In-Vehicle Vision Processors for Driver Assistance Systems

Shorin Kyo

System IP Core Research Laboratory
NEC Corporation
Kawasaki City, 211-8666
Tel : +81-044-431-7453
Fax : +81-044-431-7489
e-mail: s-kyo@cq.jp.nec.com

Shin'ichiro Okazaki

System IP Core Research Laboratory
NEC Corporation
Kawasaki City, 211-8666
Tel : +81-044-431-7453
Fax : +81-044-431-7489
e-mail: s-okazaki@cq.jp.nec.com

Abstract - This paper describes existing designs and future design trends of in-vehicle vision processors for driver assistance systems. First, requirements of vision processors for driver assistance systems are summarized. Next, the characteristics of vision tasks for safety are described. Then several in-vehicle vision processor LSI implementations are reviewed, and the design approach of one of them, the IMAPCAR highly parallel processor, is further described in detail. Finally, future trends of in-vehicle vision processors focusing on their architectures and application coverage expansion such as integration of vision for safety, Digital TV codec, and 3D graphics functions of future car navigation, are discussed.

I Introduction

Vision processing (or image recognition) is the subject of high expectations as a promising function for assisting drivers to improve driving safety. In future driver assistance systems, multiple cameras will be mounted to capture images inside and around the vehicle, along with other sensors such as radar and ultrasonic sensors, all simultaneously providing basic data for retrieving information on the environment around the vehicle to avoid traffic accidents. Such implementation will steadily raise the bar in terms of computation performance for future in-vehicle vision processors. Furthermore, due to the continuous evolution of vision tasks for coping with a wide variety of applications, and as well as various recognition targets and ambience changes, processors targeting this emerging market must also provide high flexibility (programmability). Finally, low cost and low power implementation with compact (space saving) realization are also a major requirement for in-vehicle vision processors due to limited in-vehicle space budgets and severe temperature and mechanical (vibration) conditions over a long period.

As a result, requirements of in-vehicle vision processors for future driver assistance systems can be summarized as low cost, high performance, and high flexibility. Generally the total circuitry (= total cost) of a processor LSI can be roughly classified into two categories: "operational circuitry" (for example, data-path and wired logic), which determines its theoretical peak performance; and "control circuitry" (for example, instruction issuance management and interface logic), which contributes mainly to improved its flexibility. Based on the above classification, an inevitable trade-off between performance, cost, and flexibility arises as illustrated in Figure 1 (a). Figure 1 (b) shows the Control versus Operational circuit Ratio (COR) observation of several representative processor implementations estimated from their die photos [1][6][10][12], demonstrating a fairly accurate match between the COR value of each processor category and its well-known flexibility degree: the larger the COR, the higher the degree of flexibility. The challenge of in-vehicle vision processor design will be, to provide a compromised solution against these requirements, i.e. to balance performance, flexibility, and cost by fully taking into account the features of the target application domain. This paper describes existing designs and future trends of in-vehicle vision processors for driver assistance systems focusing on this issue.

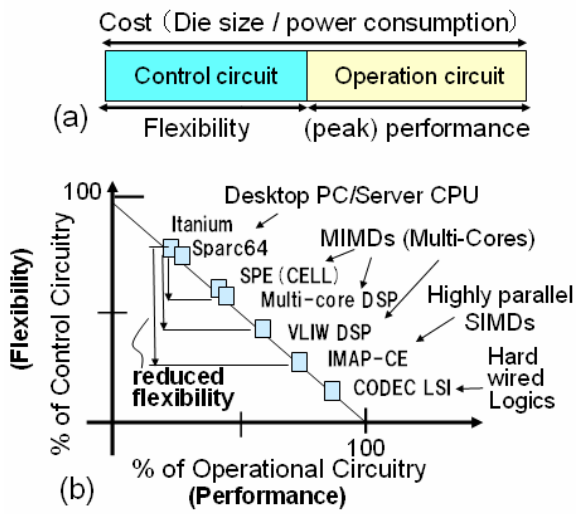

Fig. 1. Relationship between performance, cost, and flexibility, and the Control versus Operational circuit Ratio (COR).
(a) Relationship between performance, cost, and flexibility.
(b) The Control versus Operational circuit Ratio (COR) observation of several representative processor LSI implementations.

