A Dynamically Reconfigurable Architecture Combining Pixel-Level SIMD and Operation-Pipeline Modes for High Frame Rate Visual Processing

Nao Iwata, Shingo Kagami and Koichi Hashimoto
Graduate School of Information Sciences, Tohoku University, Sendai 980–8579, Japan
Email: swk(at)ic.is.tohoku.ac.jp

***Abstract*— This paper describes a new reconfigurable processor architecture specialized for high frame rate visual processing. This architecture employs a 2-D mesh processing element (PE) array in which the PEs can be configured to operate as SIMD arrays or operation-pipeline trees depending on image processing algorithms so that maximum on-chip memory consumption is reduced. To achieve high on-chip memory utilization, the architecture features that the instruction register in each PE is mapped in its local memory space and that the ALU network and the local memory network can be configured independently. Simulation results show that the proposed architecture effectively utilizes both of the SIMD and operation pipeline modes.**

I. Introduction

High-speed real-time vision systems, which operate at a frame rate such as 1,000 frames/s, have been intensively investigated and developed in recent years. One type of implementation of such systems is a vision chip, which is a CMOS image sensor in which a processing circuit is integrated with a photo detector in each pixel. Recent progress of the CMOS image sensor technology has been enabling high-speed image readout, which contributes to realizing implementation with images processors separated from the sensor.

One of the most classical and commonly-used architectures for high-performance image processing is the pixel-parallel or pixel-level SIMD array. Also for high frame rate vision, due to its inherent affinity to image processing and its simplicity of implementation, it has been employed, in particular, by the systems based on or derived from vision chips [1], [2], [3], [4]. The column-level SIMD has also been used for high-speed vision systems especially in specially targeted systems that can exploit its column-parallel nature [5].

Recent progress of FPGAs introduced more diverse hardware designs for high-speed visual processing areas. FPGA-based specialized hardware implementations of individual algorithms such as the connected component labeling and various integral transformation have been proposed for the systems employing [6] or assuming [7] high-speed CMOS imagers. A high-speed target tracking system that utilizes preprocessing in FPGA and postprocessing in PCs has also been reported [8].

Although development of high-speed vision systems have been vigorously carried out as shown above, there have been few studies from the architectural point of view aiming at general-purpose programmable high-speed vision systems.

In designing high frame rate vision systems, it is most likely that the I/O subsystem becomes a bottleneck. In other words, supplying the processors with image data at sufficient speed is more difficult than building a high-speed and highly parallel image processor. In particular, under the circumstances where I/Os between the processor and external memories frequently occur, the overall performance substantially degrades from the peak performance. The pixel-level SIMD array architectures can be regarded as being ideal from this point of view because they are built on the assumption that the data set is always within the array instead of external memories. This point, on the other hand, introduces the problem of high requirement for the memory size per pixel. According to the experience of the authors, the memory size should be more than 100 bit per pixel for various image processing algorithms to be implemented without difficulty in a pixel-parallel SIMD array, and it occupies the major part of the chip area. This results in low area efficiency and thus pixel-parallel SIMD arrays with high spatial resolution are hard to be built.

This paper discusses a new programmable parallel processing architecture for high frame rate vision systems. In particular, our goal is to achieve a compact system suited for embedded implementation. Based on a pixel-level SIMD architecture, we adopt dynamic reconfigurability to it so that the precious per-pixel memory resource is efficiently utilized.

As an existing example of employing a dynamically reconfigurable processor for high-speed vision processing, implementation of connected component labeling using the DAPDNA-2 processor [9] is notable, which achieved 320 Mpixel/s. While the DAPDNA-2 is a general-purpose processor for wide use, our goal is to build an architecture that is specialized for high-speed real-time vision processing.

II. Design Strategy

One of the reasons for the memory inefficiency of pixel-level SIMD architectures is because it must execute the entire series of image processing including graylevel processing and binary processing uniformly in a plain SIMD array.

