doi:10.3772/j.issn.1006-6748.2020.04.010

Design of a clustered data-driven array processor for computer vision①

Shan Rui(山　蕊)②*, Deng Junyong*, Jiang Lin**, Zhu Yun*, Wu Haoyue*, He Feilong*
(* School of Electronic and Engineering, Xi'an University of Posts and Telecommunications, Xi'an 710121, P. R. China)
(** Integrated Circuit Laboratory, Xi'an University of Science and Technology, Xi'an 710054, P. R. China)

Abstract

Computer vision (CV) is widely expected to be the next big thing in emerging applications. So many heterogeneous architectures for computer vision emerge. However, plenty of data need to be transferred between different structures for heterogeneous architecture. The long data transfer delay becomes the mainly problem to limit the processing speed for computer vision applications. For reducing data transfer delay and fasting computer vision applications, a clustered data-driven array processor is proposed. A three-level pipelining processing element is designed which supports two-buffer data flow interface and 8 bits, 16 bits, 32 bits subtext parallel computation. At the same time, for accelerating transcendental function computation, a four-way shared pipelining transcendental function accelerator is designed, which is based on Y-intercept adjusted piecewise linear segment algorithm. A distributed shared memory structure based on unified addressing is also employed. To verify efficiency of architecture, some image processing algorithms are implemented on proposed architecture. Simultaneously the proposed architecture has been implemented on Xilinx ZC 706 development board. The same circuitry has been synthesized using SMIC 130 nm CMOS technology. The circuitry is able to run at 100 MHz. Area is 26.58 mm^2.

Key words: array processor, data-driven, adjacent interconnection, distributed memory, computer vision (CV)

0 Introduction

Computer vision (CV) is widely expected to be the next significant technique in emerging applications[1-3]. CV algorithms can be divided into 3 types according to complexity: low-level, intermediate-level and high-level[4]. Low-level vision is the estimation of depth, motion, shape, and other physical scene properties from visual measurements[5]. Since these algorithms are fully predetermined and act identically on all input data and the initial and final data structures are fixed arrays the algorithms are ideally suited to single instruction, multiple data (SIMD) machines[6]. Intermediate-level and high-level are the critical stage in a computer vision system. The data structures are not fixed in size, and no simple one-to-one or many-to-one fixed and even distribution of data to processors exists[6]. Thus, parallel implementation of these algorithms is more suited to a multiple instruction, multiple data (MIMD) architecture. From above, it can be seen that different levels employ different methods to accelerate for getting the best acceleration. So, many heterogeneous architectures for CV emerge[7-10] recently. However, the high delay of data communication in heterogeneous architectures becomes a main problem to limit the computing efficiency for CV applications.

In order to reduce data communication delay and enhance the computing efficiency of CV application, a clustered data-driven array processor (CD-DAP) is proposed, which can support SIMD and MIMD at the same time. Multiple processing elements (PEs) are arranged in MESH structure, and connected by short line adjacent interconnection. Single PE is designed with pipelining and works as the mode of dataflow. The main contributions are summarized as follows:

- A double-buffer interface structure based on dataflow driven is designed, and it can realize ultra-low data transfer delay between adjacent PEs.
- A four-way shared pipeline transcendental function accelerator is designed. The resource utilization and computing efficiency are enhanced.

① Supported by the National Natural Science Foundation of China (No. 61802304, 61834005, 61772417, 61634004, 61602377), Shaanxi Provincial Co-ordination Innovation Project of Science and Technology (No. 2016KTZDGY02-04-02), Shaanxi Provincial Key R&D Plan (No. 2017GY-060), Shaanxi International Science and Technology Cooperation Program (No. 2018KW-006).

② To whom correspondence should be addressed. E-mail: shanrui0112@163.com

Received on Sep. 21, 2019

• A distributed shared memory structure based on unified addressing is employed. Through a way of highly efficient interconnection, the data accessing delay can be reduced dramatically.

This paper is organized as follows. Section 1 introduces related work and the motivation. After that, clustered data-driven array processor will be proposed in Section 2. Section 3 describes the mapping of computer vision algorithm. In Section 4, simulation and performance analysis will be given. The conclusion is given in Section 5.

1 Related work and motivation

Many researchers pay attention to acceleration architectures for computer vision. CPU + GPUs is one of widely employed structure. For example, Ref. [3] presented an implementation of OpenVX directed at CPUs and GPUs platforms, and discussed these analytical techniques in detail. Ref. [11] discussed that how to use GPUs' hundreds of gigaflops of processing power to realize image processing and computer vision through a way of parallel programming. Ref. [12] presented a method which can adapt CPU-GPU task parallelism, sliding window parallelism, scale image parallelism, dynamic allocation of threads, and local memory optimization. Meanwhile a face detection algorithm was realized using proposed method to accelerate computing time. CPU is mainly used to execute serial part of CV algorithms, and GPU is mainly used to compute parallel part of CV algorithms. So, CPU + GPUs can accelerate the computing part of CV algorithms well, but it is inevitable that plenty of data need to be transferred between CPU memory and GPU memory. The performance improvement is limited by the long data transfer delay.

For satisfying the necessary real-time performance for many computer vision applications, dedicated hardware accelerators are often designed, such as in Ref. [13], a hardware-based stereo matching architecture was proposed which aims to provide high accuracy and concurrently high performance in embedded vision applications. The architecture integrated an image filter and an edge-preserving filter. Dedicated hardware accelerators can satisfy the requirement of real time application well, but it cannot adapt to variety algorithms. Once algorithm changed, the hardware must be redesigned. The cost is very high and the time to market is long.

Field programmable gate arrys (FPGAs) are also used in accelerating computer vision applications, such as in Ref. [14], an FPGA-based emulation framework was proposed that can provide dynamic vulnerability analysis for hardware-accelerated computer vision applications. Ref. [15] proposed a FPGA-based accelerator architecture that can tackle a range of standard CV algorithms. The architecture consists of pipelined processing elements that can be configured to support various belief propagation settings for different CV tasks. FPGAs can shorten the time to market slightly, but it also needs to be redesigned once algorithm changes.

