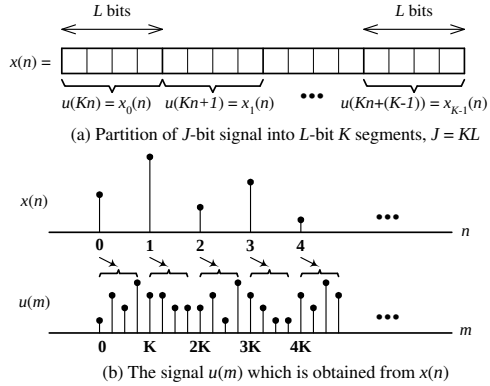

Fig. 2. Partition of $x(n)$ to yield $u(m)$

By some re-arrangement, the ROM size may be reduced by a factor of 2 to 2^{N+M}. Note that the ROM size grows exponentially as N or M increases. This drastic increase may be reduced by splitting the coefficient vector $\mathbf{A}$ and the signal vector $\mathbf{X_j}$ into two or more smaller vectors. For example, splitting each vector into two vectors, (3) becomes

$$y(n) = \sum_{j=1}^{J-1} \left[\tilde{\mathbf{A}}\tilde{\mathbf{X}}_\mathbf{j}^\mathbf{T} + \mathbf{B}\mathbf{Y}_\mathbf{j}^\mathbf{T}\right] 2^{-j} - \left[\tilde{\mathbf{A}}\tilde{\mathbf{X}}_\mathbf{0}^\mathbf{T} + \mathbf{B}\mathbf{Y}_\mathbf{0}^\mathbf{T}\right] \quad (4)$$

where $\tilde{\mathbf{A}} = [a_0\ a_1 \ldots a_N]$, $\mathbf{B} = [b_1 \ldots b_M]$, $\tilde{\mathbf{X}} = [x_j(n) \ldots x_j(n-N)]$, and $\mathbf{Y_j} = [y_j(n-1) \ldots y_j(n-M)]$. However, this will slow down the speed of the filter.

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. Pipelined Multiplier-Free Realization Using ROM's and Periodically Time-Varying Structure

In this section, we propose a new realization which uses ROM's in combination with higher signal rate and pipelined operations to obtain a multiplier-free realization. Consider a signal $x(n)$ of J bits, partitioned into K segments of L bits, with $J = KL$, as shown in Figure 2(a). As seen from the figure, we can write $x(n)$ and $ax(n)$ as

$$\begin{aligned} x(n) &= \sum_{k=0}^{K-1} u(Kn+k)2^{-kL}, \\ ax(n) &= \sum_{k=0}^{K-1} au(Kn+k)2^{-kL} \end{aligned} \quad (5)$$

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 accumulated and scaled by 2^{-kL}.

Based on the above idea, we can realize the IIR filter (1) as depicted in Figure 3. Note that the scaling by 2^{-kL} for all the coefficients are performed by a single unit inside the dotted box in Figure 3. Hence, we achieve hardware reduction. The scaled and accumulated of the partial results are delayed by one unit then, downsampled by a factor of K in order to retain the original signal rate. The hard-lines in Figure 3 denote the original signal rate where the normal-lines denote the K times higher signal rate.

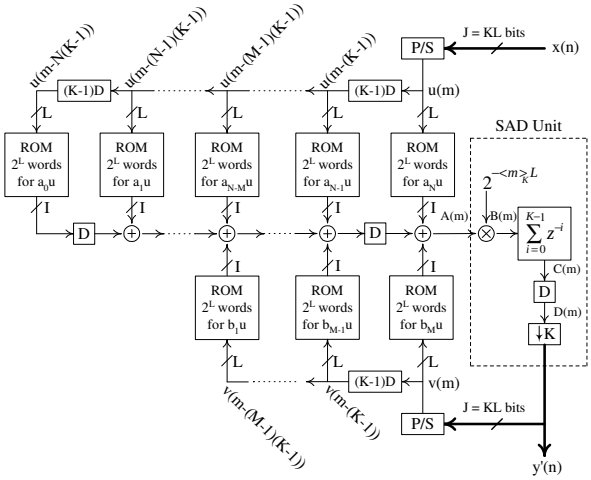

Fig. 3. The proposed realization of N^{th} order IIR filter

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 $x_i(\cdot)$ and $y'_i(\cdot)$ denote the i^{th} subword of $x(\cdot)$ and $y'(\cdot)$, respectively, each of which is L-bit wide, $\lfloor\gamma\rfloor$ denotes the integer part of γ, and $< m >_K$ is $m \mod K$. The output before scaling is

$$A(m) = \sum_{i=0}^{N} a_{N-i}u(m-iK) + \sum_{i=0}^{M-1} b_{M-i}v(m-iK) \quad (7)$$

