AN IMPROVED PARALLEL ARCHUTECTURE FOR MPEG-4 MOTION ESTIMATION IN 3G MOBILE APPLICATIONS

Donglai Xu, Rui Gao

SST, University of Teesside
Middlesbrough, TS1 3BA, UK
d.xu@tees.ac.uk

Hadj Batatia

IRIT
ENSEEIHT, Toulouse, 31071, France
batatia@enseeiht.fr

ABSTRACT

A high-parallel VLSI core architecture for MPEG-4 motion estimation is proposed in this paper. It possesses the characteristics of low memory bandwidth and low clock rate requirements, thus primarily aiming at 3G mobile applications. Based on a one-dimensional tree architecture, the architecture employs the dual-register/buffer technique to reduce the preload and alignment cycles. As an example, full-search block matching algorithm has been mapped onto this architecture using a 16-PE array that has the ability to calculate the motion vectors of QCIF video sequences in real time at 1 MHz clock rate and using 15.5 Mbytes/s memory bandwidth.

1. INTRODUCTION

The third generation (3G) wireless system provides the high-speed mobile platform with Internet Protocol (IP) [1], which allows the implementation of many types of IP-based internet applications, such as e-mail service, web page browsing and image/video transmission. Among these, real-time video applications represent an important part of mobile multimedia [2]. However, due to the inherent data intensity of video, compression techniques are required to reduce bit rates. This is achieved largely by exploiting temporal data redundancy in video streams such as motion estimation (ME) techniques. Since the 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. However, most of them target at 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 high parallel and low power consumption architecture that is based on a one-dimensional tree architecture is presented [5]. It features the high data utilisation by using parallel pipelining and the low clock rate by introducing the dual-register/buffer technique that reduces idle clock cycles.

The rest of the paper is organised as follows. In section 2, two typical ME algorithms are briefly described. Section 3 presents the proposed VLSI architecture in detail with emphasis on the key component PE array. In section 4, the performance of the architecture is analysed in terms of the minimum clock rate and the minimum memory bandwidth requirement. Finally, the conclusions are drawn in section 5.

2. MOTION ESTIMATION ALGORITHMS

Recently, MPEG-4 video standard has been introduced to cover wireless multimedia applications [2]. It adopts block-matching algorithms with alpha binary plane to achieve motion estimation [3, 6]. Figure 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 within a *search area* 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.

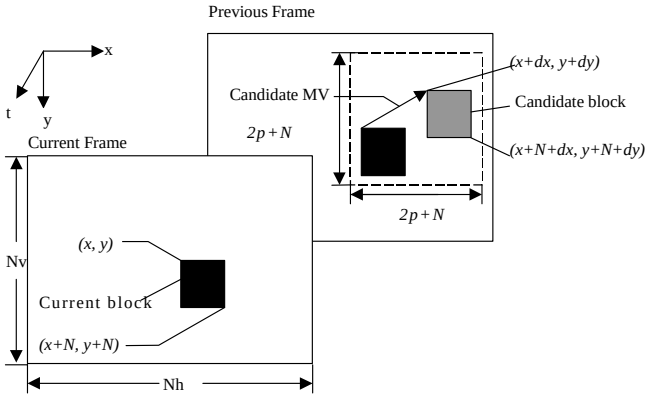

Figure 1. Block-matching

2.1. Full-search block-matching algorithm

Because of its low distortion and regular data flow, the full-search block-matching has been one of the most widely used ME algorithms. In this algorithm, the current block located at the pixel *(x,y)*, as shown in the figure 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

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

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. Similar SAD for 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 continues until all blocks are matched and a final minimum SAD is obtained. The motion vector is considered as being the displacement *(dx, dy)* of the block corresponding to this minimum [3].

2.2. Object-based motion estimation

Recently finalised MPEG-4 standard emphasises object-based motion estimation, which estimates the movements of the objects in a video sequence, rather than blocks [6]. To support the arbitrary-shaped objects motion estimation, an alpha binary plane has to be defined. The alpha plane contains the information of whether a pixel is inside the object or not [3]. Thus, the SAD for the object can be represented below:

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

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 the figure 2.

