Exploiting Coarse-Grain Parallelism in the MPEG-2 Algorithm

Eiji Iwata
Media Processing Laboratories
Sony Corporation
iwata@av.crl.sony.co.jp
and
Kunle Olukotun
Computer Systems Laboratory
Stanford University
kunle@ogun.stanford.edu

Abstract

As the demand for multimedia applications increases, the performance of algorithms such as MPEG-2 video compression on general-purpose microprocessors is becoming more important. In this paper, we propose a number of coarse-grained parallel implementations of MPEG-2 decoding and encoding. We evaluate the performance of these implementations on a single-chip multiprocessor, and compare the performance against a baseline wide issue superscalar processor. On both MPEG-2 decoding and encoding, the single-chip multiprocessor achieves significantly better performance than a wide issue superscalar processor. We also find that the coarse-grained parallelism in MPEG-2 is orthogonal to the fine-grained parallelism which can be exploited by multimedia extensions to the instruction set architecture such as Intel MMX.

keywords: MPEG-2 decoding, MPEG-2 encoding, coarse-grained parallelism, single-chip multiprocessor, multimedia extensions

1 Introduction

The increased use of multimedia applications has significantly increased the computing demands placed on desktop computers. One way to satisfy these computing demands is to augment the general-purpose microprocessor with a digital signal processor (DSP) chip [1] or media processor [2]. Another approach, favored by microprocessor vendors, is to increase the power of the microprocessor by adding multimedia instructions to the instruction set architecture (ISA). Examples of this approach are Intel MMX [3], Sun VIS [4], and MIPS MDMX [5]. Multimedia applications such as the MPEG-2 full-motion video compression algorithm contain a significant amount of both coarse and fine-grained parallelism. Multimedia instructions can exploit fine-grained parallelism in multimedia applications. However, to achieve maximum performance, it is also important to exploit the coarse-grained parallelism in these applications.

In this paper we propose the use of coarse-grained parallelism and a multiprocessor microarchitecture as a new approach for achieving high performance on multimedia applications. We describe several parallel implementations of the MPEG-2 algorithm, which is the most widely-used multimedia application. We evaluate these implementations on a simulation model of a single-chip multiprocessor. We also compare the performance of the single-chip multiprocessor with a wide-issue superscalar.

Our results show that on both MPEG-2 decompression (decoding) and compression (encoding), multiprocessors that exploit coarse-grained parallelism perform 50-130% better than a wide issue superscalar processor, which can only exploit fine-grained parallelism.

The remainder of this paper is organized as follows. In Section 2, we briefly introduce the MPEG-2 algorithm. In Sections 3 and 4, we describe several implementations of MPEG-2 decoding and encoding that exploit coarse-grained parallelism. We describe the simulation methodology used to evaluate these implementations in Section 5, and in Section 6 we show the results of our performance comparison between single-chip multiprocessors and wide issue superscalar processors. Finally, we conclude in Section 7.

2 The MPEG-2 Algorithm

MPEG-1 [7] is a lossy video compression standard which enhances still-picture compression, using the Discrete Cosine Transform (DCT) and run-length coding, with motion compensation. Motion compensation exploits temporal redundancy in the video stream and provides much higher compression ratios. An MPEG stream has a hierarchical image data structure which consists of levels organized in the following manner: video sequence, group of pictures (GOP), picture (frame), slice, macroblock, and block (see Figure 1). One of the important features of MPEG is that there are 3 types of picture, which are used for reducing the temporal redundancy. In the first picture type, called intra (I), all macroblocks in the picture are encoded without motion compensation. I-pictures are independent of other pictures and thus can provide points in the MPEG stream where decoding can start. In the second picture type, called predicted (P), in addition to intra macroblocks, some macroblocks are encoded with motion compensation based on a previous I or P-picture. In the third picture type, called bidirectionally predicted (B), there are some macroblocks which are encoded with motion compensation based on either previous or succeeding I or P-pictures.

MPEG-2 [7] is an enhanced version of the MPEG-1 standard. Several kinds of image size including HDTV and coding schemes such as spatial and temporal scalable codings are integrated and determined by profile and level. Main Profile at Main Level (MP@ML) is the most common profile and level, and can be used for wide range of applications such as digital video disk (DVD) and Broadcast Satellite Service. We use MP@ML in all our experiments.

Figure 2 shows the block diagram of an MPEG-2 encoder. The encoder stages are: Motion Estimation (ME), Forward DCT, Quantization (Q), Variable Length Coding (VLC), Rate Control and Mode Decision. The input is image data and the output is a serial bitstream.

Figure 3 shows the block diagram of an MPEG-2 decoder. The decoder stages are: Variable Length Decoding (VLD), Inverse Quantization (IQ), Inverse DCT and Motion Compensation (MC). The input is a serial bitstream and the output is image data.

