SCALABLE IMPLEMENTATION SCHEME FOR MULTIRATE FIR FILTERS AND ITS APPLICATION IN EFFICIENT DESIGN OF SUBBAND FILTER BANKS

Liang-Gee Chen Po-Cheng Wu Tzi-Dar Chiueh

Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, R.O.C.

Abstract - A scalable implementation scheme for multirate FIR filters in consideration of both the processing time and the silicon area is presented in this paper. According to our various requirements, the flexible and efficient implementation scheme can simultaneously reduce both the time cost T to $\frac{1}{k}$T and the area cost A to $\frac{k}{M}$A (M is the decimation or interpolation rate, k is any factor of M). Furthermore, by employing the scalable implementation scheme, we also propose an efficient design technique for subband filter banks.

INTRODUCTION

Recently, there has been rapid progress in the area of multirate digital signal processing. The applications of multirate systems include subband coding of video, audio, and speech signals, fast transforms using digital filter banks, wavelet analysis of all types of signals, and many other fields [1]. In multirate systems, decimation and interpolation filters are the most important building blocks. A great amount of literature deals with the theory and design of decimation and interpolation filters [2], [3]. However, issues concerning the VLSI implementation scheme for multirate filters have not been investigated thoroughly. Since the speed of processing time and the silicon area are the crucial factors in the VLSI implementation, a scalable implementation scheme to flexibly and efficiently implement the multirate FIR filters is presented in this paper.

One of the most important applications of multirate systems is subband coding (SBC). Since it was introduced by Crochiere *et al.* [4] in 1976, subband coding is so far one of the most effective coding approaches for video and audio applications. Because subband coding needs filter banks to split the input signal (see Fig. 1), aside from the coding mechanism, the filter banks are the kernel of this coding scheme. In the past, various types of subband filter banks have been proposed, such as quadrature mirror filters (QMF)

0-7803-2612-1/95 $4.00

[4], conjugate quadrature filters (CQF) [5], and symmetric short kernel filters (SSKF) [6], etc. For subband coding systems, implementation of their filter banks is the most important task. Since filter banks usually deal with large amount of data, high speed computing hardware is indispensable. In order to achieve both high performance and low complexity, by employing the scalable implementation scheme, we propose an efficient design technique suitable for all types of subband filter banks.

The organization of this paper is as follows: Section 2 presents a scalable implementation scheme for multirate FIR filters in consideration of both the processing time and the silicon area. In Section 3, based on the scalable implementation scheme, an efficient design of subband filter banks is proposed. Finally, we state conclusions in Section 4.

SCALABLE IMPLEMENTATION SCHEME FOR MULTIRATE FIR FILTERS

When the decimation rate is M, the decimation filter can be implemented directly by a lowpass filter followed by a M-fold decimator. However, because the output samples of the lowpass filter are almost discarded by the decimator, the hardware utilization of such implementation is inefficient. Similarly, in the M-fold interpolation filter, the input samples of the lowpass filter are almost zero, thus direct implementation is also inefficient. Based on the polyphase decomposition, Bellanger *et al.* [2] presented an implementation scheme which reduces the processing time T to $\frac{1}{M}$T. On the other hand, Miyazaki *et al.* [7] presented another implementation scheme which reduces the silicon area A to $\frac{1}{M}$A. For example, let us consider the decimation-by-4 filter and assume that the filter has 8 taps: $a_0, a_1, a_2, \cdots, a_7$. Fig. 2 shows Bellanger's implementation scheme which reduces the time cost T to $\frac{1}{4}$T. Miyazaki's implementation scheme which reduces the area cost A to $\frac{1}{4}$A is shown in Fig. 3.

As mentioned in Noll's article [8], the generic throughput rate of a certain circuit obtained by proper optimization of the efficiency in a given technology will seldom fit the systems requirements. The problem is that typically the throughput rate of a DSP system is fixed (T = constant) and thus we want to optimize the silicon area (A) and the design effort. Only if we are able to find a strategy to match the throughput rates without changing the efficiency (equivalent to moving on a hyperbola with $A \cdot T$ = constant in an A-vs-T complexity diagram), then we can derive the optimal silicon area. Therefore, on the architectural level we have to search for so-called scalable solutions by e.g., allowing double speed at double effort (here silicon area) or half speed at half effort, and so on. Consequently, in order to obtain the scalable solutions, we present a scalable implementation scheme which can flexibly implement the multirate FIR filters by reducing both the time cost T to $\frac{1}{k}$T and the area

cost **A** to $\frac{\mathbf{k}}{\mathbf{M}}\mathbf{A}$ (M is the decimation or interpolation rate, **k** is any factor of M). The total reduction factor with respect to the AT product criterion is still $\frac{\mathbf{1}}{\mathbf{M}}$.

