TLM/Network Design Space Exploration for Networked Embedded Systems

N. Bombieri, F. Fummi, and D. Quaglia
Dipartimento di Informatica, Università di Verona
15 Strada le Grazie, I-37134, Verona, Italy
bombieri@sci.univr.it, [franco.fummi|davide.quaglia]@univr.it *

ABSTRACT

This paper presents a methodology to combine Transaction Level Modeling and System/Network co-simulation for the design of networked embedded systems. As a result, a new design dimension is added to the traditional TLM refinement process to represent network configuration alternatives. Each network configuration can be used both to drive architecture refinement and exploration and to validate the system after each refinement step. A general criterion to map functionalities to System and Network models is presented. As a case study, the proposed methodology is applied to the design of a Voice-over-IP client.

Categories and Subject Descriptors

I.6.8 [**Simulation and Modeling**]: Types of Simulation—*Combined*; C.0 [**Computer Systems Organization**]: General—*Systems specification methodology, System architectures*

General Terms

Design, Performance

Keywords

Networked embedded systems, Transaction-level modeling

1. INTRODUCTION

Networked embedded systems represent a considerable portion of the number of embedded systems as they range from network intermediate systems (e.g., routers and access points) to mobile phones and wireless sensor networks; for this reason their fast and efficient design is ever more strategic for manufacturers.

Transaction Level Modeling (TLM) is becoming a usual practice to simplify system-level design and architecture exploration. It allows the designers to focus on the functionalities of the system, while abstracting away implementation details that will be added at lower abstraction levels [3, 9]. Traditional TLM approach consists of three levels as depicted in Fig. 1.a. SystemC well supports the TLM approach since it provides a set of primitives at different abstraction levels ranging from function calls and packet transfers to ports and signals [7, 10].

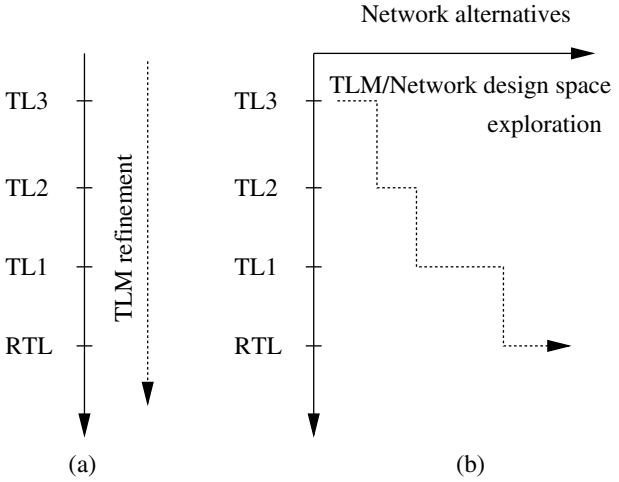

Figure 1: Traditional TLM refinement (a) and the proposed TLM/Network design space exploration (b).

Joint HW/SW/Network modeling and simulation of networked embedded systems has been shown to be an efficient approach to improve design quality and shorten time-to-market [5, 6]. In those works, SystemC has been used to model HW and SW parts of the system while a well-known network simulator –NS-2 [8]– has been used to model the external network. Some changes have been done to the simulator kernels to perform a synchronized simulation.

However, some points have not been clarified: *1)* the decision of which components should be modeled in SystemC and which in the network simulator, *2)* the connection of system components at different ISO/OSI network layers, and *3)* the relationship between the TLM approach and the System/Network co-simulation.

The focus of this paper is to investigate these points and to provide a methodology which integrates Transaction Level Modeling and System/Network co-simulation. The result is

*Research activity partially supported by the European project FP6-2005-IST-5-033506 - ANGEL

CODES+ISSS'06, October 22–25, 2006, Seoul, Korea.

a two-dimension design space as depicted in Fig. 1.b. The vertical dimension addresses both the refinement of the system model and the optimization of its algorithms; during this process the network model is used both to drive architecture exploration and to verify that communication requirements are met. The horizontal dimension represents the design space of the network model in which different topologies and parameters can be varied. In the paper, these issues will be addressed both from a theoretical point of view and by using a case study related to the design of a Voice-over-IP client.

