A Scalable SIMD Digital Signal Processor for High Quality Multifunctional Printer Systems

Hyeong-Ju Kang[a], Yongwoo Choi[a], Kimo Kim[a], In-Cheol Park[*a],
Jung-Wook Kim[b], Eul-Hwan Lee[b], Goo-Soo Gahang[b]
[a]Korea Advanced Institute of Science and Technology 373-1, Guseong-dong, Yuseong-gu, Daejeon, Republic of Korea
[b]Digital Printing Division, Samsung Electronics 416, Maetan-dong, Yeongtong-gu, Suwon, Republic of Korea

ABSTRACT

This paper describes a high-performance scalable SIMD digital signal processor (DSP) developed for multifunctional printer systems. The DSP supports a variable number of datapaths to cover a wide range of performance and maintain a RISC-like pipeline structure. Many special instructions suitable for image processing algorithms are included in the DSP. Quad/dual instructions are introduced for 8-bit or 16-bit data, and bit-field extraction/insertion instructions are supported to process various data types. Conditional instructions are supported to deal with complex relative conditions efficiently. In addition, an intelligent DMA block is integrated to align data in the course of data reading. Experimental results show that the proposed DSP outperforms a high-end printer-system DSP by at least two times.

Keywords: Digital signal processor, DSP, image processing, multifunctional printer system

1. INTRODUCTION

Today's printer systems are evolving to complex ones covering a lot of functions that were separately implemented in the past, such as facsimiles, copiers, and scanners, as the functions can share many image processing algorithms such as color correction, contrast control, image enhancement, and so on[1-5]. To cope with frequent changes in the processing algorithms, software implementation on a powerful digital signal processor (DSP) is preferred to the traditional hardware implementation.

A DSP is a micro-processor that has features suitable for digital signal processing[6]. Since a digital signal processing algorithm requires many filter-like operations, $\sum hx$, a DSP usually has a single- or double-cycle multiplication-and-accumulate (MAC) unit. In addition, a DSP has a separated data memory bus because digital signal processing algorithms are data-intensive and a data memory is frequently accessed. Furthermore, a DSP has simultaneous instructions, where an ALU operation and a memory access are performed at the same time.

Since recent image processing algorithms process enormous amount of data, a DSP is evolved to have a parallel architecture[6]. For example, the TMS320C6xxx architecture of Texas Instruments can perform 8 instructions in parallel. It has two datapaths, each of which contains two arithmetic units, one multiplier, and one memory access unit[7]. Many of DSP companies, such as Analog Devices, Motorola, and Hitachi, also develop their own DSP architectures that have many data processing units[8-10].

Such parallel DSPs, however, have a property that is not suitable for image processing. The DSPs usually have very-long-instruction-word (VLIW) architectures, where operations in an instruction should not have data dependency. Since an image processing algorithm usually performs successive operations on a piece of data, it is difficult to make enough long instructions. Moreover, the VLIW architecture requires a complex control unit, which can be a serious overhead.

Some of the DSPs are not good at processing various formats of image data, either. A piece of data can be of conventional length like 8 bits, 16 bits, or 32 bits, and unconventional length like 10 bits, 12 bits, and so on. The DSPs

* icpark@ee.kaist.ac.kr; phone 82 42 869 3461; fax 82 42 869 4410

have a few special instructions to process conventional-length data, but has few instructions for unconventional-length data. In addition, a pixel usually consists of 3 pieces, red, green, and blue, which format is not desired by the usual DSPs.

In this paper, we present a high-performance DSP, named as PrimDSP, specialized for multi-functional printer systems, which has many features related to image processing algorithms. PrimDSP employing the RISC-like five-stage pipeline structure is based on a single-instruction-multiple-data (SIMD) architecture[11] to apply the same operation to a bundle of image data and can have up to 16 datapaths, each of which is associated with a corresponding local data memory. The number of datapaths is scalable according to the requirement of applications.

