A Real-Time Capable Many-Core Model

Stefan Metzlaff, Jörg Mische, Theo Ungerer
Department of Computer Science
University of Augsburg
Augsburg, Germany
{metzlaff,mische,ungerer}@informatik.uni-augsburg.de

***Abstract*—Processors with dozens to hundreds of cores are an important research area in general purpose architectures. We think that the usage of processors with high core numbers will also emerge in embedded real-time computing. Therefore we propose in this paper a concept of a many-core that fits the main requirements for embedded real-time systems: predictability, performance and energy efficiency. At first we enunciate a set of assumptions how these requirements can be met by a many-core: (1) small and simple cores, (2) a cache-less memory hierarchy, (3) a static switched network on chip, and (4) an independent task and network communication analysis. Based upon these assumptions we present a model of a real-time capable many-core. To estimate the architectural parameters of the model we started the development of a cycle accurate simulator. In the current state the simulator is capable of reaching adequate performance for the number of cores.**

***Index Terms*—Many-Core; Real-Time; Processor Architecture**

I. Introduction

In general purpose computing and non real-time embedded application domains multi-core solutions are common, due to their superior performance and reduced power consumption. Recently, multi-cores are emerging in safety critical hard real-time systems, too. As the number of cores increases in general purpose processors and may reach the dimension of hundreds or thousands this trend will also hit embedded computing and – as we suppose – embedded hard real-time (HRT) systems. Those real-time capable systems have to reach several architectural goals: Primarily they must provide a predictable timing to the applications, but a high performance with low energy consumption is also of importance.

We believe that just multiplying the number of common processor cores and connecting them with a best-effort network is not enough to design efficient real-time capable many-cores. The increase of the number of cores beyond 32 requires changes in memory organization, communication interconnect and the programming model such that approaches for real-time capable multi-cores with low core numbers [1] cannot be applied. Therefore we will present a set of assumptions of how such a real-time capable many-core should be designed. Then we combine these assumptions to a many-core architecture model.

The paper is organized as follows: first assumptions are made and motivated for a real-time capable many-core. In Section III the architectural model for such a many-core is proposed. The research topics that are of interest for the real-time capable many-core and need to be addressed are pointed out in Section IV. A simulator for the proposed many-core model is described in Section V. Also a first estimate of the speed of the developed simulator is provided. In Section VI the feasibility of the implementation of a many-core design in hardware is shortly discussed. The paper is concluded with Section VII.

II. Assumptions for a HRT Capable Many-Core

Assumption 1. A small and simple core is the best building block for a real-time capable many-core.

Complex cores with out-of-order execution and several sources of speculation like sophisticated branch predictions speed up the sequential execution and therefore are efficient for single core processors. But parallel execution is the predominant target of many-cores (or should be predominant to achieve high performance) and hence these features are too costly in terms of power and area, compared with only a decent performance increase. In [2] it is stated that in a multi-core the addition of a performance feature to a core makes only sense, if the performance increase is higher than the increase of chip area. Otherwise it is more beneficial (in terms of performance, area and power) to add more cores. Therefore the complexity of the cores should be limited to provide the best performance for a given energy budget.

Also from the perspective of predictability the worst case execution time (WCET) analysis is easier and of higher precision when simple cores are used. Much effort was made to analyze features of complex processors (e. g. for out-of-order pipelines [3] and branch prediction [4]), resulting in a significant increase of analysis complexity when integrating the different analysis steps to reach tighter WCET estimates [5]. To ease the analysis of the tasks running on the cores within the many-core and allow precise analysis without having to take care of possible timing anomalies in the core [6] a simple and predictable core architecture is favorable.

The main focus of a real-time capable many-core is not the core architecture itself, it is the interaction of the tasks running in parallel and the upper-bounding of the interferences that occur by that. Anyhow a core architecture that is easy to analyse is the foundation of a predictable many-core model.

Assumption 2. Distributed memory with the option to access the memory of other nodes provides a predictable memory access and enough bandwidth for embedded applications.

