Reconfigurable Hardware Implementation of an Improved Parallel Architecture for MPEG-4 Motion Estimation in Mobile Applications

R. Gao, D. Xu and J.P. Bentley

ABSTRACT — *A reconfigurable hardware implementation of a high-parallel architecture for MPEG-4 motion estimation is proposed in this paper. It possesses the characteristics of low power dissipation and low cost, thus primarily aiming at video-based mobile applications. The architecture employs a dual-register/buffer technique to reduce preload and alignment cycles and a high-parallel pipeline to reduce power consumption of redundant memory access. As an example, a content-based full-search block-matching algorithm has been mapped onto this architecture using a 16-PE array. This has the ability to calculate the motion vectors of 20fps QCIF video sequences in real time at 8.2 MHz clock rate with 36.76mw power dissipation using a Xilinx Spartan II FPGA.*

Index Terms — MPEG-4, motion estimation, VLSI, full-search.

I. INTRODUCTION

With the latest development of wireless systems, various conventional multimedia services can be provided on mobile platforms e.g. real-time video/image transmission, video-on-demand, email/ web browsing and so on. Among these, real-time video-based applications play an important role of mobile multimedia service [1, 2]. However, due to the inherent data intensity of video sequence, compression techniques are required to reduce the video size. This is mainly achieved by reducing spatial and temporal redundancies within video streams. Generally, the motion estimation (ME) technique is introduced to explore temporal redundancies, which are the main concern of video compression. Since ME operations can take up to 80% of the computational burden of a complete video compression procedure, it is the most important component in real-time video applications [3]. Many VLSI architectures for ME have been proposed for video compression. However, most of them target MPEG-1/2 video coding applications, such as videophone, video conferencing, video broadcasting, etc. These architectures are not particularly suitable for mobile and low power applications [4]. In this paper, a low-power FPGA implementation of a high parallel architecture [5] is presented. The proposed ME architecture and its implementation aim at low-power, low-cost mobile applications. It achieves the features of low power by adopting high data utilisation parallel pipeline architecture [3] and low cost by using FPGA implementation. In addition, three groups of buffers are integrated into the architecture for storage of current block, search range and temporary sum absolute difference (SAD), respectively, in order to reduce the unnecessary memory access. Moreover, dual-register processing elements (PE) are adopted to accelerate ME calculation.

The rest of this paper is organised as follows. In section 2, ME algorithms are briefly presented. In section 3, the proposed VLSI architecture is described in detail focusing on dual-register PE architecture and buffers. Section 4 analyses performance parameters in terms of minimum required clock rate and minimum memory bandwidth. FPGA implementation results and comparisons with other architecture are given in section 5. Finally, conclusions are drawn in section 6.

II. MOTION ESTIMATION ALGORITHMS

Recently, the MPEG-4 video coding standard, which features high compression rate, high interactive and object-oriented compression [6, 7], has been introduced to wireless multimedia transmission [1, 2]. It adopts block-matching algorithms with alpha binary plane to achieve motion estimation [3, 8]. Fig. 1 illustrates the principle of the block matching motion estimation technique. First, the video frames are segmented into $N \times N$ non-overlapping rectangular blocks. Every block within the current frame is matched to the corresponding blocks (candidate blocks) within a given *search range* on the previous frame. A *matching criterion*, or *distortion function*, that measures the similarity between the current block and candidate block is calculated. Then, a *motion vector* to the position of the candidate block, which has the minimum measurement with the current block, is generated to replace the real movement of the objects in a compressed video stream [3]. Thus, the temporal redundancy within a video sequence is reduced.

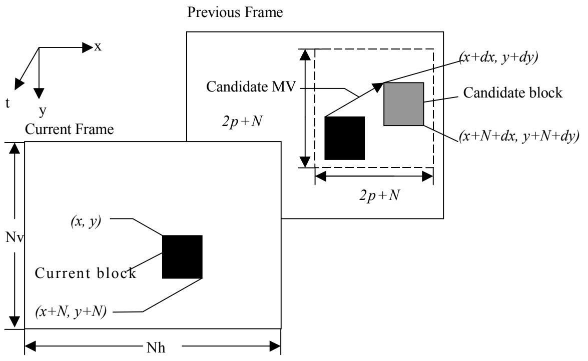

Fig. 1. Block matching motion estimation

A. Full-search block-matching algorithm

Because of its low distortion and regular data flow, the full-search block-matching algorithm has been widely used in

Contributed Paper
Manuscript received July 11, 2003

motion estimation. In this algorithm, the current block located at the pixel *(x,y)*, as shown in the Fig. 1, is matched to every candidate block within a *(2p+N-1)×(2p+N-1)* search window, where *[-p, p-1]* is the pixel search range. For every candidate block with a displacement *(dx, dy)*, a sum of absolute difference (SAD) is calculated, which is given by

$$SAD(dx,dy)=\sum_{m=x}^{x+N-1}\sum_{n=y}^{y+N-1}\left|I_k(m,n)-I_{k-1}(m+dx,n+dy)\right| \quad (1)$$

where $I_k(m,n)$, $I_{k-1}(m,n)$ are the intensity values of the pixels located at position *(m, n)* in current and previous blocks, respectively. A similar SAD for the next candidate block is calculated and compared to the existing SAD. The block giving the smaller SAD is kept as the minimum candidate. This process loops until all candidate blocks are matched and a final minimum SAD is obtained. The motion vector is the displacement *(dx, dy)* of the block, which has minimum SAD within the current block [3].

