CRISP: Coarse-Grain Reconfigurable Image Signal Processor for Digital Still Cameras

Jason C. Chen[1], Chun-Fu Shen[2], and Shao-Yi Chien[1]
[1]Media IC and System Lab
Graduate Institute of Electronics Engineering, National Taiwan University
[2]Vivotek Inc.

***Abstract*— This paper presents a novel preview-based coarse-grain reconfigurable image signal processor (CRISP) for digital still cameras (DSCs). The two modes in DSCs, which have quite different hardware considerations, make traditional implementation methods inefficient. One is preview mode, which needs real-time constraints and the other one is picture-taking mode, which requires high flexibility and capability for various algorithms in it. Low cost design of CRISP considers simpler image pipelines in preview mode and extends flexibility required in picture-taking mode with proper hardware resources devotion. Algorithmic similarity in image pipelines and successful hardware classification lead it to a combination of low cost and high efficiency. Coarse-grain modules connected by reconfigurable interconnection make it a good compromise between dedicated hardware and DSPs, which are suitable for only one, not all of two modes in DSCs respectively. The experimental results show that the total gate count of it is 38.6K with 5.8K Byte memory. It can save more than 75% area from high end DSP, such as Trimedia TM1300. Besides, CRISP reduces execution cycle number of image pipeline tasks, such as 2-D filters to only 0.17% of that required by TM1300.**

I. Introduction

Digital Still Camera (DSC) market grows rapidly in recent years. The image resolution now is towards more than 10M pixels. As image data increase enormously, image processing pipelines in DSC [1] become vital. Image processing pipelines in DSC contain two different modes. One is pipeline in preview mode, which has real-time constraints for view-finding. This mode usually requires highly efficient hardware to meet real time constraints. The other is pipeline in picture taking mode which requires more complicated processing functions in order to generate high quality pictures [2]. This mode does not have real time constraints but requires flexibility to support various and complex algorithms.

Unlike image or video coding standards, such as JPEG and MPEG-4, there exits no standard for image processing algorithm and many implementation methods can be utilized. There are mainly two traditional ways to implement image processing pipeline in DSC: Application Specific Integrated Circuits (ASIC) [3] and DSP [4]. ASIC owns high efficiency and high performance for dedicated algorithms but it, however, does not have much flexibility for different algorithms. On the other hand, DSP can support many kinds of image processing algorithms, but its cost is much higher and it is not efficient at most time. Some other methods are also presented, such as array processors [5] and pixel processors [6]. Array processor has many processing elements arranged in 2-D array and can process many data simultaneously. It is not very applicable since high cost and data access bottleneck from memory. Pixel processors are used by attaching functional units behind each cell of image sensors for immediate processing. The cost and complexity, however, grow enormously and become unacceptable for high resolution applications.

Reconfigurable architecture seems to be a compromised solution between ASIC and DSP. It owns high efficiency and still remains enough flexibility for different application requirements. Thus it can be as efficient as ASIC is in preview mode and can be flexibly reconfigured, which is similar to what DSP does in picture-taking mode. Coarse-grain architecture [7] is composed with some processing elements (PEs) which are dedicated to some particular kinds of operations, such as addition and multiplication. It is very suitable for image pipelines in DSC, which contain many low-level image processing functions such as MACs and filters. Coase-grain architecture will achieve higher efficiency than FPGA [8] and more hardware sharing capability. Therefore, in this paper we design a coarse-grain reconfigurable image signal processor which has low hardware cost, high efficiency for preview modes and remains high flexibility for various algorithms in picture-taking modes.

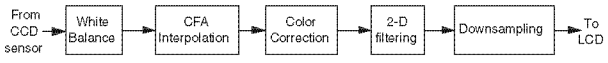

Fig. 1. Image processing pipeline example for preview mode.

II. Image Pipelines in DSC

In this section, image pipelines for preview mode and picture-taking mode will be introduced. Moreover, the classification of required algorithms will be presented.

