Component-based Specification for Multi-Processor System-on-Chip Design

Valentina Zadrija and Vlado Sruk
Faculty of Electrical Engineering and Computing, University of Zagreb
Unska 3, 10000 Zagreb, Croatia
valentina.zadrija,vlado.sruk@fer.hr

***Abstract*—This paper presents a component-based design approach in modeling Multi-Processor Systems-on-Chip (MPSoC). As a component-based specification, SaveComp Component Model (SaveCCM) component technology was employed. In contrast to widely used component-based technologies, SaveCCM is light-weighted, HW/SW platform-independent and considers specific non-functional constraints inherent in heterogeneous MPSoC domain, e.g. deadline, memory consumption or WCET. In order to efficiently describe a class of dataflow applications usually present in embedded systems domain, e.g. image or signal processing algorithms, the research method presented in this paper introduces additional extensions to the SaveCCM model. SaveCCM specification is further synthesized to Transaction Level Model (TLM) to evaluate different MPSoC design options and eliminate possible bottlenecks in early stages of design. TLM was automatically generated using Embedded Systems Environment (ESE) toolset. In order to enable the ESE design flow, SaveCCM specification is first transformed into ESE behavioral model. As a case study, JPEG encoder application was implemented in MPSoC platform based upon the SaveCCM specification using the execution time quality attribute. Design space exploration results are presented and given further evaluation.**

I. Introduction

With rising complexity of embedded systems imposed by the industrial requirements, heterogeneous multi-processor Systems-on-Chip (MPSoC) have emerged as a trade-off solution between general purpose processors and custom hardware implementations. MPSoC allow high performance and low power design, while keeping the time-to-market short enough for consumer's electronics. Shortened time-to-market imposes shortened overall design time demanding a design automation from the user-level specification to the target implementation.

In order to enable the design automation, specification needs to be expressive, well-defined and unambiguous. Having a well-defined SW specification, faclititates system analysis and verification, and consequently increases the productivity of the SW development process lowering design cost. Component-based software engineering (CBSE) is a widely used technique that has proven to be effective in increasing reusability and efficiency of the office and web application development and design. However, it has yet to be proven to work in MPSoC domain due to the specific domain requirements like resource consumption (memory, processing power, communication), dependable and safe operation through time, timeliness or quality-of-service. Most of widely used component technologies like CORBA [1], Microsoft COM [2] or Enterprise JavaBeans (EJB) [3] are inherently heavy-weighted and impose large overhead on the run-time platform. In recent years, there have been several attempts [4], [5], [6] to employ component technology in embedded system domain, mainly focusing on modeling control systems intended for operation on a single Electronic Control Unit (ECU).

Among recently proposed component technologies, SaveComp Component Model (SaveCCM) [7] has emerged as a promising approach to CBSE for embedded systems due its HW and SW platform independence and facilities to specify non-functional properties inherent for embedded systems. However, SaveCCM was primary intended for modeling of real-time embedded control systems and its execution semantics are geared towards such systems. In order to describe general problems usually present in embedded systems domain like multimedia or digital system processing applications, additional extensions to the communication mechanism were implemented.

In this paper, SaveCCM technology is employed for MPSoC design. Presented design flow starts with the SaveCCM software specification and generates the high-speed Transaction-Level Model (TLM) using Embedded Systems Environment (ESE) toolset [8]. Generated TLM presents system-level Model of Structure which serves as a virtual prototype of the component-based specification running on the corresponding multi-processor platform. TLM-based design space exploration yields a set of promising solutions, which can be further refined for board implementation. As a case study, JPEG encoder MPSoC design [9] was implemented following the proposed design flow using execution time quality attribute. Different HW/SW partitioning schemes are evaluated according to the specification constraints.

II. Related work

In recent years, a lot of research has been done regarding the specification model that enables automation of the Multi-Processor System-on-Chip design. Closest to our work is actor-oriented SysteMoC approach employed by SystemCoDesigner design flow [10]. SysteMoC is implemented as a SystemC library, combining state-based and process-based model, and consequently decoupling communication and computation. SystemMoC enables scheduling analysis and automatic HW accelerator synthesis. Daedalus framework [11] employs Kahn process networks (KPN) Model of computation, which is generated automatically from sequential C code given in SANLP form. Daedalus enables the automatic design space exploration upon KPN and corresponding platform model. However, KPN require dynamic scheduling and do not explicitly expose process interface and SW reusability. Koski [12] enables the automatic design space exploration from KPN model described using UML Statecharts. Similar to our approach, UML interface enables back-annotation of the specification with performance measures obtained from lower-levels simulation.