For adapting to the fast-changed computer vision algorithms, multi-core array architectures are introduced, such as in Ref. [16], a polymorphous array processor was proposed that consists of several levels of clusters of processors and seamlessly integrates data parallelism, thread parallelism, operation parallelism, and distributed instruction parallelism. And OpenVX is implemented on this architecture. Ref. [17] proposed a novel framework that is for fast prototyping and optimization of OpenVX applications for heterogeneous SoCs with many-core accelerators. Ref. [18] presented a method for early parallel performance estimation on embedded multiprocessors from sequential application traces. Ref. [19] proposed a dynamically reconfigurable hybrid architecture for vision processing. So multi-core array architectures are very promising target for CV applications.

Based on above, CD-DAP architecture is proposed. On one hand, it can support data-level parallelism well, and it can execute serial computation well. At the same time, it supports flexible programming. One the other hand, the data transfer delay can be reduced sharply because data do not need to be transferred between different frameworks.

2 Clustered data-driven array processor

CD-DAP architecture consists of PE array, memory banks (MBs) array, fast switching unit and router as shown in Fig. 1. PE array, MB array, and one fast switching unit are constructed one cluster. PE array includes 16 pipelined PEs and 4 transcendental function accelerators. PEs in array are connected with adjacent interconnection. One transcendental function accelerator is shared by 4 adjacent PEs. Data communication in one cluster is realized through fast switching unit and adjacent interconnection. Data communication between clusters is realized through routers. The proposed structure can be easily scaled with routers.

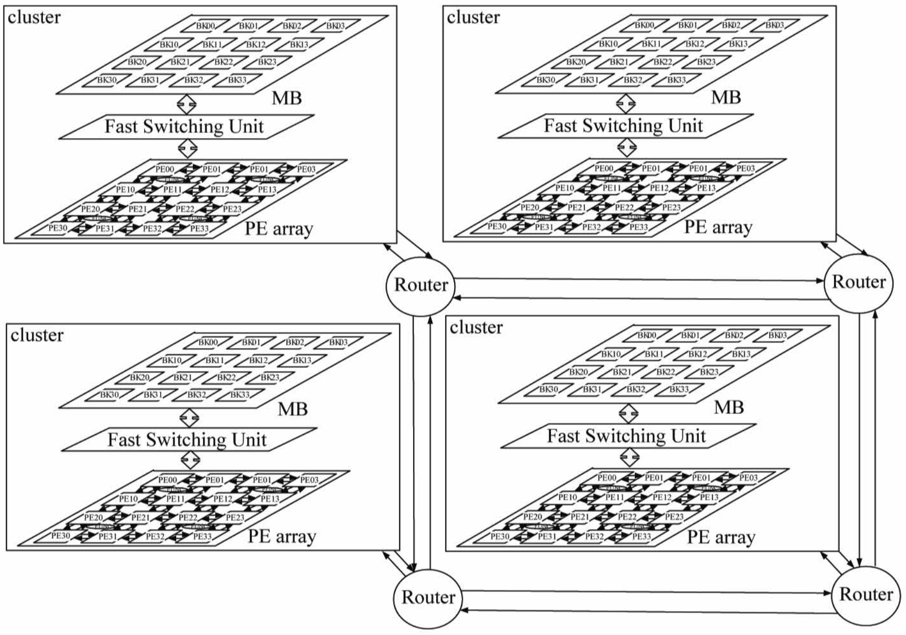


Fig. 1 CD-DAP architecture

2.1 Pipelined PE based on two-buffer dataflow driven interface

Three-level pipelining architecture is employed in PE design. The first level is mainly used to read code from configurable RAM. The second level is mainly used to receive data from 4 input ports, register file, or h-register. The third level is mainly used to process buffered data from up-level and send processing results to corresponding destination, may be to 4 output ports, register file, or h-register. The third level can be fired only when the needed data arrive.

Its detail architecture is shown in Fig. 2. It consists of 4 two-buffer units, Pc fresh unit, instruction RAM, data receiving unit, data processing unit, generate ready unit, data fan out unit, control unit, pipeline code, register file and h-register.

Two-buffer unit is in charge of receiving data from east, west, south, or north direction. In order to reduce data transfer delay, double buffers are used, which are alternately working. Simultaneously the buffer is realized through dataflow driven. When the input data is arriving, and at least one buffer is empty or buffers is full but the buffered data is already being received by next level unit, the buffer is fired to save. The delay of data transfer between adjacent PEs can be near zero.

Pc fresh unit is used to refresh the value of program point, which decides the next code address. It is controlled by current stage of pipelining. If the code which is already in third level is processing finished, Pc fresh can be done immediately. The detail value is decided by current finished code. If the code is jumping or branch instruction, it is refreshed by corresponding jumping address, or Pc is just added by one.

Data receiving unit is in charge of saving data for data processing unit, including left data and right data. Left data may be from 4 input ports, register file, or h-register, and right data may be from 4 input ports, register file, immediate or h-register. And saving can be done only when the next pipeline is finished.

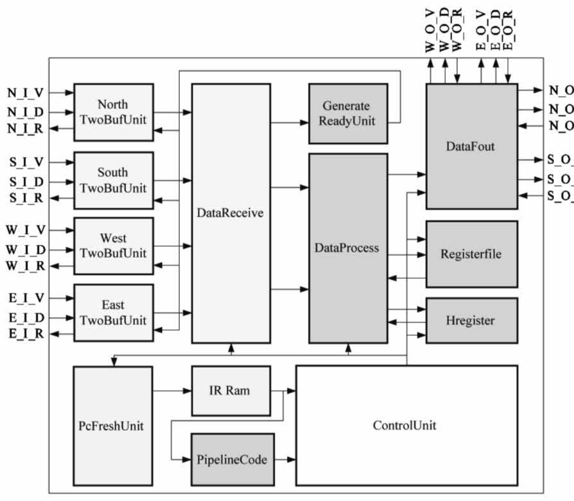


Fig. 2 Architecture of pipelined PE

Data processing unit is in charge of processing data from data receiving unit. Some operations are supported, such as add, add immediate, sub, sub immediate, shift, and, or, xor, not, branch, jump, multiply, nop. For satisfying requirements of computer vision application better, sub-word parallelism operation is also supported, including four 8 bits add/sub operations in one code, two 16 bits add/sub operations in one code.

