

# Performance of On-Chip Multiprocessors for Vision Tasks\*

(Summary)

Y. Chung<sup>1</sup>, K. Park<sup>1</sup>, W. Hahn<sup>1</sup>, N. Park<sup>2</sup>, and V. K. Prasanna<sup>2</sup>

<sup>1</sup> Hardware Architecture Team, Electronics and Telecommunications Research Institute,  
P.O.Box 8, Daeduk Science Town, Daejeon, 305-350, Korea  
{yongwha, kyong, wjhan}@computer.etri.re.kr

<sup>2</sup> Department of EE-Systems, EEB-200C, University of Southern California,  
Los Angeles, CA90089-2562, USA  
{neungsoo, prasanna}@halcyon.usc.edu

**Abstract.** Computer vision is a challenging data intensive application. Currently, superscalar architectures dominate the processor marketplace. As more transistors become available on a single chip, the “on-chip multiprocessor” has been proposed as a promising alternative to processors based on the superscalar architecture. This paper examines the performance of vision benchmark tasks on an on-chip multiprocessor. To evaluate the performance, a program-driven simulator and its programming environment were developed. DARPA IU benchmarks were used for evaluation purposes. The benchmark includes integer, floating point, and extensive data movement operations. The simulation results show that the proposed on-chip multiprocessor can exploit thread-level parallelism effectively.

## 1 Introduction

**High performance computing** is needed to satisfy the computational requirements of vision tasks. Several efforts have been directed towards providing parallel processing support for vision. A brief summary of leading research efforts in parallel processing for vision can be found in [1,2]. These efforts can be grouped into three categories, based on the nature of computing platforms they utilize: *special-purpose VLSI processors*[3], *specialized vision systems*[4], and *general-purpose parallel machines*[2].

In this paper, we examine the performance of a collection of vision tasks on an **on-chip multiprocessor**. The performance of general-purpose **Commodity-Off-The-Shelf(COTS)** microprocessors has been improving at a phenomenal rate over the last decade. Currently, the most COTS processors such as *Intel Pentium Series*, *Compaq*

---

\* The work at USC was supported by the DARPA Data Intensive Systems program under contract F33615-99-1-1483 monitored by Wright Patterson Airforce Base.

*Alpha21264, IBM PowerPC620, Sun UltraSparc-2, HP PA8000, and MIPS R10000* use the **superscalar** design technique[5]. Such superscalar processors execute multiple instructions in a single cycle by exploiting **Instruction-Level Parallelism(ILP)**[5]. However, significant speed-ups may not be achieved by using this technique because of the limitation imposed by the instruction window size and available ILP in a typical sequential program[6]. Moreover, considerable design effort is required to develop such high performance processors. Researchers have been studying alternatives to the superscalar architecture[5]. The “on-chip multiprocessor”[7] is one of the alternatives considered as a next generation processor. The key feature of such an architecture is a multiprocessor in a single chip that shares an off-chip 2<sup>nd</sup> level cache to exploit **Thread-Level Parallelism(TLP)**[7], in addition to ILP. For scientific workloads, the on-chip multiprocessor has been shown to overcome the ILP limitation[8]. However, the performance of the on-chip multiprocessor on vision tasks (that have varying computational characteristics) is not known.

To investigate the suitability of the on-chip multiprocessor for vision tasks, we conducted performance studies using a dedicated architectural simulator. It is a program-driven, cycle-level simulator consisting of a Pre-Processing Unit as an instruction simulator and a Post-Processing Unit as a performance simulator. Also, a programming environment was developed to support multithreaded programming on multiple processor cores. Our simulations focused on the performance characteristics of the on-chip multiprocessor including the **Instructions Per Cycle(IPC)** and the total number of **execution cycles** as the number of processor cores is increased. Vision tasks were chosen from the widely studied **DARPA Image Understanding Benchmark**[9]. The simulation results show that “on-chip multiprocessor” is an attractive candidate architecture for vision tasks.

The organization of the paper is as follows. Overview of the vision tasks and the on-chip multiprocessor architecture considered in this paper are given in Section 2 and 3, respectively. In Section 4, the architectural simulator and its programming environment are explained. Simulation results are shown in Section 5, and concluding remarks are made in Section 6.

