Received January 1, 2018, accepted February 4, 2018, date of publication February 19, 2018, date of current version March 15, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2807406

A Novel Data Reuse Method to Reduce Demand on Memory Bandwidth and Power Consumption For True Motion Estimation

WEIZHI XU[1], HUI YU[2], DIANJIE LU[1], FANGAI LIU[1], AND ZHIYONG LIU[3]

[1]School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China
[2]School of Management Science and Engineering, Shandong Normal University, Jinan 250014, China
[3]State Key Laboratory for Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China

(Corresponding author: Hui Yu (huiyu0117@sdnu.edu.cn))

This work was supported in part by NSFC project under Grant 61520106005, Grant 61602285, Grant 61772321, Grant 61602284, Grant 61402268, and Grant 61402270, in part by the National Key Research and Development Program of China project under Grant 2017YFB1010000, and in part by the Shandong Provincial Natural Science Foundation, China, under Grant ZR2015FQ009, Grant ZR2016FP07, and Grant ZR2014FQ009.

ABSTRACT Motion estimation (ME) is a kernel algorithm in many video applications. Full search integer ME (FSIME) can find the best result but it usually takes plenty of time. Traditionally, only intra-frame data reuse is considered for FSIME. In this paper, a new inter-frame data reuse method is proposed to further utilize inter-frame data reuse for true ME. ME in frame rate up-conversion (FRUC-ME), a kind of true ME, is used as a case study. For FRUC-ME with the new inter-frame data reuse method, a frame is loaded to the on-chip buffer only once instead of twice and used for two interpolated frames. Two levels of the new method are proposed, Inter-D and new Inter-E, which give a good tradeoff between off-chip memory bandwidth and on-chip buffer size. New data access order is used to implement the new data reuse method. The proposed data reuse method (Inter-D) demands less off-chip memory bandwidth than its intra-frame counterpart (Intra-D), and the off-chip memory traffic and power consumption are both reduced by 37.5% for FRUC-ME.

INDEX TERMS Motion estimation, full search, inter-frame, data reuse, frame rate up-conversion.

I. INTRODUCTION

Motion estimation (ME) is a kernel algorithm for many video applications. It can be used for video encoder to reduce temporal redundancy between frames [1]–[4]. It is also used to track the projected object motion and produce the interpolated frame for frame rate up-conversion (FRUC-ME) [5]–[7]. FRUC-ME is usually considered as a kind of true motion estimation (Fig. 1). ME can also be used to derive depth information of a scene for 3-D reconstruction [8], [9].

Block-based ME is usually used in video applications because it is simple and regular. Block matching is adopted to search for the most matching macro-block (MB) in another frame. The accuracy of ME is important. For example, it can affect the quality of the interpolated frame in FRUC. Full search integer ME (FSIME) search all MBs in the assigned area and gives the best MB but it needs too much off-chip memory traffic. On the other hand, FSIME is suitable to be implemented in hardware because its computation and memory access are regular. Fast search methods are proposed to reduce the search points of FSIME to cut down the time overhead [10]–[14]. However, they do not ensure that the optimal results are found. Furthermore, computations and memory accesses of fast search are not regular. This paper tries to reduce the demand on memory bandwidth and power consumption of FSIME.

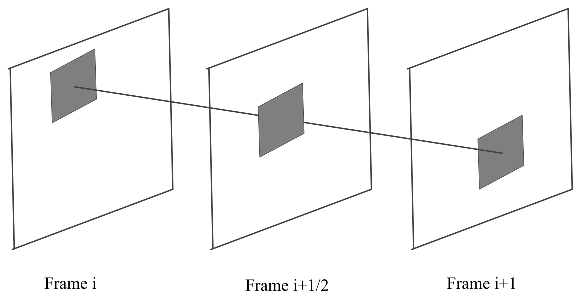


FIGURE 1. Motion estimation in frame rate up-conversion.

The off-chip memory access speed cannot catch up with the computing speed in recent years, so reducing off-chip memory traffic of FSIME is very important especially for real-time video applications [15], [16]. Some data reuse methods are proposed for FSIME to reduce off-chip memory traffic [17]–[20]. However, they mainly exploited intra-frame data reuse within previous frame while inter-frame data reuse between different frames was not considered. For FRUC-ME only with intra-frame data reuse, each frame has to be loaded from off-chip memory to on-chip memory twice, the first time as current frame, the second time as previous frame [21]. This consumes not only a lot of memory bandwidth but also plenty of energy, which is an especially important factor for mobile devices and data centers.

On the other hand, different processors (CPU, GPU, FPGA or ASIC) can offer very different on-chip buffer sizes, from several KB to several MB. A proper data reuse method can exploit the best of available cache size. Take data in Table 3b as an example. Intra-D and Inter-E are two traditional data reuse methods [17], and Inter-D is a data reuse method proposed in this paper. Intra-D data reuse needs at least a 60KB on-chip buffer which can reduce the off-chip memory bandwidth requirement to 124 MByte/sec, and Inter-E needs at least a 4MB on-chip memory which can reduce the off-chip memory bandwidth requirement to 62 MByte/sec (Table 3b). If the on-chip memory size of available platform is 256KB, we can only use Intra-D which still demands a lot of off-chip memory bandwidth and the cache size is not sufficiently used. If there is a data reuse method between Intra-D and Inter-E, e.g. 241KB on-chip memory with 77 MByte/sec off-chip memory bandwidth (Inter-D), it will be the best data reuse method for this cache size.

In this paper, a new method is proposed to exploit inter-frame data reuse for FRUC-ME, a kind of true ME. For FRUC-ME with the new method, most of the frames are processed as current frame and previous frame at the same time in order to be loaded into on-chip memory only once. Two levels (new Inter-E and Inter-D) of the proposed method are given for a good tradeoff between on-chip memory size and off-chip memory traffic. The inter-frame data reuse method (Inter-D) requires less off-chip memory traffic than its intra-frame counterpart (Intra-D). On the other hand, the proposed method is compatible with intra-frame data reuse methods [17], [18] and data access orders [19], [20]. We carefully design the buffer and memory access order for the implementation of the new data reuse methods and the reduction of the buffer size. We give a new definition of *Ra* which evaluates the degree of redundant access to off-chip memory and analyze different data reuse methods. Experiment result shows that the new technique can reduce both the memory traffic and the power consumption effectively (by about 37.5%) in comparison with the existing intra-frame reuse method.

