Information Technology Journal, 2012
ISSN 1812-5638 / DOI: 10.3923/itj.2012.

Parallel Computing System for Image Intelligent Processing

Zongtao Duan, Tao Lei and Haiwei Fan
Information Engineering School, Chang'an University, Xi'an, 710064, China

Abstract: In this paper, a parallel computing system for image intelligent processing is described. This parallel computing system includes two main parts: A parallel computer and a set of software tools. The parallel computer is constructed by a host processor, a SIMD coprocessor, a stream memory and a controller of this memory. Furthermore, the parallel computer can be extended by different kinds of organization with the host processors and the SIMD coprocessors. Programmers write image intelligent processing programs in a C-like language and a set of software tools maps these programs to code that runs on the parallel computer. These two parts work together to provide a high performance for image intelligent processing.

Key words: Parallel computing, image intelligent processing, data parallel and stream

INTRODUCTION

In this paper, we describe a parallel computing system for image intelligent processing applications development and show how high performance image intelligent processing applications are realized on this system. This system includes a parallel computer and a set of software tools.

The parallel computer is mainly constructed by a host processor, a SIMD coprocessor, a stream memory and a controller of this memory (Shen, 1999). Figure 1 shows the organization of the computer. The host processor is the main controller of the whole system. The stream memory is the memory between main memory and the coprocessor's local register file. Its function is to enhance memory system's speed to match the coprocessor's requirement (Serebrin *et al*., 2002).

Image intelligent processing applications are programmed using a C-like language. This C-like language includes traditional serial statements and data parallel statements. We call the serial statements part as stream programs, the data parallel statements part as kernel programs. The stream programs run on the host processor. The kernel programs run on the SIMD coprocessor. Image data and kernel programs are transferred from main memory to stream memory. Finally, image data arrive at coprocessor's local register file and the kernel programs arrive at coprocessor's program memory. The whole transformation is controlled by the stream memory controller. In the following we will give more details about these.

APPLICATIONS

We want to implement image intelligent processing on this system. Tasks in this kind of image processing can be divided to three levels. They are low level tasks, middle level tasks and high level tasks.

Low level tasks in image processing are also called preprocessing which is used to organize data for later image processing. Preprocessing mainly includes image enhancement and image recovery. In the opinion of computing, preprocessing can be viewed as three kinds of processes that have single pixel depended processing (such as threshold, quantification and coding etc.) part pixels processing (such as filtering, convolution etc.) area pixels processing and whole pixels processing (digital transformation such as FFT). All of the above processing has common characteristics as followings: The first is space inflexibility that is to say the same instruction can be executed on the whole of the image pixels. We call this characteristic data independence. The second is locality i.e., for each pixel its processing result only depends on its neighboring pixels. The third is the fixed data structure i.e., the image processing result is still an image array. From all the above we conclude that the image preprocessing has obvious data parallel.

The aim of middle level processing is to find out the outstanding features and the proper primitive elements from the images. The middle level processing includes image partition and description of image partition. The middle level processing corresponds to a transformation from image to feature. So, middle level processing is a mixture of data parallel and task parallel.

Corresponding Author: Zongtao Duan, Information Engineering School, Chang'an University, Nan Er Huan Zhong Duan, Xi'an 710064, China

The aim of high level processing is to recognize the content of image. This kind of processing can be divided into two periods that are symbol description and scene understanding. From these feature, we can conclude that the high level processing has obvious task parallel.

When a parallel computer system can fit these typical features in image processing, the performance of this system needs to be greatly enhanced. This paper will discuss this kind of image processing system and how to promote its performance.

PARALLEL COMPUTER

The parallel computer includes a host processor, a SIMD coprocessor, a stream memory and a controller of this stream memory as Fig. 1 illustrated.

Parallel computer organization: The host processor is the main controller of the whole system. We call the host processor and the stream memory controller a serial processor. The stream memory controller manages the data transfer between the stream memory and the memory on serial processor. The memory management's function is to enhance memory system's data transfer speed to match the coprocessor's computing requirement.

In Fig. 1, the host processor, the stream memory controller, the stream memory and the SIMD coprocessor are always on a same chip and always the stream memory is constructed by register files.

Parallel computer architecture: The parallel computer is a new generation image processing computer. As Fig. 2 illustrated this computer includes a key component which is called SIMD coprocessor. Always we call the SIMD coprocessor as PE (Processing Element) array. The host of the parallel computer can be a PC or a workstation. The PE array connects with host. The host processor manages the whole computer. The host transfers kernel source programs and data to the PE array's program memory and stream memory. Then the kernel programs are started on the PE array. Finally, the computing results are fetched from stream memory to the host for further analysis. The host also does initialization, diagnosis and testing of the PE array. PE array is suitable to do matrix computing, so we can view the PE array as an accelerator which matches matrix computing.

The PE array is constructed by many processing elements in a regular array. The PE array includes N×N PEs. Each PE includes ALU, local register file, buffer and router which does data communication. All of the PEs are connected by a conventional ring grid. The number of the PEs is decided by the computing requirement and the VLSI realization level.