Various quad/dual instructions are supported to pack four data of byte length or two data of half-word length into one word length form. As image processing algorithms usually apply the same operation to all data, such instructions lead to less code size and faster processing. To process image data that have various bit-lengths, PrimDSP is able to perform bit-field extraction/insertion before and after ALU operations. PrimDSP supports up to 4 extraction/insertion patterns at the same time, and one of them can be specified in the related instruction.

The DMA in PrimDSP has intelligent functions. The conventional DMA only transfers data between a local memory and an external memory. In addition to the conventional DMA function, the proposed DMA can transform the image data structure for easy image processing. If a pixel data size is 3 bytes, the pixel boundary is not aligned to the word boundary. To process the pixel data in image processing algorithms, the misaligned data should be aligned to the word boundary, leading to much overhead in software. The proposed DMA supports the alignment function for the data requested from PrimDSP, because the function can be implemented with a little hardware.

To compare the performance of PrimDSP with a commercial SIMD DSP, three image processing algorithms are implemented in assembly languages, and the average number of cycles required to process one pixel in the commercial SIMD DSP and PrimDSP is measured. The experiments show that PrimDSP consumes only half the number of cycles compared to the other DSP.

This paper is organized as follows. The base architecture of PrimDSP is presented in Section 2 and the instruction features are explained in Section 3. Section 4 describes PrimDSP's DMA functions and Section 5 shows the experimental and synthesis results. Concluding remarks are made in Section 6.

2. BASE ARCHITECTURE

PrimDSP's architecture is developed to achieve high-performance in image processing algorithms. It employs a SIMD architecture and a RISC-like five-stage pipeline. Each datapath has an ALU, a MAC unit, a register file with 32 general purpose registers, two extractors, and one inserter. PrimDSP also contains a local data memory and a program cache.

2.1. SIMD Architecture

PrimDSP is based on the SIMD architecture as shown in Fig. 1, as a group of image data is processed in the same way in many image processing algorithms. A single datapath processor cannot meet the performance specification of today's printer systems. Another parallel architecture, the VLIW architecture, allows different operations on each datapath, which increases control part overhead. Moreover, to fully utilize the parallelism of the VLIW architecture, the operations to be executed in a cycle should not have any dependency among them, which rarely occurs in image processing algorithms.

The number of datapaths in PrimDSP can be changed according to the performance requirement, as there are many types of multifunctional printer systems requiring different performance specification. For low-performance systems, excess datapaths increase chip area without contributing the performance. PrimDSP can support up to 16 datapaths, so it is well suitable for a wide range of printer systems.

The local data memory is regarded as a set of sub-memories, each of which is associated to a datapath. Every datapath can access its sub-memory at the same time. However, the data memory can be treated as one memory when a datapath wants to access an arbitrary position of the memory.

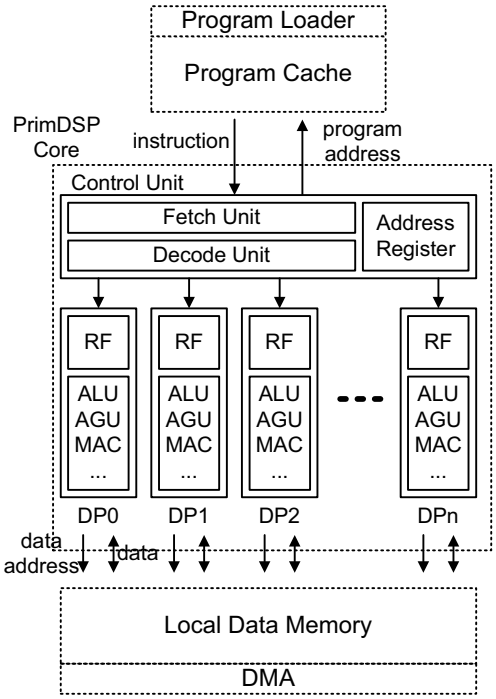

Figure 1: The SIMD architecture of PrimDSP.