Inside the dotted box, we have

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

Hence, the output is

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

Substituting the value of $u(m)$ and $v(m)$ into (9), with $p=(n-i)K-1-k$ and $q=nK-1-k$, we obtain

$$\begin{aligned} y'(n) &= \sum_{i=0}^{N} a_{N-i}\left[\sum_{k=0}^{K-1} x_{<p>_K}\left(\left\lfloor\frac{p}{K}\right\rfloor\right)2^{-<q>_K L}\right] \\ &\quad + \sum_{i=0}^{M-1} b_{M-i}\left[\sum_{k=0}^{K-1} y'_{<p>_K}\left(\left\lfloor\frac{p}{K}\right\rfloor\right)2^{-<q>_K L}\right] \\ &= \sum_{i=0}^{N} a_{N-i}\left[\sum_{k=0}^{K-1} x_{K-1-k}(n-i-1)2^{-(K-1-k)L}\right] \\ &\quad + \sum_{i=0}^{M-1} b_{M-i}\left[\sum_{k=0}^{K-1} y'_{K-1-k}(n-i-1)2^{-(K-1-k)L}\right] \\ &= \sum_{i=0}^{N} a_{N-i}\sum_{l=0}^{K-1} x_l(n-i-1)2^{-lL} \\ &\quad + \sum_{i=0}^{M-1} b_{M-i}\sum_{l=0}^{K-1} y'_l(n-i-1)2^{-lL} \\ &= \sum_{i=0}^{N} a_{N-i}x(n-1-i) + \sum_{i=0}^{M-1} b_{M-i}y'(n-1-i) \end{aligned} \quad (10)$$

Comparing this with (1), we see that $y'(n)=y(n-1)$. Hence, Figure 3 realizes the IIR filter (1) with a unit delay. Note that $N+M$ ROM's of 2^L words are needed for 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 dotted box 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 similar to SAD unit in [5].

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, we have a tradeoff between 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 ($J=KL$).

The proposed structure also gains advantage over DA realization when the filter order becomes large. As filter order increases, it results in exponentially increased the ROM size in Figure 1 and linearly increased the number of ROM's in Figure 3. The processing speed of the DA realization is reduced since increasing the ROM size increases the access time of the system. Although the number of ROM's for the proposed realization is increased, it does not effect the processing speed for every filter order since the ROM's size remain unchanged.

4. Comparison

As the examples of comparing the DA realization with the proposed structure realization, let us consider a 5^{th} and 10^{th} order IIR filter with wordlength $J=32$ bits. The area and speed tradeoff between both realizations are shown in Table I (for the 5^{th} order) and in Table II (for the 10^{th} order). Note that, for the case $L=2$, ROM's can be replaced by logic gates. We assume that carry-ripple adders are used in both realizations, that the access time of a 2^{20}-word ROM is 40 ns, and that the time response of a 32-bit 2-input and a 32-bit 3-input carry-ripple adders are 24 ns and 40 ns, respectively.

We see that as L increases we achieve faster processing, but the hardware increases. Comparison of Table I and II shows that the advantage of the proposed realization is more when the order is larger.

5. Conclusion

We have presented a pipelined multiplier-free realization of IIR filters using ROM's and PTV structure. By a proper selection of a pair of parameters (K and L) of the proposed realization, we can gain advantages in both chip area and speed as compared to that of DA realization. Furthermore, when the filter order is larger, the speed of the DA realization is reduced while the speed of the proposed structure realization remains unchanged.

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] S.P. Ghanekar, S. Tantaratana, and L.E. Franks, "A Class of High Precision Multiplier-Free FIR Filter Realizations Using PTV Coefficients," IEEE Trans. on Signal Processing, pp. 822-830, April 1995.

[5] S. Tantaratana, "Multiplier-Free IIR Filter Realizations with Periodically Time-Varying Coefficients," Journal of Circuits, Systems, and Computers, vol. 7, No. 4, 1997.

