EFFICIENT DATA TRANSFER OPERATIONS FOR A SIMD PROCESSOR ARRAY SYSTEM

Hanno Lieske, Shorin Kyo, Shohei Nomoto, Sunao Torii, Yuki Kobayashi, Yasuyuki Ninomiya, Shinichiro Okazaki

Advanced LSI Systems Research, LSI Research Laboratory, Renesas Electronics Corporation, Japan

ABSTRACT

The SIMD (single instruction, multiple data) control style achieves a very cost effective processor element (PE) control mechanism. In this paper, two efficient data transfer operations for a SIMD PE array processor are proposed to address the inefficiency of most SIMD processor arrays in performing irregular memory access per PE. First, the SIMD random access provides simultaneous data transfer of randomly, from each PE independently addressed elements. Second, the SIMD ROI (region of interest) access enables concurrent data transfer of ROI areas with optional different ROI parameters for each PE. Speed-up values for both new transfer operations show an at least six times faster execution and the synthesis output reports an area increase of merely 1% for a 32 PE SIMD array processor configuration.

Index Terms— Data transfer, SIMD, ROI, Random Access

1. INTRODUCTION

Data processing algorithms can be grouped based on their irregularity. Fig. 1 shows both the classification of algorithms and parallelism exploitation methodologies based on data processing irregularities as well as the proposal goal of this paper which is to expand the efficiently operating SIMD area. In x-direction, the control flow irregularity level increases from left to right side while in y-direction the data access irregularity level increases from down to up side.

The lower left side holds algorithms which have a high control flow and data access regularity, like for example the pixel wise data manipulation on image pixels. These algorithms normally have a large data level parallelism and are best suited for the cost effective SIMD array processors, for which various designs have been proposed [1][2][11].

For the upper left side, despite that the algorithms have regular control flow, due to the irregularity of data access, most existing SIMD array processors remain in-effective because normally most of them are equipped with a DMA (direct memory access) transfer unit for burst transfer of data locating at continuous addresses. SIMT (single instruction multiple threads) [3] architectures extend the SIMD paradigm to address the irregular data access issue based on

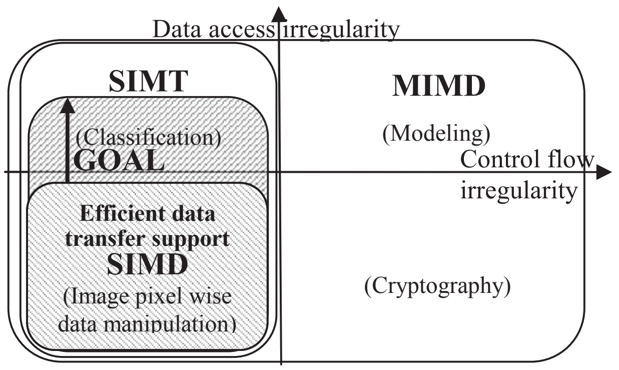
Data access irregularity
SIMT
MIMD
(Classification)
GOAL
(Modeling)
Control flow irregularity
Efficient data transfer support
SIMD
(Image pixel wise data manipulation)
(Cryptography)

Fig. 1 Parallelism exploitation methodology classification based on irregularities and the proposal goal of this paper

splitting the single instruction stream into threads which are processed sequentially for dynamically hiding the latency of irregular data accesses by transferring the data in the background of foreground running threads. The MIMD (multiple instructions, multiple data) processing style in form of MIMD processors [4] or mixed mode SIMD/ MIMD processors [5] is of course also capable of fully exploiting irregular data accesses; in fact, due to the multiple instruction streams of the MIMD processing style, irregularity on data access or control flow level doesn't result in a penalty.

However, increase in hardware cost is normally high to support SIMT processing, due to the need of many additional sets of register files for storing thread contexts. MIMD processors also suffer the disadvantage of a larger die cost and power consumption compared to SIMD architectures. It is shown in [5], e.g., that a MIMD processor element can require four times more logic than a SIMD processor element.

This paper proposes a cost effective way to support background irregular data accesses for SIMD array processors. Specifically, two data transfer operations are proposed, the SIMD random access for single data elements chosen by PE, and the SIMD ROI access for region of data elements chosen by PE.