We organize the rest of the paper as follows. We analyze the data locality of FSIME in Section II. The new inter-frame data reuse method for FRUC-ME is presented in Section III. Case studies are shown in Section IV. Section V is the conclusion.

II. DATA LOCALITY ANALYSIS FOR FSIME

In this section, we first explain some concepts (Fig. 2). MB, BS, SR, SRS and frame are five levels of data regions. Macro Block (MB) is a pixel block with size of $N \times N$. Block Strip (BS) is a row of MBs within the search range of the frame, with the size of $N \times SRH$. Search Range (SR) is the search range in the previous frame for the current block, with the size of $SRV \times SRH$. SR Strip (SRS) is a row of search ranges in the frame with the size of $SRV \times W$. The size of a frame is $W \times H$. Current Block (CB) is the macro-block in process within the current frame. A scan order is used for the according data reuse method. Different scan orders (raster scan, snake scan, smart snake scan and so on) will lead to different off-chip memory traffic [19], [20]. Processing Element Array (PEA) is the computing unit to get the motion vector.

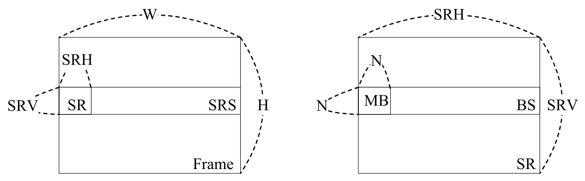


FIGURE 2. Basic concepts.

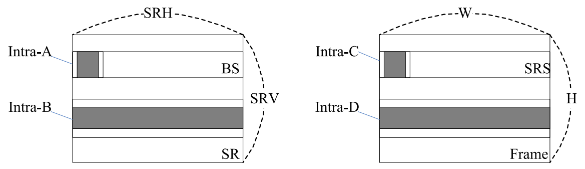


FIGURE 3. Intra-frame data reuse methods.

Four intra-frame data reuse levels are given in Fig. 3 [17]. Intra-A is within BS in an SR. Intra-B is between adjacent BS in an SR. Intra-C is within an SRS. Intra-D is between adjacent SRS. Intra-A+ [19] is between Intra-A and Intra-B, and Intra-C+ [18] is between Intra-C and Intra-D. Intra-A+ is similar to Intra-C+ but they apply to different data ranges and need different memory sizes. These data reuse levels only utilize the data reuse within previous frame. However, they still demands too much off-chip memory traffic. Each frame needs to be loaded into on-chip memory twice, one time as current frame and the other time as previous frame for FRUC-ME.

III. INTER-FRAME DATA REUSE METHOD FOR FRUC-ME

Each frame has to be loaded into on-chip memory twice for FRUC-ME only with intra-frame data reuse. In order to reduce this kind of overhead, we propose an inter-frame data reuse method to load frames into on-chip memory only once for two typical kinds of FRUC-ME, unidirectional FRUC-ME (FRUC-UME) and bidirectional FRUC-ME (FRUC-BME) [5]. FRUC-UME only needs to search/load one SR for one interpolated MB, while FRUC-BME needs two SR in two adjacent frames. Two inter-frame data reuse levels (new Inter-E and Inter-D) are developed to achieve better tradeoff between off-chip memory traffic and on-chip memory size.

A. NEW RA (REDUNDANT ACCESS FACTOR) FOR FRUC-ME

Previous work did not consider the potential for data reuse between different frames, so inter-frame data redundancy existed when computing *Ra*. The off-chip memory traffic for previous frame was included, but the off-chip memory traffic for current frame was neglected. So we give a new definition of *Ra* to compare different data reuse levels. In (1), new *Ra* includes both redundant access to previous frame (*Ra_pre*) and redundant access to current frame (*Ra_cur*). *Ra_pre* describes the data redundancy of loading pevious frame to on-chip memory in (2). *Mem_pre* is the off-chip memory traffic to load previous frame. *Ra_cur* rises from off-chip memory traffic to load current frame. *Ra_cur* is defined in (3), where *Mem_cur* equals memory traffic to load one current frame. A larger Ra stands for higher memory bandwidth requirement.

$$Ra = Ra\_pre + Ra\_cur. \tag{1}$$

$$Ra\_pre = \frac{Mem\_pre}{Pixel\ count\ in\ one\ frame}. \tag{2}$$

$$Ra\_cur = \frac{Mem\_cur}{Pixel\ count\ in\ one\ frame}. \tag{3}$$

For FRUC-UME, *Ra* of intra-frame data reuse levels can be computed by (1) (Table 1). *Ra_cur* always equals 1 for FRUC-UME because each current frame is loaded only once. As an example, *Ra* of Intra-C is computed as in (4).

$$\begin{aligned} Ra &= Ra\_pre + Ra\_cur \\ &= (1 + SRV/N) + (W \times H)/(W \times H) \\ &= (1 + SRV/N) + 1. \end{aligned} \tag{4}$$

TABLE 1. Redundant access (*Ra*) and On-chip Buffer Size for FRUC-UME.

Level	Ra	On-chip buffer size
No reuse	SRV×SRH+1	0
Intra-A [17]	SRV(1+SRH/N)+1	N(N-1)
Intra-B [17]	(1+SRV/N)(1+SRH/N)+1	(N+SRH)(N-1)
Intra-C [17]	(1+SRH/N)+1	(SRH+N-1)(SRV+N-1)
Intra-C+ [18]	(1+SRH/nN)+1	(SRH+N-1)(SRV+nN-1)
Intra-D [17]	1+1	(SRH+W-1)(SRV-1)
Inter-D	1+1/m	m(SRH+W-1)(SRV-1)
Inter-E [17]	1	2WH
New Inter-E	1	WH+2NW

Ra of Intra-D is computed as in (5).

$$\begin{aligned} Ra &= Ra\_pre + Ra\_cur \\ &= 1 + (W \times H)/(W \times H) \\ &= 1 + 1 \end{aligned} \tag{5}$$

According to (1), *Ra* of intra-frame data reuse levels for FRUC-BME can also be computed (Table 2). For example,

TABLE 2. Redundant access (*Ra*) and On-chip buffer size for FRUC-BME.