The paper is organized as follows. Section 2 introduces the problem of modeling a networked embedded system by presenting Transaction Level Modeling and Network modeling. Section 3 describes a novel methodology which combines SystemC and NS-2 for the joint simulation of the System and the Network, respectively. In particular, some methodological criteria, not addressed by the previous literature, are described. In Section 4 such criteria are applied to the design of a Voice-over-IP client. Finally, conclusions are drawn in Section 5.

2. NETWORKED EMBEDDED SYSTEMS

Networked embedded systems (NES's) are HW/SW systems designed to perform specific applications in which network communications are essential. Their design requires the capability of modeling and simulating both their behavior/architecture (the *System*) and the complex communication environment in which they operate (the *Network*).

The *System* usually consists of a CPU, a memory to store application code and data, one or more network interfaces, I/O interfaces for data acquisition and user interaction, and other components –ASIC's or FPGA's– designed to efficiently perform specific functions. An application-specific SW is deployed over this HW architecture often together with an operating system which bridges HW and SW components. Among system modeling languages, SystemC [7] is gaining increasing attention for its great flexibility in describing devices at different abstraction levels and for its interoperability with other languages, e.g., VHDL.

The *Network* consists of a set of nodes connected to the System through communication links. Different parameters can be defined for each link, e.g., the type of channel (wired/wireless), the bandwidth, and the delay. Even if general purpose languages can be used to reproduce the behavior of the network, specific network simulators, like NS-2 [8], already provide models for well-known network protocols, e.g., Ethernet, WiFi, and TCP/IP.

2.1 Transaction-level modeling of the System

Transaction Level Modeling (TLM) [3, 9] aims at standardizing the refinement process of the System model to enable re-use between abstraction levels within the same project and between projects belonging to different manufacturers. Three abstraction levels are defined. At the highest level (TLM level 3 – TL3) a functional specification is created to provide a proof of concept. At this level, it is not yet determined which modules will be implemented in HW and which in SW. A first functional partitioning between data and control is performed. Data transfers are described through abstract types and point-to-point communications between modules. The behavior is modeled as a set of un-timed event-driven function calls as in common application SW. In the next refinement level (TL2) timing performance is analyzed and a first HW/SW partitioning is performed according to several constraints (e.g., performance, cost, and component availability). The abstract architecture is mapped into a set of interconnected resource-constrained blocks. Data transfers are described in terms of bit width and message size in order to estimate bus bursts. Pipelined structures are introduced by splitting complex operations into a timed sequence of simpler operations. At the lowest TLM level (TL1) SW blocks are implemented and interfaces of HW blocks are defined (even if pins are still hidden). A bus model is introduced and clock-accurate protocols are mapped to the chosen HW interfaces and bus structure; transactions are mapped directly to bus cycles.

Modules belonging to different TLM levels, or even to RTL, cannot communicate directly since data transfers are described at different levels of detail. For this reason and to enable module re-use, *transactors* have been introduced in the TLM approach [4].

2.2 Network modeling

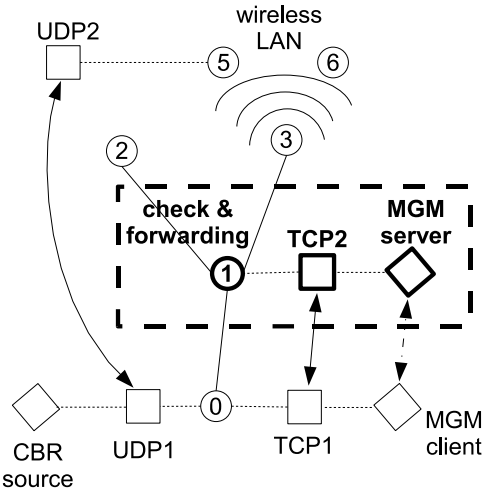

Figure 2: Simple network modeled in NS-2.

