Design Space Exploration of a multi-core JPEG

V. Zadrija and V. Sruk
Department of Electronics, Microelectronics, Computer and Intelligent Systems
Faculty of Electrical Engineering and Computing, University of Zagreb
Complete Address: Unska 3, Zagreb, 10000, Croatia
Phone: (003851) 6129-554 Fax: (003851) 6129-653 E-mail: valentina.zadrija@fer.hr

Abstract - This paper presents the design space exploration of a multi-core implementation of the JPEG algorithm using the Embedded System Environment (ESE). ESE is a tool-set, which enables multi-core system design by high-level modeling of both the hardware platform and software application. In order to define an application model for ESE, sequential JPEG code written in C was partitioned and translated into concurrent processes, which communicate via abstract channels. The application model is then mapped on the system platform captured as a graphical netlist consisted out of SW and HW cores, buses and buffers. ESE provides means for automatic translation of these models to Transaction Level Models (TLM) in order of seconds. High-speed TLM simulation was used to identify possible bottlenecks and evaluate different design options for both HW and SW partitioning of the JPEG algorithm application. The obtained experimental results have shown that such approach may find a good solution regarding specific design constraints in a very short time.

I. Introduction

The high-performance requirements and stringent design constraints imposed on modern embedded systems are making the designer's job increasingly difficult. This is especially challenging in the field of mobile devices, which usually have to support a wide range of multimedia capabilities, which require high-performance and low power consumption at the same time. This is impossible using a traditional design flow based on a single general-purpose or DSP processor and sequential software. These systems require careful planning of system architecture and design of custom hardware. On the other hand, the market demands that new devices must be available as soon as possible. Shortened time-to-market imposes shortened design time, which makes it extremely difficult to evaluate different design options and design custom hardware. Single-processor software-based design flow can be viewed as the approach which has the highest design productivity (i.e. shortest time-to-market), but lowest design quality in terms of performance and power utilization. The problem with this approach is that it is often unable to meet the required design constraints. The other extreme is the implementation of the whole design in custom hardware, which offers high design quality but usually prolongs time-to-market to levels unacceptable for consumer devices. Design approach based on heterogeneous multi-core systems-on-chip represents a trade-off between these extremes. Such approach allows for a design of high-performance and low-power systems, while keeping time-to-market short enough for consumer electronics.

Heterogeneous multi-core systems are essentially a collection of general-purpose processors, DSP processors, custom hardware accelerators and various other cores, such as memories, bus modules, peripheral devices etc. Hardware accelerators can either be specifically designed or automatically generated in order to accelerate a particular time-critical application task. The main advantage of using multi-core systems is their ability to use task-level parallelism inherent to the application. Parallel execution enables performing the same amount of work in fewer clock cycles, thus lowering the system frequency while maintaining the required performance. Lowering the frequency reduces the power consumption and enables meeting low-power constraints.

However, multi-core systems are difficult to design, both from hardware and software point of view. Because of the wide choice of processing elements (PEs) and possible hardware/software partitioning strategies and parallelization techniques, design space exploration is extremely time-consuming process. To efficiently explore the design space and find appropriate hardware platform and adequate software mapping in a reasonable amount of time, systems are usually modeled on a higher level of abstraction. Such high-level model is then used to drive simulation-based evaluation of alternative designs and to synthesize and implement the final system. According to the application-specific constraints, design can be optimized with respect for speed, chip area or power consumption. Special system-level design tools like the Embedded System Environment (ESE) are used to facilitate this design process and automate generation of different models employed for simulation, synthesis and implementation [4].

Image encoding algorithms have become a popular example in studying multi-core system on chip design methodology, hardware/software co-design as well as custom acceleration. In this paper, we present the design space exploration of a multi-core JPEG encoder application [19] using ESE tool-set. Different HW/SW partitioning schemes are analyzed and corresponding functional and timed properties are estimated and verified.

