A Heterogeneous Parallel Processor for High-Speed Vision Chip

Jie Yang, Yongxing Yang, Zhe Chen, Liyuan Liu, *Member, IEEE*, Jian Liu, and Nanjian Wu, *Member, IEEE*

***Abstract*— This paper proposes a heterogeneous parallel processor for high-speed vision chip. It contains four levels of processors with different parallelisms and complexities: processing element (PE) array processor, patch processing unit (PPU) array processor, self-organizing map (SOM) neural network processor, and dual-core microprocessor unit (MPU). The fine-grained PE array processor, middle-grained PPU array processor, and SOM neural network processor carry out image processing in pixel-parallel, patch-parallel, and distributed-parallel fashions, respectively. The MPU controls the overall system and executes some serial algorithms. The processor can improve the total system performance from low-level to high-level image processing significantly. A prototype is implemented with 64 × 64 PE array, 8 × 8 PPU array, 16 × 24 SOM network, and a dual-core MPU. The proposed heterogeneous parallel processor introduces a new degree of parallelism, namely, patch parallel, which is for parallel local-feature extraction and feature detection. It can flexibly perform the state-of-the-art computer vision as well as various image processing algorithms at high speed. Various complicated applications, including feature extraction, face detection, and high-speed tracking, are demonstrated.**

***Index Terms*— Computer vision, face detection, heterogeneous, high speed, object tracking, parallel processing, single-instruction-multiple-data (SIMD), vision chip.**

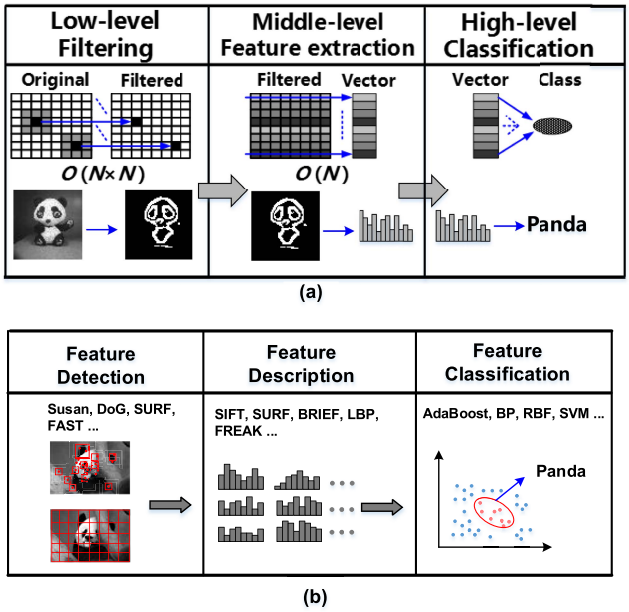

Fig. 1. (a) Traditional image processing flow. (b) Modern computer vision processing flow.

I. Introduction

HIGH-SPEED image processing and recognition can be applied to many fields, such as industrial assembly line control, defect detection, robot vision, and human-computer interaction [1], [2]. Various methods can be used to accomplish image processing tasks. Traditional vision systems cannot achieve high processing rate due to the serial image transmission and serial image processing bottlenecks. For example, SIFT algorithm costs more than half a minute to process a 1024 × 768 image on an embedded ARM A8 processor [3]. GPU is a highly parallel architecture that has been popular in vision processing in recent years [4], but the cost of the GPU system is high, and it requires complex supporting environments. More importantly, CPU and GPU usually consume tens to hundreds of watts power, which rule them out for many low-power embedded applications. Vision chip concept was first introduced in 1990s [5], [6]. It integrates high-speed image sensor and 2D array of single-instruction-multiple-data (SIMD) pixel-parallel processing elements (PEs) on a single chip. It not only increases the bandwidth of image transmission and the speed of image processing but also brings low-power, low-cost, and compact system features. The vision chips with the 2D array of pixel-parallel PEs can perform simple image processing very efficiently and make high-speed image processing possible. The reported vision chips usually have processing rate from hundreds to thousands of frames per second [7]–[22]. However, the high performance of PE array processors is obtained by trading off the processor flexibility. It is, therefore, difficult to carry out complex algorithms. As a result, early vision chips are usually application specific and they only perform certain image processing operations [7], [8], [10], [12], [14]. Later, to increase the processor flexibility, programmable vision chips were reported [9], [15]–[19], [21], [22]. Based on the concept that image processing can be classified into low level, middle level, and high level [19], [23], we first proposed the SIMD real-time vision chip (SRVC) architecture that includes both 2D PE array processor and 1D row-parallel processors (RPs) [15]. The chip can finish low-, middle-, and simple high-level image processing algorithms. The SRVC architecture was further developed in work [19]. It integrates PE array processor, row-parallel row processor, and a nonparallel microprocessor unit (MPU) to manage low-, middle-, and high-level image processing, respectively. To address the high-level processing

Manuscript received March 30, 2016; revised July 30, 2016 and September 19, 2016; accepted October 10, 2016. Date of publication October 19, 2016; date of current version March 5, 2018. This paper was recommended by Associate Editor H.-J. Lee.

The authors are with the Institute of Semiconductors, Chinese Academy of Sciences, Beijing 100083, China (e-mail: nanjian@red.semi.ac.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2016.2618753

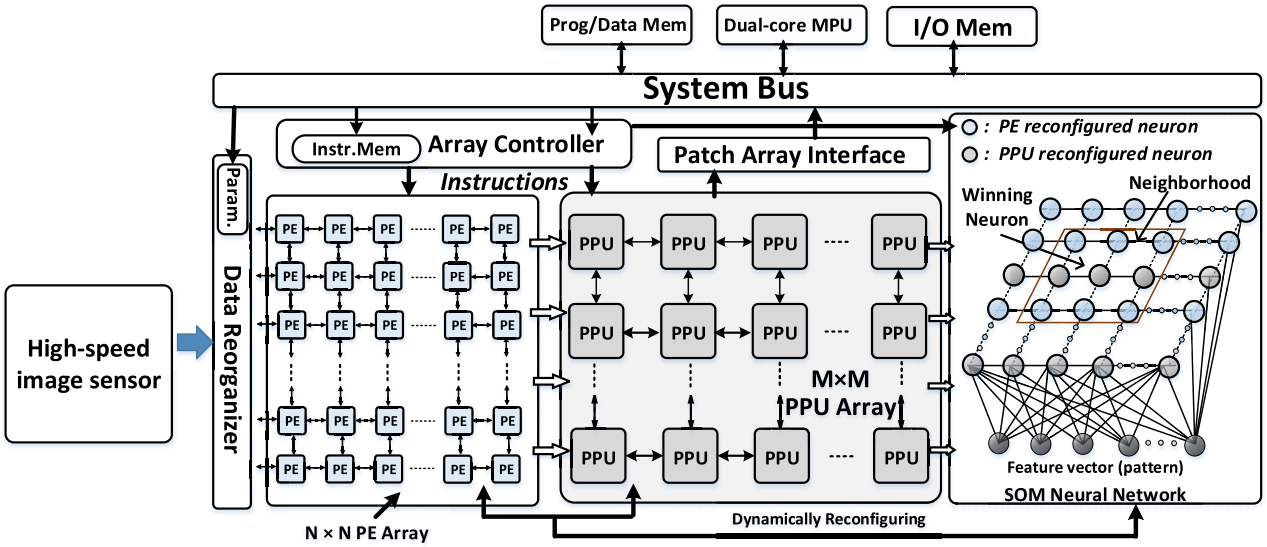

Fig. 2. Proposed heterogeneous parallel processor architecture.

speed bottleneck issue of the nonparallel MPU, we proposed a vision chip with self-organizing map (SOM) neural network to speed up feature classification [21]. The SOM neuron network is reconfigured from PE array processor, similar to work, such as [24] and [25]. Although these vision chips achieve good performance in some vision applications, their design concepts were still within the scope of traditional image processing flow. Fig. 1(a) shows the traditional image processing flow, and it is composed of three parts, namely, filtering, holistic feature extraction, and feature classification. Filtering is used for contour, edge extraction, or morphological operations. The holistic feature is a single global vector extracted on the whole image to represent the object. However, holistic features are sensitive to illumination, angle, and background changes, which affect the classification accuracy greatly. Fig. 1(b) shows the modern computer vision processing flow; it first extracts the area of interests or divides the image into multiple small patches, and then in each area or patch, a local feature is extracted. At last, all the extracted features are used in classification. Compared with the traditional processing flow, it greatly increases the accuracy of detection and classification and is widely used in vision applications [26]. But, the vision chips developed so far cannot carry out this flow in an efficient way. First, the RP was designed specifically for holistic feature extraction, so it cannot extract local features efficiently. Second, the vision chips lack the consideration of the nature of image signal processing and computer vision algorithms, because most algorithms are carried out on image patches rather than the whole image, and different image patches can be processed in parallel. Current vision chips all fail to utilize this degree of parallelism to further improve system performance.