Level	Ra	On-chip buffer size
No reuse	2×SRV×SRH	0
Intra-A [17]	2SRV(1+SRH/N)	2N(N-1)
Intra-B [17]	2(1+SRV/N)(1+SRH/N)	2(N+SRH)(N-1)
Intra-C [17]	2(1+SRH/N)	2(SRH+N-1)(SRV+N-1)
Intra-C+ [18]	2(1+SRH/nN)	2(SRH+N-1)(SRV+nN-1)
Intra-D [17]	1+1	2(SRH+W-1)(SRV-1)
Inter-D	1+1/m	(m+1)(SRH+W-1)(SRV-1)
Inter-E [17]	1	2WH
New Inter-E	1	WH+2NW

Intra-C *Ra* of FRUC-BME is $2(1 + SRV/N)$ as in (6) because SRs of both the previous frame and the current frame are loaded for an interpolated frame.

$$\begin{aligned} Ra &= Ra\_pre + Ra\_cur \\ &= (1 + SRV/N) + (1 + SRV/N) \\ &= 2(1 + SRV/N). \end{aligned} \tag{6}$$

B. INTER-E DATA REUSE METHOD FOR FRUC-ME

A straight-forward method for the implementation of inter-frame data reuse is to put whole frames on chip. Two frame buffers and one PEA are adopted to implement Inter-E (Fig. 4) for FRUC-UME or FRUC-BME. At first, Frame 0 and 1 are loaded into on-chip Frame Buffer 0 and 1 respectively. Frame 0 and Frame 1 are the previous frame and the current frame respectively. After Frame 0 and 1 are processed, Frame 2 is loaded into Frame Buffer 0 as current frame and Frame 1 is considered as the previous frame. This process is repeated so that each frame is loaded into the on-chip buffer only once (Fig. 5).

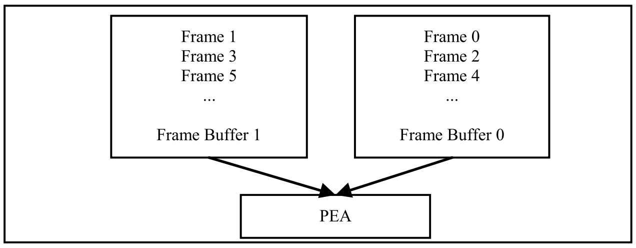
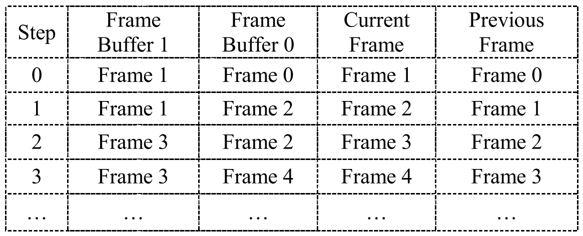


FIGURE 4. Inter-E implementation for FRUC-UME or FRUC-BME.



Step	Frame Buffer 1	Frame Buffer 0	Current Frame	Previous Frame
0	Frame 1	Frame 0	Frame 1	Frame 0
1	Frame 1	Frame 2	Frame 2	Frame 1
2	Frame 3	Frame 2	Frame 3	Frame 2
3	Frame 3	Frame 4	Frame 4	Frame 3
...	...	...	...	...

FIGURE 5. PEA processing order for Inter-E of FRUC-ME. The frames stored in the two frame buffers are listed for each step.

Parallelism can be increased by using more frame buffers and PEAs. In Fig. 6, two PEAs compute in parallel for two current frames and three frame buffers are needed. Note that,

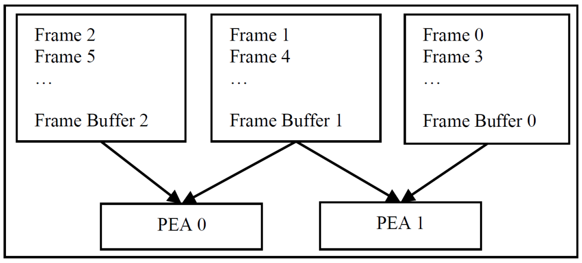


FIGURE 6. FRUC-ME architecture for Inter-E with two PEAs.

we use PEA to implement the new data reuse method in this paper and PEA is popular for processing ME but PEA is not essential to implement the new data reuse scheme. For example, an architecture with programmable on-chip memory (like GPU) can also be used to implement the data reuse scheme.

Two frame buffers are needed to implement Inter-E in Fig. 4. The frame buffer size can be reduced by reusing the frame buffer. In Fig. 7, we partition the frame buffer to *H/N* + *2* BS buffers. The frame buffer is designed as a circular buffer to hold two frames at the same time for FRUC-UME. One BS buffer contains one BS and *H/N* BS buffers can hold a whole frame. Two more BS buffers are used to store the non-overlapped part of the two adjacent frames. It is assumed that *SRV* equals *2N* but the method is still effective by changing the number of the BS buffers or the BS buffer size when *SRV* does not equal *2N*. Frame *i* and *i* + *1* are the previous frame and the current frame respectively. When PEA is processing Frame *i* + *1* BS2, Frame *i* (BS1, BS2 and BS3) are combined to be an SRS. At this time, Frame *i* BS0 will not be used so it is replaced by Frame *i* + *1* BS2 in the BS buffer (Fig. 8). Similarly, Frame *i* BS1 and Frame *i* + *1* BS3 share one BS buffer. Therefore, one BS buffer is reused by a previous frame and a current frame. After adopting the circular buffer, the frame buffer size is cut down nearly by half, from *2WH* to *WH* + *2NW*. This memory size reduction method can be applied to not only on-chip memory but also off-chip memory. Inter-E with circular buffer is called new Inter-E in this paper.