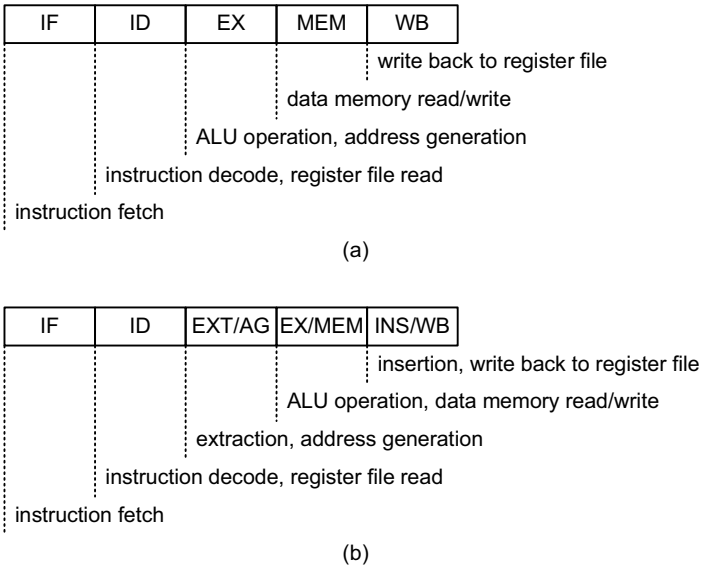

Figure 2: (a) The typical RISC pipeline structure and (b) PrimDSP pipeline structure.

2.2. Pipeline Structure

PrimDSP follows the traditional RISC pipeline structure with five stages, IF-ID-EX-MEM-WB as shown in Fig. 2a. An instruction is fetched from a program memory at IF stage and decoded at ID stage. Register values are read at ID stage, too. An ALU operation is performed at EX stage. A data memory is accessed at MEM stage, and the result is written to registers at WB stage. The pipeline structure, however, is somewhat modified in PrimDSP for image-processing.

To process various bit-length data, PrimDSP performs extraction/insertion operations before and after ALU operations. If extraction/insertion operations are performed with an ALU operation, the critical path becomes very long.

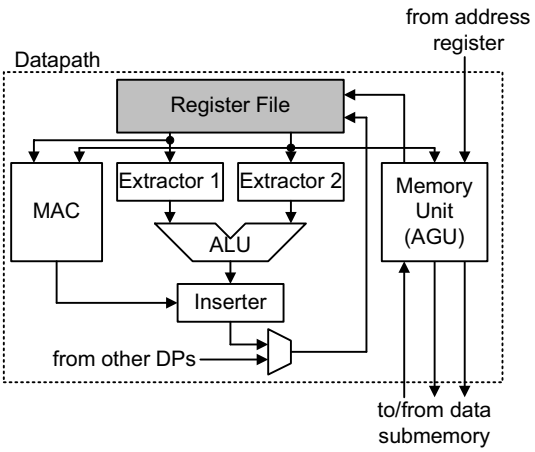

Figure 3: Datapath structure.

Those operations, therefore, should be performed at separated stages. As EXT stage and INS stage are placed before and after EX stage, the pipeline structure is changed to IF-ID-EXT-EX/MEM-INS/WB as shown in Fig. 2b.

In a typical RISC machine, the address for a data memory access is computed in an ALU of EX stage. In PrimDSP, as EX stage is processed in parallel with MEM stage, the address is generated at EXT stage with a separate address generation unit (AGU). A multiplication is performed through two stages, EXT/AG and EX/MEM, because multiplication usually takes two times as long delay as addition.

2.3. Datapath Structure

Each datapath has a dedicated register file, several special purpose registers, two extractors, an AGU, an ALU, a MAC unit, and an inserter as shown in Fig. 3. The extractor extracts a part of data read from the register file. The extracted bit-field is processed by the ALU, and the inserter inserts a part of the calculated data into a register. The AGU generates the address for a data memory access, and the MAC unit multiplies the source operands and accumulates the result into the accumulator.

