Journal of Science and Technology
ISSN: 2456-5660 Volume 7, Issue 09 (November 2022)
www.jst.org.in *DOI:https://doi.org/10.46243/jst.2022.v7.i09.pp199 - 211*

Cluster Computing for Web-Scale Data Processing

M Venkataratnam, M. Basha, L. Hari Prasad
Assistant Professor[1] ,Associate Professor[2,3,]
Department of ECE
mvenkataratnam.ece@anurag.ac.in, mbasha.ece@anurag.ac.in,lhariprasad.ece@anurag.ac.in
Anurag Engineering College, Kodada, Telangana

To Cite this Article

M Venkataratnam, M. Basha, L. Hari Prasad , **Cluster Computing for Web-Scale Data Processing"**
Journal of Science and Technology, Vol. 07, Issue 09,November 2022, pp199-211

Article Info
Received: 24-10-2022 *Revised: 9-11-2022* *Accepted: 18-11-2022* *Published: 30-11-2022*

ABSRACT;

Power efficiency and real-time processing capability are two major issues in today's mobile video applications. We proposed a novel Motion Estimation (ME) engine for power-efficient real-time MPEG- 4 video coding based on our previously proposed content-based ME al- gorithm [8], [13]. By adopting *Full Search* (FS) and *Three Step Search* (TSS) alternatively according to the nature of video contents, this algo- rithm keeps the visual quality very close to that of FS with only 3% of its computational power. We designed a flexible Block Matching (BM) Unit with 16-PE SIMD data path so that the adaptive ME can be performed at a much lower clock frequency and hardware cost as compared with previousFS based work. To reduce the energy cost caused by excessive external memory access, on-chip SRAM is also utilized and optimized for paral- lel processing in the BM Unit. The ME engine is fabricated with TSMC 0.18 μm technology. When processing QCIF (15 fps) video, the estimated power is 2.88 mW@4.16 MHz (supply voltage: 1.62 V). It is believed to bea favorable contribution to the video encoder LSI design for mobile appli-cations.

key words: *content-based, motion estimation, power-eflcient, real-time, MPEG-4*

Introduction

Motion Estimation (ME) is supposed to consume as much as 70% of the total computational power of an encoder [1].Many fast algorithms have been developed to realize dras- tic speedup over FS such as *Three Step Search* (TSS) [2], *Four Step Search* (FSS) [3] and Diamond Search (DS) [4], etc. However, most of them are not suitable for hardware implementation due to the following two facts: 1) Many fastalgorithms require irregular memory access, which is not fa-vorable for hardware implementation. 2) A single algorithmcannot guarantee good visual quality for various kinds of video.

Adaptive algorithms like PMVFAST [5] and ASDS [6] emerged with an object to maximize the speed up over FS while minimizing the visual quality loss. This is gener- ally realized by dynamically modifying ME strategies for

different situations. However, like in most of the fast al- gorithms, memory access irregularity persists and the the- oretical speedup often fails to guarantee high efficiency in hardware. In addition, the sophisticated decision making mechanisms in those adaptive algorithms often involve com- plicated mathematical analysis, which can become cumber-some overhead. This is another tough problem on the way tosuccessful hardware implementation as few examples haveever been observed except for [7].

So far, FS is still dominant in hardware design. Variouskinds of parallel architecture are utilized for high efficiencywhile the drawbacks of huge power consumption and largecircuit size are inevitable. Hence when it comes to applica-tions like mobile visual communication, FS will no longerbe ideal. New efficient algorithms with high accuracy alongand practical hardware architecture will be highly necessary.In Sect. 2, we will briefly review one of our previous work, a content-based ME algorithm [8], [13], which ap-proximates FS with satisfactory accuracy, reducing the com-putation burden to only 3% to 4% of that of FS. It is the
target algorithm of our ME hardware design.

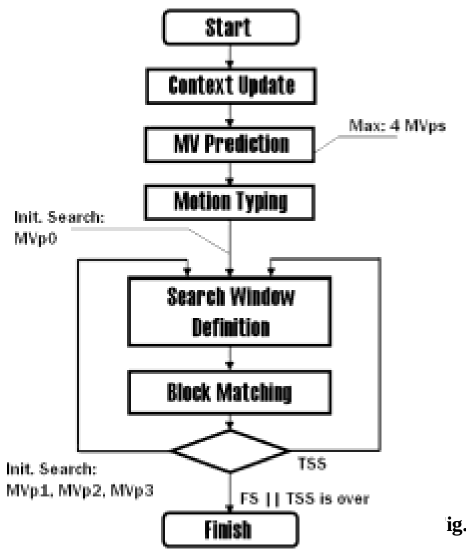
Then in Sect. 3, we will describe the hardware archi- tecture design of the proposed ME engine. We use a Hierar- chical Finite State Machine (HFSM) based Control Unit is designed to perform video contents analysis and execution control. We also designed a Flexible Block Matching Unit (BM Unit) so that the adaptive ME can be performed at a much lower clock frequency and hardware cost as compared with previous FS based work. To reduce the power caused by excessive external memory access, on-chip SRAM is also utilized and optimized for parallel processing with the 16- PE SIMD datapath in the BM Unit.

