Parallel Processing Techniques For High Performance Image Processing Applications

Monika Hemnani
Dept. of Electronics
RCOEM
Nagpur, India
hemnanimt1@rknec.edu

***Abstract*—The complexity of many image processing applications and their stringent performance requirements have come to a point where they can no longer meet the real time deadlines, if implemented on conventional architectures based on a single general-purpose processor. Acceleration of these algorithms can be done by parallel computing. Parallelism can be accomplished both at hardware and software levels by various tools and methodologies. The various methods hence discussed prove to be helpful and thus a combination of both the custom hardware and software tool helps in speeding up the image processing algorithm. Different methodologies that can be used for parallel computation are discussed.**

Keywords— High performance computing; parallelism; MPSoC; multicore

I. INTRODUCTION

A variety of parallel architectures are being used today to cope up with the computationally intensive tasks in image processing. Recent embedded multimedia systems, which are complex in design and have a shorter time to market, require more computation power. High Performance Computing (HPC) caters to the ever increasing demand for supercomputing facilities. It is for those tasks which for some reason or another just won't run on a single processor system. The tasks may be sorting through a huge amount of data, solving a difficult numerical problem or repeating the same calculation over and over for different situations. To develop the proper parallel implementations of image processing algorithms, HPC resources which are usually heterogeneous systems, composed of many CPUs and accelerators such as Graphics Processing Units (GPUs) and Field-Programmable Gate Arrays (FPGAs).

The application domains in which parallel and/or distributed image processing is required are: remote sensing, medical image analysis, optical character recognition, biometric systems, product inspection/ sorting in industries, reconnaissance, space image processing, automatic target recognition, robot vision ,etc. Image processing operations used for these applications are feature extraction, noise removal, image orientation, template matching, thermography, object identification, region matching and inference. These image processing operations can be done at three levels. Low level image processing where operators operate at pixel level. Intermediate level image processing where operations are done on abstractions derived from the pixels of the image. High level image processing that is used to interpret the image content [1].

Speed up of operations can be done by either pipelining or parallel processing. Variants of pipelines are: instruction pipeline, arithmetic pipeline and processor pipeline.

II. PARALLELISM IN IMAGE PROCESSING

Performance evaluation of efficient implementation of image processing algorithm can be evaluated on the following criteria: (i) maximal automation during the process (ii) possibility of handling heterogeneous multicomponent architectures, (iii) maximal independence with regards to the architecture, (iv) reduced design time, (v) efficiency of the implementation both in terms of execution time and resource requirements, (vi) enhanced quality and robustness of the final executive. These factors, for HPC, can be achieved by dividing the process on different hardware resources and by bringing parallelism in software.

Different approaches in parallelism are: data, task, instruction and temporal parallelism. Data parallelism is achieved by running different input data sets on processing elements. Skeletons which are algorithmic abstractions common to a series of applications, can be made [2]. These can be implemented in parallel and are very easy to use for a typical image processing user. Task parallelism is achieved by implementing multiple tasks on different cores concurrently. If an instruction that is to be executed, is not dependent on the output of the previous instruction. Then these two independent instructions can be executed parallel, this is instruction parallelism. Temporal parallelism can be exploited by dividing different iterations of an operation on the processing elements. Effectuation of parallelism can be done at hardware and software levels [3], [4].

Parallel processing hardware can be classified into four types based on Flynn's classification of high-speed computers [5]. Single Instruction Single Data (SISD), Single Instruction Multiple Data (SIMD), Multiple Instruction Single Data (MISD), Multiple Instruction Multiple Data (MIMD) [6], [7] as shown in Fig.1, Fig.2, Fig.3 and Fig.4 respectively. Instruction level parallelism and data level parallelism can be achieved by dividing the independent chunks of the codes according to these architectures.