A register file contains 32 general purpose registers each of which is 32bits. There are two read ports and two write ports at the register file. In addition to the register file, a datapath also includes an accumulate register, an insertion register, 4 increment registers, and a condition register. Some registers are common for all datapaths to achieve easy data transfer among datapaths, which include 16 address registers, 12 increment registers, and many special purpose registers.

2.4. Other Features

Beside the above features, there are features included to improve performance in PrimDSP. An intensive forwarding scheme is used, and DSP-like instructions are adopted to handle loops, such as a decrement and branch instruction and a zero overhead loop instruction. A DMA unit is employed to enable data transfer between the local data memory and the external memory without consuming the processing power of PrimDSP, and a program cache is integrated to speed up the program accesses. The program cache can operate as a downloadable memory.

3. INSTRUCTION FEATURES

3.1. Parallel and Single-Datapath Instructions

An instruction that is performed in every datapath at the same time is called a parallel instruction. These instructions include typical ALU instructions, multiplication/MAC instructions, load/store instructions, and extraction/insertion ALU instructions, and have two source operands and one destination operand. One of the two source operands can be an immediate constant.

Special ALU instructions are minimum and maximum instructions that select the smaller and the larger of the two source operands, respectively. These operations are often used to process the median filtering. If the instructions are not supported, the operations should be implemented with compare and branch instructions, which takes a number of instructions and cycles. PrimDSP, however, performs the operations in only one cycle by using the minimum and maximum instructions.

Instructions being performed in only one datapath are called single-datapath instructions. An instruction in this category has a field specifying the datapath to be executed. Typical ALU operations can be included in the single-datapath instructions, and there are two special move instructions. One of them broadcasts a register value of a datapath to a register of all datapaths. The other transfers a register value of a datapath to a register of another datapath. The single-datapath instructions also contain multiplication/MAC instructions and load/store instructions

3.2. Simultaneous Instructions

Simultaneous instructions perform an ALU operation in parallel with a load operation. This is possible because ALU and load operations use different units. The source operands of an ALU operation are read from the register file, and the base address of a load instruction is read from address registers. The ALU operation is performed in an ALU, and the load operation in an AGU. The results of ALU and load operations can be written to the register file at the same time as the register file has two write ports.

Because of limited instruction length, there are restrictions in specifying the operands in simultaneous instructions. The ALU operation has only two operands, and the result of the ALU operation is written into one of the source operands. The load operation supports only post-increment addressing.

3.3. Quad/Dual Instructions

Image data are usually represented in a 8-bit or 16-bit format. To reduce memory usage, four 8-bit data or two 16-bit data are packed in a 32-bit memory entry. In a conventional DSP, each data contained in a 32-bit entry is processed separately. It takes a number of cycles to repeat extracting a field of data and processing it. Quad/dual instructions are included in PrimDSP to process all the data fields contained in a 32-bit entry simultaneously.

A quad instruction processes four 8-bit data fields at a time. Some of instructions in this category are illustrated in Fig. 4a. The quad-multiplication instruction multiplies each 8-bit data in one source operand by the corresponding 8-bit data in the other source operand, and the quad-multiplication-and-accumulation instruction accumulates the multiplication results. The quad-individual-multiplication instruction multiplies each 8-bit data in one source operand by the other 32-bit source operand. The multiplication results are right adjusted to prevent overflow. The quad-addition instruction operates in the same way as the quad-multiplication instruction except that it performs addition instead of multiplication. If an addition result cannot be represented in 8 bits, the result is saturated. The quad-accumulation-with-mask instruction accumulates each 8-bit data of one source operand if the corresponding bit in the other operand is set.

A dual instruction processes two 16-bit data fields simultaneously. This instruction group includes dual-multiplication, dual-multiplication-and-accumulation, and dual-individual-multiplication. The instructions are illustrated in Fig. 4b.