The applications used for performance evaluations presented in this paper are mpeg2decode (mpeg2play) and mpeg2encode [8]. Mpeg2decode only uses integer instructions while mpeg2encode includes some floating point instructions for FDCT and Rate Control. Mpeg2play is a more optimized version of mpeg2decode; however, it does not meet the MPEG-2 standard. Therefore, we use an optimized mpeg2decode based on mpeg2play that does satisfy the MPEG-2 standard. In our experiments with MPEG-2 decoding, we used a bitstream encoded with following parameters:

· N (distance between I-pictures) = 15
· M (distance between I/P-pictures) = 3
· Frame size = 704 $\times$ 480 pixels
· Bit rate = 5Mbps

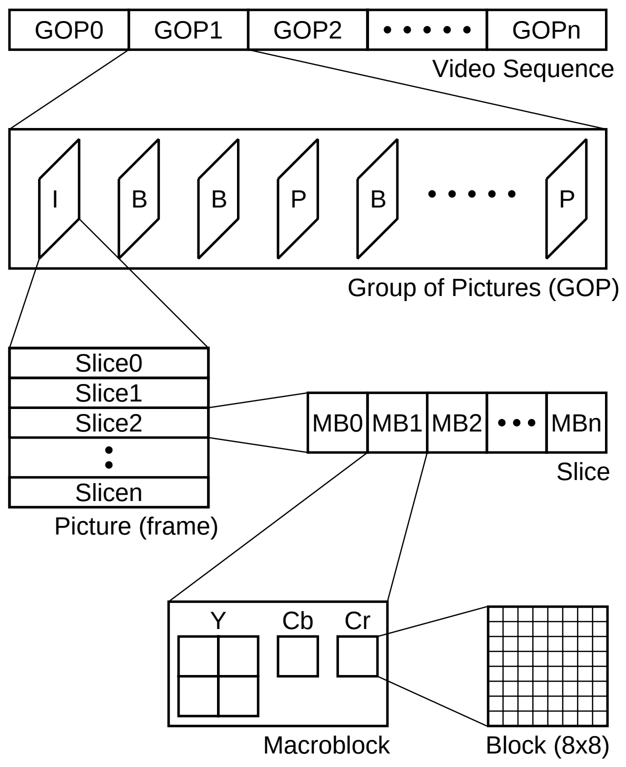

Figure 1: MPEG video stream data structure.

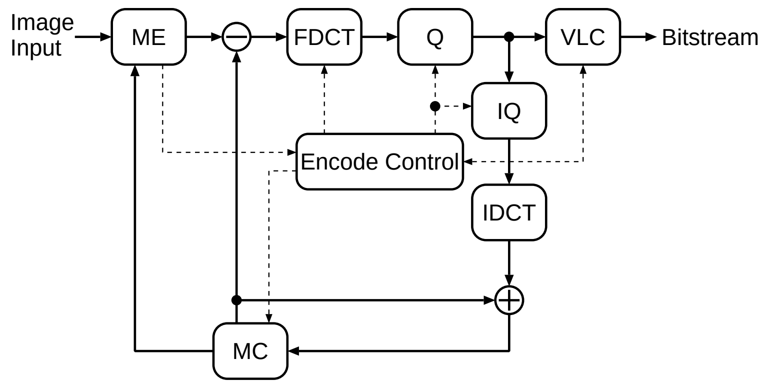

Figure 2: Block diagram of an MPEG-2 encoder.

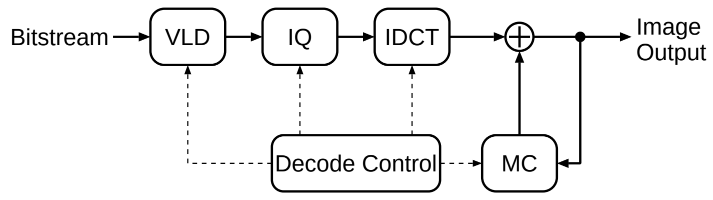

Figure 3: Block diagram of an MPEG-2 decoder.

3 Parallel MPEG-2 Decoding

We began our effort to parallelize the MPEG-2 decoding algorithm by profiling the algorithm running on a workstation to determine the execution time breakdown of the different stages of the algorithm. Figure 4 shows the percentage of execution time spent in each stage while decoding 30 frames. We lump the IQ portion of the execution time under VLD. Figure 4 shows that the portion of execution time spent in DCT, MC and VLD is roughly equal. Parallelizing the VLD stage — a challenging task because it requires sequential access to the bitstream — is required to achieve good speedup on the decoding algorithm, because approximately one fourth of execution time is spent in VLD.

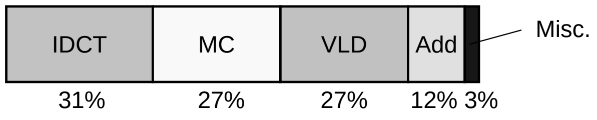

Figure 4: Breakdown of execution time for MPEG-2 decoding of 30 frames.

Figure 5 shows the configuration of an MPEG-2 bitstream and the execution sequence for a sequential implementation and the four parallel implementations we propose in this paper.

The basic strategy used in our parallel implementations of MPEG-2 decoding is to process the macroblocks or slices in parallel. We did not consider processing the pictures or GOPs in parallel, because there are many irregular data dependencies caused by motion compensation between pictures and GOPs. Although Bilas et.al. [9] describe a parallel MPEG-2 decoding algorithm at the GOP level, their method requires that the GOPs have no interdependencies, and this assumption does not hold for most MPEG-2 bitstreams. However, there are no data dependencies between slices in one picture. Therefore, each slice can be executed completely in parallel. Once the VLD is concluded for a particular macroblock, macroblock processing is also parallelizable.

