IMPLEMENTATION OF A DIGITAL COLOR COPIER USING A VLIW SIMD ARCHITECTURE

Seokhoon Ju and Wonyong Sung

School of Electrical Engineering
Seoul National University
San 56-1, Shillim-dong, Kwanak-gu, Seoul 151-742 Korea
Phone: +82-2-885-7411, Fax: +82-2-882-4656
Email: {jush, wysung}@dsp.snu.ac.kr

ABSTRACT

In this paper, we developed real-time image processing programs for a digital color copier using TMS320C6416 digital signal processor. The processor is good for real-time image processing because of multiple and packed-data processing functional units. However, it needs careful programming to exploit deep pipelining, multiple functional units and packed-data instructions. To improve the unit utilization ratio, we developed a few techniques, such as pack/unpack reduction, resource maximization and operation reduction, and multi-pixel processing. All the critical functions for the implementation of a digital color copier, which include shading correction, X-zooming, 2D filtering, and vector half-toning, are implemented. It is shown that a 720 MHz C6416 CPU can perform all the real-time processing needed for a 20 PPM (page per minute) 600 dpi A4 size color copier. We used C programming with intrinsic functions and linear assembly programming followed by the assembly optimizer.

1. INTRODUCTION

Digital color copiers are becoming a commercial product. However, real-time processing of color image requires more than 3 times of arithmetic and memory access operations when compared to black and white (B/W) processing. Currently, many digital copiers are implemented using hardwired image processing circuits because of the demand for high throughput. However, the hardware based systems are disadvantageous for implementing complex functions, such as editing and compression. The Texas Instruments' digital signal processor TMS320C6416 can achieve a very high processing rate due to its VLIW (Very Long Instruction Word) architecture and packed data processing support [1]. The CPU can process up to 8 instructions at each clock because of its pipelined and VLIW characteristics. It can also process up to four pixels of 8-bit data at each instruction because of the packed data processing capability. The architecture of the TMS320C6416 is shown in Fig. 1.

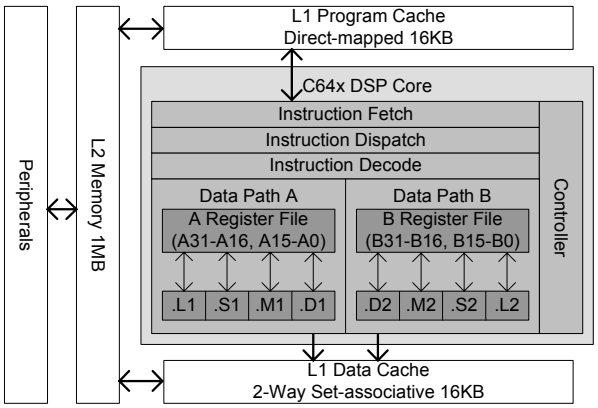

Fig.1. TMS320C6416 architecture

The prototype digital copier developed performs the basic image processing steps depicted in Fig. 2 [2].

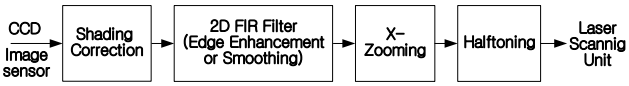

Fig.2. Image processing flow of a digital color copier

Table 1 shows the number of arithmetic operations per color pixel for each basic image processing step.

Table 1. Number of arithmetic operations per color pixel

	Shading correction	2D FIR filter(7x5)	X-Zooming	Vector error diffusion
Multiply	3	105	6	36
Addition	0	102	3	39

As shown in this table, 2D FIR filtering and vector half-toning are the most demanding blocks for the real-time implementation. This also implies that a 20 PPM color copier requires the computing capability of approximately

3 GOPS (Giga Operations Per Second) even if all the load/store and control overheads are neglected. Thus, a real-time implementation is not possible without utilizing the VLIW and SIMD architectural characteristics.
As for the software development environments, we used both C program with intrinsic functions and linear assembly programming followed by the assembly optimizer [3]. Manual parallel assembly programming is avoided because of programming difficulties and lack of portability.

2. OPTIMIZATION METHODS

We employed a few optimization techniques that can be utilized for the implementation of other image processing algorithms using parallel and pipelined architectures.