On the other hand, component based software engineering (CBSE) is a highly popular and widely adopted software engineering paradigm in desktop environments, office and web applications. In comparison to described specification models, CBSE emphasizes reusability and consequently scalability, maintainability and reconfigurability. Moreover, CBSE greatly facilitates software development process shortening time to market. There have been several attempts to employ CBSE in embedded system domain [4], [5], [6], [13], however, to our knowledge, none of them addresses specifically Multi-Processor System-on-Chip synthesis in their design flow. Design flow employed in our research aims to utilize CBSE advantages in MPSoC synthesis.

Remainder of the paper is organized as follows: Section III provides an overview of the component-based MPSoC design flow. Description of the employed SaveCCM component technology is given Section IV, while we describe ESE design flow in Section V. In Section VI, we present results of the case study with JPEG encoder mapped to a range of different MPSoC platforms. Section VII concludes the paper.

III. Component-based design flow

The overall design flow employed in this research is based on (i) SaveCCM software specification, (ii) transformation of component-based specification into ESE behavioral model using adapter component, (iii) automatic Transaction Level Model synthesis and (iv) TLM-based design space exploration, Fig. 1. Steps (iii) and (iv) are part of Embedded System Environment design flow.

In the first step of component-based design flow, application is defined using SaveCCM component model. Specification is captured graphically through an extension of UML 2.0 component diagrams [14] and described as a netlist in XML. Component's behavior is defined using C functions. Each component is assigned quality attributes modeling design constraints. In order to describe a subset of multimedia applications, which follow dataflow programming paradigm, additional extensions to SaveCCM model are proposed. Overview of the SaveCCM model and proposed extensions is given in Sections IV and IV-A, respectively.

In order to enable automatic TLM synthesis and design space exploration within ESE toolset, SaveCCM specification is first transformed into ESE behavioral model comprised out of C processes communicating through message passing channels. The transformation is encapsulated within the *Adapter* component described in Section V-A, Fig. 1.

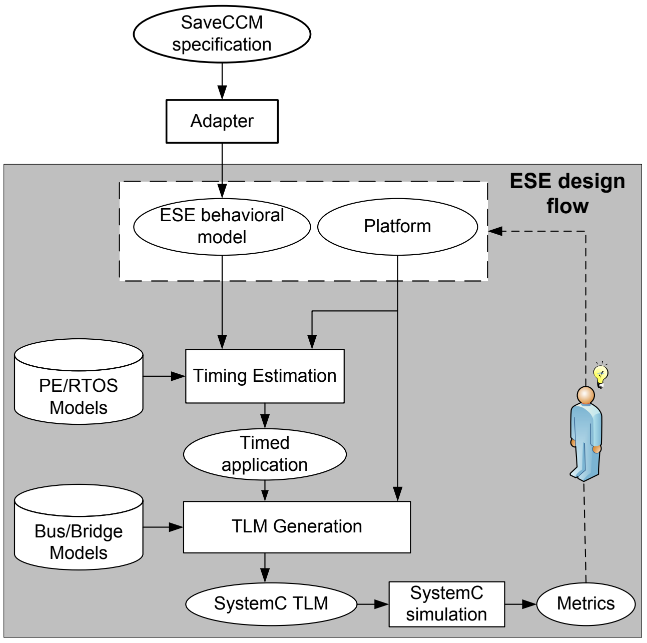

Fig. 1. SaveCCM-based MPSoC design flow

ESE behavioral model is further automatically synthesized into Transaction Level Model in order of few seconds. TLM is annotated with performance measures for computation algorithms and communication transactions and can therefore be used as Model of Performance in design space exploration. Thanks to TLM high speed simulation, design space exploration in terms of different HW and SW partitioning, mapping and scheduling strategies can be performed efficiently through ESE graphical user interface. Results of the design space exploration process can further be refined for board implementation.

IV. SaveComp Component Model

SaveComp Component Model (SaveCCM) [15] is a component based model for small vehicular embedded systems, developed within SAVE [16] and DICES [17] projects at the Malardalen University, Sweden and University of Zagreb, Croatia. SaveCCM syntax and semantics are described as follows.