One problem that arises in parallelizing MPEG-2 decoding is that since bitstreams have varying length, it is impossible to detect the start of a slice or macroblock without scanning the bitstream. It is not useful for each PE to independently examine the whole bitstream. It is more efficient for each PE to sequentially scan a separate portion of the bitstream and pass information about where to start scanning to the next PE.

The configuration of MPEG-2 bitstream is shown in Figure 5-a. In the bitstream we used in our experiments, each picture has 30 slices and each slice has 44 macroblocks. Each picture, slice or macroblock has its own header which consists of control information needed for decoding. Figure 5-b shows the sequential execution of decoding. A processor sequentially executes VLD for the slice header, VLD for macroblock header and macroblock, and then the remainder of the decoding process.

The parallel implementation at macroblock level (DM) is shown in Figure 5-c. We assume a multiprocessor with four processors. In the DM implementation, a particular PE (e.g. PE1) executes VLD for a macroblock after VLD for a slice header. When PE1 finishes VLD for one macroblock, it passes information about where to begin access in the bitstream to PE2. Both VLD for the next macroblock on PE2 and the decoding process for current macroblock on PE1 are executed simultaneously. This process continues for the other PEs so that each PE executes VLD for a macroblock sequentially, and then executes the rest of the decoding process in parallel. At the end of picture, a barrier keeps all PEs synchronized.

The disadvantage of the DM implementation is the load imbalance caused by the difference in the decoding time for different macroblocks. From Figure 4, the fraction of time spent in VLD is

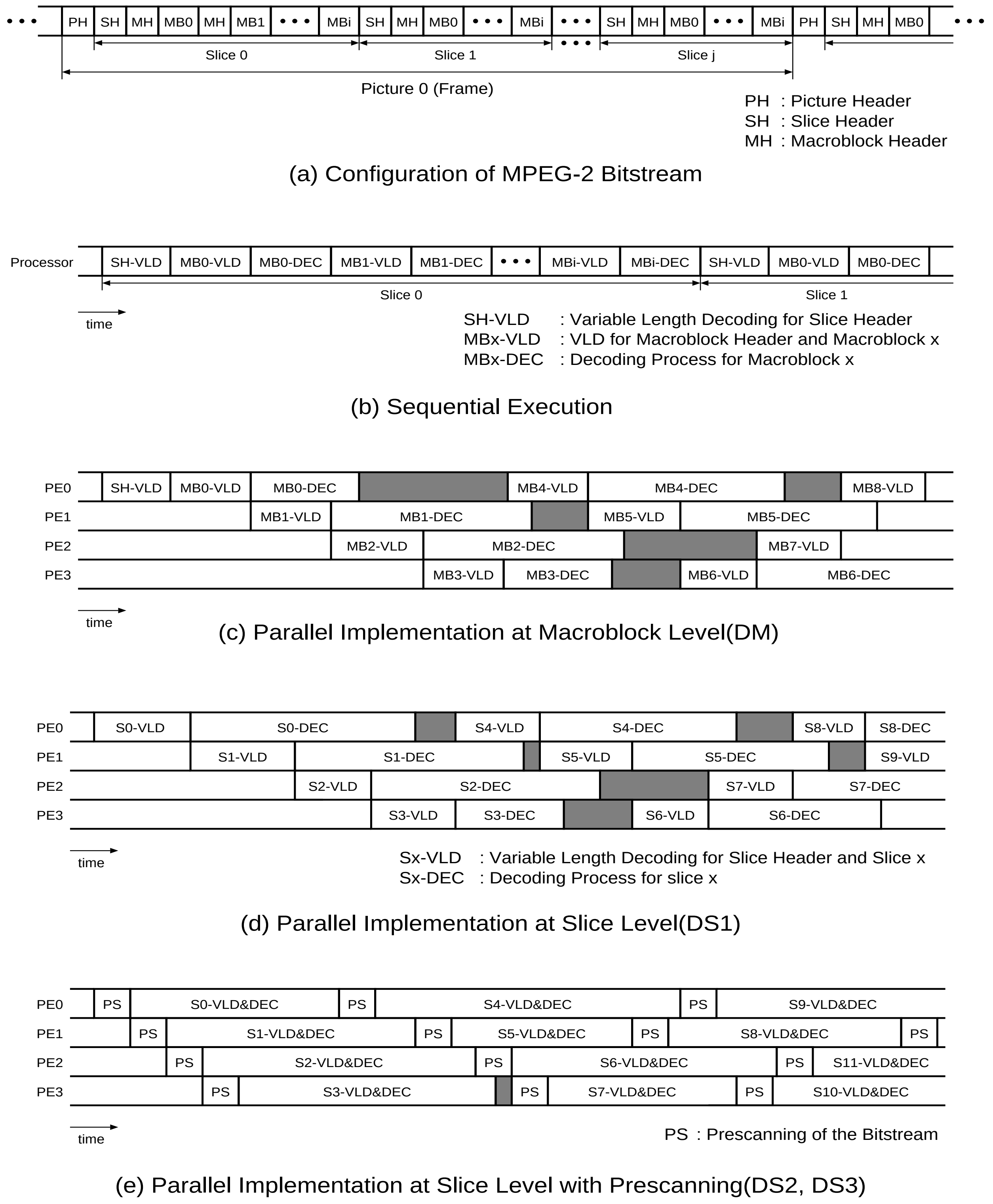

Figure 5: Execution time sequences for sequential execution and parallel implementations of MPEG-2 decoding.