3.1. Pack/Unpack Reduction

Color pixels are usually represented by three separate data, such as R, G, B or C, M, Y. However, most of the image processing steps for a color copier are conducted for each color independently, except for the vector half-toning. Thus, the color data format shown in Fig. 3-(a) incurs much pack/unpack operations. In this work, the color data format is stored as shown in Fig. 3-(b). The employed color format is also important for efficient SIMD processing because a SIMD operation can process four pixels, not three, at a time. Note that the color format shown here is different from the color image sampling point structure.

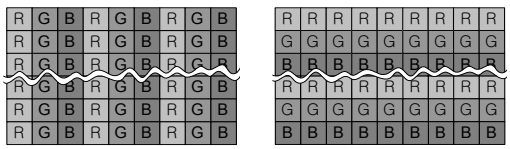

Fig.3. Color data storage format

3.2. Resource Maximization and Operation Reduction

Many of the image processing algorithms, such as 2-D filtering and shading correction, contain a good number of parallel operations. In this case, the maximum throughput is determined by the ratio of the number of functional units and the number of arithmetic operations. Thus, one of the most important things is to reduce the number of operations, which can be either arithmetic or load/store operations. For the case of an FIR filtering, a conventional approach is to use the symmetrical characteristics as shown in Fig. 4. But, this structure increases the number of pack/unpack operations since the distance of the pixels to be added is not a constant.

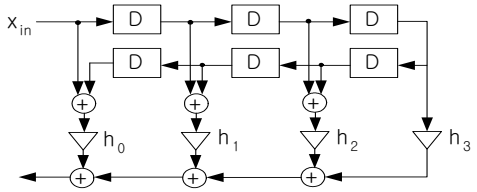

Fig.4. A linear phase FIR filter

In this work, the vertical symmetry of filter coefficients is utilized as shown in Fig. 5. The 1-D filtered results are not only used for the current pixel but also stored for the pixel several lines later. This approach halves the number of arithmetic operations. Assume that a filter kernel of 7x5 is employed, where 7 is the tap length in the horizontal direction. Then, for example, the fifth line of input data is multiplied with the same coefficients for the calculation of line number 3 and line number 7. Note that this does not require pack/unpack operations although this increases the number of store operations slightly.

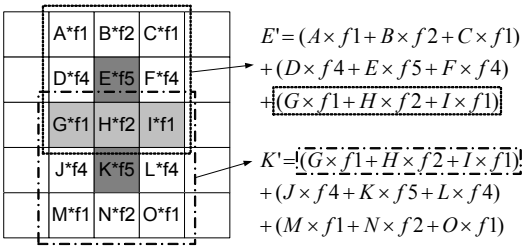

Fig.5. Use of vertical symmetry

For the case of shading correction, the saturated arithmetic operation is conducted not only by using **SPACKU4** but also using **MIN2** and **PACKU4** operations. Although the latter instructions are not efficient, the **SPACKU4** is only supported by the **.S** functional unit. Thus, it is better to use **MIN2** and **PACKU4** to utilize, otherwise, idle functional units.

3.3. Parallelism Increase Using Multi-Pixel Processing

Although C64x DSP can execute many operations simultaneously, it is not always possible to utilize most of them. A signal flow for an algorithm can be classified as shown in Fig. 6. Figure 6-(a) shows the case when there is no recursive or feedback loop in the algorithm as can be found in 2-D FIR filtering and X-zooming. Figure 6-(b) shows the case where there is a locally recursive dependency loop. In this case, it is possible to achieve a high utilization ratio by inserting D, E, and F operations at the same pipelining. However, in this case, the throughput is determined by the computational complexity in the recursive part, E and F. Thus, it is important that E and F do not require much computation. Figure 6-(c) shows the case when an algorithm requires a very long step of

recursive processing as can be found in the error diffusion process. In this case, the utilization ratio cannot be high unless some parallelization algorithms are employed. But, the parallelization of an algorithm requires a significant overhead in most cases. The approach that we employed is the skewed multi-line processing [4]. It has been shown that two pixels at different lines with some skew in the horizontal direction can have no dependency relation in the error diffusion process. In this case, the signal flow is transformed as shown in Fig. 6-(d).