Each data field in the quad/dual format can be signed or unsigned. Four signed/unsigned patterns are saved in a special purpose register, and a quad/dual instruction has a field to indicate one of the four patterns.

3.4. Extraction/Insertion Instructions

The size of image data can be different from the convention 8 or 16 bits to achieve compact representation. In this case, the quad and dual instructions are not sufficient. To support the case, extraction/insertion instructions are included, which perform extraction and insertion before and after an ALU operation as shown in Fig. 5. An ALU operation is performed for the extracted data, and then a part of the ALU result is inserted into the insertion register. The insertion result is written to a general purpose register and the insertion register.

To specify extraction and insertion patterns, PrimDSP supports four groups (EI groups) of pattern registers. An extraction/insertion instruction has a field to indicate which EI group is used. An EI group consists of two extraction

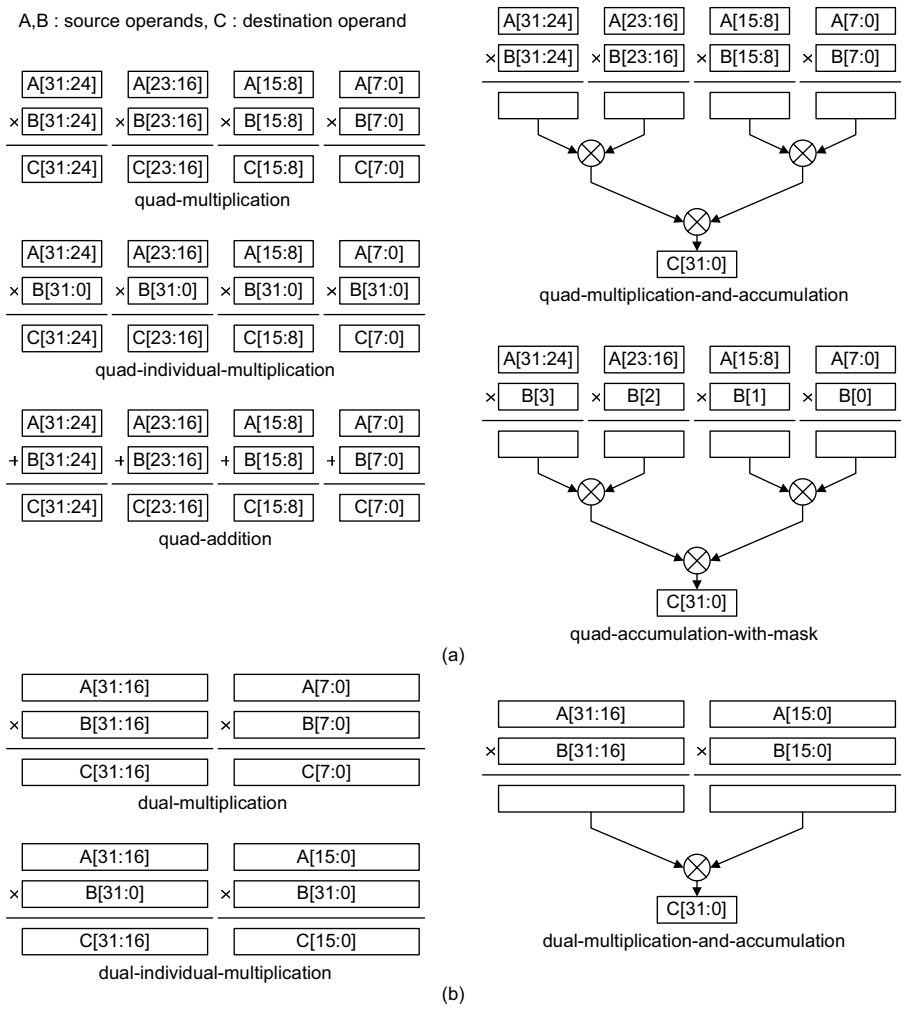

Figure 4: (a) Quad instructions and (b) dual instructions.

pattern registers and an insertion pattern register. An extraction pattern register specifies extraction position, extraction width, signed/unsigned extraction, and automatic position increment. The meaning of each field is illustrated in Fig. 6a. The signed/unsigned extraction field indicates whether the extracted data is signed or not. The two extraction pattern registers in an EI group specify the extraction patterns of two source operands.