approximately 27% of total execution time, but this value actually depends on macroblock mode. For example, it is not necessary to execute motion compensation for an intra macroblock. In addition, since the compression ratio of an I-picture compared to that of P/B-picture is relatively low, execution time for VLD increases for I-pictures. Therefore, the sequential execution time spent in VLD increases for an I-picture.

Figure 5-d shows the parallel implementation of MPEG-2 decoding at slice level (DS1). DS1 uses the same ideas as DM except the parallelization is done at the slice level instead of the macroblock level. The time variation for decoding for each slice is less than the time variation for each macroblock. Hence, the overhead caused by macroblock modes can be reduced. In addition, overhead of synchronization can be reduced because granularity of parallelism at the slice level is much coarser than at the macroblock level. It is possible in an MPEG-2 bitstream that one row in a picture can have more than one slice. In this case, it is better to assign the whole row to a PE to increase the granularity of parallelism by utilizing information about vertical position of slice in the slice header.

Figure 5-e shows the parallel implementation at slice level with prescanning of the bitstream (DS2,DS3). Each slice has a particular start code at the beginning of the slice header. This start code has a 24 bit value that is byte aligned. Therefore, each PE can detect this start code while scanning the bitstream and then fetch the portion of the bitstream which corresponds to one slice without VLD [9]. We call this operation prescanning. The scanning method based on an 8 bit shift is used in implementation DS2 and a method based on a 32 bit shift and conditional decision is used in implementation DS3. With DS2/DS3, the overhead caused by the time variation for decoding different macroblocks is significantly reduced, because execution time for prescanning is much smaller than the time for VLD. Each PE executes VLD for the prescanned bitstream in parallel, since only the prescanning is serialized. This method cannot be applied to macroblocks, because a macroblock header does not have a start code, so it is impossible to detect the start of a macroblock without VLD.

4 Parallel MPEG-2 Encoding

Just as in the case of decoding, we profiled the encoding algorithm to determine the breakdown of the execution time. We divided encoding algorithm into three phases of computation: P1,P2, and P3. P1 contains motion estimation(ME), FDCT, MC, subtract and mode decision. P2 contains Q, VLC, and rate control. P3 contains IQ, IDCT, and addition. Figure 6 shows the breakdown of execution time into the three computation phases for 30 frames. In order to reduce simulation time, we implemented a more sophisticated hierarchical search algorithm for motion estimation instead of the full search algorithm implemented in original program. In spite of this, 89% of total execution time is used for P1 which includes motion estimation. Note that P2 and P3 require only 6% and 5% of total execution time, respectively. As motion estimation dominates the execution time of encoding and we can easily exploit the coarse-grained parallelism in motion estimation, parallelizing encoding is less difficult than decoding.

Figure 7 shows the sequential execution of the encoding algorithm and the execution sequence of the six parallel implementations we propose. Unlike MPEG-2 decoding, there are data dependencies across macroblocks in the P2 phase, because a quantization scale and a number of generated bits of the previous macroblock are required for rate control and to calculate the quantization scale for the current macroblock. Therefore, VLC, which is contained in the P2 phase, must be sequentially executed at either the macroblock level or the slice level. Hence, the most important decision to be made in the parallelization of MPEG-2 encoding is when during the picture each PE should execute VLC.

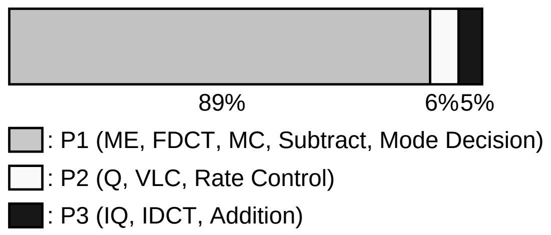

Figure 6: Breakdown in execution time in MPEG-2 encoding.