B. Content-based Motion Estimation

Natural video scenes contain many types of arbitrary-shaped video objects. However, in previous video coding schemes (MPEG-1/2), they are treated as unique video blocks as shown in Fig. 2. The recently finalised MPEG-4 standard emphasises content-based video coding, which is different from previous versions of video coding standards [7, 8]. In the MPEG-4 standard, video frames are segmented into video objects as demonstrated in Fig. 3.

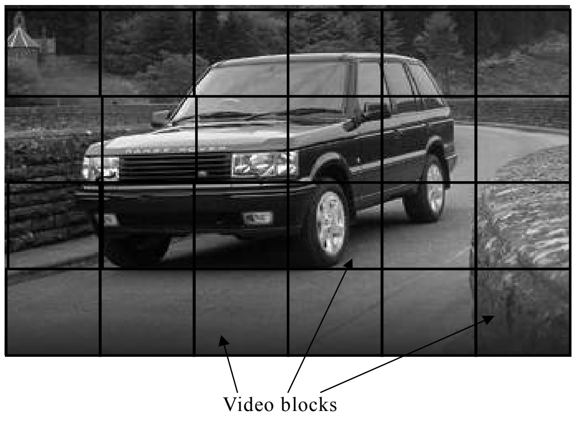

Fig. 2. Video blocks in previous video coding standard.

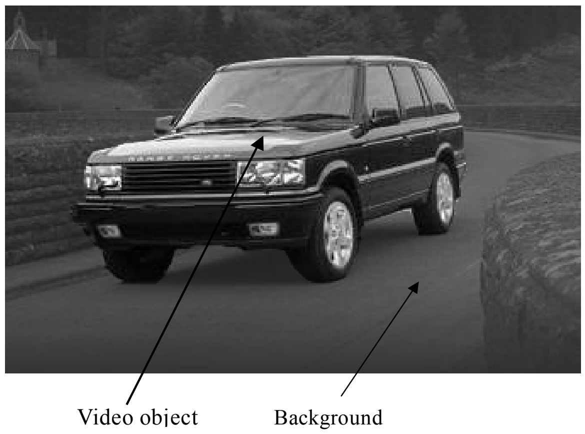

Fig. 3. Video object and background in content-based video coding standard.

In order to support the motion estimation of arbitrary-shaped video objects, an alpha binary plane has to be defined. The alpha plane contains the information on whether a pixel is inside the object or not [3]. Thus, the SAD for the video object can be represented below:

$$SAD(dx,dy)=\sum_{m=x}^{x+N-1}\sum_{n=y}^{y+N-1}\left|I_k(m,n)-\right.$$
$$\left.-I_{i-1}(m+dx,n+dy)\right|\times Alpha(x,y) \quad (2)$$

where alpha *(x, y)* is the binary value for the *(x, y)* pixel in the current block. The value is one when the pixel is inside the object; otherwise, it is zero, as shown in Fig. 4. Thus, the SAD is calculated only for the video object rather than the video block.

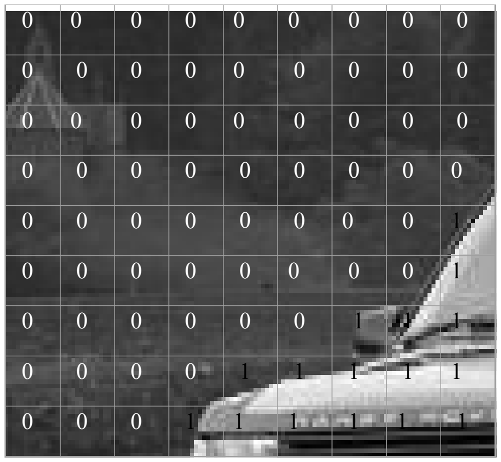

Fig. 4. Illustration of alpha binary plane

III. The Proposed Architecture

A. System Overview

The Fig. 5 shows the block diagram of the proposed ME architecture, which includes five components: memory unit, address generator, PE array, minimum unit and control CPU [3].

The memory unit is divided into two modules, one of which is to store current frame data and alpha plane data; another is for previous frame data. The address generator computes the addresses, at which the candidate pixels for block matching are stored. It also fetches the pixel data from the memory unit and feeds them into the PE array. The PE array computes the absolute difference between previous and current frames and sends result to the minimum unit. Then, the SADs of all parallel-processed blocks are generated in the minimum unit, and these SADs are compared to find the minimum one to be stored in the minimum SAD register. Meanwhile, a minimum flag signal is output to the control CPU, which, jointly with address generator, gives the location where a motion vector is found.

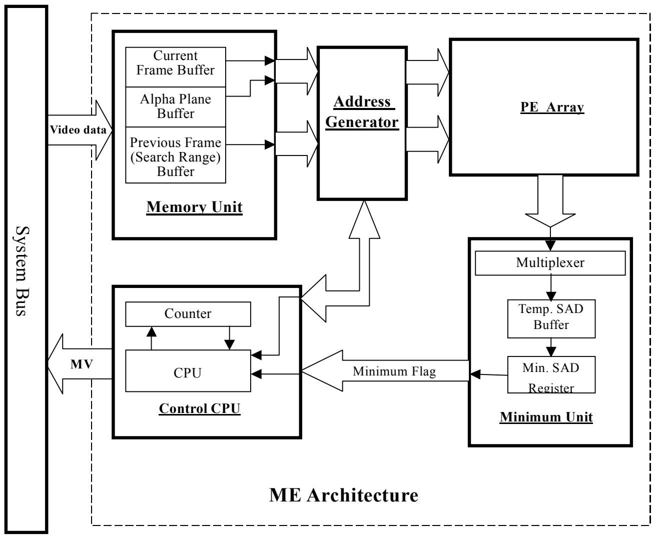

Fig. 5. System block diagram

B. PE Array

The PE array is a key component of ME architecture. It predominantly determines the performance of the system in terms of memory bandwidth and minimum clock rate for real-time processing. Based on a one-dimensional tree architecture presented in [9], PE array architecture uses additional preload cycles and in order to increase the parallelism of the data flow, a group of parallel-pipelined processing elements have been adopted, as shown in Fig. 6.