This paper proposes a novel heterogeneous parallel vision chip processor. It consists of three von Neumann-type processors: PE array processor, patch processing unit (PPU) array processor and dual-core MPU, and one non-von Neumann-type processor, namely, the SOM neural processor. The PE array and PPU array processors can process an image in pixel-parallel and patch-parallel fashions, respectively. The PE array processor can pipeline the two phases of data transfer and image processing. Pixel-parallel operations can be performed in PE array with high efficiency. The PPU array can carry out feature extraction in patch parallel while it also performs binary classification in a distributed-parallel fashion. The non-von Neumann-type SOM neuron network carries out the multiclass classification in a vector-parallel way. PE and PPU arrays can be dynamically reconfigured into the SOM neuron network so that the resource usage of the architecture is greatly reduced. Complicated image processing and computer vision algorithms are realized in this architecture in high performance and a flexible fashion. The proposed heterogeneous processor architecture introduces a new degree of parallelism, namely, patch parallel, which is designed for parallel local-feature extraction and feature detection. The ability of local-feature processing enables the architecture to work for modern computer vision algorithms while the patch-parallel processing improves the processing performance.

This paper is organized as follows. The vision chip architecture is presented in Section II, and the module implementation is described in Section III. In Section IV, the implementation and experimental results are shown. We demonstrate how weak classifiers are implemented in PPU and the performance of AdaBoost classifier in our architecture in Section V. Finally, the discussions and conclusions are provided in Section VI.

II. Architecture

A. Hierarchical Heterogeneous System Architecture

Fig. 2 shows the proposed processor architecture. It mainly consists of three von Neumann-type processors and a bioinspired non-von Neumann-type SOM neuron network processor. The von Neumann-type processors consist of an $N \times N$ pixel-parallel PE array processor, an $M \times M$ patch-parallel PPU array processor, and a dual-core MPU. The PE array and PPU array perform the low-level pixel-parallel and middle-level patch-parallel algorithms, respectively. The MPU manages the whole system while finishing some serial algorithms.

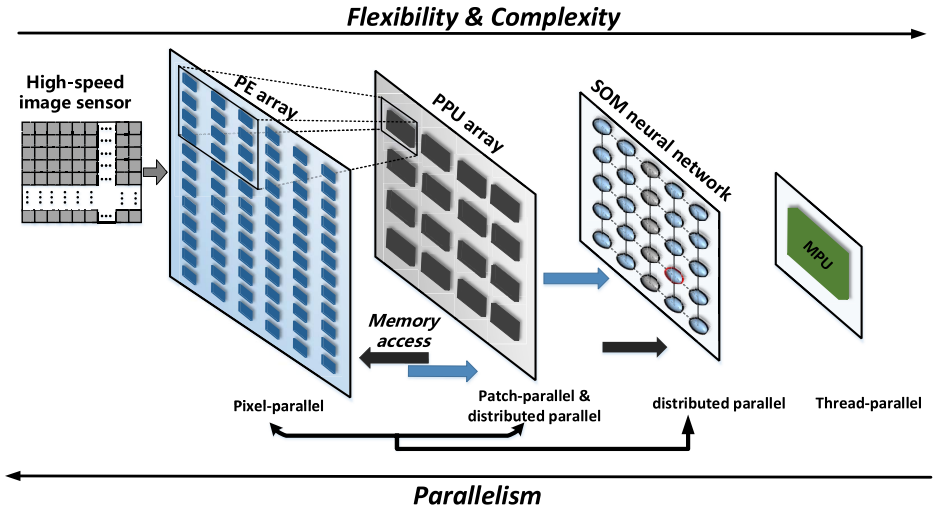

Fig. 3. Hierarchical view of the proposed heterogeneous architecture. The four kinds of processors have different granularity and complexity.

The non-von Neumann-type SOM neuron network is dynamically reconfigured from the PE array and PPU array. The PPU array and the SOM neuron network can carry out feature classification in distributed-parallel fashion.

Fig. 3 gives the hierarchical view of the proposed architecture. The four different kinds of processors have different granularities and complexities. The PE array processor performs pixel-parallel image processing algorithms in SIMD scheme. Each PE is a simple 1-b processor. However, the massive parallel PE array exhibits enormous computation power in prerequisite algorithms for many recognition and detection applications, such as 2D filters, background reduction, feature from accelerated segment test (FAST) [27], and local binary pattern (LBP) [28]. With the help of the massively parallel PE array processor, the processing speed of these procedures can be largely improved. The algorithm details are explained in Section IV.

The PPU is more flexible and powerful than PE. It is capable of executing complex algorithms, such as orientation assignment, local-feature extraction, and block matching. Each PPU corresponds to a PE subarray with a mesh structure, as shown in Fig. 3 (dashed lines). Currently, in our implementation, the mapping of PPUs to PEs is fixed, and corresponds to an 8×8 grid of PEs for each PPU. This mapping relationship significantly differs from that between the PE array and the row-processor array in the reported architectures [15], [19], [21] where one row of PE processor corresponds to one row processor. It is more suitable for modern image signal processing and computer vision algorithms than the traditional one row of PE to one row-processor mapping scheme. Nowadays, local-feature representations have become dominant among various ways to describe image feature [29]–[34]. They are widely applied in trackers and classifiers to accomplish tracking, detection, and recognition applications. The extraction of the local feature is mainly based on processing image patches and histogram statistics. However, an image consists of significant numbers of patches; thus, the whole procedure can be computationally expensive and time-consuming.

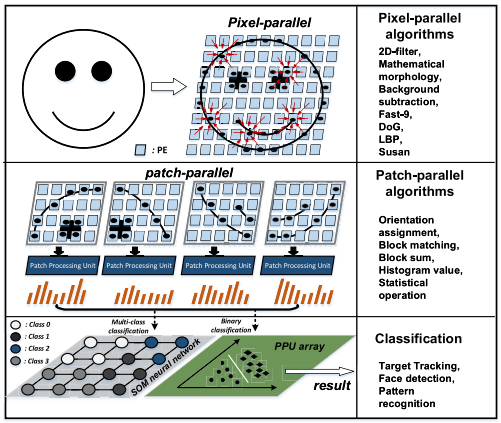

Fig. 4. Typical processing flow of the proposed architecture and representative algorithms for each step.

The mapping relationship between PE array and PPU array is designed to facilitate the local-feature extraction procedure. The PPU can access the data memory of a PE subarray that stores an image patch. Thus, it can directly perform local-feature extraction immediately after the PE array processing. It removes the bottleneck of image data access between the row processor and PE memory in one row processor to a row of PEs mapping relationship. In addition, the SIMD PPU array can carry out classification algorithms, such as AdaBoost in distributed-parallel fashion.

The SOM neuron network is adopted in this architecture to speed up multiclass classification in vector parallel. The SOM neuron plane is partitioned into several nonoverlapping regions, and each of these regions corresponds to a feature class. In the SOM neural network, each neuron can store a K-dimension reference vector (RV). Online training and updating of the RV can be completed with the learning vector quantization method. A typical processing flow of the proposed hierarchical heterogeneous architecture is shown in Fig. 4.

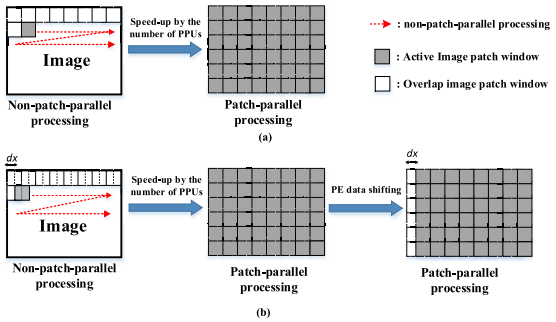

Fig. 5. (a) Serial and patch-parallel processing methods. Gray boxes indicate active patch. (b) Patch overlapping processing in serial and patch-parallel processing.