978-1-4673-7918-2/16/$31.00 ©2016 IEEE

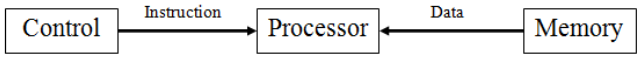

Fig. 1. SISD

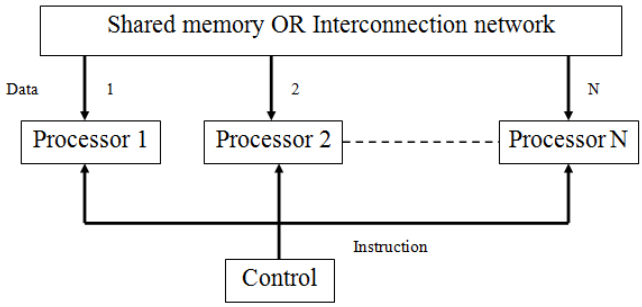

Fig. 2. SIMD

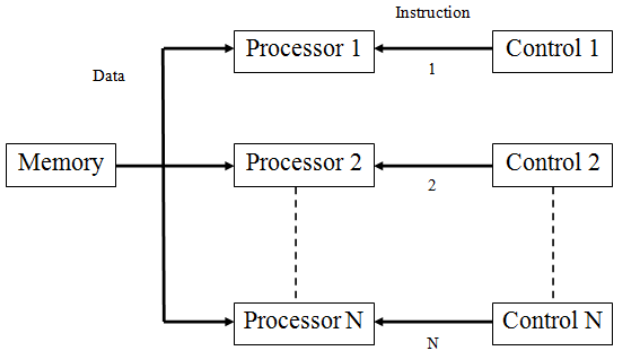

Fig. 3. MISD

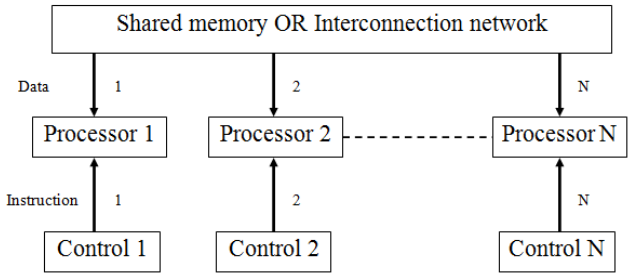

Fig. 4. MIMD

III. HARDWARE

In parallel processing hardware, task is distributed simultaneously on multiple compute resources. Different mechanisms for exploiting parallelism at hardware level are based on:

A. Multi-Processor System On Chip (MPSoC)

A multiprocessor system can be made on a single chip. MPSoCs offer the option to host an application, divided into various tasks, on the different processors on a single chip. Image processing can be partitioned into control and data functions, which can be processed in parallel on different processors. Image processing applications call for complex multiprocessor designs to meet the real time deadlines while overcoming other vital limitations such as low area and power consumption. FPGA based MPSoC platforms, which are made using soft core processors, can provide a quick and convenient way of realizing parallel computing units as required for these applications.

B. Different Topologies of Processors

Based on the algorithm, the processors in the multiprocessor system can be connected in different topologies [8]. Multiprocessor system can be made by softcore processors on FPGAs [9]. Most widely used soft core processors are: Microblaze, Picoblaze, Nios II, Xtensa, OpenRisc and Leon 3. Different variations of topologies are: ring network, star network, completely meshed network, linked star network and hypercube [10]. The master processor sends the group of pixels of images to the slave processors for computation. The slaves give their outputs to the master which combines all the pixels and generates output.

C. Memory Access Methods

In the multiprocessor system on chip, the processors either have a shared memory or distributed memory. The type of data access can be determined according to the need of the algorithm. To reduce the communication overhead single port memories can be used as dual port memory [11]. Also Scratchpad memory (SPM), a software-controlled on-chip memory, can be used in embedded systems as an alternative to hardware-controlled cache due to its advantage in die area, power consumption, and timing predictability [12]. In multi-core systems, SPMs can be accessed by both local core and remote cores. In image processing the operations are performed on masks, instead of sending the whole mask to different processing elements, pointers to those masks can be sent. This would reduce time.

D. Accelerators

Hardwares are faster than softwares, so a dedicated hardware can be designed for computationally intensive software code. This hardware that does acceleration is called as hardware accelerator. An example of hardware accelerator is graphics processor unit (GPU). GPUs are specialized circuits designed to rapidly manipulate and modify memory

for accelerating creation of images in a frame buffer intended for output to a display. GPUs are used in mobile phones, embedded systems, workstations, personal computers and game consoles. Modern GPUs are very proficient in manipulating computer graphics and image processing, and their highly parallel structure makes them more effective than general-purpose CPUs for computationally heavy algorithms.

E. Reconfigurable Computing

Reconfigurable architectures do computation using hardware that can adapt at the logic level to solve specific problems [13]. They have the ability to continually change the functionality of the device. The logic adapts itself based on computation proceeds, based on input and intermediate result. The device needs to store multiple contexts and switch between them. The main processor controls the behavior of the reconfigurable hardware. The reconfigurable hardware is tailored to perform a specific task. Once the task is completed the hardware can be adjusted to do some other task. For the image processing algorithm, the large digital hardware generated on a FPGA can be partitioned into small modules so that each module runs on the target device one after another.