A. Image Pipelines Introduction

As mentioned above, there are two modes of image processing pipelines in DSC. In preview mode, image pipelines help generate color images on LCD screen from captured raw image data. Pipelines for this mode require real-time capability for view-finding and are usually not very complicated. An image pipeline example for preview mode can be illustrated in Fig.1. In picture-taking mode, more complex image pipelines are required for larger scale of input image data processing

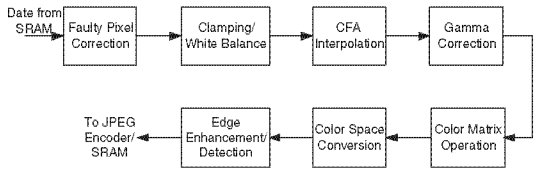

Fig. 2. Image processing pipeline example for picture-taking mode.

TABLE I
HARDWARE CLASSIFICATION OF TYPICAL IMAGE PROCESSING PIPELINE.

Classified Hardware Modules	Functionality
Internal memory and 2-D registers	Local data reuse
Interpolation module	Color interpolation
ALU module	General purpose operations
Pixel-based operation module	Brightness/Contrast enhancement, Hue/Saturation enhancement, Auto white-point detection, White-Balance, Color gain adjustment
Mul. and Acc. (MAC) module	Gamma correction, Edge enhancement, Anti-crosstalk, Defect pixel compensation
Resample module	Digital zoom-in/zoom-out

and high quality image storage. Various image pipelines are proposed, where no standard approaches exist and no real-time constraints are required. Therefore, highly flexible architectures are necessary pipeline in this mode. An image pipeline for picture-taking mode can be illustrated in Fig. 2.

B. Hardware Classification

Although image processing pipelines contain many processing blocks, their functionalities, however, share considerable algorithmic similarity [9]. After inspecting algorithms in image processing pipelines, we made an algorithmic classification based on hardware functionality, which is shown in Table I.

III. PROPOSED CRISP ARCHITECTURE

In this section, proposed processor architecture and its modules are shown based on the classification of Sec. II.

A. Processor Architecture Design

In this paper, a preview-based reconfigurable architecture design concept is proposed. Since low level image processing algorithms share lots of operation similarity, our idea is to devote low cost hardware for requirements in preview mode and add some hardware resources for higher flexibility and processing capability. In picture-taking mode, which has no real-time constraint, the hardware can be reconfigured to support algorithms having more complexity. Difference between traditional ASIC approach and our proposed approach is shown in Fig. 3 and Fig. 4. In Fig. 3, where ASIC approach is shown, dedicated hardware is used for image pipeline for both modes. Since this hardware can not be reconfigured, its hardware design consideration is worst case bound, as shown in Fig. 3(b). When in preview mode, Fig. 3(a), only small parts

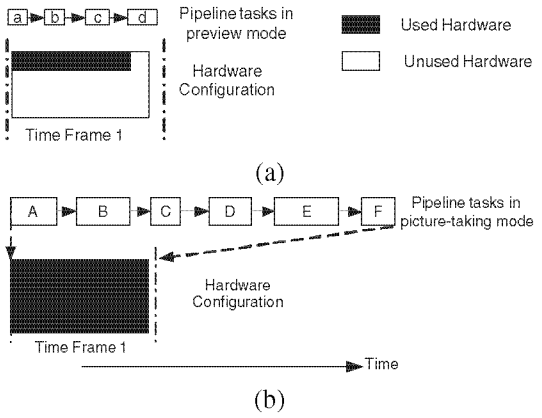

Fig. 3. Image pipelines in (a) preview and (b) picture-taking mode by ASIC.

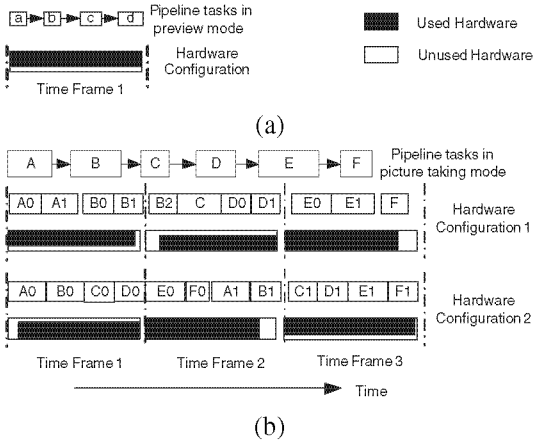

Fig. 4. Image pipelines in (a) preview and (b) picture-taking mode by CRISP.