Table I

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	$T_{2-input\ adder}$	24 ns
	$A_{2-input\ adder}$	250 gates	latency	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	$T_{3-input\ adder}$	40 ns
(2 ROM's + larger adder)	$A_{3-input\ adder}$	500 gates	latency	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	$A_{Logic\ gates}$	$[2N+1] \times 100$	$T_{Logic\ gates}$	1 ns
With $\mathbf{K = 16, L = 2}$		1,100 gates	$T_{3-input\ adder}$	40 ns
	$A_{3-input\ adder}$	3,250 gates		
	A_{delay}	2,250 gates	latency	41 ns
	$A_{accumulator}$	1,000 gates	Clock cycles required/output	**16**
	TOTAL size	**7,600 gates**	Output signal rate	**1.5 Msamples/s**
Proposed realization	A_{ROM}	$[2N+1] \times 300$	T_{ROM}	7 ns
With $\mathbf{K = 8, L = 4}$		3,300 gates	$T_{3-input\ adder}$	40 ns
	$A_{3-input\ adder}$	3,250 gates		
	A_{delay}	2,250 gates	latency	47 ns
	$A_{accumulator}$	1,000 gates	Clock cycles required/output	**8**
	TOTAL size	**9,800 gates**	Output signal rate	**2.7 Msamples/s**
Proposed realization	A_{ROM}	$[2N+1] \times 2,500$	T_{ROM}	9 ns
With $\mathbf{K = 4, L = 8}$		27,500 gates	$T_{3-input\ adder}$	40 ns
	$A_{3-input\ adder}$	3,250 gates		
	A_{delay}	2,250 gates	latency	49 ns
	$A_{accumulator}$	1,000 gates	Clock cycles required/output	**4**
	TOTAL size	**34,000 gates**	Output signal rate	**5.1 Msamples/s**
Proposed realization	A_{ROM}	$[2N+1] \times 560,000$	T_{ROM}	26 ns
With $\mathbf{K = 2, L = 16}$		6,160,000 gates	$T_{3-input\ adder}$	40 ns
	$A_{3-input\ adder}$	3,250 gates		
	A_{delay}	2,250 gates	latency	66 ns
	$A_{accumulator}$	1,000 gates	Clock cycles required/output	**2**
	TOTAL size	**6,166,500 gates**	Output signal rate	**7.6 Msamples/s**

Table II

AREA AND SPEED COMPARISON BETWEEN DA AND THE PROPOSED REALIZATION FOR THE 10^{th} ORDER IIR FILTER

	Area		Speed	
DA	TOTAL size	A_{ROM}	T_{ROM}	40 ns
Wordlength = 32 bits		$2^{2N} = 2^{20}$-word ROM	$T_{2-input\ adder}$	24 ns
		10,000,000 gates	latency	64 ns
			Clock cycles required/output	**32**
			Output signal rate	**488 ksamples/s**
DA	A_{ROM}	2×2^{10}-word ROM	T_{ROM}	14 ns
Wordlength = 32 bits		20,000 gates	$T_{3-input\ adder}$	40 ns
(Split one large size ROM into	$A_{3-input\ adder}$	500 gates	$T_{2-input\ adder}$	24 ns
two smaller ROM's. Additional	$A_{2-input\ adder}$	250 gates	latency	78 ns
32-bit 2-input adder is needed	A_{delay}	2,250 gates	Clock cycles required/output	**32**
to sum the partial results)	TOTAL size	**23,000 gates**	Output signal rate	**400 ksamples/s**
DA	TOTAL size	$2 \times A_{ROM}$	T_{ROM}	40 ns
Wordlength = 16 bits		2×2^{20}-word ROM	$T_{3-input\ adder}$	40 ns
(2 ROM's + larger adder)		**20,000,000 gates**	latency	80 ns
			Clock cycles required/output	**16**
			Output signal rate	**781 ksamples/s**
Proposed realization	$A_{Logic\ gates}$	$[2N+1] \times 100$	$T_{Logic\ gates}$	1 ns
With $\mathbf{K = 16, L = 2}$		2,100 gates	$T_{3-input\ adder}$	40 ns
	$A_{3-input\ adder}$	6,500 gates		
	A_{delay}	4,500 gates	latency	41 ns
	$A_{accumulator}$	1,000 gates	Clock cycles required/output	**16**
	TOTAL size	**14,100 gates**	Output signal rate	**1.5 Msamples/s**
Proposed realization	A_{ROM}	$[2N+1] \times 300$	T_{ROM}	7 ns
With $\mathbf{K = 8, L = 4}$		6,300 gates	$T_{3-input\ adder}$	40 ns
	$A_{3-input\ adder}$	6,500 gates		
	A_{delay}	4,500 gates	latency	47 ns
	$A_{accumulator}$	1,000 gates	Clock cycles required/output	**8**
	TOTAL size	**18,300 gates**	Output signal rate	**2.7 Msamples/s**
Proposed realization	A_{ROM}	$[2N+1] \times 2,500$	T_{ROM}	9 ns
With $\mathbf{K = 4, L = 8}$		52,500 gates	$T_{3-input\ adder}$	40 ns
	$A_{3-input\ adder}$	6,500 gates		
	A_{delay}	4,500 gates	latency	49 ns
	$A_{accumulator}$	1,000 gates	Clock cycles required/output	**4**
	TOTAL size	**64,500 gates**	Output signal rate	**5.1 Msamples/s**
Proposed realization	TOTAL size	A_{ROM}	T_{ROM}	26 ns
With $\mathbf{K = 2, L = 16}$		$[2N+1] \times 560,000$	$T_{3-input\ adder}$	40 ns
		12,000,000 gates	latency	66 ns
			Clock cycles required/output	**2**
			Output signal rate	**7.6 Msamples/s**