Design Space Exploration for a Wireless Protocol on a Reconfigurable Platform

Laura Vanzago[1], Bishnupriya Bhattacharya[2], Joel Cambonie[3] and Luciano Lavagno[4]

[1]*STMicroelectronics, Advanced System Technology, via C. Olivetti 2, 20041 Agrate (MI), Italy*
[2]*Cadence Design Systems, 2670 Seely Ave. San Jose CA 95134, USA*
[3]*STMicroelectronics, Advanced System Technology, 12 rue Jules Horowitz, F-38019 Grenoble, France*
[4] *Politecnico di Torino, Corso Duca Degli Abruzzi 24, 10129 Torino, Italy*

Abstract

This paper describes a design space exploration experiment for a real application from the embedded networking domain - the physical layer of a wireless protocol. The application models both control oriented as well as data processing functions, and hence requires composing tasks from different models of computation. We show how the cost and performance of communication and computation can be quickly evaluated, with a reasonable modeling cost. While the example uses a specific tool, the methodology and results can be used in a more general context.

1 Introduction

The design of wireless protocols and their implementations on heterogeneous architectures, including dedicated IPs, programmable logic and one or more embedded processors, under a tight time-to-market pressure is a difficult task. One challenge is to provide validation and performance estimation of a given mapping of the protocol functionalities on the architecture without doing cycle accurate simulation at the RTL level. This enables a pragmatic approach to architecture and application performance optimization, striving to achieve both flexibility and efficiency, by running simulations with several testbenches. Despite the high level of abstraction used in the modeling of both the application and the architecture components (programmable and non-programmable), one must be able to evaluate performance indices such as throughput, latency, and resource usage that are essential to optimize the configurable platform parameters for the application at hand. The refinement of the initial specification onto the architecture must be performed in a unified framework that also addresses the problem of IP reuse both for functional and architecture IPs.

Our case study consists of the design of part of an ETSI standard protocol stack called Hiperlan/2 [1], specifically focusing on the data-dominated physical layer and its implementation onto a real heterogeneous architecture aimed at low power transceivers for wireless applications.

The adopted design methodology has the fundamental goal to quickly evaluate several architectural solutions that are available thanks to the reconfigurable nature of the target architecture. We address this problem by orthogonalizing the models of function, architecture, communication, computation and timing and by using a design framework that supports the above methodology and facilitates the model integration with the aid of a graphical user interface and design flow management support. This methodology, as proposed in [2] and [3], permits an efficient design space exploration based on IP reuse since each orthogonal concern can be optimized and refined separately from the others.

The application functionality is described as a network of sequential processes, Finite State Machines and dataflow actors, on top of a modeling and simulation infrastructure based on the C++ language. The communication between functional blocks is heterogeneous by mixing the Co-design Finite State Machine (CFSM) [2] and DataFlow (DF) models of computations [4], as dictated by the mixed control/data processing characteristic of the application. For this case study, we used the VCC™ tool from Cadence Design Systems [5], extended to model dataflow, but the method could be adapted to other application domains and tools.

In order to execute performance simulations, where the timing effects of a function/architecture mapping [2] are taken into account, we also used an original technique to model the mapping of an application on a complex abstract

1530-1591/03 $17.00 © 2003 IEEE

pipelined architecture of the datapath. Latency and throughput of the pipeline can be customized through the usage of parameters.

The behavioral communication arcs, both control (CFSM) and dataflow, are separately refined by describing the effect of the schedulers and communication protocols used in the architecture [6]. These communication behaviors are organized in libraries of C++ classes known as pattern and architecture services [7] that can be customized using various parameters (e.g. CPU clock speed, bus transfer speed, arbitration latency, interrupt response latency and so on) and that describe the communication at the transaction level of abstraction. The communication refinement methodology is based, as in SpecC [6] and SystemC [8], on the separation between interface and its implementation. In VCC, this programming paradigm, coupled with a clear separation between architecture, functionality and timing, enables a more general reuse methodology, supported by graphical editors, configuration management and so on.