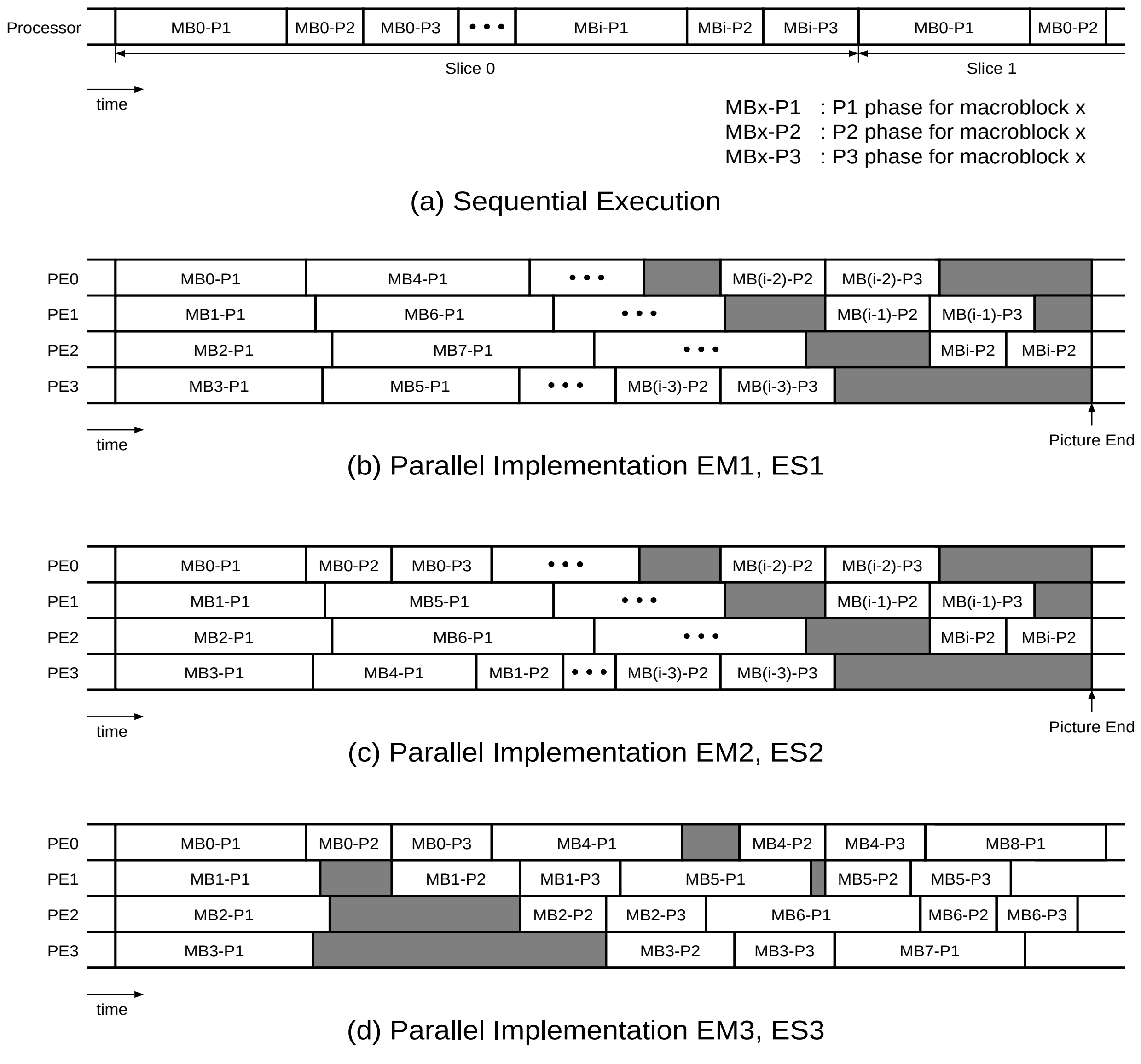

Figure 7: Execution time sequences for sequential and parallel implementations of MPEG-2 encoding.

Figure 7-a shows the sequential execution of encoding. A processor sequentially executes encoding phases for each macroblock in the order P1, P2, P3.

The first parallel implementation, EM1, is shown in Figure 7-b. In EM1, each PE sequentially executes the P2 and P3 phases after all PEs finish the execution of the P1 phases for all the macroblocks in one picture. As shown in Figure 7-b, execution of P2 and P3 phases for different macroblocks can be overlapped but the P2 phases must execute sequentially. A similar parallelization method to EM1 can be applied at the slice level. We call this implementation ES1.

For the parallel implementations EM1 and ES1 the hit rate of the I-cache is expected to improve because all P1 phases are completely separated from all the P2 and P3 phases in one picture. However, the hit rate of the D-cache is expected to degrade because the PEs assigned to a P1 phase and P2 or P3 phase for the same macroblock might be different.

Figure 7-c shows the parallel implementation EM2. EM2 is an improved version of EM1. The key difference is that at the time that a PE finishes the P1 phase for a particular macroblock, it begins to execute the P2 and P3 phases instead of going on to execute the P1 phase for the next macroblock. This better distributes the sequential execution of the P2 phases and might lead to the PEs being stalled for less time. If the P2 phase is being executed by another PE, the PE executes a P1 phase for the next macroblock as in implementation EM1. This method can be also applied at the slice level, called ES2.

The parallel implementation EM3 is shown in Figure 7-d. EM3 is a straightforward parallelization of the encoding algorithm in which each PE executes all three phases for a macroblock. Macroblocks are encoded in parallel except that the P2 phases are executed sequentially. EM3 has different cache memory characteristics than EM1 and EM2. There is a possibility of worse I-cache performance because each PE has to fetch instructions needed for the P1, P2, and P3 phases macroblock by macroblock. On the other hand, the performance of the D-cache is expected to improve because the same PE consecutively executes P1,P2 and P3 phases for one macroblock. This parallelization method can be also applied at the slice level, called ES3.

