The Cost of Data Dependence in Motion Vector Estimation for Reconfigurable Platforms

Su-Shin Ang [†1], George Constantinides [†], Wayne Luk [*], Peter Cheung [†]

[†]*Department of Electrical and Electronics Engineering, Imperial College*
South Kensington campus, London SW7 2AZ, United Kingdom
[1]su-shin.ang@imperial.ac.uk

[*]*Department of Computing, Imperial College*
180 Queen's Gate, South Kensington Campus, London SW7 2AZ, United Kingdom

***Abstract*— Motion vector estimation is frequently performed as a prelude to the exploitation of temporal redundancies in video applications. As a result, a large volume of work has been done to develop techniques to avoid the heavy memory access requirements of full search motion vector estimation. Often, these approaches introduce data dependence to the algorithm, leading to memory accesses which cannot be determined at design time. Consequently, this complicates the exploitation of data re-use in hardware. In this work, the cost of data dependence is quantified. Experiments indicate that a data dependent fast motion vector estimation approach is faster than full search by up to 47% in the absence of data re-use optimisation. However, full search is approximately 16 times faster than the 'fast' motion vector estimation algorithm when a static line buffering scheme and a parallel caching scheme are used respectively to exploit data re-use. Therefore, it is established that data dependence in motion vector estimation is very expensive in terms of hardware performance.**

I. Introduction

Future embedded systems are required to be transparent to multiple processing and communication standards, while preserving the quality of service. Since Field Programmable Gate Arrays (FPGAs) allow the implementation of highly parallel as well as reconfigurable systems, FPGAs afford the target application considerable performance and flexibility. For this reason, FPGAs are likely to play an increasingly important role in future embedded systems.

To increase the performance of a given application, typical features of input data may be exploited to reduce the number of external memory accesses and computations. On the other hand, data dependence may make it more difficult for hardware features to be exploited. We show how this is possible and the extent to which data dependence affects the performance of a motion vector estimation application.

Block Matching Algorithms (BMA) are popular in motion vector estimation. The MPEG 2 standard provides substantial freedom in the way in which the match for a given reference block is obtained. One way in which BMA may be sped up is by reducing the number of search points; this is achieved by limiting the matching process to the area where the target block is most likely to lie. Indeed, numerous techniques have been conceived to limit the search area; the most well-known ones include the three step search [1] and diamond search [2] techniques. Inevitably, the quality of the motion vectors from these techniques is sub-optimal. Further, the speed-up obtained from these techniques is often quantified by the reduction in the number of search points compared to full search. However, such a comparison does not account for the data dependence which is introduced and the consequent difficulty to exploit parallelism in hardware.

A number of other approaches [3], [4], [5] have been developed to speed up BMA. These techniques reduce the amount of data that needs to be processed for a given match and are unaffected by the statistical nature of the input data. Therefore, they are orthogonal to this work.

Recent work [6] compares the performance and power consumption of different motion estimation approaches using the same hardware framework. However, different motion vector estimation approaches have varying degrees of data dependence. Therefore, using the same type of memory sub-system to exploit data re-use for different techniques is sub-optimal.

In our work, comparisons are made between custom hardware implementations of a data dependent motion vector estimation algorithm [7] and full search, an alternative technique where the memory access pattern is independent of input data. A caching approach [8], which can cope with dynamic memory accesses and exploit hardware parallelism is used with the data dependent algorithm. Full search is optimised with a scratchpad memory (SPM). In the absence of memory sub-systems, the data dependent approach is faster than full search by up to 47%. However, with the addition of memory sub-systems, the data-dependent approach is found to be slower than full search by up to 16 times.

This paper is organised as follows: a brief description of the data dependent algorithm, as well as the memory sub-systems, will be discussed in Section II and Section III respectively. Results from the comparison will be presented in Section IV and analysed in Section V. We conclude and discuss future work in Section VI.

II. Motion estimation algorithms

Full search is a data independent motion vector estimation algorithm. Typically, for each image block in a given frame

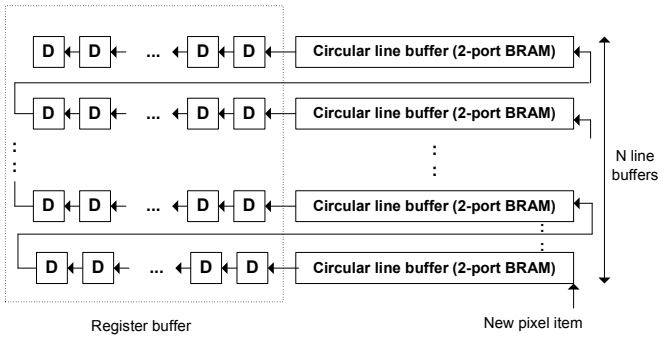

Fig. 1. Scratchpad memory, N: width of square search area.

of an image sequence, the best match is found within a rectangular area with pre-defined dimensions, centered around the same location in the adjacent frame, by minimising an error criterion for all search points in the search region. The sum of absolute difference error criterion (SAD) is used here because it requires minimal computational resources and produces motion vectors with reasonable quality [3].

