A LOW-POWER PROGRAMMABLE DSP CORE ARCHITECTURE FOR 3G MOBILE TERMINALS

Takahiro KUMURA†, Daiji ISHII†, Masao IKEKAWA†, Ichiro KURODA† and Makoto YOSHIDA‡

† Computer and Communication Media Research Laboratories, NEC Corporation
‡ 1st System LSI Div. System LSI Operations Unit, NEC Corporation

ABSTRACT

We have developed a new-generation, general-purpose digital signal processor (DSP) core with low power dissipation for use in third-generation (3G) mobile terminals. The DSP core employs a 4-way VLIW (very long instruction word) approach, as well as a dual-multiply-accumulate (dual-MAC) architecture with good orthogonality. It is able to perform both video and speech codec for 3G wireless communications at 384 k bit/sec with a power consumption of approximately 50 mW. This paper presents an overview of both the DSP core architecture and a DSP instruction set, and it also gives some application benchmarks.

1. INTRODUCTION

3G wireless standards will make it possible to achieve multimedia communications at faster bit rates, up to 2 M bit/sec. In Japan, new wireless communication systems based on 3G wireless standards will be launched into service by the year 2001, and their initial transmission services will support up to 384 k bit/sec unrestricted digital data [1]. This wide bandwidth will allow us to deliver audio and visual entertainment and to communicate via both video images and voice. Although voice communications are the dominant service in existing mobile phone systems, video communications (e.g., MPEG-4) and fast digital data transmission through the Internet may be expected to be the leading services in 3G systems.

MPEG-4 is a promising standard for audio and visual communications over wireless networks [2]. For efficient implementation of this standard, subsets of the MPEG-4 systems have been grouped into profiles, and a number of approaches to the implementation of a low-resolution profile (simple profile level 1, SP@L1, 176x144 pixels) have been reported for use in MPEG-4 video communications on low-power mobile terminals [3–6]. While the small display size supported by SP@L1 may be satisfactory for a small number of initial users, the need for a larger display size, e.g., simple profile level 2 (SP@L2, 352x288 pixels), can surely be anticipated. Further, noise suppression and echo cancellation will also be required for comfortable video communications.

Moreover, in addition to multimedia applications, 3G mobile terminals will need to be able to perform modem processing (e.g., rake combining, channel equalization, and forward error correction (FEC)), and all this will require a larger amount of signal processing power than can be provided by existing terminals, at the same time that 3G terminals will have to keep power dissipation as low as that of existing terminals.

In low power applications, while dedicated-design approaches may produce more efficient implementation than do programmable-DSP approaches, programmable DSPs have an advantage in their flexibility and shorter development periods. Since 3G wireless standards are still evolving, the advantage of DSP programmability is very important, as is the need to be able to support low power, high performance, and high flexibility [7, 8].

To meet this demand, we have developed SPXK5 (a development code name), a new-generation 4-way VLIW-type DSP core with low power dissipation for use in 3G mobile terminals. In this paper, we present an overview of both the new DSP core architecture and a DSP instruction set, and we also give examples of possible multimedia and modem processing applications: specifically, MPEG-4 codec, Viterbi decoding, and adaptive filtering. Finally, we describe the results of an evaluation of the core in actual implementation.

2. ARCHITECTURE

Over the past few years, a number of manufacturers have announced the development of high performance DSPs that employ a VLIW-type architecture [8]. While a VLIW-type architecture provides their DSPs with the high performance needed for multimedia and modem processing applications, power dissipation is not low enough to be practical for mobile terminals. We, however, have succeeded in designing a VLIW-type architecture for a DSP capable that is practical for use in 3G mobile terminals. The newly developed DSP core, SPXK5, has the advantage of well-balanced instruction level parallelism based on a VLIW-type, general-purpose register architecture.

Figure 1 illustrates SPXK5's seven functional units: two MAC units, two arithmetic logical units (ALUs), two data address units (DAUs), and a system control unit (SCU). Up to four units among these seven units can work during the same clock cycle. The MACs execute 16- x 16-bit multiply and 40(16)-bit + 16- x 16-bit multiply-accumulate operations. The ALUs execute add/subtract, shift, and logical operations. Most of the arithmetic operations on the MACs and ALUs can be performed using saturation functions. The DAUs move data between the local memory and general-purpose registers. The SCU controls branch and zero-overhead loop program sequences, as well as conditional-execution instructions. SPXK5 has eight 40-bit general-purpose registers (R0, R1, ..., R7), eight 32-bit address registers (DP0, DP1, ..., DP7), and