Section II describes in greater detail both the performance and flexibility requirements of vision applications for safety, and then reviews current commercial LSI implementations for automotive application. In Section III, one of such LSI implementation, IMAPCAR, is described with further details about its architectural and LSI implementation features, as

978-1-4244-1922-7/08/$25.00 ©2008 IEEE

well as flexibility. Finally, Section IV describes future trends toward better solutions, balancing performance, flexibility, and cost requirements, including the direction of fusing vision processing architectures with other media processing architectures for Digital TV codec and 3D graphics. Section V presents the conclusions.

II. Vision Applications and Processor Implementation Characteristics for Automotive Applications

A. Variety of vision applications and vision algorithms

Vision processing for driver assistance includes the recognition of the vehicle's surrounding environment as well as recognition of inside vehicle conditions. Surrounding environment recognition targets include on-road vehicles, obstacles and pedestrians as well as signals, traffic signs, on-road signs (marks) and other temporary board signs (such as "under construction"). Inside vehicle condition recognition includes recognition of the driver's status such as driver drowsiness (eyelid status monitoring), face/eye direction and driver posture.

Beside the above stated variety, the capability of vision processing for automotive application should be robust and tolerate various outside scene conditions and environmental changes. These conditions include road and traffic conditions, lighting conditions (day, night, tunnels), and also weather conditions (sunny, cloudy, rainy). Rapid changes of these conditions should also be considered. In order to recognize in a robust way various types of objects under each varied environmental condition, the use of more kinds of vision processing algorithms is required. For example in on-road vehicle detection, it is necessary to utilize appropriate vision processing algorithms depending on the relative position (front, rear, side) of the target vehicle. Various preprocessing algorithms are also required according to weather and lighting conditions for increasing the reliability of recognition results.

Typical on road vehicle recognition consist of two steps. The first step is quick search and detection of target candidates (hypothesis generation). The second step is validation of candidate correctness (hypothesis verification) [14]. During the hypothesis generation step, various kinds of algorithms are proposed including vehicle detection using knowledge of symmetry, color, shadow, corner, vertical/horizontal edge, texture and light for vehicles[5]. Vehicle detection can also be performed by using stereo information using a disparity map, as well as motion information based on optical flow calculations. For the hypothesis verification step, template based methods using correlations of images and templates are proposed. Learning algorithms that learn the vehicle appearance characteristics from a set of images are also used. For example, vehicle and non-vehicle pattern classification problems are solved through feature extraction using PCA (Principal Component Analysis) and classification using Neural Nets or Support Vector Machines. Moreover, these algorithms are also frequently used in combination with tracking algorithms [14].

B. Performance requirement and processing characteristics

Vision processing for safety purposes, which is critical for human life, must be realized with extremely high reliability and recognition accuracy. In some cases, more than several hundred GOPS (Giga Operations Per Second) are required to realize such highly reliable recognition and real-time processing of vast amounts of data within a limited time. Figure 2 anticipates the required computational complexities, the expected date of realization of these applications, and also the performance scaling of embedded single core CPUs, multi-core CPUs, and dedicated vision processors, showing that performance scaling of general purpose processor will not meet the performance requirements of vision applications for safety in the near future without the use of specifically designed vision processor ASSPs.

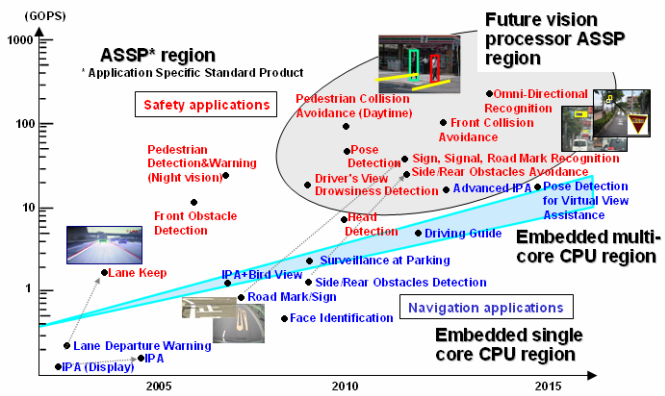

Fig. 2. In-vehicle vision complexities, performance requirements, and expected performance scaling of processors.