Network simulators reproduce the functional behavior of protocols, manage time information about transmission and reception events and simulate packet losses due to congestion or link failure.

Network Simulator, NS-2 [8], is the most widely used discrete event simulator for computer networks. It is implemented in C++ and its source code is open. It is widely used in many research activities because it includes modules for the simulation of well-known protocols both wired and wireless [12]. The modeling approach of NS-2 follows the well-known *ISO/OSI reference model*, i.e., a layered architecture in which each layer provides services to the upper layer and uses services of the lower layer.

Fig. 2 shows a simple network described by using NS-2 entities. Round entities are called *Nodes*; they are connected together by wired links (continuous lines) or wireless links as in case of `Node` 3, 5, and 6. Nodes and links reproduce the lowest three ISO/OSI layers and, in particular, `Node 1` reproduces the behavior of a router for what concerns packet forwarding. Square entities are called *Agents* and represent

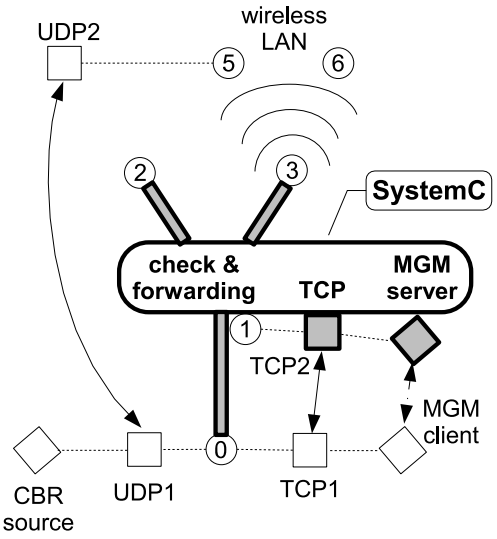

Figure 3: System/Network co-simulation applied to the design of an Internet router.

the Transport Layer; they are attached to nodes and connected together to reproduce end-to-end UDP or TCP connections (continuous lines with arrows). To complete the design of the router, on top of `Node 1` we put `Agent TCP2` which reproduces the TCP agent implemented in the operating system of the router (e.g., to support a remote shell). Rhomboidal entities are called *Applications* and reproduce application layer sessions (dotted arrows) which inject packets into the network. Application `MGM server` reproduces the router management service which can be implemented on top of its TCP/IP stack. *Nodes*, *Agents* and *Applications* are connected together by object references in the network simulator (dashed lines). Functionalities related to the router are grouped by the dashed box.

The set of entities (`CBR source`, `UDP1`, `UDP2`) represents a constant-bit-rate flow of UDP packets (e.g., a Voice-over-IP conversation). According to the routing rules, packets are delivered through the wired link between `Node 1` and `3` and the wireless link between `Node 3` and `5`. The set of entities (`MGM client`, `TCP1`, `TCP2`, `MGM server`) represents a remote management session build upon a TCP connection. It is worth to note that the router can be considered as an intermediate system for what concerns packet checking and forwarding and as an end system for what concerns TCP and management services.

3. SYSTEM/NETWORK CO-SIMULATION

In this Section we present a co-simulation methodology based on a timing-accurate integration of SystemC and NS-2. SystemC executable models and NS-2 have conventional event-driven kernels; they schedule the execution of events in non-decreasing order of timestamp. In SystemC, events are associated to read and write operations on ports, while in NS-2 events are associated to packet transmission and reception between nodes, agents, and applications. In our work we follow the approach from [5] in which modified versions of SystemC ports and NS-2 entities allow the interaction between the corresponding models. For example, a packet can be moved from a SystemC module to an instance of the special NS-2 link by writing it on a special output port. Conversely, a packet arrived to the link from the network is delivered to a special SystemC input port and triggers the associated method as traditional write operations. Both kernels have been modified to interact each other to reach a global synchronization of events and to exchange data.