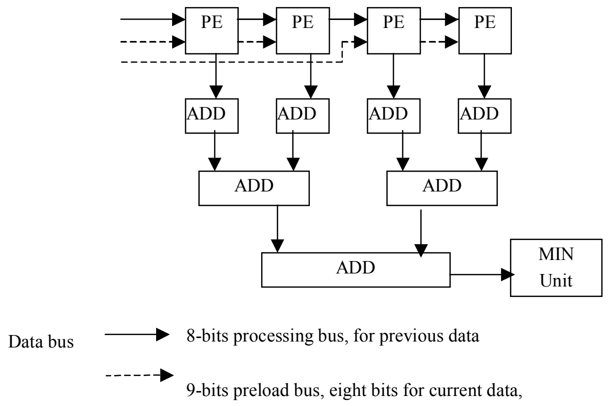
PE
PE
PE
PE
ADD
ADD
ADD
ADD
ADD
ADD
ADD
MIN Unit
Data bus
8-bits processing bus, for previous data
9-bits preload bus, eight bits for current data,

Fig. 6. PE array structure

In this architecture, motion estimation is carried out in two stages, preload and matching. In preload cycles, as shown in Fig. 7, current block data and alpha plane data are preloaded into the PE array. They are stored locally in the register within corresponding PEs. Then, as illustrated in Fig. 8, in the matching cycles, previous block data are loaded into the PEs by parallel pipelining. Before matching to the preloaded current data, previous data must align with the current data. It takes N_{PE} clock cycles to align previous data with current data, where N_{PE} is the number of PEs. While the SAD calculation starts, the previous data shift from left to right within the PE array until they match the corresponding current data already in the PE array. In every clock cycle, N_{PE} absolute difference values for each of the parallel-processed blocks are calculated; they are summed up by a group of adders in the PE array (Fig. 6). The summed result is then sent to the minimum unit to calculate the SADs for each of the matching points and the unit finds the minimum SAD for motion vector.

Apparently, there should be N_{PE} processing elements in the

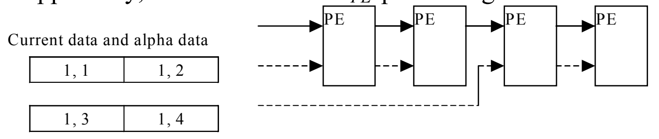

Fig. 7. Preload cycles

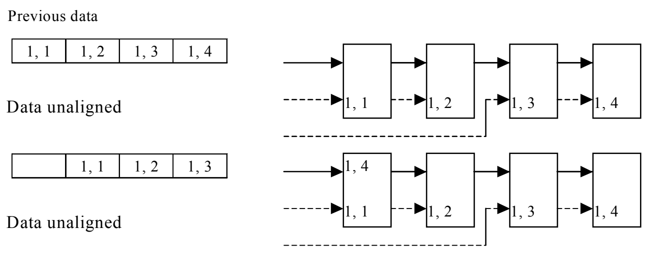

Fig. 8. Matching cycles

PE array. Here, we assume that N_{PE} is 16, and as an example, full search BMA is chosen to evaluate this architecture. In this case, N_p candidate blocks can be processed simultaneously, and the pipelining can be organised as in Fig. 9.

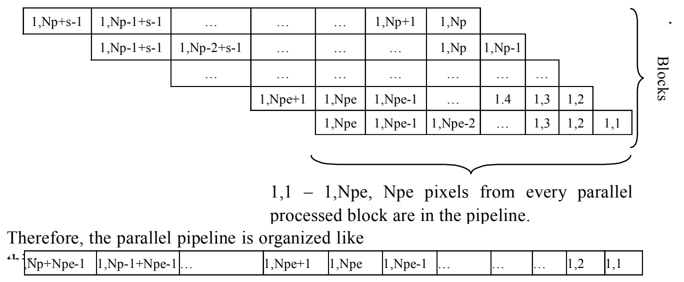
1,Np+s-1 | 1,Np-1+s-1 | … | … | … | 1,Np+1 | 1,Np
1,Np-1+s-1 | 1,Np-2+s-1 | … | … | … | 1,Np | 1,Np-1
… | … | … | … | … | … | …
1,Npe+1 | 1,Npe | 1,Npe-1 | … | 1.4 | 1,3 | 1,2
1,Npe | 1,Npe-1 | 1,Npe-2 | … | 1,3 | 1,2 | 1,1
Blocks
1,1 – 1,Npe, Npe pixels from every parallel processed block are in the pipeline.
Therefore, the parallel pipeline is organized like
Np+Npe-1 | 1,Np-1+Npe-1 | … | 1,Npe+1 | 1,Npe | 1,Npe-1 | … | … | … | 1,2 | 1,1