Instead of considering all search points in a given region, the search area may be reduced by exploiting the high spatial correlation of motion vectors. In addition, the local variation of pixel values tends to be smooth. The data dependent algorithm, which is a simplifed version of [7], takes advantage of these two typical features for a given image block. Unlike full search, the pixel accesses are dependent on the image data and the surrounding motion vectors.

III. Memory sub-systems

For the full search algorithm, the memory access pattern is known at design time. Therefore, a scratchpad memory is used, as shown in Figure 1. It consists of a register buffer, where N^2 pixels are stored, as well as N circular pixel line buffers. To determine the best match, pixels in the register buffer are copied, in one cycle, to a two-dimensional shift register where SAD computation takes place. This ensures that the pixel addresses for SAD computation are static during run time. In addition, pixel data within the register buffer may be updated for the next reference block concurrently with motion vector estimation for the current block. The full search algorithm, implemented with the line buffering scheme, is a credible benchmark for comparison with the data dependent algorithm. Indeed, for a substantial search area of 29 by 29 pixels, it is faster than real-time performance for an image frame size of 144 by 176 pixels wide, which conforms with the Quarter Common Intermediate standard (QCIF) [9].

On the other hand, the memory access pattern of the data dependent algorithm is unknown at design time. Consequently, a custom parallel cache is used, as shown in Figure 2. The cache, shown within the dark grey region of the figure, consists of N sub-caches. Each sub-cache is a variant of a direct-mapped cache, implemented using on-chip block RAMs for tag and data storage as well as tag comparison logic. Potentially, up to N data items may be accessed from the

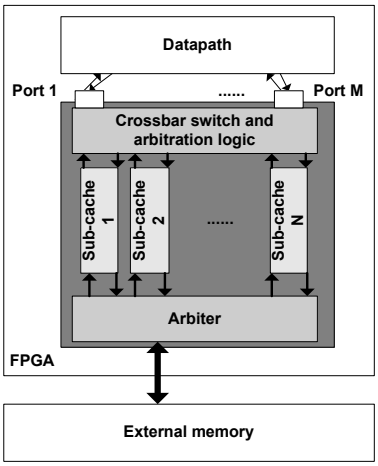

Fig. 2. Cache architecture [8].

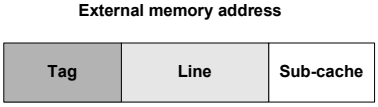

Fig. 3. Cache addressing scheme [8].

datapath in parallel. When the same sub-cache is accessed from two or more locations in the datapath, two semaphore-based arbiters are used to arbitrate accesses to the same target sub-cache and external memory in the event of multiple cache misses respectively.

The external memory address may be used to retrieve cached data items, as shown in Figure 3. The least significant bits of the address, labeled *Sub-cache*, are used to identify the sub-cache that the required address maps to. The more significant address bits, or *Line*, are used to determine the location of the data item within the sub-cache. The most significant address bits represent the tag, which is used to determine whether an access results in a cache hit or miss.

IV. Results

The objective of this work is to quantify the cost of data dependence experimentally, in terms of execution time and resource usage, by comparing the custom hardware implementations of the data dependent and full search motion estimation algorithms. Data re-use is exploited by using the cache and line buffering schemes described in Section III. The image frame and image block sizes used by both algorithms are 144 by 176 pixels wide and 4 by 4 pixels wide respectively [9]. The image sequence [10] describes a news reporting event with limited abrupt movements and no scene changes.

The RC300 board [11], which contains a Xilinx Virtex XC2V6000 FPGA [12], is used as the platform for comparison. Resources available on the FPGA include 33792 slices and 144 block RAMs. Two external synchronous SRAMs (SSRAM) are used to store image frames. Each SSRAM contains only one port of access and each access requires two cycles [13]. Handel-C is used for the description of both

TABLE I

EXECUTION TIME FOR DATA DEPENDENT AND INDEPENDENT ALGORITHMS WITH SEQUENTIALISED MEMORY ACCESSES. NIT: NUMBER OF REFINEMENT ITERATIONS, FTH: FULL SEARCH THRESHOLD, SAR: FULL SEARCH WINDOW SIZE.

