A Real-Time Message-Passing Synchronization for Multi-SIMD Passively Parallel Machine

Yang Ni and Alain Mérigot
AXIS Institut d'Electronique Fondamentale
Université Paris-Sud
91405 Orsay Cedex France Tél. 69 41 65 72

Introduction

The artificial vision involves both image processing and image analysis. In the image processing, especially in the low level image processing, 2D SIMD parallel architecture can be very efficient. But in the image analysis, hierarchical data structures would be very efficient. With hierarchical representation of images, 2D SIMD parallel paradigm would not be adequate, the pyramid architecture proposed by [1] can accommodate efficiently this hierarchical data structure and speed the image analysis. In a pyramid architecture, the sole SIMD control paradigm is not the most efficient one, the relative independence between different SIMD layers suggests that a MIMD paradigm would be a good solution. When we combine SIMD and MIMD control paradigm in one machine, the major difficulty is the synchronization speed in the MIMD part. SIMD control paradigm dictates often a fine grain parallelism, and MIMD control paradigm dictates often a complexity. So a MIMD-SIMD or Multi-SIMD demands a fast and complex control.

In this paper, we present a real-time synchronization mechanism by message passing. The message passing mechanism can give an universal programming interface to SPHINX. This message passing mechanism is support by special hardwares proposed in following sections, complex interlayer communications can be synchronized as fast as the micro-instruction execution. In the section 2, we will point out the essential difficulties in the Multi-SIMD control paradigm. Then in the section 3, we will analyse the two existing methods, and we illustrate an universal solution with message passing. Finally, we will give some fonctionnel simulation results and conclusion.

The difficulties of Multi-SIMD control in SPHINX

Multi-SIMD control is to synchronize independent SIMD machines to execute a given algorithm. So from the complexity point of view, it is equivalent to MIMD control, besides very often Multi-SIMD control must treat the low level parallelism, that is microinstruction level. An efficient Multi-SIMD control must have an ability to satisfy both the control complexity and the real-time constraint. For SPHINX, the synchronization time must be less than 300ns which is one micro-instruction execution time. If the synchronization time exceeds this limit, the system performance will be heavily degraded.
The interlayer communication is realized by a communication buffer of 1-bit in processing elements, so the processing element has a very limited message buffering ability and this communication register must be seen as a common resource between consecutive layers. The synchronization time is proportional to the size of messages and the communication register can form a contention hot spot and produce possible dead-locks.

Message Passing Synchronization

The early proposed methods are all based on a pipelined interaction between SIMD layers. [3] proposed a cellular automata method. This method uses a quite simple cell with a fixed interacting rule which reflects the synchronization policy. The structure of the cellular synchroniser is depicted in the figure 2. Due to the simplicity, this cellular automata can have multiple evolutions during one micro-instruction, so we can optimise some global shift like operations, as is shown in the figure 3 where a global data shift operation can be executed in one cycle in the place of N cycle. This method can support monodirectional dataflow efficiently in SPHINX, but when some more complex dataflow produces deadlocks, one example is shown in the figure 4. So complex dataflow in SPHINX must be sequentialized into simple monodirectional dataflow with this control mechanism[4]. [5] proposed another cellular synchroniser, as shown in the figure 5, which uses dual FIFO and context switching in the SIMD layers to accommodate up-down dataflow (data flow crossing) without deadlock. But deadlock is still possible in this control mechanism and it is only a partial solution for SPHINX.

The main problem in SPHINX is the interlayer communication. With the shared 1-bit communication register, the synchronization should be intensive during a communication. By example, exchanging one byte needs 8 synchronization operations, when complex dataflow occurs, more synchronization operation would be required. So the dataflow pattern must be limited to a small subset in order to have efficient synchronization; it's to say that the synchronization overhead should not compensate the speedup produced by MIMD parallelism. One of the remedies to this problem is to introduce message-passing communication between consecutive layers. In a message passing system, only one synchronization is required for each message passing, independent of message size. The message passing mechanism can be seen as an universal control paradigm for concurrent systems[6], the main problem of this control mechanism in parallel machine is on the aggregate data structure, such as in the dataflow machines. But in SPHINX, different SIMD layers can be seen as dedicated to aggregate data (data parallelism) and so does the interlayer communication. So the aggregate message can be treated just as scalar message, and message passing mechanism would be optimal in this context. This suggests also a fact that Multi-SIMD architecture could be very good candidate for the dataflow machine.