Due to the support of these two types of irregular data access, the efficiency of using SIMD processors for major classification algorithms is expected to be greatly improved. For example, the region in external memory holding feature or dictionary data for the case of the Viola & Jones

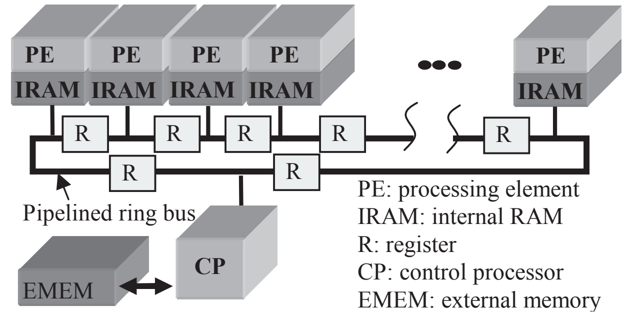

Fig. 2 Example SIMD architecture model block diagram

algorithm [6], which is a frequently used classification algorithms for pedestrian detection [7] or face detection [8], will become efficiently accessible by SIMD processors.

Other works [9][10] face the irregular data load problem by adopting multi-port memories to access data in the external memory (EMEM) for all PE in parallel, which eliminates the EMEM data transfer bottleneck but increases the chip costs due to a higher number of ports. Additionally, the solution in [9] with large multi-port memory has the drawback of large memory area increase, while the solution with small multi-port memory[10] limits the area of usability.

The remainder of this paper is structured as follows. Section 2 gives an overview of a SIMD architecture model. Section 3 presents operation and implementation details of the new proposed data transfer operations, the SIMD ROI and the SIMD random access for SIMD architectures. Section 4 shows the effectiveness of the proposed transfer operations and summarizes synthesis results. Finally section 5 gives a conclusion.

2. SIMD ARCHITECTURE MODEL

A block diagram of the SIMD architecture model is shown in Fig. 2. This architecture model consists of an array of PEs. Each PE has its own internal RAM for local data storage. Hereafter the collection of internal RAM of all PE will be collectively referred to as IMEM. A CP (control processor) is used to broadcast instructions to the PE array in a SIMD style. For data transfer between EMEM and IMEM, normally a pipelined ring bus can be used to implement autonomous DMA burst transfers. The DMA controller located in CP controls the ring bus so that data transfers are performed by cycle-wise shifting data through the register on the ring bus independently with the PE array operation. Hereafter such DMA controller for data transfer between EMEM and IMEM will be specifically referred to as the line transfer (LT) control unit. To simplify the design of the LT control unit, the bit width of a ring bus buffer is chosen as a multiple of the EMEM and IMEM access bit width per cycle.

3. NEW LINE TRANSFER MODES

The key idea for achieving efficient implementation of the SIMD ROI and the SIMD random access is achieved by taking out the control part of these functions from the CP and placing this control part in a separate hardware control unit, the access control unit (Fig. 3), which can operate in parallel to the CP.

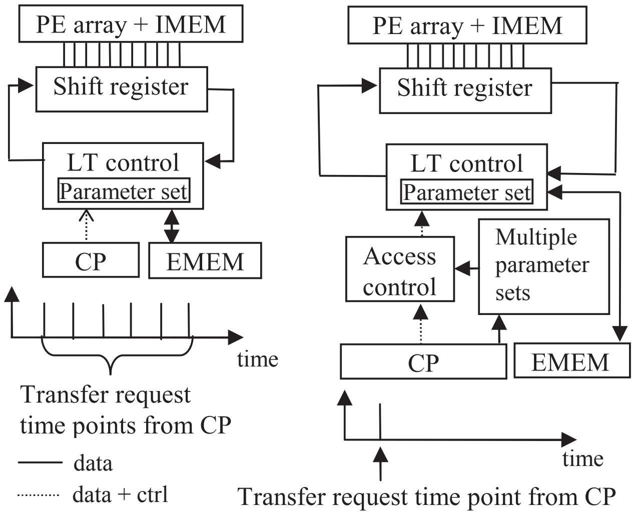

Fig. 3 Conventional (left) and proposed (right) architecture

On the left side a conventional architecture is shown. In such architecture, SIMD ROI or SIMD random data access normally will need to be implemented in foreground by using the CP to issue sequential address calculations and load/store instructions. As the CP also controls the PE array, using CP instructions to perform ROI or random data access decreases the performance of the PE array.

