A 16-BIT PARALLEL MAC ARCHITECTURE FOR A MULTIMEDIA RISC PROCESSOR

Ichiro Kuroda, Eri Murata, Kouhei Nadehara, Kazumasa Suzuki†
Tomohisa Arai††, Atsushi Okamura††
C&C Media Research Laboratories, NEC Corporation
Silicon Systems Research Laboratories, NEC Corporation†
Microcomputer Division, NEC Corporation††
1-1, Miyazaki 4-chome, Miyamae-ku, Kawasaki 216 JAPAN

Abstract - This paper presents a parallel MAC(multiply-accumulation) architecture designed for DSP applications on a 200-MHz, 1.6-GOPS multimedia RISC processor. The datapath architecture of the processor is designed to realize parallel execution of a data transfer and SIMD parallel arithmetic operations. SIMD parallel 16-bit MAC instructions are introduced with a symmetric rounding scheme which maximizes the accuracy of the 16-bit accumulation. This parallel 16-bit MAC instruction on a 64-bit datapath is shown to be efficiently utilized for DSP applications such as the convolution in the multimedia RISC processor. By using the parallel MAC instruction with the symmetric rounding scheme, the 2D-IDCT which satisfies the IEEE1180 can be implemented in 202 cycles.

INTRODUCTION

Recent microprocessors employ new instruction set to accelerate the performance for multimedia applications. The typical implementation divides a long word ALU datapath into several small word ALU's, and several pieces of independent short-word data can be processed by a single instruction. This approach was first introduced in graphics instructions for RISC microprocessors for embedded applications as well as those for workstations [1] assigning four short words for R, G, B and α.

Recent microprocessors for personal computers also employ this approach increasing the processing capability for video decompression and other DSP applications [2]. The 64-bit coprocessor(floating point) datapath realizes parallel operations for short words such as four SIMD parallel 16-bit arithmetic operations. Parallel multiplication or multiply-accumulation instructions as shown in Fig. 1 are implemented on the datapath to enhance the performance for DSP applications. However, as usual in parallel architectures, it is not easy to achieve four times DSP performance enhancement by fully utilizing these parallel function units. This requires considerations in the architecture and the instruction set design as well as those in software/firmware design that implements DSP applications.

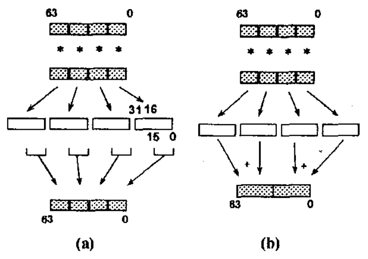

Figure 1: Parallel Multiplication/MAC instructions

One of the design issues is the limitation of the register word length to store multiplication/ multiplier-accumulation results, which require double word length of each operand word length. That is, if the 64-bit coprocessor register file is used as the accumulator for the four SIMD parallel 16-bit MAC operations, only 16-bit part of the 32-bit result can be used for each MAC operations. This limitation introduces rounding error which may be critical for DSP applications.

Another design issue is the DSP algorithm mapping into four SIMD parallel MAC units providing enough data bandwidth on the microprocessor datapath. As in the programmable DSPs, data transfer operations with address generations should be realized in parallel with the arithmetic operations to keep each unit working every cycle. Moreover, providing data set for four parallel MAC units on the implementations of DSP algorithms require architectural consideration especially for the algorithms like convolution. Word shift operations in the convolution calculation require unaligned data load of packed four 16-bit data.

This paper provides a solution which tries to solve these problems to realize a high performance DSP architecture on a multimedia RISC processor datapath. In the following sections, a datapath architecture and a media instruction set introduced in a multimedia RISC processor V830R[3] will be described first with the considerations for the above design issues. Second, parallel MAC instructions with 16-bit accumulations are introduced with a novel rounding scheme. Then algorithm mapping of typical DSP operations such as convolution and 2D-IDCT(Two dimensional inverse discrete transform) are described, which demonstrate the performance of the architecture for DSP applications. Especially for 2D-IDCT, 16-bit arithmetic accuracies realized in the architecture are shown to be enough for MPEG decoder implementations[4][5].