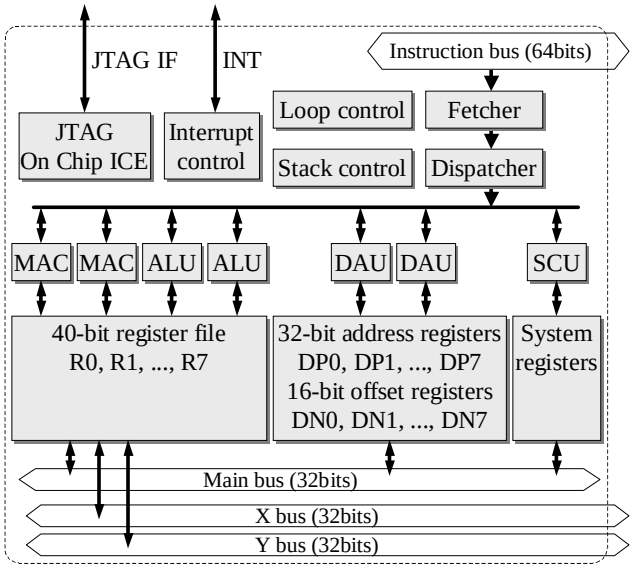

Fig. 1. SPXK5 block diagram showing seven functional units and four buses.

eight 16-bit offset registers (DN0, DN1, ..., DN7) that individually correspond, respectively, to each of the address registers. Each of these general-purpose registers (R0, R1, ..., R7) can be used as a 40-bit accumulator, and each of both the high and low 16-bit portions of these resisters (e.g., R0H, R0L) can also be used both as an operand in multiplication and as a 16-bit accumulator. This increases the processor's orthogonality.

In conventional DSPs, the instruction memory space and data memory spaces are separate and have distinct addresses. In SPXK5, however, these two kinds of memory spaces have been unified into one large memory space with a 32-bit address. This unified memory space consists of several banks, each of which is connected to three buses: two 32-bit data buses (X bus and Y bus) and a 64-bit instruction bus. Through these three buses, the core can access three different banks.

For faster operational frequency, SPXK5 has a six-phase pipeline: instruction fetch, dispatch queue, decode, address register update, execution phase 1, and execution phase 2. All ALU operations are executed in execution phase 1. MAC operations are executed in both execution phases 1 and 2, and their results only become valid for other units after the completion of phase 2.

To fabricate SPXK5, we used a 0.13-micron logic process. The DSP core measures 2 square millimeters and can operate at 250 MHz (1000 MIPS), with a power consumption of 0.05 mW/MIPS at 0.9 volts.

2.1. Application example: LMS adaptive filter

Since SPXK5 has two MAC units, it can perform finite impulse response (FIR) filters twice as fast as DSPs with only one MAC unit. Here, let us consider the example of a least mean square (LMS) adaptive filter to illustrate how the 4-way VLIW-type architecture and the two MAC units on SPXK5 are efficiently exploited. The LMS algorithm employed in transversal filter structures is widely

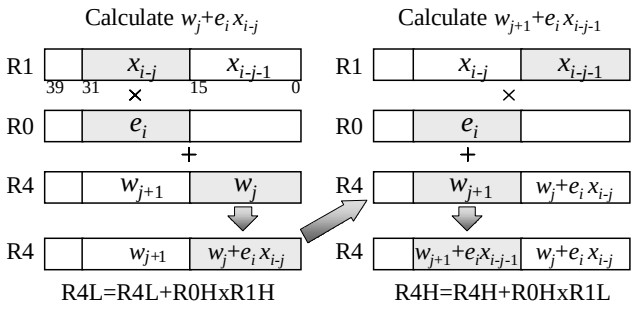

Fig. 2. Update filter coefficients in LMS on SPXK5. R*d*L and R*d*H denote, respectively, the low and high 16-bit portions of register R*d*.

used in many digital signal processing applications (e.g., echo cancellation, channel equalization, etc.). We define the input, output, and desired signals at time index i as, respectively, x_i, y_i, and d_i, and the jth filter coefficient as w_j. The LMS adaptive filter can then be formulated as