Control unit is the key part, which controls Pc refreshing, data receiving and processing, ready signals generating, and register file writing. Pipeline code is used to buffer code from instruction RAM. And the buffered code is used to control data processing, result fan out, ready signals generating, and register file writing. Instruction RAM is mainly used to save application codes, which are from H-tree network. The H-tree network can be seen in Fig. 3. Register file includes 32 generate registers. H-register includes 2 specific registers, which are used to save the result from multiply. Hi is used to save upper 32 bits of result, and ho is used to save lower 32 bits of result. Some auxiliary circuits are also used to realize by-pass for avoiding data hazard.

Generate ready unit is used to generate ready signals to east, west, south, or north direction, indicating the data from east, west, south, or north direction whether be received already. Data fan out unit is used to send result from data processing unit to corresponding destination according to code. There are 4 direction output registers. Only the register is empty or the register is full but the data is already received, the result can be written.

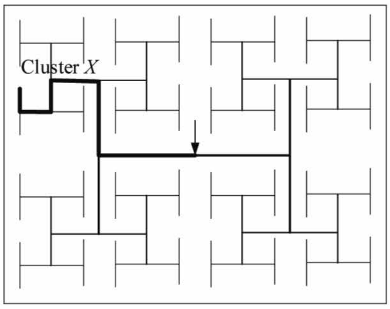


Fig. 3 The architecture of H-tree

2.2 Transcendental function accelerator

Transcendental function accelerator supports sin, cos, square, square-root, log, exponent, and arctan computing. One transcendental function accelerator is shared by 4 adjacent PEs, as shown in Fig. 4.

The architecture of transcendental function accelerator is also shown in Fig. 5. It comprises one pre-processing unit, 2 sin/cos function pipeline units, 2 other function pipeline units, and one fan out unit.

Pre-processing unit is used to distribute requests from 4 PEs to computing pipeline. When conflicting between requests occurs, a random arbitration algorithm is employed. Fan out unit is used to send computation result to corresponding output ports according to the result of arbitration. So, the result of arbitration from pre-processing unit must be pipelined as the same as data.

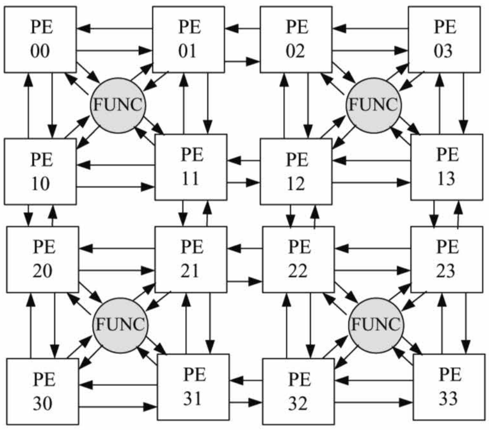


Fig. 4 The shared architecture

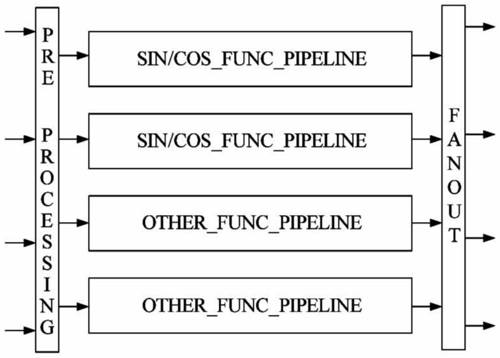


Fig. 5 Architecture of transcendental function accelerator

Sin/cos function pipeline unit is used to execute sin and cos computation. The processing delay can reach one cycle, because piecewise linear algorithm is employed. According to the result of software statistic, the style of 16 segments has lower error and fast speed hardware, so it is employed in this work. Its circuit can be seen in Fig. 6.

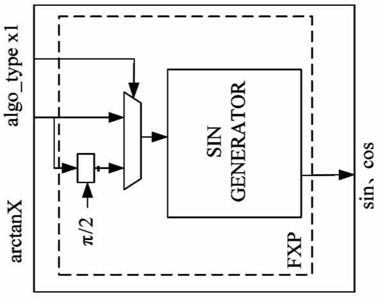


Fig. 6 Architecture of sin/cos function pipeline unit

Other function pipeline unit has three-level pipeline structure. Logarithmic system is employed for simple computation. The input data must be translated into logarithm form, and then execute corresponding add, sub, shift operations, finally translate the result into exponent form. For example, the processing of square is shown as Eq. (1).

$$x^2 = 2^{\log_2 x^2} = 2^{2\times\log_2 x} = 2^{(\log_2 x) << 1} \tag{1}$$

So other function pipeline unit has one log convertor, some auxiliary circuits, and one pow convertor, which can be seen in Fig. 7. In this paper, the log and pow convertors are realized which employs piecewise linear algorithm. A style of 16 segments is also used, and the convertor errors can be reached 0.011% and 0.013% for log and pow respectively.

2.3 Distributed shared memory structure

A distributed shared memory structure is designed in this paper. Its architecture is shown in Fig. 8(a). For long distance data communication, a network on chip (NoC) is used. A 4 virtual channels router is designed, supporting packet switching. Four 32 bits data will be transferred once. *XY* routing algorithm is employed. The delay of virtual channel router can be reached at one cycle.