SaveCCM depicts a component as an encapsulated unit of behavior [18], i.e. a single source code function with interfaces exposed through a set of ports, e.g. *Producer* and *Consumer* components, Fig. 2. Components communicate with each other by exchanging data and control signals through well defined ports and connections, where control flow is clearly separated from the data flow. Control flow is represented using trigger ports and signals, which are used to start the execution of otherwise passive components. Data flow is represented through data ports and signals. In addition, data and control flow can be combined forming data/trigger ports and signals. In this way, components become data-triggered, which is

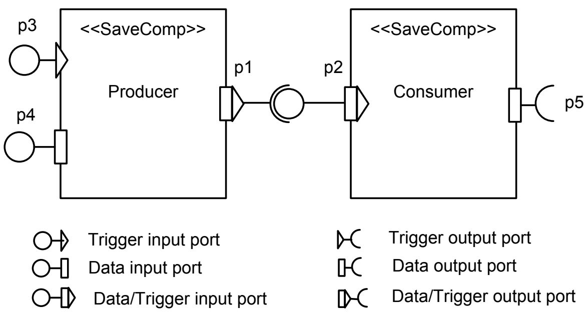

Fig. 2. SaveCCM application example

convenient for modeling dataflow applications. Furthermore, ports are implemented as one-place buffers with overwrite semantics and can be further classified as input and output. Ports are connected using connections, which can be classified as either immediate or complex. Immediate connection *C(p1, p2)* represents atomic transfer of data or trigger signals from one port to another, e.g. connection between *Producer* and *Consumer* components, Fig. 2. On the other hand, complex connection *C(p1, p2, behavior)* represents data or control transfer over channels with possible delay and information loss [18]. Similar to components, complex connections can also have assigned behavioral models, e.g. timed automata or storage buffers.

Modeling of hierarchy is supported through complex component and assembly mechanisms. Complex components describe the functionality through internal structure, while assemblies represent more loose form of behavior representation, hiding the internal structure, rather than exploiting composition mechanism, e.g. *Entropy coding* in Fig. 3. Switches are special types of components that enable the system reconfigurability during runtime i.e. depending on the inputs of a switch component, different components are activated, e.g. *Color switch* in Fig. 3.

SaveCCM components employ concurrent execution model, which is convenient for implementation in MPSoC systems that exploit concurrency by default. Execution model of particular component clearly separates communication from the computation and facilitates the system analysis [15]. SaveCCM employs a specific communication model that follows the semantics of real time control systems with the timeliness as the most important constraint. For example, consider a SaveCCM application comprised out of *Producer* working at a rate *r1*, and *Consumer* working at rate *r2*, where *r1* >*r2*, and the immediate connection *C (p1, p2)*. Because of the fact that *r1* >*r2*, *Producer* writes data to p1 faster than *Consumer* reads it from p2, certain values are overwritten at p2 and consequently they are never received by *Consumer*. In such applications, at certain point of time, only the current value obtained by the *Producer* is relevant for the *Consumer* in order to control the system behavior, not the values sampled prior in time.

A. Modeling dataflow applications in SaveCCM

Specific communication model employed by SaveCCM is not suitable for modeling image or signal processing applications usually implemented in embedded systems domain. Such applications require a reliable communication mechanism, where each data packet sent by *Producer* component is received and processed accordingly by *Consumer* component. In order to describe such behavior, complex connection mechanism *C(p1, p2, behavior)* is extended through FIFO buffers:

$$C(p1, p2, FIFO(n)) \tag{1}$$

FIFO(n) buffer enables components to retain asynchronous communication model, while preserving order and value of data. This ensures reliable communication because *Producer* blocks when the FIFO buffer is full, while *Consumer* blocks when FIFO buffer is empty. Main issue of this implementation is to determine the size *n* of the FIFO buffer with respect to resource constraints while maintaining the non-blocking communication mechanism. This property is also dependent of the underlying platform and could further be optimized in the design space exploration, described in more detail in Section VI.

B. Modeling non-functional properties

Heterogeneous MPSoC impose various design constraints, e.g. resource consumption (memory, processing power, communication), timeliness, quality-of-service, predictability, dependable and safe operation through time. Therefore, violation of specified requirements, even of a proper functional response violates the system functionality. SaveCCM enables describing these non-functional requirements by annotating particular SW components with quality attributes of the required value. However, certain non-functional requirements are HW platform-dependent and will vary for the same SW components according to the target platform. Kopetz and Suri [19] propose to distinguish between SW components and system components. Non-functional properties, such as performance, cannot be specified for a software component in isolation. Such properties must either be specified with respect to the given hardware platform, or parameterized on the underlying platform. A system component, on the other hand, is defined as a self-contained hardware and software subsystem, and can satisfy both functional and non-functional properties.