II. Design Space Exploration Overview

Design space exploration is a process of exploring and evaluating different design alternatives in order to find a configuration, which fits specific design constrains [9]. The values for certain design parameters can be determined statically (e.g. the estimated maximum frequency), but evaluating a design alternative usually involves simulation. Simulation is also used for verifying functional correctness of the system when changes are made. The growing complexity of modern systems, i.e. the increasing amount of software and the number of PEs are making low-level simulation extremely difficult and time-consuming. With the advent of multi-core systems, rapidly growing software content and shortened time-to-market, RTL modeling and verification is no longer practical. It is certain that size, complexity and heterogeneity of future multi-core systems will present a problem even for models targeted at Instruction Set Simulators (ISS). To enable faster simulation and faster design time, systems need to be modeled on a higher-level of abstraction. High-level models sacrifice some of the accuracy in measuring design parameters to enable simulating the design in a reasonable amount of time. In this way, it is possible to explore several design alternatives and design process can converge more quickly to a solution that meets specified constraints.

Transaction level modeling has emerged as the next level of abstraction for system design. Transaction Level Model (TLM) approach clearly separates communication details among modules from the implementation specific details of these modules [5]. Communication is modeled using channels, which are simply a repository for communication services. Transaction requests take place by calling interface functions of these channels, which encapsulate low-level details of the information exchange. At the transaction level, the emphasis is more on the functionality of the data transfers - what data are transferred to and from what locations - and less on their actual implementation.

A. Related Work

In recent years, there has been a lot of research on DSE addressing the problem at the higher level of abstraction. As a result, several modeling environments for system design and synthesis were developed. Balarin et al. [2] present Metropolis, a DSE environment that integrates tools for simulation, verification, and synthesis. Metropolis supports refinement and abstraction, thus allowing top-down and bottom-up methodologies with a meet-in-the-middle approach. Sesame [15] is a tool for performance evaluation and exploration of heterogeneous architectures for the multimedia application domain. In Sesame, application specification is given as a Kahn process network modeled with a C++ class library. Platform is built from building blocks accessible in the library. By simulating TL model of the application, performance evaluation can be done. In order to co-simulate the application and the architecture, a trace-driven simulation approach technique is employed. Kopetz [12, 13] proposes a component model for dependable automotive systems. Both approaches aim to achieve dependability and reliability of heterogeneous multi-core systems by using predefined platform templates. However, their design flow requires platform-specific input models.

Using TLM, different design flows were developed. Approach presented in [16] uses SpecC to generate TLM in order to perform design space exploration. In such approach, modeling abstraction requires implementation decisions for synchronization to already be made. Moreover, there is no discussion of modeling communication processes such as bridges and routers. However, designers are still required to understand complex channel modeling in a non-standard SpecC language. In [3] timed Programmer view (PVT) level within TLM modeling approach was proposed. System is modeled on the two complementary sublevels of the PVT with respect to accuracy and simulation speed up. However, proposed approach includes modeling system components in SystemC language and therefore requires for a designer to have deep knowledge of the SystemC.

There have been several other approaches that start from a very high-level specification and go to a cycle accurate model or synthesizable model implemented in a FPGA board. Paper [10] proposed system design flow starting from the specification in Simulink, which is then converted to different abstraction levels by the refinement to Instruction Set Simulators. Namely, specification is converted to a Simulink Combined Algorithm and Architecture level, a Virtual Architecture, Transaction-accurate Model and finally to a Virtual Prototype which is cycle accurate. The main drawback of this approach is that designer needs to be familiar with Simulink to specify the design, before being able to do any transformation/refinement. Another approach based on UML specification that results in a synthesized model mapped to a FPGA-based platform is presented in [11]. In [8] authors have proposed a design flow where specification is given in a subset of a SystemC language, while target platform template is built from components specified in the library. As already mentioned above, the common disadvantage of all described approaches is that designers need to learn and use another language in order to perform the design space exploration.

ESE requires application specification in C language and provides means for automatic generation of the TL model. In contrast to all previously described techniques and tools, system designer does not have to learn any additional language to model the system and perform the design space exploration.

III. JPEG ENCODER ALGORITHM

JPEG (Joint Photographic Experts Group) is a widely used still image compression standard, very popular in embedded systems, especially in multimedia devices and digital cameras [11]. The JPEG standard specifies two classes of encoding and decoding approaches, namely lossless and lossy compression [7]. In this paper, lossy compression is discussed. Generally, three types of lossy compression are defined by the JPEG standard; baseline sequential, progressive and hierarchical method. However, the most popular compression method is indeed baseline sequential because it provides sufficient capabilities for wide range of applications. JPEG baseline compression operates on blocks of pixels and is based on forward discrete cosine transform (DCT) and Huffman entropy encoding. The bitmap is first segmented into 8 × 8 non-overlapping pixel blocks from left to right and top to bottom, Fig. 1. On each of these blocks, DC level shifting is performed followed by DC transformation and quantization. Zig-zag pattern scanning mechanism is applied in order to transform the image block into a vector. Vectors are then entropy coded using either Huffman or arithmetic coding algorithm.

