A Low Complexity Hardware Architecture for Motion Estimation

Daniel Larkin, Valentin Muresan and Noel O'Connor
Centre for Digital Video Processing, Dublin City University, Dublin, Ireland
Email: {larkind, muresanv, oconnorn}@eeng.dcu.ie

***Abstract*— This paper tackles the problem of accelerating motion estimation for video processing. A novel architecture using binary data is proposed, which attempts to reduce power consumption. The solution exploits redundant operations in the sum of absolute differences (SAD) calculation, by a mechanism known as early termination. Further data redundancies are exploited by using a run length coding addressing scheme, where access to pixels which do not contribute to the final SAD value is minimised. By using these two techniques operations and memory accesses are reduced by 93.29% and 69.17% respectively relative to a systolic array implementation.**

I. Introduction

The ongoing global trend to shift multimedia applications from desktop to mobile platforms has encountered several technical hurdles: demanding real-time applications, low bandwidth mobile networks, and mobile device hardware (HW) limitations. The latter include low computational power, low memory capacity, short battery life and strict miniaturisation requirements. Therefore the computational complexity associated with modern video codecs such as H.264, is highly undesirable on mobile devices from a power consumption perspective. The greatest scope for power savings (10x-20x) occur at the algorithmic level, by using such techniques as exploiting the nature of the media processing operations to be accelerated (e.g. regularity, redundancy) [1].

Motion estimation (ME) is the most computationally demanding task within all video codecs. It is used to exploit the temporal redundancies in video sequences, by (typically) employing a block matching algorithm (BMA) to find the best match for a block of pixels in the current frame by searching in a reference frame. The similarity of a block match (BM) is evaluated using a distortion metric, of which the sum of absolute differences (SAD) is the most popular, due to its optimum trade off between complexity and quality [2]. The SAD formula for a 16×16 pixel macroblock (MB) is:

$$SAD\left(B_{curr}, B_{ref}\right) = \sum_{i=1}^{i=16} \sum_{j=1}^{j=16} \left|B_{curr}\left(i,j\right) - B_{ref}\left(i,j\right)\right| \quad (1)$$

Where B_{curr} is the block under consideration in the current frame and B_{ref} is the block at the current search location in the search frame. The reference block with the lowest value SAD is chosen for further processing.

This paper proposes an efficient low complexity HW architecture for motion estimation. To reduce the complexity overhead, binary block matching is employed in conjunction with a one-bit pixel preprocessing filter.

The rest of this paper is organised as follows: section II details related prior research. Section III proposes a new binary motion estimation routine which exploits early termination properties in the distortion metric calculation and exploits redundancies in the binary data with a run length coding (RLC) addressing scheme. Section IV details an associated hardware architecture. Section V details hardware synthesis results and power consumption estimates, whilst section VI draws conclusions about the work presented.

II. Related Research

There are numerous ways to reduce the complexity of the full search BMA. Fast heuristic search strategies such as the 3 step search, logarithmic search, diamond search and block based gradient descent have all been used to reduce the number of search locations [2]. From a hardware implementation perspective the generation of non regular addresses increases the control logic considerably. Also optimal motion vectors are not guaranteed. On the other hand fast exhaustive search strategies that employ such techniques as conservative SAD estimations [3] or early exit mechanisms [4] achieve the same results as the full-search ones, but reduce computation by skipping irrelevant candidate blocks [2]. Another option to reduce complexity is to use binary motion estimation (BME) approaches, which reduce the complexity contribution of the distortion metric by quantising 8 bit pixels to a binary representation [5] [6] [7] [8]. This greatly simplifies the SAD operation (eqn. 1) since the subtraction between the two processed binary valued pixel reduces to a simple XOR calculation (eqn. 2) with the absolute function inherent.

$$SAD\left(B_{curr}, B_{ref}\right) = \sum\sum \left(B_{curr}\left(i,j\right) \oplus B_{ref}\left(i,j\right)\right) \quad (2)$$

Using BME as a preprocessing stage is proposed in [5] to discount poor BM prior to a full resolution ME. A pixel is quantised to a binary value based upon the value of the pixel relative to the mean of an NxN surrounding pixel block. Edge filtering is used in [6] to binarise the input pixels prior to doing a full search BME. However, in sequences with an absence of distinct edges, this approach can result in poor motion vectors. Natarajan et al presents a 2D systolic array BME hardware architecture, which employs a 17x17 convolution-based 1-bit transform [7]. A BME architecture is proposed in [8], which uses a hierarchical search strategy. In previous BME research no attempts have been made to optimise the processing element (PE) datapath. We will present

two redundancies within the datapath and propose solutions to exploit them. This work assumes binarisation of the texture has already been completed.