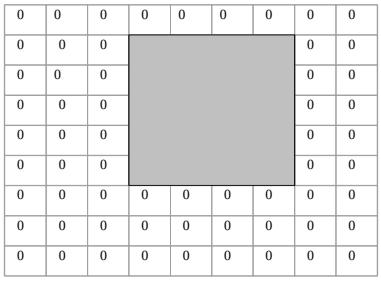

Figure 2. Alpha binary plane

3. PROPOSED ARCITECTURE

In this section, we describe the main components of the proposed architecture and give details of the PE array, which is the most computationally intensive part of the system.

3.1. System overview

The figure 3 shows the block diagram of the 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 is to store current frame data and alpha plane data; the other is for previous frame data. The address generator computes the addresses, at which the candidate pixels for the block matching are stored. It also fetches the pixel data from 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.

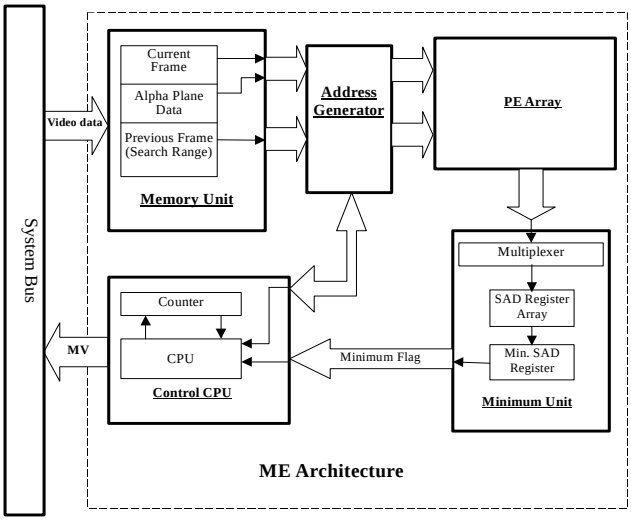

Figure 3. System block diagram

3.2. PE array

The PE array is the key component of the 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 the [5], the PE array architecture uses additional preload cycles and to increase the parallelism of the data flow, a group of parallel-pipelined processing elements have been adopted, as shown in the figure 4.

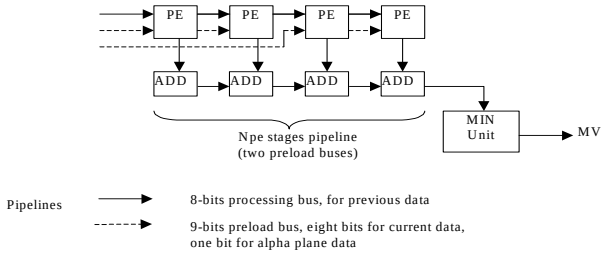

Figure 4. PE array architecture

In this architecture, motion estimation is carried out through two stages, preload and matching. In preload cycles, as shown in the figure 5, the current block data and the alpha plane data are preloaded into the PE array. They are stored locally in the appropriate PEs. Then, as illustrated in the figure 6, in the matching cycles, the previous block data are loaded into the PEs by parallel pipelining. Before matching to the preloaded current data, the previous data must align with the current data. It takes N_{PE} clock cycles to align the previous data with the current data, where N_{PE} is the number of PEs. While the SAD calculation starts, the previous data shift from the left to the right within 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 the adders in the PE array (Figure 4). The summed result is then sent to the minimum unit to calculate the SADs for each of the matching points and finds the minimum SAD for motion vector.

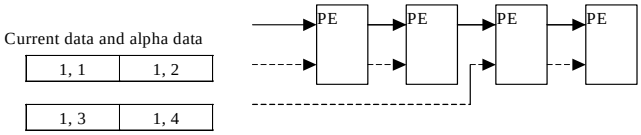

Figure 5. Preload cycles

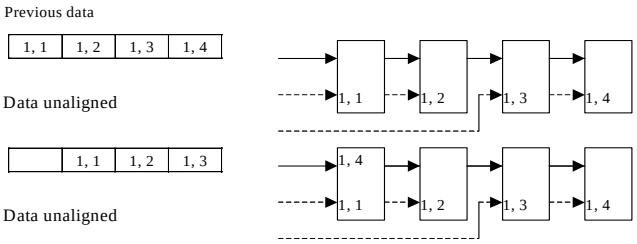

Figure 6. Matching cycles

Apparently, there should be N_{PE} processing elements in the PE array. Here, we assume that N_{PE} is 16, and as an example, full search BMA has been chosen to evaluate this architecture. In this case, N_p blocks of the candidates can be processed simultaneously, and the pipelining can be organised as in the figure 7.

