Scheduling and NoC Traffic Reduction in T-SDF Architecture

Roberto Giorgi, Nikola Puzovic[1]

Dept of Information Engineering, University of Siena, Via Roma 56, 53100 Siena, Italy

ABSTRACT

As transistor size shrinks and chip complexity increases it is possible to place more transistor onto a singe chip, and thus it is possible to integrate more then one processor on a single chip. Clock frequency is also increased, and because of wire delay it is not possible to reach all parts of a chip in a single clock cycle, and interconnection network is becoming a bottleneck in such systems. Our research is focused on creating a multiprocessor on chip architecture based on SDF architecture, by placing multiple processing cores on a single chip, and by interconnecting them to work together. We are investigating possible ways to connect cores, ways in which threads are scheduled on individual cores and ways to reduce network traffic, specially the coherence traffic by using PSCR protocol.

KEYWORDS: multicore chip, memory coherence, NoC traffic reduction

1 Introduction

Our research is focused on expanding SDF architecture [1][2] by placing more than one SDF core on a single chip. In SDF, program is partitioned into non-blocking execution threads and where all memory accesses are decoupled from program execution. Each thread is partitioned into preload phase, execution phase and post-store phase (where results are stored into the memory) in order to decouple memory accesses from execution. By using coarse-grained threads SDF eliminates unnecessary dependencies among the instructions and by using dataflow among threads it eliminates the need for runtime instruction scheduling, which decreases hardware complexity. By keeping the SDF core simple, it is possible to put more cores onto single chip. That arises new design challenges in the areas of interconnecting all cores and scheduling the threads on multiple cores. T-SDF is en evolution of SDF architecture where tiling paradigm is used to address wire delay problem [3]. In this work, scheduling of threads on available cores and traffic reduction on NoC are investigated.

2 Current research trends

Research on NoC is very intensive because it is clear that efficient on-chip communication facilities are necessary in order to overcome clock skew problems and power consumption [4][5][6]. It has been shown [4] that busses can cost-efficiently connect few tens of

[1] {giorgi,puzovic}@dii.unisi.it

components and that point to point links can be used for less number of components that need interconnection, and hence it is shown that no complex router or communication hardware is needed to connect a small number of elements efficiently.
The problem of scheduling parallel tasks with given precedence relationship onto a multiprocessor architecture is known to be NP-complete [7]. Most of the research is done in the filed of priority-based scheduling for real-time systems, which is not of our interests. In the area of tiled architectures, in most cases scheduling is done statically at compile time (TRIPS [8], RAW [9], Synchroscalar [10]), while in other architectures like Wavescalar [11] it is done dynamically, at runtime. Wavescalar is pure dataflow architecture, and it schedules instructions based on the locality so that instruction is executed on execution unit that takes least possible time for it's data to reach it.

3 Our proposal and future work

It has been shown [6] that requirements for connectivity among processors (when averaged on 100 applications) decreases in such a way that, when processor count exceeds 10, each processor requires direct connection with less then 6 processors, and this number decreases as processor count is increasing. Based on those observations, it could be useful to divide the CPU into clusters, where each cluster contains several processing elements together with control logic, data and instruction caches and communication hardware. That will enable processors within a cluster to use a fast synchronous interconnection and to communicate with processors outside cluster by using slower and more complex network.
Based on the optimal size of cluster, which is yet to be determined, interconnection inside a cluster will be implemented by using a shared bus or other fast interconnection network, while communication among clusters will be carried out by the CMP NoC infrastructure.

3.1 Scheduling algorithm

In this execution paradigm, all threads are equal in terms of priority, which makes scheduling easier, and there is no need for implementing complex priority-scheduling algorithms. It is important for overall performance that scheduling algorithm is able to distribute threads across the processors in order to minimize communication costs. Threads that communicate among themselves will be scheduled to execute on one cluster. Generally speaking, scheduling can be performed either in hardware (at runtime), at software (at compile time) or in a combined fashion. When scheduling is performed at compile time, and if program profile is present, very efficient scheduling can be achieved while hardware complexity is kept at minimum (only hardware that is needed is hardware for dispatching threads to cores, and not for scheduling).
Flaw of this approach is that at compile time it is not known if other threads will be executing on some of the threads, so it may happen that a new thread is scheduled on a core that is already occupied or too busy to handle it.
Another approach would be to perform complete scheduling in hardware. Since, there is no priority among threads, only information about the occupancy of processing elements needs to be taken into account when making a decision where to schedule a thread. For that reason, one distributed structure keeps information about number of threads currently executing on each core in a cluster, and at the moment when new thread is issued, it makes a decision where to schedule a thread. Decision could be made by

scheduling a thread to the core inside a cluster that has the least number of threads running on it, and in that case one hardware structure for making the decision is needed in each cluster. Also, decision could be made by using some heuristics or even simple methods like round-robin assignation of threads. In case when there is no possibility to schedule a thread within the same cluster where it was created, it must be scheduled on some other cluster. It could also be interesting to investigate a combined approach where compiler gives hints to the hardware about a set of threads that should be placed onto one cluster, and hardware decides about scheduling those threads (and the ones without a hint) within a cluster (or outside a cluster, if needed).

3.2 Reducing Coherence Traffic

In both cases – software or hardware scheduling – and even with static assignment of threads to core , read/write access to data that is stored in the local memory of another cluster may generate unnecessary coherency traffic on the NoC, since such data may appear as shared. We already successfully experimented a valid solution [12] [13] to address this problem, which allows us to keep the data near the processor where they are needed.