III. Exploiting BME Redundancies

A. Early SAD Termination

By employing early termination techniques the processing overhead can be reduced. Early SAD termination means that in certain block matches it is possible to cancel all further operations for that block because the accumulated partial SAD result is larger than the minimum SAD found so far within the search window. Further processing of that particular reference MB will only make the SAD result larger. Therefore the final SAD result will also be greater than the minimum. To exploit this feature, we propose that during each SAD processing operation, the partial SAD calculated to date is subtracted from a deaccumulation register, which initially holds the value of the best SAD value calculated thus far. If a sign change occurs during the deaccumulation step, there is no need to continue further processing since the current minimum SAD has already been exceeded. In order to allow cancellation, a partial SAD must be available. This presents a challenge for typical systolic array hardware architectures, due to the granularity of the calculation. The problem is overcome in [4] and our proposed architecture further extends the granularity of early termination through a pixel subsampling technique. This will be described in Section IV.

B. Exploiting Data Addressing Redundancies

Another characteristic of binary data that can be exploited to reduce computational overhead becomes apparent by observing that there are unnecessary memory accesses and operations when both B_{curr} and B_{ref} pixels have the same value. This happens because the XOR in eqn. 2 gives a zero result when both $B_{curr(i,j)}$ and $B_{ref(i,j)}$ have the same value. To minimise this effect, we propose using a RLC addressing scheme. However to use the RLC addressing the SAD calculation must be reformulated to the form given in eqn. 3 [9].

$$SAD = TOT_{ref} - TOT_{cur} + 2 \times DIFF_{cur} \quad (3)$$

Where TOT_{curr} is the total number of white pixels in the current MB $DIFF_{curr}$ is the number of white pixels in the current MB but not in the reference MB and TOT_{ref} is the total number of white pixels in the reference MB. Equation 3 is beneficial from a low power hardware perspective because:

- TOT_{curr} is calculated only once per search
- TOT_{ref} can be updated in 1 clock cycle
- Incremental addition of $DIFF_{curr}$ allows early termination if the current minimum SAD is exceeded
- By using run length coding to address the $DIFF_{curr}$ pixels, irrelevant data access is minimised.

The run length code is generated in parallel with the first match of the search step, an example of typical RLC is illustrated in fig. 1. It is possible to do this during the first match because SAD early termination is not possible at this point.

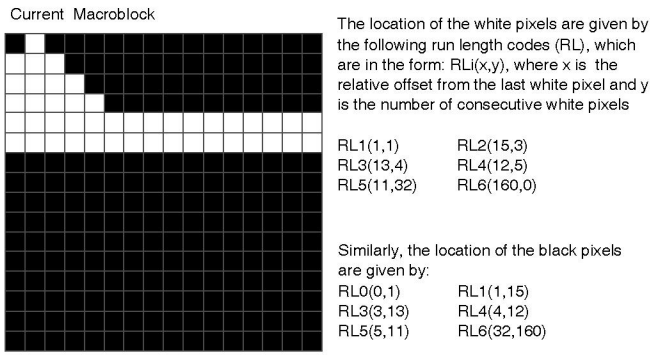


Fig. 1. Regular and Inverse RLC pixel addressing

The first match always takes $N \times N$ (where N is the block size) cycles to complete and this provides ample time for the run length encoding process to operate in parallel. After the RLC encoding, the logic would be powered down until the next current block is processed. In situations where there are fewer black pixels than white pixels in the current MB, it is possible to use the black pixels instead to calculate the SAD with eqn. 4. Fewer pixels translates into fewer operations to be completed, which has associated throughput and switching benefits.

$$SAD = TOT_{cur} - TOT_{ref} + 2 \times DIFF_{cur_{BLACK}} \quad (4)$$

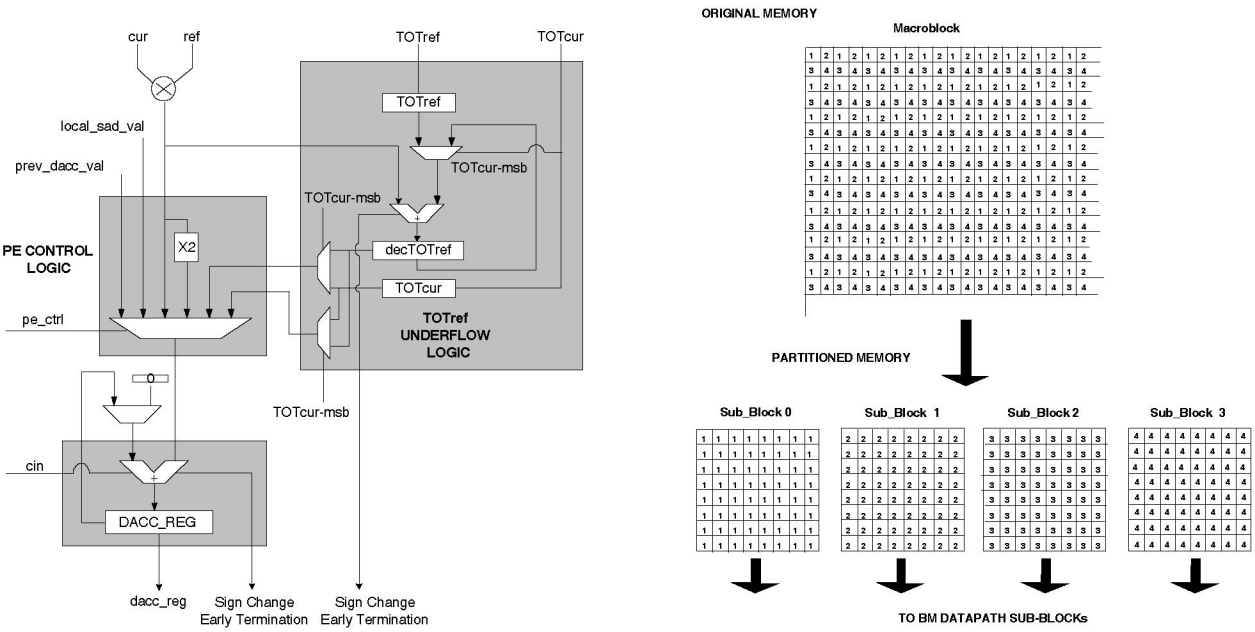
The location of the black pixels can be automatically derived from the RLC for the white pixels. Thus, by reusing the white pixel's RLC, additional memory is not required and furthermore the same SAD datapath can be reused with minimal additional logic. The choice of which mode to use is decided by the MSB of TOT_{cur}. To further minimise memory accesses when using the inverse run length mode, we propose decrementing a copy of the TOT_{ref} register (see fig. 2(a)) each time a white pixel in the reference block is accessed. If the copy of the TOT_{ref} register decrements to zero, no further contributions to the SAD are possible, since all the white pixels have been examined and early termination is possible.

IV. Architecture Design

The proposed architecture can be implemented with varying degrees of parallelism depending on the critical requirements (area, power, throughput, technology) of the final system. The basic PE will now be described, followed by a parallel architecture which uses 4 processing elements.

A. Basic RLC SAD Processing Element

Fig. 2(a) shows a simplified view of the SAD PE. At the first clock cycle the minimum SAD encountered so far is loaded into $DACC\_REG$. During the next cycle TOT_{curr} or TOT_{ref} is added to $DACC\_REG$ (depending if $TOT_{cur}[MSB]$ is 0 or 1 respectively). On the next clock cycle $DACC\_REG$ is deaccumulated by TOT_{ref} or TOT_{curr}. If a sign change occurs at this point the minimum SAD has already been exceeded and no further processing is required. If a sign change has not occurred the address generation unit (AGU) retrieves the next run length code from memory. If $TOT_{curr}[MSB] = 0$ the run



(a) RLC SAD Processing Element

(b) Pixel Subsampling

Fig. 2. RLC SAD processing element and Pixel Subsampling

length pair code is processed unmodified. On the other hand if $TOT_{ref}[MSB] = 1$, the inverse run length code is processed. In either case the processing results in an macroblock pixel address. This address is used to retrieve the relevant pixels from the reference MB and the current MB. The pixel values are XORed and the result is left shifted by one place and then subtracted from the $DACC\_REG$. If a sign change occurs, early termination is possible. If not, the remaining pixels in the current run length code are processed. If the SAD calculation is not cancelled, subsequent run length codes for the current MB are fetched from memory and the processing repeats.

When a SAD has been calculated or terminated early, the AGU moves the reference block to a new position. Provided a circular or full search is used TOT_{ref} can be updated in one clock cycle. This is done by subtracting the previous row or column (depending on search window movement) from TOT_{ref} and adding the new row or column.

B. 4xPE Block Matching Architecture

In order to exploit early termination, an intermediate partial SAD must be generated. This requires SAD calculation to proceed in a sequential manner, but this reduces throughput, and is not desirable for real time applications. To increase throughput, parallelism must be exploited. Our proposed parallel architecture can be seen in fig. 3. The architecture carries out motion estimation on one MB at a time. In the case of a 16x16 block match, the MB is split into four 8x8 blocks by using a simple pixel subsampling technique shown in fig. 2(b). Each of the four PE operates on one 8x8 block. The four parallel PE generate partial SAD values. The decision-making unit then uses these accumulated SAD values to make a SAD early termination decision. The early termination can occur at any point during the BM processing. If early termination does not occur in all 4 PE, it is necessary to examine the 4 PE SAD values and see if a new minimum SAD has been found, thus the update stage is invoked. This checks if the SAD is less than the minimum SAD encountered so far. The update logic consists of a single adder/subtracter and an accumulator. The update is carried out sequentially to reduce area.

Rather than reducing throughput by stalling the PE while the update state operates, the update stage can run in parallel with a new block match. Further SAD cancellations can occur in the next match without effecting the performance of the update logic. If the new match is cancelled then it would also cancel for the new updated minimum SAD value. If after accumulating the block level SAD values, the result is negative a new minimum SAD has been not found. The update can then stop processing and the logic is powered down. However if the accumulated block level SAD value is positive, this means that a new minimum SAD has been found. As a result, the block level minimum SAD and total minimum SAD values now need to be updated. In addition an adjustment must be made to the current block matches in progress. This is because that block match started deaccumulation from the old minimum SAD (MIN_SAD_{OLD}) rather than the new minimum SAD (MIN_SAD_{NEW}). Since $MIN\_SAD_{NEW} = MIN\_SAD_{OLD} - DACC\_REG$, an adjustment of $DACC\_REG$ is required at the PE. This value is stored in pe_i_dacc registers in the update block. This demonstrates that the update can occur in parallel with the next block match, which is of considerable benefit since it means the PE does not need to be stalled until the update has finished.

TABLE I

RLC BME_4xPE VERSUS CONVENTIONAL SYSTOLIC ARRAY BME

	Memory Accesses (1 bit pixels)		Operations (1 bit XOR & addition)		Clock Cycles	
Sequence	**2D SA [7]**	**BME_4xPE**	**2D SA [7]**	**BME_4xPE**	**2D SA [7]**	**BME_4xPE**
Akiyo	1.5206×10^9	4.5298×10^8	4.0170×10^9	2.6105×10^8	3.2252×10^7	6.8858×10^7
Hall Monitor	1.5206×10^9	4.8202×10^8	4.0170×10^9	2.7847×10^8	3.2252×10^7	7.3362×10^7
Foreman	1.5206×10^9	4.7137×10^8	4.0170×10^9	2.6942×10^8	3.2252×10^7	7.0999×10^7
Average	69.17% reduction		93.29% reduction		220.37% increase	

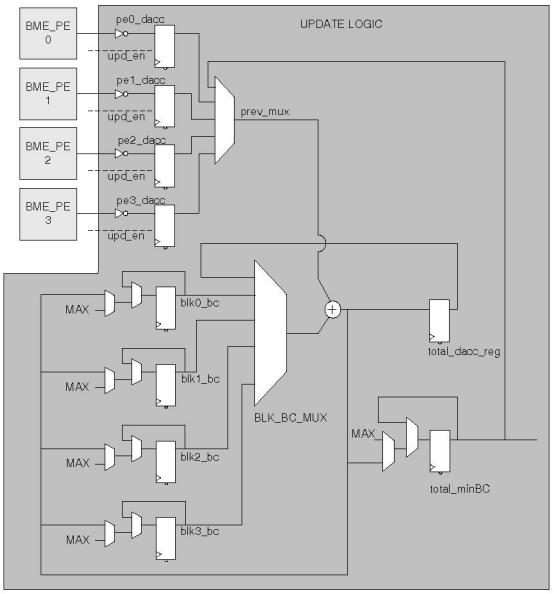


Fig. 3. 4xPE Block Matching Architecture

V. EXPERIMENTAL RESULTS

The motion compensated PSNR is dependant predominately on the choice of the binarisation filter, consequently PSNR will not be considered further in these results. The 4xPE design was captured using Verilog HDL. The design was targeted to a Xilinx Virtex 2 FPGA and also synthesised using a 90nm TSMC library characterised for low power. The results for the datapath can be seen in table II. Synplicity Pro and Synopsys Design Compiler were used for synthesis, whilst Xilinx XPower and Synopsys Prime Power were used to generate the power consumption figures. The ASIC implementation, due to the dedicated logic rather than configurable logic has smaller area and lower power than the FPGA implementation. From an area and power perspective direct comparisons with previous hardware implementations of BME is difficult since they do not quote these figures [7] [8]. For this reason and issues with normalisation across different semiconductor processes we have chosen to benchmark our implementation in terms of 1 bit pixel memory accesses, 1 bit operations and number of clock cycles (see table I). Using standard MPEG-4 test sequences with a BM size of 16x16 and a search window of -8/+7, experiments showed we achieved a 93% average reduction in the number of operations compared to a SA implementation [7]. This figure also compares favourable to the early SAD termination HW design presented in [4], which achieves 45% to 51% reduction in operations. Our improvement can be attributed to the subsampling and the use of RLC addressing for the binary data. A systolic array implementation cannot take advantage of RLC addressing and this leads to the concern that our architecture could suffer from input/output bandwidth inefficiencies. However this is not the case, as can be seen in table I, due to the early termination, on average 69.17% fewer 1 bit pixel memory accesses are required. The reduction in operations and memory accesses comes at the expense of reduced throughput compared to [7], that requires a constant 271 cycles per macroblock. This compares to our design, which after allowing for early termination, requires on average 598 clock cycles per MB. If throughput is essential our design scales to 16 PE, however the effectiveness of early termination is reduced.

TABLE II

BME 4XPE SYNTHESIS RESULTS

	Area	**Max Freq.**	**Power**
Virtex 2 FPGA	14,615 gates	120 MHz	12.951 mW
TSMC 90nm	10,117 gates	250 MHz	1.220 mW

VI. CONCLUSIONS AND FUTURE WORK

One concern with using BME is that for small BM sizes the quality of the motion vectors degrades. This along with more accurate benchmarking and research into binarisation filters, which have not been discussed, will form the basis of future work. Overall this paper has presented an efficient BME architecture, which reduces computational complexity through the use of an novel binary early termination SAD architecture which uses a RLC addressing scheme. Reducing the number of computations and memory accesses is of considerable benefit since it reduces dynamic power consumption in the datapath.

REFERENCES

[1] M. Pedram and J. M. Rabaey, *Power Aware Design Methologies*. Kluwer Academic Publishers, 2002.

[2] P. M. Kuhn, *Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation*. Springer, June 1999.

[3] V. Do et al., "A Low-Power VLSI Architecture for Full-Search Block-Matching Motion Estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 14, pp. 393 – 398, Aug. 1998.

[4] M. Takahashi et al, "A 60-MHz 240-mW MPEG-4 Videophone LSI with 16-Mb Embedded RAM," *IEEE J. Solid-State Circuits*, vol. 35, no. 11, pp. 1713–1721, Nov. 2000.

[5] Feng, Lo, Mehrpour, and Karkowiak, "Adaptive block matching motion estimation algorithm using bit plane matching," in *IEEE Int Conf Image Processing, Washington D.C., USA,*, vol. NA, 1995, pp. 496–499.

[6] M. Mizuki, U. Desai, I. Masaki, and A. Chandrakasan, "A binary block matching architecture with reduced power consumption and silicon area," in *Proc. IEEE ICASSP-96*, vol. 6, Atlanta, USA, 1996, pp. 3248–3251.

[7] B.Natarajan, V. Bhaskaran, and K. Konstantinides, "Low complexity block based motion estimation via one bit transform," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 4, pp. 702–706, Aug. 1997.

[8] J. H. Luo et al, "A Novel All-Binary Motion Estimation (ABME) with Optimized Hardware Architectures," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 8, pp. 700 – 712, Aug. 2002.

[9] D. Larkin, V. Muresan, and N. O'Connor, "An Effi cient Motion Estimation Hardware Architecture for MPEG-4 Binary Shape Coding," in *Irish Signals and Systems Conference*, Dublin, Ireland, Sept. 1-2, 2005.