An insertion pattern register consists of insertion position, insertion width, extraction position, saturation, and automatic position increment. The function is shown in Fig. 6b. The saturation field is for the case when the ALU result cannot be represented with the length specified in the insertion width. If the saturation field is set, a saturated value is inserted.

Since an EI group consists of three registers, it takes three cycles to specify an extraction/insertion pattern. This overhead can be serious when many extraction/insertion patterns are required in image processing algorithms. PrimDSP has four virtual EI registers to reduce the overhead when typical extraction/insertion patterns are used. Image processing algorithms usually use 8-bit or 16-bit extraction/insertion patterns that can be represented with a small number of bits. A virtual EI register has a length enough long to specify 8-bit or 16-bit patterns and enough short to be assigned in an instruction. If an EI register is assigned to a value A, the extraction and insertion registers in the corresponding EI group are assigned to the pattern corresponding to A.

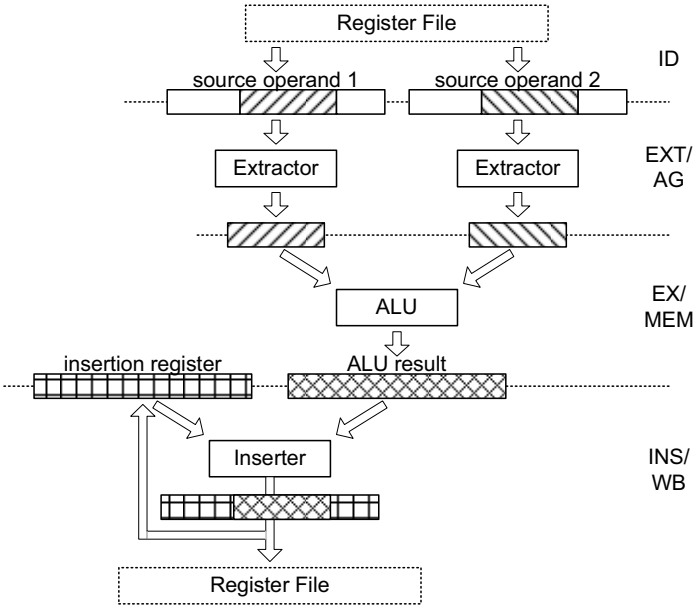

Figure 5: An extraction/insertion instruction.