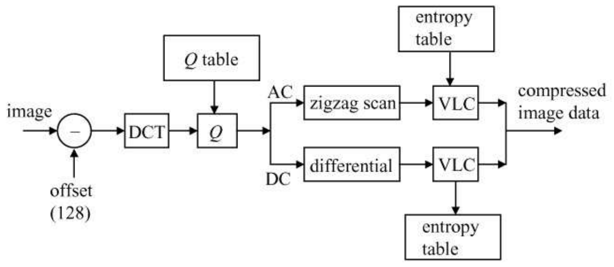

Fig. 1. Baseline JPEG encoder block diagram [7].

JPEG can also be used in coding of video, on the basis that video is a succession of still images. In this case, the process is called Motion JPEG (M -JPEG) [7]. As a response to the increasing demands of multimedia content in variety of applications, especially on Internet, JPEG2000 standard was issued [14]. Despite the fact that new versions of the standard have been issued, JPEG is still widely used and falls into the wide category of computationally intensive digital signal processing (DSP) problems. Therefore, conclusions derived from the design space exploration process for this algorithm can be later applied to other similar algorithms.

A. Embedded JPEG implementations

In recent years many JPEG-related (JPEG, JPEG 2000, Motion JPEG) algorithm implementations in embedded systems were proposed. JPEG related encoding algorithms, especially blocks performing discrete transformations (Discrete Wavelet Transformation or Discrete Cosine Transformation) are very computationally consuming. In order to speed up the transformation process, hardware accelerators are used. Paper [1] presents an implementation of the JPEG 2000 encoder and decoder algorithm defined in Part 1 of the standard. Custom hardware modules are used to implement discrete wavelet transform, intra-subband bit-plane coding, and binary arithmetic coding. System architecture has been implemented in VHDL. Zhang et al. in [21] applied loop transformation techniques on a scalable JPEG 2000 coder during the architectural exploration stage. The emphasis in this paper was on the maximization of the throughput between JPEG 2000 building blocks. Suggested HW/SW partitioning was based upon profiling experiments on the standard PC. I.e. the most time-consuming blocks like DWT and entropy coding were implemented in hardware, while the rest of the application was implemented using standard software cores. Effectiveness of their approach was proven on Xilinx FPGA. In [20] FPGA-based multi-core system for JPEG encoder application was presented. Several different interconnections between cores were explored and trade-offs between them were analyzed. However, proposed tools and methods cope with the design of the JPEG application on the low level, designing the RTL model manually. In order to fill the gap between the application specification given in high-level language like C and RTL model given in VHDL/Verilog, in this paper, we present a methodology for system level design with an application level model as input and TL model as a result. Using ESE, such TL model can be further refined into Pin-Cycle Accurate model [5] suitable for board implementation.

IV. MODEL BASED DESIGN SPACE EXPLORATION IN ESE

In this section, model based design exploration process using ESE is described in detail. ESE [6] is a toolset for modeling, synthesis and validation of multi-core embedded system designs. It has been developed at the Center for Embedded Computer Systems (CECS) at the University of California Irvine. ESE is comprised out of two parts: ESE FrontEnd and ESE BackEnd. ESE Front End provides automatic generation of SystemC transaction level models (TLMs) from graphical capture of system platform and application C/C++ code. ESE Back End provides automatic synthesis from TLM to Pin-Cycle Accurate Model (PCAM) consisting of RTL interfaces, system SW and prototype ready FPGA project files. ESE generated RTL can be synthesized using standard logic synthesis tools and system SW can be compiled along with application code for a given processor. ESE automatically creates Xilinx EDK projects for download to Xilinx boards [18].

Overview of the design process using ESE FrontEnd [4] is given in Fig 2. Modeling process begins at the application level, which is comprised out of C processes communicating through synchronized point to point channels and shared variables. System platform is captured graphically in ESE as a net-list of the process elements (PEs), memories, buses and communication interfaces. For given application, model elements are then mapped to the corresponding platform components, i.e. application processes are mapped to the PEs, while channels are mapped to routes in the platform.