Similar methodologies and design frameworks have been evaluated in the automotive [9], telecommunications [10], and multimedia [11] domains. However, some of these case studies (e.g. [9] and [10]) mainly focused on control and decision-dominated applications, that lend themselves well to the CFSM model of computation, based on lossy communication. Others, such as [11], required to manually introduce explicit queues among processes in order to specify the application using an extension of Kahn Process Networks [12].

Our case study is interesting because it shows the results that can be obtained on a real application, from the wireless networking domain, by using abstract functional and performance models to rapidly explore several design alternatives on a reconfigurable FPGA-based datapath. Due to the data-intensive nature of the application, we extended to the dataflow model of computation some performance analysis and communication refinement techniques that were previously applied to more reactive models. In addition, we defined a technique that allows to model timing delays of complex pipelined architecture components by preserving the original structure of both the functional model and of the hardware implementation.

The rest of the paper is organized as follows. Section 2 describes the functional modeling of the application where we used the mixed control/dataflow Model of Computation illustrated in Section 3. Section 4 details the architecture modeling and refinement of the transmitter network onto the architecture. Results are given in Section 5. Section 6 concludes with the insights gained from this project and charts some directions for future work.

2 The Hiperlan/2 Application

Our target application is the physical layer specification for High Performance Radio Local Area Network type 2 (Hiperlan/2) that is based on the Orthogonal Frequency Division Multiplexing (OFDM) modulation scheme. The physical layer is multi-rate in order to improve the radio link capabilities due to different conditions of interference and of distance between mobile terminals and the access point. i.e. the data rate, ranging from 6 to 54Mbit/s, is established by a link adaptation scheme and is varied by using several signal alphabets for modulating the OFDM sub-carriers. This property introduces an element of dynamic configuration that matches well the use of re-programmable components in the architecture. The Hiperlan/2 transmitter and a significant part of the Hiperlan/2 receiver were modeled, as shown in the behavioral diagram of Figure 1.

2.1 Modeling the Hiperlan2 Physical Layer

A careful definition of the block granularity is a key aspect of the mapping-based system-level design methodology, since functional blocks constitute the unit for partitioning over architectural resources. Moreover, they directly expose the parallelism in both computation and communication that can be exploited by the resources instantiated in the architecture.

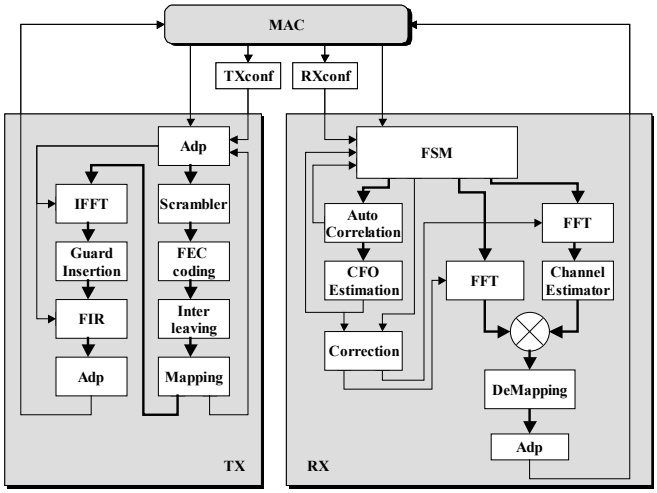


Figure 1. Hiperlan/2 : behavioral diagram.

Our architecture exploration process started with the import in VCC of the transceiver functional model of Figure 1, which had been initially developed and validated using a combination of C code and purely functional dataflow using COSSAP ™ (from Synopsys) [15]. The C procedures originally used to describe the transmitter (TX) and receiver (RX) have been partitioned with a low implementation effort. It essentially involved copying and pasting the data processing code into multiple C++ blocks, and re-defining some control functions, requiring about 150 additional lines out of about 2500 lines of untimed C++ behavioral code.