The need for large hierarchical caches in general-purpose multi-cores is due to the memory pressure that is caused by the different applications executed on the cores demanding instructions and data. The WCET analysis of private L1 caches and shared L2 caches in multi/many-cores is possible, but it is very costly in terms of analysis time [7] or requires compiler support to deliver tight estimates [8]. Hence from a WCET analysis point of view a memory hierarchy without caches that consists of one level of fast private memory and a second level of non-cached shared memory not only eases the memory analysis but also reduces a source of its imprecisiveness. Additionally, in many-core systems, maintaining the coherency between caches is an essential but costly issue, therefore a flat cache-less memory hierarchy is also attractive when considering chip area and power consumption.

The downside of a flat cache-less memory model is the reduced scalability due to the fixed memory size per node and the lower throughput. The memory restrictions can be relaxed by borrowing memory from neighbor nodes, using messages over the interconnect to access it. We intend to lean on the programming model of Partitioned Global Address Space (PGAS), which is successfully used in high performance computing to handle the degree of parallelism that many-cores imply. The combination of node number and local memory address naturally spans a global address space to access the memory of neighboring nodes, if the local memory is to small. The exploitation of the memories from other nodes requires the strict definition of the interaction of different tasks. Otherwise the inter-task interferences cannot be tightly upper bounded. Furthermore the usage of non-local memories necessitates an underlying communication network that provides timing guarantees for messages of HRT tasks.

To deal with the other drawback, the slower memory access for a non-local memory, we propose to use part of the local memory as software-managed scratchpad. It is known that such scratchpads can provide a comparable memory throughput like caches with better predictability. Therefore the memory requirements of a task have to be determined and scheduled to the available resources a priori. But this is no cutback for HRT applications, because the resource planning and schedulablity analysis have to be done during system design anyway. We expect that such a memory model can provide the same performance (considering WCET) like a hierarchical cache model and see promising evidence that justifies research.

Assumption 3. A statically scheduled mesh network for HRT tasks targets a predictable timing and a high utilization.

It is commonly known that large core numbers are only possible with tiled architectures for power, scaling, and manufacturing reasons. These tiles are usually connected by a mesh network on chip (NoC). Dynamically routed networks [9] that are inspired from distributed systems can deliver a high utilization of the provided network bandwidth. These networks require the buffering of messages to ease short phases of high congestion. Dynamic routing decisions and the use of large buffers result in complex routers and so a high energy consumption, whereas simple routing decisions with small message buffers are superior from the energy efficiency aspect.

The problem of dynamically routed networks is that the load is hard or even impossible to predict. So from the real-time perspective the communication of two nodes can be affected or even made impossible by the communication of other nodes. Therefore a minimum bandwidth and a maximum latency must be guaranteed for a real-time capable communication.

These requirements can be fulfilled by a time triggered interconnect where certain time slots are reserved for individual point-to-point connections. Such a communication scheme is based on periodic repeated static schedules for the message switching and routing in a network. This *Time Division Multiple Access (TDMA)* technique as e. g. proposed in [10][11] is adequate, because the communication bandwidth of real-time tasks is reserved during system integration. Consequently no interference of unplanned communication can affect the timing characteristics of the communication during runtime. To further increase the predictability of the communication the absence of buffers in the switches [10] should be assumed.

For the support of non real-time communication, unused TDMA slots can be dynamically used to provide mixed criticality [12]. Also the deliberate buffering of non real-time messages can further enhance the network utilization.

Assumption 4. A tight WCET analysis of the system can be reduced to an independent task analysis per core and an analysis of the network communication.

The timing of a task running on a core can be determined with commonly used static analysis techniques of single core processors. The only uncertainty concerns the timing of non-local memory accesses (e. g. for sharing, synchronization, or if the local memory is to small). To solve this the analysis has to use a given bandwidth guarantee and a maximal memory access latency to calculate the WCET. Then a network analysis can build a network communication schedule with the given bandwidth requirements from each task. The schedule is statically defined and guarantees each task the amount of bandwidth by virtual communication channels. Since the distance to the memory that is accessed by a node is affecting the communication latency and the utilization of the chosen path, the placement of the tasks in the network is of importance. So the scheduling of the network communication will guarantee the required bandwidth for each task, otherwise no valid network communication schedule can be build. Thereby the timing of the tasks is left unchanged during integration and it is known that the deadlines of all applications are meet.

