A system-level multiprocessor system-on-chip modeling framework

Virk, Kashif Munir; Madsen, Jan

Published in:
International Symposium on System-on-Chip, 2004. Proceedings.

Link to article, DOI:
10.1109/ISSOC.2004.1411154

Publication date:
2004

Document Version
Publisher's PDF, also known as Version of record

A System-level Multiprocessor System-on-Chip Modeling Framework

Kashif Virk   Jan Madsen
Informatics and Mathematical Modeling
Technical University of Denmark
{virk,jan}@imm.dtu.dk

Abstract

We present a system-level modeling framework to model system-on-chips (SoC) consisting of heterogeneous muliprocessors and network-on-chip communication structures in order to enable the developers of todays SoC designs to take advantage of the flexibility and scalability of network-on-chip and rapidly explore high-level design alternatives to meet their system requirements. We present a modeling approach for developing high-level performance models for these SoC designs and outline how this system-level performance analysis capability can be integrated into an overall environment for efficient SoC design. We show how a hand-held multimedia terminal, consisting of JPEG, MP3 and GSM applications, can be modeled as a multiprocessor SoC in our framework.

1 Introduction

Networks on chip (NoC's) are receiving considerable attention as a solution to the interconnect problem in highly-complex chips. The reason is two-fold. First, NoC's help resolve the electrical problems in new deep-submicron technologies, as they structure and manage global wires. At the same time, they share wires, lowering their number and increasing their utilization. NoC's can also be energy-efficient and reliable, and are scalable compared to buses. Second, NoC's also decouple computation from communication, which is essential in managing the design of billion-transistor chips. NoC's achieve this decoupling because they are traditionally designed using protocol stacks, which provide well-defined interfaces separating communication service usage from service implementation. Using networks for on-chip communication when designing systems-on-chip (SoC), however, raises a number of new issues that must be taken into account. This is because, in contrast to existing on-chip interconnects (e.g., buses, switches, or point-to-point wires), where the communicating modules are directly connected, in a NoC, the modules communicate remotely via network nodes. As a result, interconnect arbitration changes from centralized to distributed, and issues like out-of order transactions, higher latencies, and end-to-end flow control must be handled either by the intellectual property block (IP) or by the network.

Multimedia is an increasingly important application area for NoC platforms, in particular, for the new generations of hand-held devices where high-quality audio and video have to be delivered under strict resource and energy constraints. Baiceanu et al. [1] have analyzed the consequences of applying rate-monotonic (RM) scheduling on multimedia applications, i.e. an MPEG player. They argue that the complexity and dynamic behavior of this type of application makes static solutions infeasible and, hence, adaptive methods have to be used. [7] presents a more extensive survey of OS support, and, in particular, scheduling methods for multimedia applications. The presented methods are discussed in the context of basic system requirements for multimedia. In [5], Nieh and Lam present an integrated processor scheduling algorithm for multimedia applications, where both audio and video streams have to be manipulated within well-defined timing requirements, whereas conventional interactive and batch activities still have to be handled. The scheduling algorithm uses two different scheduling policies within the same scheduler, i.e. multimedia tasks are handled by an EDF scheduling algorithm, whereas conventional tasks are scheduled by a Round-Robin scheduling algorithm. The approach of having several scheduling policies within the same scheduler is further explored by Goyal et al. in [3]. They present a framework for hierarchical CPU scheduling in which different scheduling algorithms are employed for different parts of a multimedia application in order to better support the variety of best-effort, hard, and soft real-time characteristics which are typically found in multimedia computing environments. In [6], the scheduling of audio and video multimedia applications is brought to multiprocessor systems. Although a multiprocessor scheduling algorithm has been presented, the network communication latencies have not been taken into account.

In this paper, we present a system-level NoC model, which is an extension of our previous multiprocessor SoC modeling framework [4]. The extended model is able to model heterogeneous multiprocessor architectures interconnected through a an on-chip network architecture, such as a mesh or a torus. We show how a hand-held multimedia terminal, consisting of integrated JPEG encoding and decoding, and MP3 decoding as well as GSM encoding and decoding for the wireless transmission, can be modeled at

0-7803-8558-6/04/$20.00 ©2004 IEEE.

the system-level in our modeling framework.

2 System-Level Modeling

To address the system-level design challenges described above, we need an extended system-on-chip design process, including the effects of the network-on-chip, with the ability to evaluate options and make critical architectural decisions based on a system-level representation in advance of a detailed design. A key pre-requisite is a library of abstract component models that captures their respective performance, power, and physical characteristics.

The primary goal of system-level modeling for embedded systems is to formulate a model within which a broad class of designs can be developed and explored. Moreover, the difficulty of verifying the design of complex systems can be reduced by decomposing a system into smaller subsystems, independently verifying an implementation of the subsystems, and then proving that the composition of the subsystem specifications satisfies the overall system specification. In order to do so, accurate modelling of the system and all the interrelationships among the diverse processors, software processes, physical interfaces and interconnections is needed.

The scheduling problem, central to the analysis of the complexity of concurrent programs, depends on the way in which the scheduled tasks are mapped on the processing elements which, in turn, is linked with the physical architecture of the computing platforms.

A real-time operating system is meant to provide some assurances about the timely performance of tasks. Unfortunately, most mechanisms used in the basic RTOS services are not compositional in nature. Even if a mechanism can provide assurances individually to each task, there is no systematic way to provide assurances for an aggregate of two except in trivial cases.

To support the designers of single chip-based embedded systems, which includes multiprocessor platforms running dedicated RTOS's, we have developed a modeling environment based on SystemC [2, 4]. In our abstract RTOS modeling framework, we deal with generalized abstract tasks, processing elements, and communication infrastructures. For the purposes of modelling, three distinct but closely-related RTOS services have been identified, namely, task scheduling, execution synchronization, and resource allocation.

3 Model Implementation

We have implemented our system-level modeling framework in SystemC. SystemC is in a class of languages that target modeling of hardware and software systems, and it has the desirable feature of being able to simulate models at a very high level of abstraction together with low-level ones. Figure 1 gives an overview of our system-level SoC model, including the processor model and the NoC model which will be described in this section.

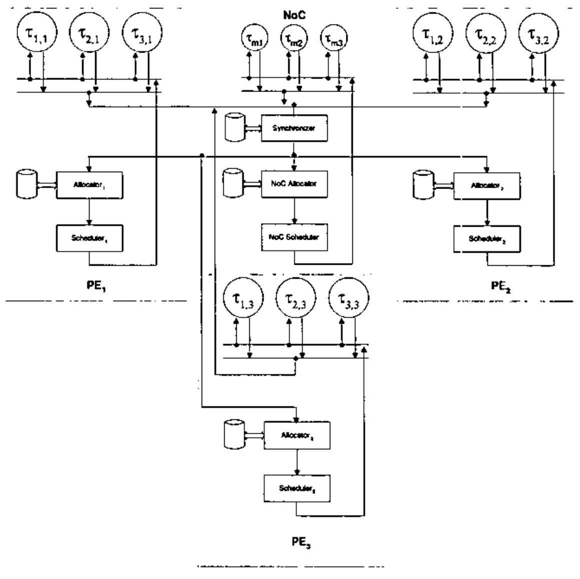

Figure 1. System-level System-on-Chip model

3.1 Abstract RTOS Model

Our abstract RTOS System Model [2] deals with the analysis of the execution behavior of a real-time application running on a heterogeneous multiprocessor platform. In our model, such an application is represented as a multithreaded application comprising a set of tasks where each task, τ, can be decomposed into a sequence of task segments, τ_i. Each task segment, τ_i, is required to precede a given set of other task segments. Moreover, each task segment also excludes a given set of other task segments for the use of shared resources. For each task, we are given a release time, r_k, a release-time offset, o_i, a start time, s_k, a best-case execution time, $bcet_i$, a worst-case execution time, $wcet_i$, a deadline, d_i, a period, T_i, and a context switch time, csw_i. A similar set of parameters can be computed for each task segment, τ_i, relative to the beginning of the task containing that task segment. The multiprocessor platform is modelled as a collection of Processing Elements, PE_k, and Devices, D_k, interconnected by a set of Communication Channels, C_k. Each PE_k is modelled in terms of the RTOS services provided to the tasks comprising the application. Based on the principle of composition, three basic RTOS services are modeled: a scheduler, a synchronizer, and a resource allocator.

The scheduler is modeled around the priority-based preemptive scheduling policy which is one of the most preferred scheduling policies for the execution of tasks in real-time systems due to its higher schedulability. According to our scheduler model, whenever a task becomes ready or finishes execution, the scheduler is called and it then looks for a ready task with maximal priority to continue execution. In our synchronizer model, synchronization is regarded as a means to prevent undesirable task interleavings by the scheduler. Our synchronizer model is responsible for establishing the correctness of the results computed by the multiprocessor platform and it implements the Direct Synchronization (DS) protocol [9].

