A DSP Architecture For Motion Estimation Accelerating

ZHANG Chun, YANG Kun, MAI Songping, WANG Zhihua

Tsinghua University, China

ABSTRACT

A modified DSP architecture to accelerate motion estimation (ME) algorithms is presented in this paper. The proposed SIMD and VLIW architecture is a trade-off between ASIC implementation and DSP implementation of ME, which can perform subtract, absolute and add (SAA) operation on 8 pixels and fetch 8 new pixels from memory at the same time. Flexible align addressing mode is provided to support efficient and continuous SAA operation on video stream. The DSP is estimated 20 times faster than SISD architecture to perform ME algorithms.

1. Introduction

Motion estimation (ME) is a key issue in many video compression system since it exploits the temporal correlation between adjacent frames in a video sequence to reduce the data inter-frame redundancy[1]. It is an essential part of several video-coding standards, such as ITU-T H.261[2], H.263[3], MPEG-2[4] and MPEG-4[5] and H.264/AVC[6]. Since ME operations can take up 80% of the computational burden of a complete video compression procedure, it is the most important in real-time video applications[7]. In this paper, a modified DSP architecture is presented. It is an application specified architecture specially designed for implementation of ME algorithms, which dedicates to reach a trade-off between fast process speed and implementation flexibility.

In section 2, ME algorithms and hardware implementations of such algorithms are briefly presented. In section 3, some modifications of a traditional DSP architecture to enhance the performance of the ME algorithm implementation are described. In section 4, a comparison with the old architecture is made. Finally, conclusions are drawn in section 5.

2. Motion estimation algorithms and hardware implementations

When the frame rate is sufficiently high, there is a great amount of similarity between successive frames. It is more efficient to code the difference between frames than to code the frames themselves. Motion estimation is aimed to reduce such strong temporal redundancy present in video sequence. So far, the most successful technique for motion estimation is block matching algorithm (BMA), which has been adopted by the video coding standards referred in section 1. As shown in figure 1, the current frame of a video sequence is divided into non-overlapped N×N blocks, named current block, and for each of them, a block in the previous frame, named candidate block, is searched in the searching area to be matching with the prior one. A matching criterion, or distortion function, which measures the similarity between the current block and the candidate block is defined as below:

\[
MAD (u, v) = \sum_{i=0}^{N-1} \sum_{m=0}^{N-1} | F_i(x + l, y + m) - F_{i+1}(x + u + l, y + v + m) |
\]  

MAD means Mean Absolute Difference. \( F_i \) and \( F_{i+1} \) refer to current block and candidate block to be compared. The values of \( u \) and \( v \) are limited in a \((W+1) \times (H+1)\) area.

The simplest way to find the candidate block is so called Full Search (FS) algorithm. In this algorithm, the current block shown in figure 1 is matched to every candidate block within a \((W+1) \times (H+1)\) search area, enabling the optimal motion vector to be found. But its high computational complexity makes it difficult to be implemented in real-time applications. In order to reduce the computational complexity of FS, many fast block-matching algorithms have been developed. Among them, the most successive ones are TSS (Three Step Search)[8] algorithm and some algorithm induced from TSS[9][10][11]. Most of these fast algorithms sacrifice search quality by introducing some assumptions and constraints. For example, use only a subset of the search area to reduce the total number of searches. For example, if \( W=H=14 \), FS algorithm needs 255 MAD computations, while TSS algorithm only needs 25 MAD computations and 15 MAD computations for NTSS algorithm in best case.

There are two traditional hardware implantations for motion estimation, ASIC implementation and DSP implementation. In the first case, by introducing parallel Process Elements (PE) and specially designed memory architecture and control logic, very high computation speed can be achieved. Typical parallel architectures are presented in [12][13], 9 PE for MAD computation are used for the TSS algorithm and 18 PE are used for the NTSS algorithm. Such architectures overcome the
irregular data flow and achieve efficiency close to 100%. They also reuse data and reduce the control burden. In the second case, BMA is implemented in high performance DSP, such as TMS320 series of Texas Instrument, and this leads to some design flexibility. By using some programming mechanisms like code level parallelism and efficient usage of internal memory, we can increase the DSP performance [14] [15]. In this paper, labor is made to find out a trade-off between the prior two method, that is, to find out a DSP architecture specially optimized for the core computation of BMA that can have both high speed, high efficiency of ASIC and flexibility of DSP.

3. Optimization to traditional DSP architecture for BMA

The Optimization is based on a 32-bit traditional RISC core OR1200 [16] from OpenCore’s OR1k family [17]. OR1200 is a typical 32-bit, 5-stage-pipeline RISC processor. It has open source C compiler, architectural simulator and RTL source code, which is very suitable for further development.