of hardware are used because of the simpler tasks in preview modes, and it is not very effective since most devoted resources are unused. Figure. 4 shows image pipeline implemented by the proposed preview-based reconfigurable hardware. When in preview mode, most hardware resources are used to meet real-time requirements, as shown is Fig. 4(a). When taking a full resolution image, no real-time issue is required, and the hardware can be reconfigured to handle complex processing tasks. Figure 4(b) shows two possible configurations. Configuration 1 separates total pipeline by functional blocks, and each block can be separated into subblocks. In time frame 1, only block A and subblock B0, B1 are completed. In time frame 2, hardware is reconfigured and tasks before block E are finished. The whole image pipeline can be done in three time frames until block F is finished. It is also possible to separate image pipeline tasks by amounts of data. In configuration 2, image data is divided into two parts, and the whole pipeline flow can be done part by part.

Based on this idea, our proposed processor architecture is shown in Fig. 5. It contains coarse-grain reconfigurable PEs based on hardware classification in Sec. II. Each kind of PEs can be reconfigured respectively to process some operations. The outputs from each PE can be stored in the off-chip

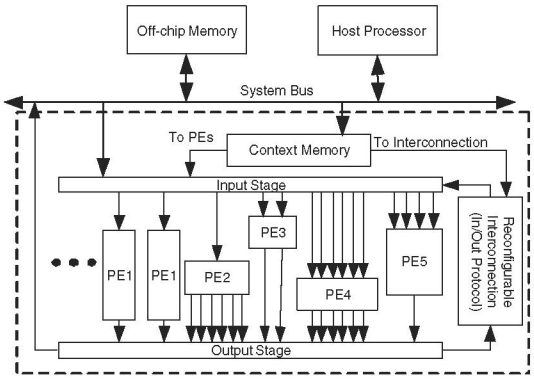

Fig. 5. Architecture of CRISP.

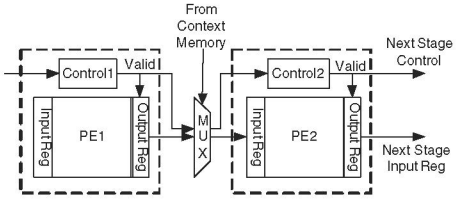

Fig. 6. Interconnection between two PEs.

memory for execution in next time frame. Moreover, some selected PEs can be connected by on-chip reconfigurable interconnection to perform more complex operations. The interconnection protocol of two PEs is shown in Fig. 6. The outputs of PE1 are designated to the inputs of PE2 by interconnection control from context memory. Besides data transfer, PE1 also passes a valid signal which generated by control unit in PE1. When the outputs of PE1 are ready to the inputs of PE2, the valid signal is asserted and data transfer is permitted. Also, the control unit in PE2 start to generate its valid signal for next stage transfer. Finally, the number of each kind of PEs is also changeable for different requirements.

B. PE Module Design

According to the classification in Sec. II, PE modules can be designed as follows.

1) Memory Module: Memory modules are one of the most important part in image processing . These modules are used for local data reuse and 2-D data extraction. One memory module is composed of three kinds of elements: eleven line buffers, three 2x2, and four 3x3 window register files, which are shown in Fig. 7(a). Every kind of elements can be combined with each other and can achieve much more flexible memory usage. Figure 7(b) shows two possible memory configurations.

2) Color Interpolation Module: Color interpolation modules are used to handle Bayer pattern image data interpolation. Outputs of this module are stored in registers of ALU modules and therefore can be combined with ALU modules to perform some complicated interpolation algorithms [10]. Figure 8(a) shows top view of this module and every block represents a FU shown in Fig. 8(b). C and D in Fig. 8(a) represents *cross*

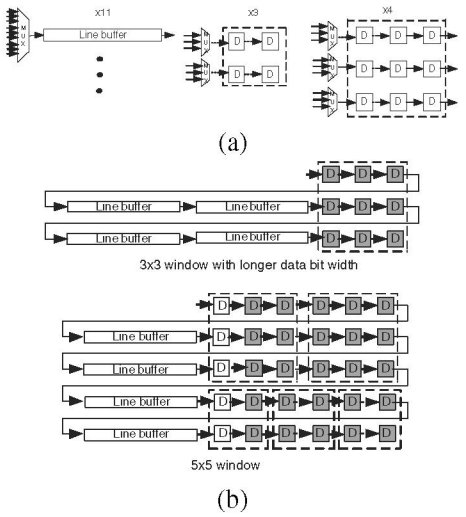

Fig. 7. (a) Memory module elements and (b) some of their combined configurations.

and *diagonal* named by the input source from 3x3 window registers. C1, C2, D1, and D2 can access data in two of four 3x3 window registers and C3, C4, D3, and D4 can access data in the rest two. Outputs of C12, D12, C34, D34, C1234, and D1234 can be sent to registers in ALU modules, where the interpolated RGB data can be selected.

3) Pixel Operation Module: These modules will do operations pixel by pixel. Data flow of this module is serial-in and serial-out pixel by pixel.

4) ALU Module: ALU modules are similar to simple datapath in general purpose processors. They can be combined with other modules for more complex algorithms. It contains ALUs, multipliers, comparators and some registers.

5) Multiplication and Accumulation (MAC) Module: MAC modules are used to handle 2-D filters and matrix operations. There are nine FUs and each FU can generate its own outputs or be connected to perform matrix operations, as shown in Fig. 9. They can also be combined with sorters in ALU modules for nonlinear filter operations, such as median filters.

6) Resample Module: Resample modules are required for digital zoom-in and zoom-out. They can support fractional zoom-in/zoom-out ratio. Four points Bilinear interpolation is used so one line buffer is required for one color channel resampling.

IV. Experimental Results

A. Hardware Cost

The Verilog HDL description of the design was synthesized with the Synopsys Design Compiler tool. The UMC Artisan 0.18um ASIC library was used to map the design with results shown in Table II. The total area is 4.066mm^2 which is much smaller than today's high-end DSP, such as Trimedia TM1300 [11], whose area is 16.9mm^2 with 0.18um process. The proposed processor saves over 75 % area from traditional DSP solution.

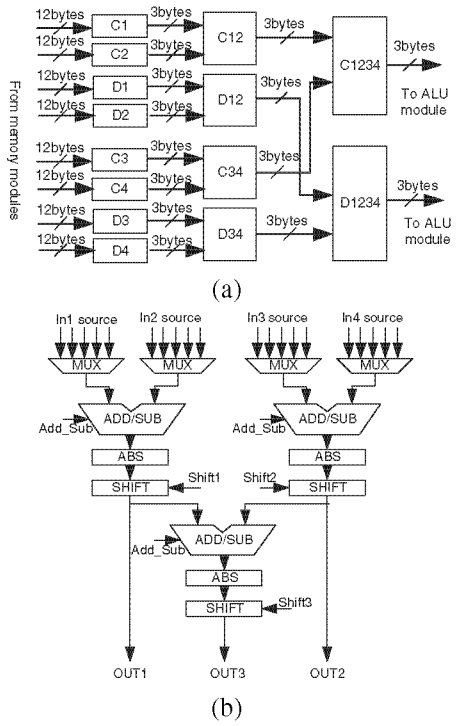

Fig. 8. (a) Color interpolation module and (b) its FUs .

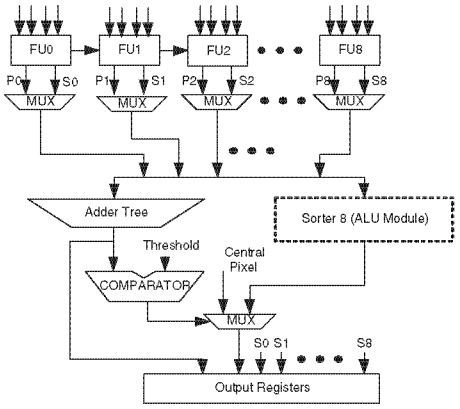

Fig. 9. MAC modules.

TABLE II

GATE COUNT OF EACH KIND OF MODULES IN CRISP.