5 Simulation Methodology

We developed the parallel implementations described above for mpeg2decode and mpeg2encode, and executed them in the SimOS simulation environment [10]. SimOS models the CPUs, memory hierarchy and I/O devices of uniprocessor and multiprocessor systems in sufficient detail to boot and run a commercial operating system. SimOS uses the MIPS-2 instruction set and runs the Silicon Graphics IRIX 5.3 operating system, which has been tuned for multiprocessor performance. SimOS supports multiple CPU models. We use the most detailed model, that accurately models the details of an out-of-order superscalar microprocessor, for the results presented in this paper. We evaluated the MPEG-2 algorithm using the five microprocessor models described below.

· single issue single processor (SP1)
· 2-way issue superscalar single processor (SP2)
· 6-way issue superscalar single processor (SP6)
· 2-way issue superscalar, with 4 processors (MP4)
· single issue, with 8 processors (MP8)

In terms of chip size, SP6 and MP4 are approximately the same [11]. The key characteristics of the five models listed above are shown in Table 1.

	SP1 & MP8(8×SP1)	SP2 & MP4(4×SP2)	SP6
# of CPUs	n	n	1
Degree superscalar	1	2	6
# of architectural registers	n × 32int/32fp	n × 32int/32fp	32int/32fp
# of physical registers	n × 40int/40fp	n × 40int/40fp	160int/160fp
# of integer units	n × 1	n × 1	6
# of floating point units	n × 1	n × 1	6
# of load/store ports	n × 1	n × 1	8 (one per bank)
BTB size	n × 256	n × 512	2048
I cache	32KB/n, 2-way S.A.	32KB/n, 2-way S.A.	32KB, 2-way S.A.
D cache	32KB/n, 2-way S.A.	32KB/n, 2-way S.A.	32KB, 2-way S.A.
L1 hit time	1 cycle	1 cycle	2 cycles
L1 cache interleaving	N/A	N/A	8 banks
Unified L2 cache	256KB, 2-way S.A.	256KB, 2-way S.A.	256KB, 2-way S.A.
L2 hit time / L1 penalty	5 cycles	5 cycles	4 cycles
Memory Latency / L2 penalty	50 cycles	50 cycles	50 cycles

Table 1: Key characteristics of the three microarchitectures

		BP Rate	I cache	D cache	L2 cache
Model	IPC	%	% MPCI	% MPCI	% MPCI
SP1	0.9	78.6	0.0	1.2	0.1
SP2	1.4	79.7	0.0	1.2	0.1
SP6	2.7	80.1	0.0	3.1	0.4
MP4(DS3)	1.2	86.0	0.6	4.1	0.7
MP8(DS3)	0.8	89.4	1.0	6.3	1.8

Table 2: Performance of MPEG-2 Decoding

6 Performance Comparison

Figure 7 shows simulation results of parallel MPEG-2 decoding for the five processor models described in Section 5, each decoding a 15 frame sequence. We eliminated the file output process because it would be replaced by direct display output in a real multimedia system. We simulated the sequential program on the SP1, SP2, and SP6 models, and we simulated all the parallel implementations described in Section 3 on the MP4 and MP8 models. Table 2 shows the IPC, branch prediction rates, and cache miss rates for MPEG-2 decoding executing on the five models. The cache miss rates are presented in terms of misses per completed instruction (MPCI).

The fastest method on both MP4 and MP8 is DS3. The fact that the performance of DS2 and DS3 is better than that of DM and DS1 indicates that the overhead of prescanning is smaller than the overhead of waiting for the sequential execution of VLD. Assuming the operating frequency of processor models is 500MHz [11], it is necessary to finish the decoding process for 15 frames within 250 million cycles in order to achieve real-time MPEG-2 (MP@ML) decoding. With this assumption, real-time MPEG-2 (MP@ML) decoding is easily achieved without a multimedia extended instruction set on SP2, SP6, MP4 and MP8 (see Figure 7). Real-time MPEG-2 (Main Profile at High Level) decoding for HDTV requires performance which is approximately 6 times higher than MP@ML, so none of our architectures could perform this decoding without further enhancement.

Figure 8 shows simulation results for 6 frames of parallel MPEG-2 encoding with the five pro-