Generally vision processing algorithms can be categorized into low-level, intermediate-level, and high-level processing (refer to Figure 3), where tasks such as preprocessing and feature detection be classified to low-level, tasks such as statistical measurements, matching, and classification be classified to intermediate-level, and finally tasks such as hypothesis verification be classified to high-level processing. In order to achieve high performance with low power consumption, the existence of rich data locality and a large amount of pixel level parallelism in low-level processing should be fully exploited by using highly parallel processor architectures. Task level parallelism and somewhat increased irregular data access patterns of intermediate-level processing should also be exploited by a medium level parallel processor architectures. On the other hand, high-level processing which utilizes decision information calculated by intermediate-level processing normally has less parallelism. Its reduced computational complexity compared with low- and intermediate-level processing enable realization by traditional CPUs.

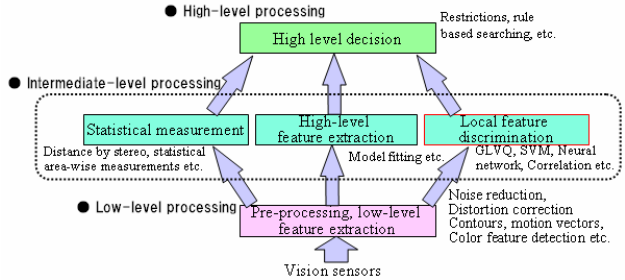

Fig. 3. Processing structure of vision applications.

C. *In-vehicle processor LSI implementations*

Based on the knowledge of the flexibility and performance requirements of vision processing, while in addition tries to as well fulfill the low cost requirements from a commercial product point of view, vision processor design approaches that have actually been chosen for use in commercial products thus far can be divided into three types. The first design approach is based on using parameter tunable dedicated hardware that cooperates with embedded CPU. The second design approach fuses dedicated hardware with embedded CPU in a tighter way to improve flexibility. Finally the third design approach uses a fully programmable highly parallel architecture to maximize performance, while tries to overcome the reduced flexibility by establishing and promoting parallelizing techniques for the architecture.

VCHIP [9] developed by Hitachi belongs to the first design approach, and has been adopted by NISSAN as the processing engine of an in-vehicle lane departure warning system in 2001. VCHIP consists of fundamental image processing dedicated circuits including several filtering functions, binarization and histogram processing, and is viewed as a hardware accelerator from the CPU. Degree of flexibility of VCHIP is limited to several parameter-setting of the dedicated hardware. This approach allows maximum cost reduction, and thus is desirable if target vision applications are limited and their algorithms can be fixed at an early stage.

Visconti [4][12] developed by Toshiba belongs to the second design approach, and has been adopted by HONDA as the processing engine of its intelligent night vision system at 2004. Visconti consists of 3 MeP (Media embedded Processor) modules, where each MeP module is a programmable 3-way VLIW coprocessor with five 8-parallel (8-bit x 8) SIMD arithmetic units, and achieves 18 GOPS at 150 MHz. The chip contains 21M transistors with 1 W power consumption operating at 1.5 V using the 0.13 μ m process technology. The design approach of MeP uses the so called SWAR (SIMD within a register) technique, similar to the well-known MMX technology of desktop CPU, to accelerate computation at the instruction sub-word level in low cost, while integrates multiple MePs to exploit task level parallelism. Degree of flexibility of Visconti originates from its combination of both a restricted SIMD and MIMD supports, which will be further elaborated in Section IV.

IMAPCAR [11] developed by NEC and NEC Electronics, which is a re-design of IMAP-CE [6] for fulfilling the temperature and reliability requirements for automotive use, takes a third design approach. It has been adopted by TOYOTA LEXUS as the processing engine for obstacle detecting pre-crash safety system at 2006. IMAPCAR employs the highly parallel SIMD linear array processor architecture, aiming at providing a highest performance in low-cost, while still maintaining a degree of flexibility supported by a fully programmable PE array RISC instruction set. The chip contains 26.8M transistors with power consumption estimated to average 1 to 2 W operation at 1.2V using the 0.13 μ m process technology, and achieves 100 GOPS at 100MHz. The next section describes in detail the design strategy and implementation details of IMAPCAR.