Fig. 9. Pipeline organisation

C. Dual Register/Buffer Structure

The architecture presented in the section III B suggests that current data and alpha plane data need to be loaded only once when processing N_p candidate blocks. Hence, as N_p is increased, the bandwidth required for processing current data is sharply reduced. However, extra clock cycles are needed to preload current data and align previous data with current data, thus the PEs are in idle status during the preloading and the alignment. For instance, 16 cycles are needed to preload current data and alpha plane data, and 16 cycles are needed to achieve the data alignment for a 16-PE architecture. This can cause a high clock speed requirement. To solve the problem, a dual register/buffer structure has been introduced in the processing elements. As illustrated in the Fig. 10, in each of the PEs, there are two 8-bit registers for the previous data and two 9-bit registers for the current and alpha plane data, respectively. This allows preloading and matching to be performed simultaneously. While the PE is matching the data in register Group A, the following data are preloaded into register Group B. Furthermore, when the matching operations of the data in Group A are completed, the PE switches

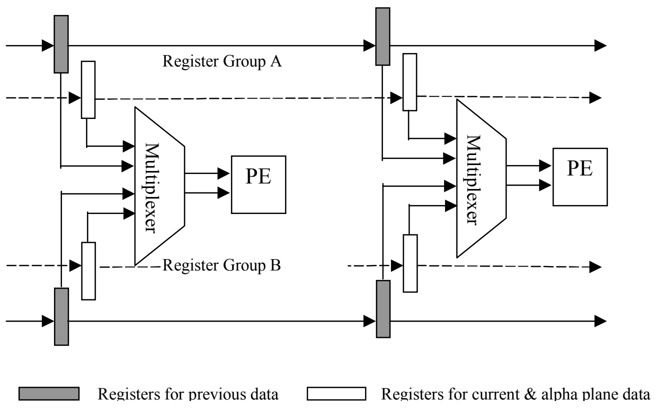
Register Group A
Multiplexer
PE
Register Group B
Multiplexer
PE
Registers for previous data
Registers for current & alpha plane data

Fig. 10. Dual register/buffer structure

operational mode to match the data in Group B, while the Group A register is being accessed during preloading cycles.

D. High Parallelism

Whereas different from other architectures presented in [3, 4, 10, 11, 12] high-parallel processing can be easily achieved with the proposed architecture, because high number of blocks (more than 16) can be processed simultaneously. Fig. 11 illustrates the data flow organisation of the architecture processing 32 blocks in parallel. In this illustration the pixel search range is *[-8, 7]* and the block size is *16×16*. When the data of the first row (from *[1, 1]* to *[1,16]*) of the current block is in the PE array, as shown in Fig. 11 (a), all previous data need to be matched in the search range, as shown in the Fig. 11 (b). The data required to match the first sixteen blocks is the first row of the search range, as shown in the Fig. 11 (c). To achieve high-parallel processing, we simply load the data from the blocks 17 to 32 (i.e., the second row of the search range) after completing the matching of blocks 1 to 16 without changing the current data, as shown in Fig. 11 (d). Hence, there is no need to access the memory for another group of current data, nor preloading cycles. Furthermore, with the dual register/buffer structure, data in the second row are loaded into register Group B at the same time as matching the first row data in register Group A. This allows the alignment cycles to be skipped for the previous data.

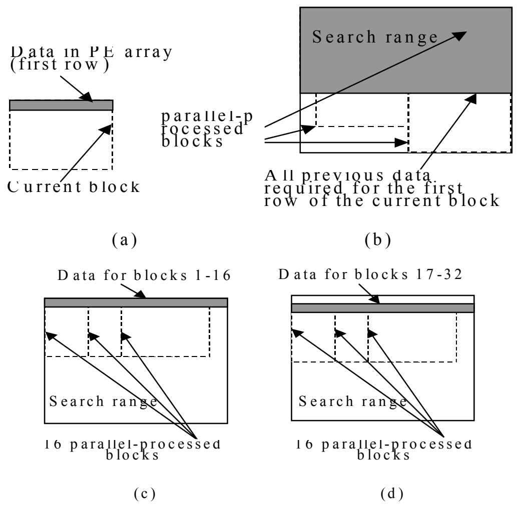
Data in PE array (first row)
Current block
(a)
Search range
parallel-processed blocks
All previous data required for the first row of the current block
(b)
Data for blocks 1-16
Search range
16 parallel-processed blocks
(c)
Data for blocks 17-32
Search range
16 parallel-processed blocks
(d)

Fig. 11. Parallel data flow for previous block

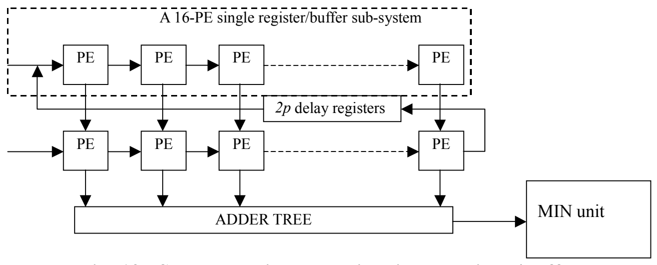

Fig. 12. Scaled architecture with single register/buffer

E. Scalability

Both single register and dual register architectures can be used to construct more powerful 2D ME engines without changing the main structure. For instance, Fig. 12 illustrates a 2-D system, which is stacked by 16-PE architectures with single register. The dash-line box shows a 16-PE ME engine, which is defined as a sub-system in scaled architectures. Each sub-system processes 16 pixels, which are within a row in the search range (Fig. 11 c). Obviously, compared to the non-scaled ones, the system through-put is increased and the minimum clock rate required is further reduced. If two or more rows of previous data within the search window are processed simultaneously in ME engines, in other words, the sub-block is larger than a single previous data row, some of the data rows will be overlapped in adjacent *2p* candidate sub-blocks, as shown in Fig. 13. Here the two dash-line boxes show the data needed for adjacent *2p* candidate sub-blocks. The overlapped

