An Integrated Framework for Model-Based Design and Analysis of Automotive Multi-Core Systems

Khalid Latif, Charles Emmanuel Effiong, Abdoulaye Gamatié, Gilles Sassatelli, Leonardo B. Zordan, Luciano Ost, Piotr Dziurzanski, Leandro Soares Indrusiak

An Integrated Framework for Model-Based Design and Analysis of Automotive Multi-Core Systems

Khalid Latif*, Charles Effiong*, Abdoulaye Gamatié*, Gilles Sassatelli*, Leonardo Zordan*, Luciano Ost*, Piotr Dziurzanski† and Leandro Indrusiak†
*LIRMM (CNRS and University of Montpellier), Montpellier, France. Email: firstname.lastname@lirmm.fr
†Real-Time Systems Group, Computer Science Department, University of York. Email: firstname.lastname@york.ac.uk

***Abstract*—With technological advances, significant changes occur in automotive domain. Modern automobile combines various functionalities, ranging from safety critical functions, e.g. control systems for engine and break, to navigation and infotainment. The latter ones are especially performance-demanding. To meet their requirements, automotive industry is increasingly turning to multi-core systems. As a result, the design complexity gets increased. This paper presents a design exploration framework relying on an automotive-specific design environment named Amalthea. Applications are modeled in Amalthea as annotated hierarchical task graphs. Their simulation on multi-core platforms considers instruction-level core models interconnected by a packet-based communication network. The effectiveness of our framework is demonstrated on a representative case study.**

***Keywords*—Automotive applications, design methodology, multicore processing**

I. Introduction

The increasing use of electronic systems in automotive industry is making automobiles safer, light weight, secure and comfortable. The key driving factors for evolution of electronic systems in automobiles are the demand for advanced driver assistance systems, active and passive safety systems and the fulfillment of environmental and relevant legal requirements. This evolution has turned the automobiles into very complex high-technology products in recent years, and the trend does not seem to decline in the same way with time [5].

In modern automobiles, computing functionalities can be roughly classified as: soft real-time (fuel, and power indicators), hard real-time (airbags and breaks), and infotainment. The immediate challenge for designing a system, which can fulfill such diverse requirements, is to find an optimal combination of design tradeoffs regarding resource allocation, job scheduling, and communication interconnection that fit well the application performance and real-time requirements [9]. Additionally, due to the computational limitations of single-core systems, automotive industry is increasingly turning to multi-core systems for the deployment of its electronic control units [1], and thus increasing the hardware/software partitioning, resource allocation and scheduling complexity.

This requires an approach where thorough design space exploration can be quickly achieved to decide between different design alternatives. Then, overall design productivity is increased dramatically. Usual performance estimation techniques often employ two types of models, namely simulation-based and analytical [10]. The inherent complexity of modern platforms boosts the simulation time of simulation-based models which makes it unfeasible for vast design space exploration. This motivates the use of computationally simple models which incorporate the system structure to obtain mathematically solvable models. Analytical models have adopted this approach to address the aforementioned issues.

At earlier design stage, it is not possible to test all design combinations with high accuracy, because the detailed simulations are very slow. Thus, it is an immediate requirement by the designers to have a fast yet reasonably accurate simulation and evaluation platform, specifically for automotive domain. There are several frameworks devoted to multicore system design exploration [7] [6] [11] sharing similar goals as our proposal. They provide different trade-offs between simulation speed and accuracy. However, our approach is specifically well-adapted to automotive applications designed within the model-based open-source development environment devoted to automotive multicore systems, named Amalthea [1].

Contributions of this paper. We address the simulation speed and accuracy issues to evaluate resource allocation techniques for automotive multicore systems. The novelty here consists of the implementation of a framework tailored for automotive applications via a seamless integration of Amalthea design environment to a fast multi-core simulation platform. The resulting framework is an open-source and flexible environment, which supports a rich set of modeling features for automotive domain and enables the investigation of new scheduling and mapping policies beyond existing ones. It relies on instruction-level core models interconnected by a packet-based communication network [8]. As an example of design analysis, an automotive application is considered in the paper.