III. The IMAPCAR Highly Parallel Linear Array Processor

D. *Basic components*

Figure 4 shows the main building blocks of IMAPCAR. IMAPCAR consists of 128 8-bit 4-way VLIW RISC PE each equipped with a 24-bit MAC (Multiply Add Accumulation), 256KB (2KB/PE) of data RAM (IMEM), one CP (16-bit RISC Control Processor) with 32KB program and 2KB data caches, and a DMA engine for data transfer between the IMEM and external SSRAM (Synchronous SRAM). A shift-register style ring network is used for inter-PE communication. Another shift-register configuration is also used for transferring up to three channels of NTSC format video data in parallel with PE operation, into the IMEM or external SSRAM. As a result, IMAPCAR can handle up to two video streams of 640 or 768 pixels wide, or up to three video streams of 512 pixels wide at the same time. This mechanism makes it easier to efficiently compose multi-channel vision applications, such as the stereo vision applications of the TOYOTA LEXUS system [2].

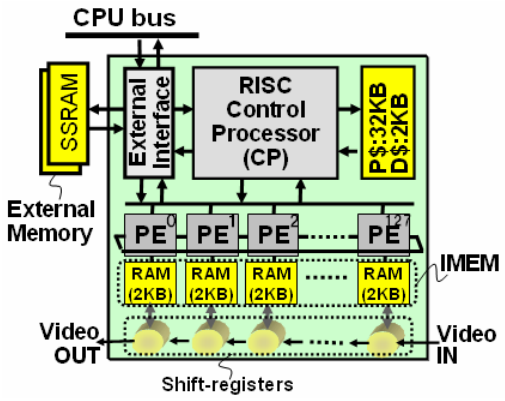

Fig. 4.IMAPCAR block diagram.

The LSI is packaged using a 500-pin TBGA and satisfies the temperature range requirements (-40 to +85 degrees Celsius) for automotive use. To ensure long-term supply and as well full reliability control for automobile usage, the SSRAM is adopted as external memory devices instead of SDRAM. Furthermore, program code area in the external memory and the program cache body within the CP are protected by adding 4bits of ECC (Error Correction Code) in addition to each 32bit program word, while other memory area including data cache body and IMEM are protected by one parity bit per 8bit data word. Figure 5 shows the die photo and a photo of its PCI evaluation board. In case the external interface detects parity error, or ECC error that cannot be recovered, an interrupt signal will be generated and passed to the external supervisor CPU.

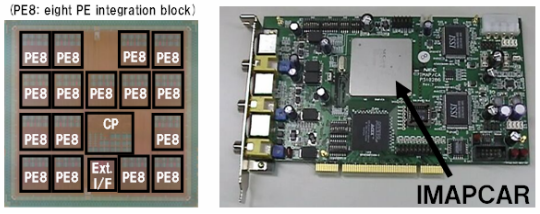

Fig. 5.Die photo and IMAPCAR PCI test board.

E. *Programming support*

A data parallel C extension called 1DC (one dimensional C) is used as the programming language of IMAPCAR. In 1DC, entities associated with the PE array are declared by using a "sep" ("separate") keyword. A "sep" data item possesses a number of scalar elements that is a multiple of the number of PEs. Explicitly parallel operations are specified using "sep" variables in 1DC expressions. Six major primitive operator extensions of 1DC from C are shown in Figure 6.

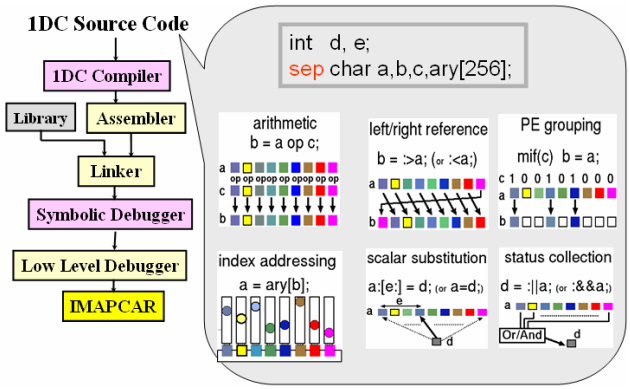

Fig. 6. Six primitive 1DC syntax forms (extension from C) and the 1DC compiler system

Figure 7 shows the IMAPCAR programming model. Column-wise mapping of image to each PE is assumed. The collection of all PE local memories (IMEM) is called the 2-D memory plane, where the source, destination and work images can be explicitly allocated by using 1DC. The line drawn on the 2-D memory plane is called a Pixel Updating Line (PUL). The PUL consists of a collection of memory addresses (or pixel locations) upon which the PEs work simultaneously in each time unit. Each PUL can be regarded as directly corresponding to one variable of the "sep" data structure, and thus can be easily generated and controlled by user programs.

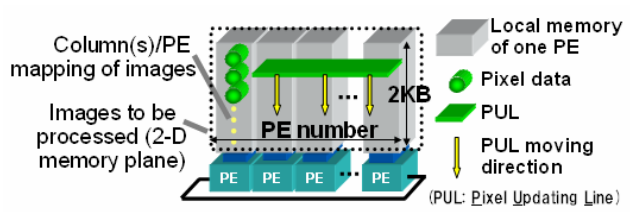

Fig. 7. IMAPCAR program working model

Parallelizing methods are designed based on the idea of sweeping PUL in various ways across the 2-D memory plane [7]. As described in the next subsection, the one RAM/block per PE configuration enables each PE to access a different memory address at the same time, thus enabling sweeping of the PUL in various ways across the 2-D memory plane. A mixture of multiple PULs is usually effective; for example, a 2-D FFT is implemented by first applying a row-wise PUL (1D-FFT), followed by a row-systolic PUL (transposing columns to rows), and finally another row-wise PUL (1D-FFT). Several other examples of PUL mixtures are shown in Figure 8.

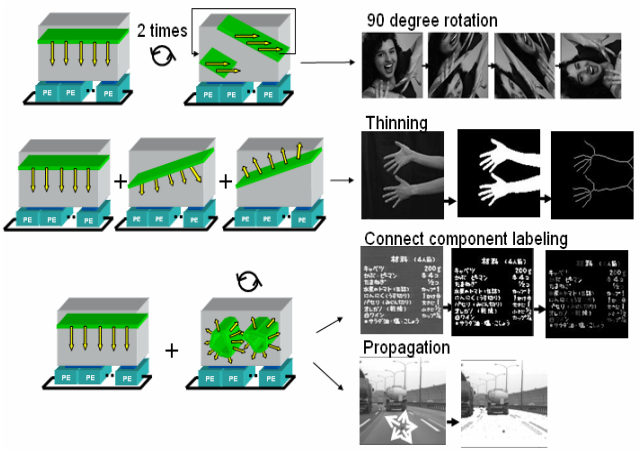

Fig. 8. Examples of PUL combinations for parallelizing typical vision tasks.

F. *Hardware support of 1DC execution*

One important design goal of IMAPCAR is to achieve an efficient execution of 1DC codes. The hardware features of IMAPCAR that enable efficient execution of the six essential syntax forms of 1DC are summarized here.

- Column(s)/PE Image Data Mapping

Automatic column-wise mapping of image data to each PE is a precondition for designing most algorithms based on various parallel methods using 1DC. Efficient mapping is achieved by using multiple inter-PE shift register paths for video I/O.

- Arithmetic Operation

Based on the 4-way VLIW execution unit of each PE, a maximum of three "sep" data arithmetic operations and a maximum of one memory access operation can be accomplished within one clock cycle. The 1DC optimizing VLIW compiler provides an efficient way of VLIW code generation from 1DC descriptions.

- Left/right Reference Operation

Left/right reference operation is one of the most frequently used operations, and is also an essential functionality for implementing for example image filters using the PE array. Each PE is designed to perform a direct register-to-register data access of neighborhood PE using the inter-PE shift register path of the ring network, enabling execution of the left/right reference operation within one clock cycle.