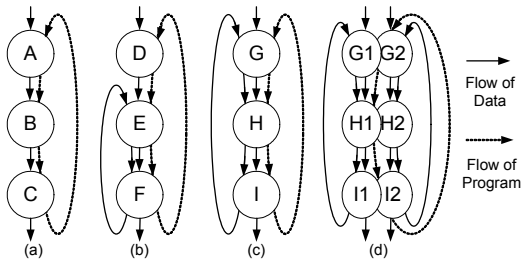

Fig.6. Data flow graphs for image processing

4. IMPLEMENTATION RESULTS

4.1 Shading Correction

Shading correction compensates for the non-uniform light intensity using a multiplication with a pre-stored gain value for each pixel. We not only use **SPACKU4** and **SHRU2** instructions in **.S** units but also **MIN2** and **PACKU4** instructions in **.L** units. A multiple number of pixels are processed at a time to utilize both SIMD arithmetic and multiple functional units. Table 2 shows the number of cycles for each pixel and the average unit utilization ratio in terms of the number of pixels and when utilizing **MIN2** and **PACKU4** instructions. Note that the number of instructions to execute for each pixel is not the same, thus a code having a smaller unit utilization ratio can be executed faster at sometimes.

Table 2. Optimization results of shading correction

	No SIMD	Using Only SPACKU4			Using SPACKU4, MIN2, PACKU4
		2 pixel	4 pixel	**8 pixel**	**8 pixel**
Cycles/pixel	14.09	2.52	1.41	**1.40**	**1.21**
Avg. unit utilization	1.71	4.28	4.3	**3.4**	**5.63**

4.2. 2D FIR Filtering

2D FIR filtering requires a large number of multiplications and additions as shown in Table 1. We assumed a linear phase filtering, which has symmetric coefficients, but do not assume any other constraints on the filter coefficients because the coefficients should be downloadable. As explained in Section 3.2, it is possible to halve the number of arithmetic operations by saving the 1-D filtered results and reusing it later. However, the implementation results show that the load/store operations limit the performance as shown in Table 3. In order to reduce the number of load/store operations, two adjacent pixels are processed at a time. Although the two pixel processing does not show the performance improvement when using C programming with intrinsic functions, it shows a better result when using linear assembly programming followed by the assembly optimizer. The results are summarized in Table 3.

Table 3. Implementation results for 2-D filtering

	1 Pixel	2 Pixel	**2 Pixel (Linear ASM)**
cycles/pixel	14.16	15.30	**11.39**
Avg. unit utilization	7.5	6.78	**8**
Cycles of loop	4 x 3	4.5 x 3	**3 x 3**

4. 3. X-zooming

A digital copier independently magnifies or reduces the original image in X-Y directions. The X-zoom is the scaling of the original image along a scanned line and is performed by digital signal processing, while the Y-zoom is conducted by changing the scanning speed. We employed the interpolation based method for X-zoom. Since a wide range of zooming ratio is needed, it is necessary to find out the value of the hypothetical pixel point. Note that this location is pre-computed according to the zooming ratio to reduce the real-time processing overhead [2]. The problem of X-zooming is that it requires many unaligned load operations. Note that three tables are needed, one stores the index values of 16 bit, and the other two keep the ratio values of 8 bit. Thus, the overhead of table load can be much reduced by merging these tables, although this require unpack after the load. The implemented results are shown in Table 4, where the linear assembly programming followed by the assembly optimizer produces a much better results than the compiled one.

Table 4. Optimization results of X-Zooming

	1 pixel	2 pixel	**2 pixel (Linear ASM)**
Cycles/pixel	6.33	12.45	**4.42**
Avg. unit utilization	4	1.43	**4.38**

4. 4. Vector Half-toning Process

In this paper, the vector error diffusion (VED) algorithm is selected for color printing for better quality [5]. Note that the vector error diffusion utilizes all the colors for faithful half-toning, but as a result, it requires 9 times of arithmetic operations assuming the same filter kernel when compared to monotone processing. Figure 7 shows the overall vector error diffusion process.

