Avoiding Data Conversions in Embedded Media Processors

Ben Juurlink
benj@ce.et.tudelft.nl

Asadollah Shahbahrami
shahbahrami@ce.et.tudelft.nl

Stamatis Vassiliadis
stamatis@ce.et.tudelft.nl

Computer Engineering Laboratory
Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology, The Netherlands
Phone: +31 15 2787362, Fax: +31 15 2784898.

ABSTRACT

Complex application-specific media instructions and kernels are emulated with simple to implement extended subword instructions. We show that assuming extended register file entries to accommodate intermediate results and by implementing a few simple instructions, packing/unpacking, saturation, and frequently used complex instructions can be practically eliminated. It is shown that in most emulations there is a potential performance improvement, making the proposed scheme suitable for embedded processors with a limited hardware budget.

Categories and Subject Descriptors

C.1.2 [**Processor Architectures**]: Multiple Data Stream Architectures—*SIMD*

General Terms

Design, Performance

Keywords

Multimedia applications, SIMD instructions.

1. INTRODUCTION

Microprocessor vendors have developed multimedia instructions in order to exploit the computational characteristics of multimedia applications. Some of these ISA extensions include complex instructions like Sum-of-Absolute-Differences (SAD) and averaging. Additionally, for precision and data representation reasons, they use saturation and pack/unpack instructions. In this paper we consider eliminating these instructions using wider registers, having in mind the design of embedded processors with a limited hardware budget.

We investigate the possibilities of using simple instructions to emulate complex instructions with a Modified MultiMedia eXtension (MMMX). To determine the effectiveness of our proposal we consider ten frequently used multimedia kernels. We have established that for eight of the ten kernels the dynamic number of instructions is reduced significantly (up to 2.7x) using the proposed approach. For only two kernels the dynamic number of instructions increases, by 1.4x and 1.2x, respectively.

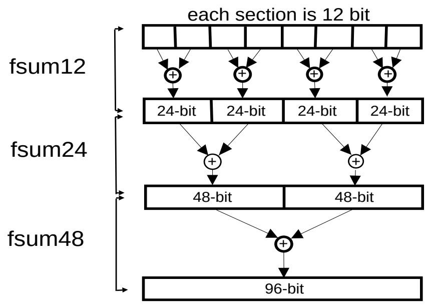

Figure 1: The structure of the three `fsum` instructions in MMMX.

2. MODIFIED MMX

For reasons of space, we cannot present the MMMX instruction set architecture. The most important difference to MMX is that the MMMX multimedia registers are 96 bits wide instead of 64 bits. In other words, for every byte of data there are four bits of extra precision. This allows many computations to be performed without overflow. According to [1], a 12-bit data format is sufficient for 85.7% of the processing in MPEG-4 encoding. In addition, MMMX provides some instructions which can be used to emulate more complex MMX instructions. For example, Figure 1 illustrates how eight 12-bit values in a register can be accumulated using the `fsum12`, `fsum24`, and `fsum48` instructions.

3. SUMMARY OF RESULTS

We have implemented ten kernels. Table 3 lists all the kernels that we have implemented using MMMX and MMX/SSE[1]. The table shows the storage formats of the input and output data, the maximum data size during intermediate computations, the dynamic number of instructions required when the kernel is implemented in MMX, and the dynamic number of instructions required for MMMX. When the kernel is typically performed on fixed-size blocks (such as DCT and SAD) we present the number of instructions required for that in-

[1]The MMX/SSE implementations of DCT, IDCT, and block match are due to Slingerland and Smith [3].

Table 1: Dynamic number of instructions required to implement the considered kernels in MMX and MMMX, the storage format, and the maximum data size during computation.

Kernel	Storage format	Max. data size	#instr. MMX	#instr. MMMX
Sum of abs. diff.	unsigned byte	9 bits	194	274
Sum of Squared diff.	unsigned byte	24 bits	582	260
DCT	16-bit signed	24 bits	909	553
IDCT	16-bit signed	24 bits	685	641
Arithmetic avg.	unsigned byte	9 bits	176	208
Block match vert. and hor. interpolation	unsigned byte	10 bits	1037	573
Add block	unsigned byte	9 bits	170	106
FIR	unsigned byte	12 bits	$44 \cdot N \cdot M/4$	$34 \cdot N \cdot M/8$
Color space conv.	unsigned byte	12 bits	$N \cdot (125 \cdot M/8 + 3) + 19$	$N \cdot (74 \cdot M/8 + 3) + 19$
fade-in-fade-out	unsigned byte	12 bits	$3 \cdot N \cdot M + 5 \cdot N + 19$	$9 \cdot N \cdot M/8 + 5 \cdot N + 19$

put size. If there is not a typical input size, we present the runtime formula.