Section 4 is an evaluation report of the VLSI im- plementation. The ME engine is fabricated with TSMC 0.18 μm technology, taking up roughly 19.6 k gates as well as 6 k-bit SRAM. When processing QCIF (15 fps) video, the estimated power is 2.88 mW@4.16 MHz (supply volt- age: 1.62 V). Section 5 concludes this paper.

The Algorithm

The Content-Based Motion Estimation Algorithm

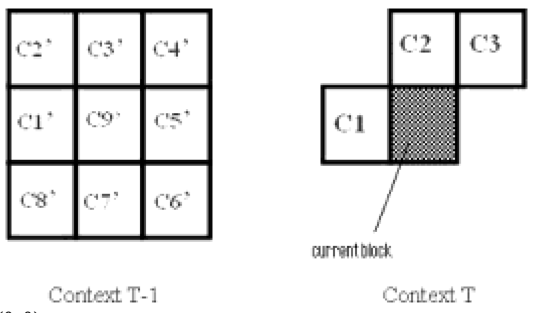
In our previous work [8], we proposed a new adaptive ME algorithm where FS with Adaptive Search Window (ASW) and TSS will be employed alternatively on macro block



ig. 1 Flow of the revised algorithm.

Table 1 Motion type decision.

T-1		T	Motion Type
Low	→	Low	CHAOS
High	→	Low	
Low	→	High	CRITICAL
High	→	High	SIMPLE



$\vec{V_0} = (0, 0)$

$\vec{V_1} = Median(\vec{V_1}, \vec{V_2}, \vec{V_3})$

$\widetilde{\vec{V_2}} = \vec{V_9}^{*}, \vec{V_3} = Median(\vec{V_5}^{*}, \vec{V_7}^{*}, \vec{V_8}^{*})$

Fig. 2 Context for MV distribution analysis.

(MB) level according to different video contents. The dis- tinction of video contents is done by classifying the motionof each MB into several types according to the motion vector (MV) distribution within its neighborhood. Figure 1 showsthe flow chart of the algorithm.

As illustrated in Fig. 2, *Context T-1* is defined in the reference frame and *Context T* in the current frame.

First, up to four initial search centers are predicted. ($\vec{V_k}$ and $\vec{V_k}^{*}$ stand for MVs in the two contexts, while $\tilde{\vec{V_k}}$ denotes the initial search center candidates.) Initial search (only check the predicted initial search center candidates) will find the one with minimum matching distortion and make it the search center.

Then motion typing starts from the analysis on the spa-tial correlation between the MVs in the same context. Two quantitive indicators δ_{T-1} and δ_T are used to describe the discrepancy of MVs in Context T-1 and Context T respec- tively (see (1) and (2)) while two thresholds, $Th1$ and $Th2$ are introduced for δ_{T-1} and δ_T respectively. The thresh- old values are obtained from experiments where a series of video sequences are used and all the possible values of $Th1$ and $Th2$ are tried. Finally the threshold values are decided as a compromise between matching accuracy and compu- tational complexity, as mentioned in [8], [13]. If either of δ_{T-1} and δ_T appears to be under the threshold the coherence of MVs in the corresponding context is classified as “High,” otherwise “Low.” In our algorithm,

$$\delta_{T-1} = Max_3(\|\vec{V_k}^{*} - \vec{V_9}^{*}\|) \ (k = 1, 2, 3, ..., 8) \qquad (1)$$

$$\delta_T = \quad \|\vec{V}_k - \widetilde{\vec{V}_1}\| \qquad (k = 1, 2, 3) \qquad (2)$$

$i=1$

: sum of the absolute differences of both x and y compo-nents of two vectors

Then the motion type will be decided via temporal cor-relation analysis between the two consecutive frames, which is shown in Table 1.

According to each of the motion types, the ME algo- rithm will be dynamically specified. In the case of “SIM- PLE,” FS tends to obtain an optimum of initial search cen- ter with extremely small search range, while in the case of “CHAOS,” TSS with large search range will be used to han-dle fast motion with higher efficiency than FS.

The reason why FS and TSS are adopted is given here. As mentioned earlier, fast algorithms always fail to see suc- cessful hardware implementation due to the irregular search pattern and consequential irregular memory access. Adap- tive algorithms seem to be more impractical because the multiple search patterns indicate redundant circuit when only one pattern is used at a time.

However, it is observed that FS and TSS can be exe- cuted in a similar manner except that:

1. the step sizes of scan in horizontal and vertical direc- tions are different (FS scans each search position with a step size of one pixel)
2. the numbers of iterations are different (TSS will work until the step size is reduced to 1)
3. the search window are different (In each iteration, TSS will update its search window, which is decided by the search center, search range and the position of the MB)

Hence it is natural to merge the initial search (only check the predicted initial search center candidates), FS and TSS into a configurable block matching process as described in Fig. 1.