For the proposed architecture on the right side, some small access control logic for support of SIMD ROI and SIMD random access control as well as register for holding multiple parameter sets are added to assist the existing LT control unit. For SIMD ROI and SIMD random access data transfers, first the CP preloads the parameter sets. Then, instead of generating the request addresses inside the CP, the CP passes the transfer control over to the access control logic, which generates the request addresses autonomously. After generation, the single element transfer request addresses are sent to the LT control unit which autonomously transfers the data between EMEM and IMEM.

3.1. Line transfer in ROI mode

The line transfer in ROI mode supports a background operating SIMD ROI transfer, a concurrent transfer of the ROI areas from/to all PE, where each PE specifies the parameters of its ROI area independently from the other PEs.

For the SIMD ROI transfer, two fields have to be prepared in IMEM. The first field holds the ROI parameters, consisting of start address $A_1 \ldots A_N$, width $W_1 \ldots W_N$ and height $H_1 \ldots H_N$, where N represents the number of PE. The second field is reserved for the data. In a first step, the parameter fields of all PE are read from IMEM, transferred

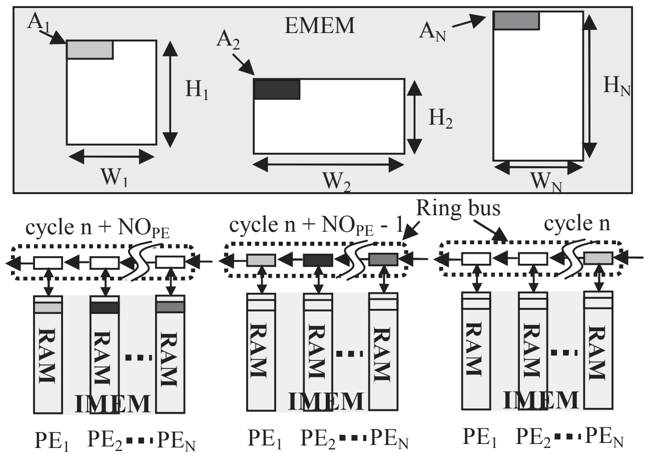

Fig. 4 Line transfer in ROI mode read transfer example

over ring bus and stored inside the LT control unit. In a second step, the start addresses are cycle wise sent to EMEM starting with address A_1 and end up with address A_N.

The addresses ($A_1 \dots A_N$) are repeatedly used as the base address while each time an offset address is also added to produce the real access address for the EMEM. The value of a counter is used as the offset address, which incrementally counts up until the number of time equals to the width specification ($W_1 \dots W_N$). Then, a common row offset parameter specifying the distance between two neighbored rows in EMEM is used to increment the addresses ($A_1 \dots A_N$) so as to update the base addresses to point to the next rows of all ROI areas. For a read access, the received EMEM data is passed to the ring bus. Supposing that the ring bus buffer is A bytes wide and the EMEM allows per cycle B bytes access with $A/B \in IN>0$, the ring bus is every time shifted after A/B EMEM data transfer cycles.

After the data for all PE have been received from EMEM and shifted to the correct position on the ring bus, the data is stored concurrently to IMEM. In Fig. 4, the data transfer operating mode is exemplarily shown for a read transfer where A is equal to B, i.e., element width is equal to bus width. In the initial state, the ring bus is here empty (white colored ring bus buffers). In a first step (right side), one element is loaded from ROI area A1 to the ring bus (gray colored ring bus buffer). Then the data is shifted on the ring bus and the next element is transferred from the next ROI area. In the middle, the state is shown where one element from each ROI area has been transferred to the ring bus. In the following clock cycle, the data is written in parallel from the ring bus buffer to IMEM (left side).

3. 2. Line transfer in random access mode

The line transfer in random access mode operates very similar to the line transfer in ROI mode. It supports a background operating SIMD random access transfer, where on internal side again two fields have to be prepared to hold the random access parameters and the data. The difference is that instead of one parameter set consisting of start address, width and height a number of random access addresses are stored into the IMEM parameter field. Then, instead of transferring the parameter set and then all the data sets, one address for each PE followed by one data element for each PE is transferred in a loop over all data elements.

4. EVALUATION AND SYNTHESIS RESULTS

To show the effectiveness of the new line transfer modes without interference of the chosen EMEM type, example SIMD random access and SIMD ROI transfers have been evaluated without loss of generality for a memory access time of one clock cycle.