A PARALLEL DATAPATH ARCHITECTURE FOR DSP APPLICATIONS

The datapath architecture of the multimedia RISC processor V830R is shown in Fig. 2. The datapath consists of an integer unit and a media unit. The integer unit is a 32-bit RISC microprocessor datapath which has a 32-bit x 32 integer register file, whose instruction set was introduced in the previous generations of V800 series[6]. The media unit is a 64-bit coprocessor which has a 64-bit x 32 media register file. This large register file contributes in realizing efficient parallel algorithm implementations using parallel-pipelined media instructions, which require to use a number of intermediate variables. The instruction set for the media unit includes SIMD parallel arithmetic operations, shuffle operations as well as data transfer operations.

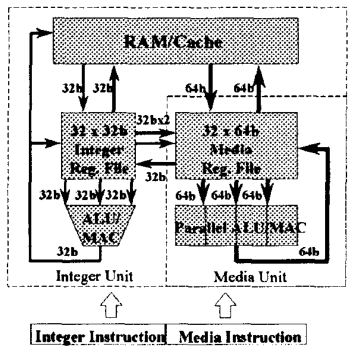

Figure 2: Datapath Architecture

The processor employs a superscalar control (in-order issue/out-of-order completion) which issues instructions for the integer unit and the media unit in parallel. 64-bit aligned load and store instructions for the media register file as well as data move instructions between the media register file and the integer register file are issued as the instructions for the integer unit. This enables the parallel instruction issue of an arithmetic instruction in the media unit and a data transfer instruction for the media register file like in programmable DSPs.

SIMD Parallel 16-bit MAC Instructions

V830R instruction set provides two solutions for the restriction of the result word width on the implementation of multiply/multiply-accumulate instructions. One solution is to have two SIMD parallel 16-bit multiply-accumulate instructions to realize four 32-bit accumulations in two cycles, which is shown in Fig. 3. Although it takes two cycles for four parallel MAC operations, enough accumulator word length is ensured.

Another solution is to have SIMD parallel 16-bit multiply-accumulate instructions with 16-bit accumulations. This approach include instructions which extract the lower 16-bit data as well as the higher 16-bit data, which is shown in Fig. 4(a)(b). For the case with Fig. 4(b), simply truncating the 32-bit data in the accumulator into 16-bit data may introduce the truncation error accumulation.

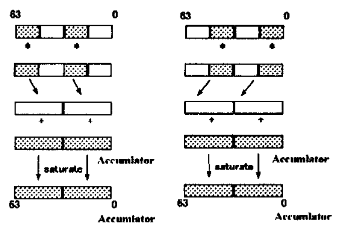

Figure 3: 32-bit accumulation

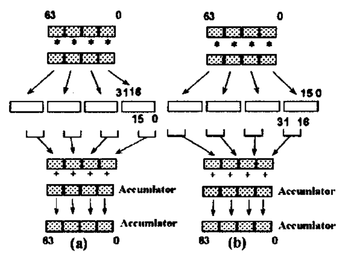

Figure 4: 16-bit accumulation

16-bit MAC instructions with symmetric rounding

In order to minimize the truncation error and maximize the precision using the 16-bit accumulation, a symmetric rounding scheme for the saturated 1-bit left shift data of the multiplication result is introduced. This 1-bit left shift is effective to maximize the result precision, especially when the result of the multiply-accumulation is further used as the input for another multiply-accumulate operations. Figure 5 shows the symmetric rounding scheme which rounds to the positive and the negative infinity according to the sign bit of the multiplication result. This rounding scheme is realized by adding 0x4000 if the multiplication result is positive and adding 0x3fff if the multiplication result is negative. This symmetric rounding achieves the reduction of the biased error accumulation compared with the truncation or the ordinary rounding scheme which adds 1 next to the result LSB before truncation. The effects of this rounding scheme is verified by the comparison with the other rounding scheme in the following section. Two types of SIMD parallel MAC instructions with the 16-bit symmetric rounding are introduced, one is a vector MAC instruction and the other is a scalar MAC instruction shown in Fig. 6.