Various issues should be addressed in this context. First of all, the designer should decide which functionalities have to be modeled by the System tool and which by the Network tool; this decision, called *System/Network partitioning*, should be driven by methodological criteria as explained in Section 3.1.

The second issue concerns the decision of which kind of NS-2 entity should interact with a given SystemC module. As explained in Section 2.2, NS-2 provides different kinds of entities according to the ISO/OSI levels. Referring to the example of Fig. 3, the SystemC module implementing the IP forwarding functionality should receive from NS-2 a packet reporting the layer-3 destination address and, therefore, it should interact with a layer-3 entity. In previous works [5, 6] interactions between system and network models took place only at the transport layer preventing the use of SystemC to model components at different ISO/OSI layers. This issue will be addressed in Section 3.2.

The third issue concerns the relationship between the TLM approach and System/Network co-simulation. The validation of the refinement process should be provided. Furthermore, the interoperability between TL3 or TL2 un-timed models with timed network components should be provided as in case of interconnection of components at different TLM levels. Furthermore, the TLM refinement process can lead to a change of the network counterpart. Referring to the router example, the full implementation of the packet forwarding function requires a bit-accurate simulation of transmission over the network link. These issues will be addressed in Section 3.3 and in Section 4.

3.1 System/Network partitioning

In this Section we deal with the the subdivision of functionalities that should modeled in the System and in the Network. The general criterion is that functionalities to be implemented in the NES have to be described as components of the System model instead of the Network model. Conversely, the network simulator should be used to describe those functionalities which are not part of the NES design process but represent the environment in which it will operate. In this context, the use of third-party components provided by the network modeling tool not only speed up the design process, but also contribute to validate the designed communication structure by testing its interaction with an abstract protocol specification shared by the research community.

In Fig. 4 different approaches for System/Network partitioning are shown. Two end systems are represented by the corresponding protocol stacks; for simplicity's sake each stack is reduced to an upper layer and a lower layer represented by boxes labelled with "U" and "L", respectively. Each approach has its own advantages and drawbacks which should drive its adoption. In Fig. 4.a the SystemC simulator is used not only to model components but also to simulate network communications. In this case all TLM levels are supported since the model of the network can be both timed and un-timed; furthermore, the use of a single

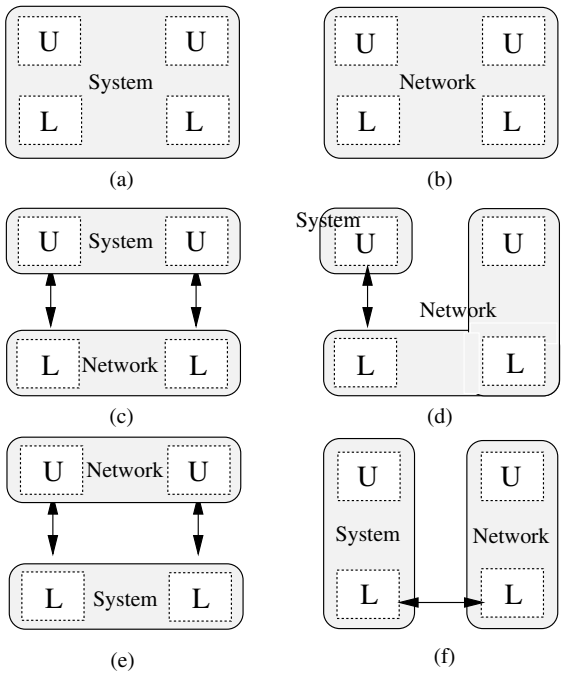

Figure 4: Different possible approaches for System/Network partitioning.

tool improves simulation speed. The main drawback is that all network protocols should be re-implemented by the designer. In Fig. 4.b only the network simulator is used as in the example reported in Fig 2. Also in this case the use of a single tool improves simulation speed. Network modeling tools describe functionalities without reproducing the interaction among different components *within the single node* as in actual systems; this fact limits the reusability of the functional description in the successive design phases. Instead, system modeling tools have the advantage that the model can be refined, i.e., transformed from a behavioral to a structural description, and traditional validation techniques can be applied to it.