Using the PGAS programming model, memory accesses can be divided into three classes: local private, distributed private, and shared accesses. With this division, a simple complexity model of parallel applications can be created [13], which is a first step towards timing analysis. Hence we think that building sound timing models for parallel applications using the PGAS programming model is achievable.

To summarize, the following steps are needed to determine

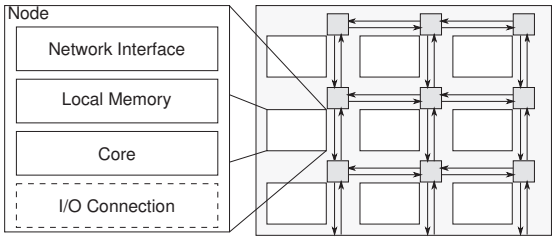

Figure 1. Many-Core Chip

the application's timing: the definition of the necessary communication bandwidth for an application, the timing analysis of each application task, and the network communication scheduling including the task placement.

III. The Many-Core Model

Figure 1 shows a schematic view of a proposed HRT capable many-core structure, based on the four assumptions made above. It consists of cores with simple in-order pipelines with very limited speculation (e. g. static branch prediction). Each node has a limited amount of local memory that holds its code and private data. If the size of the local memory is too small or memory should be shared with other nodes, a node can access the local memory of other nodes. But there is no direct datapath to the distributed memories, instead other memories can only be accessed by messages to other nodes.

To reduce the amount of remote memory a node uses, the local memory can be organized as software-managed scratchpad. This allows a better utilization of the local memory, without using caches. If properly designed, scratchpads can reach a performance comparable with caches, but are easier to analyze and less complex in design. The usage of software-managed scratchpads can also improve performance, because the access of the memory of other nodes is typically slower than the access of the local memory.

External devices (e. g. external memory, bus interfaces, or other I/O devices) are directly connected to a single node, and each node is at most connected to one external device. If for example an off-chip memory is connected as external device to a node, the node will provide a large amount of slow local memory, that can be accessed by other nodes via messages.

We suppose to use a mesh interconnect for the connection of nodes. The routing in the network is statically done by the network communication scheduler that reserves a virtual channel with a dedicated bandwidth for each point-to-point connection.

The proposed many-core model is the minimalistic derivation of the assumptions made before. If it proves right, we will be able to enhance the model while preserving its predictability.

IV. Research Topics

A static WCET analysis is of utmost importance for HRT applications, but a performance analysis of the many-core using a simulator is necessary, if measurement-based WCET analysis should be employed or mixed criticality applications will be supported. With mixed criticality best effort tasks are executed with low priority, but concurrently with HRT tasks and the performance of the non real-time tasks can be measured by using a simulator.

Furthermore we will use a simulator to examine optimal memory sizes, communication bandwidth and latency, instruction throughput, number of ports to external memory, etc., and also to find bottlenecks in the architecture. The base architecture is simple and homogeneous by intention, but only as starting point. We will try to overcome the bottlenecks by introducing more heterogeneity, hence specialized cores (e. g. for floating point calculation, DSP, graphic acceleration, etc.), more complex interconnections, and routing algorithms will be incorporated.

In contrast to the established practice of modifying state-of-the-art hardware with inherited deficiencies, we expect that this bottom-up approach will give a different perspective on many processor resources.

Another research area is the scheduling of the communication within the network to provide each application enough bandwidth to reach their WCET. Therefore the investigation of proper interfaces between the task's WCET analysis requesting a communication bandwidth and the planning of the communication channels in the network is necessary. There is also the need to find solutions that allow the WCET analysis of applications using the memory of different cores. Furthermore to reach a maximum number of applications running on the many-core a proper task placement to keep communication paths short needs to be found.