- Index Addressing Operation

Index addressing is an essential functionality for designing algorithms based on sweeping the PUL across the 2-D image plane. For achieving index addressing, each PE is assigned a separate RAM block as the local memory, which enables each PE to perform access to a mutually different memory address in the same clock cycle.

- PE Grouping Operation

PE grouping operation is used for implementing SIMD style conditional branch statements. Several hard-wired instructions, which are combinations of arithmetic and flag manipulation operations, are implemented in addition to the PE RISC instruction set, for achieving the PE grouping operation in one clock cycle.

- Scalar-Substitution/Status-Collection Operation

The tightly coupled CP-PE pipeline achieves fast communication between the CP and the PE array, by which each scalar-substitution or status-collection operation is accomplished within three pipeline stages, where the throughput is one clock cycle.

G. Evaluation

The performance of major image processing tasks and vision applications implemented on IMAPCAR by using several standard parallelizing techniques [7] are evaluated by comparing 1DC code running on a 100MHz IMAPCAR with C code running on a 2.4GHz Intel P4, a GPP (general purpose processor). As shown in Figure 8, the first benchmark, which consists of various image processing tasks, IMAPCAR achieves a performance up to 8 times better with a speedup approximately proportional to the amount of inherent parallelism of each task. In the second benchmark, which uses lane-mark and vehicle detection applications [5] containing various types of image operations, IMAPCAR achieves a performance that is approximately three times better. A few other benchmark reports can be found in [12].

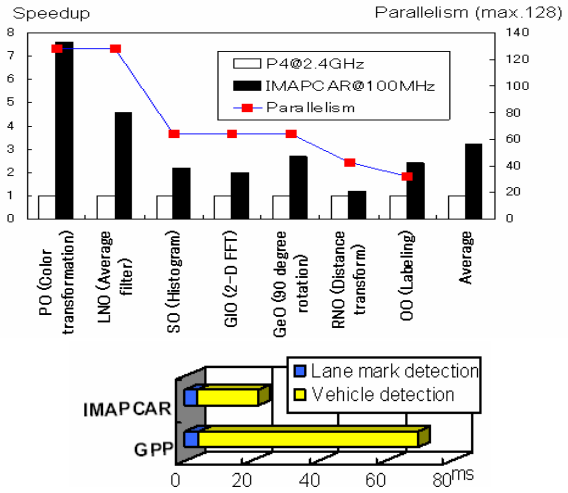

Fig. 8. Speedup ratio of C code running on a 2.4GHz P4 compared with 1DC code running on a 100 MHz IMAPCAR, using various vision tasks.

IV. Future Trends of In-Vehicle Vision Processor Designs

H. Existing design approach analysis

The implementation example of in-vehicle vision processors described in the previous sections can also be plotted onto the COR graph (refer to Section 1), as shown in Figure 9, mainly based on their die photos. For IMAPCAR and Visconti, the COR graph is plotted based on the COR value observation of the whole chip [4][15], and also on that of the area of the PE array (*IMAPCAR-PEs*) and the VLIW coprocessor (*Visconti-VLIW*) respectively. Note that because the die photo of VCHIP is not available, the plots for VCHIP and *VCHIP+CPU* integration are based on prediction.

First of all, by focusing on VCHIP, *Visconti-VLIW*, and *IMAPCAR-PEs*, a gradual rise in the degree of flexibility can be observed from the first to the third design approaches, reflecting the differences in strength of each design approach

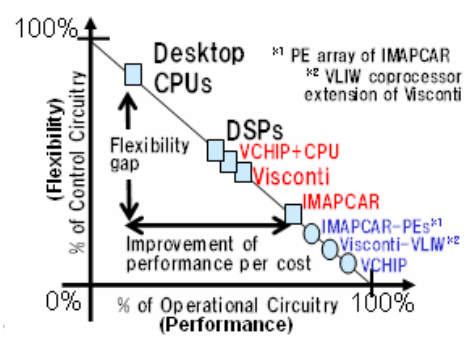

Fig. 9. COR relationship between existing in-vehicle vision processor architectural design approaches.

with regard to support of various types of low-level processing.