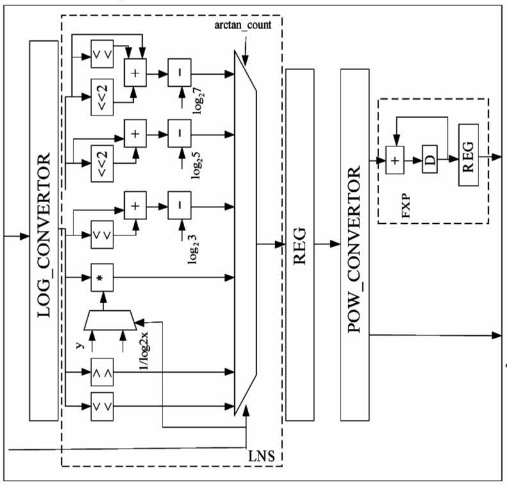
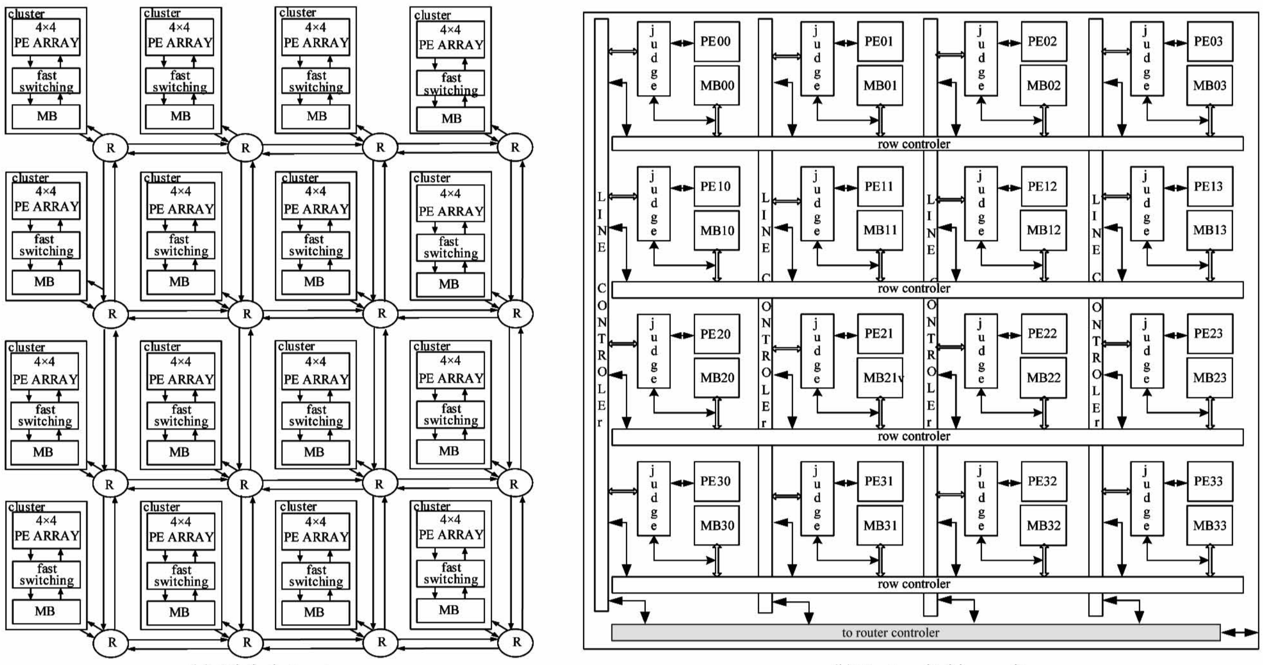


Fig. 7 Architecture of other function pipeline unit



(a) Global structure (b) Fast switching unit

Fig. 8 Architecture of distributed shared memory

For data accessing in cluster, a fast switching unit is designed. Allowing for data accessing frequency, the request from local MB can be responded immediately, and the request from other MBs must be judged through line-row 2 level controllers, and finally arrives at destination MB. The detail structure of fast switching unit is shown in Fig. 8(b).

Judge unit is used to receive request from PE, and dispatch the request to local MB or line controller according to request address. Line controller is mainly used to receive requests from the same line PEs, and dispatch 4 requests to corresponding row controller or router controller. If more than one requests need to access the same row controller, a random arbitration algorithm is employed to judge these requests. No replied requests will be postponed. Router controller is in charge of receiving requests from 4 line's controllers and choosing one to router. Row controller is in charge of receiving requests from different lines and dispatching these requests to corresponding MB according to request address. If more than one requests need to the same MB, a random arbitration algorithm is employed to judge these requests. No replied requests will be also postponed.

3 Computer vision algorithm mapping

To study the efficiency of proposed architecture, 3 computer vision algorithms are realized on proposed architecture: Sobel edge detection, Canny edge detection, and Harris corner detection.

3.1 Sobel edge detection

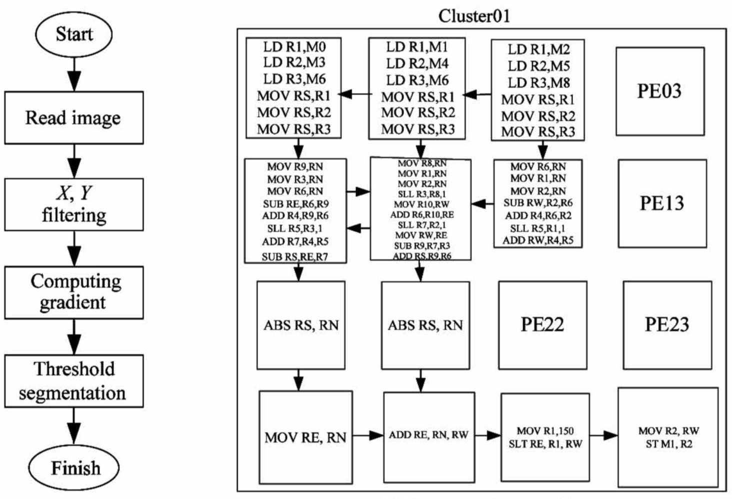
Edge detection is a fundamental technique for image segmentation, feature extraction and object tracking[20]. In edge detection, Sobel operator is a kind of commonly used template. The process of Sobel edge detection is shown in Fig. 9(a). The Sobel filtering matrices for the X and Y directions are shown in Eq. (2). And the process of filtering can be seen in Eq. (3).

$$sobel - x = [-1, -2, -1; 0,0,0; 1,2,1]$$
$$sobel - y = [-1,0,1; -2,0,2; -1,0,1] \tag{2}$$

$$\begin{aligned} filtering - x &= (m6 + 2 \times m7 + m8) \\ &\quad - (m0 + 2 \times m1 + m2) \\ filtering - y &= (m2 - m0) - (2 \times m5 - 2 \times m3) \\ &\quad + (m8 - m6) \end{aligned} \tag{3}$$

Where $m0 - m9$ stands for 3×3 source image data. For mapping Sobel edge detection on proposed architecture, one cluster is used. The mapping details are shown in Fig. 9(b). PE00, PE01, and PE02 are used to load 3×3 source image data, and send them to PE10, PE11, and PE12 to execute x and y directions filtering. For taking full advantage of data reusability, PE00, PE01, PE02 all need read data of 3×3 matrixes at the starting of each line. Next, only PE02 needs read data from memory until this line is finished, and PE00 and PE01 just receive data from PE02. They work like a sliding window.