B. Architecture Feature

1) Patch-Parallel Processing: The image processing or computer vision system usually selects and processes the minimum image data unit sequentially. The image data unit within a rectangle structure is called as an image patch. The unique local features can be extracted from the image patch. These features are critical to the following image processing. But, it is difficult to process image patches in parallel by the previously reported vision chips because their middle-level processor, namely, row processor, can directly access only one row rather than one patch of image data. The proposed architecture integrates PPU array processor for patch-parallel processing. Each PPU in the processor corresponds to a PE subarray with a mesh structure. The PPU can access the local image patch data in the PE subarray directly. All PPUs operate independently in an SIMD fashion. Thus, the high-speed patch-parallel processing is achieved. Fig. 5(a) shows the nonpatch-parallel and patch-parallel processing methods. In nonpatch-parallel processing, image patches are serially processed one by one so that the total processing time is proportional to the number of patches. In patch-parallel processing, image patches are processed concurrently. The algorithm can be accomplished by executing the procedure once. Therefore, the speedup factor is the number of PPUs.

In addition, adjacent patches usually have certain degree of overlapping to ensure that all possible features or locations are included. Fig. 5(b) shows a schematic of two adjacent image patches overlapping with each other with *dx* pixels. In our proposed architecture, this overlapping mechanism can be implemented efficiently with the help of the PE array. Instead of shifting the patch window, we shift the data stored in the PE array with *dx* pixels. With only a few operations, we can shift data in all patches with an offset of *dx* simultaneously. Then, we execute the algorithm again to extract local features in the new patches. In Section IV, we demonstrate the efficiency of this mechanism in object tracking applications.

2) Distributed-Parallel Classification: Classification is one of the most important and commonly encountered problems in image processing and computer vision. A performance- and accuracy-oriented classifier implementations are critical to a high-speed vision system. The proposed architecture performs fast classification in a distributed-parallel way to achieve this goal. Equations (1) and (2) show the two classifiers that can be used in our architecture: AdaBoost classifier and SOM neuron network

$$\text{class} = \text{sign}\left(\sum_{0}^{T-1}(h_i \times a_i) - t\right) \tag{1}$$

$$\text{Dist} = ||FV - RV_{i,j}||. \tag{2}$$

The AdaBoost classifier is a binary classifier [35]. It is comprised of T weak classifiers $h_i(x)$ $(i = 0, 1, 2T-1)$ with a corresponding weight a_i, as shown in (1). The input and output of the weak classifier are obtained features and binary logic results (0 or 1), respectively. Equation (2) gives the distance between the feature vector (FV) and RV in the SOM neuron network. FV is the obtained FV and $RV_{i,j}$ is an RV, where i, j represents the neuron coordinates. The location of the neuron with the minimum distance represents the recognized pattern class of the FV.

Typically, the number of weak classifiers and RVs would be hundreds, thus the computing of the above equations could be very slow if they are carried out in serial. However, the above classifier can be computed in distributed parallel in the architecture with PPU and SOM neuron, respectively. We rewrote (1) into

$$\text{class} = \text{sign}\left(\sum_{j=0}^{m} partial\_sum_j - t\right) \tag{3}$$

$$\text{partial\_sum}_j = \sum_{i=j\times k}^{j\times k+(k-1)} h_j(x) \times a_i \tag{4}$$

where m indicates the number of PPUs in the architecture and k represents the number of weak classifiers that are computed in each PPU. It is obvious that the m *partial_sum* can be calculated by the PPU array simultaneously. MPU performs the calculation of (3). It obtains the *partial_sum* computed in each PPU and carries out summation and it gives a detection result at last. As for (2), we store the RV $RV_{i,j}$ in each neuron and broadcast the FV to them, and then all neurons can compute (2) simultaneously.

Other hardware classifier usually consumes considerable hardware resources. However, in the proposed architecture, the AdaBoost classifier is implemented without costing any additional hardware resources, whereas the SOM neuron network requires only few additional logic gates for reconfiguration. The AdaBoost classifier and the SOM neuron network are used in different scenarios. If accurate and fast distinction of positive and negative classes is required, we use the AdaBoost classifier. If multiple classes are being classified, we use SOM neuron network instead. The weak classifiers and the RVs are distributed and stored in all PPU memory and SOM neuron memory, respectively. During classification, all memories are accessed simultaneously to eliminate bottleneck of data bandwidth between memory and PPU or SOM computation unit. In Section V, we will demonstrate how weak classifiers are implemented in PPU and the performance of AdaBoost classifier in our architecture.

3) Efficient and Flexible Pixel to PE Mapping Relationship: Great flexibility has been gained with a pixel-PE separation

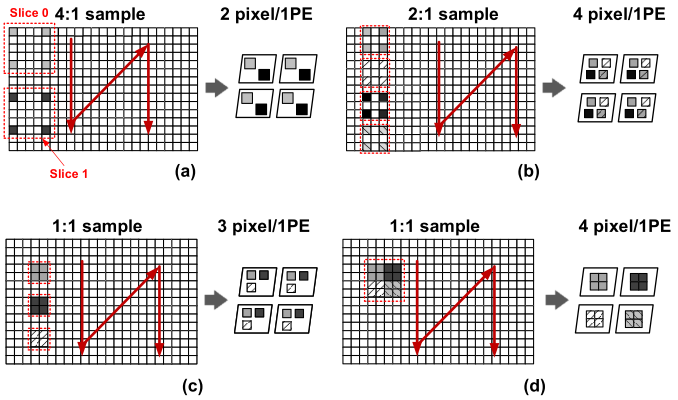

Fig. 6. Different pixel-PE mapping relationship of the proposed architecture. (a): 4:1 sample two image slices. (b): 2:1 sample four image slices. (c): 1:1 sample three image slices. (d): 1:1 sample four image slices.

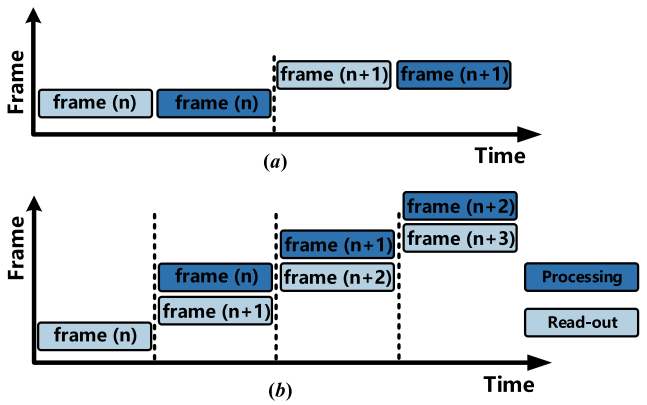

Fig. 7. Frame pipeline scheme in the proposed architecture.

scheme; however, only one image slice from the image sensor can be obtained and processed by the PE array processor in the previous vision chips [19], [21]. But, if the image slice is not the region of interests (ROI) or the ROI is distributed among multiple slices, then the processor has to wait for successive frames to get the desired image slices. This inefficiency of such pixel to PE mapping relationship deteriorates system performance. We propose an efficient and flexible mapping scheme that in each frame, the processor can sample multiple image slices with a configurable subsample interval. In Fig. 6, small squares correspond to pixels and constitute an image plane; each parallelogram corresponds to a PE in the PE array. As shown in Fig. 6(a), slice 0 and slice 1 of the image plane are 4:1 subsampled to the PE array in a single frame instead of two frames compared with previous implementations [19], [21]. Each PE stores two pixels from slice 0 and slice 1, respectively. Then, PE array can process slice 1 immediately after the processing of slice 0 is finished. Similarly, different subsample intervals and slice numbers can be applied. As shown in Fig. 6(b) and (c), 2:1 and 1:1 subsample intervals are selected, and each PE stores four and three pixels, respectively. The number of slices that can be obtained in each frame is mainly limited by the PE memory space. With the proposed method, we can scan over the image plane in a zigzag way in several frames with high efficiency [Fig. 6 (red solid lines)]. The mapping relationship also benefits the architecture when a bigger image patch is required during processing. If every PE stores one pixel, PPU will correspond to an 8×8 image patch. If four pixels are stored in each PE, as shown in Fig. 6(d), PPU will correspond to 16×16 patch.

4) Frame Pipeline Scheme: The exposure and readout of CMOS image sensor (CIS) need to consume time. As shown in Fig. 7(a), if the image processing has to wait for the CIS readout to finish, the idle time for the three processors (PE, PPU, and MPU) might be long. The existence of idle time decreases the vision chip performance. However, in previous vision chip designs, the PE array is used either for image acquisition [9], [10], [15], [18] or as memory buffer [19] for the image sensor. Thus, image acquisition and processing cannot be carried out in parallel. Similar to work [24], the proposed architecture adopts a frame pipeline scheme to hide the CIS readout latency, as shown in Fig. 7(b). When the image sensor is reading out a new frame of image data, the processors are processing the previous frame of image data. If the algorithm execution time is equal to the CIS readout latency, the idle time of the processor can be zero. This pipeline scheme is realized by introducing a simple FIFO in each PE. This is described in Section III-A.