Above described system definition along with the library of the data models for PEs, buses and RTOSes is used in Transaction Level Model (TLM) generation. ESE Front-End allows for automatic generation and simulation of the TL model in order of seconds. Transaction Level Model represents the PEs as SystemC modules and corresponding application processes as SystemC threads. Communication architecture is comprised out of bus channels and SystemC buffer modules. ESE FrontEnd provides means for automatic generation of the two types of the Transaction Level Model: functional and timed. Functional TLM presents a completely untimed model of

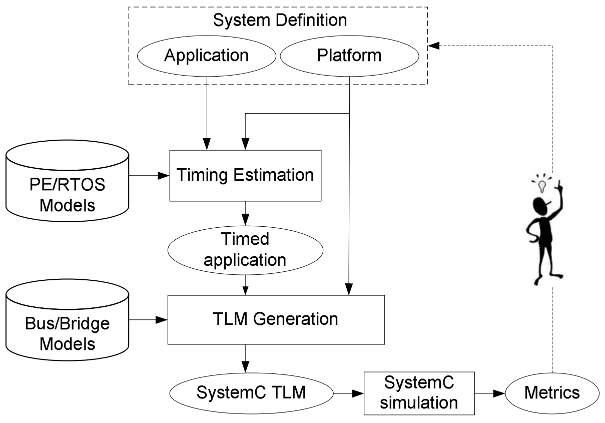

Fig. 2. ESE FrontEnd tool flow [4].

the system considering only causal dependencies. Therefore, it is adequate for the system behavior verification. On the other hand, timed TLM is generated using timed estimation algorithm and it is suitable for the performance evaluation of the system design. Given the performance analysis results of the timed TLM system definition, software and hardware partitioning can be easily refined allowing for quick and efficient design space exploration. Application and platform specification overview is given in the following chapters.

A. Application model in ESE

Application specification is comprised out of C processes communicating through abstract channels or shared variables [4]. Both, processes (*P1*, *P2*, *P3* and *P4* in Fig. 3) and channels (*Ch1* and *Ch2* in Fig. 3) may be defined through graphical user interface (GUI). In addition, C code assigned to specific processes could be modified easily. Processes use channels for synchronized communication and variables for unsynchronized communication. For both synchronized and unsynchronized communication mechanisms, ESE API calls are defined: (a) *send*/*recv* methods for process-to-process channels and (b) *read*/*write* methods for shared variable based communication. Such approach clearly separates communication interface from the actual computation code. Consequently, application model partitioning is easily refined by applying modifications in the interface implementation only. Above described API calls are sufficient to implement other complex communication services like FIFOs, mutexes, mailboxes or events.

B. Platform template in ESE

ESE FrontEnd toolset provides means for graphical definition of a multi-core system platform. In general, a platform is composed out of process elements (PEs), buses, storage cores and transducers[5], Fig. 3. Process elements are either general-purpose cores, custom HW components or IPs on which application processes are mapped. Several different application processes can be executed simultaneously on the single process element. In order to support such multi-threaded applications, several RTOS models are available in ESE. Storage cores correspond to the platform elements that don't have any active thread of computation. Application variables are mapped to memories, which are either local to process element or shared between several process elements. Buses are generic communication units that can act as point-to-point links, shared buses with arbitration or even network links. Buses have well defined protocols and may connect to compatible ports on a given core. Transducers represent generic interface cores and they can denote shared memories, bridges or routers. Internally, transducers are comprised out of buffers and provide functionality of store-and-forward static routing. In order to connect incompatible buses via different ports, transducers also implement protocol conversion mechanism. Application model channels for process communication are mapped to routes consisting of buses and bridges.

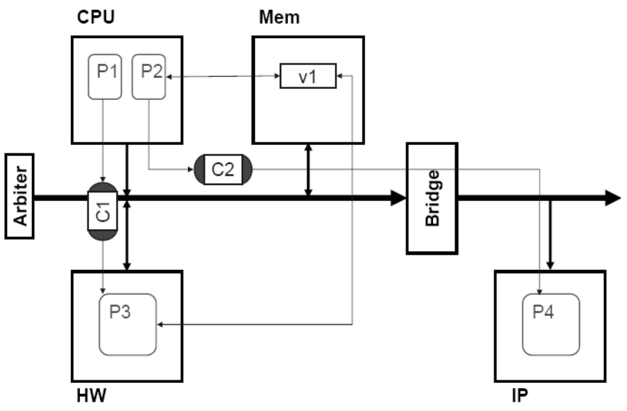

Fig. 3. Multi-core system definition in ESE [4].

