A Novel Data Reuse Method for Motion Estimation in Video Applications

Weizhi Xu[1,2], Hui Yu[3], Xin Wang[3], Yanhui Ding[1,2], Dianjie Lu[1,4], Guijuan Zhang[1,4], and Zengzhen Shao[1,2]

[1] School of Information Science and Engineering, Shandong Normal University, Jinan, China, 250358

[2] Shandong Provincial Logistics Optimization and Predictive Engineering Technology Research Center, Jinan, China, 250014

[3] School of Management Science and Engineering, Shandong Normal University, Jinan, China, 250014

[4] Shandong Provincial Key Laboratory for Novel Distributed Computer Software Technology, Jinan, China, 250358

xuweizhi@sdnu.edu.cn, huiyu0117@sdnu.edu.cn, wxyouha123@163.com, yanhuiding@126.com, ludianjie@sina.com, guijuanzhang@gmail.com, shaozengzhen@163.com

ABSTRACT

Motion estimation is a kernel algorithm in many video applications. Full search integer motion estimation (FSIME) can find the best result but it usually takes plenty of time. Previous work on accelerating FSIME exploited intra-frame data reuse within reference frame. We propose a novel data reuse scheme which uses not only intra-frame but also inter-frame data reuse for FSIME. Motion estimation in frame rate up-conversion (FRUC-ME) is used as a case study. A frame is loaded into on-chip memory only once instead of twice and used for two interpolated frames. Different inter-frame data reuse methods are presented and analysed. They give useful tradeoff between off-chip bandwidth requirement and on-chip memory size. The proposed data reuse methods all show better data reuse efficiency than the traditional methods, so the off-chip memory traffic is reduced effectively, as much as 37.5%.

CCS CONCEPTS

• **Computing methodologies** →Computer vision

KEYWORDS

motion estimation, memory traffic, FRUC

Mobimedia 2017, July 13-14, Chongqing, People's Republic of China

For its simplicity and efficiency, block matching is usually used to find the most similar reference macro-block (MB) to the current MB. Full search integer ME (FSIME) can find the optimal point in the search range but it needs a large number of computations and memory accesses. 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, but they may not find the best result and many of them are not proper for hardware implementation. Our focus is FSIME in this paper.

The speed of computing has overtaken the speed of memory access in recent years, so it is important to reduce off-chip bandwidth requirement of FSIME to improve performance particularly for real-time applications [4]. Reusing data on chip is one effective way to reduce off-chip memory traffic for FSIME [5]-[8]. However, previous work focused on intra-frame data reuse within reference frame while inter-frame data reuse was not considered. Only with intra-frame data reuse, each frame has to be loaded from off-chip memory to on-chip memory twice for FRUC-ME, the first time as current frame, the second time as previous frame [9].

Different processors offer very different on-chip memory sizes. A proper data reuse method can exploit the best of available cache size. For example, with the traditional data reuse methods [5], Level D data reuse needs at least a 60KB on-chip memory which can reduce the off-chip memory bandwidth to 124 MByte/sec and Level E needs at least a 2MB on-chip memory which can reduce the off-chip memory bandwidth to 62 MByte/sec (TABLE II). If the on-chip memory size of available platform is 256KB, we can only use Level 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 Level D and E, e.g. 241KB on-chip memory with 77 MByte/sec off-chip memory bandwidth (Inter-D in TABLE II), it will be the best data reuse method for this cache size.

In this paper, we propose a new method to exploit inter-frame data reuse for FRUC-ME. The inter-frame data reuse scheme can effectively reduce off-chip memory traffic with affordable on-chip memory size. For FRUC-ME, most of the frames are processed as current frame and reference frame at the same time in order to be loaded into on-chip memory only once. Different levels of the

proposed data reuse method for FRUC-ME give proper tradeoffs between data reuse efficiency and on-chip memory size. According to the case studies, we find that off-chip memory traffic can be reduced by 37.5% with the new data reuse method for FRUC-ME.

2 DATA LOCALITY ANALYSIS

In this paper, Macro Block (MB, a block of pixels with size of N×N), Block Strip (BS, a row of MBs), Search Range (SR, the search range in reference frame for a current block) and SR Strip (SRS, a row of search ranges) are four levels of data ranges within the frame. Current Block (CB) is an MB in the current frame. Different scan orders, such as raster scan, snake scan and smart snake scan, can be used for different data reuse methods with different off-chip memory traffic [7], [8]. PEA is the IME engine for computing the sum of absolute differences (SAD) and motion vector.