5) Enhance SOM Neuron Network: An SOM neural network can be dynamically reconfigured from the PE array processor and the PPU array processor. Because the 1D RP array is not compatible with 2D-structured SOM neuron network, the RP array processor in previous work could not be reconfigured into the SOM neurons so that the computation power of the middle-level processor was wasted [21]. However, in the proposed architecture, the PPU array intrinsically has the 2D mesh structure and is suitable for SOM neuron network implementation. Thus, the proposed architecture can reconfigure both the PE array and the PPU array into SOM neural network. As shown in Fig. 8, a dedicated condition generator and 4×4 snake chained PEs constitute a neuron with a 16-b arithmetic logic unit (ALU). With the same condition generator, the two 16-b ALUs in PPU can form two SOM neurons. The reconfiguration of PPU array further increases the computation power of the SOM neuron network. The reconfiguration costs only multiplexers for PE topological connection switch and condition generators for condition operation. However, compared with a standalone SOM neuron network implementation, the reconfiguration cost is negligible (5%) [21]. PEs in [24] and [25] can operate the function of a neuron, but the coarse grained design makes it difficult to carry out pixel-parallel operations.

III. Module Design of the Vision Chip Processor

A. PE Circuit

Fig. 9 shows the proposed PE circuit. It consists of a main memory, an FIFO, a 1-b ALU, and some multiplexers. The main memory stores temporary values and results during the image processing stage. The FIFO stores the incoming pixel data of a new frame. The PE can process the previous frame on the main memory while storing the data of the incoming frame on the FIFO without interfering with each other. After the image processing of the previous frame and the CIS procedure of the current frame are finished, the data stored in the FIFO are automatically moved to the main memory

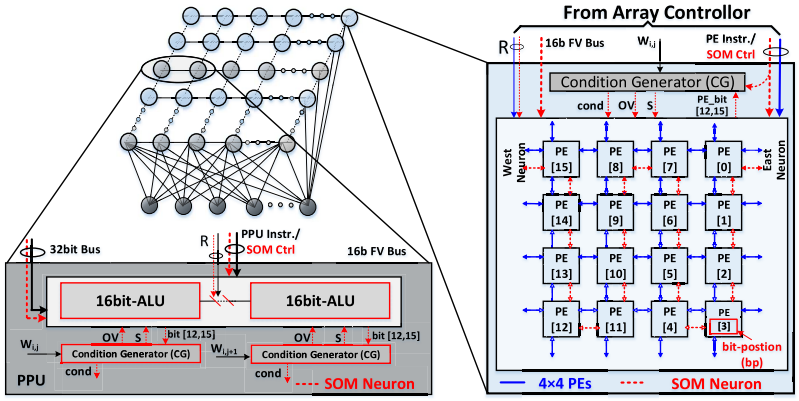

Fig. 8. Reconfiguration of PEs and PPU to SOM neuron. Every 4×4 PEs can constitute an SOM neuron while each PPU can split into two SOM neurons.

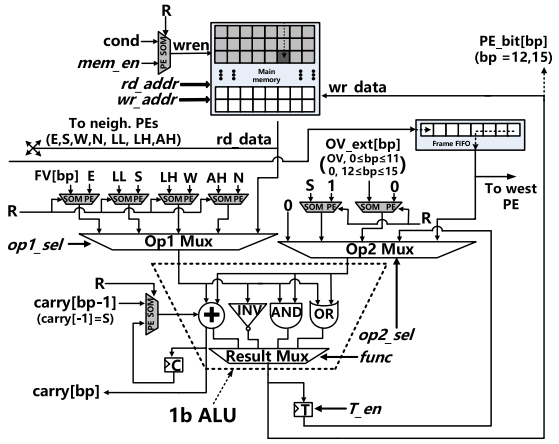

Fig. 9. PE circuit schematic. It consists of a 1-b ALU, one SRAM, and one frame FIFO. PE is connected with nearby neighbors.

by a hardware interrupt, and this procedure only takes a few instruction cycles. The ALU performs the operations of full adder, inverter, AND gate, and OR gate. The reconfiguration multiplexers (shaded in Fig. 9) can switch the topological connections between neighboring PEs for different operation modes. When reconfiguration multiplexers are selected ($R = 1$, in Fig. 9), PEs are in SOM neuron mode, and 1-b ALUs of all PEs in an SOM neuron are connected to form a 16-b ripple adder. In PE array processor mode ($R = 0$), PEs are mesh-connected, and the multiplexer *op1_mux* selects the first ALU operand from its own main memory, or from the main memory of its east, south, west, and north neighboring PEs (E, S, W, and N entries in Fig. 9). The register C stores the carryout of the ALU in the current cycle and acts as carry-in of the ALU for the successive cycles. Multiplexer *op2_mux* selects operands from constant zero, constant one, temporary register, and the output of FIFO as the second operand for the ALU. By concatenating the 1-b ALU operations, multiple-bit operations can be finished [19]. Table I lists some common functions that PE can finish where a, b, and c are 8-b variables in the PE memory. $\partial f/\partial x$ and $\partial^2 f/\partial^2 x$ are the first and the second order of image gradients, respectively. In SOM neuron mode, 16 such PEs are connected in a snake style to form an SOM neuron.

TABLE I
SOME COMMON FUNCTION OF PE ARRAY

Function	Cycles	Function	Cycles
$c = a + b$	17	$c = a - b$	18
$c = max(a,b)$	52	$c = min(a,b)$	52
$\frac{\partial f}{\partial x}$ *or* $\frac{\partial f}{\partial y}$	18	$\frac{\partial^2 f}{\partial^2 x}$ *or* $\frac{\partial^2 f}{\partial^2 y}$	52
$c = a < b \ ? \ a : 0$	26	$c = a > b \ ? \ a : 0$	27
$c = \|\|a\|\|$	72	*non-maximum suppression*	167

B. PPU Circuit

Fig. 10 shows the schematic of the proposed PPU circuit. It mainly consists of a PE input buffer (PEIB), general-purpose registers (GPRs), two 16-b ALUs, and one data memory. The PEIB serves as the interface between PE and PPU. Since PE and PPU both have their independent data memory, once the date exchange is finished, they can work simultaneously. The two ALUs can perform two 16-b or one 32-b operations by controlling the carry signal. In image processing, 32-b precision is seldom required; thus mostly, we use the ALUs in two-way SIMD mode to increase PPU performance. In this mode, the two 16-b ALUs can perform comparison, shift, and max/min extraction for nonlinear operations. We also implement an identical condition generator for the two ALUs so that they are fully compatible with the 16 PEs reconfigured SOM neuron. A bus interface is used to support the data exchange between the MPU and the PPU. Write procedure of the memory is controlled by the condition execution module that supports eight kinds of conditions based on the status of the condition registers.

Two important operations in computer vision processing are orientation assignment and histogram calculation. Usually, orientation assignment involves costly division, inverse tangent, and rounding calculations. Fortunately, the required calculation precision of orientation assignment in most algorithms is not high. Like SIFT, SURF, and histogram of oriented gradients (HoG), eight orientations are often enough for acceptable processing results. This allows us to determine the

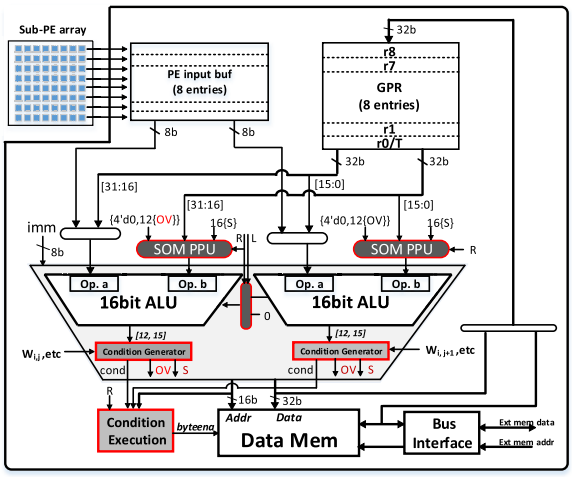

Fig. 10. Circuit schematic of the PPU.