Simulation and Evaluation

We implemented the proposed algorithm in C language and

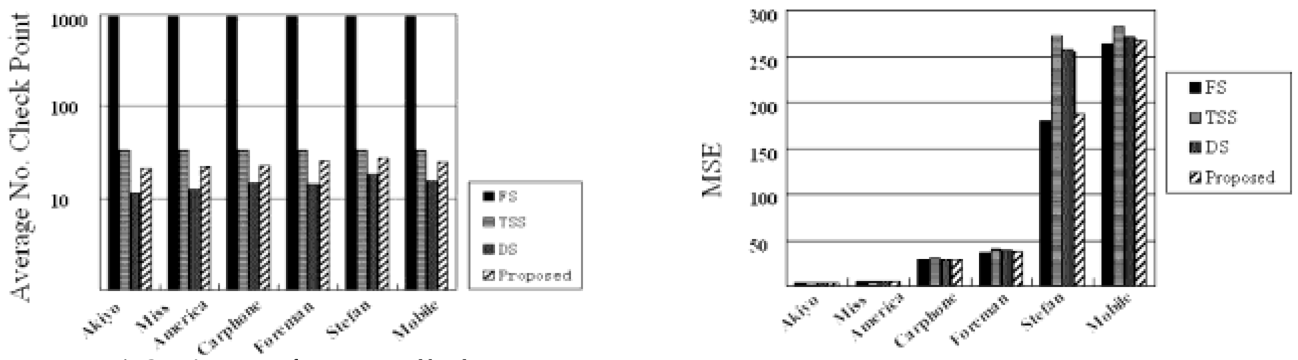


Fig. 3 A contrast of computational burden.

evaluated its performance in terms of *Average Number of Check Points* (ACP) per MB and *Mean Square Error* (MSE). We used such test sequences as follows:

- *Akiyo* (QCIF, 15 fps, 150 frames)
- *Miss America* (QCIF, 15 fps, 150 frames)
- *Carphone* (QCIF, 15 fps, 150 frames)
- *Foreman* (QCIF, 15 fps, 150 frames)
- *Stefan* (CIF, 30 fps, 150 frames)
- *Mobile* (CIF, 30 fps, 300 frames)

which are examples from simple contents to complicated contents that contain fast and detailed motion.

ACP is an indicator of computational complexity. As explained in [8], [13], in our algorithm the ACP is a function of the N_{simple}, $N_{critical}$ and N_{chaos}, i.e. the search ranges of “SIMPLE,” “CRITICAL” and “CHAOS” contents. A theo-retical worst case of ACP is $(2N_{critical}+1)^2+\delta$ per MB where δ denotes the extra check points caused by initial search. With $N_{critical} = 4$ and δ less than 4, ACP can be as many as 81 to 84. Although it hardly ever occurs as our experiments suggested, the architecture design and clock frequency deci- sion were based on the worst case computational complexity to ensure its real-time processing capability under various conditions. As illustrated in Fig. 3, although DS seems to be more efficient, its complicated search pattern causes great difficulty in hardware implementation while our FS + TSS based algorithm is free of such worries and it also demon- strated high efficiency as it constantly reduced the ACP to 3% to 4% of that of FS.

MSE describes the distortion of block matching. It is defined as (3), where M and N stand for the width and height of a frame in pixel, and C_{ij}, $C_i^*{}_j$ indicate the pixel in current frame and its match in the reference frame.

$$MSE = \frac{1}{M \times N} \sum^{N-1} \sum^{M-1} (C_{ij} - C^*_{ij})^2 \qquad (3)$$

Fig. 4 A contrast of visual quality.

involved, our algorithm can always remain the closest ap- proximation of FS in terms of block matching accuracy.

1. Hardware Architecture Design

With applications like mobile visual communication, the conflicting matters like real-time encoding capability and power-efficiency are considered key issues.

As we know, the power consumption of a LSI can be expressed by a function of operation voltage (*V*), circuit ca-pacity (*C*) and operation frequency (*f*). Normally higher *V*, *f* and bigger circuit size will lead to higher power con- sumption. When traditional algorithms like FS is adopted, the circuit size tends to be very large, and we will have to raise the V and f otherwise we cannot afford the heavy com-putation burden for real-time processing. In [9], methods to map ME algorithms into multiple *Processing Elements* (PE) are described, and in [10] and [11], a more powerful frame-level pipeline scheme is introduced which further increases the data-reuse, minimizing external memory access. How- ever, mapping FS to a systolic PE array is not an easy task, and the number of PEs is often to the 2nd order of the searchrange. As each PE consists of not only an ALU, but also reg- isters for pixel storage, the circuit size can be considerably large.

Moreover, due to the extremely huge data throughput required to meet the real-time processing requirements, we have to either spend a lot of clock cycles in data input with a limited external memory bandwidth or reduce the clock cycles by expanding the bandwidth, yet both ways lead to increased power consumption.

With the recent development of IC technology, it is get-ting more and more popular to implement local memory fordata-reuse inside a computation-intensive hardware device like ME engine since it can help to minimize external mem-ory access at a much smaller hardware cost than previous

Lower MSE indicates higher matching accuracy and higher visual quality.

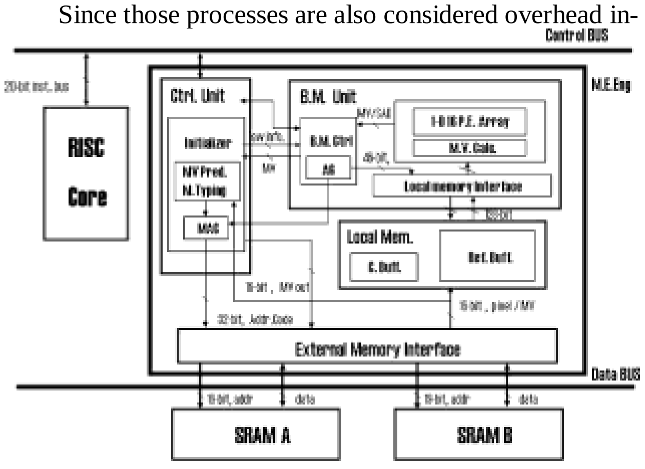
High efficiency always comes at a loss of matching ac-curacy. However, as shown in Fig. 4, for video sequences with various kinds of contents, even the typical sports videosequence *Stefan*, where complicated body movements are

approaches.