$$y_i = \sum_{j=0}^{T-1} w_j x_{i-j} \quad (1)$$

$$e_i = \mu(d_i - x_i) \quad (2)$$

$$w_j = w_j + e_i x_{i-j}, \quad j=0,1,...,T-1 \quad , \quad (3)$$

where μ is a small positive constant referred to as step size and T is the number of filter taps. While FIR filtering represented in Eq. (1) includes load and MAC operations, the updating of filter coefficients represented in Eq. (3) needs store operations as well as load and MAC operations. We can implement this adaptive filter efficiently on SPXK5 by using 16-bit + 16- x 16-bit MAC instructions. These 16-bit MAC instructions round a 32-bit result of the muliplication into a 16-bit value and add it to a 16-bit value in an accumulator. These instructions make it possible to use 32-bit load/store instructions in order to reduce the number of load/store operations for filter coefficients. As shown in Figure 2, w_j and w_{j+1} are updated in accord with Eq. (3), where each value is assumed to be a signed 16-bit value. First, a 32-bit load instruction reads two 16-bit data, w_j and w_{j+1}, from the local memory into register R4. Next, a 16-bit MAC instruction calculates $w_j+e_i x_{i-j}$, using the low 16-bit portion of R4. Another 16-bit MAC instruction then calculates $w_{j+1}+e_i x_{i-j-1}$, using the high 16-bit portion of R4. Finally, a 32-bit store instruction is used to store two 16-bit data, updated w_j and w_{j+1}, into the local memory from register R4. Since y_i can be calculated in parallel with the updating of w_j and w_{j+1}, SPXK5 operates the LMS adaptive filter at 1 cycle/tap.

3. INSTRUCTION SET

SPXK5 instructions are either 16- or 32-bit wide and can be grouped into up to 64-bit instruction packets to be executed within a cycle. This gives SPXK5 small code sizes for most programs, even though VLIW-type DSP designs usually result in large code sizes.

Our instruction set has been designed for use with high-level language compilers, such as C compilers, so as to be able to generate efficient codes in addition to enabling efficient parallel execution of digital signal processing algorithms. Such efficiency de-

pends on the orthogonality of the instruction set and on the use of certain instructions that are particularly helpful to compilers. Most MAC and ALU instructions require 3 operands, each of which can be equally chosen from among eight general-purpose registers. This increases the orthogonality of the instruction set.

Futher, each destination register in most MAC/ALU instructions has an automatic saturation mode. This allows effective implementation of speech codec standards. Conditional execution is employed for all ALU operations. A C-like algebraic assembly language helps programmers to understand and develop programs more easily. Moreover, we have introduced a number of special purpose instructions to accelerate specific 3G terminal applications, including video codec and Viterbi decoding. Features of the instruction set include:

- Single-instruction multiple-data (SIMD) type instructions for better data parallelism in ALU operations.
- Format-conversion instructions between unsigned 8-bit data and unsigned 16-bit data, for video codec.
- Minimum and maximum operations, for Viterbi decoding.

3.1. Application examples and benchmarks

Let us consider the examples of video codec and Viterbi decoding to illustrate how the SIMD-type instructions can be used to implement such applications efficiently. After that, let us look at benchmark results for basic applications.

3.1.1. Motion estimation (ME)

ME is one of the most heavily demanding operations in video encoders. It uses block-matching techniques to search a motion vector for a desired macroblock. Though either the sum of square difference (SSD) or the sum of absolute difference (SAD) might be used as a matching criterion, SAD is more commonly used because no multiplication is needed in SAD calculations. Between current and reference macroblocks, SAD may be expressed as

$$SAD = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \left| a_{m,n} - b_{m,n} \right| \quad , \tag{4}$$

where $a_{m,n}$ and $b_{m,n}$ are pixel values in, respectively, current and reference macroblocks. SIMD-type instructions, PSUB and PADDABS, can be used to calculate SAD on SPXK5. Figure 3 shows an SAD calculation flow diagram. Here, for the sake of simplicity, we assume that pixel values are stored in the local memory as unsigned 16-bit values. First, a 32-bit load instruction reads two pixel values ($a_{0,0}$,$a_{0,1}$) for the current macroblock, and another 32-bit load instruction reads two pixel values ($b_{0,0}$,$b_{0,1}$) for the reference macroblock. $a_{0,0}$ is placed in the low 16-bit portion of a register; $a_{0,1}$ is placed in the high 16-bit portion of the same register. This is also done for $b_{0,0}$ and $b_{0,1}$ in a different register. PSUB then calculates the differences between $a_{0,0}$ and $b_{0,0}$ and between $a_{0,1}$ and $b_{0,1}$. PADDABS adds the absolute of those differences into a subsequent register. By repeating these operations, SPXK5 obtains an SAD value.