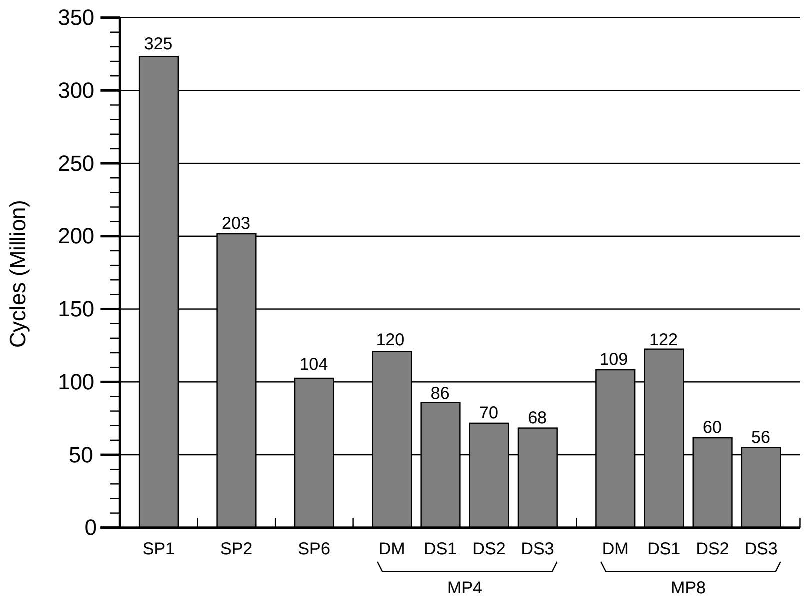

Figure 8: Results of Parallel MPEG-2 Decoding

Model	IPC	BP Rate %	I cache % MPCI	D cache % MPCI	L2 cache % MPCI
SP1	0.8	66.9	0.0	0.3	0.1
SP2	1.4	67.5	0.0	0.3	0.1
SP6	2.3	68.4	0.0	0.7	0.3
MP4(EM3)	1.4	73.3	0.1	1.0	0.1
MP8(EM2)	0.8	72.8	0.2	2.8	0.3

Table 3: Performance of MPEG-2 Encoding

cessor models. Our simulation method is the same as that of decoding. Table 3 shows the IPC, branch prediction rates, and cache miss rates for encoding on five processors. The reason for the significant degradation of the branch prediction rates is that ME contains an absolute difference operation which requires a conditional branch for each pixel.

The fastest parallel implementation on MP4 is EM3 ,while that on MP8 it is EM2. In the case of encoding, all parallel implementations on multiprocessors achieve better performance than a wide issue superscalar processor. The fact that parallelization methods at macroblock level are better than at slice level indicates that overhead of barrier synchronization between pictures is more significant than the overhead of synchronization for sequential VLC.

From the results of all implementations for decoding and encoding, we found that impact of the memory system on performance is relatively small. This indicates that the data structures used by the MPEG-2 algorithm have plenty of spatial locality and MPEG-2's working sets are small enough to fit in the primary caches even at the slice level.

The overall performance comparison among the five processor models is shown in Figure 9. The performance is measured as speedup of each model relative to the single issue single processor (SP1). A significant performance improvement is achieved for both decoding and encoding on the multiprocessor models (MP4,MP8). The speedup for parallel encoding is better than that for

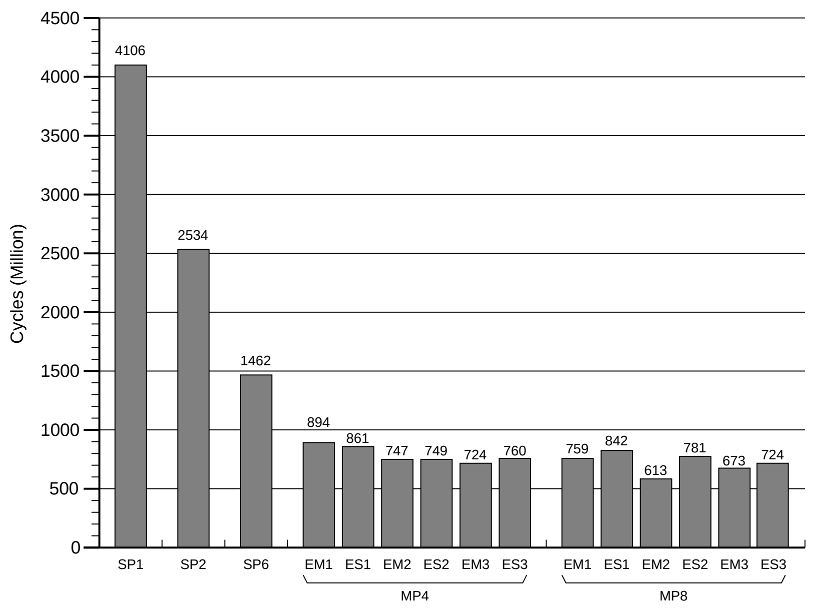

Figure 9: Results of Parallel MPEG-2 Encoding

parallel decoding, because the computational load for each macroblock is roughly equivalent due to ME, resulting in better load balance.

We also investigated the performance improvements possible from exploiting fine-grained parallelism using a multimedia extended instruction set. We found that we can obtain 50% performance improvement for MPEG-2 decoding with MMX-like multimedia extensions to the MIPS instruction set architecture. This improvement is due to the SIMD (subword parallel) processing in the IDCT and MC steps. However, it is difficult to shorten execution time for essentially sequential process such as VLD. Hence, if we combine this fine-grained approach with coarse-grained parallel implementation described in this paper, the relative increase in execution time for sequential phases of the algorithm such as VLD or VLC could cause performance degradation. However, we measured that the execution time for the sequential phase in DS2 and DS3, prescanning, is below 2% of total execution time. In the case of encoding, execution time for the P2 phase is much shorter than the P1 phase. Additionally, we set a very small search range for ME to shorten the simulation time. More realistic ME requires a much larger search range. Hence, the P1 phase would take longer in realistic encoding, and so the fraction of execution time spent in the sequential phase (P2) would be relatively small. Consequently, although we have not simulated the performance of these coarse-grained parallel implementations combined with the use of fine-grained parallelism using a multimedia extended instruction set, we believe that these forms of parallelism are orthogonal. Thus, we can obtain a combined performance improvement when coarse-grained and fine-grained are exploited together. Such performance improvements would enable real-time HDTV decoding on an architecture such as MP4 augmented with multimedia instructions.