V. CASE STUDY: MULTI_CORE JPEG DSE

In this section, a case study of the design space exploration for multi-core JPEG encoder implementation using ESE is described. First, an application model is described followed by the experimental results of the design space exploration process.

A. JPEG encoder application model

The original JPEG encoder source code was obtained from [20]. This code was already optimized for execution in embedded systems. According to the JPEG standard, application source code was partitioned manually into four C processes, Fig. 4. Data flow of the application is as follows. Bitmap image is first segmented into blocks by *JPEG_main* process. Subsequently, *Color_conversion* process converts block by block of the original image to a suitable color space, namely RGB information contained in the original image blocks is encoded in YCbCr color space. Blocks are then transformed using discrete cosine transformation by *DCT* process. *Entropy_coding* process is comprised out of several operations. First, it performs quantization of the DCT transformed blocks, then converts such image blocks into a vector by zigzag pattern scanning. Finally, it performs entropy coding using Huffman's algorithm.

For inter-process communication, message-passing mechanism implemented by send/recv API calls in ESE was used. Send/recv methods impose handshake synchronization semantics, where the receiver process blocks until the sender have sent the data [5].

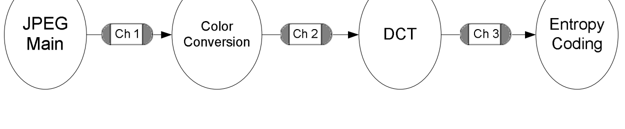

Fig. 4. JPEG encoder application model.

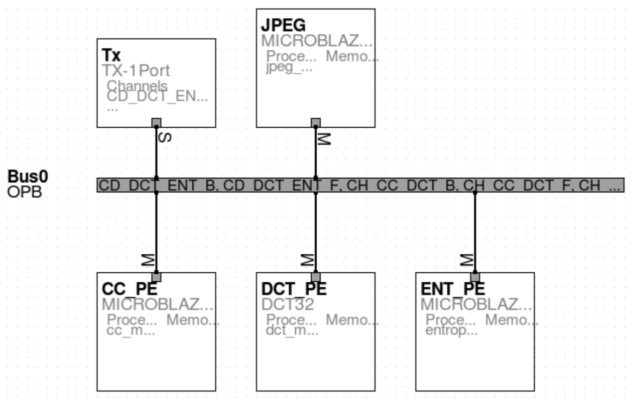

Fig. 5. Homogeneous multi-core platform (4mB).

C. Experimental results

According to the application model described in previous section, the goal was to find a good hardware and software partitioning schemes with respect to computation distribution and communication overhead between cores.

In the first experiment, we have mapped the entire application model to the single MicroBlaze core. As it was to be expected, timed TLM simulation results have shown that the DCT function is computationally the most consuming function of the entire application (takes 80 % of the overall execution time). Thus, DCT was identified as an ideal candidate for offloading onto a separate core or migration to the custom hardware.

According to the performance estimation results obtained from the first experiment as well as the software partitioning presented in chapter A, JPEG application code was partitioned over the four MicroBlaze cores (4mB), Fig. 5.

Nevertheless, utilization of the DCT process element was still high (95.5 %), identifying it as a bottleneck. This suggested the usage of hardware accelerator to speed up the DCT function. In third experiment, following platforms were used: three MicroBlaze cores and DCT32 custom hardware accelerator (3mB + DCT32). DCT32 is hardware accelerator designed specifically to perform DCT function [17].

To evaluate the advantages of using the DCT32 hardware accelerator, we first measured the speedup of a custom DCT32 processor over a MicroBlaze core when running DCT algorithm only, Table I. As a performance metric, performance time was used. Results show that usage of the DCT32 hardware accelerator increases performance of the DCT function 4.81 times, which is a significant speed up.