In the other cases both simulators, SystemC and NS-2, are used and arrows represent interactions between them. The common drawback of these approaches is that synchronization increases simulation time. In Fig. 4.c the upper layer of the NES is the focus of the design process and thus it is modeled by SystemC. The interaction of two instances of this component takes place through the corresponding lower layer entities modeled by the network simulator. The designer takes advantage by the use of a specific tool for network simulation considering that such lower entities are outside the design scope. Also in Fig. 4.d the focus of the design process is the upper layer of the NES but, for validation purpose, a SystemC instance of the component interacts with a peer entity modeled by the abstract protocol specification of the network simulator. In this case, the generation of test patterns is simplified by the use of a specific tool for the reproduction of network behavior. In Fig. 4.e the focus of the design process is the lower layer and therefore it is modeled by SystemC which also reproduces the communication channel. The upper layer is modeled by the network simulator thus simplifying the generation of test application models which are outside the design scope. Finally, in Fig. 4.f both the upper and lower layers belong to the design process and thus they are modeled by SystemC and validated by the reference stack specification of the network simulator.

3.2 Interactions at different ISO/OSI layers

Fig. 3 presents a co-design approach for the example of Fig. 2. The functionalities highlighted in the SystemC implementation of the router belong to different ISO/OSI layers and should exchange information with the corresponding entities in NS-2. For this reason, three new entities have been added to the network simulator, i.e., the `ns_sc_link`, the `ns_sc_agent`, and the `ns_sc_app`; in Fig. 3 they are represented by the shaded shapes.

The `ns_sc_link` connects NS-2 *Nodes* with SystemC modules whose functionalities belong to the lowest three ISO/OSI layers. This kind of entity conveys layer-3 addresses and bit-accurate packet descriptions. The `ns_sc_agent` connects NS-2 *Agents* with SystemC modules whose functionalities belong to the transport layer. This entity conveys transport addresses and acknowledgements (in case of TCP connections). The `ns_sc_app` connects NS-2 *Applications* with SystemC modules whose functionalities belong to the application layer. This entity conveys the application content of network transmissions.

In NS-2 the *Link* simply reproduces a transmission channel, while the *Node* is a more complex entity which also contains the rules for channel access; therefore, when a new kind of network is introduced in NS-2 (e.g., wireless networks), the *Node* should be extended while the *Link* remains unchanged. For this reason, to maintain compatibility with the future releases of NS-2, we decided to modify the *Link* entity.

3.3 Verification of the refinement process

System/Network co-simulation allows to apply the main verification techniques to check the correctness during the refinement flow. Functional verification based on assertions represents the main verification technique joining dynamic and static verification [1]. In the Assertion Based Verification (ABV), assertions are the central focus of the verification process; they detect bugs and guide testbenches in the stimuli production. An assertion is a precise description of the expected behavior when a given input is presented to the model. ABV raises the level of verification from RTL to transaction level where tests are closer to the design specification. Consequently, design functions are efficiently exercised (with minimum time expense) and effectively monitored by detecting hard-to-find bugs. Several approaches based on Transactor-based Verification (TBV) and assertions have been recently proposed from both EDA companies and academic researchers [4, 2]. In these approaches, transaction-level testbenches and assertions can be directly reused on the RTL implementation through transactors thus avoiding time-consuming and error-prone manual conversions.

4. CASE STUDY

The methodological criteria described in the previous Section have been applied to a case study consisting in the design of a Voice-over-IP (VoIP) client. Fig. 5 shows the block diagram of the whole System/Network scenario. The VoIP

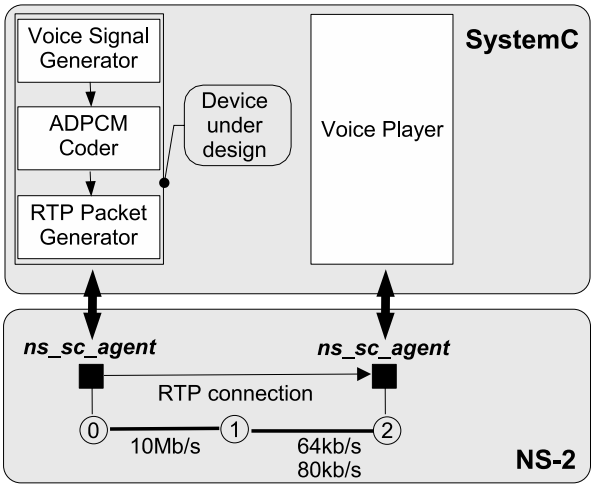

Figure 5: Block diagram of the case study: the upper gray box represents the System model at the highest abstraction level (TL3) while the lower gray box represents the Network model.

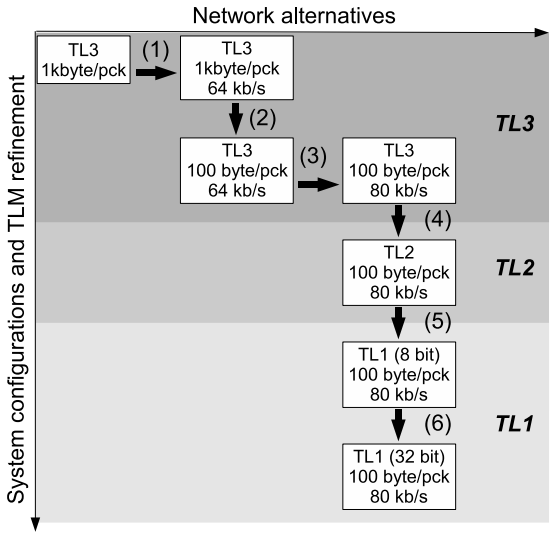

Figure 6: Design flow in the 2-dimension space represented by System and Network modeling.

client captures voice (16 bit/sample at 16 ksample/s), reduces its rate from 256 kb/s to 64 kb/s by using an ADPCM compression scheme, and then sends the encoded bitstream as a sequence of Real-time Transport Protocol (RTP) [11] packets to the voice player for decoding and reproduction; the payload size is a design parameter as explained below. The Voice Player is an abstract functional model of receiver used to test the client under design.

For what concerns System/Network partitioning, the approach of Fig. 4.c has been chosen, i.e., the application and transport levels are embodied in the System while the lowest three ISO/OSI layers are modeled in NS-2 together with the communication channel. In fact, in this example, the focus of the design process are the highest ISO/OSI layers, i.e, the compression scheme and the creation of packets. Since NS-2 interacts with SystemC modules at transport level, instances of the `ns_sc_agent` are used.

4.1 Model of the network

In the lower part of Fig. 5 the NS-2 model of the network is depicted. `Node 0` represents the lowest three layers of the VoIP client under design. The 10 Mb/s links between `Node 0` and `Node 1` models a traditional high-capacity LAN while the link between `Node 1` and `Node 2` models a geographical backbone with constrained capacity. Since a minimum end-to-end bandwidth is required for the correct behavior of the VoIP client, the capacity of this link is set to such minimum to validate the system.

Voice packets generated by SystemC flow through `Node 0` to `Node 2`. Statistics on delay and packet drops are generated by NS-2 and used to evaluate the quality of service as a function of different design choices.

4.2 TLM/Network design space exploration

Fig. 6 shows the System/Network design space exploration in the 2-dimension space represented by System configurations and Network alternatives; different System/Network configurations are represented by boxes. A change of the System model leads to a vertical shift while a change of the Network model leads to a horizontal shift. First of all, System/Network partitioning is performed on the global scenario (Step (1)); the approach of Fig. 4.c has been chosen as explained before.

At the beginning of the design flow each packet contains 1000 byte of the encoded bitstream leading to a bitrate of about 64 kb/s. According to the statistics provided by NS-2, packets enter the network every 0.125 s and they reach destination after a further delay of 0.168 s. Since this end-to-end delay is not acceptable for an interactive voice application, the System has been modified to reduce the payload size to 100 byte (Step (2)). In this case, a feedback from the network is used to improve system design. These smaller packets enter the network every 0.0125 s and they reach destination after a further delay of 0.055 s; however, with a bottleneck capacity of 64 kb/s, the increased header overhead leads to a packet loss rate of 16.6%. For this reason the minimum capacity required on the bottleneck link is increased to 80 kb/s leading to an end-to-end delay of 0.065 s (Step (3)). In this case, a change in the system design leads to a change in the network model.

Step (4) refers to the TLM refinement step from TL3 to TL2. The left part of the System model depicted in Fig. 5 represents the model of the VoIP client at the highest TLM level, i.e., TL3. The un-timed functionalities are modeled regardless to the implementation architecture and system blocks communicate through function calls. At TLM Level 2 (TL2) HW/SW partitioning is performed; the Voice Signal Generator and the RTP Packet Generator becomes SW modules executed by the CPU while the ADPCM module is implemented in HW. A high level model of the bus is inserted into the System and a flat proof of communication protocol is implemented. Some architectural details are added in order to evaluate a first communication scheme considering, for example, bus sharing, pipeline and FIFO channels.

Step (5) refers to the TLM refinement step from TL2 to TL1 in which a bus model is described and a clock-accurate protocol is mapped to the chosen HW. Figure 7 shows the

Level of detail	# sc_module	# sc_port	processes	code rows	clk cycles/pck	CPU time (s)
TL3 (1 kbyte/pck)	5	3	1	1320	1	0.03
TL3 (100 byte/pck)	5	3	1	1320	1	0.03
TL2	6	17	5	1651	1	0.03
TL1 (8 bit)	12	24	12	5759	35,280	19.57
TL1 (32 bit)	12	26	12	5953	8,400	5.24

Table 1: Statistics about the System model at different stages of the design flow.

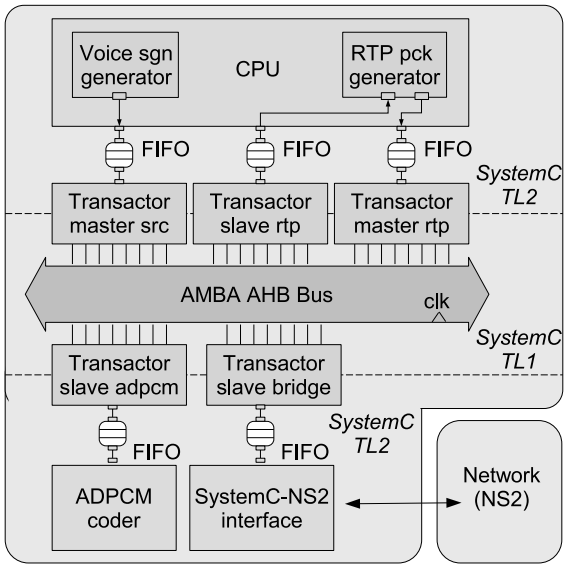

Figure 7: Architecture of the System at the lowest Transaction Level (TL1).

architecture of the System at TL1. The AMBA AHB Bus is chosen as communication channel and its SystemC implementation replaces the previous high level bus representation.

Architecture exploration can be performed at TL1 to reach the optimal trade-off between cost/complexity and performance. In particular we tested 8 and 32 bit bus configurations (Step (6)).

4.3 Experimental results

Table 1 reports System-related statistics at different stages of the design process depicted in Fig. 6. