Communication-Computation overlap in massively parallel System on Chip

Hana Krichene, Mouna Baklouti, Mohamed Abid, Philippe Marquet, Jean-Luc Dekeyser

Communication-Computation overlap in massively parallel System on Chip

Hana Krichene
LIFL - INRIA Lille-North Europe labs
University of Lille 1
Lille, France
National School of Engineers of Sfax
CES lab
University of Sfax
Sfax, Tunisia
Email:hana.krichene@inria.fr

Mouna Baklouti
and
Mohamed Abid
National School of Engineers of Sfax
CES lab
University of Sfax
Sfax, Tunisia
Email: mouna.baklouti@enis.rnu.tn
mohamed.abid@enis.rnu.tn

Philippe Marquet
and
Jean-Luc Dekeyser
LIFL - INRIA Lille-North Europe labs
University of Lille 1
Lille, France
Email: Philippe.Marquet@lifl.fr
jean-luc.dekeyser@lifl.fr

***Abstract*—This paper presents the concept and the implementation of a communication-computation overlap in massively parallel System on Chip (mpSoC). This paradigm allows to decrease the execution time of parallel programs using specific strategies in the programming level and partially decoupled system control in the hardware level. The related experiment in VHDL language for system design and in assembly language for case study implementation are described.**

I. Introduction

Many modern application domains are concerned by the conjunction of parallel algorithms and high computing resources. They include signal and image processing applications such as software radio receiver, sonar beam forming, or image encoding/decoding. Furthermore, the implementation of the system on a single chip will be of prime interest for those applications that also require some degree of embeddedness.

Nowadays, SoCs, used for massively parallel computing, consist of many processing units connected by communication networks where running the data to be processed and the results of various intermediate calculations. When it comes to ten thousand of processing units, the network must be efficient to avoid slowing down the whole system. The main complaint of these systems against communication networks is that they consume computation time: they must cut messages into packets in the sender side and reassemble it in order in the receiver side, without error on path. A communicating program on the network will therefore spend time to make these communications, instead of using that time to calculate. Given that the cost of power consumption and area occupation, this mess is unsatisfactory.

Several models of communication-computation overlap, e.g., Message Passing Interface (MPI) [1], Parallel Virtual Machine (PVM) [2], multi-threaded MPI [3] and Hybrid MPI/SMPS [4] were implemented to reduce execution times and improve speedups. MPI and PVM rely on non-blocking elementary communications, asynchronous send and receive primitives as provided by well-known message-passing libraries while the multi-threaded MPI model proposes MPI point-to-point operations in order to overlap communication and computation. Compared to single-threaded active polling communication used by most of the implementations, multi-threaded provides more parallelism. Hybrid MPI/SMPS uses shared memory programming model to introduce asynchrony necessary to overlap communication and computation. If the architecture allows the integration of these high level program software solutions, the speed of parallel applications could double. But they still be limited by the hardware complexity to adopt these approaches.

In this work, we define a new mechanism to amortize the time spent by the communication phase. In fact, processing unit should indicate that it wants to send or receive something, then return directly to his computations while the network would transfer requested. This is called overlapped communications by computation.

The objective of this paper is to reconsider the interest and the feasibility of communication-computation overlap into massively parallel System on Chip. This model allows for communications operations in the background while the program continues to run. It is thus possible to hide a portion of any communication costs.

The next section presents our proposed overlapping communication-computation model. Section 3 discusses the structure implementation and the performance evaluation through the matrix multiplication case study.

II. Communication-Computation overlap model

Distinguish calculation stages of those communication needs the separation of these two stages in different blocks. This repartition should be provided by the designer at programming level. Then, the overlapped execution of these blocks will be done in parallel according to the program description. Separate communication from computation provides an opportunity to ensure the autonomy execution in mpSoC. To ensure this feature while avoiding the centralized control, we define second hierarchical control level, over than the

master controller, achieved by the slaves controllers. We will discuss this structure in the next section.

A. Hardware design: Master-Slave control structure

A novel control structure was proposed for the massively parallel System-on-Chip, referred to as master-slave control structure [5]. Its concept departs from the centralized configuration and instead of a uni-processor master controlling a set of parallel Processing Elements (PEs), the master cooperates with a grid of parallel slave controllers which supervises the activities of cluster of PEs. We define, as shown in fig. 1, the hardware implementation of this configuration in massively parallel system:

- The Master Control Unit (MCU), which controls the order execution in the whole system. It is a simple processor, which fetches and decodes program instruction and broadcasts execution orders to Slave Control Unit. It controls the end execution to establish synchronous communication.
- The Slave Control Unit (SCU), which controls: local node and PEs activities, parallel instructions execution and synchronous communication. It is a crucial component in the master-slave control structure. The SCUs grid allows independent parallel execution.