Consider a typical image processing flow in high-speed real-time vision. In earlier stages of image processing, after image acquisition, graylevel processing including noise reduction and feature enhancement such as edge enhancement and feature point detection are carried out. In these graylevel processing, the bits expressing the graylevel of input image must be allocated for each pixel at least, and in general much more is needed as temporary storage for computation. In later stages, on the other hand, the required memory amount is often greatly reduced since these stages mainly deal with reduced image data such as binary images or regions of interest.

When we design a pixel-level SIMD image processor, we must determine the per-pixel memory size so that it is not less than the peak memory usage over the series of image processing. Most of the on-chip memories are only used for a limited time and just idle for the other time.

Focusing on this point, we design the proposed architecture assuming that the pixel-level SIMD processing is applied mainly to the later stages after the required memory amount is reduced, which results in a reduced memory size for each pixel. On the other hand, a series of graylevel image processing over the whole image is carried out in a systolic array manner where the processing elements (PEs) are connected so that they form operation-pipeline networks according to the processing algorithms. The input image data from the frame memory are first fed into these pipeline networks, and the outputs from the pipelines, of which the data size are typically much reduced, are then processed in an SIMD manner.

To this end, we propose a 2-D mesh PE array architecture that is basically a pixel-level SIMD array and is also dynamically reconfigurable to form pipeline networks with each PE serving as a pipeline stage. Main contributions of this paper are that (1) the local memory networks can be configured independently from the ALU networks, which enables flexible utilization of the local memories in the operation-pipeline mode, and that (2) the instruction register in each PE can be used as data storage in the SIMD mode, which enables maximum utilization of the storage area.

III. Architecture

A. Whole Structure

The whole structure of the proposed architecture is shown in Fig. 1. It basically consists of a 2-D mesh-connected PE arrays. Each PE is mainly composed of an ALU, some registers, a local memory and communication links with adjacent PEs.

Instructions to the PEs, of which the size is 32 bits, are delivered through an instruction bus that is common to all the PEs in the SIMD mode. On the other hand, in the operation-pipeline mode, an instruction for a PE is stored in a dedicated instruction register within the PE. A PE executes the stored instruction every cycle repeatedly.

For our goal, employing a 1-bit ALU like many existing pixel-level SIMD arrays [1], [2], [3], [4] will suffer from large overhead because we must implement the 32-bit instruction register and relatively complicated inter-PE communication links in each PE. This motivated us to employ a multi-bit ALU instead of 1-bit ones, namely an 8-bit ALU. We do not assume that the number of the PEs equals to the number of pixels. A block of adjacent pixels is allocated to a PE, and the pixels within a block are iteratively processed.

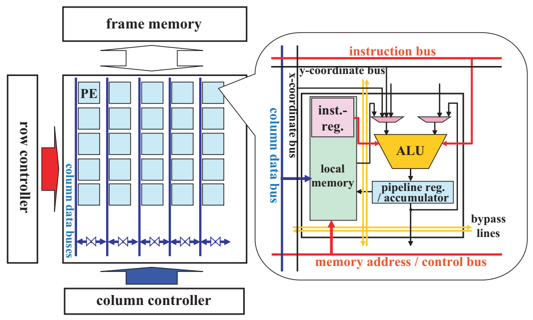

Fig. 1. Whole structure of the PE array.

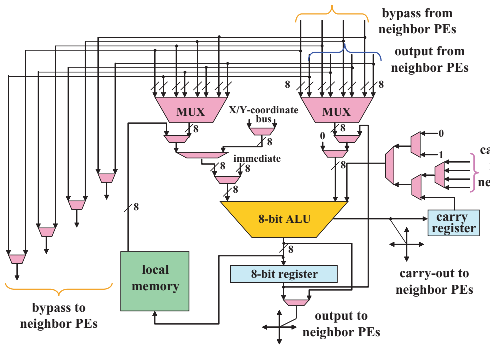

Fig. 2. Structure of the PE.