V. Simulation

To evaluate the many-core model we currently develop a simulator, that is so far used as virtual machine to allow porting applications to the new platform. At present two core architectures are supported, OpenRISC and Infineon Tricore, but we plan to add other to check which architecture fits the requirements of the many-core best.

The periodicity of the TDMA message transportation (Assumption 3) can be used to speed up the simulation. Because the times when messages are send and the times when they arrive are fixed within a static period, the interconnect behavior need not be simulated once every cycle but only once per period. We assume that the length of a period in cycles is given by N.

The simulation can be divided into two alternating phases. During the *execute* phase, a fixed number of cycles N (one period) is simulated for each core. It is assumed that there is no interference between the cores during these N cycles. Therefore this part can be simulated in parallel to gain simulation speed. Afterwards the message exchange between the cores is simulated in the *transport* phase. Because there are lots of interactions between the cores, this phase is to be executed sequentially. Due to the low complexity of the cores and the absence of a memory hierarchy with coherence protocol, the simulation speed of the individual cores is relatively high.

Table I
SIMULATION SPEED DEPENDING ON NUMBER OF SIMULATED CORES

No. of Cores	Host Time [s]	Simulated Cycles per Core and Second
16	262	11 200 000
32	306	9 800 000
64	378	8 200 000
128	575	5 700 000
256	1 591	2 300 000

Given an arbitrary network routing algorithm, a precise simulation of the message timing is only guaranteed for an interval length of 1 cycle. But using such a short interval is bad for the simulation speed on a parallel host machine, as the frequent alternation between parallel and sequential results in a huge overhead. Therefore a large N is preferred in terms of simulation speed and can be chosen due to the periodicity of a time triggered interconnect.

Table I gives a first impression of the simulation speed for an N of 8. A 512×512 integer matrix multiplication was used as benchmark program. The table gives the total time the simulation took on an Intel Core 2 Quad desktop computer at 2.8 GHz and the simulated cycles per core and second.

VI. TECHNICAL DISCUSSION

The TILE-Gx8036 processor from Tilera [14] is a networking processor with 36 cores. Each core has a 32bit three-way VLIW pipeline and 320 kB private cache. The total mesh bandwidth is 66 Tbps at 1.5 GHz clock frequency. This results in a interconnection bandwidth of 1 222 Bits per core and cycle. Tilera plans to increase the core number up to 100. Given these numbers and assuming a comparable core pipeline complexity, in our model a 100 core with 256 kB memory per core and four bidirectional connections of 128 Bits per cycle should be possible with current technology.

For an estimation of the size with FPGA technology, we take the sizes in logic elements (LE) of the Altera Cyclone II family for an OpenRISC processor core (8 700 LE) and a TTNoc switch fabric [10] (440 LE) which should be of similar complexity as the proposed TDMA switch. The latest Altera Stratix 5SGXB6 chip has space for 597 000 LE-equivalents which is enough for about 64 cores with 100 kB each.

The Picochip PC102-100 [15] is a digital signal processor that consists of a so-called picoArray with 240 small cores with 768 bytes of memory and 68 cores with 8.5 or 64 kB of memory. The memory is separated, hence the cores exclusively communicate via a statically scheduled network. It is intended as replacement for FPGAs, with comparable price and energy consumption, but using a simple and familiar ANSI C programming model.

VII. CONCLUSION

We proposed in this paper a real-time capable many-core model based on four assumptions to preserve the timing predictability of the model. The other main design goal is energy efficiency. Upon these assumptions we created an architectural design and started the implementation of a simulator. The paper shows that the simulator is capable of reaching an adequate performance for the number of simulated cores. Furthermore we showed that a many-core consisting of small cores can be implemented in current FPGA technologies.

For further work we will model the timing of the cores in the simulator and evaluate different architectural parameters as e. g. memory sizes and network bandwidth. The resulting architecture is intended to be implemented into an FPGA to provide complexity and the energy consumption estimates for the many-core. In conjunction with the architectural development we will be working on the usage of the distributed shared memory in many-cores for real-time applications, the network scheduling, and the integration of the proposed architecture in an analysis process.