## 2 Selected Vision Tasks

The vision tasks considered in this paper are selected from the Image Understanding Benchmark[9]. This benchmark performs the recognition of a “mobile” sculpture, given the input images from intensity and range sensors. The benchmark performs low-level operations such as *convolution, thresholding, connected components labeling, edge tracking, median filter, Hough transform, convex hull, and corner detection*. It also performs *grouping* operations and *graph matching* which are representative examples of intermediate-level and high-level processing, respectively. The benchmark utilizes information from two sensors in order to complete the interpretation process. It makes use of both integer and floating-point representations.

The benchmark performs both bottom-up (data-directed) and top-down (knowledge or model-directed) processing. The top-down processing can involve processing of low and intermediate-level data to extract additional features from the data, or can involve control of low and intermediate-level processes to reduce the total amount of computation required.

In the benchmark, the processing begins with low-level operations on the intensity and depth images, followed by grouping operations on the intensity data to extract candidate rectangles. These candidates are used to form partial matches with the stored models. For each of these models multiple hypothetical poses may be established. For each of the pose, stored information is used to probe the depth and intensity images in a top-down manner. Each probe tests a hypothesis for the existence of a rectangle in a given location in the images. Rejection of a hypothesis, which only occurs when there is strong evidence that a rectangle is actually absent, results in the elimination of the corresponding model pose. Confirmation of the hypothesis results in the computation of a match strength for the rectangle, and it also results in the updating of its representation in the model pose with new size, orientation, and position information. After a probe has been performed for every unmatched rectangle in the list of model poses, an average match strength is computed for each pose that has not been eliminated. The model pose with the highest average is selected as the best match. More details of the benchmark can be found in [9].

### 3 On-Chip Multiprocessor

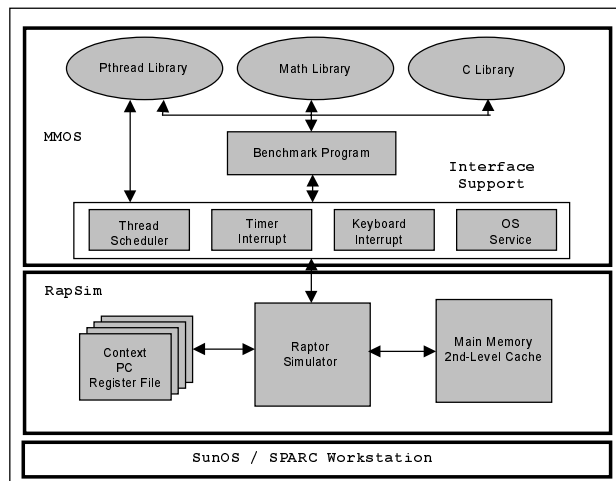
In this paper, we use *Raptor*[10], an on-chip multiprocessor, consisting of four independent processor cores, called General Processor Units(GPUs), and one graphic co-processor, called Graphic Co-processor Unit(GCU). Due to the limited die size, we have chosen four GPUs that are integrated into a single chip. The GCU is shared by four GPUs. Also, in order to control GPUs/GCU and to provide an interface to outside world, additional four component units are included in Raptor namely: Inter-processor Bus Unit(IBU), External Cache Control Unit(ECU), Multiprocessor Control Unit(MCU), and Port Interface Unit(PIU). The IBU is a shared bus connecting the GPUs and the ECU. The MCU distributes the interrupts across the GPUs and provides synchronization resources among the GPUs. The PIU is a mutiprocessor-ready bus interface to communicate with the exterior of the Raptor. The four GPUs execute all instructions except extended graphic instructions with their own register files and program counters, but share the ECU through the IBU. A GPU performs graphic instructions with Single Instruction Stream Multiple Data Stream(SIMD) style and pixel processing hardware. The salient features of Raptor can be summarized as follows:

- Single chip 4-way multiprocessor sharing off-chip 2<sup>nd</sup> level cache
- 64-bit data and 64-bit virtual address
- SPARC V9 Instruction Set Architecture(ISA)
- Extension of graphic instruction set
- Multiple cache structure consisting of on-chip 1<sup>st</sup> level cache and off-chip 2<sup>nd</sup>

- level cache
- Harvard structure of 1<sup>st</sup> level cache consisting of 16 Kbyte instruction cache and 16 Kbyte of data cache
- On-chip 2<sup>nd</sup> level cache controller handling 4 Mbyte of unified off-chip 2<sup>nd</sup> level cache

## 4 Simulation Environment

To evaluate the Raptor quantitatively, we developed a dedicated simulator, called *RapSim*. Also, a programming environment, called *MMOS* (Multithreaded Mini-OS), was developed to support a multithreaded programming on the multiple GPUs. The overall environment of the *RapSim* and the *MMOS* is shown in Fig. 1.