Four intra-frame data reuse levels are presented in [5], Level A, Level B, Level C and Level D [10]. Level A+ (between Level A and B) [6] is similar to Level C+ (between Level C and D) [7]. However, Level A+ is often used to get data from SRAM to registers while Level C+ is usually implemented to load data from off-chip memory to on-chip memory. Level E is considered to be impractical because it demanded a large on-chip memory [5]. Traditionally, Ra (the redundancy access factor), is used to evaluate data reuse efficiency [5], but the memory traffic of loading the reference frame is not considered for Ra.

3 NEW DATA REUSE METHOD FOR FRUC-ME

Previous work tried to improve intra-frame data reuse. But each frame has to be loaded into on-chip memory twice for FRUC-ME. In order to reduce this kind of data redundancy, we propose an inter-frame data reuse method to load frames into on-chip memory only once for FRUC-ME. Three levels of inter-frame data reuse are developed in this section.

3.1 New Ra for FRUC-ME

Previous work [5]-[8] did not consider the inter-frame data redundancy when computing Ra, and the memory traffic of loading the current frame was neglected. So we propose a new definition of Ra to compare different intra-frame and inter-frame data reuse levels. In (1), new Ra includes both intra-frame redundant access (Ra-intra) and inter-frame redundant access (Ra_inter). Ra_intra describes the data redundancy of loading reference frame in (2). $memref_{load}$ is the memory traffic to load reference frame. Ra_inter rises from loading current frame to on-chip memory. Ra_inter is defined in (3), where $memcur_{load}$ equals memory traffic to load one current frame. A larger Ra stands for higher memory bandwidth requirement.

$$Ra = Ra\_intra + Ra\_inter \quad (1)$$

$$Ra\_intra = \frac{memref_{load}}{pixel\ number\ in\ a\ frame} \quad (2)$$

$$Ra\_inter = \frac{memcur_{load}}{pixel\ number\ in\ a\ frame} \quad (3)$$

For FRUC-ME, Ra of intra-frame data reuse levels (TABLE I) is computed according to (1). Ra_inter always equals 1 because each current frame is loaded only once. For example, Intra-C Ra is computed as follows.

$$(1 + SR_V / N) + (W \times H) / (W \times H) = (1 + SR_V / N) + 1$$

3.2 Inter-E Data Reuse Method

One way to implement inter-frame data reuse is to put whole frames on chip. Two frame buffers and one PEA can be used to implement Inter-E data reuse for FRUC-ME. 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 on-chip buffer only once. Note that, we use PEA to implement the proposed data reuse scheme 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.

3.3 Inter-D Data Reuse Method

We propose Inter-D to reduce the buffer size of Inter-E. Multiple SRSs instead of whole frames are kept on chip for Inter-D. One SRS buffer stores one SRS. One PEA processes two current frames during one time period for FRUC-ME in Fig. 1. Two SRS buffers for Frame i and i-1 and one CB buffer for Frame i+1 are integrated on chip. Frame i is the current frame for Frame i-1 and the previous frame for Frame i+1. The goal is to load 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 the CB strips (or current BSs) of Frame i and Frame i+1 alternately (Fig. 2) 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. After that, PEA processes BS1 of Frame i+1 in Step2. By this means, the CB strips of Frame i are always on chip when they are needed and do not have to be loaded from off-chip memory so Frame i are inter-frame reused in the SRS buffer. However, Frame i-1 and i+1 are still needed to be loaded twice. Therefore, Ra of Inter-D is calculated as follows.

$$\frac{W \times H}{W \times H} + \frac{1}{m} \times \frac{W \times H}{W \times H} = 1 + 1/m$$

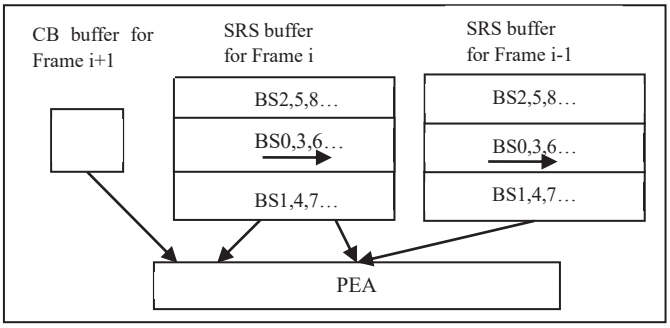

Fig. 1. Inter-D Architecture for FRUC-ME.