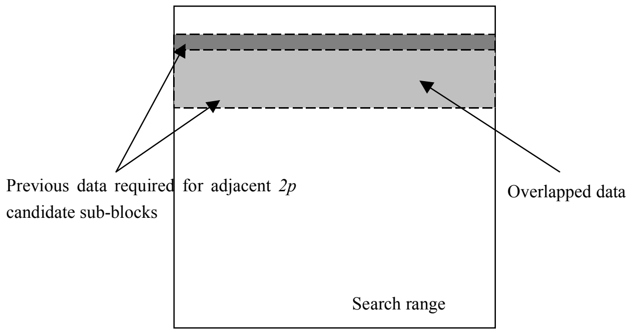

Fig. 13. Data overlapping

data can be reused for calculating the next *2p* candidate sub-blocks by passing them from the lower sub-system to the upper one. Therefore, we connect the end of the match bus in the lower sub-system to the match bus of the upper one (Fig. 12) to achieve data reuse. In single buffer architecture, *2p* shift registers are added in order to delay the data from the lower sub-system and synchronise the operations. In the dual buffer, one group of buffers can be used as a shift register while the other one is in process of operation. Thus, a dual buffer PE can be used to construct a scaled system without adding extra components. As a result, the memory bandwidth can be reduced by improving the data reuse efficiency.

IV. PERFORMANCE ANALYSIS

This section analyses minimum required memory bandwidth and minimum clock rate, which are important parameters in terms of power consumption and speed.

A. *Minimum Required Clock Rate*

If we denote $N_h \times N_v$ for the frame size, *fps* for the frame rate, *[-p, p-1]* for search range and $N \times N$ for block size. Then the minimum clock rate of single register PE architecture is given by

$$C_{clk\_single} = \frac{[N_{PE}/N_{pre} + (N_{PE}-1) + 2p] \times N^2 \times (2p)^2 \times fps \times N_h \times N_v}{N_p \times N^2 \times N_{PE}} \quad (3)$$

where N_{PE} is the number of processing elements; N_{pre} is the number of preload buses; N_p is the number of parallel processed candidate blocks [5].

For the PE array with the dual register/buffer structure, there are only $N_{PE} - 1$ preload cycles at the beginning of motion estimation process and $2p$ matching cycles per sub-block. Hence, the minimum clock rate can be calculated as:

$$C_{clk\_dual} = 2p \times \frac{N_p}{N} \times \frac{1}{N_p} \times (2p)^2 \times \frac{N^2}{N_{PE}} \times \frac{N_h \times N_v \times fps}{N^2} + N_{PE} - 1 \quad (4)$$

The pictorial representation of these formulas is given in Fig. 14, which clearly shows the relationship between the minimum clock speed and N_p (where N_{pre} is 4).

Similarly, for scaled single register architectures, we have:

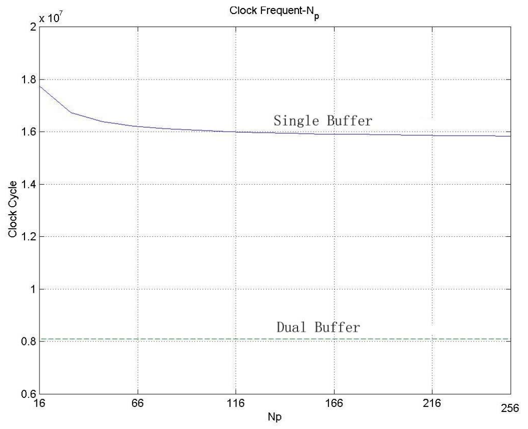

Fig. 14. Relationship between Minimum Clock Speed and N_p

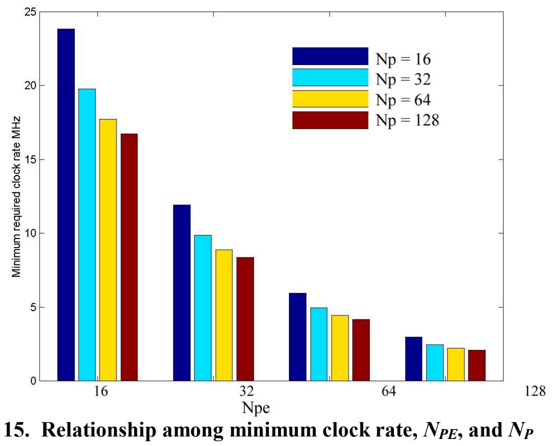

Fig. 15. Relationship among minimum clock rate, N_{PE}, and N_P

$$C_{clk} = (N + ((N-1) + 2p) \times \frac{N_p}{N}) \times \frac{1}{N_p} \times \frac{N^2}{N_{PE}} \times (2p)^2 \times \frac{N_h \times N_v \times fps}{N^2} \quad (5)$$

If we adopt N_{PE} = *16, 32, 64, 128* and N_p = *16, 32, 64* and *128,* respectively, the pictorial presentation of the minimum clock rate can be given in Fig. 15.

From the figure, it is clear that the clock rate is reduced as N_{PE} and N_P increases. Also as N_{PE} increases, parallelism decreases.