Finally, performance of the heterogeneous system was measured and compared with the homogeneous system. As a quality metric, computational utilization of the process element is used, Table II. Due to *DCT* process element speedup, utilization of other process elements in heterogeneous system is increased 3.58 times. However, in comparison to other process elements, *Entropy_coding* PE has achieved greatest utilization (78.8%). That is because this PE receives its inputs directly from the *DCT* process element. Utilization of the DCT process is decreased for 25.6% because process elements producing its inputs (*JPEG_main* and *Color_conversion*) aren't able to process data at adequate speed.

Furthermore, described platforms were compared with respect to overall performance time, Table I. Heterogeneous multi-core system performance is increased 3.57 times in comparison to the homogeneous multi-core system. The achieved speed up of the DCT function for 4.81 times increases overall system performance for 3.57. As already described above, that because the system is only partially balanced, i.e. in comparison with the DCT process element, the elements generating its inputs work at lower rate.

VI. CONCLUSION

In this paper, model based design space exploration of the JPEG encoder multi-core implementation using ESE toolset was explored. ESE allows for system level design approach employed in the development of the JPEG multi-core system. An application model was defined as a set of the C processes communicating via message passing channels. A system platform was defined as a graphical netlist and with corresponding design parameters assigned. Furthermore, both system platform and application can be easily extended and refined. Thus, only few modifications of the existing model are required for modeling various other design options. Developed application level model is

TABLE I
DCT PROCESS EXECUTION TIME COMPARISON

	MicroBlaze	DCT32	Speed-up ratio
Performance time(cycles)	89.115	18.513	4.81

TABLE II
PROCESS ELEMENT UTILIZATION COMPARISON

PE	4mB	3mB+DCT32	Utilization ratio
JPEG_main	8.9%	31.8%	3.58 times
Color_conversion	18.4%	65.7%	3.58 times
DCT	95.5%	71.0%	25.6%
Entropy_coding	22.0%	78.8%	3.58 times

TABLE III
SYSTEM PERFORMANCE COMPARISON

	4mB	3mB+DCT32	Speed-up ratio
Performance time (cycles)	272.200.230	76.064.760	3.57

mapped on a corresponding platform forming the system definition. For the defined system model, Transaction Level Model (TLM) is automatically generated in ESE in order of seconds. High speed timed TLM simulation allows for system performance analysis using the utilization of particular process elements and communication overhead as quality metrics. According to the performance estimation results, both system application and platform are easily refined. Thus, different design choices are efficiently explored and evaluated.

Experimental results of the JPEG encoder implementation have shown that presented TLM-based exploration approach may find a good solution regarding the design constraints in a very short time (order of a few seconds). Several multi-core systems were modeled using ESE. Due to high-speed simulation of the TLM, models were easily refined eliminating the bottlenecks in the process. Design space exploration using ESE tool-set could be efficiently used for further improvements on parallel execution of the JPEG encoding algorithm.

VII. ACKNOWLEDGEMENTS

The work presented in this paper is developed within the Application-oriented Embedded System Technology project supported by the Unity through Knowledge Fund. This work builds upon many years of research in the system level design at the Center for Embedded Computer Systems (CECS), University of California at Irvine. The authors wish to thank CECS for allowing download of the ESE tool-set and Samar Abdi for support. In addition, we would like to thank Sun Wei, Joris van Emden and Marcel Lauwerijssen from Technical University Eindhoven for providing the JPEG application source code. Finally, special thanks to Roko Grubišić for his help in overall system implementation.

REFERENCES

[1] Andra, K.; Chakrabarti, C.; Acharya, T.; (2003). A high-performance JPEG2000 architecture. *IEEE Transactions on Circuits and Systems for Video Technology* , 13 (3), 209 - 218.

[2] Balarin, F.; Watanabe, Y.; Hsieh, H.; Lavagno, L.; Passerone, C.; Sangiovanni-Vincentelli, A.; Metropolis: an integrated electronic system design environment, *Computer*, vol. 36, no. 4, pp. 45–52, 2003.

[3] Ben, R.A.; Niar, S.; Meftali, S.; Dekeyser, J.-L.; (2007). An MPSoC Performance Estimation Framework Using Transaction Level Modeling. *Proceedings of the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications* (str. 525 - 533). Daegu, Korea: IEEE Computer Society.