V. Component-based TLM synthesis

SaveCCM component-based specification annotated with quality attributes is used as an input for Transaction Level Model (TLM) synthesis. For component-based TLM synthesis, Embedded systems Environment (ESE) [20] design flow is employed, Fig. 1. ESE is toolset for MPSoC design, developed at the Center for Embedded Computer Systems at the University of California at Irvine. ESE enables efficient MPSoC design by generating the TLM automatically from C/C++ specification, i.e. ESE behavioral model.

On the other hand, SaveCCM is a light-weighted component technology [7] and as such uses the underlying SW platform execution model in order to support the components during run-time. Such an approach eliminates the need for the component framework that usually supports components at runtime in most component technologies. In [7], SaveCCM application entities are transformed into platform-independent SaveOS and SaveIO API calls, which are further mapped onto RTXC operating system calls and CCSimTech Win 32 simulation environment. Consequently, in order to employ ESE-based TLM synthesis, SaveCCM application is transformed into ESE behavioral model, i.e. *Adapter* component in Fig. 1.

A. ESE behavioral model generation

ESE behavioral model [21] is implemented as a Model of Computation comprised out of C processes communicating through channels. Each process communicates with the environment through a set of ports connected to channels. Channels can implement one of a three interprocess communication types, i.e. blocking process-to-process communication, shared memory communication and communication using FIFO channels. Communication is encapsulated in well defined API calls, i.e *send/recv* API calls for blocking process-to-process communication, *read/write* for shared memory communication and blocking *send/recv* API calls for FIFO-based communication.

Transformation of the SaveCCM specification into ESE behavioral model is performed in two steps, i.e. (i) process allocation and (ii) communication allocation. In process allocation, each SaveCCM component is transformed into a separate C process. However, depending on the computational intensity of the given component, several components can be mapped into a single process.

Communication allocation step depends on the chosen connection type employed for intercomponent data exchange, i.e. immediate or complex connection. Immediate connections represent migration of data between ports, which are implemented as one place buffers with overwrite semantics. Such behavior can be accomplished using shared memory mechanism in ESE, i.e. intercomponent communication is mapped onto ESE *read/write* API calls. Complex connections that represent a reliable communication channel with functionality described through FIFO buffers can be implemented accordingly using FIFO channels in ESE, i.e. *blocking send/ blocking receive* API calls. At the time being, the transformation process is performed by hand. However, since it is based on the well defined semantics of both SaveCCM and target ESE behavioral model, it can easily be automated.

B. Automatic TLM synthesis in ESE

Transaction Level Model synthesis [20] is based upon the ESE behavioral model obtained from SaveCCM specification and corresponding platform model. Platform model is captured graphically as a net-list comprised out of elements available in ESE component library. In general, MPSoC platform can be comprised out of various processing elements (PEs), storage elements, buses and communication interfaces. Once platform is defined, ESE behavioral model is mapped onto a given platform, where processes are mapped onto processing elements, either soft or custom processors; variables to memories and channel to routes. In general, routes are comprised out of source and target processing elements and communication media connecting them, e.g. buses, bridges or shared memories [20].

According to the defined platform and application model, TLM synthesis is performed automatically in order of a few seconds. TLM sythesis is comprised out of two steps, i.e. (i) timing estimation and (ii) SystemC generation, Fig. 1. Timing estimation algorithm is employed in order to provide estimates of the computation and communication for the system under design. The obtained timed model along with the bus and communication interface models is used in SystemC generation. Generated Transaction Level Model represents processing elements as SystemC modules and corresponding application processes as SystemC threads. Communication architecture is comprised out of bus channels and SystemC buffer modules.

Transaction Level Model enables high speed simulation and consequently design space exploration. According to the simulation results and the quality attributes defined in the SaveCCM specification, both application and platform can be easily refined through ESE graphical user interface.

VI. Case study: JPEG encoder implementation

In order to verify SaveCCM-based MPSoC design approach, we have implemented JPEG encoder application. The case study implements lossy, baseline JPEG encoding method.