The rise in flexibility from the first to the second design approach is at the cost of the increased expense for additional switches between computing units. By adding such extra switches, computing units within the dedicated circuits of the first approach acquire the flexibility of being separately controlled by each instruction word of a very long instruction word (VLIW) generated from user programs, and fed by operands from various register sources. In this way, higher flexibility with respect to computing unit usage can be achieved by the second approach, thus supporting more kinds of low-level processing.

However, while the second design approach allows only collective use of the extended computing units, that is, allows only data located in a continuous memory address to be efficiently fed at the same time to the extended computing units, the third design approach demonstrated by IMAPCAR allows more flexible memory data supply towards a highly parallel set of computing units, at the cost of a highly banked memory array configuration. However, the cost of such configuration is estimated to be a modest increase in die size of approximately 10 percent, compared to the case where eight 2 KB (8b/W) RAMs are unified into a single 16 KB (64b/W) RAM for each 8 PE integration block (refer to Figure 5). The resulting flexibility has been demonstrated in the previous section to drive the PUL in various ways across the 2D-memory plane (refer to Figure 8), by which acceleration of various low-level processing tasks, as well as the support of various memory access patterns existing in intermediate-level processing tasks are achieved.

On the other hand, the plots *VCHIP+CPU*, Visconti, and IMAPCAR in Figure 9 show a gradual increase of opportunity for achieving higher control flexibility. Under the same cost constraint, integrating VCHIP with more complex or higher frequency CPUs may become possible. So does Visconti achieve the integration of three individual MeP modules into a single chip. In contrast, as IMAPCAR uses most of its die area for a huge PE array, little area is left for the CP (refer to Figure 5). As a result, while Visconti and IMAPCAR consume a similar number of transistors and power under the same process technology, the peak performance of the former is one fifth that of the latter, while in exchange task level parallelism is easier exploited by the former. To support task level parallelism while at the same time maintain a performance as high as that of IMAPCAR without raising power and transistor consumption requires further architectural evolution.

I. Future trend and architectural design directions

As described in the previous subsection, the design approaches of in-vehicle vision processor have each tried to balance the performance and flexibility requirements under a fixed cost (such as up to 2W of power consumption). Consequently, further pursuing low-power technologies such as DVFS and clock gating at the circuit level, as well as further pursuing low-cost realization of performance and flexibility against various low- and intermediate-level vision tasks at the architectural level will be the next important subjects. As data level parallelism has been thus far successfully exploited by previous low-cost architectural techniques such as VLIW instructions and highly parallel SIMD arrays, the next major architectural level topic for in-vehicle vision processor design will be to find low-cost design approaches for exploiting task level parallelism so as to further accelerate intermediate-level processing of vision tasks. Such direction of evolution will enhance the versatility of vision processors, however, the strict cost limitation coming from the product side and the performance and flexibility requirements from the vision application side will drive such research activities, whereby the architectural design approaches for vision processors will be differentiated from those for more general-purpose oriented future multi- or many-core processor design approaches. One research activity toward this challenge can be found in [8], where a novel hardware component reuse design methodology is proposed to support task level parallelism through low-cost dynamic reconfiguration of a pure highly parallel SIMD architecture.

The authors anticipate that, at the end of the realization of a wider application coverage in-vehicle vision processor, the role of vision processors will be combined with that of media processors for car navigation systems, resulting in a unified multi- or many-core processor architecture for both in-vehicle vision applications and media applications including digital TV codec and 3D graphics [3]. Note that as mobile GPUs are also evolving in a similar direction, i.e. increasing programmability and support of thread level parallelism, competition between mobile GPUs, vision processors, and other possible media processor candidates, or further architectural fusion between them, may also be an interesting research field in the near future.

V. Summary and Conclusions

This paper first summarizes the requirements for in-vehicle vision processor designs. Next, the characteristics of vision applications for driver assistance systems, as well as the required computational complexities and the expected date of realization of these applications are predicted. Then, several commercially available in-vehicle vision processor implementations are reviewed, one of the LSI design is reported, and its design approach towards addressing the stated requirements of in-vehicle vision processors is examined. Finally, the relationships between these existing in-vehicle vision processor designs are analyzed. The research trends toward further fulfilling the stated requirements, and the emergence of a unified multi- and many-core processor architecture that can commonly covers digital TV codec, vision applications and 3D graphics, are also predicted.