Window size/pixels	Parameters of DD			FS /ms	DD /ms
	Nit	Fth	Sar/pixels		
11	4	31	11	71.6	118
17	2	8	17	235	155
19	2	8	19	317	180
21	4	8	21	398	221
25	4	6	25	606	318
29	4	5	29	862	457

algorithms. The SPM and cache, shown in Figures 1 and 2, are implemented using on-chip block RAMs and registers. The access time for block RAM access is one cycle, but logic overheads prolong access time to two cycles for the semaphore-based system which is the same as external memory access time. However, a reduction in overall cycle count occurs by parallelising accesses to the sub-caches. Dual-port block RAMs are used in the implementation of the circular line buffers in the SPM so that buffer data may be updated and read to the search area concurrently. The *N* by *N* search area and the two-dimensional shift register are implemented using register arrays so that an entire block of pixels may be read in one cycle.

In the first experiment, both algorithms are implemented without optimised memory sub-systems; the execution intervals of both algorithms are subsequently measured. Since the cycle count of the data dependent algorithm is dynamic, the cycle count is measured by implementing a hardware counter that runs in parallel with the motion estimation algorithm. In each case, the execution time is the product of the number of clock cycles and the maximum achievable clock frequency. For the second experiment, the SPM and cache are used to exploit data re-use for the data independent and data dependent algorithms respectively. The resource usage, in addition to the execution time, are measured. In both cases, the parameters of the data dependent algorithm are set such that the quality of the motion vectors produced are comparable with the motion vectors produced by full search. This is achieved by exhaustively computing the total SAD of all image blocks for several implementations of the data dependent algorithm with different parameters, and choosing the set of parameters with comparable SAD value to full search, and minimum execution time, for a given search window size.

Results from the first experiment will be presented in Section IV-A. The execution time and resource usage for both algorithms will be shown in Section IV-B.

A. Algorithms with sequentialised memory accesses

Table I shows the execution time for full search and the data dependent algorithm, for increasing search window size, in the absence of optimised memory sub-systems; accesses to memory data items are sequentialised. *Window size* indicates the width of one side of a square search window region. For a given reference block, a motion vector is first obtained through coarse estimation. This motion vector is then refined iteratively up to *Nit* times if a pre-determined SAD criterion, *Fth*, is not met. If the SAD criterion is still not met after refinement, full search is carried out on a search window which is *SAR* pixels wide.

TABLE II

EXECUTION TIME FOR DATA DEPENDENT AND INDEPENDENT ALGORITHMS WITH OPTIMISED MEMORY SUB-SYSTEMS.

Window size /pixels	Full Search		
	Cycle count $/10^5$	Clock speed /Mhz	Total /ms per frame
17	3.79	52.5	7.21
19	4.73	51.9	9.12
21	5.80	51.9	11.2
25	8.30	51.3	16.2
29	11.4	51.3	22.2
Window size /pixels	Data Dependent		
	Cycle count $/10^5$	Clock speed /Mhz	Total /ms per frame
17	36.39	32.4	113
19	40.69	31.5	129
21	49.64	31.9	156
25	71.13	31.5	226
29	99.09	30.8	322

The data dependent algorithm requires full search if a SAD threshold is not met. Therefore, when the search window size is 11 pixels wide, it is observed that the saving from avoiding full search is smaller than the overhead of carrying out data dependent search on top of full search. However, the savings in processing time grow with the search window size such that beyond a search window size of 17 pixels, a reduction of 47% in execution time occurs for a search window size of 29 pixels.

B. Motion vector estimation with memory sub-systems

Tables II and III shows the execution time and resource usage for both algorithms as the search window size increases. Unlike the situation where memory accesses are sequentialised, the execution time of the data dependent algorithm is larger than full search by an average of 14.4 times. Specifically, the cycle count and minimum clock period are substantially larger in the case of the data-dependent algorithm compared with full search. However, the resource usage of the cached designs does not increase with the search window size. In contrast, the memory resources required by Full Search increases with the search window size and consumes up to 38.2%, 55.4% and 73.9% more slices, block RAMs and flip flops than the data dependent algorithm respectively.

V. ANALYSIS

From the experimental results, it is seen that the introduction of the memory sub-system causes the full search algorithm to

TABLE III
RESOURCE USAGE FOR DATA DEPENDENT AND INDEPENDENT ALGORITHMS WITH MEMORY SUB-SYSTEMS

Search window size /pixels	Full Search		
	Slice count	Block RAM count	Flip flop count
17	5852	18	7000
19	5047	20	6699
21	5693	22	7926
25	7187	26	10783
29	9069	30	14145
Window Size /pixels	Data Dependent		
	Slice count	Block RAM count	Flip flop count
17	3905	10	2242
19	3881	10	2242
21	3880	10	2238
25	3898	10	2244
29	3904	10	2267

out-perform the data dependent algorithm. There are four main reasons for this occurence.

1) Algorithmic: the data dependent algorithm resorts to full search if the SAD threshold is not met. However, compared to the SPM, the cache is not optimal for full search.
2) Dependencies: dependencies occur between the steps in the data dependent algorithm and other image reference blocks because of the re-use of motion vectors. These data dependencies make it difficult to pipeline the design. Consequently, hardware parallelism cannot be fully exploited.
3) Architecture: semaphore arbitration is used to regulate accesses to sub-caches and external memory. Clock period degradation occurs because the delay of the semaphore arbitration system increases with the number of cache ports.
4) Variability in memory access time: the number of cycles required to retrieve a data item is dependent on whether a cache hit or miss occurs. This variability makes it difficult to optimally pipeline memory access and computational operations.

The first two reasons are responsible for the large discrepancy in cycle count between the data dependent and full search algorithms. Due to the third reason, the maximum clock frequency for the cached design is substantially lower than full search.

In the process of determining the optimum cache parameters, it is found that the slice usage does not vary significantly with the number of cache lines due to the granularity of the platform: the block RAMs which are used have substantial capacity, affording an increasing number of cached pixel data items to be packed within the same block RAM without involving additional logic and memory resources.

VI. CONCLUSION

In summary, data dependence in motion vector estimation has been found to be extremely expensive in terms of hardware performance. In the absence of data re-use exploitation, the data dependent algorithm is up to 47% faster than full search. However, full search becomes faster than the data dependent algorithm by up to 16 times when data re-use is exploited using a memory sub-system that is able to cope with dynamic memory accesses, in the case of the data dependent algorithm. Further, clock period degradation and the granularity of the platform are found to have a significant impact on performance.

Future work includes developing ways to automatically generate a memory sub-system for a given design, and to tune the parameters of the memory sub-system to achieve optimal performance and resource usage.

ACKNOWLEDGMENT

The authors would like to thank the EPSRC (EP/C549481/1) for its support in this work.

REFERENCES

[1] T. Koga, K. Linuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," in *Proceedings of the National Telecommunication Conference*, New Orleans, USA, 1981, pp. 438–442.
[2] J. Y. Tham, S. Ranaganath, M. Ranaganath, and A. Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IEEE transactions on Circuits and systems for video technology*, vol. 8, pp. 369–377, August 1998.
[3] P. Khun, *Algorithms, Complexity, Analysis and VLSI Architectures for MPEG-4 Motion Estimation*. Boston, USA: Kluwer Academic Publishers, 1999.
[4] Y.-K. Ko, H.-G. Kim, J.-W. Lee, Y.-R. Kim, H.-C. Oh, and S.-J. Ko, "New motion estimation algorithm based on bit-plane matching and itsVLSI implementation," in *Proceedings of the IEEE Region 10 Conference*, Chiang Mai, Thailand, 1999, pp. 848–851.
[5] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 3, pp. 148–157, April 1993.
[6] S. Yang, W. Wolf, and N. Vijaykrishnan, "Power and performance analysis of motion estimation based on hardware and software realizations," *IEEE Transactions on computers*, vol. 54, pp. 714–726, June 2005.
[7] J. Chalidabhongse and C. Kuo, "Fast motion vector estimation using multiresolution-spatio-temporal correlations," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, pp. 477–488, June 1997.
[8] S. Ang, G. Constantinides, W. Luk, and P. Cheung, "A flexible multiport caching scheme for reconfigurable platforms," in *Proceedings of Applied Reconfigurable Computing Workshop*, Delft, Holland, 2006, pp. 205–216.
[9] Webopedia. (2006) QCIF definition. [Online]. Available: http://www.webopedia.com/TERM/Q/QCIF.html
[10] British Broadcasting Corporation. (1997) Experimental image sequence. [Online]. Available: http://www.doc.ic.ac.uk/∼sa4/person.avi
[11] Celoxica. (2006) RC300 datasheet. [Online]. Available: http://www.celoxica.com/techlib/files/CEL-W0504181A36-257.pdf
[12] Xilinx. (2005) Virtex 2 datasheet. [Online]. Available: http://www.xilinx.com/bvdocs/publications/ds031.pdf
[13] Celoxica. (2004) RC300 manual. [Online]. Available: http://www.celoxica.com/techlib/files/CEL-W04110816VG-316.pdf