The SAD kernel is used for motion estimation and is very time-consuming. In this kernel the difference between corresponding pixels is computed. This difference can be 9 bits wide which implies that an 8-bit data format is insufficient. Some media extensions support a SAD instruction. This is a very special-purpose instruction, however, that has limited usefulness except for this kernel. In our MMMX implementation, the SAD is synthesized using the `fsum` instructions described in Section 2. This increases the dynamic number of instructions by 1.4x but this seems reasonable given that the SAD is emulated with simpler instructions.

The Sum of Square Differences is similar to the SAD kernel except that the squared difference is used instead of the absolute difference. For this kernel, MMMX reduces the dynamic number of instructions by 2.2x.

The *Discrete Cosine Transform* (DCT) and its inverse (IDCT) are broadly used in image/video compression applications such as JPEG/MPEG. These applications partition the input image into blocks of 8×8 and perform a two-dimensional (2D) DCT on each block. We have implemented the LLM algorithm [2]. The dynamic number of instructions needed to perform a 2D DCT is 909 for MMX versus 553 for MMMX (reduction of 1.6x). Moreover, MMMX requires fewer and less wide multiply-add operations. For the 2D IDCT the reduction is smaller (1.1x) because the input data is 12-bit and intermediate results are wider than 12-bit.

Arithmetic average is used for horizontal or vertical interpolation in the block match kernel of MPEG. In MMX it can be implemented using the special-purpose `pavgb` instruction. The usefulness of `pavgb` is limited, however. For example, it cannot be used for both horizontal *as well as* vertical interpolation. In MMMX `pavgb` can be synthesized using add and shift operations. This slightly increases the dynamic number of instructions (by 1.2x), but this seems reasonable given that add and shift operations are more general and simpler to implement.

The block match kernel performs the SAD but also horizontal and vertical interpolation. In this case the MMX code cannot employ the `pavgb` instruction because that would result in rounding errors. For this kernel MMMX reduces the dynamic number of instructions significantly (by 1.8x) even though the MMX code employs the SAD instruction.

The add block kernel adds two 8×8 blocks of pixels. One block consists of unsigned bytes and the other is stored as signed 16-bit values but in reality these values are 9-bit. Consequently, MMMX reduces the dynamic number of instructions by 1.6x.

The Finite Impulse Response (FIR) filter has many applications in image processing. For an $N \times M$ image, MMMX requires $34 \cdot N \cdot M/8$ instructions to be executed whereas MMX requires $44 \cdot N \cdot M/4$ instructions, a reduction of 2.6x.

Another kernel that we have implemented is RGB-to-YUV color conversion. For large images, MMMX reduces the dynamic number of instructions by 1.7x. In addition, MMMX requires only half as many multiplications as MMX.

The last kernel implements a fade-in-fade-out effect. For this kernel, MMMX achieves a reduction of the dynamic number of instructions by 2.7x.

4. RELATED WORK

Our approach is similar to the approach described in [3] in that we use extra wide registers to represent intermediate results. The major differences are the instruction set we use and the fact that we use 96- rather than 192-bit registers. In addition, we have evaluated the efficacy of the approach by implementing media kernels using the proposed ISA.

5. CONCLUSIONS

In this paper we have proposed simple instructions that process extended data to eliminate saturation, (un)packing, and some complex instructions. We have proposed the Modified MMX (MMMX) ISA and evaluated it by implementing several frequently-used multimedia kernels using MMMX. The results show that MMMX reduces the dynamic number of instructions significantly (up to 2.7x) compared to MMX for eight out of ten kernels. For two kernels the dynamic number of instructions increases, but this is because complex, special-purpose instructions were synthesized using simple, general-purpose instructions in MMMX. This strongly indicates that the proposed scheme is suitable for inexpensive hardware embedded processors.

6. REFERENCES

[1] Dasu A. and Panchanathan S. Reconfigurable Media Processing. *Parallel Computing*, 28(7):1111–1139, 2002.

[2] Loeffler C., Ligtenberg A., and Moschytz G. S. Practical Fast 1-D DCT Algorithms With 11 Multiplications. In *Proc. Int. Conf. on Acoustical and Speech*, volume 2, pages 988–991, 1989.

[3] Slingerland N. and Jay Smith A. Measuring the Performance of Multimedia Instruction Sets. *IEEE Trans. on Computers*, 51(11):1317–1332, 2002.