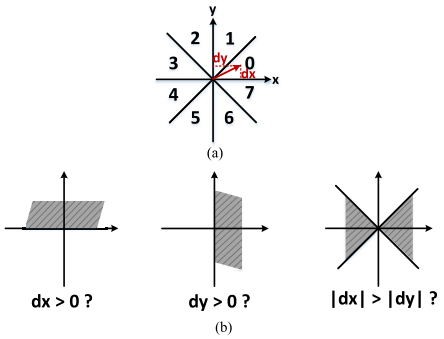

Fig. 11. Gradient orientation with boundary comparison. (a) Simple orientation assignment. (b) Three boundary conditions for orientation assignment.

orientation by simply comparing region boundaries. As shown in Fig. 11(a), the gradient orientation of a pixel can be assigned to one of the eight equally spaced angular regions based on its gradients on the x- and y-directions. Given a gradients pair (dx, dy), we can test three boundary conditions, as indicated in Fig. 11(b), and the logical results are then combined to generate a corresponding bin number between zero and seven. The two 16-b ALUs can simultaneously work on two gradient pairs to reduce processing time. More accurate boundary tests can also be implemented, for example, $\tan 20^\circ \approx 0.37$; we can test boundary $(16 \times dy + 8 \times dy) > (8 \times dx + 1 \times dx)$ where the multiplication can be performed by shift operations. For histogram calculation, we first calculate the sum of a base address and the variable as the new address. Then, the value stored in this address is loaded to the GPR and added by one. At last, the value is stored back to the memory. The above-mentioned procedure repeats until all variables are calculated.

C. SOM Neuron Circuit

There are two types of SOM neuron circuits in the proposed architecture: PE-based SOM neuron and PPU-based SOM neuron. The PE-based SOM neuron constitutes 16 snake-style-connected PEs. In the 16 PEs, the 1-b ALU carryout of the PE at the ith bit position is directly connected to the carry-in signal of the 1-b ALU at the $(i + 1)$th bit position. The LL, LH, and AH ports connections of PE circuits at different bit positions are listed in Table II. The condition generator circuits are shown in Fig. 12. Although the PPU and PE reconfigured SOM neuron circuits are different, their underlying functions are the same: computing absolute difference between the RV and the FV; clamping operation to avoid overflow of FV or RV. The two kinds of SOM neurons can finish the key steps for SOM neuron network training and recognition with the same instructions. More details about the SOM neuron function and circuit can be referred in our earlier work [21].

TABLE II
CONNECTION BETWEEN NEIGHBORING PE IN THE SOM MODE

bit-position (bp)	0	15	other bp
LL (Logic Low)	East neuron PE[15]	This neuron PE[14]	This neuron PE[bp-1]
LH (Logic High)	This neuron PE[1]	West neuron PE[0]	This neuron PE[bp+1]
AH (Arith. High)	This neuron PE[1]	This neuron PE[15]	This neuron PE[bp+1]

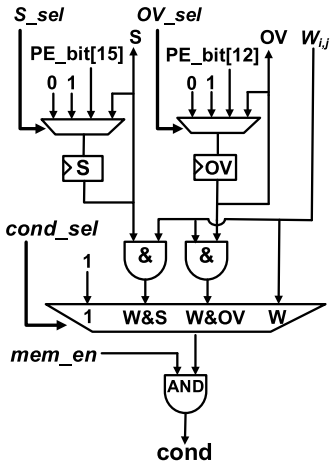

Fig. 12. Circuit schematic of the condition generator. The italic signals are from the array controller.

IV. IMPLEMENTATION AND MEASUREMENT RESULTS

A. Architecture Implementation

To develop and verify the proposed architecture in a more efficient and less expensive way, we implemented a test system with a high-speed image sensor [36] and a high-performance FPGA [37]. The resolution of the image sensor is 256 × 256. The standard frame rate for this image sensor is 1000 frames/s. 64×64 PE array, 8×8 PPU array, and a dual-core 32-b RISC MPU are realized on the high-performance FPGA. The PE and PPU have 96 and 512 b of memory, respectively. The architecture can run at 50 MHz on the FPGA. The test system is shown in Fig. 13. The high-speed image sensor and the FPGA are connected with each other through a PCB board. The on-FPGA Ethernet interface was used for communication

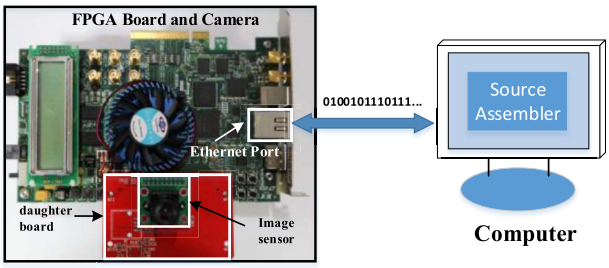

Fig. 13. Prototype system. It consists of an FPGA development board, an image sensor daughter board, and compilation environment on PC.

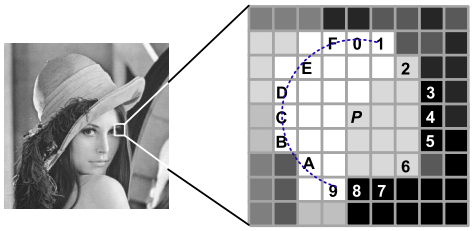

Fig. 14. FAST feature detector. The numbered pixels are used in the feature detection. The pixel ***p*** is the candidate pixel. The arc is indicated by the dashed line that passes through nine contiguous pixels, which are brighter than ***p*** by the threshold.

with a PC. Software tools have been developed for compiling and downloading instructions for the processors.

Due to the PE array compatibility with earlier works, some previously reported algorithms, such as spatial filtering, morphology, complex filtering, and M-S function-based salient edge extraction, can be directly mapped onto this paper. In the rest of this section, we focus on the architectures ability to perform more complicated algorithms.

B. Feature Detection

Feature detection is a prerequisite step for various computer vision algorithms. The architecture can perform the typical feature detection algorithms proposed in recent years [38]. FAST feature detector is one of the most popular detectors among them; according to [39], FAST-9 (FAST with $n = 9$) is the best in terms of processing speed and repeatability among various feature detectors. As shown in Fig. 14, the FAST feature detector operates on a discrete circle around a candidate point p, and the circle contains 16 pixels. ***p*** is classified as a feature pixel if there exists a contiguous arc of at least ***n*** pixels that are all brighter or darker than ***p*** by a threshold ***t***. We implemented the FAST-9 feature detector on the proposed architecture in the following steps.

1) The candidate pixel ***p*** subtracts the ith pixel on the circle with t and -t, and the sign bit b_i and d_i of the result can be obtained, respectively.
2) Exhaustive segment tests are performed on the sign bits b_i and d_i following:

$$B = \sum_{i=0}^{15} \prod_{j=0}^{8} U(b_{i,j}) \quad D = \sum_{i=0}^{15} \prod_{j=0}^{8} U(d_{i,j}) \tag{5}$$

$$U(x_{i,\ j}) = \begin{cases} x_{i+j-15}, & i+j > 15 \\ x_{i+j}, & \text{otherwise.} \end{cases} \tag{6}$$

TABLE III
PERFORMANCE COMPARISON WITH OTHER ARCHITECTURES

	This work	[3]	[40]	[41]
Algorithm	FAST	FAST SIFT HOG	FAST	FAST
Architecture	Pixel-parallel PE array	Multicore processor	Embedded processor	Other
Frequency	50MHz	168MHz	1GHz	25Mhz
Throughput (pixel/cycle)	1.2	0.03	0.04	0.77

Table III shows the comparison of feature detection performance with other architectures [3], [40], [41]. High throughput is achieved by pixel-parallel operation of the PE array. With some mathematical optimization, we can perform SIFT feature detector with PE array as well [42], [43].

C. Local Feature Extraction

Local-feature representations have made robust detection and recognition possible. They play a critical role in modern computer vision society. LBP and HoG are two heavily used local features for detection and recognition. The two local features were experimented on the proposed architecture. The basic LBP operator is shown in Fig. 15(a). It assigns a binary string to every pixel of an image by thresholding the 3×3 neighbors of each pixel with the center pixel value. The eight results surrounding binary digits are used to determine the LBP value of the center pixel. Research indicated that more than 90% of the eight binary digit patterns have equal or less than two 0/1 or 1/0 transitions [28]. These patterns are shown in the first two rows of Fig. 15(b). They are classified as uniform patterns, and the number of 1s in the eight surrounding bits is assigned as their pattern value. The remaining 10% of patterns, which are shown in the last row of Fig. 15(b), have more than two 0/1 or 1/0 transitions. They are assigned with value 9, and classified as nonuniform patterns. To get the LBP value of every PE, we first obtain the binary string b_i and then calculate the LBP value

$$f(x) = \begin{cases} 9, & \text{if } (\sum \text{XOR}_i > 2) \\ \sum_{i=0}^{7} b_i, & \text{else} \end{cases} \tag{7}$$

$$\text{XOR}_i = \begin{cases} b_i \oplus b_{i+1}, & 0 \le i \le 7 \\ b_i \oplus b_{i-7}, & i = 7. \end{cases} \tag{8}$$

The LBP operator is finished by the PE array in pixel parallel. It takes less than 800 instruction cycles. Thanks to the PE-PPU mapping relationship in our architecture, after the LBP value generation procedure is finished, each PPU can directly access a patch of the LBP image and compute the local histogram. The histogram computation of all patches can be finished with the PPU array in 1300 cycles.