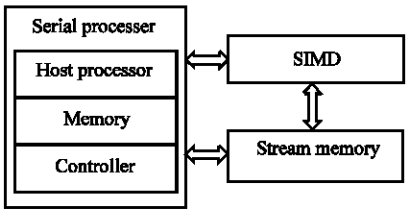

Fig. 1: Parallel computer organization

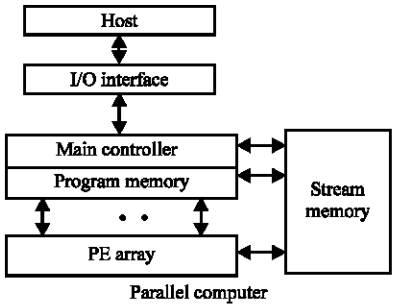

Fig. 2: Parallel computer architecture

Because the PE array is regular and each PE is simple, the PE array chip can be realized in small size. Furthermore, with many PE array chips we can get a parallel computer.

The parallel computer provides a three levels memory hierarchy. This three levels memory is the local register files of PE's, the stream memory and the main memory. This hierarchy is used to exploit the parallelism and locality of image applications.

The architecture of the parallel computer meets the computation and bandwidth demands of image applications by directly processing the naturally occurring data streams within image applications. The PE array is designed to be a coprocessor that operates on image data streams. Many PEs can provide high performance for real-time image processing.

The stream memory effectively isolates the ALUs from the main memory, making the PE array a load/store architecture for data streams. All data stream operations transfer data streams to or from the stream memory. For instance, the PE array transfers data streams directly out of and into the stream memory, isolating PE array transfers from main memory accesses and computation. This simplifies the design of the processor. Meanwhile the stream memory and the PE array can works in pipeline. So data streams transfer speed is promoted by this stream memory.

Parallel computer extension: The above parallel computer is a basic image processing computer. The host processor is a SISD computer. The coprocessor is a SIMD

computer which can be an 8 PEs chip or a 16 PEs chip. We can construct 8 PE chips or 16 PE chip on a board. In this kind of organization we can construct a 64 PEs or 256 PEs in one parallel computer. Furthermore, the 256 PEs on a board (for example a PCI board) can also be constructed in the number of 8, 16 or more to mount on a computer main board. In this kind of extension for the parallel computer we can get a parallel computer which can match the image processing application's performance requirements.

Meanwhile, in the application section we know that image processing application has data parallelism and task parallelism characteristics. In order to construct a parallel computing system to match the need of different parallelism for image processing, we can use the basic image processing computers to construct an intelligent parallel computer for image intelligent processing. For example, use cable to connect several basic parallel computers together we can construct a real task parallel computer which can match three kinds of parallelism for image intelligent processing.

In a word, the extension of the parallel computer can be realized in four types. The first way is to extend the number of PEs on a chip. The second way is to extend the number of PE chips on a board. The third way is to extend the number of boards on a computer main board. The fourth way is to extend the number of computer main boards.

SOFTWARE

The basic image processing computer decomposes an application into a series of kernel programs that operate on image data. We call this kind of image data streams. Streams are sets of sequential data records that lend themselves to high-performance computation through their regular and predictable structure. In simple words stream is a data collection that could be operated in parallel (Buck *et al*., 2004).

Stream programs are written in the SISD statement of a C-like language. The kernel programs are written in the SIMD statement of the C-like language. The use of these two parts in the C-like language for image processing allows the programmer to use the familiar high-level C language's structure (Duan, 2005). The two-level programming system reflects the division of programs on the basic image processing computer: The kernel programs running on the SIMD coprocessor, while the stream program running on the serial processor corresponds to a high-level description of the image processing.

There are several program parts work together to execute an image application, as illustrated in Fig. 3. Stream programs run on the serial processor, call the kernel programs and initiate the transfer of data streams. Stream programs instruct the SIMD coprocessor to run kernel programs and then send out the resulting data (Duan, 2004).

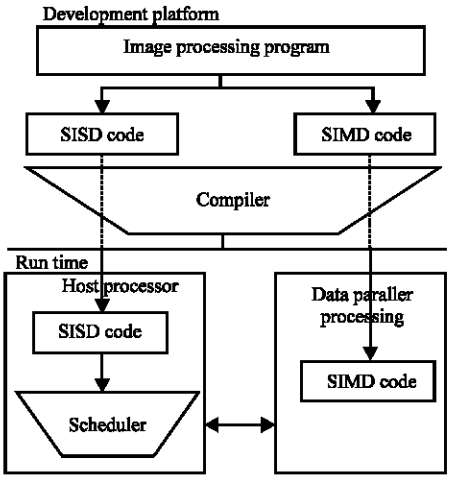

Fig. 3: Software partitions

Stream program and stream scheduler: Stream programs are written in the C-like language with calls into kernel programs. Kernel program calls are treated as functions that take input and output streams as arguments. The stream scheduler allocates space in serial processor's memory and in stream memory used to store streams that are passed between kernel programs. The scheduler determines when to place streams in the stream memory and when to move them to and from serial processor's memory to take advantage of producer-consumer locality between different kernel programs. Efficient stream programs minimize traffic between the stream memory and the serial processor's memory. When the image data is larger than the stream memory, the scheduler breaks the large stream into smaller batches. One batch is loaded into stream memory at a time. The kernel program executes on this batch before computation begins on the next rows. In this way, raw image data arrives in the stream memory and passes between kernel programs through temporary storage in the stream memory. Expensive memory traffic is encountered only when final results are written out.

To ensure high resource utilization, the stream scheduler performs software pipelining at the stream level, where kernel programs are executed concurrently with memory operations.

Kernel program: Kernel programs are user specified programs which are executed over the set of input streams to produce elements onto the set of output streams. Kernel programs operate implicitly over the entire set of input streams. The kernel will execute once for each

element in the input stream (s). The run-time will guarantee that all input streams are the same length. If a kernel is called with input streams of differing lengths, the kernel will immediately fault, producing no output. Kernel programs are declared similar to standard C-functions. An example kernel is shown below:

```
void kernel foo (float a <>, out float b <>, float p)
{
b = a+p;
}
```

The parameter 'a' is the input stream, 'b' is the output stream, 'p' is a constant parameter. The body of the kernel foo will be executed for each input element of 'a' in parallelism, producing a new stream 'b', which contains a set floats which are p larger than the corresponding 'a' elements. The keyword kernel is a function descriptor used to descript that this is a kernel function, not a normal C function.

Kernel functions are called in the same method as normal C function however the stream arguments should be declared streams.

Programmers can distribute several kernels on different PE array chip, so we can get several kernels in parallel. This kind of parallelism can match task parallelism of image intelligent processing.

Parallel pes communication function: The permute operation is used to explicitly interchanges values between PEs. The permute operation looks like permute (PE-permutation, expression).

The permute operation takes a constant describing the permutation and the value to be permuted as arguments. The permutation is an integer in which the nth nibble specifies the index of the PE from which the nth PE gets the value of the permuted variable. For instance, "permute (0x65432107, x)", rotates the value of x in each of the PEs to the left by specifying that PE 7 gets the value from PE 6 and so on.

System working process: Two different types of programs are compiled and loaded on the basic image processing parallel system. Both stream programs and the stream scheduler are compiled for the serial processor with the C-like language. On boot, the serial processor reads a small boot program from boot RAM and executes the full stream programs. Also, the compiler compiles kernel programs for the SIMD coprocessor. The stream scheduler loads compiled kernel programs into the SIMD coprocessor's microcode store.

For the extension parallel computer, the programmer needs to divide the application into task parallelism in advance. The application programmer divides image processing application in task parallelism and data parallelism in the view of image processing's three level processing. For the task parallelism programs, communication can be realized in the function of MPI tools.

CONCLUSION

The research on parallel computing system for the image intelligent processing is popular. There are some research parts for parallel computing system for image intelligent processing being discussed in this article. The system has two functions. First the system can be used as an evaluation platform for parallel system of image processing. Second the system can be used as a development and debugging platform for image processing applications. Through several kinds of image applications' analysis and evaluation we can get some suggestions to improve the parallel computer's architecture which could help us get new parallel system for image intelligent processing with higher speed and higher performance.

FUTURE WORKS

In Fig. 1, we get a serial processor that is a SISD processor. The coprocessor is a SIMD processor. When we deploy the SISD processor and the SIMD coprocessor on a board or a chip, we get a single computer system. When several this kind of boards or chips organize together, then setting another host processor as a management node we can get a total MIMD processor which supports data parallelism, task parallelism and pipeline parallelism. Then the image processing will be greatly promoted. Also we need new software technology to support these kinds of parallelism.

There are many multi core processors. The parallel computer is a kind of multi core processor. Many researches about multi core processor's software components are doing now, such as runtime system, programming language and compiler. These are still a challenge in the parallel computing area.

ACKNOWLEDGMENTS

The authors thank the 863 plans projects of China (2009AA11Z203), Postdoctoral Science Special Foundation funded project (200902581), Specialized Research Fund for the Doctoral Program of Higher Education (200807101004), the Special Fund for Basic Scientific Research of Central Colleges, Chang'an

University (CHD2009JC115) under these foundations the present work was possible.

REFERENCES

Buck, I., T. Foley, D. Horn, J. Sugerman, K. Fatahalian, M. Houston and P. Hanrahan, 2004. Brook for GPUs: Stream computing on graphics hardware. ACM Trans. Graphics, 23: 777-786.

Duan, Z.T., 2004. An embedded image processing platform. J. Inform. Comput. Sci., 1: 41-46.

Serebrin, B., J.D. Owens, C.H. Chen, S.P. Crago and U.J. Kapasi *et al*., 2002. A stream processor development platform. Proceedings of the IEEE International Conference on Computer Design, September 16-18, 2002, Freiburg, Germany, pp: 303-305.

Shen, X.B., 1999. MPP Architecture. In: MPP Embedded Computer Design, Shen, X.B. (Ed.). Tsinghua University Press, China, pp: 250-251.