SaveCCM specification of JPEG encoder is defined graphically through a subset of UML 2.0 component diagrams, Fig. 3. JPEG encoder itself is implemented as an assembly comprised out of five functional components, i.e. *JPEG input*, *Color switch*, *Color conversion*, *DCT* and *Entropy coding*. With well defined set of input and output ports, an assembly component can be integrated into a larger system exploiting the scalability and reusability. *JPEG input* component performs header processing and segmentation of the original bitmap image obtained at the input ports of the assembly. *Color switch* component relays the original image blocks to the *Color conversion* component in the case of color image or directly to the *DCT* component in the case of grayscale image. Subsequently, if activated, *Color conversion* component converts block by block of the original RGB image to a YCbCr color space. Blocks are then transformed using discrete cosine transformation by *DCT* component. *Entropy coding* component is implemented as an assembly comprised out of three basic components, i.e. quantization, zigzag pattern scanning and entropy coding using the Huffman's algorithm. Interfaces of components are exposed through a set of combined data-trigger ports. Therefore, execution is data-driven, which is common behavior for multimedia applications that follow the dataflow-oriented paradigm. Components are interconnected using complex connections implemented as FIFO queues. Moreover, each component within the encoder assembly is assigned an execution time quality attribute, that is further used

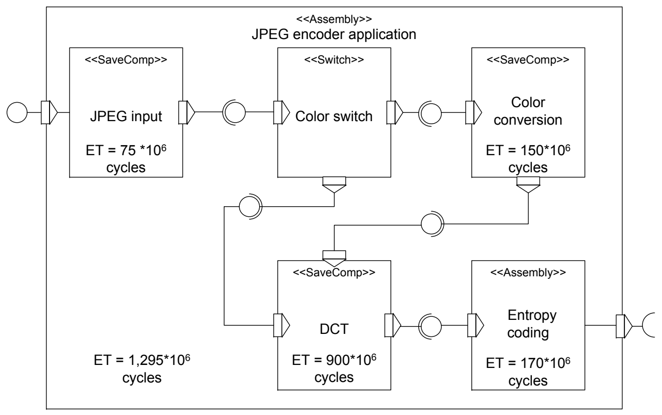

Fig. 3. SaveCCM JPEG application

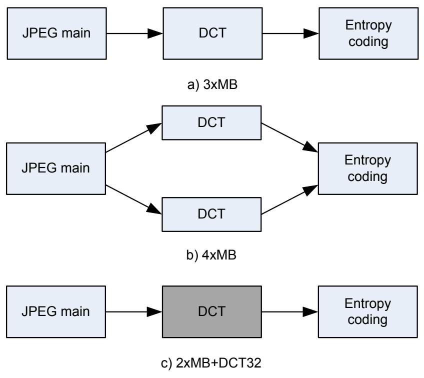

Fig. 4. JPEG partitioning schemes

in the system synthesis, Fig. 3. Execution time of the entire JPEG encoder assembly can be obtained by summarizing execution times of particular components.

First step towards the TLM synthesis is transfomation of the described SaveCCM model into ESE behavioral model, i.e. process allocation and channel allocation. The initial process allocation is obtained according to the profiling results on a single MicroBlaze platform. Results have shown that DCT component is computationally the most intensive component in the application (72% of processor utilization), resulting in a process allocation scheme comprised out of three ESE processes, i.e. *JPEG main*, *DCT* and *Entropy coding*. Computationally less intensive components, *JPEG input*, *Color conversion* along with the *Color switch* switch are comprised into a single process, while their execution time quality attributes are summarized accordingly forming the execution time attribute of *JPEG main* process. In the channel mapping step, SaveCCM complex connections are mapped onto ESE FIFO channels. Size of the FIFO channel is subject of the design space exploration process. Initially, the size of all FIFO channels is set according to the size of a single data packet transfered over the channel.

Once the SW partitioning scheme has been defined, platform and application-to-platform mapping can be defined accordingly. Among the numerous process elements available in ESE platform component library, models of Microblaze soft processor, special purpose DCT32 HW accelerator and OPB bus are employed in our case study. Design space exploration results are presented in the following section.

A. Experimental results

In the design space exploration process driven by defined SaveCCM application and corresponding execution time quality attribute, several different HW/SW partitioning and mapping schemes are explored and evaluated, Fig. 4. Table I presents the performance evaluation results for *DCT* process and the overall JPEG encoder application. For each design, we report the execution time (ET), speed up with respect to the single processor implementation (MT) and the variance (VAR) according to the required execution time given in Fig. 3. Lower VAR values represent better results, i.e. achieved execution time is lower in comparison to the required execution time for the given component.