The local memories are controlled on the row basis. An address bus and control lines are common to the PEs within a row, but they are independent across different rows. A data bus is placed for each column, which is common to the PEs within a column. The contents in the local memories and instruction registers in the PEs are directly accessed through these column data buses. Adjacent column data buses are programmably connected or disconnected so that they can serve as column-wise independent buses or access buses for shared memory blocks consisting of adjacent multiple columns.

The PE array also contains image coordinate value buses, which are useful for pixel-level SIMD processing depending on image coordinates such as geometrical feature (e.g. centroid) extraction [3], [4].

B. PE Structure and ALU Networks

The detailed structure of a PE is shown in Fig. 2. A PE contains an 8-bit register in addition to the local memory. In the SIMD mode, this is used as an accumulator for arithmetic and logical operations. The ALU takes operands from the accumulator, the local memory and the accumulators of the four neighbor PEs. The result of the ALU operation is stored in the accumulator and a carry register. Bit-wise logical operations can be applied to processing eight binary image pixels at a cycle, that is, bitwise SIMD operations within a PE in the pixel-level SIMD array. Because we assume that the pixel-level SIMD mode is mainly for binary image processing, most of the ALU bits would be wasted without this technique.

In the operation-pipeline mode, on the other hand, the 8-bit register is used as a pipeline stage register. Each PE takes operands from at most two neighbor PEs, and writes the result to the pipeline stage register.

A simple 4-neighbor 2-D mesh topology is not sufficient for complicated data flows. Several measures are taken to address this issue. Each PE contains bypass connections for the left from/to right directions and the upper from/to lower directions. A PE can take operands also from the four neighbor bypassing outputs. Moreover, a result of an ALU operation can be sent directly to the neighbor PEs without storing in the pipeline stage register, which allows multiple ALUs in adjacent PEs to be chained. The carry input used in arithmetic operations can also be taken from neighbor PEs without going through the carry registers, which allows wider-bit operations such as 16-bit or 24-bit arithmetics in a cycle.

These additional components contribute to increasing the degrees of freedom of layout, and various data flows can be implemented in the 2-D PE arrays. This flexibility is obtained at the cost of manageability of delays. This management must be covered by a development software and it should be addressed in future work.

The instruction register is mapped in the PE local memory space, and it can be used as a data storage in the pixel-level SIMD mode. Typically this is useful because pixel-level SIMD processing requires as much memory as possible. On the other hand, if we have extra memory for a target algorithm, the instruction registers should be left unchanged so that we can save time to reload the instructions when the array is reconfigured back to the operation-pipeline mode again. Moreover, if we have so much extra memory that several instructions can be stored in the local memory, the multiple context reconfiguration is possible where multiple pipeline networks can be switched instantaneously. Note that reconfiguration to the SIMD mode can always be done instantaneously because no instruction loading is needed.

C. Local Memory Networks

Operation pipelines for image processing algorithms typically include relatively large memory components, e.g. line buffers or processing result stores, within or around the pipeline networks. These memory components are difficult to be implemented on a simple array of PEs with tightly-coupled ALU and local memory. More specifically, the ALUs tend to be idle at a part of the array where the memories are needed, and vice versa.

To address this problem, the proposed architecture employs a local memory network of which the topology can be configured independently from the ALU operation-pipeline network. The local memory can be connected directly to its left and right PEs without any help of the ALU. It can also be connected to the column data bus.

When an operation pipeline requires a line buffer, it is implemented as a shift register in which the local memories are connected horizontally. Since a word of the local memory contains eight bits, a shift register for an M-pixels 8-bits image is implemented using one PE when the local memory of a PE contains M words. When a required shift register is too long to be implemented in a row of PEs, several rows can be concatenated by connecting them at the left or right ends through the ALUs.