We employ two methods to develop the scalable implementation scheme. Method I: reducing the time cost **T** to $\frac{\mathbf{1}}{\mathbf{2}}\mathbf{T}$. This we represent by the symbol as shown in Fig. 4(a). The polyphase components aligned in the vertical direction indicate that each of the components possesses its own filter hardware. Method II: reducing the area cost **A** to $\frac{\mathbf{1}}{\mathbf{2}}\mathbf{A}$. This we represent by the symbol as shown in Fig. 4(b). The polyphase components aligned in the horizontal direction indicate that these components share the same filter hardware. Therefore, the operation of these components is time-interleaved. Note that the above two methods can be transformed by each other, i.e., the polyphase components can either possess their own hardware individually or share the hardware together.

First, let us consider the decimation filter. Suppose the decimation rate is 4, i.e., M = 4, and the lowpass filter $H(z)$ has 8 taps: $a_0, a_1, a_2, \cdots, a_7$. We decompose $H(z)$ into 4 polyphase components: $E_0(z)$, $E_1(z)$, $E_2(z)$, and $E_3(z)$ as follows:

$$H(z) = E_0(z^4) + z^{-1}E_1(z^4) + z^{-2}E_2(z^4) + z^{-3}E_3(z^4), \tag{1}$$

where

$$\begin{aligned} E_0(z) &= \sum_{n=-\infty}^{\infty} h(4n)z^{-n}, & E_1(z) &= \sum_{n=-\infty}^{\infty} h(4n+1)z^{-n}, \\ E_2(z) &= \sum_{n=-\infty}^{\infty} h(4n+2)z^{-n}, & E_3(z) &= \sum_{n=-\infty}^{\infty} h(4n+3)z^{-n}. \end{aligned} \tag{2}$$

Thus, $E_0(z)$ contains taps a_0 and a_4; $E_1(z)$ contains taps a_1 and a_5; $E_2(z)$ contains taps a_2 and a_6; $E_3(z)$ contains taps a_3 and a_7. By employing Method I and Method II individually, we obtain 8 different permutations, i.e., 8 different design types as shown in Fig. 5(a). The above 8 design types can also be transformed by each other as shown in Fig. 5(b). For example, in type (i), if we regard $E_0(z)$ and $E_1(z)$ together as a new component; $E_2(z)$ and $E_3(z)$ together as another new component, type (i) can be transformed into type (ii) by use of the Method I to Method II transformation. Subsequently, type (ii) can be further transformed into types (iii), (iv), and (v). Similarly, by use of the Method II to Method I transformation, type (v) can be transformed into type (vi), and type (vi) can be further transformed into types (vii), (viii), and (i).

Now, we want to select the design types which can efficiently utilize the hardware, i.e., operate continuously with no idle time. Because types (iii), (iv), (vii), and (viii) are irregular designs, they can not utilize the hardware efficiently. In type (vi), since the input data flow is illustrated as shown in Fig. 6(a) (the underline indicates that the filter hardware of the polyphase

component is in use, and the × indicates that the filter hardware of the polyphase component is idle), therefore, the hardware cannot operate continuously. In contrast, the input data flow of type (ii) is illustrated as shown in Fig. 6(b). Here, we find the hardware operation can be kept continuous. Thus, type (ii) is an efficient design. Fig. 7 shows its corresponding hardware architecture. Type (i) is the Bellanger's implementation scheme as shown in Fig. 2. Type (v) is the Miyazaki's implementation scheme as shown in Fig. 3. The input data flows of these two design types are shown in Figs. 6(c) and (d). Both of them can utilize the hardware efficiently.