PE10 is mainly used to receive data from PE00, sends the result of $m2 - m0$ to PE11 firstly, computes $m0 + 2 \times m1 + m2$ next, receives the result of $m6 + 2 \times m7 + m8$ from PE11, and executes sub operation getting the final result of x direction filtering Gx.



(a) The process (b) The mapping details

Fig. 9 Sobel edge detection

PE11 is mainly used to receive data from PE01, computes $2 \times m5 - 2 \times m3$ firstly, receives the result of $m2 - m0$ from PE10, receives the result of $m8 - m6$ and $m8 + 2 \times m7 + m6$ from PE12, sends the result of $m6 + 2 \times m7 + m8$ to PE10, executes add operation getting the final result of y direction filtering Gy.

PE12 is mainly used to receive data from PE02, computes $m8 - m6$ and $m8 + 2 \times m7 + m6$, and sends the results to PE11. PE20 and PE21 are used to execute absolute operation. PE31 is used to execute $|Gx| + |Gy|$. PE32 is used to compare the threshold value. PE33 is used to write back results.

3.2 Canny edge detection

Canny edge detection[21] is a kind of classic image edge detection algorithm, and it has wide application. The process of Canny edge detection is shown in Fig. 10(a). Gussian filtering is implemented on cluster 01. Each of PE00 and PE32 is used to process 6 lines image data, each of PE01 – PE31 is used to process 4 lines image data. PE33 is used to send result to next cluster.

Gradient computation and threshold segmentation are implemented on cluster 02. PE33 is used to receive filtering results from upper cluster. PE32, PE23, PE22, PE21, PE11, PE12 are used to compute gradient. PE10 is used to compare the threshold value and write back the results. The details can be shown in Fig. 10(b).

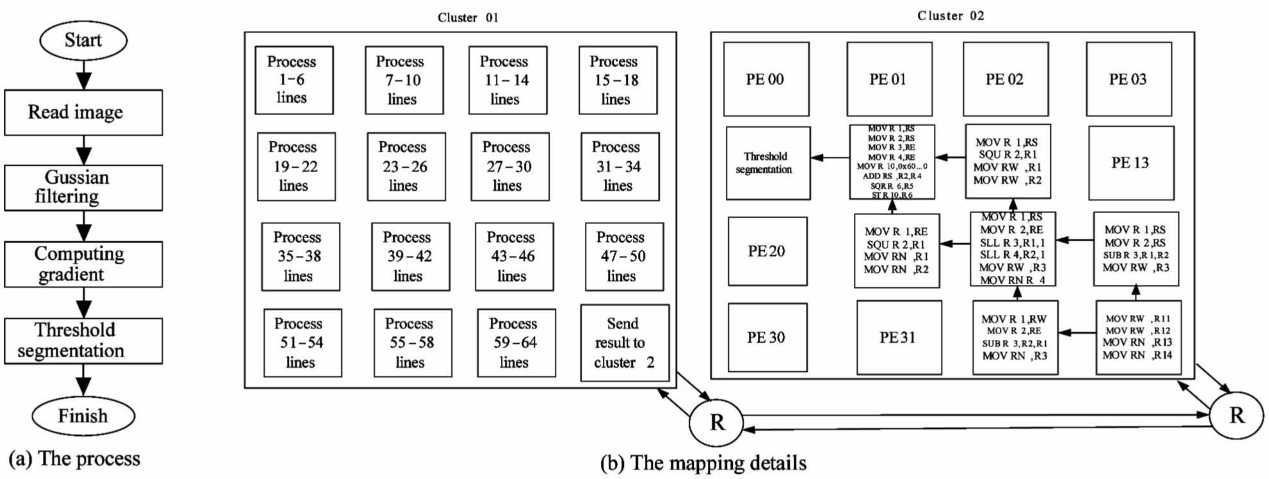


Fig. 10 Canny edge detection 3.3 Harris corner detection

Harris corner detection[22] is widely used in the area of feature extraction of computer vision algorithm. It is simple and has strong stability. The process of Harris corner detection is shown in Fig. 11 (a).

Three clusters are employed to map it. Cluster 0 is used to implement Sobel filtering. There PE00, PE01, PE02 are used to compute the results of x direction filtering Gx. PE30, PE31, PE32 are used to compute the