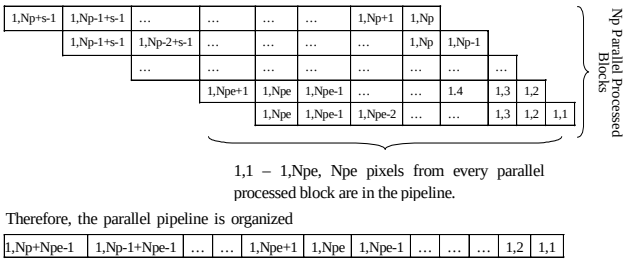

Figure 7. Pipeline Organisation

3.3. Dual register/buffer

The architecture presented in the section 3.2 suggests that the current data and the alpha plane data need to be loaded only once when processing N_p candidate blocks. Hence, as the 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 the previous data with current data, thus the PEs are in idle status during the preloading and the aligning. For instance, $N_{PE}/4$ (with four preload bus) cycles are needed to preload current data and alpha plane data, and N_{PE} cycles are needed to deal with data alignment. This can cause high clock speed requirement. To solve this problem, dual register/buffer structure has been introduced in processing elements. As illustrated in the figure 8, 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, to allow 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. In addition, when the matching operations of the data in Group A are completed, the PE switches operational mode to match the data in Group B, while the Group A register is during preloading cycles.

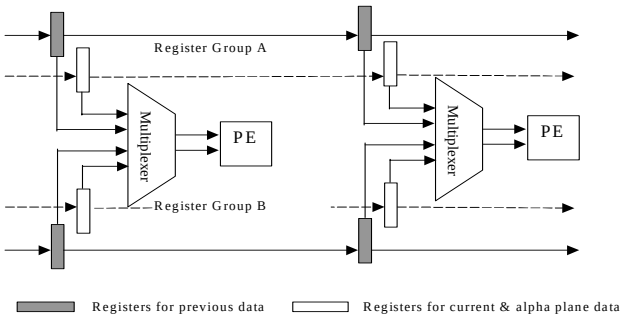

Figure 8. Double register architecture

3.4. High parallelism

Different from other architectures presented in the [3, 4, 5], high-parallel processing can be easily achieved on the proposed architecture This is due to high number of blocks (more than 16) can be processed simultaneously. Figure 9 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 figure 9 (a), all previous data need to be matched in the search range, as shown in the figure 9 (b). The data required to match the first sixteen blocks is the first row of the search range, as shown in the figure 9 (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 to16 without changing the current data, as shown in figure 9 (d). Hence, there is no need to access the memory for another group of current data, nor preloading cycles. Furthermore, with dual register/buffer structure, the data in the second row are loaded into register Group B at the same time of matching the first row data in register Group A. This allows the alignment cycles to be skipped for the previous data.

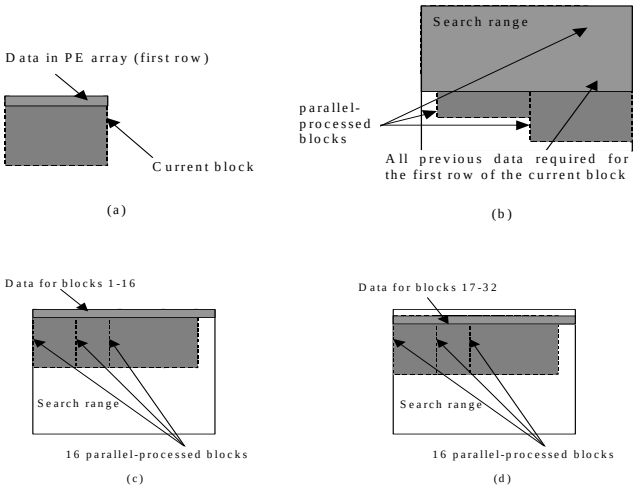

Figure 9. Paralleled data flow for previous block (search range)

4. PERFORMANCE ANALYSIS

The architecture is aiming at the 3G mobile platform, which currently has 64 kbits bandwidth to upload and transfer data. Thus, a minimum compression rate of 70 is required to achieve real time video applications with acceptable visual quality [4]. If we adopt QCIF as typical video format in mobile applications, uncompressed and compressed video data per frame will be (176×144×8)/1024 = 198 kbits and 198/70 = 2.829 kbits, respectively. Therefore, the video transmission rate over the 3G platform should be 64/2.829 = 22.628 frames.