Data reuse efficiency can be improved by increasing the number of SRS buffers and processing more current frames in the same period of time. If there is only one current frame in one time period, the data can only be intra-frame reused and each frame is loaded twice (Fig. 3). The data reuse efficiency increases as the number of frames processed during one time period increases, e.g. only one fourth of the frames are loaded twice when four instead of two current frames are processed in one time period. In addition, Inter-D can exploit data reuse between adjacent SRSs, which means that Inter-D is compatible with Intra-D.

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
.....	.....

Fig. 2. The order of the current BSs processed by the PEA.

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 .....
.....

Fig. 3. Load times of the frames in a frame sequence for Inter-D or Inter-C.

3.4 Inter-C Data Reuse Method

Inter-C is proposed to further reduce the on-chip memory size to multiple SR buffers. Inter-C architecture for FRUC-ME is described in Fig. 4. Two SR buffers and one CB buffer are integrated on chip. PEA processes CBs of two current frames alternately in one time period. Frame i is the current frame for Frame i-1 and the previous frame for Frame i+1. Pixels of Frame i are loaded into SR buffer and shared by Frame i-1 and i+1. One important thing is how to arrange the processing order of the CBs so that the data in SR buffer for Frame i are always used both as CB for Frame i-1 and as SR for Frame i+1. We make the accesses of all current frames (Frame i and i+1) take the same start point and scan order. For example, after matching CB0 of Frame i+1 with SR0 of Frame i, PEA goes to process CB0 of Frame i. CB0 is contained in SR0 which is already in the buffer so CB0 is reused on chip for Frame i. Then PEA goes to process CB1 of Frame i+1 and CB1 of Frame i (Fig. 5). By this means, the data of Frame i are inter-frame reused and are loaded only once from off-chip memory, but Frame i-1 and i+1 are still needed to be loaded twice. Ra of Inter-C is computed as follows.

$$(1+SR_V/N)+1/m\times(W\times H)/(W\times H)=(1+SR_V/N)+1/m$$

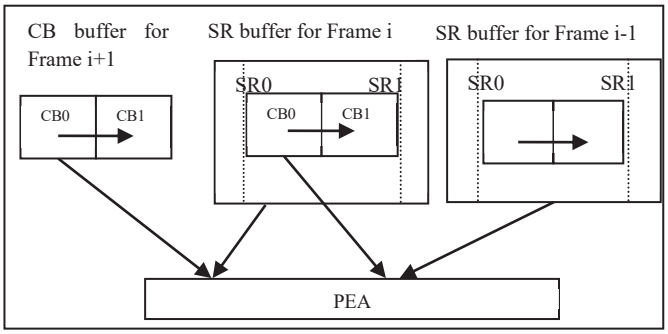

Fig. 4. Inter-C Architecture for FRUC-ME.

Step	CB in process
0	CB0 of Frame i+1
1	CB0 of Frame i
2	CB1 of Frame i+1
3	CB1 of Frame i
.....	.....

Fig. 5. The order of the CBs processed by PEA.

TABLE I RA AND ON-CHIP MEMORY SIZE FOR FRUC-ME

Level	Ra	On-chip Memory Size
Intra-C	(1+ SR_V /N)+1	(SR_H+N-1) (SR_V+N-1)
Inter-C	(1+ SR_V /N)+1/m	m(SR_H+N-1) (SR_V+N-1)
Intra-C+	(1+ SR_V /nN)+1	(SR_H+N-1)(SR_V+nN-1)
Inter-C+	(1+ SR_V /nN)+1/m	m(SR_H+N-1)(SR_V+nN-1)
Intra-D	1+1	(SR_H+W-1) (SR_V-1)
Inter-D	1+1/m	m(SR_H+W-1) (SR_V-1)
Inter-E	1	2WH

Data reuse efficiency can be improved by using more SR buffers and processing more frames in the same time period (Fig. 3). If there is only one current frame processed during a time period, each frame in the frame sequence is loaded twice. The data reuse efficiency increases when the number of frames

processed during the same time period increases. Furthermore, Inter-C can exploit data reuse within SRS in SR buffer as Intra-C.