Hence in the architecture design of our ME engine, wesought such an approach as outlined below.

Hierarchical Finite State Machine (HFSM) for the original. video contents analysis and adaptive block matching control.

Since those processes are also considered overhead in-



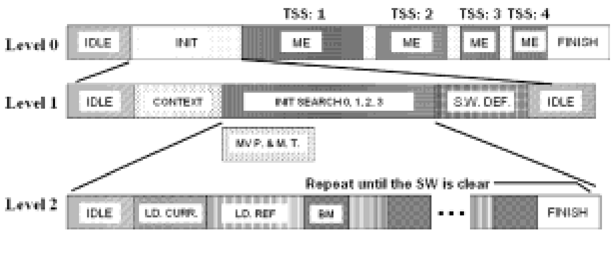


Fig. 5 Block diagram of proposed ME engine.

Fig. 6 Timing chart of the ME engine.

troduced by the algorithm, our aim was to minimize itin terms of both clock cycles and circuit size.

- *On-chip SRAM* for pixel reuse.
  It supports concurrent access to 16 pairs of pixels for SAD calculation, and the pixel reuse will help greatly reduce the external memory access as well as the re- lated power consumption.
- *16-PE SIMD Block Matching Unit* (BM Unit) with configurable parameters for FS and TSS.
  The proposed algorithm helped to greatly reduce the circuit scale of BM, and we adopted SIMD for parallel processing to further lower the clock frequency. The configurability for both FS and TSS will help avoid re-dundancy by circuit reuse.

The block diagram of the ME engine show in in Fig. 5.

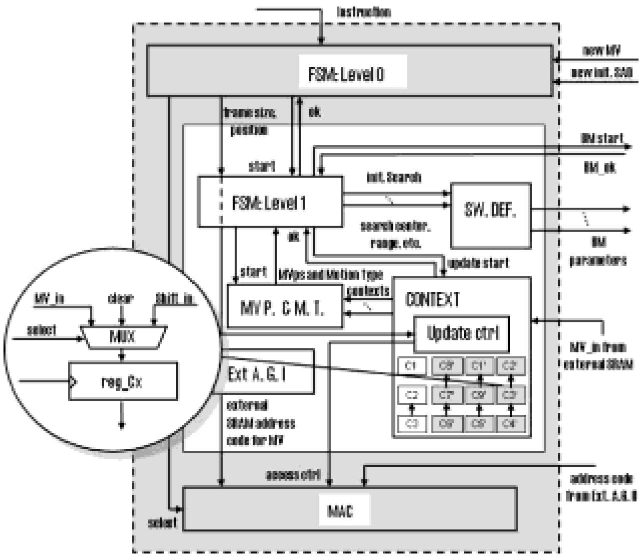


Fig. 7 HFSM in control unit.

It will be driven by a RISC processor through a 20-bit instruction bus, and the instructions will pass information to the ME engine such as frame size, frame index, block index and so on. The SRAM A and SRAM B are considered ex- ternal storage for pixels and MVs, since our algorithm also utilizes the MVs of previously processed MBs. They are connected with the ME Engine via a 16-bit bus, and thus during one read cycle, 2 pixels or 1 MV can be read into the ME engine (8 bits for each pixel or one component of a MV).

It should be noted that we might as well have pixel stor- age or external memory bus of other configurations as long as available. Naturally, varying the condition will have sig-nificant impact upon the design and the performance.

Hierarchical Finite State Machine (HFSM)

The execution control of our ME engine is realized via a HFSM as shown in Fig. 6, which depicts an example whenwe happen to have adopted TSS for the adaptive block matching of a MB.

Level 0 is the top level FSM, and we can see the TSS isfinished within four iterations (denoted by "ME," with stepsizes of 8, 4, 2 and 1) while "INIT" is a stage for initializa-tion, which is a unique process introduced by the proposed

Level 1 is a sub-FSM for "INIT," inside which Con- text Update ("CONTEXT"), MV Prediction ("MVP."), Motion Typing ("M.T.") and Search Window Definition ("S.W.DEF.") are included (see Fig. 1). The "S.W. DEF." will be performed before each of the four iterations to up- date the search range and search center.