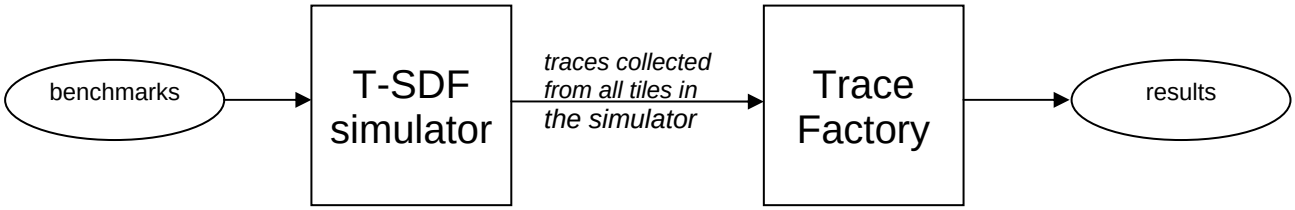

Figure 1 : Infrastructure of the simulation environment

We analyzed the behaviour of PSCR protocol and its effects on network traffic in T-SDF architecture. In order to perform the analysis, we are using simulator for T-SDF architecture in which we are recording all memory accesses, and as a result we get traces from all processors. In architecture that is used for obtaining traces we have expanded the regular SDF architecture into a multiprocessor by connecting several basic SDF processors by a bus. Scheduling of threads is performed in such way that newly enabled thread is assigned to the processor that has the minimal workload (as described above). In this way, we were able to maintain balanced workload on all processors.. Such traces are then passed to Trace Factory [13] simulator that will perform analysis with different coherence protocols in a shared-bus shared-memory multiprocessor environment in order to determine the most appropriate coherence protocol for the architecture (in terms of efficiency and bus traffic). Figure 1 shows the process that is used to obtain traces and to perform simulations. We found, as preliminary result, that PSCR protocol allows us to scale up of a factor 2x in execution time compared to standard solutions such as MESI protocol.

4 References

[1] Krishna M. Kavi, Roberto Giorgi, Joseph Arul, "*Scheduled Dataflow: Execution Paradigm, Architecture, and Performance Evaluation*", *IEEE Trans. Computers*, ISSN:0018-9340, Los Alamitos, CA, USA, vol. 50, no. 8, Aug. 2001, pp. 834-846.

[2] K. Kavi, J. Arul, R. Giorgi, "*Performance Evaluation of a Non-Blocking Multithreaded Architecture for Embedded, Real-Time and DSP Applications*", *14th Int.l Conf. on Parallel and Distributed Computing Systems (ISCA-PDCS-01)*, ISBN:1-880843-39-0, Richardson, TX, USA, Aug. 2001, pp. 365-371.

[3] S.Bartolini, R. Giorgi, E. Martinelli, Z. Popovic, "*Recent Proposals for Tiled Architectures*", ACACES 2005 POSTER ABSTRACTS, L'Aquila, Italy, July 2005, pp. 47-50.

[4] L. Benini and G. De Micheli, "*Networks on chip: a new SOC paradigm*", *IEEE Computer*, pp.70-78, January 2002.

[5] J. Henkely, W. Wolf, S. Chakradhar, "*On-chip networks: A scalable, communication-centric embedded system design paradigm*", Proceedings of the 17th International Conference on VLSI Design (VLSID'04), 2004

[6] N. K. Bambha, S. S. Bhattacharyya "*Joint Application Mapping/Interconnect SynthesisTechniques for Embedded Chip-Scale Multiprocessors*" IEEE Transactions on Parallel and Distributed Systems, Vol 16, No 2, February 2006, pp. 99-112

[7] J. Ullman. *NP-Complete scheduling problems*. Journal of Computing System Science, Volume 10, pages 384 – 393, 1975.

[8] K. Sankaralingam, R. Nagarajan, H. Liu, C. Kim, J. Huh, D. Burger, S. W. Keckler, C. R. Moore, "*Exploiting ILP, TLP and DLP with the Polymorphous TRIPS Architecture*," 30th Int'l Symp. on Computer Architecture, pp. 422-433. ACM Press, June 2003.

[9] M. B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffmann, P. Johnson, Jae-Wook Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, A. Agarwal, "*The Raw Microprocessor: A Computational Fabric for Software Circuits and General Purpose Programs*," IEEE Micro vol.22 Issue 2, pp. 25-35, Mar/Apr 2002.

[10] J. Oliver, R. Rao, P. Sultana, J. Crandall, E. Czemikowski, L. W. Jones IV, D. Franklin, V. Akella, F. T. Chong, "*Synchroscalar: A Multiple Clock Domain, Power-aware, Tile-based Embedded Processor*," 31st Int'l Symp. on Computer Architecture, pp. 150-161. IEEE CS, June 2004.

[11] S. Swanson, K. Michelson, A. Schwerin, M. Oskin, "*WaveScalar*," in the 36th Proc. Int'l Symp. on Microarchitecture (MICRO-36), pp. 291-302. IEEE Press, Dec. 2003.

[12] Roberto Giorgi, Cosimo Antonio Prete, "*PSCR: A Coherence Protocol for Eliminating Passive Sharing in Shared-Bus Shared-Memory Multiprocessors*", *IEEE Trans. Parallel and Distributed Systems, Vol. 10, No. 7*, ISSN:1045-9219, Los Alamitos, CA, USA, July 1999, pp. 742-763.

[13] R. Giorgi, C.A. Prete, G. Prina, L. Ricciardi, "*Trace Factory: Generating Workloads for Trace-Driven Simulation of Shared-Bus Multiprocessors*", IEEE Concurrency, ISSN:1092-3063, Los Alamitos, CA, USA, vol. 5, no. 4, Oct. 1997, pp. 54-68.