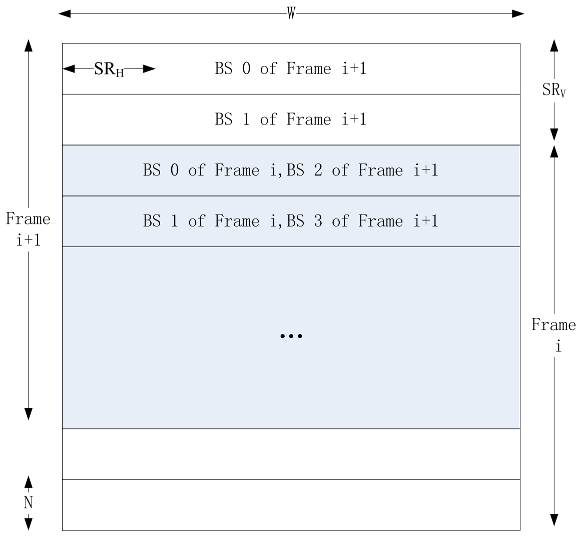

W
SR_H
BS 0 of Frame i+1
BS 1 of Frame i+1
BS 0 of Frame i,BS 2 of Frame i+1
BS 1 of Frame i,BS 3 of Frame i+1
…
SR_V
Frame i+1
Frame i
N

FIGURE 7. Circular frame buffer to implement Inter-E for FRUC-ME. The buffers in grey are shared between Frame *i* and Frame *i* + *1*.

Step	Current BS in process	Reference BS	Replaced BS in BS buffer
0	BS0 of Frame i+1	BS0,1 of Frame i	
1	BS1 of Frame i+1	BS0,1,2 of Frame i	
2	BS2 of Frame i+1	BS1,2,3 of Frame i	BS0 of Frame i
3	BS3 of Frame i+1	BS2,3,4 of Frame i	BS1 of Frame i
4	BS4 of Frame i+1	BS3,4,5 of Frame i	BS2 of Frame i
…	…	…	…

FIGURE 8. The BS replacement strategy and the BS processing order of the circular buffer for FRUC-UME.

The circular frame buffer size is also *WH* + *2NW* as in Fig. 7 for FRUC-BME but the two adjacent frames are processed in an interleaved way (Fig. 9). Each frame is loaded only once for Inter-E or new Inter-E so *Ra* is 1 according to (2). However, new Inter-E still demands a large on-chip memory (more than one frame).

Step	Current BS in process	Reference BS	Replaced BS in BS buffer
0	BS0 of Frame i+1	BS0,1 of Frame i	
1	BS1 of Frame i+1	BS0,1,2 of Frame i	
2	BS0 of Frame i	BS0,1 of Frame i+1	
3	BS2 of Frame i+1	BS1,2,3 of Frame i	BS0 of Frame i
4	BS1 of Frame i	BS0,1,2 of Frame i+1	
5	BS3 of Frame i+1	BS2,3,4 of Frame i	BS1 of Frame i
6	BS2 of Frame i	BS1,2,3 of Frame i+1	
7	BS4 of Frame i+1	BS3,4,5 of Frame i	BS2 of Frame i
…	…	…	…

FIGURE 9. The BS replacement strategy and the BS processing order of the circular buffer for FRUC-BME.

C. INTER-D DATA REUSE METHOD FOR FRUC-ME

We propose Inter-D to further reduce the on-chip memory size of Inter-E. Multiple SRSs instead of whole frames are put on chip for Inter-D. One SRS buffer stores one SRS. Two current frames are processed alternately in one time period for FRUC-UME (Fig. 10). Two SRS buffers for Frame *i* and Frame *i-1* and one CB buffer for Frame *i* + *1* are integrated on chip. In computing Frame *i-1*, Frame *i* assumes the role of current frame, while in computing Frame *i* + *1*, Frame *i* assumes the role of previous frame. The goal is to load

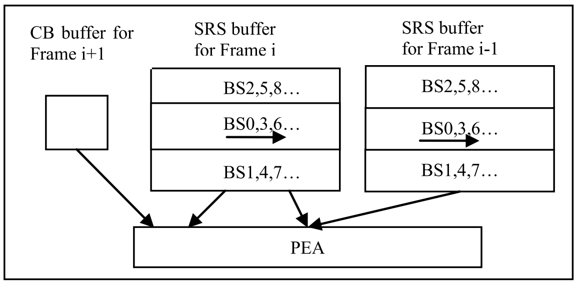


FIGURE 10. **Inter-D Architecture for FRUC-UME. Two SRS buffers, one CB buffer and one PEA are put on chip.**

Frame *i* into on-chip memory only once instead of twice. The CBs of Frame *i* are contained in the SRS buffer for Frame *i*. So we make the PEA process current BSs (or CB strips) of Frame *i* and Frame *i + 1* alternately (Fig. 12) and take the same scan order for the two frames. After processing BS0 of Frame *i + 1* in Step0, the PEA goes to process BS0 of Frame *i* in Step1. BS0 of Frame *i* is already in the SRS buffer for Frame *i* after Step0, so it is reused on chip in Step1. Then PEA goes to process BS1 of Frame *i + 1* in Step2. By this means, the current BSs of Frame *i* are always in on-chip memory when they are needed so Frame *i* is reused in SRS buffer. Nevertheless, Frame *i-1* and Frame *i + 1* are still needed to be loaded twice. If *m* current frames are processed alternately in one time period, there is one frame which is loaded twice from off-chip memory to on-chip memory every *m* frames. Therefore, Inter-D *Ra* for FRUC-UME is calculated as in (7).

$$\begin{aligned} Ra &= Ra\_pre + Ra\_cur \\ &= WH/WH + (1/m)\,(WH/WH) \\ &= 1 + 1/m. \end{aligned} \tag{7}$$

For FRUC-BME, the difference from FRUC-UME is that an SRS buffer is used to replace the CB buffer for Frame *i + 1* (Fig. 11) because both SR of the previous frame and SR of the current frame are needed for each interpolated MB. The processing order of BSs (Fig. 12) and calculating method of *Ra* are the same as FRUC-UME.