Specification is initially mapped onto a single MicroBlaze platform (MT) using the RTOS model from ESE component library. Obtained performance results violate the execution time constraints required by the specification, i.e. *DCT* process execution time exceeds the required value by 1.78 times. Consequently, overall JPEG encoder application fails to meet the required design constraints by variance of 2.52 times. In order to increase the performance of the system under design, computationally intensive *DCT* process is offloaded onto a separate MicroBlaze processor (3xMB), Fig. 4a. Offloading *DCT* process onto a separate processor alleviates the expensive context switching operation, however it has no effect over the particular *DCT* process execution time. Performance of the overall system increases by 1.97 times with variance of 1.28, which still violates the requirements. By mapping the *DCT* process over two MicroBlaze processors using the 2-way interleaved execution (4xMB, Fig. 4b), we achieve the *DCT* speed up by two times with positive variance with respect to the design constraints, i.e. VAR = 0.89. Overall system performance is increased by 3.83 times and variance of 0.66 with respect to desired constraints.

Finally, we employed the heterogeneous platform comprised out of 2 MicroBlaze processors and special purpose DCT32 HW accelerator (2xMB+DCT32), Fig. 4c. By employing DCT32 accelerator we increase *DCT* process execution time by 18 times with variance of 0.10. However, overall system performance increases for 8,37 times, due to the disproportion in processing rate of the DCT32 accelerator and MicroBlaze processor performing *Entropy coding* further in pipeline.

VII. Conclusion

In this paper, we presented SaveCCM component-based design flow for Multi-Processor Systems-on-Chip and corresponding experimental results for JPEG image compression algorithm. In comparison to widely used MPSoC specification models, component-based specification promotes SW reusability, scalability and maintainability. SaveCCM specification

TABLE I
JPEG ENCODER DSE RESULTS

	DCT			JPEG encoder application		
Design	**ET (10^6 cycles)**	**Speed up**	**VAR**	**ET(10^6 cycles)**	**Speed up**	**VAR**
MT	1,598	1.00x	1.78x	3,261	1.00x	2.52x
3xMB	1,598	0.99x	1.78x	1,654	1.97x	1.28x
4xMB	799	1.99x	0.89x	850	3.83x	0.66x
2xMB+DCT32	88	17.99x	0.10x	400	8.14x	0.31x

model is defined using a subset of UML 2.0 component diagrams, where each component is assigned an execution time quality attribute. The execution time is further used to control MPSoC synthesis.

In order to rapidly evaluate different design alternatives and detect possible errors at early stages of design, we have employed the Embedded Systems Environment (ESE) toolset. ESE enables the design space exploration at a high level of abstraction by generating Transaction Level Model (TLM). TLM provides estimates of the computation and the communication for the given SW and platform specification. The SW specification of the ESE toolset consists out of C processes communicating through message-passing channels. In order to apply SaveCCM to ESE design flow, we developed an adapter component, which performs model transformation of SaveCCM specification into ESE behavioral model. By simulating generated Transaction Level Model, we performed the design space exploration according to the required execution time design constraint.

The effectiveness of our component-based approach to MPSoC synthesis is illustrated by experimental results obtained from JPEG encoder MPSoC design, where several MPSoC design alternatives are evaluated in a very short time. For further work, we propose integration of additional non-functional constraints into our design flow, as well the automation of the proposed component-based MPSoC synthesis.

ACKNOWLEDGMENT

The work presented in this paper is developed within the Application-oriented Embedded System Technology project supported by the Unity through Knowledge Fund. The authors wish to thank Ana Petricic, Juraj Feljan and Luka Lednicki at the DICES project, University of Zagreb, for support. In addition, we would like to thank Jan Carlson from the MRTC research center at the University of Varsteas, Sweeden for providing examples of the SaveCCM-based applications [15]. Finally,special thanks to the Center for Embedded Computer Systems (CECS) for allowing download of the ESE tool-set and Samar Abdi for support.

REFERENCES

[1] OMG. (2006) Corba component model v 4.0. [Online]. Available: http://www.omg.org/spec/CCM/4.0/

[2] Microsoft. (2009) Component object model. [Online]. Available: http://msdn.microsoft.com/en-us/library/ee663262(VS.85).aspx

[3] Sun Microsystems. (2009) Enterprise javabeans version 3.0. [Online]. Available: http://java.sun.com/products/ejb/docs.html

[4] R. van Ommering, F. van der Linden, J. Kramer, and J. Magee, "The Koala Component Model for Consumer Electronics Software," *Computer*, vol. 33, no. 3, pp. 78–85, 2000.