Consequently, for the decimation-by-4 filter, the total design types which can utilize the hardware efficiently are types (i), (ii), and (v). After checking carefully, we can derive the following relation equation (3) for M = 4.

$$\begin{aligned} \mathrm{M} &= 4 \\ &= 1 \times 4 \qquad \text{type (i)} \\ &= 2 \times 2 \qquad \text{type (ii)} \\ &= 4 \times 1 \qquad \text{type (v).} \end{aligned} \tag{3}$$

The multiplicands in the above equation indicate that how many polyphase components share the same filter hardware, and the multipliers indicate the total number of filter hardware.

When M = 8, we decompose the decimation-by-8 filter into 8 polyphase components. Similarly, we apply the same steps as in M = 4 and list all the design types according to the different permutations of its eight polyphase components. After careful examination and elimination of design types which cannot keep the hardware continuously in operation, we finally obtain 4 design types which can utilize the hardware efficiently. The relation equation for M = 8 is shown in Eq. (4).

$$\mathrm{M} = \mathbf{8} = 1 \times 8 = 2 \times 4 = 4 \times 2 = 8 \times 1. \tag{4}$$

For the same reasons, when M = 9, there are 3 different design types which can utilize the hardware efficiently. Their relation equation is shown in Eq. (5).

$$\mathrm{M} = \mathbf{9} = 1 \times 9 = 3 \times 3 = 9 \times 1. \tag{5}$$

When M = 12, there are 6 different efficient design types. Their relation equation is shown in Eq. (6).

$$\mathrm{M} = \mathbf{12} = 1 \times 12 = 2 \times 6 = 3 \times 4 = 4 \times 3 = 6 \times 2 = 12 \times 1. \tag{6}$$

From above discussions, we can derive the principle of the scalable implementation scheme by considering both the processing time and the silicon area for the decimation-by-M filter. The scalable implementation scheme is:

When the decimation rate M *has* **n** *different factors,*
there are **n** *different design types which can utilize the hardware efficiently.*

Here we compare the numbers of multipliers, adders, and registers required in different design types for the decimation-by-8 filter. Suppose the decimation filter has 16 taps: $a_0, a_1, a_2, \cdots, a_{15}$. According to Eq. (4), there are four different design types that can utilize the hardware efficiently: (i) Reducing the time cost **T** to $\frac{1}{8}$**T**. (ii) Reducing the time cost **T** to $\frac{1}{4}$**T** and the area cost **A** to $\frac{1}{2}$**A**. (iii) Reducing the time cost **T** to $\frac{1}{2}$**T** and the area cost **A** to $\frac{1}{4}$**A**. (iv) Reducing the area cost **A** to $\frac{1}{8}$**A**. The comparison result is shown in Table 1.

Finally, the required numbers of multipliers, adders, and registers for the scalable implementation scheme can be summarized as follows:

$$\#\text{ of multipliers} = Nk/M, \tag{7}$$

$$\#\text{ of adders} = \begin{cases} N-1, & \text{for } k=M, \\ Nk/M+k-1, & \text{otherwise,} \end{cases} \tag{8}$$

$$\#\text{ of registers} = \begin{cases} N-k, & \text{for } k=M, \\ Nk/M, & \text{otherwise,} \end{cases} \tag{9}$$

where M is the decimation rate, N is the number of filter taps, and k indicates the different design types.

Similarly, using the same discussions and deductions, we can obtain the same scalable implementation scheme for the interpolation filters.

EFFICIENT DESIGN OF SUBBAND FILTER BANKS

First, let us consider the analysis filter banks. In subband coding, the 4-band analysis filter bank is the basic structure used to split the input signals into 4 subbands. Usually, the 4-band analysis filter bank is tree-structured and contains two stages as shown in Fig. 1.

To design the 4-band analysis filter bank efficiently, we assume "**a**" is the area cost and "**t**" is the time cost required in stage 1. According to the original design as shown in Fig. 1, the silicon area required in stage 2 is double that in stage 1. That is, **2a** is the area cost required in stage 2. On the other hand, since there is the decimation operation in stage 1, the quantity of data for filtering in each branch of stage 2 is half that of stage 1. Therefore, the processing time required in stage 2 is half of **t**, i.e., $\frac{1}{2}$**t**. From the above discussion, we find that because stage 2 is cascaded after stage 1, stage 2 can't work until stage 1 finishes its job. Therefore, there will be $\mathbf{2a} \times (\mathbf{t} - \frac{1}{2}\mathbf{t}) = \mathbf{at}$ hardware idle in stage 2. In other words, the hardware utilization in the original design for the 4-band analysis filter bank is inefficient.