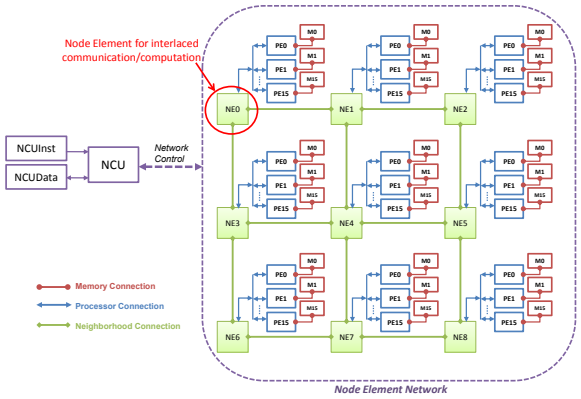

Fig. 1. Hardware prototype of Master-Slave control structure for mpSoC

The hardware architecture is composed of a single MCU and multiple Slave controllers (SCUs) combined with local processing element (PE) (or a cluster of 16 PEs), known collectively as Nodes. The MCU and SCU array are connected through single level hierarchical bus and the SCUs are connected together through X-net interconnection network. This network is clocked synchronously with the SCUs and respectively with the PEs. SCU controllers in the grid care for the instruction execution activities that involve a large degree of parallelism and the communication activities that need to coordinate all the PEs in the grid. Note that the SCU controllers do not limit performance; they do not become a sequential bottleneck. They participate only in the controlling and scheduling of very large groups of PEs execution at a time.

Using master-slave control structure, the PEs in massively parallel system can execute independently and then can communicate synchronously. Separate communication phase from computation one, not only allows the programmer to optimize distinct processors for their intended tasks, but also gives the ability to overlap these two phases which reduces the system execution time.

B. Software program: Master-Slave program management

Based to the decoupled control design presented in the previous section, the programmer can easily manage the master-slave program to overlap communication by computation stage. Therefore, the basic idea to implement this paradigm is to divide the principal program into small blocks of parallel instructions, called Slave Program (SP), and send these blocks to the activated PEs of the system. Then, according to a predefined mask, the SCU sends the order of begin execution. In parallel to computation, the master manages a synchronous inter-node communication, and so on until the end of the program.

To implement this master-slave program we have defined some instructions as shown in fig. 2:

Master side

- **initial.mask**: master controller informs the activated node, using broadcast with mask technique [6], that PEs must start the parallel execution of SP blocs.
- **SP.execution**: slave controller orders the parallel execution of SP blocks. The program in SP blocks is the same for all PEs, but the control flow may be different in each node according to the mask. Each SP block starts with *Begin*-instruction and finishes with *End*-instruction. When SP blocks are executed, the communication is established. GO signal is activated.
- **wait.ortree**: master controller waits for the SP execution end of all the active PEs to start a new parallel cycle. This end execution is controlled by the ortree signal [5].

Slave Side

- **wait.go**: slave PE waits for go signal to start SP execution.
- **end.SP**: slave PE informs the master that the SP execution is finish by sending the end signal.

We require also that the SP block code contains only memory reference of local PE or local register in the relative node and no reference to the global memory or the local memory of the neighbor PE.

Referring to these requirements and specifications, it is possible to perform communication operations in the background while the slave program continues to run, as it is shown in fig. 2. However, This feature is allowed only if the execution time of SP block is higher than time spent for communication phase.

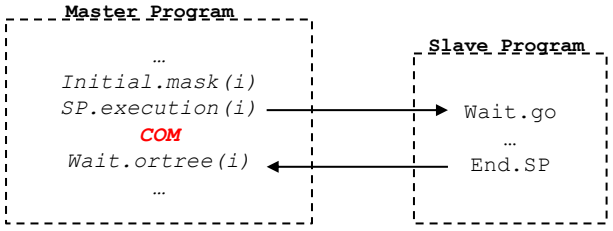

Fig. 2. Computation phase overlaps communication

III. Case study: Matrix multiplication

One of the basic computational kernels in many data parallel codes is the multiplication of two matrices (C=AxB). For this application we have implemented two architectures: the first one is a 64 PEs and the second one is a cluster of a 64 (PE+ muladd_IP). These computation units are arranged in 8x8 node-grid with each PE has a 64bytes data stack . To perform multiplication, all-to-all row and column broadcasts are performed by the nodes. The following codes are executed by the master and all slave-PEs simultaneously (Fig. 3):