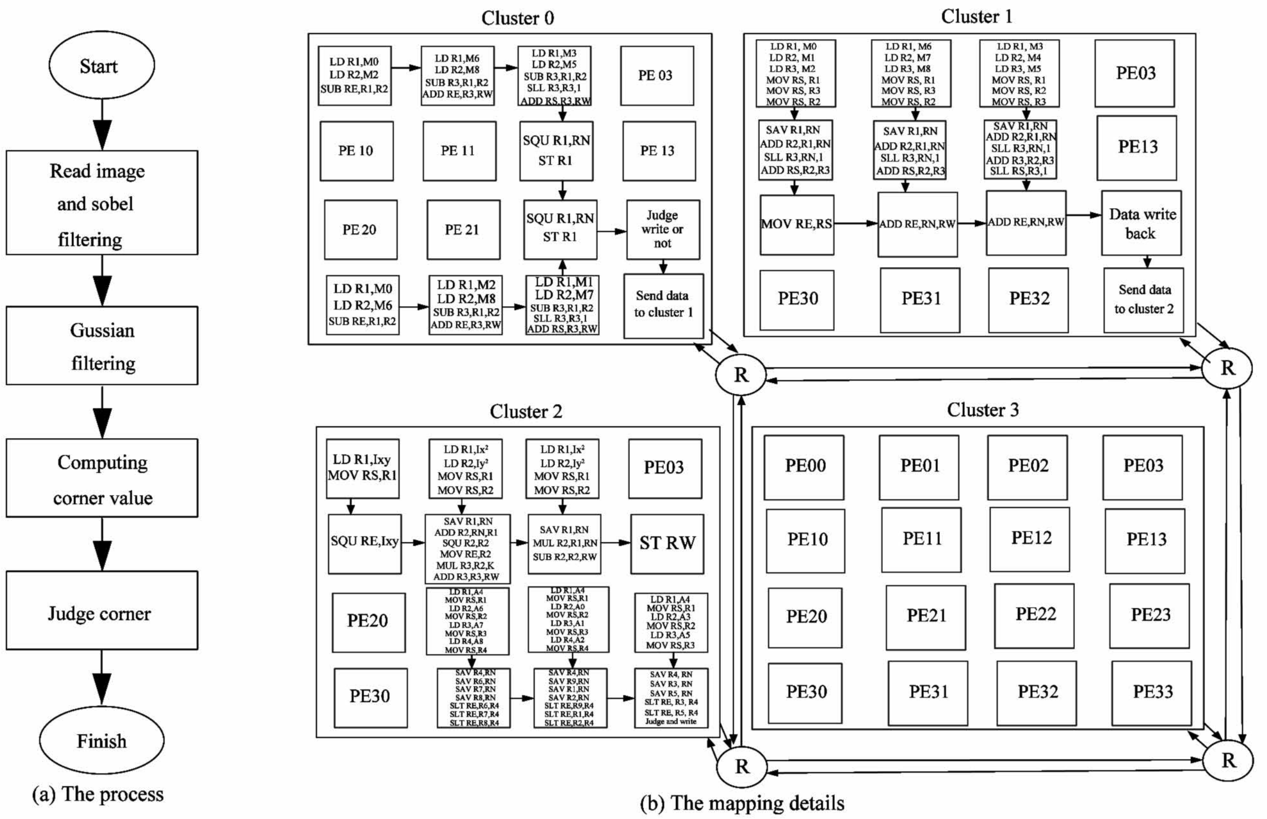


Fig. 11 Harris corner detection

results of y direction filtering Gy. PE12 is used to compute the result of Gx^2. PE22 is used to compute the result of Gy^2. PE23 is used to judge writing back or not. PE33 is used to send results to cluste 1. The details can be shown in Fig. 11(b).

Cluster 1 is in charge of Gussian filtering. The processing of Gussian filtering can be shown in Eq. (4).

$$\begin{aligned} Gussian = {} & (m0 + 2 \times m1 + m2) \\ & + (m6 + 2 \times m7 + m8) \\ & + 2 \times (m3 + 2 \times m4 + m5) \end{aligned} \tag{4}$$

In cluster 1, PE00, PE01, PE02 are used to read 3 × 3 image data, and send them to PE10 – PE12 respectively. PE10 – PE12 receive data from upper PEs. Simultaneously, PE10 computes $m0 + 2 \times m1 + m2$. PE11 computes $m6 + 2 \times m7 + m8$. PE12 computes $2 \times (m3 + 2 \times m4 + m5)$. PE21 is used to compute $(m0 + 2 \times m1 + m2) + (m6 + 2 \times m7 + m8)$, and send the result to right PE. The final filtering result can be gotten from PE23. PE33 is used to send the results to cluster 2. The details can be shown in Fig. 11(b).

Cluster 2 is in charge of corner value computation and corner judging. PE00 – PE13 are used to compute corner value. When they finish, PE21 – PE33 will start to work and judge whether a corner or not. The details can be shown in Fig. 11(b).

4 Simulation and performance analysis

Some computing vision algorithms are realized on proposed structure. The results can be seen in Fig. 12. Image in Fig. 12(a) is original image. Images in Figs12 (b) – (g) are the result of Sobel edge detection, Canny edge detection, Harris corner detection, Fast corner detection, Gassion pyramid, and Histogram respectively.

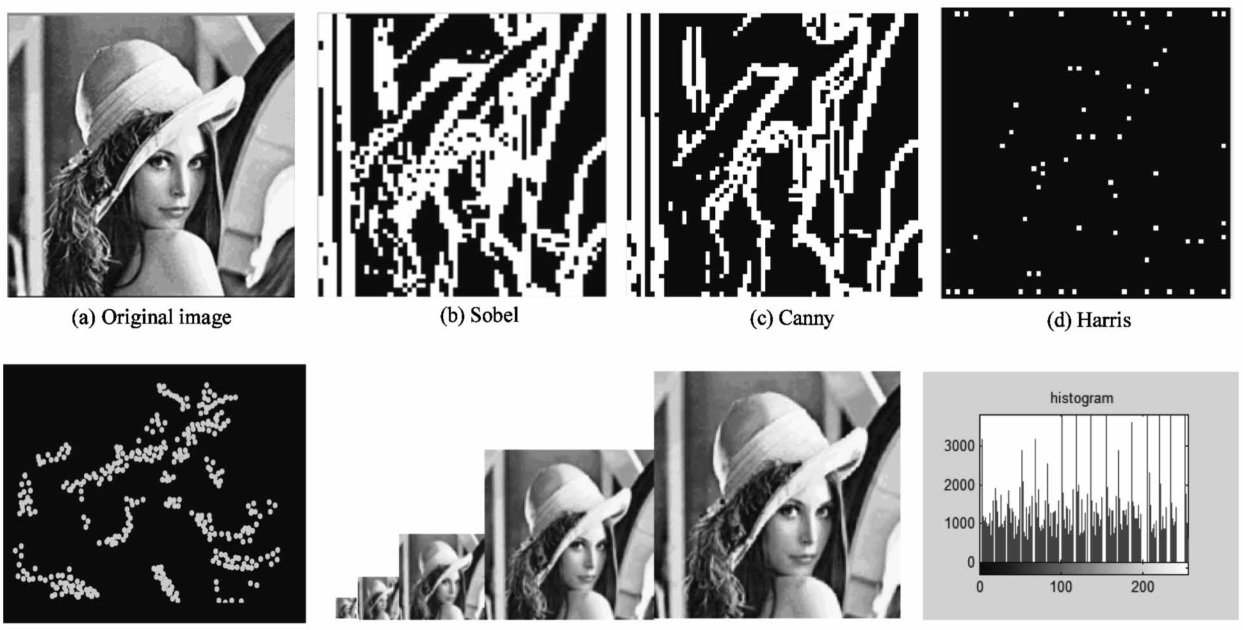


Fig. 12 The running results of computer vision algorithms

Computing time of Sobel, Canny, Harris, Fast, Gaussian pyramid, and Histogram realized on CD-DAP and the results of peek signal to noise ratio (PSNR) and root-mean-square error (RMSE) are shown in Table 1.