For architectures with dual register/buffer structure, the clock rate can be further reduced by scaling architectures, as presented in Section III. E. The minimum clock rate can be calculated using:

$$C_{clk\_dual} = 2p \times \frac{N_p}{N} \times \frac{1}{N_p} \times (2p)^2 \times \frac{N^2}{N_{PE}} \times \frac{N_h \times N_v \times fps}{N^2} + N_{PE} - 1 \quad (6)$$

Fig. 16 illustrates the minimum required clock rate for ME engines with a dual register/buffer, where we select N_{PE} = *16, 32, 64, 128* and N_p = *16, 32, 64* and *128* as the parameters.

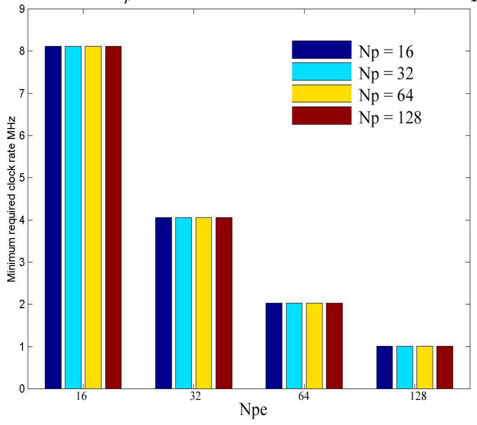

Fig. 16. Minimum required clock rate for architectures with dual register/buffer

The results for the non-scaled architecture show that the minimum required clock rate is entirely determined by N_{PE}.

B. *The minimum memory bandwidth*

Power consumption is another important consideration for the intended mobile applications. For ME algorithms, memory access operations are dominant factor that contributes to power consumption, rather than clock rate [3]. For the proposed architecture, minimum memory bandwidth can be determined as below.

If M_{BW} represents the total amount of data fed to the PE array per second, then M_{BW} is equal to the quantity of memory access for every candidate block multiplied by the number of candidate blocks. Both single and dual buffer architectures have the same amount of memory access in calculating every block. Therefore, the memory bandwidth for both architectures can be given as:

$$M_{bw} = (Q_{current\&alpha} + Q_{previous}) \times \frac{1}{N_p} \times N_{sub} \times N_{can} \times \frac{N_h \times N_v \times fps}{N^2} \quad (7)$$

where the $Q_{current\ \&\ alpha}$ is the quantity of current and alpha data memory access for every sub-block; and the $Q_{previous}$ is the quantity of previous memory access for N_p candidate sub-blocks.

Then,

$$M_{bw} = (N_{PE} \times 9 + (\frac{N_p}{N}(N+2p-1)\times 8)) \times \frac{1}{N_p} \times \frac{N^2}{N_{PE}} \times (2p)^2 \times \frac{N_h \times N_v \times fps}{N^2} \quad (8)$$

If the preload bus has 9 bits, 8 bits are needed for the current block and 1 bit for the alpha plane data. The matching data bus is an 8-bit bus for the previous data. The above formula can also be represented pictorially, as shown in Fig. 17.

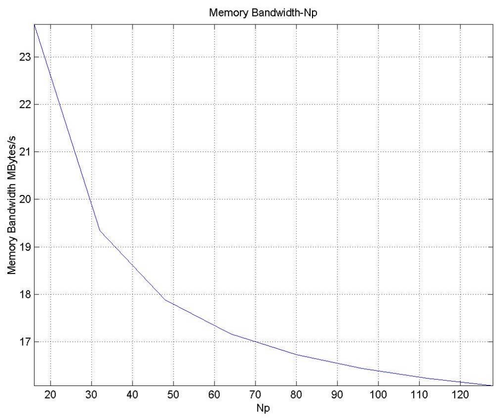

Fig. 17. Relationship between Minimum Memory Bandwidth and N_p

The minimum memory bandwidth can be reduced by scaling the architecture. We use the same approach to analyse the minimum memory bandwidth for scaled architectures as we did with non-scaled architectures. Thus, we have

$$M_{bw} = (N_{PE} \times 9 + (\frac{N_{PE}}{16} + \frac{N_p}{2p} - 1) \times (2p+N-1) \times 8) \times \frac{1}{N_p} \times \frac{N^2}{N_{PE}} \times (2p)^2 \times \frac{N_h \times N_v \times fps}{N^2} \quad (9)$$

The memory bandwidth can be presented by the bar chart as shown in Fig. 18, where we used 2, 4, 6, and 8 sub-system and 16, 32, 64, 128 parallel processed blocks, respectively.

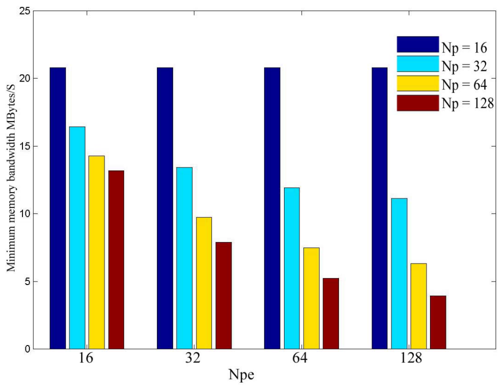

Fig. 18. Relationship among memory bandwidth, N_p, and N_{PE}

From the figure, we can see that the minimum memory bandwidth is sharply reduced as N_p and N_{PE} increases. Moreover, the higher N_{PE} is quicker the minimum clock rate declines with the increase of N_p. In other words, the scaled architecture with more sub-systems and a higher number of parallel processing blocks has the higher data utilisation rate. A detailed analysis of clock rate and minimum bandwidth for the different systems has been carried out and the results are given in Tab. 1. Here CLK is minimum required clock cycle for real-time processing, N_{cycle} and Bytes/ MB are for the number of clock cycles and memory access (in bytes) in the process of calculating a block, respectively. N_{cycle} represents the architecture's hardware utilisation efficiency, which determines the minimum required clock rate. Bytes/MB represents performance in terms of data re-use, which can reduce the power consumption with less memory access. Apparently, while the non-scaled architecture has reasonable minimum speed and memory bandwidth required for mobile video applications, the scaled architecture provides the higher speed and memory bandwidth, which are crucial for future high-resolution applications.