In particular, we separate the control from the data processing functions because they require different models of computation (CFSM and dataflow respectively) at their

interface. In addition, such separation allows us to evaluate different mappings onto specialized cores and onto the FPGA, as discussed in Section 5.

Figure 1 shows the transmitter and receiver networks organized as separate hierarchical blocks, together with a simplified model of the MAC. The model includes the blocks called *TXconf* and *RXconf* that control changes of modulation scheme or switch between transmission and reception modes. At the functional level they are not needed, but act as placeholders for delays that will be added during the refinement step. The function of the MAC model is to trigger the transmitter and receiver executions alternately and to configure the modulation scheme of the TX network, according to the Hyperlan/2 specification. The MAC uses the CFSM method to communicate with TX, RX and configuration blocks. TX and RX are modeled as dynamic dataflow networks.

3 Modeling of Dataflow Networks

In VCC, blocks can be modeled in C++ (ANSI C and hierarchical FSMs are also available) using a port-based interface model that offers a simple implementation-independent communication API. A block is activated if it receives a token on any one of its inputs. Every input port has a one-place buffer, which implies a lossy communication between blocks in the sense that it is possible for the sender to overwrite a token on the receiver's input port before the receiver had a chance to read the previous token.

In this project, we extended the simulator infrastructure to support the dataflow [4] Model of Computation, where blocks communicate with each other through FIFO (first in first out) channels. Each dataflow port has a data rate (the number of tokens consumed or produced by that port at each activation), and the block is activated only when all input ports have sufficient tokens (the firing rule is satisfied). Our DF prototype implements a general dynamic dataflow (DDF) semantics, where the firing rule of a block is not fixed at compile-time but can change at run-time. The delay on a dataflow channel (denoting some number of tokens initially present in the FIFO) can be specified as a parameter on a dataflow input port.

The implementation guarantees loss-less communication in functional simulation, assuming infinite- length FIFO channels or blocking write operations. For practical implementations onto architecture, the FIFO length for a channel can be specified as a parameter on the receiving input port during communication refinement.

The dataflow capability described above is provided in the form of a simple communication pattern service in the VCC library that models the sender and receiver sides of the DF channel. This pattern has to be instantiated on each communication arc that follows DF semantics. In this prototype, no attempt is made to perform static scheduling for SDF blocks, instead the block firing rule is checked dynamically for each DF block. Externally (e.g., to the simulator engine) a DF block appears just as a CFSM block, and all the DF-specific activities are performed by the DF pattern. Hence interfacing with CFSM blocks is naturally supported, resulting in an unconstrained mix-and match of CFSM and DF blocks, which is valuable for design space exploration. In our application, we have extensively used this mix-and-match feature to describe the heterogeneous aspects of the OFDM transceiver, composing together both control and data processing blocks that are naturally modeled by CFSM and DF semantics, respectively.

Such composition of CFSM and DF blocks is also possible in other existing tools like Ptolemy [13] and CoCentric System Studio™ (from Synopsys; formerly called El Greco) [14] The additional value of our approach is the possibility to execute performance simulation of a mixed–MoC design to evaluate the implementation onto a transaction based model of a real architecture. In fact Ptolemy and El Greco provide a more sophisticated hierarchical synchronous modeling framework including both dataflow, and finite state machines, where static (or quasi static) scheduling techniques are applied if possible. However, for the purpose of this case study, their quasi-static scheduling strategy and synchronous semantics makes the mapping of individual functional blocks onto a distributed heterogeneous architecture like ours problematic. In fact, depending on the mapping, the static or quasi-static scheduler of the simulator may conflict with the architecture implementation.

4 The Architectural Platform

In this project we mapped the Hiperlan/2 model onto a real reconfigurable and heterogeneous platform for low power transceivers used in wireless applications. It is specialized for an OFDM-based physical layer, but supports also the implementation of high-level protocol tasks on an embedded processor.