3.1.2. MPEG-4 codec

Let us next estimate the operational frequency required to implement MPEG-4 video codec SP@L2 (352x288 pixels, 15 frame/sec)

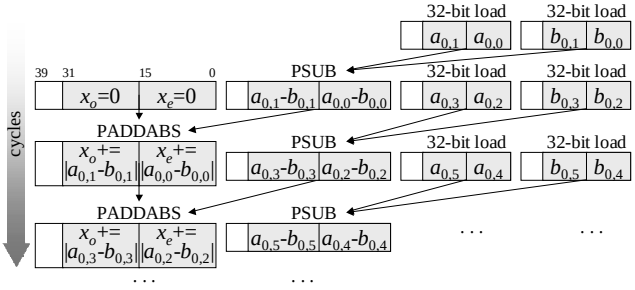

Fig. 3. An SAD calculation flow diagram

on SPXK5. We have already implemented H.263 video codec on μPD7701x at 50 MHz [3] and MPEG-4 video codec SP@L1 (176x144 pixels, 15 frame/sec) on the same architecture at 75 MHz [4]. The main components in MPEG-4 video codec are ME, discrete cosine transform (DCT), inverse DCT, motion compensation, quantization, and inverse quantization. Estimated perfomance for ME is 2.5 times faster on SPXK5 than that on μPD7701x. Either an 8-point DCT or IDCT can be implemented in 17 cycles (twice as fast as μPD7701x [9]). Other components can also be operated roughly twice as fast as those on μPD7701x. Therefore, a 200-MHz SPXK5 is sufficient to implement MPEG-4 video codec SP@L2. Further, a 250-MHz SPXK5 will possess the processing power required for both the video codec and speech codec, and it will consumes 50 mW at 0.9 volts.

Let us compare power consumption between a dedicated-design approach and our programmable-DSP approach for MPEG-4. The dedicated MPEG-4 video codec LSI reported in [6] consumes 240 mW at 60-MHz clock frequency when it executes both SP@L1 video codec and speech codec. The power consumption of 250-MHz SPXK5 is dramatically lower than that of the dedicated MPEG-4 LSI, where a 250-MHz SPXK5 can handle both SP@L2 video codec and speech codec. While the dedicated MPEG-4 LSI includes a 16-Mbit DRAM, the power consumption of our DSP core would be significantly lower even if the consumption of the LSI's 16-Mbit DRAM were ignored.

3.1.3. Viterbi decoding

The Viterbi decoding algorithm is a maximum-likelihood decoding algorithm for convolutional codes and is commonly used in FEC because it is simple to implement and offers large coding gain [10]. The trellis diagram for an R=1/n convolutional code can be subdivided into a number of basic modules because of the inherent symmetry in the trellis structure. These basic modules can be represented as state transitions between two old states (m,m+M/2) in stage k and two new states (2m,2m+1) in stage k+1, where m=0,1,...,M/2-1 (see Figure 4). We define a path metric at state m in stage k as $p_{m,k}$, and also define a branch metric as bm. The path metric is a likelihood ratio between the original encoder input and a recreated set of state transitions. The branch metric is a likelihood ratio of a state transition. The two path metrics in stage k+1, $p_{2m,k+1}$ and $p_{2m+1,k+1}$, may be expressed as

$$p_{2m,k+1} = \max[\, p_{m,k} + bm, p_{m+M/2,k} - bm] \tag{5}$$

$$p_{2m+1,k+1} = \max[\, p_{m,k} - bm, p_{m+M/2,k} + bm] \quad , \tag{6}$$

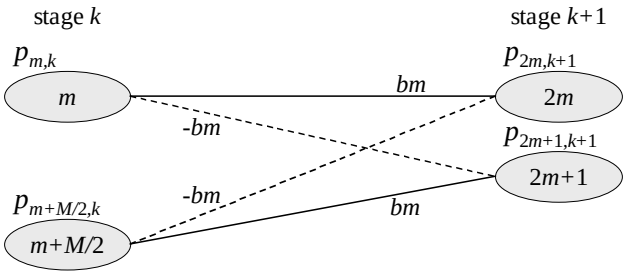

Fig. 4. Add-compare-select (ACS) operations

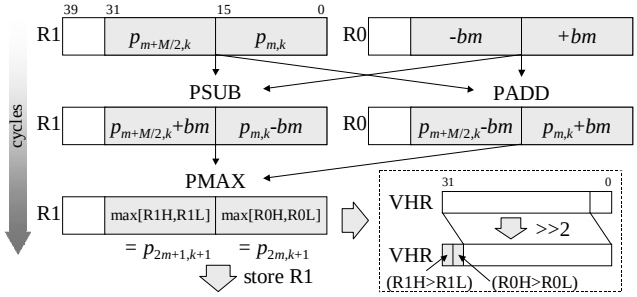

Fig. 5. ACS operations on SPXK5

where max[*x*,*y*] represents a larger value of either *x* or *y*. The operations represented by Eqs. (5,6) are referred to as add-compare-select (ACS) operations. The speed of ACS operations is crucial to the successful implementation of a Viterbi decoder on a DSP.