TABLE I

DETAILED ANALYSIS RESULTS FOR THE SELECTED SYSTEMS

N_{PE}	N_p	CLK Single Register	CLK Double Register	Memory Bandwidth (Byte/s)	N_{cycle} with Single Register	N_{cycle} with Dual Register	Bytes /MB
16	128	17,233,920	8,110,095	16,853,760	8,704	4,096	8,512
32	128	8,616,960	4,055,055	9,979,200	4,352	2,048	5,040
64	64	4,561,920	2,027,535	9,155,520	2,304	1,024	4,624
64	128	4,308,480	2,027,535	6,541,920	2,176	1,024	3,304
128	64	2,280,960	1,013,775	7,682,400	1,152	512	3,880
128	128	2,154,240	1,013,775	4,823,280	1,088	512	2,436

C. *Analyses result comparisons*

The comparison of N_{cycle} and *Bytes/MB* with other architectures are shown in Tab. 2. The proposed 16-PE system has the same N_{cycle} as other 16-PE architectures. However the memory access parameter (*Bytes/MB*) is reduced about 29% compared to SEII in [10]. The 64-PE system reduced the N_{cycle} by 51% compared to the QCIF profile in [12].

TABLE 2

ANALYSIS RESULT COMPARISONS

	SE II [10]	[11]	QCIF profile [12]	Proposed architecture 1	Proposed architectur e 2
PE number	16	16	64	16	64
Algorithm	Content -based Full-search	Full-search	Enhanced Full-search	Content-based Full-search	Content-based Full-search
N_{cycle}	4,111	4,111	2,116	4,111	1,024
Bytes/MB	12,047	----	----	8,512	3,304

V. FPGA IMPLEMENTATION AND COMPARISONS

A. *FPGA Implementation*

FPGA (Filed Programmable Gate Array) has proved to be a high-performance and economic method of digital IC implementation due to its short development period and flexibility. Aiming at low-power, low-cost mobile video applications, a 16-PE, 128-block-parallel dual buffer PE architecture with two embedded block RAM and one distributed RAM is prototyped on a single Xilinx Spartan II

FPGA chip. We chose block size *N×N = 16×16* and search range *[-7, 8]* for MPEG-4 content-based full-search motion estimation. The VLSI area in terms of gate equivalents is shown in Tab. 3. The PE array has the highest cost of all the hardware component this is because of its complexity. Due to the regularity of full-search algorithm, the AGU and main controller only occupy a small number of logic gates, which indicated a low-cost design. The gate equivalent of the chosen architecture is about 10k.

TABLE 3
GATE EQUIVALENTS

Module	Gate Equivalents
PE Array	8,860
Min Unit (without RAM)	756
Main Controller	243
AGU	372
Total (without RAM)	10,231

The minimum required clock rate and power consumption for a group of popular video profile are given in Tab. 4. 12.5% is used as default switching rate for average condition during the simulation, while 100% represents the worst case. For 20 *fps QICF* real-time motion estimation, the proposed architecture operates at 8.2 *MHz* and consumes 36.76 *mw* energy. Therefore, the proposed architecture and its implementation is suitable for targeted low-cost and low-power mobile video applications.

TABLE 4
PERFORMANCE IN FULL-SEARCH MODE

Video Profile	Video Format (Pixel × Pixel)	Frame Rate per Second (fps)	Minimum Clock Rate for Real-time ME (MHz)	Power Consumption (mw) Average/Worst*
Mobile Applications	QCIF	20	8.2	36.76/98.19
Video Conference	CIF	20	32.6	108.3/356.3
		30	49	168.62/575.25
HDTV 720i	1280×720 interlaced	30	222	700.64/2362.10

B. *Implementation Comparisons with Other Architectures*

The Tab. 5 gives the implementation comparison between proposed architecture with some existing ones. It shows that it can operate at a relatively lower speed while processing the same among of video data as other architectures. Furthermore, its power consumption is also much lower than other comparable architectures as shown in the table.

TABLE 5
IMPLEMENTATION RESULT COMPARISON

	SE II [10]	[11]	QCIF Profile [12]	CIF Profile [12]	Proposed QCIF	Proposed CIF
PE	16	16	64	256	16	16
Clock Rate	100MHz	147MHz	6.56 MHz	72 MHz	8.2 MHz	49 MHz
Design Technology	ASIC	ASIC	ASIC	ASIC	FPGA	FPGA
Process /FPGA model	0.25 μm	----	0.25 μm	0.25 μm	Xilinx Spartan II XC2S50	Xilinx Spartan II XC2S50
Voltage	2.5V	2.1V	2.5V	2.5V	2.5V	2.5V
On Chip RAM	2,481 Byte	----	--9K Bits	9K Bits	1,344 Bytes	1,344 Bytes
Algorithm	Content-based Full-search; TSS; Sub sampling	Full-search	Enhanced Full-search	Enhanced Full-search	Content-based Full-search	Content-based Full-search
Supported Video Format	QCIF/2QCIF/CIF	QCIF/CIF/PAL	QCIF	CIF	QCIF	CIF
Power Consumption	----	3.48w worst	15mw	170mw	36.76mw 98.19 worst	168.62 mw 575.25 worst
Chip Area/Gate	0.5mm^2/22k	11.5 mm^2	1.6 mm^2/29k	1.6 mm^2/29k	20×20 mm^{2*}/10k	20×20 mm^{2*}/10k