For the emulation of the line transfer in random access mode, the fastest implementation is a successive request of single random accesses from CP to EMEM for each PE. Therefore, first one address from each PE has to be transferred from IMEM to CP and after access of the EMEM in case of the read direction, read data elements have to be transferred back through CP to PE before they are stored concurrently from PE into IMEM. The time for the access can be calculated by (1). The outer loop runs over NO_{ELEM_ROW}, while the inner loop runs over the number of PEs. t_{ELEM} defines the time required to transfer one parameter from IMEM to CP and to transfer one data element from EMEM through CP to PE. t_{PE} and $t_{ELEM\_ROW\_I}$ define the time for the invariant part on PE level and data element row level respectively.

For a processor with the proposed line transfer mode, the timing can be calculated by (2). The equation has one loop running over NO_{ELEM_ROW}. The time t_{PARA} defines the time for the parameter transfer from IMEM to CP, which has to be added for each element row transfer. The time t_{ELEM_ROW} defines the time for one element row transfer from EMEM to IMEM and t_{REQ} defines the time for the invariant part of the SIMD random access transfer operation. The clock cycles (cc) for the parameters below have been determined by MODELSIM simulation of the design in [5] with and without new proposed line transfer function. Fig. 5 left side shows the SIMD random access transfer speed-up chart for NO_{PE} equal to 32. The asymptote of the speed-up value S, which is defined by the maximal speed-up S_{max}, is given by (3). For example, S_{max} will be about 6.28 if NO_{PE} is 32.

$$T_{EMU} = t_{ELEM\_ROW\_I} + NO_{ELEM\_ROW} * (t_{PE} + NO_{PE} * t_{ELEM}) \quad (1)$$

$$T_{IMP} = t_{REQ} + NO_{ELEM\_ROW} * (t_{PARA} + t_{ELEM\_ROW}) \quad (2)$$

$$t_{ELEM\_ROW\_I} = 5cc,\ t_{PE} = 7cc,\ t_{ELEM} = 9cc$$

$$t_{REQ} = 9cc,\ t_{PARA} = 10cc;\ t_{ELEM\_ROW} = 37cc$$

$$S_{max} = (t_{PE} + NO_{PE} * t_{ELEM}) / (t_{PARA} + t_{ELEM\_ROW}) \quad (3)$$

For the emulation of the line transfer in ROI mode, the fastest implementation for the read direction is first the word wise direct read access to the elements of each ROI area by the CP, and second transferring the read elements from CP to PE, and finally storing the elements into IMEM. The time for the access can be calculated by (4). The equation has three loops; the outer loop runs over the number of PEs (NO_{PE}), the second loop runs over the number of data element rows inside the ROI area (ROI_V) while the inner loop runs over the number of data elements per row (ROI_H). The variables t_{PE}, t_V, t_H define the time for the invariant part on each level while the time t_{ELEM} defines the time required to transfer one data element from EMEM through CP to PE and to store the data element into IMEM.

For a processor with the proposed line transfer mode, the data transfer time can be calculated by equation (5). The equation holds one loop running over the number of data element rows ($NO_{ELEM\_ROW} = ROI_V * ROI_H$). t_{ELEM_ROW} defines the time for each data element row transfer, while t_{PARA} defines the time for the initial parameter transfer from IMEM to CP and finally t_{REQ} defines the time for the invariant part of the SIMD ROI transfer operation. The clock cycles for the parameters have been determined by MODELSIM simulation of the design in [5] with and without new proposed line transfer function.

Fig. 5 right side shows the SIMD ROI transfer speed-up chart for an example 32PE SIMD processor ($NO_{PE} = 32$). When increasing the size of the ROI area, the speed-up decreases because the invariant initialization parts get less important. The asymptote of the speed-up value S, which id defined by the minimum speed-up S_{min}, is given by (6). For NO_{PE} equal to 32, S_{min} will be about 6.05.

$$T_{EMU} = t_{PE} + NO_{PE} * (t_V + ROI_V * (t_H + ROI_H * t_{ELEM})) \quad (4)$$

$$T_{IMP} = t_{REQ} + t_{PARA} + NO_{ELEM\_ROW} * t_{ELEM\_ROW} \quad (5)$$

$$t_{PE} = 9cc,\ t_V = 11cc,\ t_H = 5cc,\ t_{ELEM} = 7cc$$

$$t_{REQ} = 9cc,\ t_{PARA} = 9cc,\ t_{ELEM\_ROW\ (NOPE = 32)} = 37cc$$

$$S_{min} = (NO_{PE} * t_{ELEM}) / t_{ELEM\_ROW} \quad (6)$$

The equations show that next to the background control for the newly proposed line transfer modes, the transfer time itself can be also reduced by not transferring the data one-by-one but in data element rows with NO_{PE} data elements.