In order to utilize the hardware efficiently, we employ two methods, Method I: reducing the time cost **T** to $\frac{1}{2}$**T**, and Method II: reducing the area cost **A** to $\frac{1}{2}$**A**, as described in the previous section, for stage 1 and stage 2 respectively. Therefore, there are four different new design techniques generated for the 4-

band analysis filter bank. The performance comparison of the different design techniques (including the original design) is shown in Table 2.

From Table 2, we find if we apply Method I, reducing the time cost **T** to $\frac{1}{2}$**T**, to stage 1, and Method II, reducing the area cost **A** to $\frac{1}{2}$**A**, to stage 2, the area cost and the time cost are both the same **a** and $\frac{1}{2}$**t** in stage 1 and stage 2. Consequently, the total area cost is **2a** and the total time cost is $\frac{1}{2}$t. The *AT* product is **at** and no hardware is idle in stage 2. That is, the performance of the new design technique is three times more efficient than the original design for the 4-band analysis filter bank. In contrast, other design techniques, as listed in Table 2, will make some hardware idle in stage 2. Thus, they are inefficient designs.

Similarly, using the same discussions and deductions, we can obtain the same efficient design for the synthesis filter banks.

CONCLUSIONS

In the field of video filters, linear scalable solutions for time-sharing become increasingly more important due to the progress of VLSI technologies. Therefore, in this paper, we present a scalable implementation scheme for multirate FIR filters in consideration of both the processing time and the silicon area. According to our various requirements, the scalable implementation scheme can simultaneously reduce both the time cost T to $\frac{1}{k}$**T** and the area cost **A** to $\frac{k}{M}$**A** (M is the decimation or interpolation rate, **k** is any factor of M). We also compute the numbers of multipliers, adders, and registers required in this scheme. After that, by employing the scalable implementation scheme, we propose an efficient design technique suitable for all types of subband filter banks. The advantage of the proposed design is that the hardware utilization of the filter banks is very efficient. Consequently, it helps to develop a high performance subband coding system.

References

[1] P. P. Vaidyanathan, *Multirate Systems and Filter Banks.* Englewood Cliffs, NJ: Prentice Hall, 1993.

[2] M. G. Bellanger, G. Bonnerot, and M. Coudreuse, "Digital filtering by polyphase network: application to sample rate alteration and filter banks," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. 24, pp. 109–114, Apr. 1976.

[3] T. Saramaki, "A class of linear-phase FIR filters for decimation, interpolation, and narrow-band filtering," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. 32, pp. 1023–1036, Oct. 1984.

[4] R. E. Crochiere, S. A. Webber, and J. L. Flanagan, "Digital coding of speech in subbands," *Bell Systems Technical Journal,* vol. 55, pp. 1069–1085, Oct. 1976.

[5] M. J. T. Smith and T. P. Barnwell III, "Exact reconstruction techniques for tree-structured subband coders," *IEEE Trans. Acoust., Speech, Signal Processing,* pp. 434–441, 1986.

[6] D. LeGall and A. Tabatabai, "Subband coding of digital image using symmetric short kernel filters and arithmetic coding techniques," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* pp. 761–764, New York, NY, Apr. 1988.

[7] Y. Miyazaki, T. Nishitani, M. Ishikawa, M. Edahiro, and K. Mitsuhashi, "Chrominance/luminance signal separation and synthesis chips developed with a DSP silicon compiler," *IEEE Trans. Circuits Syst. for Video Tech.,* vol. 2, no. 2, pp. 245–254, June 1992.

[8] T. G. Noll, "High throughput digital filters," in *VLSI Implementations for Image Communications,* P. Pirsch (Editor). Elsevier Science Publishers, pp. 171–215, 1993.