II. Design Concepts

We present the background information on the design concepts, including both application modeling and execution platform design, which form the basis of our framework.

A. Automotive Application Modeling

Existing application modeling techniques capture the characteristics which are critical to their dedicated domains. For automotive domain, the modeling techniques are expected to address performance, reliability, assistance to develop vehicular software, and interaction with cross-domain systems.

1) AUTOSAR: AUTOSAR (AUTomotive Open System ARchitecture) is an open and standardized software architecture which has been devised to manage the growing complexity

of automotive electronic systems while paving the way for further improvement in performance, safety and environmental friendliness [2]. In AUTOSAR, runnable entities are the smallest code-fragments which can be scheduled by the operating system. The software implementing the automotive functionality is encapsulated in software components (SC). The SCs can encapsulate runnables and can be defined by specifying interfaces, execution rates and timing constraints. However, the AUTOSAR SC is an atomic component, which means that each instance of SC is statically assigned to one electronic control unit. Thus, dynamic allocation of runnable entities on different control units is not allowed. On the other hand, in AUTOSAR all cores share a single uniform address space to access the memory and other peripherals. This can be a limitation for distributed-memory multicore system design.

2) Beyond AUTOSAR: Amalthea: Amalthea is another model-driven open source platform targeting the development of multi-core embedded-systems for the automotive domain [1]. In addition to the support for multi-core systems, features of Amalthea are: AUTOSAR compatibility, support for product-line engineering and high level of variability found in modern motor vehicles. The Amalthea application model is based on three main entities, which are listed as follows: *label:* representing data element (memory access); *runnable:* the smallest unit of code that can be scheduled by OS and performs calculations and read/write accesses to labels; and *task:* is a cluster of runnables.

In Amalthea, an application model is a directed task graph, where vertices represent tasks, while directed edges represent either inter-task activation or communication. Figure 1 represents an arbitrary Amalthea application model with seven tasks and three labels. To specify the inter-task communication, it can be observed that task T1 communicates with task T3 via label L3. For this purpose, T1 will perform a write operation on L3 while T3 will perform the read operation on L3. The size of L3 represents the exchanged data volume between T1 and T3. At lower granularity level, a task is composed of runnables as illustrated in Figure 1. Task T4 is composed of eight runnables. The runnable graph also provides information about precedence relationship between runnables.

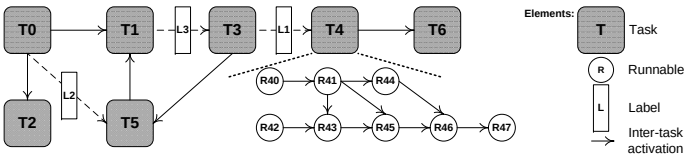


Fig. 1. Example of Amalthea application model.

Beyond the aforementioned Amalthea basic concepts, all the configuration and specification data/information are stored within a model environment for standardization purposes. In addition to basic concepts, software, hardware, operating system, stimulation, constraints, mapping and component concepts are parts of an Amalthea model [1].

B. Simulation Platform

Modern embedded and high-performance computing requirements, including power consumption, cost, and design time are the key parameters, which are leading the integrated circuit technology road map from single-core to multi- and many-core systems. Automotive industry has also adopted these advancements accordingly and modern automobiles are equipped with complex systems like navigator, infotainment system, inter- and intra-vehicle communicator, and dozens of sensors for safety and security. A multi-core system, addressing the computational requirements of these devices needs an interconnection platform for inter-core communications.

To minimize the simulation time for real-time automotive applications, we opted for the transaction level model of Network-on-Chip (NoC) proposed by [8]. This model can achieve fast and accurate simulation of on-chip communications. Its authors have minimized the total number of simulation events by modeling the resource arbitration mechanism at time granularities, which are greater than a clock cycle or a time for a flit transmission, namely a priority preemptive wormhole NoC with virtual channel arbitration.

Different core models have already been proposed with different abstraction levels. Each abstraction level targets different simulation requirements such as simulation speed and accuracy [3]. Here the critical question is: what abstraction level is suitable for our framework? For fast simulation and real-time analysis for automotive applications, we opted for the core model, where computation part of the core performs accurate computation instruction analysis while the communication part of the core is event based to speedup the simulations.

Considering the above core and NoC models, we implemented in SystemC (around 10k lines of code) a multicore platform capable of executing Amalthea application models and providing a number of simulation results. Among these results are *value change dump* (vcd) format files that the designer can display with gtkwave environment[1] for a convenient fine-grain analysis of system temporal behavior. Further computed results are shown later in the case study.

III. A SEAMLESS DESIGN FLOW

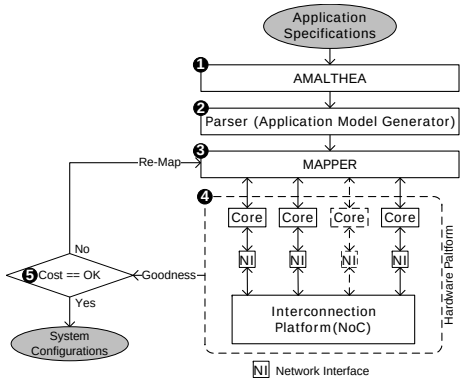


Fig. 2. Simulation Flow

The connection of the implemented platform to Amalthea environment is concretely exploited by a designer via a seamless flow shown in Figure 2, through the following steps:

1) The application specifications are modeled in Amalthea. The Amalthea environment is an Eclipse

[1] http://gtkwave.sourceforge.net

integrated plugin. It supports a variety of modeling features and characteristics which have already been introduced in section II-A. For a complete description, one can refer to [1].

2) The Amalthea model is fed to a parser, which translates this model to an intermediate format in C++ that can then be executed on further platforms.
3) The mapper performs mapping algorithms for the previous application format, generated by the parser, to generate task and communication mapping onto the multicore platform. At this stage, it is necessary to provide mapping methods for multicore architectures which are scalable and able to deal with the real-time dynamics of applications.
4) During executions, cores support scheduling techniques, such as Earlier Deadline First, Rate Monotonic or Application Inherited-Prioritization. Remote communication between cores is achieved by NoC.
5) The *Goodness* of the mapping and scheduling techniques is jointly evaluated in terms of volume of remote access traffic, APL, deadline miss rate, link load variance, task completion time and platform size. It is the task of the application designer to define the *Goodness* criteria according to the application and performance requirements, and ensure that the performance requirements are satisfied without violating important constraints like deadline miss rate for safety critical systems.

If the *Goodness* satisfies the design constraints, system configurations are finalized. Otherwise, mapper is reported with unsatisfied constraints to regenerate a new mapping in the third step and the flow follows the subsequent steps.

IV. Case Study: DemoCar application

The DemoCar is a test application for the proposed framework which is composed of three kinds of runnable entities as shown in Figure 3. The runnables with high activation rate have 5 ms period, and the ones with low activation rate have 100 ms period. Runnables with event based activation start on event occurrence. Such events are notified either on message-reception or at start of the system. The activation messages can be either the input messages or the output messages (feedback). The input messages are the typical automobile attributes such as engine speed, temperature or the battery voltage. Similarly the output messages are triggered cylinder number, ignition time or the desired throttle position.

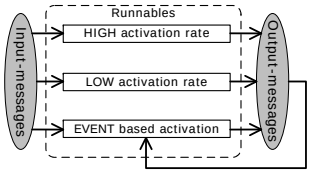


Fig. 3. DemoCar Application overview.

A. Setup information

To evaluate the proposed framework, simulations were performed on an Intel Quad Core i5-4670 at 3.40GHz host with 64-bit Ubuntu OS and 8GB RAM. The applications are simulated for different NoC sizes with 2D-mesh topology, fixed packet length of 32 flits, each flit of 8 bits, router input ports with eight virtual channels and two-position buffers each, and XY routing.