REFERENCES

[1] T. Ungerer, F. Cazorla, P. Sainrat, G. Bernat, Z. Petrov, C. Rochange, E. Quinones, M. Gerdes, M. Paolieri, J. Wolf, H. Cassé, S. Uhrig, I. Guliashvili, M. Houston, F. Kluge, S. Metzlaff, and J. Mische, "Merasa: Multicore execution of hard real-time applications supporting analyzability," *IEEE Micro*, vol. 30, pp. 66–75, 2010.

[2] A. Agarwal and M. Levy, "The kill rule for multicore," in *Proceedings of the 44th annual Design Automation Conference*, ser. DAC '07. New York, NY, USA: ACM, 2007, pp. 750–753.

[3] C. Rochange and P. Sainrat, "A context-parameterized model for static analysis of execution times," in *Transactions on High-Performance Embedded Architectures and Compilers II*, ser. Lecture Notes in Computer Science. Springer Berlin/Heidelberg, 2009, pp. 222–241.

[4] A. Colin and I. Puaut, "Worst case execution time analysis for a processor with branch prediction," *Real-Time Systems*, vol. 18, pp. 249–274, 2000.

[5] R. Heckmann, M. Langenbach, S. Thesing, and R. Wilhelm, "The influence of processor architecture on the design and the results of WCET tools," *Proceedings of the IEEE*, vol. 91, no. 7, pp. 1038–1054, 2003.

[6] T. Lundqvist and P. Stenström, "Timing anomalies in dynamically scheduled microprocessors," in *Real-Time Systems Symposium*. Published by the IEEE Computer Society, 1999, p. 12.

[7] A. Gustavsson, A. Ermedahl, B. Lisper, and P. Pettersson, "Towards WCET Analysis of Multicore Architectures Using UPPAAL," in *10th International Workshop on Worst-Case Execution Time Analysis (WCET 2010)*, vol. 15, Dagstuhl, Germany, 2010, pp. 101–112.

[8] D. Hardy, T. Piquet, and I. Puaut, "Using bypass to tighten wcet estimates for multi-core processors with shared instruction caches," in *Real-Time Systems Symposium*, 2009, pp. 68 –77.

[9] T. Ye, L. Benini, and G. Micheli, "Packetization and routing analysis of on-chip multiprocessor networks," *Journal of Systems Architecture*, vol. 50, no. 2-3, pp. 81–104, 2004.

[10] C. Paukovits and H. Kopetz, "Concepts of switching in the time-triggered network-on-chip," in *Embedded and Real-Time Computing Systems and Applications, 2008. RTCSA '08. 14th IEEE International Conference on*, 2008, pp. 120 –129.

[11] K. Goossens, J. Dielissen, and A. Radulescu, "Aethereal network on chip: concepts, architectures, and implementations," *Design Test of Computers, IEEE*, vol. 22, no. 5, pp. 414 – 421, 2005.

[12] J. Barhorst, T. Belote, P. Binns, J. Hoffman, J. Paunicka, P. Sarathy, J. Scoredos, P. Stanfill, D. Stuart, and R. Urzi, *A Research Agenda for Mixed-Criticality Systems*, Cyber Physical Systems Week 2009 Workshop on Mixed Criticality, San Francisco, CA, USA, 2009.

[13] M. Bakhouya, J. Gaber, and T. El-Ghazawi, "Towards a Complexity Model for Design and Analysis of PGAS-Based Algorithms," in *High Performance Computing and Communications*, ser. Lecture Notes in Computer Science. Springer Berlin/Heidelberg, 2007, pp. 672–682.

[14] *TILE-Gx8036 Processor Specification Brief*, Tilera Corporation, 2011. [Online]. Available: www.tilera.com/tile-gx8000

[15] G. Panesar, D. Towner, A. Duller, A. Gray, and W. Robbins, "Deterministic parallel processing," *Int. J. Parallel Program.*, vol. 34, pp. 323–341, 2006.