M = 8, N = 16 (taps) T ⇒ T/k, A ⇒ Ak/M		Multipliers		Adders		Registers		Proportion
(1) k = 8	T/8, A	N	(16)	N-k+k-1	(15)	N-k	(8)	N
(2) k = 4	T/4, A/2	N/2	(8)	N/2+k-1	(11)	N/2	(8)	N/2
(3) k = 2	T/2, A/4	N/4	(4)	N/4+k-1	(5)	N/4	(4)	N/4
(4) k = 1	T, A/8	N/8	(2)	N/8+k-1	(2)	N/8	(2)	N/8

Table 1: Comparison of the numbers of multipliers, adders, and registers required in four different efficient design types for the decimation-by-8 filter.

Methods		Stage 1		Stage 2		Total	Total	AT	Stage 2
Stage 1	Stage 2	Area	Time	Area	Time	Area	Time	Prod.	Idle
Original Design		a	t	2a	t/2	3a	t	3at	at
(II) A/2	(II) A/2	a/2	t	a	t/2	3a/2	t	3at/2	at/2
(I) T/2	(I) T/2	a	t/2	2a	t/4	3a	t/2	3at/2	at/2
(II) A/2	(I) T/2	a/2	t	2a	t/4	5a/2	t	5at/2	3at/2
(I) T/2	(II) A/2	a	t/2	a	t/2	2a	t/2	at	0

Table 2: Performance comparison of five different design techniques for the 4-band analysis filter bank.

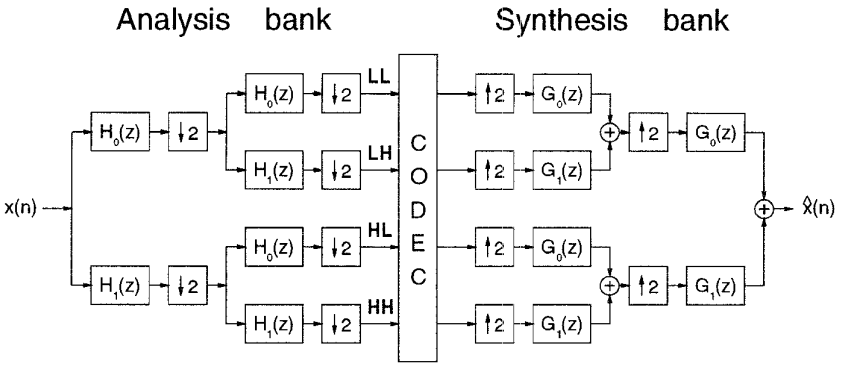


Figure 1: The 4-band subband analysis/synthesis filter bank.

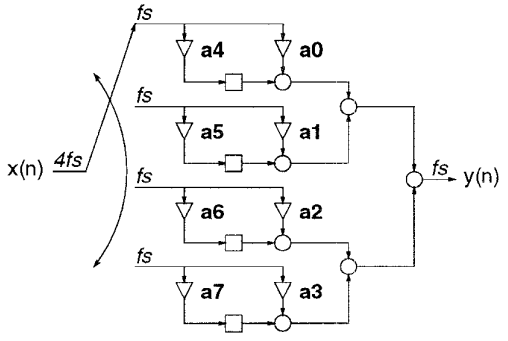


Figure 2: Hardware architecture of the decimation-by-4 filter for reducing the time cost $\mathbf{T}$ to $\frac{1}{4}\mathbf{T}$.

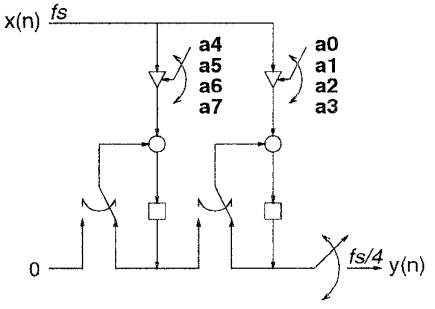
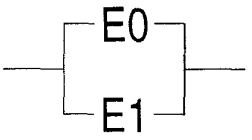
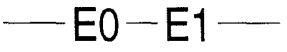


Figure 3: Hardware architecture of the decimation-by-4 filter for reducing the area cost $\mathbf{A}$ to $\frac{1}{4}\mathbf{A}$.