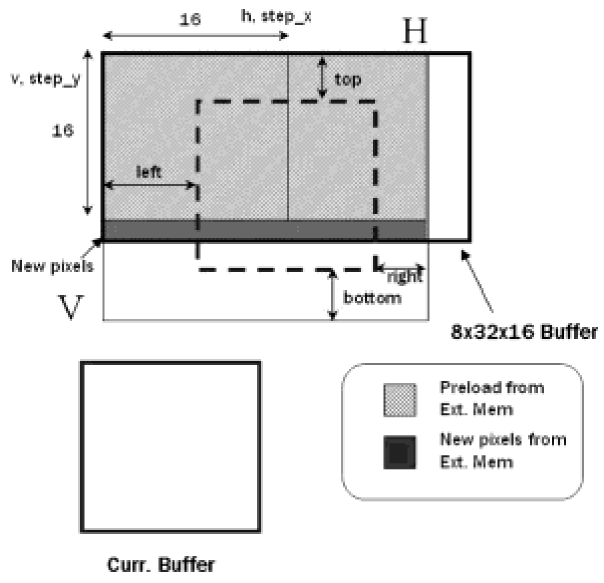
Both *Level 0* and *Level 1* are implemented in the "Con-trol Unit" of the ME engine, which is further described be- low.

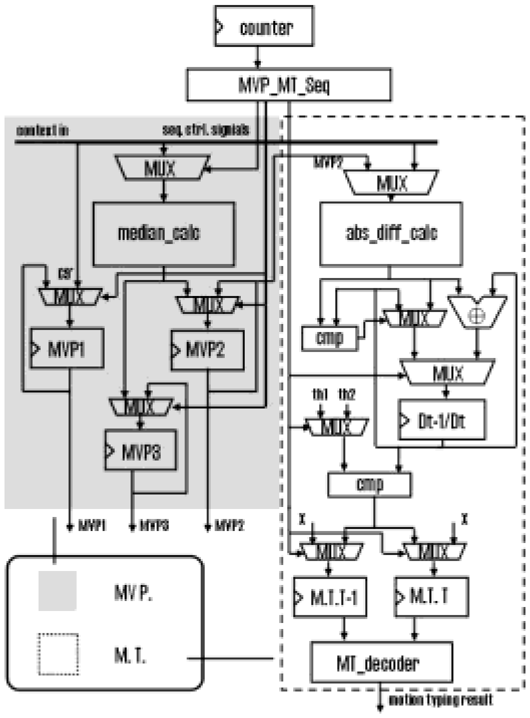
As shown in Fig. 7, the "FSM:Level1" is triggered by the "FSM:Level0" and it will first update the contexts for video contents analysis. In order to reduce the external memory access for old MVs, we devised a register file for the 12 MVs of both contexts as we found that the raster

scan mode makes part of the MVs in both contexts reusable. Hence each register of MVs, e.g. C_1 or C'_8, can either be updated with shift-in values or MVs from external SRAM.

In the "Update Control" in the "CONTEXT," a look-up ta-ble of "update codes" is prepared which decides the manner of update according to the position of the current MB with a simple decoding logic. It will guide a small FSM in the "Update Control" to scan and update all the MVs within 16cycles, with a minor overhead.

In Fig. 8, a detailed picture of "MVP. & M.T." is given.In fact, Fig. 6 also tells us that this process is executed





concurrently with “INIT SEARCH” to save clock cycles. Therefore we adopted a sequential style to save hardware by circuit reuse. The “MVP MT Seq” is a sequencer that is driven by a counter which sees that the whole process is fin-ished in a sequential manner within 13 cycles. There is only one “median calc” to calculate the median value of three input MVs, and one “abs diff calc” for the absolute differ- ence which is needed in “M.T.” There is a register “Dt-1/Dt” which is shared by δ_{T-1} and δ_T. They are decided in two consecutive cycles and finally the motion type is decided.

Level 2 is a sub-FSM for adaptive BM, including ini- tial search, FS, and TSS. It is implanted in the “B.M. Ctrl” module of “BM Unit” (see Fig. 5) which is described later.

On-Chip SRAM

We have a *Current Block Buffer* (CB) (“C. Buff.” as shownin Fig. 5) that consists of 16 pieces of 8 bits 16 words SRAM, and a *Reference Area Buffer* (RB) (‘Ref. Buff.’ as shown in Fig. 5) that consists of 16 pieces of 8 bits 32words SRAM.

The pixels of the current block will be loaded into the CB during the first initial search and will be reused. One row of pixels in a MB (8 bits for each pixel) can be accessed within one clock cycle.

The mapping of RB is shown in Fig. 9. and Fig. 10. We adopt a 2-D Modular Addressing Scheme here and a maxi- mum search window of 32 32 pixels can be dynamically accommodated in the RB. **Fig. 9** Search window and reference area buffer.

X

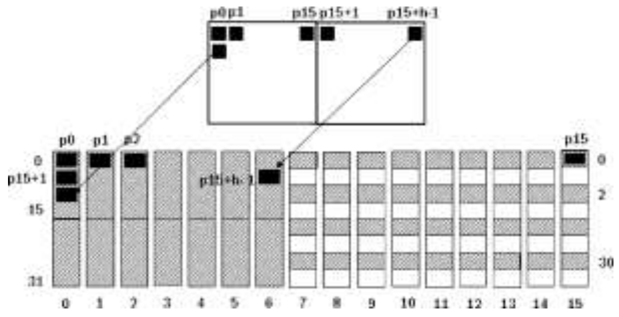


Fig. 10 Reference area buffer mapping.

BM Unit

Now that we know the only difference between FS and TSS is step size in the scanning of search positions in both hor- izontal and vertical direction and the number of iterations, both algorithms can be realized with the same hardware ar- chitecture only by varying variables such as current block position, search center, size of reference area, and step size,etc.