The reported results are for one application iteration. For each configuration, the simulation time, execution time, deadline miss rate, communication traffic load, average packet latency and network level load balancing in terms of standard deviation (STD) are reported. On the basis of these evaluation parameters, designer can select a suitable NoC size, network level mapping approach and core level scheduling technique to fulfill the the application requirements.

Here we present two arbitrary runnable mapping strategies to evaluate the proposed design flow: *zigzag* and *heuristic-local-maximized*. On the one hand for *zigzag*, considering a 2D mesh organization of cores, runnables are mapped one-by-one, globally from top-left to bottom-right. Typically, in a 4×4-mesh NoC, when the mapper maps a runnable at the last node (bottom-right, (3,3)) during the mapping process, next runnable is mapped at first node (top-left, (0,0)). This mechanism repeats till the mapping of last runnable. On the other hand for *heuristic-local-maximized*, the runnables are mapped onto the nodes, where they can have maximum local-label-accesses to minimize the network load, which in-turn minimizes the Average Packet Latency (APL). However, in this case, some nodes can be overloaded for computation. APL and load balancing are some of the key evaluation parameters for application mapping techniques onto NoC architectures.

B. Results

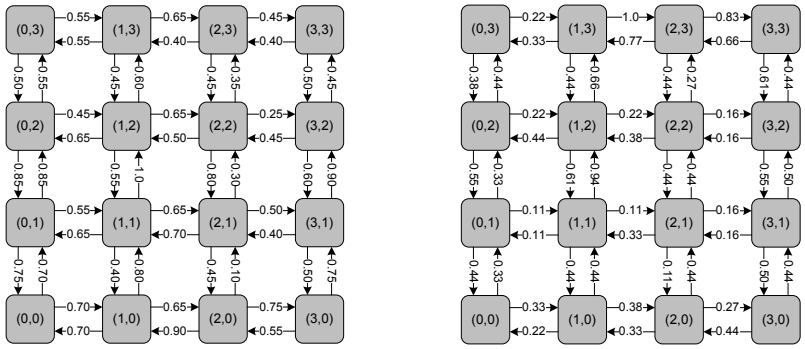
The modeled application contains 3 task entities with 43 runnable entities and 71 label entities. Out of 43 runnables, 22 runnables operate at high activation rate, 4 runnables operate at low activation rate, and 17 runnables get activated on events. There are 10 input message sources and 4 output message sinks. The system operating frequency was 100 MHz. Our parser takes around 3 millisec to generate the corresponding intermediate representation. It can be observed from Table I that different combinations of mapping and scheduling techniques address different requirements of the application such as heuristic-local-maximized mapping with inherited priority based scheduling minimizes the deadline miss rate.

Figure 4 presents the link load for zigzag and heuristic-local-maximized mappings of DemoCar application onto 4×4 2D-mesh NoC. The presented values have been normalized with reference to the maximum link load value for fair comparison. As mentioned above, heuristic-local-maximized mapping minimizes the communication volume, however Figure 4(a) shows the load is comparatively less balanced than the load for zigzag mapping in Figure 4(b). For example, if the load values of top and bottom edges are compared, the loads are comparatively balanced for zigzag mapping while for heuristic-local-maximized mapping, the top edge is heavily loaded as compared to the bottom edge. Such overloading on certain edge can become the performance bottleneck for system performance. The load balancing has been evaluated in terms of STD values.

In order to have a preliminary assessment of our design flow, a realistic application consisting of engine control [4]

#	Mesh Size	Simulation Time (sec.)		Execution Time (μs)		#Deadline misses		#Packets in NoC		APL (ns)		Link Load STD	
		Heu.	ZZ	Heu.	ZZ	Heu.	ZZ	Heu.	ZZ	Heu.	ZZ	Heu.	ZZ
1	2×2	0.012	0.015	11.86	12.87	0	4	116	167	726.10	645.28	3.10	3.11
2	2×3	0.017	0.027	12.31	14.38	0	2	128	181	703.45	851.06	5.64	11.65
3	3×3	0.027	0.049	13.03	15.57	0	3	140	199	671.72	1030.54	3.02	3.59
4	3×4	0.036	0.070	13.68	15.99	0	1	132	194	847.28	1115.30	5.36	6.29
5	4×4	0.050	0.093	15.65	16.00	1	3	139	209	830.77	1208.63	3.75	3.63

TABLE I. SIMULATION RESULTS FOR DEMOCAR WITH ZIGZAG (ZZ) AND HEURISTIC-LOCAL-MAXIMIZED (HEU.) MAPPINGS.