Meanwhile, computing time of Sobel, Canny, Harris realized on CD-DAP compared with which realized on FPGA and GPU can be seen in Table 2. From this table, it can be seen that totally processing time of a 512 × 512 image needs 0.283 ms for Sobel, 0.682 ms for Canny, 4.985 ms for Harris on CD-DAP, which is faster than FPGA and GPU.

CD-DAP based on one cluster including 16 PEs, one transcendental function accelerator, one fast switch unit and one router has been implemented on Xilinx ZC 706 develop board. The source occupation can be seen in Table 3. The synthesis result of 4 clusters and the result compared with other implementation architectures can be shown in Table 4. The circuit can be run at 100 MHz.

Table 1 Performance statics of some algorithms

Algorithm name	Image size	# PEs	PSNR	RMSE	Delay
Sobel	512 × 512	16	6.19	125	0.283
Canny	512 × 512	32	5.997	127.8	0.682
Harris	512 × 512	64	5.726	132	4.985
Fast	64 × 64	16	5.963	128	2.9
Gaussian pyramid	64 × 64	16	11.917	64.67	1.03
Histogram	64 × 64	16	7.290	110.2	98.2

Table 2 Computing time of Sobel, Canny, Harris (unit: ms)

Structure	Image size	Sobel	Canny	Harris
FPGA[23-25]	512 × 512	1.31	0.46	18.57(640 × 480)
GPU[26-28]	512 × 512	2.1	3.31	1.7(560 × 555)
CD-DAP	512 × 512	0.283	0.682	4.985

Table 3 Synthesized results for one cluster of CD-DAP

Module	FF	LUT	36k BRAM	Power (W)
16 × PE	3 904	26 400	16	-
Transcendental function accelerator	927	5 322	0	-
Fast switch unit	1 310	6 432	0	-
Router	140	746	0	-
Total	6 281	38 900	16	0.844

The same circuit has been synthesized using SMIC 130 nm COMS technology. The circuit can be run at 100 MHz. Area is 26.58 mm^2. Compared with Refs[19,30,31], the proposed architecture has higher frequency. Area in Ref. [19,30] are larger than the proposed architecture. The architecture in Refs[19, 30,31] has 32 kB, 171 kB, 16 kB data memory respectively, however CD-DAP based on 4 clusters has 256 kB data memory. 4096, 3072, 80 PEs are integrated in Ref. [19,30,31] respectively. The proposed architecture has 64 PEs, and can be scaled easily, meanwhile compared with Ref. [19,30,31], the PE is more complicated and functional. 8 bit, 16 bit, 32 bit data width can be supported simultaneously in this work, however only 10 bit data width is supported in Ref. [19], 8 bit data width is supported in Ref. [31] and 11 bit data width is supported in Ref. [30].

Table 4 Performance and resource usage

		Ref. [19]	Ref. [30]	Ref. [31]	Ref. [15]	Ref. [29]	The proposed
FPGA	Develop board	-	-	-	1 Xeon CPU + 4 Xilinx Virtex-5	Xilinx Virtex-6	Xilinx ZC706
	Clock	-	-	-	150 MHz	150 MHz	100 MHz
	LUT	-	-	-	-	134 520	232 252 (4 clusters)
	FF	-	-	-	-	148 323	120 184 (4 clusters)
	36K-BRAM	-	-	-	-	139	-
	Power	-	-	-	-	5.056 W	-
ASIC	Technology	180 nm	130 nm	130 nm	-	-	130 nm
	Clock	50 MHz	80 MHz	20 MHz	-	-	100 MHz
	Area(mm^2)	82.32	113	7.75	-	-	26.58
	# of PE	4096	3072	80	-	-	64
	Data memory	32 kB	171 kB	16 kB	-	-	256 kB
	Data width	10 bit	11 bit	8 bit	-	-	8 bit, 16 bit, 32 bit

5 Conclusions

A clustered data-driven array processor for computer vision is proposed. To reduce data transfer delay, a double buffer dataflow driven interface is designed. For improving data parallel computation, 8 bits, 16 bits, 32 bits subtext parallel computation has been supported. For accelerating computer vision applications further, a four-way shared pipelining transcendental function accelerator based on Y-intercept adjusted piecewise linear segment algorithm is designed. Simultaneously, a distributed shared memory structure based on unified addressing is also employed. Through employing fast switching unit realizing data transfer in cluster and NoC using for data communication between clusters, data accessing delay is reduced dramatically.

Sobel, Canny, Horris, Fast, Gaussian pyramid, and Histogram algorithms are implemented on proposed architecture. The computing time is statistic. Simultaneously, CD-DAP based on 4 clusters has been implemented on Xilinx ZC 706 develop board. The same circuitry has been synthesized using SMIC 130 nm COMS technology. The circuitry can be run at 100 MHz. Area is 26.58 mm^2.