**Fig. 1.** Simulation Environment

The **RapSim** is a program-driven micro architecture simulator that models the four GPUs and a memory hierarchy shared by the four GPUs. The *RapSim* consists of a Pre-Processing Unit and a Post-Processing Unit. The Pre-Processing Unit of the *RapSim* is an instruction set simulator while the Post-Processing Unit is a performance simulator.

The Pre-Processing Unit consists of four components: a processor model for executing instructions, data structures for register files, proxy model for processing system calls, and a model of 1<sup>st</sup> level cache. The Pre-Processing Unit fetches the instructions and the data from the shared memory hierarchy including 2<sup>nd</sup> level cache and executes the instructions, and generates an on-the-fly trace consumed by the Post-Processing Unit. The Pre-Processing Unit starts the simulation by loading a benchmark binary file compiled and statically linked with the *MMOS* library into memory model. During the loading of the benchmark binary, a proper starting program counter is set in the processor model. Trap table and trap handlers are initialized in the memo-

ry model and a stack is constructed in the memory model. Then the processor model executes the instructions using the internal resources like the execution units, register files, and 1<sup>st</sup> level cache.

As the Pre-Processing Unit runs its instruction streams, it generates an on-the-fly trace, a sequence of executed instructions. Each entry of the trace contains enough information so that the Post-Processing Unit can perform the performance simulation using the trace as inputs. The Post-Processing Unit is a RISC pipeline model conducting performance simulation by using the instruction traces generated from the Pre-Processing Unit. It is modeled as a 2-issue **superscalar** including Reservation Stations(RS) and a Reorder Buffer(ROB) to support out-of-order executions. Two instructions in a Trace Buffer are fetched and pre-decoded in a cycle. The pre-decoded instructions in an Instruction Buffer are decoded and issued into proper Reservation Stations(RS), and the Reorder Buffer is updated simultaneously. Each execution unit runs safe instructions from a proper Reservation Station resolving dependency problems.

The **MMOS** provides the RapSim with a multithreaded programming environment to utilize four GPUs efficiently. The MMOS has a Pthread[11] library, C library, and RapSim interface. The C library allows multiple threads to access the shared C library without synchronization problems, whereas the Pthread library provides synchronization and scheduling requirements among multiple threads. The RapSim interface connects the MMOS to the RapSim, and schedules and assigns threads into the processor models of the RapSim. The simulation parameters used in the experiment are listed in Table 1.

**Table 1.** Simulation Parameters

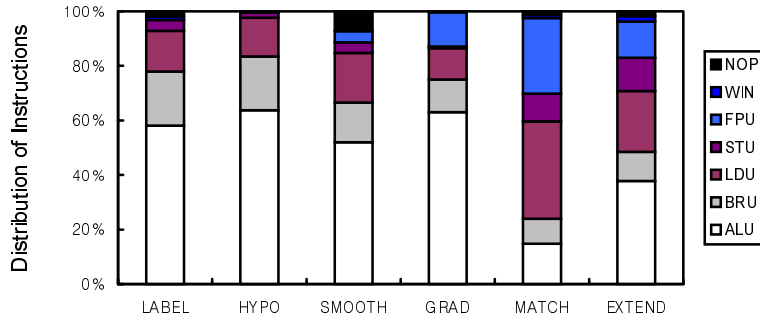
Parameter	Default Value
1 <sup>st</sup> level cache size	16 Kbyte I-cache, 16 Kbyte D-cache / 32 bytes per line
2 <sup>nd</sup> level cache size	4 Mbyte / 32 bytes per line
Write update policy	1 <sup>st</sup> level cache to 2 <sup>nd</sup> level cache : write through 2 <sup>nd</sup> level cache to main memory : write back
1 <sup>st</sup> level cache access latency	1 cycle
2 <sup>nd</sup> level cache access latency	4 cycles
Main memory access latency	10 cycles