From (1) it is easy to notice that, the core computation of BMA is the accumulation of the absolute value of the result of a subtraction, which is defined to be SAA (Subtract Absolute Add). So, the first step of optimization is to design a computation unit to achieve a SAA calculation in a single clock cycle. Typically, video data length is 8 bits, by introducing SIMD (Single Instruction Multi Data) technology, 4 pairs of video data can be processed simultaneously in a 32-bit architecture. In 32-bit architecture, memory access is also processed in 32-bit manner, this leads to the possibility that some 8-bit video data of the total 4 8-bit video data is stored in one 32-bit memory word while the other 8-bit video data is stored in the next 32-bit memory word. In order to ensure the correct operation, alignment must be introduced. Two align unit extracts two group of 4 8-bit data from two pair of 32-bit source registers to been the input data of the SIMD process element. Finally, we get the architecture of SIMD process element shown in figure 2, which has 4 32-bit registers as input data, 4 SAA computation units, and stores the SAA result in two 32-bit accumulators. Data alignment method is also shown in figure 3.

By introducing the SIMD element, the modified DSP can fulfill 4 SAA calculations in a single clock cycle. But two extra clock cycles are needed for two memory load instruction to load 2 necessary 32-bit data words as the input data of the SIMD element. Actually, 3 clock cycles are needed to fulfill a SIMD process, which means the efficiency of SIMD element is only 33%. In order to solve this problem, VLIW (Very Long Instruction Word) is introduced into the OR1200 architecture, enabling the core to execute one SIMD and two memory access instructions in a single clock cycle. Thus theoretically enhances the efficiency of SIMD element to 100%. To realize VLIW, lots of modifications must be done to the OR1200 architecture. Proper hardware resource need to be added and pipe-line must be modified to upgrade the OR1200 to be a VLIW processor. One additional memory access unit is necessary to support two memory access in one clock cycle. Data cache is divided into two parts, one for current frame and one for reference frame.

4. Performance comparison

As mentioned in the prior section, the basic calculation of MAD includes subtract, absolute and accumulate. It will take three clock cycles for the original OR1200 to complete such a calculation. Plus the two clock cycle to load data from memory, five clock cycles are necessary for a complete MAD basic calculation. To get a MAD value from two 16 × 16 blocks, 1280 clock cycles are needed. While in a single clock cycle, 4 MAD basic calculations can be achieved, that is, only 64 clock cycles are needed for a MAD value mention before. The speed of the modified architecture is 20 times as the original one. Meanwhile, the modified architecture needs only 1/4 memory space to store the video data for an OR1200 implementation. In H.264/AVC coding, if we choose CIF(352 × 288) format and fix the block size to 16 × 16 and let the frame rate to be 24fps, 9504 block motion estimations per second are necessary. Ideally, by using TSS algorithm, 15.2M clock cycles are needed every second to achieve such motion estimations. The typical working frequency of OR1200 is 250MHz, satisfying the implementation of real-time motion estimation in H.264/AVC coding.

5. Conclusion

Theoretical analysis shows that, SIMD+VLIW architecture is suitable for motion estimation algorithms as well as motion compensation algorithm. It can greatly accelerate the calculation of BMA while save memory space at the same time. It also efficiently use the process element specially designed for BMA core calculation. The DSP nature of the architecture also enables programming flexibility to implement variety of BMA algorithms without doing any change to the architecture. Of course, compared with the parallel PE architecture in [12] [13], the process speed of present architecture is not enough. But, if necessary, by implementing more ILP (Instruction-Level Parallelism) such as lengthening the VLIW to 6 instructions, the process ability can be enhanced. In the future work, some BMA algorithms will be implemented into the modified architecture to get the exact performance on motion estimation of the modified architecture.

5. Acknowledgements

This work was supported by the China national high technology program (863) 2003AA1Z1100 and the China National Pivot Fundamental Research (973) Program (No. G2000036508).
6. References


Figure 1: Block matching motion estimation. Search a candidate block that yields the minimum MAD in the searching area, the corresponding candidate location (u,v) is the motion vector needed.
Figure 2: The SIMD process element for core computation of BMA. A specially designed computation unit that can fulfill one SAA calculation in one clock cycle is shown in the dashed frame. Four such units are placed in the SIMD element. Four 32-bit registers and two align logic unit supply 4 pair of 8-bit video data for the SAA unit. The result of the SIMD process is stored in two 32-bit accumulators.

AC[2:0]  
000  
001  
010  
011  
100  
101  
110  
111  

Figure 3: Align unit extracts four 8-bit data from a pair of 32-bit registers according to the value of Align Control (AC) signal.