References

[1] Ratha N K, Jain A K. Computer vision algorithms on reconfigurable logic arrays [J]. *IEEE Transactions on Parallel and Distributed Systems*, 1999, 10(1):29-43

[2] Backes L, Rico A, Franke B. Experiences in speeding up computer vision applications on mobile computing platforms[C]//International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation, Samos, Greece, 2015:1-8

[3] Elliott G A, Yang K, Anderson J H. Supporting real-time computer vision workloads using OpenVX on multicore + GPU platforms [C] // International Conference on Real Time and Networks Systems, San Antonio, USA, 2015: 77-86

[4] Kokkinos I . UberNet: training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory [C] // 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, USA, 2017: 1063-1069

[5] Chakrabarti A, Xiong Y, Gortler S J, et al. Low-level vision by consensus in a spatial hierarchy of regions[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, USA, 2015: 4009-4017

[6] Wallace A M, Michaelson G J, Mcandrew P, et al. Dynamic control and prototyping of parallel algorithms for intermediate- and high-level vision [J]. *Computer*, 1992, 25(2):43-53

[7] Maggipinto M, Terzi M, Masiero C, et al. A computer vision-inspired deep learning architecture for virtual metrology modeling with 2-dimensional data[J]. *IEEE Transactions on Semiconductor Manufacturing*, 2018, 31(3): 376-384

[8] Wang D, Foran D J, Qi X, et al. HetroCV: auto-tuning framework and runtime for image processing and computer vision applications on heterogeneous platform[C] // International Conference on Parallel Processing Workshops, Beijing, China, 2015:119-128

[9] Puglia L, Vigliar M, Raiconi G. Real-time low-power FPGA architecture for stereo vision[J]. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2017, 64 (11): 1307-1311

[10] Kotselidis C, Clarkson J, Rodchenko A, et al. Heterogeneous managed runtime systems: a computer vision case study[C] // The 13th ACM Sigplan/sigops International Conference on Virtual Execution Environments, Xi'an, China, 2017:74-82

[11] Fung J, Mann S. Using graphics devices in reverse: GPU-based image processing and computer vision[C] // IEEE International Conference on Multimedia and Expo, Hannover, Germany, 2008:9-12

[12] Lee Y, Jang C, Kim H. Accelerating a computer vision algorithm on a mobile SoC using CPU-GPU co-processing: a case study on face detection[C] // International Conference on Mobile Software Engineering and Systems, Austin, USA, 2016:70-76

[13] Ttofis C, Theocharides T. High-quality real-time hardware stereo matching based on guided image filtering[C] // Design, Automation and Test in Europe Conference and Exhibition, Dresden, Germany, 2014:1-6

[14] Chadjiminas I, Kyrkou C, Theocharides T, et al. In-field vulnerability analysis of hardware-accelerated computer vision applications [C] // International Conference on Field Programmable Logic and Applications, London, UK, 2015:1-4

[15] Choi J, Rutenbar R A. Configurable and scalable belief propagation accelerator for computer vision[C] // International Conference on Field Programmable Logic and Applications, Lausanne, Switzerland, 2016:1-4

[16] Guo Z, Han J, Li T. Implementing OpenVX on a polymorphous array processor[C] // IEEE International Conference on Communication Technology, Hangzhou, China, 2015:598-601

[17] Tagliavini G, Haugou G, Marongiu A, et al. ADRENALINE: an OpenVX environment to optimize embedded vision applications on many-core accelerators[C] // IEEE International Symposium on Embedded Multicore/Many-Core Systems-On-Chip, Turin, Italy, 2015:289-296

[18] Schwambach V, Cleyetmerle S, Issard A, et al. Estimating the potential speedup of computer vision applications on embedded multiprocessors [J]. *Computer Science*, 2016, 2007(1):1-14

[19] Shi C, Yang J, Han Y, et al. 7.3 A 1 000 fps vision chip based on a dynamically reconfigurable hybrid architecture comprising a PE array and self-organizing map neural network[C] // Solid-State Circuits Conference Di-

gest of Technical Papers, San Francisco, USA, 2014: 128-129

[20] Wang J, Wang H, Wu S, et al. Design and implementation of real-time Sobel edge detection on FPGA for mobile device applications[C] // ACM International Workshop, Hangzhou, China, 2015:9-14

[21] Ogawa K, Ito Y, Nakano K. Efficient canny edge detection using a GPU[C] // 1st International Conference on Networking and Computing, Higashi-Hiroshima, Japan, 2011:279-280

[22] Hsiao P Y, Lu C L, Fu L C. Multilayered image processing for multiscale Harris corner detection in digital realization[J]. *IEEE Transactions on Industrial Electronics*, 2010, 57(5):1799-1805

[23] Amara A B, Pissaloux E, Atri M. Sobel edge detection system design and integration on an FPGA based HD video streaming architecture[C] // Design and Test Symposium, Hammamet, Tunisia, 2017:160-164

[24] Sangeetha D, Deepa P. An efficient hardware implementation of Canny edge detection algorithm[C] // International Conference on Vlsi Design and 2016 International Conference on Embedded Systems, Kolkata, India, 2016:457-462

[25] Aydogdu M F, Demirci M F, Kasnakoglu C. Pipelining Harris corner detection with a tiny FPGA for a mobile robot[C] //IEEE International Conference on Robotics and Biomimetics, Shenzhen, China, 2013:2177-2184

[26] Dore A. Performance analysis of Sobel edge filter on heterogeneous system using opencl[J]. *International Journal of Research in Engineering and Technology*, 2014, 15(3): 53-57

[27] Huang Y, Bai Y, Li R, et al. Research of Canny edge detection algorithm on embedded CPU and GPU heterogeneous systems[C] // International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, Changsha, China, 2016:647-651

[28] Luo S, Zhang J. Accelerating Harris algorithm with GPU for corner detection[C] // 17th International Conference on Image and Graphics, Qingdao, China, 2013:149-153

[29] Nieto A, Vilariño D, Brea V M. PRECISION: a reconfigurable SIMD/MIMD coprocessor for computer vision systems-on-chip[J]. *IEEE Transactions on Computers*, 2016, 65(8):2548-2561

[30] Millet L, Chevobbe S, Andriamisaina C, et al. A 5500-frames/s 85-GOPS/W 3-D stacked BSI vision chip based on parallel in-focal-plane acquisition and processing[J]. *IEEE Journal of Solid-State Circuits*, 2019, 54(4): 1096-1105

[31] Schmitz J A, Gharzai M K, Balkır S, et al. A 1000 frames/s vision chip using scalable pixel neighborhood-level parallel processing[J]. *IEEE Journal of Solid-State Circuits*, 2017, 52(2): 556-568

Shan Rui, born in 1986. She received her Ph. D degree from Xidian University in 2018. She also received her M. S. degree from Xi'an University of Posts and Telecommunications in 2011. Her is an associate professor at Xi'an University of Posts & Telecommunications. Her research focuses on integrated circuit design.