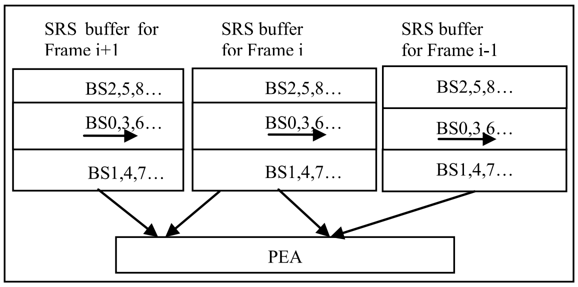
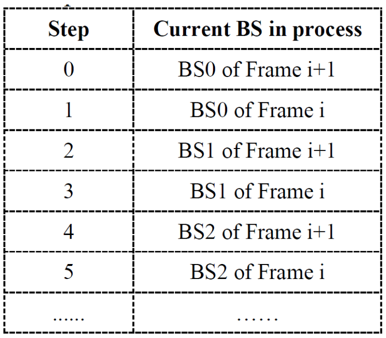


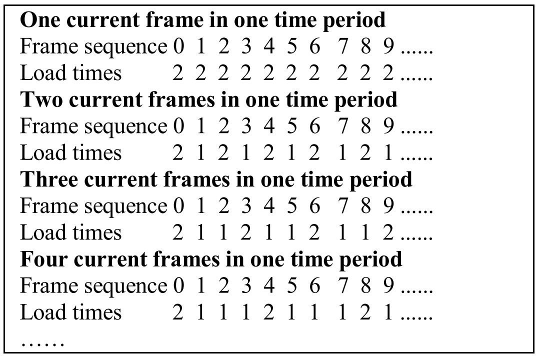
FIGURE 11. **Inter-D architecture for FRUC-BME. Three SRS buffers and one PEA are put on chip.**

Data reuse efficiency can be improved by increasing the number of SRS buffers. In this case, more current frames are processed in the same period of time. If there is only one



Step	Current BS in process
0	BS0 of Frame i+1
1	BS0 of Frame i
2	BS1 of Frame i+1
3	BS1 of Frame i
4	BS2 of Frame i+1
5	BS2 of Frame i
……	……

FIGURE 12. **The order of the current BSs processed by the PEA for FRUC-UME or FRUC-BME.**



One current frame in one time period

Frame sequence	0	1	2	3	4	5	6	7	8	9	......
Load times	2	2	2	2	2	2	2	2	2	2	......

Two current frames in one time period

Frame sequence	0	1	2	3	4	5	6	7	8	9	......
Load times	2	1	2	1	2	1	2	1	2	1	......

Three current frames in one time period

Frame sequence	0	1	2	3	4	5	6	7	8	9	......
Load times	2	1	1	2	1	1	2	1	1	2	......

Four current frames in one time period

Frame sequence	0	1	2	3	4	5	6	7	8	9	......
Load times	2	1	1	1	2	1	1	1	2	1	......

……

FIGURE 13. **Load times of the frames in a frame sequence for Inter-D.**

current frame in one time period, the data can only be intra-frame reused and each frame is loaded twice (Fig. 13). The off-chip memory traffic decreases as we add more current frames which are processed in the same time period, e.g. only one fourth of the frames are loaded twice when four current frames are processed in the same time period. Furthermore, Inter-D can exploit data reuse between adjacent SRSs within the previous frame in SRS buffer, which means that Inter-D does not affect the use of Intra-D. On the other hand, parallelism can also be increased with more PEAs.

D. COMPARISON BETWEEN DATA REUSE LEVELS

Redundant access (*Ra*) and on-chip buffer size are computed for different data reuse levels of FRUC-UME and shown in Table 1, where *n* is the parameter used in Intra-C+ data reuse. No reuse means that no data reuse schemes are used. It is assumed that Intra-C+ takes stitched zigzag scan and all the other data reuse schemes take raster scan. The proposed Inter-D data reuse method has a smaller Ra than its intra-frame counterpart Intra-D. When *m* becomes larger, *Ra* becomes smaller and on-chip buffer size becomes larger. For example, Inter-D *Ra* will be reduced to nearly 1/2 of Intra-D *Ra* when *m* is large enough. The size of the CB buffer is not considered when computing the on-chip buffer size because it is much smaller than SR, SRS or frame buffer. Inter-D on-chip buffer size is *m* times of Intra-D on-chip buffer size. Inter-E has the smallest *Ra*. The proposed new Inter-E reduces the

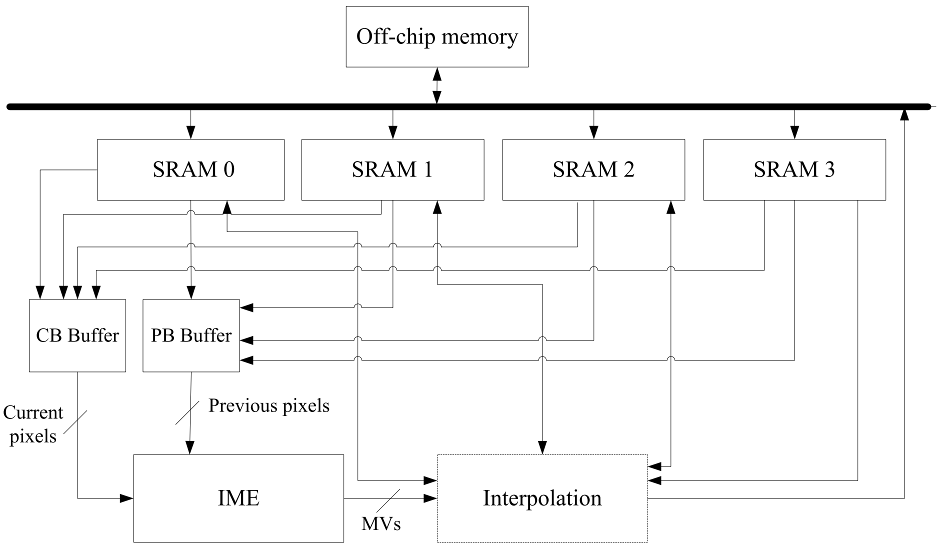


FIGURE 14. The proposed architecture for Inter-D FRUC-UME.

buffer size of Inter-E nearly by half while it has the same *Ra* as Inter-E.