IV. SOFTWARE

Parallelism at software level can be explored by designing of a language/annotation elements to make parallelizable code, exploiting available threading mechanisms for parallelism, automatic parallelization of sequential code and partitioning of image to achieve desired objectives. Various tools and techniques for software parallelism are:

A. Message Passing Interface (MPI)

MPI is a standardised message passing system[14]. It is a language independent communication protocol used for parallel programming. This library has features of portability and scalability[15]. The image that has been processed by one processing element can be given to other processing element by MPI, instead of writing data into memory and the second processing element again reading from the memory. Thus MPI saves time.

B. OpenMP

It is an application programming interface that is used on shared memory multiprocessor system. It comprises of library routines, compiler directives and environment variables which influence the run time behaviour. The programming is done in C, C++ and Fortran and it is supported on most of the operating systems and processor architecture. It uses scalable and portable model which gives simple and flexible interface to develop parallel applications. OpenMP is used for multi-threaded image processing applications to be implemented on multicore CPUs [16]. OpenMP instructs the complier how multithreading of a section of code can be done. The multithreaded image processing code can achieve a good speedup when executed on multiple cores.

C. MATLAB's Parallel Computing Toolbox with MATLAB Distributed Computing Server

Parallel Computing Toolbox in MATLAB helps us to solve computationally intensive problems using multicore processors, GPUs, and computer clusters. High-level constructs, special array types and parallelized numerical algorithms parallelizes MATLAB applications without CUDA or MPI programming. This toolbox can be used with Simulink to run multiple simulations of a model in parallel [17]. The scheduler interface provided by MathWorks parallel computing products is at the software level. The infrastructure allows us to integrate MathWorks parallel computing products with existing scheduler environments at the application layer.

D. PIMA(GE)2

The PIMA (GE) Library is used for Efficient Image Processing in a Grid environment [18]. The major features of this approach are the preservation of the ease in the development of applications and the possibility to efficiently exploit the Grid resources for their executions. PIMA (GE) 2 Grid represents a feasible solution to exploit multiple and computationally intensive image processing applications in a virtual laboratory [19].

E. Compute Unified Device Architecture (CUDA)

NVIDIA CUDA is an API for C, C++, and FORTRAN programming languages. It is developed by NVIDIA for parallel programming on GP-GPUs. It enables programmers to use the NVIDIA graphics card for massively parallel programming [20]. Since GPUs have the ability of executing a huge number of tasks in parallel through CUDA, the programmer is able to execute general purpose computation on GPU cards [21], [22]. Also CUDA has kernel functions which can be called from the host/CPU and that will be executed on the device/GPU. CUDA can provide highly data parallel processing, so it can be used for processing of images having large resolution and size, especially for airplane and satellite pictures. A compiled CUDA program can be executed on any number of processors and only the run time system needs to be given the physical processor count.

F. Thread Building Blocks

Intel Threading Building Blocks helps us easily write parallel C++ programs that take full advantage of multicore performance. These are portable and composable and have future-proof scalability. It is a C++ template library for task parallelism [23]. It has rich feature set for general purpose parallelism. Scalable memory allocation and task scheduling is done in it.

G. Cilk

Cilk is an ANSI C based general-purpose programming language designed for multithreaded parallel computing [24]. The key design principle of Cilk is that the programmers should be concerned with exposing parallelism in the image processing algorithm, while the runtime environment should be concerned with dividing the parallel work among available processors.

TABLE I. SOFTWARE TECHNIQUES APPLIED FOR PARALLEL IMAGE PROCESSING

Techniques	Features
MPI	Portable and scalable library. Language independent.
OpenMP	API for shared memory multiprocessors.
MATLAB's PCT with MCDS	Parallel computation of MATLAB programs. Computer cluster and grid support with MDCS. Support for scheduling.
PIMA(GE)2	Grid environment.
CUDA	API model in Nvidia GPU
TBB	C++ template library for multicore processors
Cilk	General purpose programming language for multithreaded parallel computing.

V. CONCLUSION

An assortment of image processing applications and approaches of image processing applications were discussed in this paper. Brief directions on the tools and techniques for achieving parallelism were provided. The methodologies, which are discussed considering the different aspects, would be helpful in getting concise understanding of parallel processing of high performance image processing applications. Depending on the platform on which the application is made, the corresponding compatible technique is adopted. A combination of custom hardware and optimized software tool would give high performance for the computationally intensive real-time systems.

REFERENCES

[1] R. C. Gonzalez and R. E. Woods, Digital Image Processing (3rd Edition). Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006.

[2] Hanen Chenini, Jean Pierre Dérutin, Romuald Aufrère and Roland Chapuis , "Parallel Embedded Processor Architecture For FPGA-Based Image Processing Using Parallel Software Skeletons," EURASIP Journal on Advances in Signal Processing 2013.