[4] Gajski, D. D.; Abdi, S.; Hwang, Y.; Yu, L.; Cho, H.; Viskic, I.; (2007, October). ESE Front End 2.0. University of California, Irvine.

[5] Gajski, D. D.; Abdi, S.; Viskic, I.; (2008). Model Based Synthesis of Embedded Software. *Proceedings of the 6th IFIP WG 10.2 international workshop on Software Technologies for Embedded and Ubiquitous Systems*. Anacarpi, Capri Island, Italy.

[6] Gajski, D. D.; Gerstlauer, A.; Abdi, S.;. (2007, January 23). Embedded System Design: Concepts and Tools. ASP-DAC 2007 Pacifico Yokohama, Japan.

[7] Ghanbari, M. (2003). Standard Codecs: Image Compression to Advanced Video Coding. London, UK: The Institution of Electrical Engineers.

[8] Haubelt, C.; Falk, J.; Keinert, J.; Schlichter, T.; Streubühr, M.; Deyhle, A.; Hadert, A.; Teich, J.; (2007). A SystemC-Based Design Methodology for Digital Signal Processing Systems. *EURASIP Journal on Embedded Systems* , 2007 (1), 15 - 37.

[9] Jerraya, A.; Wolf, W. (2005). Multiprocessor Systems-on-Chips. San Francisco, CA: Elsevier Inc.

[10] Kai Huang; Sang-il Han; Popovici, K.; Brisolara, L.; Guerin, X.; Lei Li; Xiaolang Yan; Soo-lk Chae; Carro, L.; Jerraya, A.A.. (2007). Simulink-based MPSoC Design Flow: Case Study of Motion-JPEG and H.264. *DAC 07: Proceedings of the Design Automation Conference* (pp. 39 - 42). San Diego, California: ACM New York, NY, USA.

[11] Kangas, T.; Kukkala, P.; Orsila, H.; Salminen, E.; Hännikäinen, M.; Hämäläinen, T.D.; Riihimäki, J.; Kuusilinna, K. UML-Based Multiprocessor SoC Design Framework. *ACM Transactions on Embedded Computing Systems*, New York, Vol.5, No.2, p.281-320, 2006.

[12] Kopetz, H.; Obermaisser, R.; Salloum, C.E.; Huber, B. Automotive software development for multi-core system-on-a-chip. *In Proceedings of the 4th International Workshop on Software Engineering for Automotive Systems (SEAS'07)*, Washington, DC, USA, 2007.

[13] Kopetz, H.; Bauer, G.. The Time-Triggered Architecture. Proceedings of the IEEE, 91(1):126-113, January 2003.

[14] Marcellin, M. W., Gormish, M. J., Bilgin, A., & Boliek, M. P. (2000). An Overview of JPEG-2000. *Data Compression Conference (DCC)* (pp. 523-541). Washington, DC, USA: IEEE Computer Society.

[15] Pimentel, A. D.; Erbas, C.; Polstra, S., A systematic approach to exploring embedded system architectures at multiple abstraction levels, *IEEE Transactions on Computers*, vol. 55, no. 2, pp. 99–112, 2006.

[16] Shin, D.; Gerstlauer, A.; Peng, J.; Doemer, R.; Gajski, D. (2006). Automatic generation of transaction-level models for rapid design space exploration. *Proceedings of the 4th International Conference on Hardware/Software Codesign and System Synthesis* (pp. 64-69). Seoul, Korea: ACM, New York, NY, USA.

[17] Trajkovic, J.; Gajski, D.D. (2008). Custom Processor Core Construction from C Code. *6th IEEE Symposium on Application Specific Processors*, (pp. 1 - 6). Anaheim Convention Center, CA.

[18] Xilinx Embedded Development Kit[online]. (n.d.). Retrieved from http://www.xilinx.com/.

[19] Wallace, G. K.; (1992). The JPEG Still Picture Compression Standard. *IEEE Transactions on Consumer Electronics* , 38 (1), xviii - xxxiv.

[20] Wei, S. (2005). A FPGA-based Soft Multiprocessor System for JPEG Compression. Eindhoven: Technical University of Eindhoven, the Netherlands.

[21] Zhang, C.; Long, Y.; Kurdahi, F.; (2007). A scalable embedded JPEG 2000 architecture. *Journal of Systems Architecture: the EUROMICRO Journal* , 53 (8), 524-538.