For FRUC-BME (Table 2), Inter-D data reuse method also has a smaller *Ra* than its intra-frame counterpart Intra-D. The on-chip buffer size of Inter-D is $(m + 1)/2$ times as large as Intra-D for FRUC-BME. On-chip memory size of Intra-D, Intra-C, Intra-B and Intra-A in FRUC-BME are all twice as large as their corresponding levels in FRUC-UME.

TABLE 3. Three case studies for FRUC-UME. (a) 720p, 30fps, $SRH = SRV = 16, N = 16$. (b) 1080p, 30fps, $SRH = SRV = 32, N = 16$. (c) 4K, 60fps, $SRH = SRV = 128, N = 64$.

(a)

	Ra	Bandwidth (MByte/sec)	on-chip buffer Size (KB)
No reuse	257	7105.54	0
Intra-A [17]	33	912.38	0.24
Intra-B [17]	5	138.24	0.48
Intra-C [17]	3	82.94	0.96
Intra-C+ [18]	2.25	62.21	2.45
Intra-D [17]	2	55.3	19.43
Inter-D	1.25	34.56	77.72
Inter-E [17]	1	27.65	1843.2
New Inter-E	1	27.65	962.56

(b)

	Ra	Bandwidth (MByte/sec)	on-chip buffer Size (KB)
No reuse	1025	63763.2	0
Intra-A [17]	97	6034.18	0.24
Intra-B [17]	10	622.08	0.72
Intra-C [17]	4	248.83	2.21
Intra-C+ [18]	2.5	155.52	4.47
Intra-D [17]	2	124.42	60.48
Inter-D	1.25	77.76	241.92
Inter-E [17]	1	62.21	4147.2
New Inter-E	1	62.21	2135.04

(c)

	Ra	Bandwidth (MByte/sec)	on-chip buffer Size (KB)
No reuse	16385	8154224.64	0
Intra-A [17]	385	191600.64	4.03
Intra-B [17]	10	4976.64	12.1
Intra-C [17]	4	1990.66	36.48
Intra-C+ [18]	2.5	1244.16	73.15
Intra-D [17]	2	995.33	503.81
Inter-D	1.25	622.08	2015.24
Inter-E [17]	1	497.66	16588.8
New Inter-E	1	497.66	8785.92

TABLE 4. **Three Case Studies for FRUC-BME. (a) 720p, 30fps, *SRH* = *SRV* = *16*, *N* = *16*. (b) 1080p, 30fps, *SRH* = *SRV* = *32*, *N* = *16*. (c) 4K, 60fps, *SRH* = *SRV* = *128*, *N* = *64*.**

(a)

	Ra	Bandwidth (MByte/sec)	on-chip buffer Size (KB)
No reuse	512	14155.78	0
Intra-A [17]	128	3538.94	0.48
Intra-B [17]	8	221.18	0.96
Intra-C [17]	4	110.59	1.92
Intra-C+ [18]	2.5	69.12	4.9
Intra-D [17]	2	55.3	38.86
Inter-D	1.25	34.56	97.13
Inter-E [17]	1	27.65	1843.2
New Inter-E	1	27.65	983.04

(b)

	Ra	Bandwidth (MByte/sec)	on-chip buffer Size (KB)
No reuse	2048	127401.98	0
Intra-A [17]	192	11943.94	0.48
Intra-B [17]	18	1119.74	1.44
Intra-C [17]	6	373.25	2.21
Intra-C+ [18]	3	186.62	4.47
Intra-D [17]	2	124.42	60.48
Inter-D	1.25	77.76	302.41
Inter-E [17]	1	62.21	4147.2
New Inter-E	1	62.21	2165.76

(c)

	Ra	Bandwidth (MByte/sec)	on-chip buffer Size (KB)
No reuse	32768	16307453.95	0
Intra-A [17]	768	382205.95	8.06
Intra-B [17]	18	8957.95	24.19
Intra-C [17]	6	2985.98	72.96
Intra-C+ [18]	3	1492.99	146.31
Intra-D [17]	2	995.33	1007.62
Inter-D	1.25	622.08	2519.05
Inter-E [17]	1	497.66	16588.8
New Inter-E	1	497.66	8785.92

E. INTER-D IMPLEMENTATION FOR FRUC-UME

The proposed Inter-D data reuse shows a good tradeoff between on-chip memory size and off-chip memory traffic. Therefore, we design an FRUC-UME architecture with Inter-D data reuse (Fig. 14) when $m = 4$. IME mainly includes one SAD (sum of absolute differences) Tree [22] and an MV (motion vector) selector. Interpolation module in Fig. 11 is used to generate the interpolated frame for a complete FRUC architecture. We design a two-level memory hierarchy which is used to load data from off-chip memory to IME. The first level is a 16 × 16 CB buffer together with a 16 × 16 PB (previous block) buffer. CB buffer is designed as a register array to store the current block and receives current pixels from four SRAMs. The PB buffer is also designed as a register array to store the previous block and employs Intra-A. The PB buffer receives pixels of the previous frame from four SRAMs with the order shown in Fig. 12. The second level of the memory hierarchy contains four SRAMs to receive pixels from off-chip memory, which implements Inter-D data reuse. FRUC-UME architecture with new Inter-E data reuse is similar to that with Inter-D data reuse. The main difference is that SRAM is used to implement the circular buffer (Fig. 7).

$$Bandwidth = Ra \times W \times H \times f \tag{8}$$

$$\begin{aligned} P_{DRAM} = P_{DRAM\_static} &+ \alpha_1 \times Throughput_{read} \\ &+ \alpha_2 \times Throughput_{write} \end{aligned} \tag{9}$$

IV. CASE STUDIES

We give three case studies (720p, 1080p and 4K) for analysis and comparison of different data reuse levels. *Ra* and on-chip buffer size of FRUC-UME and FRUC-BME are calculated on the basis of Table 1 and Table 2 respectively, where $m = 4$ and $n = 4$. Intra-A, Intra-B, Intra-C, Intra-C+, Intra-D and Inter-E are data reuse methods in the literature. Inter-D and New Inter-E are the proposed inter-frame data reuse methods. *Bandwidth* in (8) is off-chip memory bandwidth demand for FRUC-ME, where *Ra* is redundant access factor and *f* is frame rate.

For FRUC-UME, Inter-E needs 1.8MB on-chip buffer for 720p, 4.1MB on-chip buffer for 1080p and 16.6MB on-chip buffer for 4K. New Inter-E with the circular buffer reduces the on-chip buffer size of Inter-E nearly by half (Table 3). Inter-D has better data reuse efficiency than its intra-frame counterpart Intra-D for 720p, 1080p and 4K. The bandwidth requirement reduction of Inter-D is 37.5%, compared with Intra-D for all the three specifications.

For FRUC-BME (Table 4), the on-chip buffer size of Inter-D is 25% larger than the corresponding data reuse level of FRUC-UME when *m* equals 4 because it needs one more buffer on chip. Inter-D has better data reuse efficiency than its intra-frame counterpart Intra-D for all the three specifications, and the bandwidth requirement reductions are all 37.5%.

Considering the importance of power efficiency [23]–[25], we also evaluate the power consumption of different data reuse levels. The proposed method will not affect computational complexity and the power difference is mainly caused by difference of off-chip memory accesses, so we only evaluate the DRAM power according to (9) [26]. We assume that P_{DRAM_static} and $Throughput_{write}$ in (9) is the same for different data reuse levels. Therefore, we only give the comparison of DRAM read power for FRUC-UME (Table 5) and FRUC-BME (Table 6). α_1 equals 1.12 Watt/(GB/s) and Throughput equals Bandwidth in (8).

TABLE 5. Power consumption of different data reuse levels for FRUC-UME (1080p, 30fps, *SRH* = *SRV* = *32*, *N* = *16*).

Reuse Level	Bandwidth (GByte/sec)	Power(Watt)
No reuse	63.76	71.41
Intra-A [17]	6.03	6.75
Intra-B [17]	0.62	0.69
Intra-C [17]	0.25	0.28
Intra-C+ [18]	0.16	0.18
Intra-D [17]	0.12	0.13
Inter-D	0.08	0.09
Inter-E [17]	0.06	0.07
New Inter-E	0.06	0.07

TABLE 6. Power consumption of different data reuse levels for FRUC-BME (1080p, 30fps, *SRH* = *SRV* = *32*, *N* = *16*).

Reuse Level	Bandwidth (GByte/sec)	Power(Watt)
No reuse	127.40	142.69
Intra-A [17]	11.94	13.37
Intra-B [17]	1.12	1.25
Intra-C [17]	0.37	0.41
Intra-C+ [18]	0.19	0.21
Intra-D [17]	0.12	0.13
Inter-D	0.08	0.09
Inter-E [17]	0.06	0.07
New Inter-E	0.06	0.07

For FRUC-UME, Intra-A, Intra-B, Intra-C and Intra-D can reduce a large amount of power consumption when compared with no data reuse. The proposed inter-frame data reuse method has lower power consumption than its intra-frame counterpart. The power reduction for Inter-D is 37.5%, compared with Intra-D. For FRUC-BME, the proposed Inter-D data reuse method also has lower power consumption than its intra-frame counterpart Intra-D, and the reduction is 37.5%.

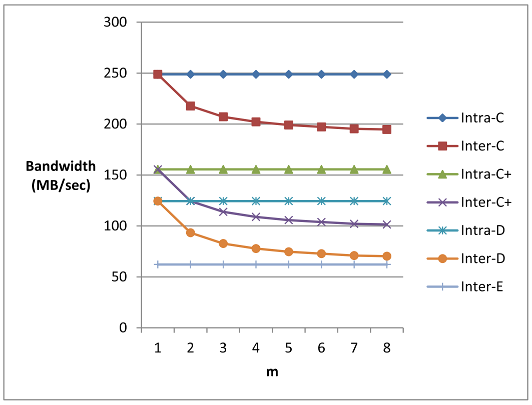


FIGURE 15. The bandwidth requirement of FRUC-UME for 1080p.

Because m is the main factor which will affect the off-chip memory traffic for the proposed data reuse method, we give the demand on bandwidth for 1080p with different m in Fig. 15 and Fig. 16 for different data reuse levels of FRUC-UME and FRUC-BME respectively. It is found that the demand on bandwidth of all the intra-frame data reuse levels and Inter-E do not change when m increases. The demand on bandwidth for Inter-D becomes lower but the magnitude of reduction becomes smaller when m increases.

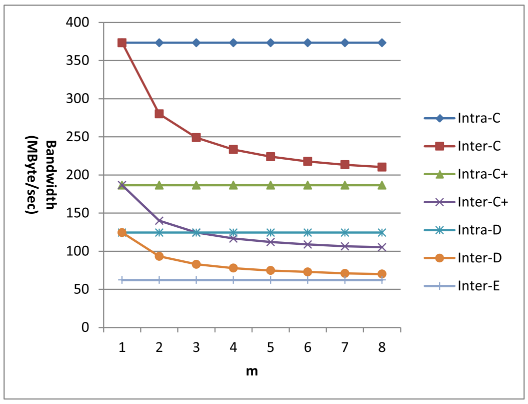


FIGURE 16. The bandwidth requirement of FRUC-BME for 1080p.

V. CONCLUSION

In this paper, we propose a new inter-frame data reuse method for full search true motion estimation. The new method includes two levels, Inter-D and new Inter-E, which avoid repeatedly loading the same frame into on-chip buffer. It effectively decreases the number of off-chip memory accesses and power consumption, which both outperform traditional intra-frame method.

REFERENCES

[1] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[2] G. J. Sullivan, J. R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[3] W. Xu, S. Yin, L. Liu, Z. Liu, and S. Wei, "High-performance motion estimation for image sensors with video compression," *Sensors*, vol. 15, no. 8, pp. 20752–20778, 2015. [Online]. Available: http://www.mdpi.com/1424-8220/15/8/20752

[4] L. Meng, J. Liang, U. Samarawickrama, Y. Zhao, H. Bai, and A. Kaup, "Multiple description coding with randomly and uniformly offset quantizers," *IEEE Trans. Image Process.*, vol. 23, no. 2, pp. 582–595, Feb. 2014.

[5] S.-J. Kang, S. Yoo, and Y. H. Kim, "Dual motion estimation for frame rate up-conversion," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 12, pp. 1909–1914, Dec. 2010.