We give Ra and on-chip memory size for different data reuse levels of FRUC-ME (TABLE I). m is the number of the current frames processed in one time period. n is the parameter of C+ scheme. Intra-C, C+, D and Inter-E are traditional data reuse methods. Inter-C, C+, D are proposed inter-frame data reuse methods. The new inter-frame data reuse methods always have better data reuse efficiency than their intra-frame counterparts. A larger m results in a smaller Ra but a larger buffer size. For example, Ra of Inter-D is nearly half of Intra-D when m is large enough. The size of the CB buffer is not considered when computing the on-chip memory size because it is much smaller than SR, SRS or frame buffer. The on-chip memory size of Inter-C, Inter-C+ and Inter-D is m times of Intra-C, Intra-C+ and Intra-D respectively. Both Inter-C+ and Intra-C+ take stitched zigzag scan [6] and all the other data reuse schemes take raster scan.

4 Case Studies

We give three case studies (1080p, 720p and 4K) for different data reuse levels. Ra and on-chip memory size are computed according to TABLE I , where m and n are both equal to 4. Bandwidth in (4) describes off-chip memory bandwidth requirement. f is the frame rate.

$$Bandwidth = f \times W \times H \times Ra \quad (4)$$

For Inter-E, the on-chip memory size is 4.1MB, 1.8MB and 16.6MB for 1080p, 720p and 4K respectively (TABLE II). The three new inter-frame data reuse schemes have better data reuse efficiency than their intra-frame counterparts for 1080p, 720p, and 4K. For 1080p and 4K, the bandwidth requirement reductions of Inter-C, C+ and D are 18.8%, 30% and 37.5% respectively, compared with Intra-C, C+ and D. For 720p, the bandwidth requirement reductions are 25%, 33.3% and 37.5% respectively for the three data reuse levels.

TABLE II THREE CASE STUDIES FOR FRUC-ME

Reuse Level	1080p, 30fps, SR_H=SR_V=32, N=16			720p, 30fps, SR_H=SR_V=16, N=16			4K, 60fps, SR_H=SR_V=128, N=64		
	Ra	Bandwidth (MByte/sec)	On-chip Memory Size(KB)	Ra	Bandwidth (MByte/sec)	On-chip Memory Size(KB)	Ra	Bandwidth (MByte/sec)	On-chip Memory Size(KB)
Intra-C	4	248.83	2.21	3	82.94	0.96	4	1990.66	36.48
Inter-C	3.25	202.18	8.84	2.25	62.21	3.84	3.25	1617.41	145.92
Intra-C+	2.5	155.52	4.47	2.25	62.21	2.45	2.5	1244.16	73.15
Inter-C+	1.75	108.86	17.86	1.5	41.47	9.80	1.75	870.91	292.61
Intra-D	2	124.42	60.48	2	55.30	19.43	2	995.33	503.81
Inter-D	1.25	77.76	241.92	1.25	34.56	77.72	1.25	622.08	2015.24
Inter-E	1	62.21	4147.20	1	27.65	1843.20	1	497.66	16588.80

5 CONCLUSIONS

In this paper, we propose a new inter-frame data reuse method for full search integer motion estimation. The new method avoids repeatedly loading the same frame into on-chip buffer. The proposed data reuse scheme effectively reduces the off-chip memory traffic and outperforms the traditional intra-frame data reuse method.

ACKNOWLEDGMENTS

This work is supported in part by Shandong Provincial Natural Science Foundation, China (No. ZR2015FQ009, ZR2014FQ009 and ZR2016FP07), the NNSF of China grant (No. 61602285, 61602284, 61402270, 61303007 and 61402268), China Postdoctoral Science Foundation Funded Project (No. 2016M592697), and Project of Shandong Province Higher Educational Science and Technology Program (No. J16LN05).

REFERENCES

[1] G. J. Sullivan, J. 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.

[2] 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.

[3] 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.

[4] 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. Syst.*, vol.21, no.1, pp.33-42, Jan. 2013.

[5] J. Tuan, T. Chang and C. 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.

[6] C. Chen, C. Huang and L. 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, April 2006.

[7] X. Wen, Au O.C., J. Xu, Lu 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.

[8] S.D. Kim, and Sunwoo M., "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.

[9] A. Akin et al., "An adaptive bilateral motion estimation algorithm and its hardware architecture," *IEEE Trans. Consum. Electron.*, vol.58, no.2, pp.712-720, May 2012.

[10] D.-X. Li, W. Zheng and M. Zhang, "Architecture Design for H.264/AVC Integer Motion Estimation with Minimum Memory Bandwidth," *IEEE Trans. Consum. Electron.*, vol.53, no.3, pp.1053-1060, Aug. 2007.