[3] Harshad B. Prajapati and Sanjay K. Vij, "Analytical Study of Parallel and Distributed Image Processing," Proceedings of International Conference on Image Information Processing,IEEE,2011.

[4] H. Nikolov, T. Stefanov and E. Deprettere, "Systematic and Automated Multiprocessor System Design, Programming and Implementation," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 27, no. 3, pp. 542-555, Mar. 2008.

[5] Kai Hwang and Naresh Jotwani, Advanced Computer Architecture, Parallelism, Scalability and Programability. Tat McGraw Hill.

[6] M. Flynn, "Very high-speed computing systems," Proceedings of the IEEE, vol. 54, no. 12, pp. 1901 – 1909, dec. 1966.

[7] M. J. Flynn, "Some computer organizations and their effectiveness," Computers, IEEE Transactions on, vol. C-21, no. 9, pp. 948 –960, sept. 1972.

[8] C. L. Sotiropoulou and S. Nikolaidis, "Design Space Exploration For FPGA-Based Multiprocessing Systems," 17th IEEE International Conference on Electronics, Circuits, and Systems (ICECS), 2010.

[9] Ye Li, Qingming Yao, Bin Tian and Wencong Xu, "Fast Double-Parallel Image Processing Based On FPGA," Vehicular Electronics and Safety (ICVES), 2011 IEEE International Conference on , vol. , no. , pp. 97,102, 10-12 July 2011

[10] P.Huerta, J.Castillo, J.I.Mártinez, andV.López, "Multi MicroBlaze System for Parallel Computing," HW/SW Codesign Group, Universidad Rey Juan Carlos, 28933 Móstoles, Madrid.

[11] S.M.Mehzabeen and I.Manju,"Efficient Optimization Of FPGA On-Chip Memory For Image Processing Algorithm," International Journal of Recent Technology and Engineering, Volume-2, Issue-2, May 2013

[12] Shouzhen Gu, Qingfeng Zhuge, Juan Yi, Jingtong Hu and Edwin Hsing-Mean Sha, "Optimizing Task and Data Assignment on Multi-Core Systems with Multi-Port SPMs," IEEE Transactions On Parallel And Distributed Systems, Vol. 26, No. 9, September 2015.

[13] Chi-Chou Kao, "Performance-Oriented Partitioning for Task Scheduling of Parallel Reconfigurable Architectures," IEEE Transactions On Parallel And Distributed Systems, Vol. 26, No. 3, March 2015.

[14] Philipp Mahr, Christian L¨orchner, Harold Ishebabi and Christophe Bobda, "SoC-MPI: A flexible Message Passing Library for Multiprocessor Systems-on-Chips," International Conference on Reconfigurable Computing and FPGAs, 2008.

[15] W. Gropp, E. Lusk, and A. Skjellum, Using MPI (2nd ed.): portable parallel programming with the message-passing interface. Cambridge, MA, USA: MIT Press, 1999.

[16] G. Slabaugh, R. Boyes, and X. Yang, "Multicore image processing with openmp [applications corner]," Signal Processing Magazine, IEEE, vol. 27, no. 2, pp. 134 –138, march 2010.

[17] G. Sharma and J. Martin, "Matlab: A language for parallel computing," International Journal of Parallel Programming, vol. 37, pp. 3–36, 2009.

[18] PIMA(GE)2. http://www.swmath.org/software/9440.

[19] A. Clematis, D.D' Agostino and A.Galizia, "The Use Of Pima(Ge)2 Library For Efffcient Image Processing In A Grid Environment," Springer.

[20] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable Parallel Programming With Cuda," in ACM SIGGRAPH 2008 classes, ser. SIGGRAPH '08. New York, NY, USA: ACM, 2008, pp. 16:1–16:14.

[21] Z. Yang, Y. Zhu, and Y. Pu, "Parallel Image Processing Based On Cuda," in Computer Science and Software Engineering, 2008 International Conference on, vol. 3, dec. 2008, pp. 198 –201.

[22] Abu Asaduzzaman, Angel Martinez, and Aras Sepehri, "A Time-Efficient Image Processing Algorithm For Multicore/Manycore Parallel Computing," IEEE South Confernce 2015.

[23] C. G. Kim, "Accelerating multimedia applications using Intel threading building blocks on multi-core processors," in Information Science and Applications (ICISA), 2011 International Conference on, april 2011, pp. 1 –7.

[24] R. D. Blumofe, C. F. Joerg, B. C. Kuszmaul, C. E. Leiserson, K. H. Randall, and Y. Zhou, "Cilk: An efficient multithreaded runtime system," Journal of Parallel and Distributed Computing, vol. 37, no. 1, pp. 55–69, Aug. 1996.