As for the HoG description, we can first obtain the image gradient in the horizontal and vertical directions by using the

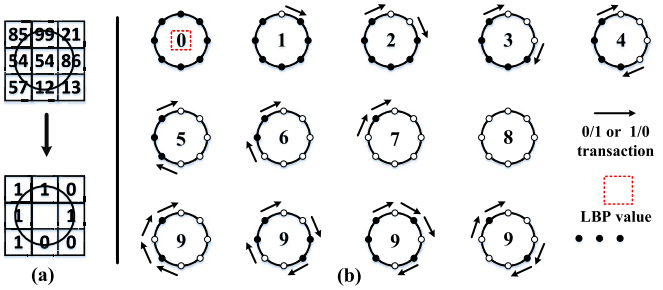

Fig. 15. Basic LBP operator and the LBP patterns. (a) Binary string is obtained by thresholding the center pixel value. (b) Dark and white dots represent zero and one in the binary string, respectively. The uniform LBP patterns have equal or less than two 0/1 or 1/0 transition (first two rows), and the nonuniform patterns have more than two transitions.

$[1\ 0\ -1]$ and $[1\ 0\ -1]^T$ derivative filters, respectively. The gradient magnitude at each pixel can then be approximated by (9) to eliminate costly square and root operation. The gradient orientation is computed with boundary test, which is described in Section III

$$\sqrt{dx^2 + dy^2} \approx |dx| + |dy|. \tag{9}$$

In each 8×8 image patch, we can calculate an orientation histogram using the gradient magnitude as weight factors. Usually, the histograms of 2×2 image patches form a block histogram and more block histograms form the final HoG feature vector. The derivative filters and gradient magnitude are performed with PE array in pixel parallel, and they consume 40 and 56 instruction cycles, respectively. The orientation and histogram calculation totally takes about 5500 cycles. The LBP operator, derivative filters, and gradient magnitude operations can be finished in work [21] with the same efficiency, because they only require PE operations. However, since each RP in [21] can access only one row of data, to compute the local histogram of LBP and HoG, the data communication overhead cause the efficiency to drop as much as 87%, and the measured clock cycles for LBP and HoG histogram calculation are 1.04k and 45.8k, respectively.

D. Object Detection and Target Tracking

We combine the local-feature representation and the distributed-parallel classification to carry out object detection and target tracking algorithms to demonstrate the ability and efficiency of the proposed architecture. The local features obtained by LBP or HoG are typically several hundred dimensions, and each dimension can be regarded as a weak classifier of AdaBoost algorithm [44]. Take face detection as an example. We used the ORL facial database [45] for training and assigned the ith weak classifier with a threshold θ_i and weight a_i. The threshold θ_i is carefully selected, so that the weak classifier has better performance in discriminating between face and nonface classes. The final classifier for face detection is given as

$$\text{sign}\left(\sum_{i=0}^{T-1} a_i * h_i(x - \theta_i) - t\right). \tag{10}$$

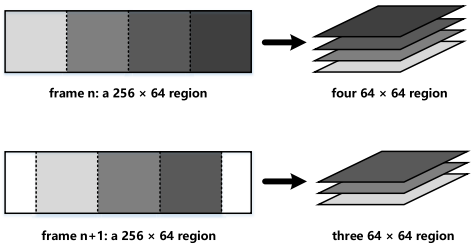

Fig. 16. Overlap the processing windows to ensure that faces in any position of the 256×64 area can be detected.

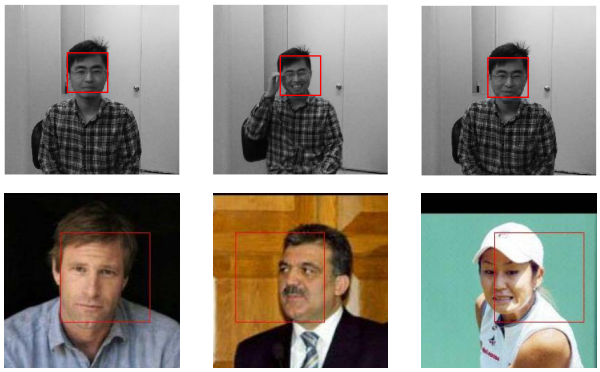

Fig. 17. Face detection results. The first row is the detection results of the image sensor and the second row is the detection results of an LFW database.

a_i is represented as a signed 16-b fixed point variable to eliminate floating point calculation while maintaining precision. Then, all $a_i * h_i(x - \theta_i)$ values were carried out in PPU in patch-parallel way. The MPU gives out the final classification result. One iteration of the face detection routine consumes about 60 μs. Cooperating with our proposed pipeline PE scheme and pixel-PE mapping relationship, the architecture can continuously execute the algorithm to scan the whole image for face detection in very high efficiency. For example, as shown in Fig. 16, in a single image readout, any 256×64 region of the image sensor can be stored in the PE array as four 64×64 regions, and it consumes only $4 \times 60 = 240$ μs to process the four regions. In the next frame, we capture the same 256×64 region and process the three 64×64 regions, which overlap with the processed four regions. It will consume $3 \times 60 = 180$ μs. Using this method, we can slide any possible regions to perform face detection for the whole 256×256 image plane. However, due to the pipelined PE scheme, the total processing time for the 256×64 region is only relevant to the image sensor readout time. Fig. 17 shows some detection results. We also feed our processor with the LFW database [46], and it achieves 91% detection rate. Actually, using the mentioned method, we can adopt or combine LBP, HoG, and other similar local representations with the distributed-parallel classification to carry out detection for other objects as long as we train the classifier with the corresponding database. For example, we can train a new classifier with the HoG feature for pedestrian detection or combine LBP and HoG for human detection.

The enhanced SOM neuron network can also be used for detection and recognition. For comparison, the face detection

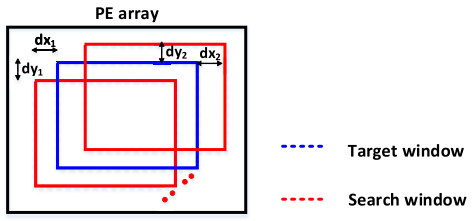

Fig. 18. Illustration of the target window and the search window in the tracking algorithm. dx and dy are the coordinate offset between the search window and the target window in horizontal and vertical directions.

Fig. 19. Experiment setup for target tracking.

and recognition algorithms described in [21] were implemented. Due to the increased network scale, the enhanced SOM neuron network achieves 6% higher detection accuracy and three additional faces can be recognized without deteriorating the accuracy and frame rate.

To demonstrate the ability of the architecture for high-speed tracking applications, a target tracking algorithm was implemented. It begins from the image acquisition of the sensor, and then the object within the initial tracking window was described with the LBP feature histogram. In the successive frame, we select several search windows that surround the tracking window of the previous frame and compute their LBP feature representations. In high-speed tracking, the difference between two successive frames will be limited to a few pixels. Thus, as shown in Fig. 18, the selected search window will mostly overlap with the target window of the previous frame, and the offsets between the two windows in horizontal and vertical directions dx and dy are limited within two or three pixels. The search window with the smallest difference compared with the target window was regarded as the new location of the target. We update the target feature every frame to adapt affine and scale changes.

The feature extraction within the search window is accomplished by cooperating with PE and PPU array. For example, if the search window has dx and dy offsets with the target window in horizontal and vertical directions, first, we shift the data in PE array with $-dx$ and $-dy$ offsets in the two directions, respectively. Then, we can obtain the search window feature by running the target window feature extraction routine again. Fig. 19 shows the experiment setup for a target tracking application. The CIS and FPGA were mounted on an actuator with two degrees of freedom. A remote control jeep model was used as the target. The control signal of the actuator is given by the MPU directly. According to the position of the calculated target position, the actuator adjusts to locating the target in the center of the view. In our experiments, we choose the target size as 56×56 and vary dx and dy from -2 to $+2$ to obtain 25 different search windows. The total processing time for the tracking algorithm is 2 ms. Tracking results in our laboratory environment are shown in Fig. 20.

V. Performance and Comparison

The proposed vision chip can execute traditional image processing as well as complex algorithms, such as feature point detection, face detection, and target tracking in high speed. The proposed patch-parallel PPU array accelerates feature extraction and matching procedures. The experimental results of various complicated algorithms demonstrate that the vision chip can achieve a high system performance.

In FAST feature detection, the architecture achieves 1.2 pixel/cycle throughput, which is the highest among the listed architectures. In face detection, the system uses LBP feature. This feature is more robust than the PPED feature, which is commonly used in vision chips. The dimension of the obtained feature is ten times of the PPED feature, and it is more robust and more representative. Thanks to the patch-parallel PPU array, the feature extraction takes only 1300 instruction cycles for all image patches. The AdaBoost classification is finished with the PPU array in a distributed-parallel manner. Compared with a serial implementation, the speedup factor of the algorithm is roughly the number of PPUs, which is 64 in our case. The implemented high-speed tracking is also more robust than its counterparts [5], [8], [10], [17]–[19], because the object feature is invariant to illumination changes and updated every frame to adapt to scale and rotation. The PE array and PPU array cooperating mechanism greatly facilitate the object search procedure, thus making high-speed robust tracking possible.

Table IV compares the proposed processor with other state-of-the-art digital programmable vision chip processors. The architecture has patch-parallel PPU array, which greatly increased the efficiency of the local-feature extraction and matching procedure. It extends the SOM neuron network with the help of PE and PPU reconfiguration, and the network scale is thus enlarged. The frame pipeline scheme of the architecture can overlap the image sensor exposure time to decrease the idle time of the processor. Constrained by the row-parallel architecture, previously reported vision chips use low dimension PPED features. In contrast, in our architecture, we can extract representative features whose feature dimension is more than ten times of PPED. Accordingly, the accuracies of detection, tracking, and recognition are improved. This architecture can carry out classification with SOM neural network or PPU array. The PPU array performs AdaBoost algorithm in distributed parallel. This method is extremely efficient when combining local features as weak classifiers. The other works, however, without patch-parallel PPU, implement classification with serial MPU or with only SOM neural network. To compare the hardware resource usage, we migrated work [21]

Fig. 20. Target tracking results.

TABLE IV
COMPARISON WITH PREVIOUS ARCHITECTURES

Ref	This paper	JSSC2014 [21]	JSSC2011 [19]	JSSC2009 [16]
Clock frequency	50MHz	50MHz	100MHz	50MHz
Patch-parallel	Yes	No	No	No
Parallelism	pixel-,patch-,thread-, distributed parallel	Pixel-, row-, thread-distributed-parallel	pixel-, row-parallel	row-parallel
MPU or CPU	32b dual-core	32b dual-core	8b single-core	32b single-core
Processor re-configurability	Dynamic between PE array, PPU array and SOM network	Dynamic between PE array and SOM network	Static among differently grained PE array processors	NO
Neural network	Yes	Yes	No	No
Neural network Scale	16×24	16×16	NA	NA
Frame Pipeline	Yes	No	No	Yes
Supported features	LBP, HoG, PPED	LBP, HoG, PPED	PPED	NA
Vector length	≥ 600	64	64	NA
Classification method	Distributed parallel (Adaboost or SOM)	Distributed parallel (SOM)	Serial (MPU)	Serial (MIPS)
Detection accuracy	High	Low	Low	NA
Hardware resources	69,672ALMs + 737Kb	43,846ALMs + 398Kb	NA	NA
Peark performance	31GOPS	12GOPS	44GOPS	76.8GOPS
System-level performance	340fps@face detection for 256×256 image	24fps@face detction for 256×256 image	10fps@face detection for 256×256 image	360fps@posture recognition for 128×128 image

onto the same FPGA platform, and about 58.9% and 85.2% additional adaptive logic module (ALM; basic logic unit in Arria V) and memory are used in the proposed work, respectively. Among the additional hardware, 90.2% ALM and 71.0% memory are contributed by the middle-grained PPU. It consumes nearly 2.5 times logic units (39 072 ALMs to 15 784 ALMs) and 240 kb additional memory (256 to 16 kb) when compared with RP design because the PPU logic is

much more complex than the previous RP, and more memory is needed for long FV storage. The PE array hardware cost increases about 12.0% and 12.5% in logic and memory due to the implementation of additional control logic and FIFO. The MPU in both systems is identical. Considering the additional hardware that improves the system peak performance nearly two times based on the Giga operations per second (GOPS) figure and enables our system for performing complex computer vision algorithms, which previous system cannot carry out, we think the additional hardware is worthwhile. Moreover, with a custom memory for vision chip, we can reduce as many as 60% of the memory area [47]. The peak performance of [21] is gained mainly through the PE array. In the proposed work, the PE array provides a similar performance, but with the help of the PPU array, we are able to achieve a total of 31 GOPS performance. We compared system-level performance using face detection algorithm. With the same image size, our system can run at 340 frames/s, whereas only 24 and 10 frames/s can be achieved in works [19] and [21]. The GOPS figure of our implementation is moderate when compared with [16] and [19], but it is sufficient for this architecture to achieve the highest system-level performance.

VI. Conclusion

A heterogeneous parallel processor for a high-speed vision chip is proposed. The architecture integrates four kinds of processors with different parallelisms and flexibilities. The PE array finishes pixel-parallel algorithms, while the PPU array performs algorithms in patch parallel. The mapping relationship between the PE array and the PPU array considers the modern computer vision algorithms. It greatly improves system performance and is suitable for different vision algorithms, especially for local-feature extraction and feature matching. The frame pipeline scheme and a flexible pixel-to-PE mapping relationship increase processor utility. A prototype was implemented with high-speed image sensor and high-performance FPGA. The architecture is able to finish multiple applications, such as feature extract, face detection, and object tracking in high speed.

References

[1] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*. New York, NY, USA: Thomson-Engineering, 2014.

[2] J. Ohta, *Smart CMOS Image Sensors and Applications*. Boca Raton, FL, USA: CRC Press, 2007.

[3] J. Clemons, A. Jones, R. Perricone, S. Savarese, and T. Austin, "EFFEX: An embedded processor for computer vision based feature extraction," in *Proc. 48th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2011, pp. 1020–1025.

[4] J.-P. Farrugia, P. Horain, E. Guehenneux, and Y. Alusse, "GPUCV: A framework for image processing acceleration with graphics processors," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2006, pp. 585–588.

[5] C. Mead, "Neuromorphic electronic systems," *Proc. IEEE*, vol. 78, no. 10, pp. 1629–1636, Oct. 1990.

[6] M. Ishikawa, K. Ogawa, T. Komuro, and I. Ishii, "A CMOS vision chip with SIMD processing element array for 1 ms image processing," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 1999, pp. 206–207.

[7] T. Komuro, I. Ishii, M. Ishikawa, and A. Yoshida, "A digital vision chip specialized for high-speed target tracking," *IEEE Trans. Electron Devices*, vol. 50, no. 1, pp. 191–199, Jan. 2003.

[8] V. M. Brea, D. L. Vilariño, A. Paasio, and D. Cabello, "Design of the processing core of a mixed-signal CMOS DTCNN chip for pixel-level snakes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 5, pp. 997–1013, May 2004.

[9] P. Dudek and P. J. Hicks, "A general-purpose processor-per-pixel analog SIMD vision chip," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 1, pp. 13–20, Jan. 2005.

[10] I. Ishii, K. Yamamoto, and M. Kubozono, "Higher order autocorrelation vision chip," *IEEE Trans. Electron Devices*, vol. 53, no. 8, pp. 1797–1804, Aug. 2006.

[11] N. Massari and M. Gottardi, "A 100 dB dynamic-range CMOS vision sensor with programmable image processing and global feature extraction," *IEEE J. Solid-State Circuits*, vol. 42, no. 3, pp. 647–657, Mar. 2007.

[12] W. Miao, Q. Lin, and N. Wu, "A novel vision chip for high-speed target tracking," *Jpn. J. Appl. Phys.*, vol. 46, no. 4S, p. 2220, 2007.

[13] J. Dubois, D. Ginhac, M. Paindavoine, and B. Heyrman, "A 10 000 fps CMOS sensor with massively parallel image processing," *IEEE J. Solid-State Circuits*, vol. 43, no. 3, pp. 706–717, Mar. 2008.

[14] D. Kim, J. Cho, S. Lim, D. Lee, and G. Han, "A 5000S/s single-chip smart eye-tracking sensor," in *IEEE Int. Solid-State Circuits Conf.-Dig. Tech. Papers*, Feb. 2008, pp. 46–594.

[15] W. Miao, Q. Lin, W. Zhang, and N. J. Wu, "A programmable SIMD vision chip for real-time vision applications," *IEEE J. Solid-State Circuits*, vol. 43, no. 6, pp. 1470–1479, Jun. 2008.

[16] C. C. Cheng, C. H. Lin, C. T. Li, and L. G. Chen, "IVisual: An intelligent visual sensor SoC with 2790 fps CMOS image sensor and 205 GOPS/W vision processor," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 127–135, Jan. 2009.