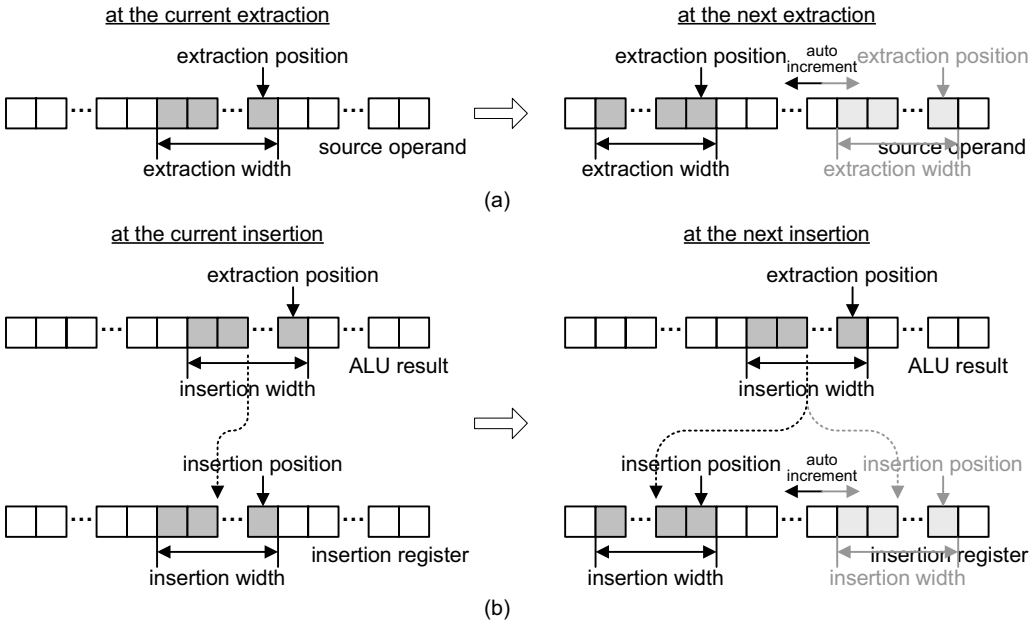

Figure 6: (a) An extraction pattern and (b) an insertion pattern.

3.5. Conditional Execution

In some image processing algorithms, a datum is processed differently according to its value. As each datapath has different data, each datapath operates differently. If these operations were performed separately in each datapath, the benefit of the SIMD architecture would be lost.

PrimDSP solves the problem by introducing conditional execution. Each datapath has 20 condition bits, which are set by an instruction comparing data values in each datapath. In addition, there are two special instructions: a conditional execution start instruction with a condition bit specified and a conditional execution end instruction. Instructions between the two instructions are performed in the only datapaths where the specified condition is set. Some of the

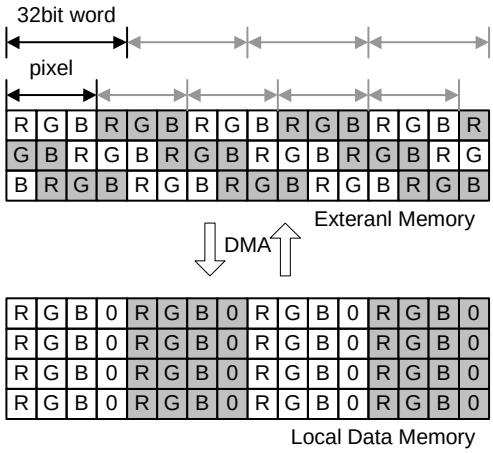

Figure 7: An intelligent DMA function.

condition bits are set automatically by logical operation of other bits to deal with very complex cases being determined by several data values.

4. DMA

In an external memory, image data are arranged in a way to reduce memory usage. The arrangement, however, is not efficient for image processing algorithms. For example, let us consider a pixel consisting of 3 bytes: red, green, and blue. In an external memory, a bundle of pixels are arranged consecutively. Therefore, the pixel boundary does not coincide with the 32bit-word boundary that is natural in computer arithmetic operations. To process a pixel, two words are often loaded to combine parts of the two words. These operations are inefficient if implemented in software.

The DMA of PrimDSP has an intelligent function to solve this problem. A conventional DMA just transfers data in an external memory to a local data memory. The proposed DMA, however, changes data arrangement while transferring them. In the above example, the DMA aligns pixel data to make the pixel boundary coincide with the word boundary as shown in Fig. 7. When transferring data in the local data memory back to the external memory, the DMA changes data arrangement into the original one.

The DMA has more rearrange modes than the mode mentioned above. For image processing algorithms that process each color data separately, for example, the DMA separates three color components and stores them into separated memory regions.

5. EXPERIMENTAL AND SYNTHESIS RESULTS

Three typical image processing algorithms are coded in assembly languages to compare the performance of PrimDSP with a commercial high-end SIMD DSP. The contrast control algorithm calculates the luminance of each pixel and obtains a new luminance by using different linear functions. The median saturation algorithm calculates the saturation value of a 3x3 region and the median value. The scanner color correction algorithm calibrates the input RGB pixel data by multiplying a 3x3 matrix.