Taking into consideration the bandwidth requirements of audio and protocol, the maximum frame rate is going to be 20 frames per second, which determines the minimum clock rate for real-time processing.

4.1. Minimum clock rate analysis

To meet the real-time processing condition, $(N_h \times N_v)/(N \times N) \times fps$ current blocks have to be matched per second, where $N_h \times N_v$ is the frame size (176 ×144 for QCIF), and $N \times N$ is the block size (16×16 in our case). With a search range [$-p$, $p-1$], for every current block, there are $N_{can} = 2p \times 2p$ candidate blocks. Therefore, $(N_h \times N_v)/(N \times N) \times fps \times (2p) \times (2p)$ pairs of blocks must be matched every second. The current blocks are divided into a group of N_{PE} -pixel sub-blocks, in which the pixels can be matched simultaneously within the PE array. The number of clock cycles to match a sub-block with N_p candidate blocks, which are processed simultaneously, is defined as C_{sub}. In addition, the number of the clock cycles needed to preload current data and alpha plane data is defined as C_{pre}, and the number of cycles for matching and aligning are defined as C_{match} and C_{align}, respectively. We have

$C_{sub} = (C_{pre} + C_{align} + C_{match}) \times N_{sub}$

$N_{sub} = (N \times N)/N_{PE}$; $C_{pre} = N_{PE} / N_{pre}$; $C_{align} = N_{PE} - 1$; $C_{match} = 2p$

where the N_{PE} is the number of processing elements; N_{pre} is the number of preload buses; N_{sub} is the number of sub-blocks within the current block. Moreover, C_{sub}/N_p is the number of clock cycles required to match a sub-block. Therefore, the minimum clock rate required is given by:

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

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

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

The above formulas suggest that the minimum clock speed required decreases while N_p increases for both single and dual register structures. And, for smaller N_p, the dual register based architecture requires much lower clock speed than that based on a single register.

4.2. Minimum memory bandwidth requirement analysis

Power consumption is another important consideration for the intended mobile applications. For an ME algorithm, memory access operations are the predominant factor contributing to the power consumption, rather than the clock rate [3]. For the parallel-pipelined architecture, as illustrated in the figure 7, the total amount of data fed to the PE array per second, M_{bw}, is equal to the quantity of memory access for every candidate block multiplied by the number of candidate blocks. Therefore,

$$M_{bw} = (Q_{current\&alpha} + Q_{Previous}) / N_p \times N_{sub} \times N_{can} \times fps \times N_h \times N_v / N^2$$

$$Q_{current\ \&alpha} = N_{PE} \times 9\ ;\ Q_{previous} = \frac{N_p}{N}(N + 2p - 1) \times 8$$

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

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

If the preload bus is 9-bit wide, 8 bits are needed for current block and 1 bit for the alpha plane data. The matching data bus is 8-bit wide for the previous data. This formula shows that the memory bandwidth is sharply reduced while N_p increases, especially when N_p is falling into the range of 16 and 32. In addition, the architectures with single and dual register/buffer structures have the same quantity of memory access per second. Therefore, they have the same minimum memory bandwidth requirement.

5. CONCLUSIONS

This paper presents a parallel VLSI architecture for motion estimation, aiming at 3G mobile applications. Initial analysis shows that the architecture requires relatively low memory bandwidth and clock rate, therefore suitable for low power consumption and low cost VLSI design/implementation. Moreover, due to the adoption of the dual-register structure, the architecture significantly speeds up data processing and therefore provides high throughput. These make the architecture ideal for the mobile video applications.

6. REFERENCES

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

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

[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] 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.

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