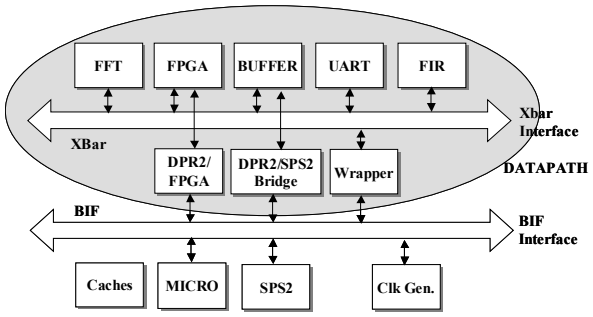


Figure 2. The architecture diagram.

Several cores are connected through a flexible communication resource, a crossbar bus, called XBar in Figure. 2. Some of the cores, i.e. FFT and FIR, implement computation-intensive functions as highly optimized IPs with limited range of programmability. Other cores, on the

other hand, are very flexible. An embedded low power FPGA [16] provides bit-level programmability, and a RISC micro-controller provides resources for dataflow management control functions, as well as for MAC functionalities. Each data item sent via the crossbar is associated with an attribute that describes which target it has to reach, and to which thread it belongs. Its arbiter uses a First Come First Served scheme with fixed priorities. Finally, a Request/Grant/Acknowledge protocol is used between the IPs to adapt the data flow to their respective computing speed.

The datapath is reconfigured dynamically between the transmission and reception phases, as well as between the transmission of several frames when a different modulation scheme is requested. The configuration mainly affects the FPGA and consists of either overwriting some internal registers, or downloading a new configuration stream from a dedicated memory.

4.1 Architectural Modeling

We modeled the datapath coprocessor of the architecture, including FFT, FIR, FPGA, memories and crossbar. Each component is described by an abstract API defining the services that it offers to the other architecture components and that impact the overall architecture behavior and performance. Those services describe, for example, transmission protocols, scheduling policies, storage delay access and capabilities, but not the functionality of each architecture element.

In this project we reused several service definitions [7] provided by the standard VCC library to describe memories, registers, schedulers, and data formatters. In addition, we have designed new service definitions to model the crossbar interconnection resource where delays like arbitration overhead, slave access delay and parallel accesses for each slave are assigned as parameters. The number of masters and slaves supported is computed from the architecture layout, thus resulting in a flexible re-usable model.

4.2 Mapping and Communication Refinement

The mapping of a function onto an architecture resource specifies a possible implementation, e.g. as hardware or software, and its performance cost in terms of estimated delays.

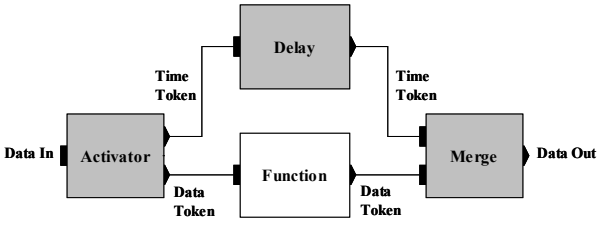
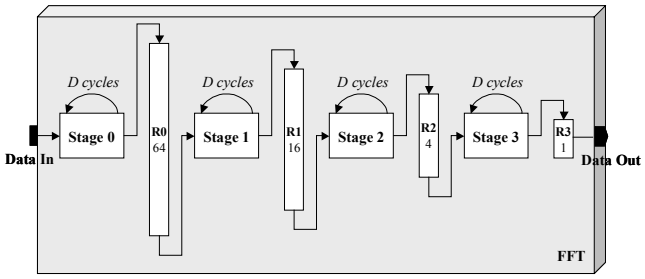


Figure 3. The delay wrapping technique used to model performance of a behavioral block.

Clearly, the quality of these delay estimates affects the precision of the overall architecture performance simulation done by the tool. In our case, all the peripherals are statically pipelined IPs. Thus, their delays can be easily estimated or derived from existent specifications or implementations.