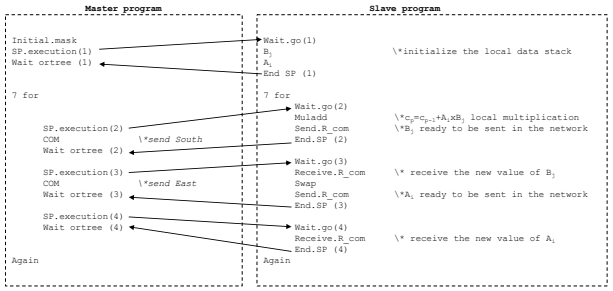

Fig. 3. Overview of matrix multiplication program

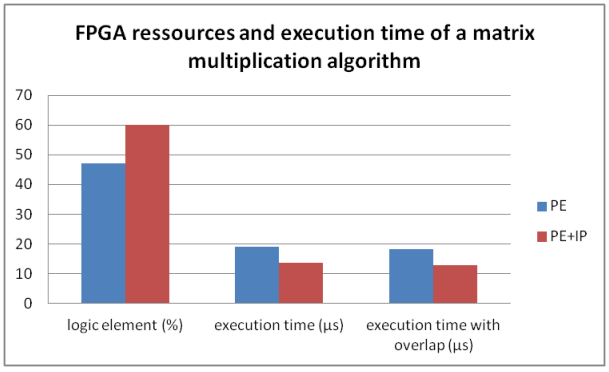

Fig. 4. Experimental results of running a MM algorithm

Fig. 4 shows the Virtex6 ML605 FPGA resource occupation and the execution time results. We validated that the architecture based on PE+muladd_IP is more efficient but occupies a large area on the chip. In fact, the multiplication operation performed in FC16 [7] PE needs 19 clock cycles but only one clock cycle in muladd_IP. However, the space on a chip is increased because the additionally area occupied by muladd_IP other than the PE area. On the other hand, we see that the execution time when using the overlapping communication-computation paradigm is decreased in the both implementations (only PE and PE+muladd_IP). This is due to the fact that communications are done when the processing units execute their locally slave program, so that we introduce a second parallelism level in addition to the massively parallel execution.

IV. Conclusion

This paper presents the paradigm of communication-computation overlap, not only in the programming level but also in the architectural level. The decoupled master-slave control structure allows the independent execution of parallel slave programs and the globally synchronous communication. Separating the computation control from the communication one allows the designer to perform easily programming strategies that overlap these two phases. Using this paradigm, we save around 5% of the globally execution time. It is thus possible to hide a portion of any communication costs.

References

[1] W.Gropp, S.Huss-Lederman, A.Lumsdaine, E-L.Lusk, B.Nitzberg, W.Saphir, and M.Snir, "Mpi: The complete reference," in *MIT Press*.

[2] A.Geist, A.Beguielin, J.Dongarra, W.Jiang, R.Manchek, and V.Sunderam, "Pvm parallel virtual machine: a users guide and tutorial for networked parallel computing," in *MIT Press*.

[3] M.Jiayin, S.Bo, W.Yongwei, and Y.Guangwen, "Overlapping communication and computation in mpi by multithreading," in *Natural Science Foundation of China*.

[4] V.Marjanovic, J.Labarta, E.Ayguade, and M.Valero, "Overlapping communication and computation by using a hybrid mpi/smpss approach," in *24th ACM international confrence on supercomputing*, New York, USA.

[5] H. Krichene, M. Baklouti, P. Marquet, J. L. Dekeyser, and M. Abid, "Master-slave control structure for massively parallel system on chip," in *Euromicro Conference on Digital System Design (DSD2013)*, Santander, Spain, Sep. 2013, pp. 917 – 924.

[6] ——, "Broadcast with mask on a massively parallel processing on a chip," in *High Performance Computing and Simulation (HPCS2012)*, Madrid, Spain, Jul. 2012, pp. 275 – 280.

[7] R. Haskell and D.M.Hanna, "A VHDL forth core for FPGAs," *Microprocessors and Microsystems*, vol. 28, pp. 115 – 125, Apr. 2004.