[6] S. Dikbas and Y. Altunbasak, "Novel true-motion estimation algorithm and its application to motion-compensated temporal frame interpolation," *IEEE Trans. Image Process.*, vol. 22, no. 8, pp. 2931–2945, Aug. 2013.

[7] H. Ho, R. Klepko, N. Ninh, and D. Wang, "A high performance hardware architecture for multi-frame hierarchical motion estimation," *IEEE Trans. Consum. Electron.*, vol. 57, no. 2, pp. 794–801, May 2011.

[8] M. T. Pourazad, P. Nasiopoulos, and R. K. Ward, "An H.264-based scheme for 2D to 3D video conversion," *IEEE Trans. Consum. Electron.*, vol. 55, no. 2, pp. 742–748, May 2009.

[9] W. Xu *et al.*, "Reconfigurable VLSI architecture for real-time 2D-to-3D conversion," *IEEE Access*, vol. 5, pp. 26604–26613, 2017.

[10] B. T. Koga, K. Linuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video-conferencing," in *Proc. NTC*, 1981, pp. G5.3.1–G5.3.5.

[11] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, no. 4, pp. 438–442, Aug. 1994.

[12] O. Ndili and T. Ogunfunmi, "Hardware-oriented modified diamond search for motion estimation in h.246/AVC," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2010, pp. 749–752.

[13] J. Y. Tham, S. Ranganath, M. Ranganath, and A. A. Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 4, pp. 369–377, Aug. 1998.

[14] L.-M. Po and W.-C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 313–317, Jun. 1996.

[15] J.-H. Hsieh and T.-S. Chang, "Algorithm and architecture design of bandwidth-oriented motion estimation for real-time mobile video applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 1, pp. 33–42, Jan. 2013.

[16] S. Yin, W. Xu, J. Li, L. Liu, and S. Wei, "CWFP: Novel collective writeback and fill policy for last-level dram cache," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 7, pp. 2548–2561, Jul. 2016.

[17] J.-C. Tuan, T.-S. Chang, and C.-W. Jen, "On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 1, pp. 61–72, Jan. 2002.

[18] C.-Y. Chen, C.-T. Huang, Y.-H. Chen, and L.-G. Chen, "Level C+ data reuse scheme for motion estimation with corresponding coding orders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 4, pp. 553–558, Apr. 2006.

[19] X. Wen, O. C. Au, J. Xu, L. Fang, R. Cha, and J. Li, "Novel RD-optimized VBSME with matching highly data re-usable hardware architecture," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 2, pp. 206–219, Feb. 2011.

[20] S. D. Kim and M. H. Sunwoo, "MESIP: A configurable and data reusable motion estimation specific instruction-set processor," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 10, pp. 1767–1780, Oct. 2013.

[21] A. Akin, M. Cetin, Z. Ozcan, B. Erbagci, and I. Hamzaoglu, "An adaptive bilateral motion estimation algorithm and its hardware architecture," *IEEE Trans. Consum. Electron.*, vol. 58, no. 2, pp. 712–720, May 2012.

[22] C.-Y. Chen, S.-Y. Chien, Y.-W. Huang, T.-C. Chen, T.-C. Wang, and L.-G. Chen, "Analysis and architecture design of variable block-size motion estimation for H.264/AVC," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 3, pp. 578–593, Mar. 2006.

[23] T. Zhang, W. Chen, Z. Han, and Z. Cao, "A cross-layer perspective on energy-harvesting-aided green communications over fading channels," *IEEE Trans. Veh. Technol.*, vol. 64, no. 4, pp. 1519–1534, Apr. 2015.

[24] T. Zhang, W. Chen, and H. Wang, "Energy harvesting aided multiuser transmission in spectrum sharing networks," *IEEE Access*, vol. 4, pp. 4038–4045, 2016.

[25] J. Wang, B. Gong, H. Liu, and S. Li, "Model and algorithm for heterogeneous scheduling integrated with energy-efficiency awareness," *Trans. Inst. Meas. Control*, vol. 38, no. 4, pp. 452–462, 2015.

[26] J. Lin, H. Zheng, Z. Zhu, H. David, and Z. Zhang, "Thermal modeling and management of dram memory systems," in *Proc. Int. Symp. Comput. Archit.*, 2007, pp. 312–322.

WEIZHI XU was born in China in 1982. He received the Ph.D. degree in computer architecture from the Institute of Computing Technology, Chinese Academy of Sciences. He was a Post-Doctoral Researcher with the Institute of Microelectronics, Tsinghua University. He is currently an Assistant Professor with the School of Information Science and Engineering, Shandong Normal University. His research interests include video processing and high performance computing.

HUI YU was born in China in 1983. She received the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences. She is currently an Assistant Professor with the School of Management Science and Engineering, Shandong Normal University. Her research interests include signal processing and machine translation.

DIANJIE LU was born in China in 1981. He received the Ph.D. degree in computer architecture from the Institute of Computing Technology, Chinese Academy of Sciences. He is currently an Associate Professor with the School of Information Science and Engineering, Shandong Normal University, China. His current research interests include cognitive wireless network and mobile video distribution network.

FANGAI LIU received the Ph.D. degree in computer architecture from the Institute of Computing Technology, Chinese Academy of Sciences. He is currently a Professor with the School of Information Science and Engineering, Shandong Normal University, China. His current research interests include parallel and distributed computing.

ZHIYONG LIU received the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 1987. He was a Professor with the Institute of Computing Technology in 1987. He was the Executive Director General of the Directorate for Information Sciences, National Natural Science Foundation of China, from 1995 to 2006. He has worked on networks, high performance architectures and algorithms, parallel and distributed processing, and bioinformatics. He is currently a Chair Professor with the Advanced Research Laboratory, Institute of Computing Technology, Chinese Academy of Sciences. His research interests include computer architectures, algorithms, networks, and parallel and distributed processing. He has served as a Board Member, a Referee, an Editor, and an Invited Editor for academic journals, program/organization committee member/chairman, advisory committee member/chairman for national and international conferences, steering expert committee member for research initiatives, and investigator/principal investigator of research projects on computer control systems, computer architectures and algorithms. He was a recipient of the China National Science Congress Prize, a National Prize for Science and Technology in China.