However, modeling the performance of a complex functionality, whose netlist structure is quite different from that of the pipelined shared datapath, required the use of the mechanism shown in Figure 3 that is general for static pipeline implementation but doesn't cover the problem of data-dependent performances.

The mechanism involved modifying the functional netlist to describe separately the timing and the functionality of each block. The *Activator* and *Merge* blocks split the timing and data information (*Data* and *Time* tokens in the Figure 3), sending each to the appropriate sub-netlist. The *Function* block is a "normal" netlist modeling a FIR, FFT or FPGA untimed functionality. On the other hand, the *Delay* block models a skeleton of the pipeline architecture, as for example that of the FFT core represented in Figure 4, and it is used to play with various architectural options, such as the number of stages, the stalls, and so on.



*Latency: (64+16+4+1) * D*
Throughput: 1/D

Figure 4. FFT skeleton pipeline.

We also defined some new patterns to represent Shared Memory and Register Direct communications, as required by the various possible architectural implementations of the data-flow communication pattern used in the functional simulation. The refinement consists of the redefinition of the implementation of the I/O interface functions by generating transactions record for debugging and analysis and bus access requests for performance analysis.

5 Results

This section presents the results of some explorations that we performed, using the architecture model of the datapath shown in Figure 2, by evaluating different mapping scenarios of the transmitter application shown in Figure. 1. The functional mapping is showed in Figure 5(a) where most of the transmitter blocks are hierarchical resulting in a one to one or many to one mapping with the architecture cores. The explored design space is specified in Figure 5(b). We report on the global latency and throughput

effects due to architectural choices such as memory configuration (single port versus multiple ports), communication refinement (direct connect versus shared memory), and FPGA configuration strategies.

A similar exploration is possible in theory with a RTL-level simulation also, but the cost, in terms of model development and modification time, would be unacceptable. The methodology and tool used in this case study, on the other hand, required only a few hours to model and simulate each design space point.

The timing views for the functionalities mapped to the cores and the performance parameters values of memories and interconnection services were derived from the IPs documentation and RTL specifications.

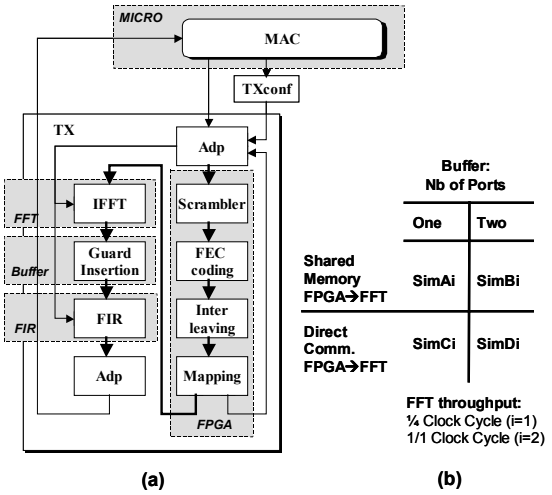


Figure 5. (a) Mapping (b) Design space exploration.

Thus the accuracy of our model, in terms of number of clock cycles with respect to the "golden" RTL model available for one of the analyzed scenarios (SimA1), is better than 10%.

The difference originates mainly from the fact that the RTL model uses the microprocessor for feeding the FPGA, while in VCC the microprocessor is not modeled.

The simulation is also very fast. Each test case described below requires between 4 and 26 seconds to simulate the transmission of 6 OFDM symbols (depending on how many probes are added to the mapping for instrumentation) on a Pentium III 600 MHz machine.

However, the main advantage of the methodology is not the speed of the simulation (about three orders of magnitude slower than real time), but the speed of mapping and refinement changes in order to explore the design space. This dramatically reduces the precious time devoted by a designer to this task with respect to RTL modeling.