[5] P. O. Mller, C. Stich, and C. Zeidler, "Components @ work: Component technology for embedded systems," *EUROMICRO Conference*, vol. 0, p. 0064, 2001.

[6] K. Wallnau and J. Ivers, "Snapshot of the CCL: A Language for Predictable Assembly from Certifiable components," Carnegie Mellon, Software Engineering Institute, PA 15213-3890, Tech. Rep. CMU/SEI-2003-TN-025, Jun. 2003.

[7] M. Akerholm, A. Moller, H. Hansson, and M. Nolin, "Towards a Dependable Component Technology for Embedded System Applications," in *WORDS '05: Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 320–328.

[8] D. Gajski, S. Abdi, Y. Hwang, L. Yu, H. Cho and I. Viskic, "ESE Front End 2.0," University of California, Irvine, Tech. Rep., Sep. 2008.

[9] V. Zadrija and V. Sruk, "Design Space Exploration of a Multi-core JPEG," in *Proceedings of MIPRO 2009, 32nd International Convention, Vol. III., CTS & CIS*, 2009, pp. 60–65.

[10] J. Keinert, M. Streubuhr, T. Schlichter, J. Falk, J. Gladigau, C. Haubelt, J. Teich and M. Meredith, "SystemCoDesigner - an automatic ESL synthesis approach by design space exploration and behavioral synthesis for streaming applications," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 14, no. 1, pp. 1–23, 2009.

[11] H. Nikolov, M. Thompson, T. Stefanov, A. Pimentel, S. Polstra, R. Bose, C. Zissulescu, and E. Deprettere, "Daedalus: toward composable multimedia MP-SoC design," in *DAC '08: Proceedings of the 45th annual Design Automation Conference*. New York, NY, USA: ACM, 2008, pp. 574–579.

[12] T. Kangas, P. Kukkala, H. Orsila, E. Salminen, M. Hännikäinen, T. D. Hämäläinen, J. Riihimäki, and K. Kuusilinna, "UML-based multiprocessor SoC design framework," *ACM Trans. Embed. Comput. Syst.*, vol. 5, no. 2, pp. 281–320, 2006.

[13] C. Norstrm, K. Sandstrm, J. Mki-Turja, M. Gustafsson, and N.-E. Bnkestad, "Experiences from Introducing State-of-the-Art Real-Time Techniques in the Automotive Industry," *Engineering of Computer-Based Systems, IEEE International Conference on the*, vol. 0, p. 0111, 2001.

[14] OMG. (2009) Introduction To OMG's Unified Modeling Language. [Online]. Available: http://www.omg.org/gettingstarted/what_is_uml.htm

[15] I. C. H.Hansson, M. Akerhol and M. Torngren, "SaveCCM - A Component Model for Safety-Critical Real-Time Systems," in *EUROMICRO '04: Proceedings of the 30th EUROMICRO Conference*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 627–635.

[16] SAVE. [Online]. Available: http://www.mrtc.mdh.se/SAVE

[17] DICES: Distributed Component-based Embedded Software Systems. [Online]. Available: http://www.fer.hr/dices/

[18] M. Åkerholm, J. Carlson, J. Håkansson, H. Hansson, M. Nolin, T. Nolte, and P. Pettersson, "The SaveCCM Language Reference Manual," Mälardalen University, Technical Report ISSN 1404-3041 ISRN MDH-MRTC-207/2007-1-SE, January 2007.

[19] H. Kopetz and N. Suri, "Compositional Design of RT Systems: A Conceptual Basis for Specification of Linking Interfaces," in *ISORC '03: Proceedings of the Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*. Washington, DC, USA: IEEE Computer Society, 2003, p. 51.

[20] D. Gajski, S. Abdi and I. Viskic, "Model Based Synthesis of Embedded Software," in *SEUS '08: Proceedings of the 6th IFIP WG 10.2 international workshop on Software Technologies for Embedded and Ubiquitous Systems*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 21–33.

[21] L. Yu, S. Abdi, and D. Gajski, "System Definition and ESE Data Structure," Center for Embedded Computer Systems, University of California, Irvine, Tech. Rep. TR 09-04, Mar. 2009.