(a) Zigzag mapping

(b) Heuristic-Local-Max. mapping

Fig. 4. Link Load for DemoCar Application (normalized w.r.t. the maximum link load value in the network).

has been addressed. The main purpose of this application is to control the combustion process in the engine in order to produce the torque according to the run-time requirements invoked by a driver. The Amalthea model of the engine control comprises 109 tasks, 1239 runnables and 10436 labels. It has been designed in collaboration with a world-leading automotive system manufacturer. For confidentiality reason, the application details are deliberately omitted. This model has been simulated at a clock speed of 1 GHz. The considered NoC mesh size varies between 7×7 to 10×10. The longest simulation time was around 50 sec. for zigzag mapping on 10×10 NoC size. Heuristic-local-maximized globally offers better real-time results in terms of number of missed deadlines.

V. CONCLUSIONS

A framework for fast exploration of automotive application design on multi-core systems was presented in this paper. The proposed framework mainly addressed the evaluation of mapping and scheduling techniques for automotive domain. For demonstration, different mapping policies (zigzag and heuristic-local-maximized) were assessed on a case study application modeled in Amalthea, a modeling language recently adopted in automotive industry. Simulation results showed the impact of mapping on computation load for a core, inter-core communication volume and network level spatial load balancing (jointly with routing algorithm). This is, among others, a very useful feedback to designer for selecting the best decisions w.r.t. real-time and performance objectives.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under the DreamCloud Project: http://www.dreamcloud-project.org, grant agreement no. 611411.

REFERENCES

[1] *AMALTHEA: Model Based Open Source Development Environment for Automotive Multi-Core Systems*. 2015. www.amalthea-project.org.

[2] *AUTOSAR: AUTomotive Open System ARchitecture*. 2015. http://www.autosar.org.

[3] T. Carlson, W. Heirman, and L. Eeckhout. Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 1–12, 2011.

[4] P. Frey. Case Study: Engine Control Application. Technical Report 2010-03, Ulmer Informatik-Berichte, 2010.

[5] H. G. C. Góngora, T. Gaudré, and S. Tucci-Piergiovanni. Towards an Architectural Design Framework for Automotive Systems Development. In M. Aiguier, Y. Caseau, D. Krob, and A. Rauzy, editors, *Complex Systems Design & Management*, pages 241–258. Springer, 2013.

[6] A. Gerstlauer, J. Peng, D. Shin, D. Gajski, A. Nakamura, D. Araki, and Y. Nishihara. Specify-explore-refine (ser): From specification to implementation. In *Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE*, pages 586–591, June 2008.

[7] M. Gries. Methods for evaluating and covering the design space during early design development. *Integr. VLSI J.*, 38(2):131–183, Dec. 2004.

[8] L. Indrusiak and O. dos Santos. Fast and accurate transaction-level model of a wormhole network-on-chip with priority preemptive virtual channel arbitration. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–6, 2011.

[9] F. Koushanfar, A.-R. Sadeghi, and H. Seudie. EDA for secure and dependable cybercars: Challenges and opportunities. In *49th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 220–228, 2012.

[10] F. Mehdipour, H. Noori, B. Javadi, H. Honda, K. Inoue, and K. Murakami. A combined analytical and simulation-based model for performance evaluation of a reconfigurable instruction set processor. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 564–569, 2009.

[11] K. Popovici and A. Jerraya. Flexible and abstract communication and interconnect modeling for mpsoc. In *Proceedings of the 2009 Asia and South Pacific Design Automation Conference*, ASP-DAC '09, pages 143–148, Piscataway, NJ, USA, 2009. IEEE Press.