5.1 Communication Refinement

In Figure 5(b) we show the design space covered by our first exploration. In this case, we vary the communication refinement between FPGA and FFT by using shared memory or a direct connection through the bus. For each case we also change the access mode to the memory (one or two access ports with the crossbar). All cases are then evaluated by using two different FFT throughput estimates (that correspond to two different FFT architectures).

The results are given in Figure 6. The chart shows the number of clock cycles (y-axis) for which each core is busy in each mapping scenario (x-axis). A balanced load, such as in SimC1 is generally better than an unbalanced one, such as in SimA1.

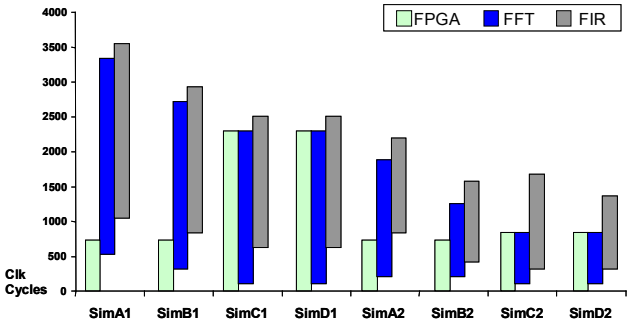


Figure 6. Results of design space exploration.

We verified that increasing the number of access port to the memory through the crossbar (SimAi/SimBi or SimCi/SimDi) does not result in a significant bit rate improvement.

From these charts, assuming a clock rate of 70 MHz, we can evaluate the datapath bit rate for a given mapping (Table 1). Those rates show that at full stream speed not all the mappings are compliant with the physical layer Hiperlan/2 specification, which requires a bit rate in excess of 12 Mbit/sec. However, each one shows different characteristics that may make it more or less desirable for other application families, such as low bit rate not-standard radio. For instance, the scenarios involving the fastest FFT architecture result in the highest bit rate at the price of a larger area and power consumption estimate.

The system model today provides only latency/throughput information, but the architectural services could be extended to provide also area and activity (power) information.

Table1. Datapath bit rates (Mb/sec) evaluated for different mappings.

Sim run	A1	B1	C1	D1	A2	B2	C2	D2
Bit Rate	5.8	7.2	8.2	8.2	9.6	13.7	12.5	15.6

5.2 FPGA Alternatives

The next exploration shows that it is possible to evaluate the cost of adding a new function to the algorithm, and explore two implementation scenarios involving dynamic embedded FPGA reconfiguration.

Starting from case SimB2, we expanded the design space by inserting a new function, called windowing, in the

behavioral network between the cyclic guard and the FIR functions. This function is used to significantly decrease the dynamic range of the OFDM signal before transmission.

We modified the functional network as shown in Figure. 7(a). Windowing must act only on the first and last ten elements of each OFDM symbol. The data flow between Guard and Fir is thus controlled by the dynamic dataflow blocks Mux and Demux while the FPGAConf block is added as a placeholder for the FPGA reconfiguration delay.

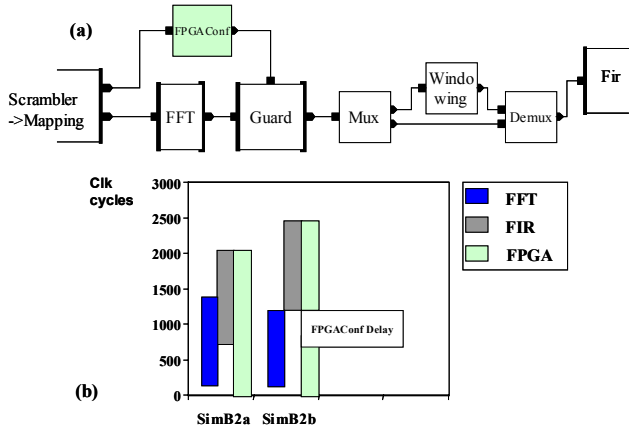


Figure 7. Design space exploration with Windowing, (a) functional network; (b) simulation results.