Modules	Gate count	Memory size(byte)
Memory	2,880	5,632
Color Interpolation	7,304	
Pixel Operation	2,731	256
ALU	4,956	
MAC	16,227	
Resample	5,152	
Total	38,650	5,888

TABLE III

EXECUTED CYCLE NUMBER OF TYPICAL IMAGE PIPELINE TASKS.

Functional Blocks(Algorithms)	TM1300	CRISP	CRISP/TM1300
White Balance($ax+y$)	5,004,000	262,144	5.23%
Color Interpolation(Bilinear)	19,675,000	262,144	1.33%
2-D Filter(3x3 Window)	158,113,843	262,144	0.17%
Color Space Conversion(3x3 Matrix)	44,031,000	262,144	0.60%
Resample(Bilinear)	119,843,742	262,144	0.20%

B. Performance

512x512 test image is used for deriving cycle count and experimental results can be shown in Table III. Our coarse-grain PEs can be successfully reconfigured to execute these tasks with one cycle per pixel respectively. Moreover, it is possible to combine several tasks by on-chip reconfigurable interconnection in one execution cycle. Therefore, the total cycle number can be much less than that of DSP execution.

V. CONCLUSION

In this paper, CRISP is proposed for image pipelines in DSC. CRISP can meet the real time constraints for preview modes, and also can be properly reconfigured for picture-taking modes. Compared with traditional high end DSP, such as Trimedia TM1300, CRISP can save more than 75% hardware cost. Moreover, the executed cycle number of functional blocks, such as 2-D filters can be reduced dramatically to only 0.17% of that executed by Trimedia TM1300.

REFERENCES

[1] J. Adams, K. Parulski, and K. Spaulding, "Color processing in digital cameras," *IEEE Micro*, vol. 18, no. 6, pp. 20–30, Nov. 1998.

[2] K. Illgner, H.-G. Gruber, P. Gelabert, J. Liang, Y. Yoo, W. Rabadi, and R. Talluri, "Programmable DSP platform for digital still cameras," in *Acoustics, Speech, and Signal Processing, 1999. ICASSP '99. Proceedings., 1999 IEEE International Conference on Volume 4*, Mar. 1999, pp. 280–285.

[3] D. Doswald, J. Hafliger, P. Blessing, N. Felber, P. Niederer, and W. Fichtner, "A 30-frames/s megapixel real-time CMOS image processor," *IEEE J. Solid-State Circuits*, vol. 35, no. 11, pp. 1732–1743, Nov. 2000.

[4] [Online]. Available: http://www.ti.com/

[5] R. Etienne-Cummings, Z. Kalayjian, and D. Cai, "A programmable focal-plane mimd image processor chip," *IEEE J. Solid-State Circuits*, vol. 36, no. 1, pp. 64–73, Jan. 2001.

[6] P. Mandolesi, P. Julian, and A. Andreou, "A scalable and programmable simplicial CNN digital pixel processor architecture," *IEEE Trans. Circuits Syst. I*, vol. 51, no. 5, pp. 988 – 996, May 2004.

[7] H. Singh, M.-H. Lee, G. Lu, F. Kurdahi, N. Bagherzadeh, and E. Chaves Filho, "Morphosys: an integrated reconfigurable system for data-parallel and computation-intensive applications," *IEEE Trans. Comput.*, vol. 49, no. 5, pp. 465 – 481, May 2000.

[8] J. Rose, A. El Gamal, and A. Sangiovanni-Vincentelli, "Architecture of field-programmable gate arrays," *IEEE Proceedings*, vol. 81, no. 7, pp. 1013 – 1029, 1993.

[9] P. W. Wong, D. Tretter, C. Herley, N. Moayeri, and R. Lee, "Image processing considerations for digital photography," in *Compcon '97. Proceedings*, Feb. 1997, pp. 280–285.

[10] S.-C. Pei and I.-K. Tam, "Effective color interpolation in ccd color filter arrays using signal correlation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 6, pp. 503–513, 2003.

[11] K. Vissers and S. Mirolo, "The performance and power consumption of the TriMedia TM32 VLIW media processor core," in *Cool Chip IV*, Apr. 2001.