Implementation of message passing synchronization in SPHINX

Communication via Interlayer Dual-Port Memory

A direct implementation in SPHINX needs two levels of synchronization, one at message object level and the other at bit transfer level, because of the 1-bit communication buffer. So a rendez-vous like synchronization must be used at message object level as in Transputer[7]. The rendez-vous like synchronization is often slow because of bilateral interactions between consecutive layers. If we can introduce some message buffering ability in the processing elements, this rendez-vous like synchronization will not be necessary and we can have some asynchronism at message level as well as at bit transfer level. In our implementation, a dual-port memory is used between consecutive layers. This dual-port memory has double roles: one as a secondary memory for processing elements which have only a 256-bit internal memory and the other as an interlayer communication buffer.
A dual-port memory can provide two independent accesses to the same memory matrix and it’s used for long time to resolve the content problem in communication[8]. The use of dual-port memory can give several advantages:

1. Messages can be buffered before read so bit level synchronization is no longer necessary.
2. Access multiplexing can provide some private communication zones, the communication register is no longer a critical section between different communication partners.

Communication Channel and its Synchronization

With the dual-port memory between consecutive layers, each pair of consecutive SIMD layers can be modeled as a shared memory parallel machine, and all the methods for shared memory parallel machine[9] could be used here. Our method is to use this dual-port memory to simulate a message passing model. In order to simplify the communication synchronization and the buffering zone in the interlayer dual-port memory, we introduce channel concept. A communication channel is a particular zone in the dual-port memory which is shared only by two communication partners in a given direction, it’s to say that sender is always sender and receiver is always receiver. A bilateral interaction can be easily simulated by using two separate channels in the opposite direction, as shown in the figure 6.

The dual-port memory access multiplexing gives an automatic isolation between different channels, the channel synchronization can be simply producer-consumer protocol. Since interlayer communication is synchronous operation in a layer because of SIMD control style, the channel synchronization can be easily realized by using one semaphore variable for each channel. As channel semaphores are only shared by two communication partners in the consecutive layers, access contention to these shared variables can be eliminated by using another dual-port memory to store these semaphores. Different semaphore updating policies[10] can be used, all of these policies can be realized with quite simple hardware and each semaphore operation takes at most two memory access cycles. In SPHINX, we use two kinds of hardwired communication primitives: blocking primitives ("!" and "+") and non-blocking primitives ("!." and "+."). Other communication operators, such as guarded operations, can be easily implemented with these primitives.

Conclusion

In this paper, we have introduced a real-time message passing synchronization method for Multi-SIMD architecture and a concrete implementation in SPHINX - a Multi-SIMD pyramid vision computer. This synchronization mechanism can give both efficient and flexible solution to the Multi-SIMD control problem. The figure shows some functional simulation results of this synchronization mechanism. One is a pyramid sorting algorithm proposed in [11] which has an intrinsic parallelism of 6 in a 11-layer binary pyramid and complex dataflow pattern, the functional simulation indicated that with our message passing control, the majority of code is executed instantaneously at 6 layers, that is the maximal parallelism in the algorithm. The other is histogramming algorithm [2] which is a highly pipelined algorithm, the functional simulation indicated a
highly parallel execution. The hardware implementation of this real-
time message passing synchronization and the development of an
associated programming language will be our future work.

References

[1] V. Cantoni et al., Pyamidal System for Image Processing,
Springer Verlag, 1986.
Pyamidal System for Image Processing, edited by V. Cantoni et al.,