A synthesis has been performed for the design in [5] including the two irregular data access modes by using Synopsys design compiler and a 40nm CMOS library. The synthesis output shows that the design is able to run at 200 MHz and that the area increased by merely 1% for a 32 PE configuration at 200 MHz. Due to the fact, that the additional logic is not lying inside the critical path, a frequency decrease does not occur.

Comparing the proposed DMA transfer modes with a SIMT design [3], the SIMT design is more general purpose in latency hiding because the proposed scheme can only achieve latency hiding if the data access addresses are in advance predictive. But since the irregular access addresses existing in major classification algorithms are predictive, and because a SIMT design will be impossible to implement with little area increase, the proposed DMA transfer modes will be more preferable, especially for cost sensitive SIMD array processor designs in the embedded area.

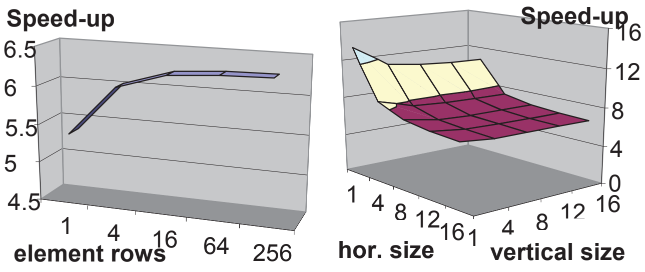

Fig. 5 SIMD random access (left) and SIMD ROI (right) data transfer speed-up for 32 PE

5. CONCLUSION

In this paper, efficient data transfer modes for a SIMD processor array system have been presented. While the line transfer in random access mode offers simultaneous data transfer of addresses, which can be randomly chosen from each PE, the line transfer in ROI mode enables concurrent data transfer of ROI areas with optional different ROI parameters for each PE. Operation details have been presented and evaluation results have been shown for both proposed transfer modes. Speed-up values report for large areas an at least six times faster data transfer execution, while the synthesis output of an example implementation shows an area increase of merely 1% for a 32 PE SIMD array configuration at 200 MHz in a 40nm CMOS process.

REFERENCES

[1] S. Kyo, et al., "A 100 GOPS In-vehicle Vision Processor for Precrash Safety Systems …", *VLSI symposium*, pp.28-29,2008
[2] M. Nakajima, et al., "A 40GOPS 250mW Massively Parallel Processor based on Matrix Architecture", *ISSCC*, 2006
[3] "Technical brief: NVIDIA GeForce GTX 200 GPU Architectural Overview", *TR TB-04044-001 v01*, *NVIDIA Corp.*, May 2008
[4] K. Guttag, et al,"A Single-Chip Multiprocessor for Multimedia: the MVP", *IEEE CG&A*, Vol. 12 No. 6, Nov. 1992, pp.53-64
[5] S. Kyo, et al., "A low-cost mixed-mode parallel processor architecture for embedded systems", *Proceedings of ICS*, 2007
[6] P. Viola and M. Jones, "Robust real-time object detection", *TR CRL 2001/01*, *The Cambridge Research Laboratory*, Feb. 2001
[7] C. Papageorgiou, et al., "A Trainable Pedestrian Detection system", *Intern. Journal of Computer Vision*, pp. 1:15-33, 2000
[8] M. Reuvers, "Face Detection on the INCA$^+$", *Master's Thesis,* University of Amsterdam, 2004
[9] D. Naishlos, et al., "Vectorizing for a SIMdD DSP Architecture", *CASES 2003*, San Jose, California, USA. ACM
[10] H. Chang, et al., "Efficient Vectorization of SIMD Programs with Non-Aligned …", CASES 2008 , USA, 2008, pp. 167-175.
[11] J.P. Wittenburg, et. al., "Realization of a Programmable Parallel DSP for High …", *Proceedings of DAC*, 1998, pp. 56-61.