Table 1 shows how many cycles the commercial DSP and PrimDSP take to process one pixel in each algorithm. In the contrast control algorithm, PrimDSP takes half of the cycles that the high-end DSP does. This is because PrimDSP does not waste cycles to rearrange image data and has powerful conditional instructions. PrimDSP is four times as efficient as the high-end DSP in the median saturation algorithm that frequently compares two operands and selects the larger or smaller. The minimum and maximum instructions are very effective for the operation. PrimDSP also performs the scanner color correction algorithm efficiently due to the intensive quad/dual instructions and the intelligent DMA.

Table 1: Average number of cycles to process image processing algorithms.

Algorithms	Commercial DSP	PrimDSP
Contrast Control	24.56 cycles/pixel	12.24 cycles/pixel
Median Saturation	39.19 cycles/pixel	9.72 cycles/pixel
Scanner Color Correction	8.02 cycles/pixel	4.52 cycles/pixel

Table 2: Synthesis results.

Core Area	Control Unit	36,893 gates
	1 Datapath	78,609 gates
	Core = Control Unit + 4 Datapaths	351,329 gates
Critical Path Delay		3.4ns

PrimDSP is implemented in Verilog HDL and synthesized with 0.13um CMOS technology. Table 2 shows the synthesis results. The equivalent gate count of a datapath is 78,609gates and that of the control unit is 36,983gates. The PrimDSP equipped with 4 datapaths occupies 351,329gates. The critical path of PrimDSP is the forwarding path through a multiplier and has 3.4ns delay. The maximum clock frequency, therefore, is 294MHz.

6. CONCLUSION

A high-performance digital signal processor has been proposed for multi-functional printer systems. The SIMD architecture is exploited to utilize the common property of image processing algorithms that the same operations are applied to a bundle of image data. The processor is also based on the traditional RISC-like pipeline to achieve a simple and fast structure. Various quad/dual instructions are supported to process multiple data at a time, and extraction/insertion instructions are employed to cope with various data sizes. The execution of a code range can be specified with a condition such that the instructions in the range are executed only in the datapaths where the condition bit is set. To reduce the computational overhead needed to align data read from the external memory, an intelligent DMA is proposed to align the memory data into a form suitable for computation while reading the data from the memory. Experimental results on typical image processing algorithms show that the proposed SIMD DSP is two times effective in performance compared to a commercial high-end DSP.

REFERENCES

1. R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Prentice Hall, 2002.
2. H. Ancin and A. K. Bhattacharjya, “Text enhancement for laser copiers,” in *Proc. Int. Conf. Image Processing*, 1999, pp. 494-498.
3. A. Pascovici and J. S. Shu, “Method and apparatus for processing a document by segmentation into text and image areas,” U.S. Patent 5 883 973, Mar. 16, 1999.
4. C. Lee, M. Eden, and M. Unser, “Near optimal geometric image scaling using oblique projection operators,” in *Proc. Int. Conf. Acoustics, Speech, and Signal Processing*, 1996, pp. 2399-2402.
5. H.-H. Cho, C.-H. Choi, B.-H. Kwon, and H.-R. Choi, “A design of contrast controller for image improvement of multi-gray scale image,” in *Proc. Asia Pacific Conference on ASICs*, 2000, pp. 131-133.
6. J. Eyre and J. Bier, “The evolution of DSP processors,” [Online]. Available: http://www.BDTI.com
7. Texas Instruments, “TMS320C6000 CPU and Instruction Set Reference Guide,” [Online]. Available: http://www.ti.com
8. Analog Devices, “ADSP-TS101S,” [Online]. Available: http://www.analog.com
9. Motorola, “SC140 DSP Core Reference Manual,” [Online]. Available: http://www.freescale.com
10. Hitachi, “SH-4R Architecture Technical Overview,” [Online]. Available: http://www.renesas.com
11. J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufman, 1996.