As shown in Fig. 7(b), we executed two different simulations by using estimated performance delay for windowing mapped on FPGA. In the first case (SimB2a), we assumed to have an embedded FPGA that was large enough to support simultaneously a configuration with mapping and windowing, that provides maximum parallelism, but little area efficiency.

In the second case (SimB2b) we assumed only one smaller embedded FPGA, which must be reconfigured on the fly to switch from the mapping function to the windowing function. This creates a configuration cost (number of cycles to reconfigure the FPGA) in the performance evaluation that results in a measurable bit rate decrease. In Fig. 7(b) we show the performance obtained when the reconfiguration cost is 500 clock cycles. Thus, in this exploration, we traded-off the size of the FPGA against the performance of the system.

6 Conclusions

We applied a design methodology, based on separating functionality from architecture, and communication from computation, to a real application and architectural platform from the wireless networking domain. During this experiment, we also extended the capabilities of the used tool to describe the mixed control/dataflow nature of the application. We applied a methodology that allows one to refine behavioral dataflow communications onto bus transactions in the architecture. We also used an intuitive method to describe timing views for functions mapped to pipelined hardware.

Further project developments will include a detailed functional model of the MAC layer and of the synchronization stage of the receiver, followed by their mapping on the microprocessor.

Acknowledgements. The Designers of the Central R&D STMicroelectronics Berkeley Lab and the Engineers of System Level Design Group from Cadence Design Systems for the valuable support.

7 References

[1] ETSI TS 101 475. Available at http://www.etsi.org

[2] F. Balarin, et al. “HW/SW Co-Design of Embedded Systems: the Polis Approach,” Kluwer Academic Publisher, 1997.

[3] B. Kienhuis, et al. “An Approach for Quantitative Analysis of Application-specific Dataflow Architectures,” *Proc. IEEE Int. Conf. On Application-Specific Systems, Architectures and Processors, 1997*.

[4] E.A.Lee, D.G.Messerschmitt, “Pipeline Interleaved Programmable DSP’s: Synchronous Dataflow Programming,” *IEEE Trans. on Acoustics, Speech, and Signal Processing, 1987*.

[5] http://www.cadence.com/technology/hwsw/ciertovcc

[6] D.Gajsky, J.Zhu, R.Domer, A.Gerstlauer, S. Zhao. “The SpecC Methodology”, UC Irvine – TR ICS-TR-99-56, 1999

[7] S.Solden, “Architectural Services Modeling for Performance in HW-SW Co-Design,” *Proc. SASIMI, Japan, 2001*.

[8] http://www.systemc.org

[9] T. Demmeler, et al. “Enabling Rapid Design Exploration through Virtual Integration and Simulation of Fault Tolerant Automotive Application,”, *SAE, 2002*.

[10] J.L. Da Silva, et al. “Wireless Protocols Design: Challenges and Opportunities,” *Proc. Int. Workshop on HW/SW Codesign, 2000*.

[11] E.A. de Kock, et al. “YAPI: Application Modeling for Signal Processing Systems,” *Proc. DAC 2000*.

[12] G. Kahn, “The semantics of a simple language for parallel programming”, *Information Proc., J.L. Rosenfeld, Ed. North Holland Publishing, Co.,1974*

[13] J.T. Buck, et al. “Ptolemy: A Framework for Simulating and Prototyping Heterogeneous Systems,” *International Journal of Computer Simulation, 1994*.

[14] J. T. Buck,, R.Vaidyanathan, “Heterogeneous Modeling and Simulation of Embedded Systems in El Greco,” *Proc. Int. Workshop on HW/SW Codesign, 2000*.

[15] http://www.synopsys.com/products/dsp/dsp.html

[16] V. George, H. Zhang, J. Rabaey. “The design of a Low Energy FPGA”, *Proc. Int. Symposium on Low Power Electronics and Design, 1999*.