Because the column data buses can access the local memories and the buses can be connected, an array of local memories can also be used as a large shared memory block. For example, output data from an operation pipeline can be stored into any local memory within the shared memory block. By storing output data corresponding to a pixel into the PE that is assigned for the pixel in the SIMD mode, the reconfiguration to the SIMD mode can take place instantaneously without any need for image data relocation. Note that output data cannot be stored into a PE of which the local memory is incorporated within a busy operation pipeline. Even in this case, the output data can be stored into a PE that is in the same column as the final target PE, and then image data relocation takes place within the column so that column-level parallelism can be exploited for fast data relocation.

IV. Preliminary Evaluation

While our final goal is to implement the architecture as a full-custom chip, for the present paper we designed our architecture in Verilog-HDL targeted for an FPGA device. We assumed that the number of pixels $N \times N = 128 \times 128$, and designed an array with 16×16 PEs each of which contains an 8 [bits] $\times$ 20 [words] local memory. We synthesized our design by XILINX XST in ISE WebPACK 8.2i for the XC3S1600E (Spartan-3E) device. The circuit occupied approximately 1,888,000 equivalent gates and the operation frequency for a single PE was estimated as 46.3 MHz.

As a typical example of high-speed real-time visual processing, a target tracking algorithm was programmed in the proposed architecture. The algorithm consists of (1) 3×3 linear smoothing filter, (2) binarization, (3) area filling with a seed point being the centroid of the target at the previous frame, and (4) centroid calculation.

The area filling algorithm includes $N(= 128)$ iterations of (3a) dilation of the window image and (3b) replacing the window with pixel-wise logical AND of the current image and the window, where the initial image of the window is the seed point. This is a popular technique to extract a target from an image without retaining whole images from frame to frame.

The processes (1) and (2) described above are programmed in the operation-pipeline mode since they handle grayscale whole image data. The layout of the filter pipeline is shown in Fig. 3. The processes (3) and (4) are programmed in the SIMD mode because they are binary image processing. The

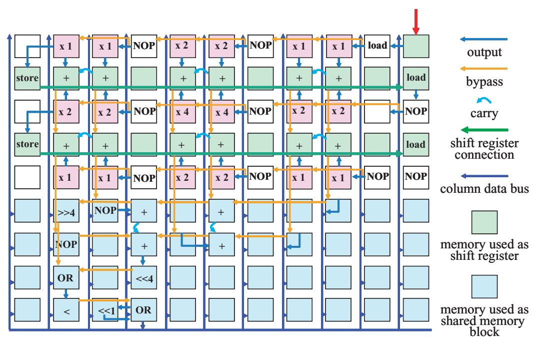
store
load
NOP
output
bypass
carry
shift register connection
column data bus
memory used as shift register
memory used as shared memory block

Fig. 3. A layout of the smoothing filter pipeline.

TABLE I
COMPUTATION TIMES OF TARGET TRACKING.

task	time [clk]
loading instruction	64
image input (128 × 128 pixels, 1 pixel/clk)	16384
3 × 3 smoothing latency	268
reconfiguration to SIMD mode (data relocation)	32
loading previous centroid point	128
area filling (repeated 128 times)	21504
centroid (0th and 1st order moments)	2830
total	41210

input image is read out from the frame memory in the raster scan order by 1 pixel (8 bits) per clock cycle.

The computation time of the above algorithm was evaluated through RTL simulation using an input sequence of 128 × 128 pixels 8-bits images. The results are shown in Table I. It shows that the frame rate of 1,000 frames/s is achieved at no more than 42-MHz clock frequency and with such narrow input image bandwidth as 8 bits per cycle. For k-times larger images to be handled, increasing the number of PEs and the input bandwidth by k times is needed.

To evaluate the effectiveness of the mode reconfiguration, we compared the required memory amount for the above target tracking algorithm in the proposed design with those in a pure operation pipeline processor and in a pure pixel-level SIMD processor. The required memory amounts over the whole array are shown in Fig. 4, where $N \times N$ pixels and $M \times M$ PEs are assumed. Note that the data for the pure pixel-level SIMD are optimistic estimates in which only the input image data size ($= 8N^2$ [bits]) is considered because temporary memory consumption highly depends on one's programming skill in pixel-level SIMD processors.