[17] Q. Lin, W. Miao, W. Zhang, Q. Fu, and N. Wu, "A 1,000 frames/s programmable vision chip with variable resolution and row-pixel-mixed parallel image processors," *Sensors*, vol. 9, no. 8, pp. 5933–5951, 2009.

[18] A. Lopich and P. Dudek, "A SIMD cellular processor array vision chip with asynchronous processing capabilities," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 10, pp. 2420–2431, Oct. 2011.

[19] W. Zhang, Q. Fu, and N.-J. Wu, "A programmable vision chip based on multiple levels of parallel processors," *IEEE J. Solid-State Circuits*, vol. 46, no. 9, pp. 2132–2147, Sep. 2011.

[20] B. Zhao, X. Zhang, S. Chen, K.-S. Low, and H. Zhuang, "A 64 × 64 CMOS image sensor with on-chip moving object detection and localization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 4, pp. 581–588, Apr. 2012.

[21] C. Shi *et al.*, "A 1000 fps vision chip based on a dynamically reconfigurable hybrid architecture comprising a PE array processor and self-organizing map neural network," *IEEE J. Solid-State Circuits*, vol. 49, no. 9, pp. 2067–2082, Sep. 2014.

[22] J. Yang, C. Shi, L. Liu, and N. Wu, "Heterogeneous vision chip and LBP-based algorithm for high-speed tracking," *Electron. Lett.*, vol. 50, no. 6, pp. 438–439, Mar. 2014.

[23] S. Kyo, S. Okazaki, and T. Arai, "An integrated memory array processor for embedded image recognition systems," *IEEE Trans. Comput.*, vol. 56, no. 5, pp. 622–634, May 2007.

[24] F. Gregoretti, R. Passerone, L. M. Reyneri, and C. Sansoé, "A high speed vlsi architecture for handwriting recognition," *J. VLSI Signal Process. Syst. Signal, Image Video Technol.*, vol. 28, no. 3, pp. 259–278, 2001.

[25] A. Broggi, G. Conte, F. Gregoretti, C. Sansoé, R. Passerone, and L. M. Reyneri, "Design and implementation of the paprica parallel architecture," *J. VLSI Signal Process. Syst. Signal, Image Video Technol.*, vol. 19, no. 1, pp. 5–18, 1998.

[26] K. Grauman and B. Leibe, *Visual Object Recognition* (Synthesis Lectures on Artificial Intelligence and Machine Learning), vol. 5. San Francisco, CA, USA: Morgan & Claypool, 2011, pp. 1–181.

[27] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European Conference on Computer Vision*. Heidelberg, Germany: Springer, 2006, pp. 430–443.

[28] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.

[29] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[30] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 1. Jun. 2005, pp. 886–893.

[31] T. Ahonen, A. Hadid, and M. Pietikäinen, "Face description with local binary patterns: Application to face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 12, pp. 2037–2041, Dec. 2006.

[32] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Comput. Vis. Image Understand.*, vol. 110, no. 3, pp. 346–359, 2008.
[33] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, "BRIEF: Computing a local binary descriptor very fast," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1281–1298, Jul. 2012.
[34] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time compressive tracking," in *European Conference on Computer Vision*. Heidelberg, Germany: Springer, 2012, pp. 864–877.
[35] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, 2004.
[36] Z. Cao, Y. Zhou, Q. Li, Q. Qin, L. Liu, and N. Wu, "Design of pixel for high speed CMOS image sensors," in *Proc. Int. Image Sensor Workshop*, 2013, pp. 229–232.
[37] *Altera Arria V*, accessed on Oct. 2014. [Online]. Available: http://www.altera.com/products/devkits/altera/kit-arria-v-starter.html
[38] J. Li and N. M. Allinson, "A comprehensive review of current local features for computer vision," *Neurocomputing*, vol. 71, nos. 10–12, pp. 1771–1787, 2008.
[39] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 1, pp. 105–119, Jan. 2010.
[40] Y. Moko, Y. Watanabe, T. Komuro, M. Ishikawa, M. Nakajima, and K. Arimoto, "Implementation and evaluation of FAST corner detection on the massively parallel embedded processor MX-G," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2011, pp. 157–162.
[41] K. Dohi, Y. Yorita, Y. Shibata, and K. Oguri, "Pattern compression of FAST corner detection for efficient hardware implementation," in *Proc. IEEE Int. Conf. Field Program. Logic Appl. (FPL)*, Sep. 2011, pp. 478–481.
[42] C. Shi, J. Yang, L. Liu, N. Wu, and Z. Wang, "A massively parallel keypoint detection and description (MP-KDD) algorithm for high-speed vision chip," *Sci. China Inf. Sci.*, vol. 57, no. 10, pp. 1–12, 2014.
[43] J. Yang, C. Shi, L. Liu, J. Liu, and N. Wu, "Pixel-parallel feature detection on vision chip," *Electron. Lett.*, vol. 50, no. 24, pp. 1839–1841, 2014.
[44] C. Shan, "Learning local binary patterns for gender classification on real-world face images," *Pattern Recognit. Lett.*, vol. 33, no. 4, pp. 431–437, 2012.
[45] *ORL Human Face Library*, accessed on Oct. 2014. [Online]. Available: https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html
[46] *Labeled Faces in the Wild*, accessed on Oct. 2014. [Online]. Available: http://vis-www.cs.umass.edu/lfw/
[47] Z. Chen, J. Yang, C. Shi, and N. Wu, "A novel architecture of local memory for programmable SIMD vision chip," in *Proc. IEEE 10th Int. Conf. ASIC (ASICON)*, Oct. 2013, pp. 1–4.

Jie Yang was born in Chongqing, China, in 1987. He received the B.S. degree in electronic science and technology from Tianjin University, Tianjin, China, in 2010, and the Ph.D. degree in microelectronics from the Institute of Semiconductors, Chinese Academy of Sciences, Beijing, China, in 2015.

He is currently a Post-Doctoral Fellow with the Department of Electrical and Computer Engineering, University of Calgary, Calgary, AB, Canada. His research interests include algorithms and very large-scale integration architecture of image processing, computer vision, wide dynamic range imaging, and deep learning.

Yongxing Yang received the B.S. degree from Tsinghua University, Beijing, China, in 2011. He is currently pursuing the Ph.D. degree with the Institute of Semiconductors, Chinese Academy of Sciences, Beijing.

His research interests include visual target tracking, image processing, computer vision, and deep learning.

Zhe Chen was born in Beijing, China, in 1989. He received the B.S. degree in electronic science and technology from Tsinghua University, Beijing, China, in 2011. He is currently pursuing the Ph.D. degree with the State Key Laboratory for Superlattices and Microstructures, Institute of Semiconductors, Chinese Academy of Sciences, Beijing.

His research interests include high-speed image processing circuits and time-offlight image sensors.

Liyuan Liu (M'11) received the B.S. and Ph.D. degrees in electrical engineering from the Institute of Microelectronics, Tsinghua University, Beijing, China, in 2005 and 2010, respectively.

He is currently an Associate Professor with the Institute of Semiconductors, Chinese Academy of Sciences, Beijing, where he is involved in high-performance analog front-end design for CMOS image sensors.

Jian Liu received the Dr.rer.nat. degree from the Julius Maximilian University of Würzburg, Würzburg, Germany, in 2003.

He joined the Institute of Semiconductors, Chinese Academy of Sciences, Beijing, China, where he is involved in semiconductor spintronics, optoelectronic detectors, and nanofabrication. He is currently a Full Professor in microelectronics and solid state electronics.

Nanjian Wu (M'05) was born in Zhejiang, China, in 1961. He received the B.S. degree in physics from Heilongjiang University, Harbin, China, in 1982, the M.S. degree in electronic engineering from Jilin University, Changchun, China, in 1985, and the D.S. degree in electronic engineering from the University of Electronic Communications, Tokyo, Japan, in 1992.

In 1992, he joined the Research Center for Interface Quantum Electronics and the Faculty of Engineering, Hokkaido University, Sapporo, Japan, as a Research Associate. In 1998, he was an Associate Professor with the Department of Electro-Communications, University of Electronic Communications, Chofu, Japan. Since 2000, he has been a Professor with the Institute of Semiconductors, Chinese Academy of Sciences, Beijing, China. In 2005, he was a Visiting Professor with the Research Center for Integrated Quantum Electronics, Hokkaido University, Sapporo, Japan. Since 2009, he has been an Honorable Guest Professor with the Research Institute of Electronics, Shizuoka University, Shizuoka, Japan. His research interests include mixed-signal large-scale integrated design and semiconductor quantum devices.