Figure 11 depicts the architecture design of the BM Unit, in which the *Level 2* FSM is nested, as the execution controller with those configurable parameters listed above. It will first load the current block and reference area pixels (“LD CURR” and “LD REF” as shown in Fig. 6) as long as necessary, with the “Ext. A.G. II” generating addresses of pixels in the external SRAMs, and the “Int. A.G.” generat- ing address for the on-chip SRAMs. In the address genera- tors, multipliers are absent since the calculation of address issimplified and can be done with only addition and shifting.

It should be noted that the *addresses* here stands for a kind of address code, which will be decoded in the *Exter- nal Memory Interface* (Ext. Mem. I/F) or the *Local Memory*

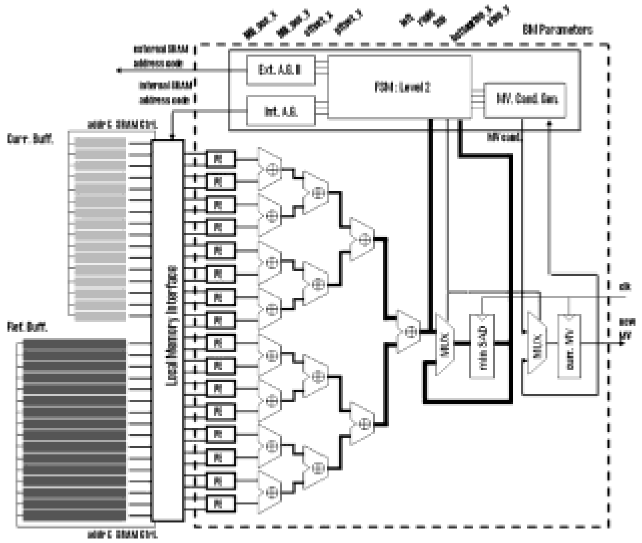


Fig. 11 Architecture view of the BM unit.

Interface (Loc. Mem. I/F). In the External Memory Inter- face, we prepared look-up tables for the generation of true addresses, in order to avoid complicated calculation and re-duce the circuit size.

During “BM” (also see Fig. 6), the BM Unit will try to search all the points in the part of search window that is cur-rently in the RB, with specified step sizes (FS: 1; and TSS:8, 4, 2 or 1). Again, “Int. A.G.” will generate addresses of pixels in both CB and RB. As illustrated in Fig. 10, the 16 pixels read from RB may not be in the natural order as search positions vary in the horizontal direction, and we de-vised a rotator inside the Loc. Mem. I/F which will re-orderthe reference pixels.

We are using the 16-PE SIMD architecture with *AdderTree* (AT) for the calculation of SAD and decision of MV. Compared with intensive systolic PE-array based previous work, the design is greatly simplified as we cancelled the distributed register-based pixel storage and complicated data path. Since we can access one row of pixels from both cur-rent block and the reference block from local memory, we can find the SAD of one position within 16 clock cycles.

Evaluation

As mentioned earlier, a FS based approach may guarantee best visual quality but generally requires huge hardware andhigh clock frequency. For example, [10] and [11] use 256 and 1089 PEs respectively, which makes it difficult to avoidhigh power cost.As for hardware implementation of fast al-gorithms, we noticed a most recent work where a 4-way pipelined architecture for TSS was proposed [12]. Table 2 is a brief contrast between this approach and ours.

[12] devised four parallel pipelines to perform TSS concurrently and thus a 64-bit input port is needed to feed four pairs of current/reference pixels from the exter- nal SRAM. To handle the 4-way parallel processing, each PE is actually an equivalence of 4 PEs used in our design. **Table 2** Contrast between 4-way TSS and our approach.

	Input port	PE #	Cycle #	Search range	On-chip SRAM
[12]	64b	9	337	[−7,+7]	0
Proposed	16b	16	432	[−7,+7]	6 kbit
			576	[−15,+15]	

Table 3 Summary of VLSI implementation.

Technology	TSMC 0.18 μm
Core size	1.99 mm × 1.99 mm
No. pins	73(without Power and Ground)
Gate count	19.6 k
SRAM	CB: 2 k-bit (8 bits × 16 words × 16 pcs)
	RB: 4 k-bit (8 bits × 32 words × 16 pcs)
Clock frequency	QCIF(15 fps): ≥ 4.16 MHz
	CIF(30 fps): ≥ 33.3 MHz
Supply voltage	1.6v

We should note that the overhead of data loading is excluded from the estimated cycle numbers here. Although their de- sign does not include on-chip SRAM, the data partitioning of the frames, which is a crucial point in their method and may as well require some pixel buffers, is not considered inthe total cycle number estimation. In addition, Our BM Unitcaconfigured to perform TSS with a maximum search range of [15, +15], which is more practical in case of fast motion.

2. VLSI Implementation

VLSI Implementation Result –

Our ME engine is implemented with TSMC 0.18 μm 6- metal CMOS technology. Table 3 is a summary of the VLSI implementation result. Since our target is low power mo- bile applications, the clock frequency is a very crucial issue. According to our calculation, in the worst case mentioned earlier in Sect. 2.2, the required clock cycle number is 2342 per MB. While in another case when only TSS is used, it becomes 2524 per MB due to more complicated data prepa-ration. Hence, the minimum clock frequency of 4.16 MHz for QCIF and 33.3 MHz for CIF will be adequate for all thecases. Figure 12 is a photograph of the chip.