7 Conclusion

In this study we have proposed several coarse-grained parallel implementations of the MPEG-2 algorithm. We have evaluated these implementations on single-chip multiprocessor models within the SimOS simulation environment. Our results show that on both decoding and encoding, mul-

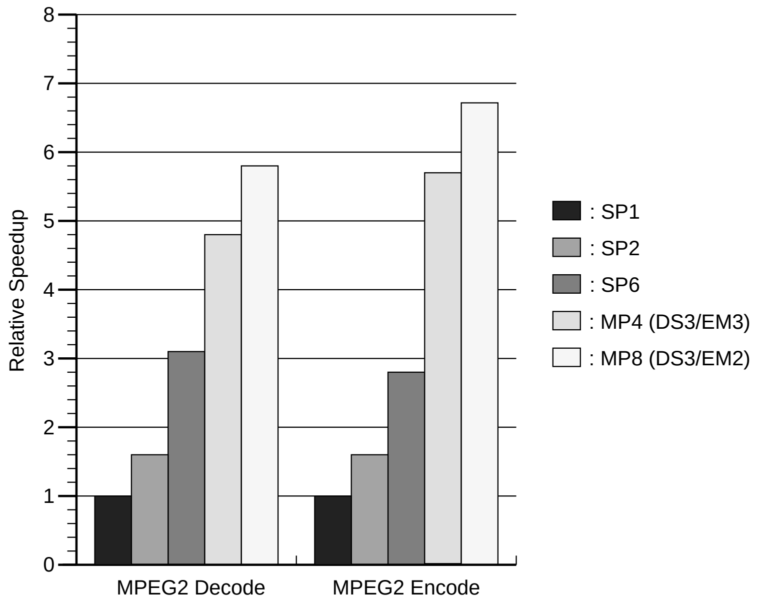

Figure 10: Performance Comparison

tiprocessors (MP4,MP8) that exploit coarse-grained parallelism perform 50-130% better than a wide issue superscalar processor (SP6), which can only exploit fine-grained parallelism.

The coarse-grained parallel MPEG-2 implementations we described in this paper are straightforward extensions of the algorithms and data structures in the sequential version of MPEG-2. The parallelization methods used do not require the programmer to have any knowledge of the underlying machine architecture beyond the idea of multiple thread support. We think that the idea of multithreading is far more intuitive than the machine-level knowledge required to take advantage of multimedia instructions. Therefore, we foresee the increased use of coarse-grain thread-level parallelism in the multimedia application domain.

Acknowledgements

We would like to thank Basem A. Nayfeh and Lance Hammond for their help with SimOS and P.A. Subrahmanyan for his insightful comments. We also appreciate the help of Takashi Miyamori with the performance evaluation of multimedia extensions to the MIPS instruction set.

References

[1] E. Iwata et.al., "A 2.2GOPS Video DSP with 2-RISC MIMD, 6PE-SIMD Architecture for Real-Time MPEG-2 Video Coding/Decoding," *IEEE International Solid State Circuits Conference Digest of Technical Papers*, pp.258-259, San Francisco, CA, 1997.

[2] Yong Yao, "Chromatic's Mpact 2 Boosts 3D," *Microprocessor Report*, vol.10, Nov., 1996.

[3] Alex Peleg, Uri Weiser, "MMX Technology Extension to the Intel Architecture," *IEEE Micro*, pp.42-50, Aug., 1996.

[4] Marc Tremblay, et.al., "VIS Speeds New Media Processing," *IEEE Micro*, pp.10-20, Aug., 1996.

[5] Earl Killian, "Extending the MIPS Instruction Set for Digital Media and Emerging Applications," *Microprocessor Forum 96*, San Jose, CA, Oct., 1996.

[6] Peter Bannon, "Enhancing Motion Video Performance in the Alpha Architecture," *Microprocessor Forum 96*, San Jose, CA, Oct., 1996.

[7] V. Bhaskaran, K. Konstantinides, *Image and Video Compression Standards.* Norwell, Massachusetts: Kluwer Academic Publishers, Inc., 1995

[8] MPEG Software Simulation Group, mpeg2encode/mpeg2decode (mpeg2play) version 1.2, http://www.mpeg.org/MSSG/, July, 1996

[9] Angelos Bilas, Jason Fritts, Jaswinder Pal Singh, "Real-Time Parallel MPEG-2 Decoding in Software," Princeton University, TR-516-96.

[10] M. Rosenblum, S. Herrod, E. Withchel, and A. Gupta,"The SimOS approach," IEEE Parallel and Distributed Technology, vol.4, no.3, 1995.

[11] K. Olukotun et.al., "The Case for a Single-Chip Multiprocessor," *Proceedings of 7th Int. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS-VII)*, Oct., 1996.