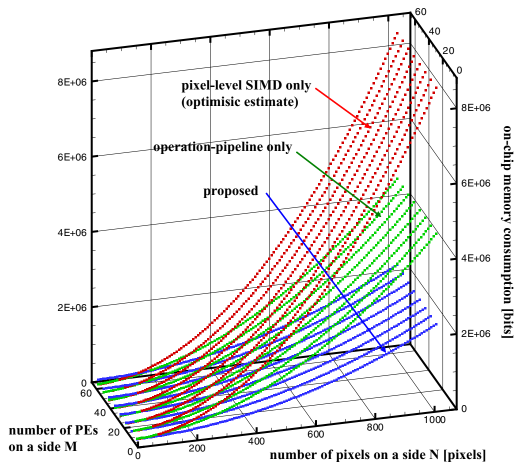

Fig. 4. Comparison of memory consumption.

With respect to the computation time, the pixel-level SIMD was always advantageous for the above target tracking algorithm provided that there are no limit for the memory size. Considering the required memory amount for the pixel-level SIMD, the authors found that the mode reconfiguration is reasonable for this algorithm.

We also found that the computation time for the area filling when we assigned 64 pixels to a PE is approximately 10 times longer than that when we assigned 1 pixel to a PE. This shows the bitwise SIMD processing within a PE is effectively utilized since the gained parallelism is approximately 6.4 for the eight-parallel SIMD operation.

V. CONCLUSION

A reconfigurable processor architecture specialized for high frame rate visual processing has been presented. The architecture exploits the reconfigurability between the operation-pipeline mode and the pixel-level SIMD mode to achieve high on-chip memory utilization. Future work will include more detailed evaluation, refinement of the design, and implementation as a custom chip.

REFERENCES

[1] R. Forchheimer and A. Åström, "Near-sensor image processing: A new paradigm," *IEEE Transactions on Image Processing*, vol. 3, no. 6, pp. 736–746, 1994.

[2] T. M. Bernard, Y. Zavidovique, and F. J. Devos, "A programmable artificial retina," *IEEE Journal of Solid-state Circuits*, vol. 28, no. 7, pp. 789–798, 1993.

[3] T. Komuro, S. Kagami, and M. Ishikawa, "A dynamically reconfigurable SIMD processor for a vision chip," *IEEE Journal of Solid-state Circuits*, vol. 39, no. 1, pp. 265–268, 2004.

[4] Y. Nakabo, M. Ishikawa, H. Toyoda, and S. Mizuno, "1ms column parallel vision system and its application of high speed target tracking," in *2000 IEEE International Conference on Robotics and Automation*, 2000, pp. 650–655.

[5] R. Johansson, L. Lindgren, J. Melander, and B. Möller, "A multi-resolution 100 GOPS 4 Gpixels/s programmable CMOS image sensor for machine vision," in *2003 IEEE Workshop on Charge-Coupled Devices and Advanced Image Sensors*, 2003.

[6] K. Fujiwara, K. Yamamoto, and I. Ishii, "Development of all-pixels processing type high-speed mega pixel vision," in *12th Symposium on Sensing via Image Information*, 2006, pp. 88–92, (in Japanese).

[7] S. Hirai, M. Zakoji, A. Masubuchi, and T. Tsuboi, "Realtime FPGA-based vision system," *J. Robotics and Mechatronics*, vol. 17, no. 4, pp. 401–409, 2005.

[8] R. Okada *et al.*, "High-speed object tracking in ordinary surroundings based on temporally evaluated optical flow," in *2003 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2003, pp. 242–247.

[9] T. Sugawara, K. Ide, and T. Sato, "Dynamically reconfigurable processor implemented with IPFlex's DAPDNA technology," *IEICE Trans. Information and Systems*, vol. E87-D, no. 8, pp. 1997–2003, 2004.