EFFICIENT PARALLEL IMPLEMENTATION FOR DSP ALGORITHMS

Implementation of Convolution Function Using Parallel MAC

For the DSP algorithm implementations using the SIMD parallel arithmetic in the media unit, unaligned 64-bit data loads that cross 64-bit boundaries to the

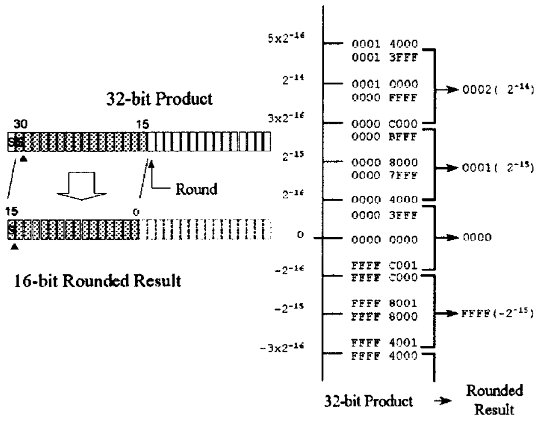

Figure 5: Symmetric Rounding

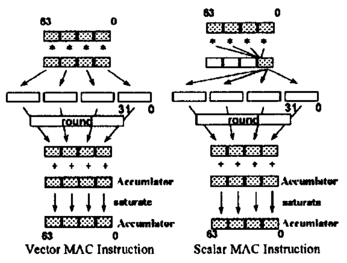

Figure 6: Vector MAC and Scalar MAC with rounding

media registers are necessary in many cases such as calculating convolution functions shown in the equation,

$$y_n = \sum_{i=0}^{M} w_i x_{n+i} = \sum_{k=0}^{M/4} W_k X_{k,n}$$

, where

$$W_k = (w_{4k}, w_{4k+1}, w_{4k+2}, w_{4k+3})$$

$$X_{k,n} = (x_{n+4k}, x_{n+(4k+1)}, x_{n+(4k+2)}, x_{n+(4k+3)})$$

.

The straight forward implementation for the calculation of convolution function requires one unaligned data load for X and one aligned data load for W for each vector MAC instruction. This number of data load can be reduced by calculating multiple output sample using the same data (W) as shown in Fig. 7. However, efficient implementation of the unaligned data load is necessary to fully utilize the parallel MAC units.

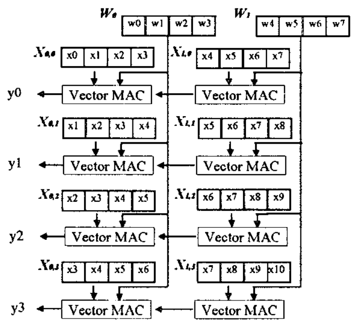

Figure 7: Convolution using vector MAC

Unaligned data load using the integer unit

The parallel issue for the integer unit and the media unit contributes in the efficient unaligned data load to the media register. As shown in Fig. 8, the unaligned 64-bit data load to the media register is realized using the following steps:

(1)Loading 32-bit word data to the 32-bit integer registers (ex.(0,1)(2,3)(4,5)).

(2)Construct 32-bit word unaligned data (ex.(1,2)(3,4)) using two aligned data. This is realized by using double word shift instructions in the integer unit which can extract any 32-bit data from the concatenation of two integer registers which is shown in Fig. 9.

(3)Move two 32-bit data in the integer registers to a media register to construct a shifted 64-bit word. (ex.(1,2,3,4))

Each 32-bit word acquired in the step (1) and step (2) is used twice to construct a 64-bit data that are send to the media registers in step (3). For example in Fig. 8, (3,4) is used to construct (1,2,3,4) and (3,4,5,6). N consecutive unaligned/aligned 64-bit words as shown in Fig. 8 can be transferred to the media registers with N/2 32-bit load instructions(1), N/2 double word shift instructions(2) and N move instructions(3). That is, each unaligned/aligned 64-bit word can be transferred to the media register every two cycles by using instructions in the integer unit.

Convolution with SIMD parallel MAC instruction

The performance for the convolution operation using parallel MAC instructions can be increased by the unaligned data loads to the media unit using the integer unit, which can be realized every two cycles as shown above. For example, the case with Fig. 7 which calculates four output sample at the same time, 9 integer instructions for data load(one for aligned load (W) and 8 for unaligned load (X)) is required for four vector MAC instructions.

By increasing the number of output samples calculating concurrently, the performance can further be increased by the reuse of the unaligned load data

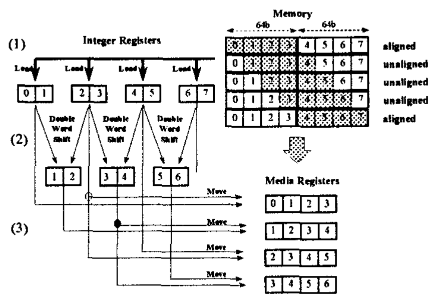

Figure 8: Unaligned Media Register Data Load Using Integer Unit

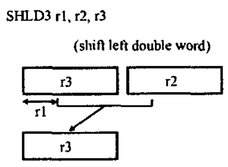

Figure 9: Double word shift left instruction

(X). The case with calculating eight output samples, 9 integer instructions for data load (one for aligned load (W) and 8 for unaligned load (X) is required for eight vector MAC instructions, because unaligned data can be used twice as shown in the Fig. 10. Table 1 shows the relation between the number of output samples and the utilization of parallel MAC units.

Table 1: Utilization of parallel MACs

Number of concurrently calculated samples	1	4	8	12
Number of parallel MAC instructions	0.25M	M	2M	3M
Number of instructions for Load	0.75M	2.25M	2.25M	2.25M
MAC utilization	33%	44%	89%	100%

2D-IDCT Using Parallel MAC With 16-bit Accumulations

An 8x8 point 2-D IDCT can be realized by eight 1-D IDCTs for rows followed by eight 1-D IDCTs for columns. SIMD parallel instructions including the scalar MAC instruction shown in Fig. 6 realize four parallel 8-point one-dimensional IDCTs for rows and columns. One dimensional IDCT for eight rows or eight columns can be realized by repeating this twice. A matrix transposition which can be realized by shuffle instructions is needed in be-

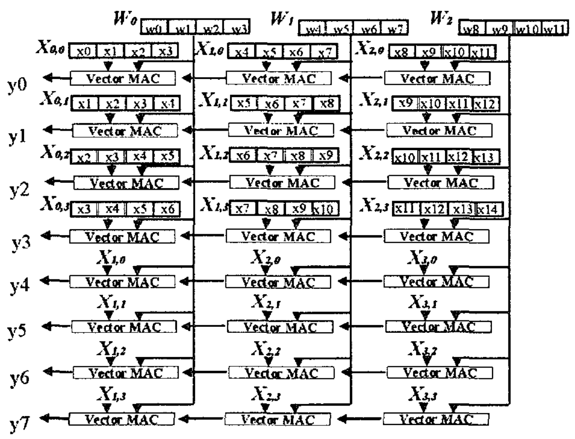
Figure 10: Convolution using SIMD parallel MAC instructions

tween IDCTs for rows and columns. The eight point 1-D IDCT algorithm using MAC operations can be realized in 36 cycles by using the algorithm shown in Fig. 11.

The IDCT calculation error using the proposed symmetric rounding multiply-accumulate instructions are verified by the simulation. Three rounding schemes such as the truncation, the ordinary rounding and the symmetric rounding are compared both without (method A) and with (method B) 1-bit shift left of the multiplier output shown in Fig. 12.

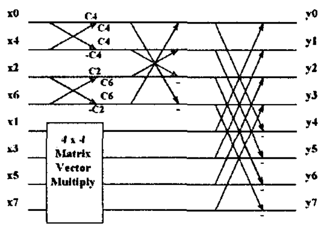
Figure 11: 8-point IDCT

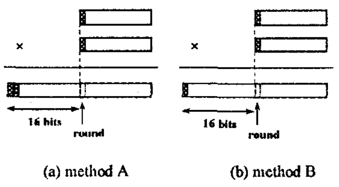

Figure 12: Rounding schemes

The three rounding schemes are applied for the method A and B at each multiply-accumulate operations (hereafter we call it m1) as well as the rounding at the final 9-bit word extractions to get 2-D IDCT output(m2) where the rounding MAC instruction can also be applied. They are also compared with IEEE1180 standard shown in Table 2 which defines the required precision for the 2D-IDCT for MPEG decoders[7] [8].

Table 2: IEEE1180

	Wst. Peak Err.	Wst. Peak MSE	Ov. MSE	Wst. ME	Ov. ME
IEEE1180	1	0.06	0.02	0.015	0.0015

(Wst.:Worst, Ov.:Overall, MSE: Mean Square Error, ME: Mean Error)

Fig. 13 shows the comparisons of the computational error using the truncation scheme for four cases. They are (1) the truncation at m1 and m2 using the method A, (2) the truncation at m1 and the symmetric rounding at m2 using the method A, (3) the truncation at m1 and m2 using the method B, (4) the truncation at m1 and the symmetric rounding at m2 using the method B. None of these cases do not satisfy the IEEE1180 standard (5).