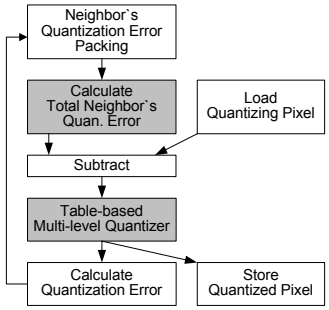

Fig.7. Data flow graph of vector error diffusion

Since the error diffusion process includes a quantization step inside of the feedback loop, it is difficult to increase the utilization ratio by processing multiple pixels. Even if C64x equips a sufficient number of functional units, the number of minimum cycles is 15 per pixel because of the long recursive loop. Although we can process three colors simultaneously, it is not efficient because C64x has only two identical data-paths.

In order to increase the number of parallelism, a skewed parallel processing algorithm is employed [4]. A table based quantization method is employed for multi-level quantization [6]. The performance is measured when the number of lines that are processed simultaneously is increased from one to three. As shown in Table 5, the performance is the best when processing two lines simultaneously. Although the parallelism increases as the number of lines to process rises, but this also demands more registers.

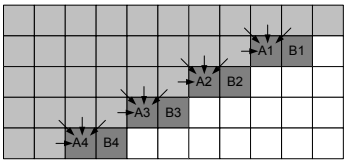

Fig.8. Skewing method for multi-pixel processing

Table 5. Optimization results of vector error diffusion

	1 line (intrinsic)	1 line (linear ASM)	**2 line (linear ASM)**	3 line (linear ASM)
cycles/pixel	19.27	19.16	**10.29**	10.53
Avg. unit utilization	2.95	2.95	**5.1**	4.9

5. CONCLUDING REMARKS

We developed a few methods for the efficient implementation of a digital color copier using TI's TMS320C6416 DSP processor. The developed code requires about 24 cycles for each color pixel, which translates that a 720 MHz C6416 CPU can perform all the real-time processing needed for a 20 PPM, 600 dpi, A4 size color copier. The number of required cycles for the color processing is not much increased when compared to the B/W copier implementation partly due to the improvement of the optimization methods [7]. Since the programmable DSP based architecture can support not only real-time image processing but also complex off-line functions, such as image compression, the DSP-based hardware and programs seem quite attractive for the implementation of next generation multi-function digital color copiers.

Table 6. Summary of the implementation

	Shading Correction	2D FIR filter	X-Zooming	Vector Error Diffusion	Total
Cycles/pixel in loop	1	9	4	10	24
Average unit utilization in loop	5.63	8.00	4.38	5.10	

6. ACKNOWLEDGMENTS

This study was supported by the Brain Korea 21 Project (0019-19990027) and the National Research Laboratory program (2000-X-7155) supported by the Ministry of Science and Technology in KOREA.

7. REFERENCES

[1] *TMS320C6414, TMS320C6415, TMS320C6416 Fixed-Point Digital Signal Processors*, Literature Num. SPRS146G, Texas Instruments, Mar. 2003.

[2] J. W. Ahn and W. Sung, "Pentium-MMX Based Implementation of a Digital Copier," *Proc. 1998 IEEE Workshop on Signal Processing Systems*, pp. 142-151, Oct. 1998.

[3] *TMS320C6000 Programmer's Guide*, Literature Num. SPRU198G, Texas Instruments, Aug. 2002.

[4] Jae-woo Ahn and Wonyong Sung, "Multimedia processor-based implementation of an error-diffusion halftoning algorithm exploiting subword parallelism," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 16, no. 2, pp. 129-138, Feb. 2001.

[5] N. Damera-Venkata and B.L. Evans, "Design and analysis of vector color error diffusion halftoning systems, " *IEEE Trans. on Image Processing*, vol. 10, pp. 1552-1565, Oct. 2001.

[6] Seokhoon Ju and Wonyong Sung, "VLIW SIMD architecture based implementation of a multi-level dot diffusion halftoning," *Proc. 2003 IEEE Workshop on Signal Processing Systems*, pp. 281-285, Aug. 2003.

[7] Taeksang Hwang and Wonyong Sung, "Implementation of a digital copier using TMS320C6414 VLIW DSP processor," *Proc. 2003 IEEE International Conference on Acoustics, Speech and Signal Processing Systems*, vol. 2, pp.621-624, Apr. 2003.