## 5 Simulation Results and Analysis

Three sets of simulations were conducted for each vision task described in Section 2. The image size was 512X512. The three sets of simulations were:

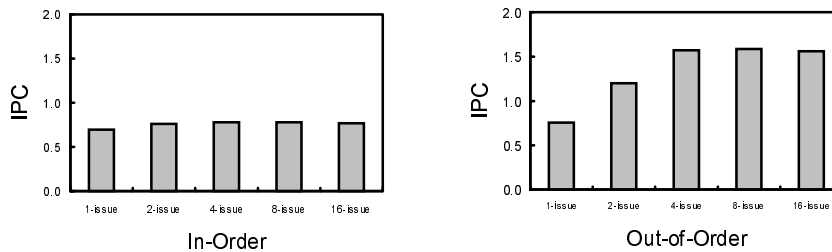
- Sequential: a non-multithreading running on a 1-GPU configuration.
- 2-Threads: 2-way multithreading running on a 2-GPU configuration.
- 4-Threads: 4-way multithreading on a 4-GPU configuration.

The *Instructions Per Cycle(IPC)* and the total number of *execution cycles* were measured as our performance metrics.



**Fig. 2.** Distribution of Instructions Executed on a 1-GPU Configuration

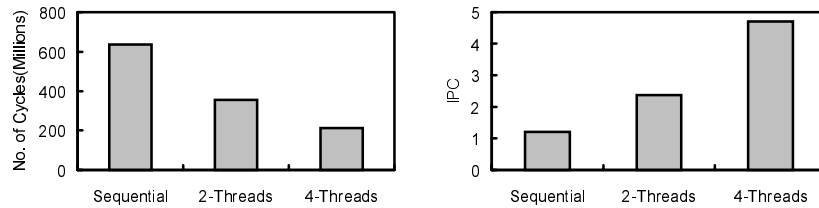
To characterize the computational requirement of each vision task in the object recognition system, we break down the instructions executed on a 1-GPU configuration into seven components, as shown in Fig. 2. In this Fig. 2, ALU, BRU, LDU, STU, FPU, WIN represent ALU, branch, load, store, FPU, window register instructions, respectively. The low/intermediate level tasks(LABEL, HYPO) perform operations on the integer representation of the intensity image. The low level tasks(SMOOTH, GRAD) perform operations on the floating-point representation of the depth image thus the FPU is used in these tasks. However, the high level tasks(MATCH, EXTEND) involve less ALU operations and involve more LDU/STU operations due to the graph traversal (used in object recognition). In addition they also contain more FPU operations due to the probe operations on the depth image data.



**Fig. 3.** Effects of Issue Width and Out-of-Order Execution

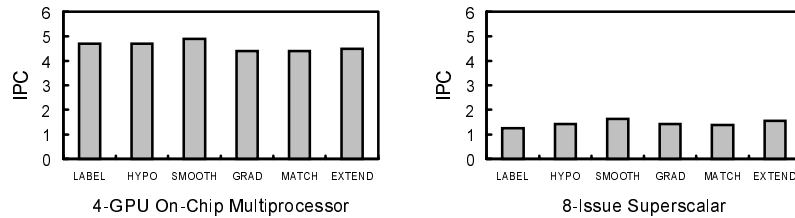
Fig. 3 shows the limitation of ILP on which a typical superscalar architecture relies. It shows the effects of the issue width on IPC with in-order and out-of-order execution. In this experiment, we used the SimpleScalar[12], which is a well-known simulator for superscalar microprocessors. The improvement by increasing the issue width with the in-order execution was negligible, whereas the IPC was increased up to the 4-issue configuration with the out-of-order execution. Overall, the absolute IPC values with out-of-order were larger than those with in-order. However, the IPC obtained by exploiting ILP only, on which typical superscalar architecture relies, was limited to 1.6 even with the 16-issue configuration. Therefore, an additional level of parallelism needs to be considered to satisfy the time performance required by the selected vision tasks.