SPXK5 uses SIMD-type instructions, PADD/PSUB/PMAX, to calculate path metrics efficiently. Figure 5 illustrates how to use SIMD-type instructions to perform two ACS operations. Here, both path metrics and branch metrics are assumed to be signed 16-bit integers. First, PADD and PSUB, respectively, calculate two candidate values for one of two path metrics at stage *k*+1 in Eqs. (5,6). The first two candidates are placed into, respectively, the high and low 16-bit portions of R0; the same is done for the second two candidates with respect to R1. Next, PMAX compares the two candidates in each register, respectively, and selects the largest from each. PMAX also stores two 1-bit selection flags in the viterbi history register (VHR) for use later in determining the maximum-likelihood surviving path after calculating path metrics in all stages. The VHR is a system register and is 32-bit wide. The flags, which are either 0 or 1, indicate comparisons between each pair of candidates placed in R0 and R1, respectively. SPXK5 can execute PADD, PSUB and PMAX instructions in two cycles to perform two ACS operations (i.e., 1 ACS/cycle).

3.1.4. Basic application performance

Finally, let us look at performance results on SPXK5 for an FIR filter, an infinite impulse response (IIR) filter, an LMS adaptive filter, and a 256-point complex fast fourier transform (FFT). SPXK5 can operate an FIR filter at 0.5 cycles/tap, an IIR filter (five coefficients per biquad) at 3 cycles/biquad, and an LMS adaptive filter at 1 cycle/tap, where each performance result is given per sample. SPXK5 can also operate a 256-point complex FFT at 4200 cycles.

4. SUMMARY

We have developed SPXK5, a new-generation 4-way VLIW-type DSP core with low power dissipation for use in 3G terminals. SPXK5 employs a register architecture with good orthogonality and is designed so that compilers and programmers are able to generate codes easily and efficiently. For low power dissipation, a DSP to employ SPXK5 core has been designed to operate at low voltages (under 1.0 volt). As has been shown here, the DSP core allows efficient low-power implementation of video codec, Viterbi decoding, and adaptive filtering, which are important applications for 3G terminals.

5. ACKNOWLEDGEMENTS

The development of the new DSP core described in this paper has been made possible by the hard work of a very large group of people at NEC Corporation.

6. REFERENCES

[1] K. Nagata, "IMT-2000 terminal and its requirements for device technologies," *Proceedings of Symposium on VLSI Circuits Digest of Technical Papers*, pp. 2–5, 2000.

[2] S. Bauer, J. Kneip, T. Mlasko, et al., "The MPEG-4 multimedia coding standard: Algorithms, architectures and applications," *Journal of VLSI Signal Processing*, vol. 23, pp. 7–26, Oct. 1999.

[3] Y. Naito, I. Kuroda, "H.263 mobile video codec based on a low power consumption digital signal processor," *Proceedings of ICASSP98*, vol. 5, pp. 3041–3044, 1998.

[4] Y. Naito, H. Matuo, E. Sasaki, et al., "MPEG-4 autio-video codec based on low power DSPs," *Proceedings of Engineering Sciences Society Conference of IEICE* ,A-4-8, p. 60, 1999, (in Japanese).

[5] M. Budagavi, W. R. Heinzelman, J. Webb, et al., "Wireless MPEG-4 video communication on DSP chips," *IEEE Signal Processing Magazine*, pp. 36–53, Jan. 2000.

[6] M. Takahasi, T. Nishikawa, H. Arakida, et al., "A scalable MPEG-4 video codec architecture for IMT-2000 multimedia applications," *Proceedings of IEEE international Symposium on Circuits and Systems*, , no. II, pp. 28–31, May 2000.

[7] A. Gatherer, T. Stetzler, M. McMahan, et al., "DSP-based architectures for mobile communications: Past, present and future," *IEEE Communications Magazine*, vol. 38, no. 1, pp. 84–90, Jan. 2000.

[8] J. Eyre, J. Bier, "The evolution of DSP processors," *IEEE Signal Processing Magazine*, pp. 43–51, Mar. 2000.

[9] M. Yoshida, H. Ohtomo, I. Kuroda, "A new generation 16-bit general purpose programable DSP and its video rate application," *VLSI Signal Processing VI*, pp. 93–101, 1993.

[10] A. J. Viterbi, *CDMA Principles of Spread Spetcrum Communication*, Addison-Wesley Publishing Company, 1995.