VI. CONCLUSION

This paper presents a parallel VLSI architecture for MPEG-4 content-based motion estimation, aiming at mobile applications. The architecture features dual register/buffer PE, high parallelism and its scalability, thus reducing memory access and offering highly efficient ME operation. It is successfully prototyped on a single Xilinx Spartan II FPGA. The simulation result shows that it can calculate motion vector in real-time for 20 *fps QICF* video sequence at a 8.2 *MHz* clock rate. At the same time, the power consumption is only 36.76 *mw*. In addition, fast-search algorithms [13, 14] and dynamic power management [15, 16] is being investigated on the architecture.

REFERENCES

[1] S.N. Fabri, S. Worral, A. Sadka, and A. Kondoz, *Real-time Video Communications over GPRS*, University of Surrey, UK.

[2] E.L.H. Zoraya, "Evolution in the Technological Revolution: Preparing for 3G Wireless Technology," *National Urban League Technology Policy Alert*, January, 2001

[3] P. Kuhn, Algorithms, *Complexity Analysis And VLSI Architecture for MPEG-4 Motion Estimation*, KLUWER ACDEMIC PUBLISHERS, London, 2000.

[4] A.J. Roach and A. Moini, "VLSI Architecture for Motion Estimation on a Single-chip Video Camera," *Visual Communications and Image Processing 2000, Proceedings of SPIE*, Vol. 4067, 2000.

[5] R. Gao, D. Xu and J.P. Bentley , "A Parallel Processing Engine for Motion Estimation in MPEG-4 Multimedia Applications", *Advances in Multimedia, Video and Signal Processing Systems, Proceeding of WSEAS ICOSMO 2002 Greece*, pp 125- 132.

[6] T. Ebrahimi and Caspar Horne, "MPEG-4 Natural Video Coding- An Overview", *Signal Processing: Image Communication*, Vol. 15, pp365-385, 2000

[7] F. Pereira, "MPEG-4: Why, what, how and when?", *Signal Processing: Image Communications*, Vol. 15, pp 271-279, 2000

[8] E. Touradj, C. Horne, "MPEG-4 Natural Video Coding - An Overview," *Signal Processing: Image communication*, Vol. 15, pp.365-385, 2000.

[9] Y.S. Jehng, L.G. Chen, and T.D. Chiueh, “An Efficient and Simple VLSI Tree Architecture for Motion Estimation Algorithms,” *IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL 41, NO. 2*, FEB. 1993.

[10] P.M. Kuhn, “Fast MPEG-4 Motion Estimation: Processor Based and Flexible VLSI Implementations”, *Journal of VLSI Signal Processing 23* 67-92, 1999

[11] J.Fco. Lo´pez, P. Corte´s, S. Lo´pez, R. Sarmiento, “Design of a 270 MHz/340 mW processing element for high performance motion estimation systems application”, *Microelectronics Journal 33 (2002) 1123–1134*

[12] L. Fanucci, L. Bertini and S. Saponara “Programmable and Low Power VLSI Architecture for Full Search Motion Estimation in Multimedia Communications*”, IEEE International Conference on Multimedia and Expo (III) 2000: 1395-1398*

[13] J. Chalidabhongse, C.-C. Jay Kuo, “Fast Motion Vector Estimation Using Multiresolution-spatio-temporal Correlations”, *IEEE Transactions on Circuits and System for Video Technology, Vol. 7, no. 3, Jun. 1997, pp 477-488*

[14] F. Dufaux, M. Kunt, “Motion Estimation Techniques for Digital TV: A Review and a New Contribution”, *Proc. IEEE, vol. 83, no. 6, pp. 858 – 876, 1995*

[15] I. Brynjolfson and Z. Zilic. “Dynamic clock management for low power applications in fpgas”, *IEEE Custom Integrated Circuits Conference*, pages 139-142, 2000.

[16] V. Krishna, N. Ranganathan and N. Vijaykrishnan, “Energy Efficient Datapath Synthesis Using Dynamic Frequency Clocking andMultiple Voltages”, *Proc. of 12th International Conference on VLSI Design, pp. 440-445, Jan. 1999.*

Rui Gao is currently PhD candidate at section of electronics & electrical, School of Science & Technology, University of Teesside. Mr. Gao received his BEng at Harbin Institute of Technology, China, and start his Mphil/PhD research at University of Teesside in 2000. His research interests are in the area of: VLSI for mobile video applications, digital video processing, video compression techniques and VLSI prototyping.

Donglai Xu received his BSc and MSc degrees, all in electronic engineering, from the Xidian University, Xian, China, in 1985 and 1990, and a PhD degree from the University of Bradford, UK, in 1999, in Electronic and Electrical Engineering.
From 1990 to 1994, he was with the Xidian University, in China, as a Research Assistant Professor where he has been involved in research of digital systems and VLSI design. Since joining the University of Teesside, UK, in 1998, he is currently a Senior Lecturer. His present research interests include digital system design, video coding and VLSI/ASIC modelling.