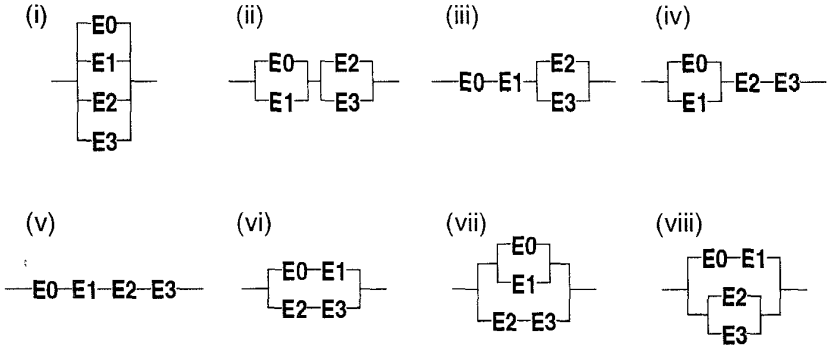


(a) Method I.

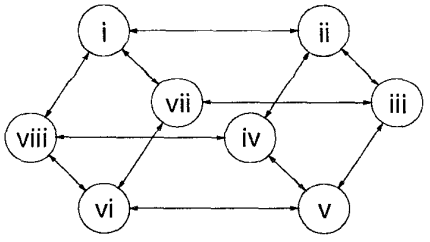


(b) Method II.

Figure 4: (a) Method I: the symbol used to represent the reduction in time cost from **T** to $\frac{1}{2}$**T**. (b) Method II: the symbol used to represent the reduction in area cost from **A** to $\frac{1}{2}$**A**.



(a)



(b)

Figure 5: (a) The 8 different design types for the decimation-by-4 filter. (b) The transformation of 8 different design types for the decimation-by-4 filter.

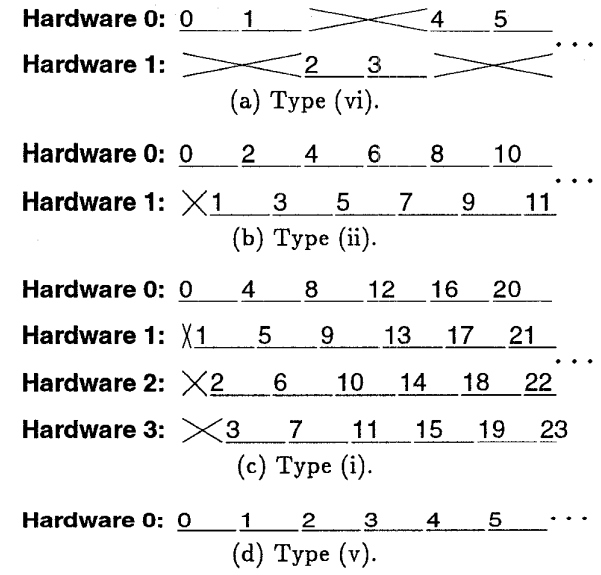


Figure 6: Illustrations of the input data flows for the decimation-by-4 filter in different design types: (a) Type (vi), (b) Type (ii), (c) Type (i), and (d) Type (v).

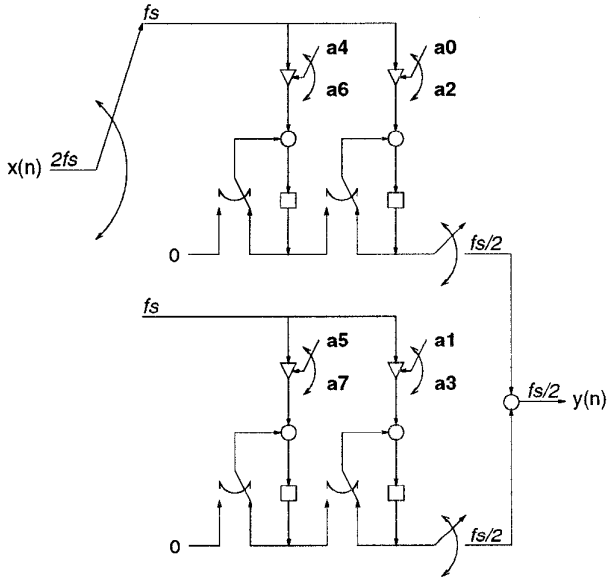


Figure 7: Hardware architecture of the decimation-by-4 filter for reducing the time cost $\mathbf{T}$ to $\frac{1}{2}\mathbf{T}$ and the area cost $\mathbf{A}$ to $\frac{1}{2}\mathbf{A}$.