Acknowledgements

The authors would like to thank Dr. Kuroda of NEC Electronics Corporation, and Dr. Nishiwaki and Mr. Takahashi of NEC Corporation for their valuable inputs.

References

[1] J. Hart, et al.: "Implementation of a 4th-Generation 1.8GHz Dual-Core SPARC V9 Microprocessor", ISSCC Digest of Technical Papers, 10.3, pp. 186-187, 2005.

[2] H.Ikai, M. Usami, M. Ohta, K. Ohue, F. Hidano: “Obstacle Sensor Using Stereo Vision”, *in Proc. of ITS Congress 2006*.

[3] I. Kuroda and S. Kyo: "Media Processing LSI Architectures for Automotives —Challenges and Future Trends—", *IEICE Trans. Electron.*, Vol.E90-C, No.10, Oct. 2007.

[4] Y. Kondo, et al.: "A 4GOPS 3Way-VLIW Image Recognition Processor Based on a Configurable Media-processor.", *ISSCC Digest of Technical Papers*, pp. 148-149, 2001.

[5] S. Kyo, K. Sakurai, S. Okazaki: "A Robust Vehicle Detecting and Tracking System for Wet Weather Conditions Using the IMAP-VISION Image Processing Board", *Proc. of IEEE International Conference on ITS,* pp. 423-428, 1999.

[6] S. Kyo, T. Koga, S. Okazaki, R. Uchida, S. Yoshimoto, I. Kuroda: "A 51.2GOPS Scalable Video Recognition Processor for Intelligent Cruise Control Based on a Linear Array of 128 4-Way VLIW Processing Elements," *ISSCC Digest of Technical Papers*, pp. 48-49, 2003.

[7] S. Kyo, T. Arai, S. Okazaki: "An Integrated Memory Array Processor Architecture for Embedded Image Recognition Systems," *Proc. of the 32nd Annual International Symposium on Computer Architecture (ISCA)*, pp. 132-145, 2005.

[8] S. Kyo, T. Koga, H. Lieske, S. Nomoto, S. Okazaki "A Mixed-Mode Parallel Processor Architecture for Embedded Systems," *Proc. of ACM International Conference on Supercomputing*, pp. 253-262, June 2007

[9] S. Muramatsu, Y. Otsuka, H. Takenaga, Y. Kobayashi, I. Furusawa, T. Monji: "Image Processor Device for Automotive Vision Systems," *IEEE Intelligent Vehicle Symposium*, Vol.1, pp.121–126, 2002.

[10] S. Naffziger, et al.: "The Implementation of a 2-core Multi-Threaded Itanium-Family Processor", ISSCC Digest of Technical Papers, 10.1, pp. 182-183, 2005.

[11] S. Okazaki, S. Kyo, F. Hidano: "IMAPCAR: A Highly Parallel Integrated Memory Array Processor for In Vehicle Image Recognition Applications, " *in Proc. of ITS Congress 2006*.

[12] T. Shiota, et al.: "A 51.2 GOPS 1.0GB/s-DMA Single-Chip Multi-Processor Integrating Quadruple 8-Way VLIW Processors", *ISSCC Digest of Technical Papers*, 10.7, pp. 194-195, 2005.

[13] K. Sakurai, S. Kyo, S. Okazaki: "Implementation of Overtaking Vehicle Detection Using the IMAPCAR Highly Parallel Image Processor", *in Proc. of ITS Congress 2006.*

[14] Z. Sun, G. Bebis, R. Miller: "On-road vehicle detection: a review, " *IEEE Trans. PAMI*, Vol. 28-5, May 2006, pp. 694-711.

[15] J. Tanabe, et.al.: "Visconti: Multi-VLIW Image Recognition Processor based on Configurable Processor", *IEEE Custom Integrated Circuits Conference*, pp.185--188, 2003.