3.2 Extension of the Abstract RTOS Model to Model NoCs

For the purpose of forming a system-level NoC simulation model, unlike a network simulator, we have abstracted away all the low-level network details except the most essential ones (e.g., topology, latency, etc.). We treat the on-chip communication network as a *communication processor* to reflect the servicing demands. A communication event within this network is modeled as a *message task*, τ_m, executing on the communication processor. When one PE wants to communicate with another PE, a τ_m is fired on the communication processor. Each τ_m represents communication only between two fixed set of predetermined PE's. Since a NoC supports concurrent communication, τ_m's need to be synchronized, allocated resources and scheduled accordingly. This is a property of the underlying NoC implementation, where the NoC allocator reflects the topology and the NoC scheduler reflects the protocol. A resource database, which is unique to each NoC implementation, contains information on all its resources. In a segmented network, these resources are laid-out as two-dimensional interconnects and are a collection of nodes (routers) and links. The NoC allocation and scheduling algorithms map a τ_m onto the available network resources.

- *NoC Allocator:* The allocator translates the path requirements of a τ_m in terms of its resource requirements such as bandwidth, buffers, etc. It attempts to minimize resource conflicts. The links and nodes in a communication path are set aside dynamically (i.e., only for the requested time slot) in the resource database. If the resource reservation process is successful, the message task is queued for scheduling.

- *NoC Scheduler:* The NoC scheduler executes the τ_m's according to the particular network service requirements. It attempts to minimize resource occupancy. In a network, resource occupation is dictated by the size of the message.

4 Hand-held Multimedia Terminal

In this section, we will demonstrate the capabilities of our system-level modeling framework by presenting the simulation results of a multiprocessor SoC-based multimedia device which concurrently runs JPEG encoding/decoding, MP3 decoding, and GSM encoding/decoding all in real-time. Figure 2 shows the five task graphs which are defining the core functionality of our multimedia device. The pre-processing steps for abstracting the application code, like the extraction of the static task graph parameters through code profiling, and mapping the task graphs to the NoC architectures have been performed manually [8]. For the purpose of demonstrating the capabilities of our modeling framework, the applications have been mapped on four processing elements (see Figure 3), three fast processors (25 MHz), and one slow processor (10 MHz). Each of the four processors has its own local memory and all the four processors are interconnected by a torus network. Using distributed memory for instructions and data greatly reduces the traffic in the network.

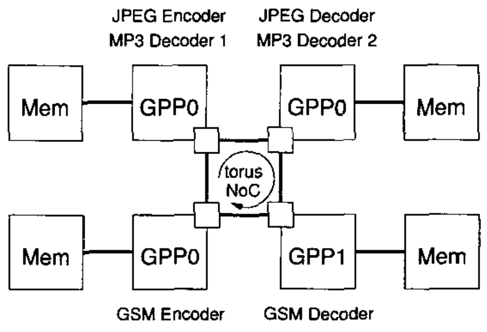

Figure 3. Multiprocessor architecture for the multimedia application.

The MP3 decoder is the most critical multimedia application and mapping its task graph on a single processor, even on a fast processor, reveals that some tasks miss their deadlines. Therefore, the MP3 application task graph has been partitioned and mapped on two fast processors which, as mentioned above, are interconnected through a NoC. The JPEG encoder and decoder applications are mapped to the same two fast processors as the MP3 decoder, whereas the GSM encoder is mapped onto a third fast processor and the GSM decoder is mapped on a slow processor. This mapping results in the exchange of communication messages between the two fast processors over the NoC.

In order to illustrate the capabilities of our modeling framework, we are using two different schedulers. RM scheduling is used on the two fast processors to handle JPEG and MP3, whereas the two GSM applications are scheduled using EDF scheduling. Table 1 summarizes the characteristics of the multimedia application.

5. Conclusions

We have presented a system-level, system-on-chip modeling framework and discussed how our original SoC model has been extended to handle the effects of the on-chip interconnection infrastructure, i.e., the network-on-chip. We have demonstrated the capabilities of our modeling framework by modeling and simulating a hand-held multimedia terminal application mapped on a heterogeneous 4-processor SoC architecture interconnected through a torus

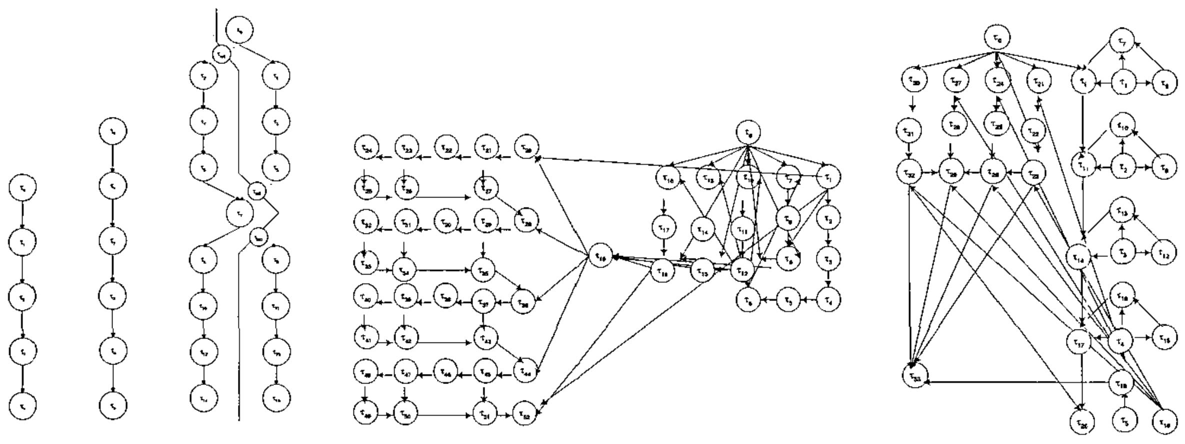

Figure 2. **The five task graphs corresponding to the multimedia applications. From left, these are the JPEG Encoder, JPEG Decoder, MP3 Decoder, GSM Encoder and the GSM Decoder.**

Application Type	Number of Tasks	Deadline	Processor	Clock Frequency	Scheduler
JPEG Encoder	5	250ms	GPP0	25MHz	Rate Monotonic
JPEG Decoder	6	500ms	GPP0	25MHz	Rate Monotonic
MP3 Decoder	16	25ms	GPP0	25MHz	Rate Monotonic
GSM Encoder	53	20ms	GPP0	25MHz	Earliest Deadline First
GSM Decoder	34	20ms	GPP1	10MHz	Earliest Deadline First

Table 1. **Parameters for the multimedia applications**

on-chip network topology. It is worth mentioning, however, that our system-level modeling framework supports more sophisticated scheduling policies and NoC topologies. Moreover, features like including the effects of the network interface and memory accesses as well as dynamic load balancing support can be built upon by adding more components to the existing framework components. We are currently extending our modeling framework to include radio and transducer components in order to be able to model wireless sensor networks, i.e., a distributed system of SoCs.

Acknowledgements

The work presented in this paper has been funded by the SoC Mobinet Project (IST 2000-30094).

References

[1] V. Baiceanu, C. Cowan, D. McNamee, C. Pu, and J. Walpole. Multimedia Applications Require Adaptive CPU Scheduling. In *Proceedings of the Workshop on Resource Allocation Problems in Multimedia Systems*, December 1996.

[2] M. Gonzalez and J. Madsen. Abstract RTOS Modeling in SystemC. In *Proceedings of the 20th IEEE NORCHIP Conference*, pages 43 – 49, November 2002.

[3] P. Goyal, X. Guo, and H. Vin. A Hierarchical CPU Scheduler for Multimedia Operating Systems. In *Proceedings of the Second Symposium on Operating System Designs and Implementations (OSDI'96)*, pages 107–122, October 1996.

[4] J. Madsen, K. Virk, and M. Gonzalez. Abstract RTOS Modelling for Multiprocessor System-on-Chip. In *International Symposium on System-on-Chip*, pages 147–150, November 2003.

[5] J. Nieh and M. S. Lam. Integrated Processor Scheduling for Multimedia. In *Proceedings of the Fifth International Workshop on Network and Operating System Support for Digital Audio and Video*, July 1995.

[6] J. Nieh and M. S. Lam. Multimedia on Multiprocessors: Where's the OS When You Really Need It? In *Proceedings of the Eighth International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 103–106, July 1998.

[7] J. Regehr, M. B. Jones, and J. A. Stankovic. Operating System Support for Multimedia: The Programming Model Matters. Technical report, Microsoft Research, September 2000.

[8] M. Schmitz, B. Al-Hashimi, and P. Eles. A Co-Design Methodology for Energy-Efficient, Multi-Mode Embedded Systems with the consideration of Mode Execution Probabilities. In *Design Automation and Test in Europe, DATE*, pages 960–965, March 2003.

[9] J. Sun and J. Liu. Synchronization Protocols in Distributed Real-Time Systems. In *Proceedings of the 16th International Conference on Distributed Computing Systems*, pages 38–45, May 1996.