The execution cycles and the IPCs on multi-GPU configurations are shown in Fig. 4. The effects of exploiting the TLP by increasing the number of the GPUs are analyzed with 2-issue, out-of-order configuration. In this paper, the execution cycle of a multi-GPU configuration is defined as the execution cycle of the GPU running the parent process. As the number of the GPUs increases, the total number of execution cycles drops and the IPC improves by exploiting TLP. That is, the IPCs obtained by exploiting the TLP from the base 2-issue configuration are 2.37 for 2-thread and 4.7 for 4-thread. On the contrary, the IPCs obtained by exploiting the ILP from the base 2-issue configuration are 1.57 for 4-issue and 1.58 for 8-issue, as shown in Fig. 3.



**Fig. 4.** Execution Cycle and IPC of Each Configuration

The IPC of each vision task in the object recognition system is shown in Fig. 5. Even with very different computational characteristics, each task shows similar IPC by exploiting TLP as well as ILP. Also, compared to the IPC on a 8-issue superscalar, TLP can provide 4 times improvement over the ILP-only solution on the average.



**Fig. 5.** IPC of Each Task on a 4-GPU On-Chip Multiprocessor and a 8-issue Superscalar

## 6 Concluding Remarks

The on-chip multiprocessor has been proposed as a promising candidate for a billion-transistor architecture. By integrating simple processor cores, it can exploit TLP (Thread-Level Parallelism) as well as ILP (Instruction-Level Parallelism). In this paper, we have evaluated the suitability of the on-chip multiprocessor for vision tasks. For this evaluation, first we developed a program-driven dedicated architecture simulator and its programming environment. Using simulations, we analyzed the IPC and the total number of execution cycles of the architecture in performing benchmark vision tasks.

Our simulation results showed that more than 4.7 IPC was achieved on the 2-issue, 4-thread on-chip multiprocessor by exploiting TLP as well as ILP. Using ILP only, less than 1.6 IPC was achieved on an 8-issue superscalar architecture. We are currently conducting further evaluation of advanced memory architectures using DARPA DIS benchmarks[13,14].

## Acknowledgement

We would like to thank Puneet Goel for his assistance in preparing the final version of this manuscript.

## References

1. Prasanna Kumar, V.: Parallel Architectures and Algorithms for Image Understanding. Academic Press (1991)
2. Wang, C., Bhat, P., and Prasanna, V.: High Performance Computing for Vision. IEEE Proceedings, Vol. 84, No. 7 (1996) 931-946
3. Annaratone, M., et al.: The Warp Computer: Architecture, Implementation, and Performance. IEEE Tr. Computers, Vol. 36, No. 12 (1987) 1523-1538
4. Weems, C., Riseman, E., and Hanson, A.: Image Understanding Architecture: Exploiting Potential Parallelism in Machine Vision. IEEE Computer, Vol. 25, No. 2 (1992) 65-68
5. Wilson, J.: Challenges and Trends in Processor Design. IEEE Computer, Vol. 30, No. 1 (1997) 39-50
6. Wall, D.: Limits of Instruction Level Parallelism. WRL Research Report, Digital Western Research Laboratory (1993)
7. Hammond, L., et al.: A Single-Chip Multiprocessor. IEEE Computer, Vol. 30, No. 9 (1997) 79-85
8. Singh, J., Weber, W., and Gupta, A.: SPLASH: Stanford Parallel Applications for Shared Memory. Computer Architecture News, Vol. 20, No. 1 (1992) 5-44
9. Weems, C., et al.: The DARPA Image Understanding Benchmark for Parallel Computers. Journal of Parallel and Distributed Computing, Vol. 11, No. 1 (1991) 1-24
10. Park, K., et al.: On-Chip Multiprocessing with Simultaneous Multithreading. Technical Report, ETRI (1999)
11. POSIX P1003.4a: Threads Extension for Portable Operating Systems, IEEE (1994)
12. Burger, D. and Austin, T.: The SimpleScalar Tool Set, Version 2.0. Technical Report, University of Wisconsin (1997)
13. Bondalapati, K., Dutta, D., Narayanan, S., Prasanna, V. K., Ragahavendra, C., and Seshadri, A.: Optimizing DRAM-based Memory System Performance. Submitted to the 27<sup>th</sup> Annual International Symposium on Computer Architecture
14. Musmanno, J. F.: DARPA DIS Benchmarks. Atlantic Aerospace Electronics Corp. (1999)