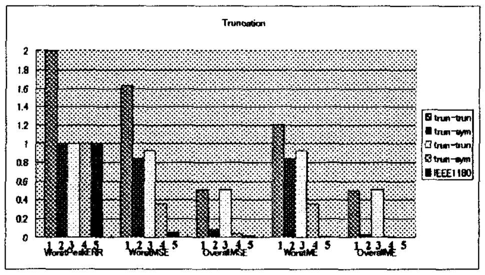

Figure 13: IDCT error for truncation schemes

Fig. 14 shows the comparisons of the computational error using the rounding schemes for four cases using the method A and the method B. For the rounding using the method A, four cases are compared with IEEE1180. They are (1) the ordinary rounding at m1 and m2, (2) the ordinary rounding at m1 and the symmetric rounding at m2, (3) the symmetric rounding at m1 and the ordinary rounding at m2, (4) the symmetric rounding at m1 and m2. All cases using the method A do not satisfy overall mean square error criterion of IEEE1180 (5). For the rounding using the method B, if the symmetric rounding is used at the final rounding, both the ordinary rounding and the symmetric rounding can be used for multiply-accumulate operations to satisfy IEEE1180.

Above simulations show that the MAC instruction with the 16-bit accumulation can be used for the 2-D IDCT calculation satisfying IEEE1180 by employing the symmetric rounding with the method B. With this four SIMD parallel MAC instruction, core processing part of an 8x8 2D-IDCT which satisfy IEEE1180 can be realized in 202 cycles including matrix transpose in the datapath architecture described in this paper.

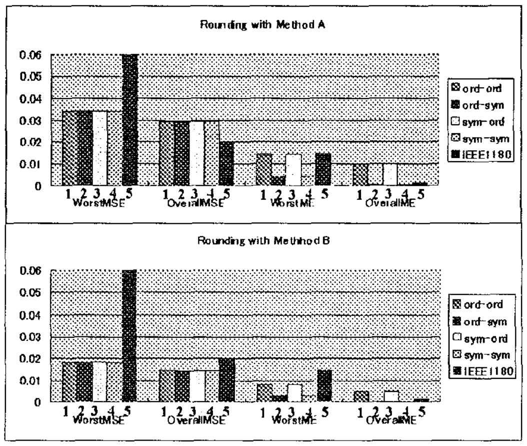

Figure 14: IDCT error for rounding schemes

CONCLUSIONS

A parallel MAC architecture for the media unit on a multimedia RISC processor is presented. A symmetric rounding scheme which maximizes the accuracy of the 16-bit accumulation is introduced for the four parallel 16-bit MAC instructions. The 64-bit unaligned data load is shown to be realized in parallel with the arithmetic operations in the media unit which can efficiently realizes calculations for convolution operations. Arithmetic precision of the 2-D IDCT using a proposed four SIMD parallel 16-bit symmetric rounding MAC is shown to satisfy the IEEE1180 standard.

References

[1] M. Tremblay, et al., "The Design of the Microarchitecture of UltraSPARC^{TM}-I ", Proceedings of The IEEE, Vol. 83, No. 12, Dec. 1995.

[2] Alex Peleg, Uri Weiser, "MMX Technology Extension to the Intel Architecture ", IEEE Micro, Vol.16, No.4, Aug. 1996, pp.42-50.

[3] T. Arai, et al.,"V830R/AV : Embedded Multimedia Superscalar RISC Processor with Rambus Interface", HOTCHIPS IX, Aug. 1997, pp.177-189.

[4] ISO/IEC JTC1 CD 11172, "Coding of moving pictures and associated audio for digital storage media up to 1.5 Mbits/s", Int. Org. Standard.(ISO), 1992.

[5] ISO/IEC JTC1/SC29/WG11, Information Technology-Generic coding of Moving Pictures and Associate Audio Information: Video, Recommendation H.262, ISO/IEC 13818-2, 1994.

[6] K. Nadehara, et al., "Low-Power Multimedia RISC", IEEE Micro, Vol.15, No.6, Dec. 1995, pp.20-29.

[7] IEEE Std 1180-1990,"IEEE Standard Specification for the Implementation of 8 by 8 Inverse Discrete Cosine Transform",1990.

[8] ISO/IEC 13818-2:1996/Cor.2 1996(E) in Annex A,"Inverse discrete cosine transform ".