Evaluation

In Fig. 13, a contrast of circuit scale in terms of gate count between all the major modules in the ME engine is given aswell as their shares of the estimated total power consump- tion. It shows that the on-chip SRAM takes up about 83% of the total area, which is equivalent to 80 k gates.

The "Ctrl Unit," which contains some special circuits for the overhead process introduced by our algorithm, takesup 7 k gates, roughly the same with the 16-PE SIMD "BM Unit." Based on the analysis in Sect. 3, we can say that even with the hardware overhead introduced by our adaptive al- gorithm, the proposed ME engine is still superior to previ-

ous FS and TSS based works in terms of circuit size, for thesame performance.

We also performed power consumption estimation with Synopsys's Power Compiler. The results shows the aver- age power consumption can be as low as 2.88 mW when our ME engine is working with a 1.6 V supply voltage, process-ing QCIF (15 fps) video sequences at 4.16 MHz. For CIF(30 fps) video, the minimum average power consumption is

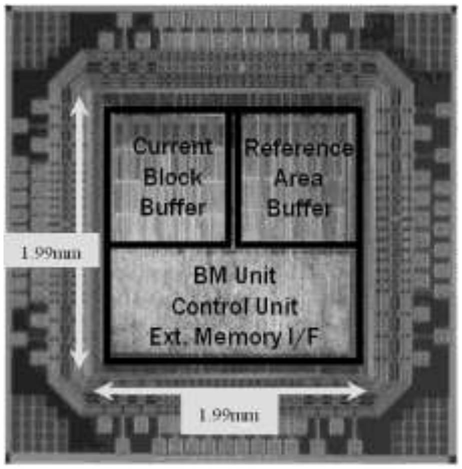


Fig. 12 A photograph of the chip.

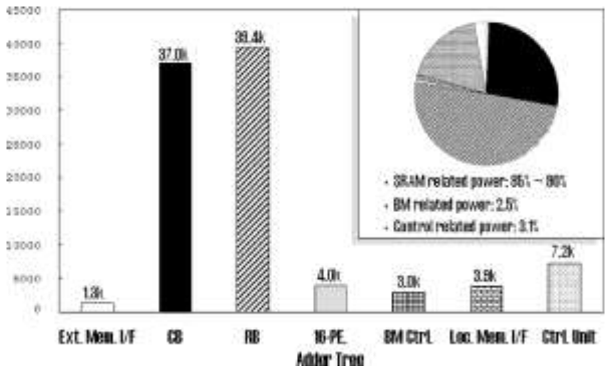


Fig. 13 Contrast of area/power consumption between major modules.

19.50 mW at 33.3 MHz.

As indicated by Fig. 13, on-chip SRAMs take up over 85% of the total power, of which "Ctrl Unit" and "BM Unit"only account for a minor portion. It should be noted that im-provements on the on-chip SRAM design will have great impact on the overall power consumption. For example, since our design will generally operate at a very low fre- quency, some customized SRAM with much lowered I/O drive than normal SRAMs will help significantly reduce thepower consumption. Increasing the external SRAM bus may also help to decrease the clock frequency, as another ap- proach of low power design.

Table 4 is a comparison between some related works done in recent years and ours. [14], [16] and [17] are FS based design. They have made an effort to enhance the flex-ibility but it is restricted to search range modification and initial motion vector prediction. [15] is a processor-based design which provides far better flexibility and it adopts fastalgorithms to reduce the computational complexity and as aconsequence, reduce the hardware cost.

As indicated by the comparison result, our design ap- pears to be more suitable for mobile applications as it suc-cessfully achieved flexible block matching with FS and TSSat far reduced hardware/energy cost.

3. Conclusions

We designed a content-based ME Engine based on our origi- nal adaptable ME algorithm. Simulations confirmed that the proposed algorithm can constantly reduce the computation burden to about 3% to 4% of that of FS with satisfactory vi-sual quality compared with some other fast algorithms for aseries of video sequences.

In the hardware architecture design, the Control Unit isdesigned based on a HFSM, with minimized hardware cost and an overhead of about 16 cycles for context update only.We introduced on-chip SRAMs for pixel data reuse, and theexternal memory access is successfully reduced. The BM Unit with configurable parameters can realize both FS and TSS with its flexible 16-PE SIMD architecture.

As the VLSI implementation results shows, the pro-

Table 4 VLSI implementation results comparison.

comparison	[14](CIF-ASIC)	[15]	[16]	[17]	Ours
Algorithm	FS	programmable Predictive DS	FS	FS	FS/TSS, dynamically configurable
Process/FPGA model	0.25 μm 2.5v	0.18 μm 1.8v	Xlinx SpartanII XC2S50	0.25 μm	0.18 μm 1.6v
Gate count/ Chip area	29 k	62 k	10 k	3.19×3.19 mm^2	19.6 k 1.99×1.99 mm^2
On-chip memory	9 kbit DPRAM	35 kbit ROM +3.4M SRAM	10.7 kbit	29.78 kbit	6 kbit SRAM
Clock frequency	18 MHz (QCIF)	67 MHz	8.2 MHz(QCIF) 49 MHz(CIF)	100 MHz	4.16 MHz 33 MHz(CIF)
Performance	QCIF/CIF, with search range [−16,+15]	—	QCIF/CIF, with search range [−7,+8]	720×576, with search range [−16,+15]	QCIF/CIF, with search range [−15,+15]
Power	42 mW(QCIF) 170 mW(CIF)	452 mW	36.79 mW(QCIF) 168.62 mW(CIF)	939 mW	2.88 mW(QCIF)

posed ME Engine can operate at 4.16 MHz for QCIF(15 fps) and 33.3 MHz for CIF (30 fps) real-time encoding at a much more reduced hardware cost of 19.4 k gates and6 k-bit on-chip SRAM (with TSMC 0.18 μm technology), compared with some previous work. The minimum power consumption for QCIF is as low as 2.88 mW, and is thusly considered a favorable contribution to the real-time video MPEG-4 encoder LSI design for mobile applications.

Acknowledgment

This work was partly supported by the funds of the MEXT via Kitakyushu innovative cluster projects.

References

P.M. Kuhn, G. Diebel, S. Hermann, A. Keil, H. Mooshofer, A. Kaup,
R. Mayer, and W. Stechcle, "Complexity and PSNR-comparison of several fast motion estimation algorithms for MPEG-4," Proc. Applications of Digital Image Processing XXI, SPIE, vol.3460, pp.486–499, San Diego, July 1998.
T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion- compensated interframe coding for video conferencing," Proc. Nat.Telecommunications Conf., pp.G 5.3.1–G 5.3.5., Nov./Dec. 1981.
L.M. Po and W.C. Ma, "A novel four-step search algorithm for fast blockmatching," IEEE Trans. Circuits Syst. Video Technol., vol.6, no.3, pp.313–317, June 1996.
J.Y. Tham, S. Ranganath, M. Ranganath, and A.A. Kassim, "A novel unrestricted center-biased diamond search algorithm for block mo- tion estimation," IEEE Trans. Circuits Syst. Video Technol., vol.8, no.4, pp.369–377, Aug. 1998.
A.M. Tourapis, O.C. Au, and M.L. Liou, "Predictive motion vec- tor field adaptive search technique (PMVFAST)—Enhancing block based motion estimation," Proc. SPIE, Visual Communications andImage Processing, vol.4310, pp.883–892, Dec. 2000.
T. Sappasitwong and S. Aramvith, "Adaptive asymmetric diamond search algorithm for block-based motion estimation," International Conference on Digital Signal Processing, vol.2, pp.1–3, July 2002.
J.K. Choi, N. Togawa, M. Yanagisawa, and T. Ohtsuki, "VLSI archi- tecture for a flexible motion estimation with parameters," Proc. IEEE Conf. ASP-DAC 2002 on VLSI Design, pp.452–457, Jan. 2002.
S. Li, Y. Jiang, T. Ikenaga, and S. Goto, "Content-based motion esti- mation with extended temporal-spatial analysis," Proc. IEEE Conf.Midwest Symposium on Circuit and Systems, vol.2, pp.461–464, July 2004.
S. Dutta and W. Wolf, "A flexible parallel architecture adapted to block-matching motion-estimation algorithms," IEEE Trans. Circuits Syst. Video Technol., vol.6, no.1, pp.74–86, Feb. 1996.
S. Kittitornkun and Y.H. Hu, "Frame-level pipelined motion esti- mation array processor," IEEE Trans. Circuits Syst. Video Technol., vol.11, no.2, pp.248–251, Feb. 2001.
L.C. Liu, J.C. Chien, H.Y.H. Chuang, and C.C. Li, "A frame- level FSBM motion estimation architecture with large search range," Proc. IEEE Conf. Advanced Video & Signal Based Surveillance, pp.327–333, July 2003.

S.T. Jung and S.S. Lee, “A 4-way pipelined processing architec- ture for three-step search block-matching motion estimation,” IEEE Trans. Consum. Electron., vol.50, no.2, pp.674–681, May 2004.
S. Li, Y. Jiang, T. Ikenaga, and S. Goto, “Content-based motion es- timation with extended temporal-spatial analysis,” IEICE Trans. Inf.& Syst., vol.E88-D, no.7, pp.1561–1568, July 2005.
L. Fanucci, L. Bertini, and S. Saponara, “Programmable and low power VLSI architecture for full search motion estimation in mul- timedia communications,” Proc. IEEE Conf. Multimedia and Expo, vol.3, pp.1395–1398, 2000.
M. Abbas, B. Talha, S. Khan, and A. Abbas, “A motion estimation chip for block based MPEG-4 video applications,” Proc. IEEE Conf. INMIC 2003, pp.253–257, Dec. 2003.
R. Gao, D. Xu, and J.P. Bentley, “Reconfigurable hardware imple- mentation of an improved parallel architecture for MPEG-4 motionestimation in mobile applications,” IEEE Trans. Consum. Electron., vol.49, no.4, pp.1383–1390, Nov. 2003.
W.-F. He, Z.-G. Mao, Z.-Q. Gao, and Y.-Z. Ye, “VLSI implementa- tion of full pixel motion estimation processor for MPEG-4 AS pro- file,” Proc. IEEE Conf. Solid-State and Integrated Circuits Technol- ogy, vol.3, pp.1633–1636, Oct. 2004.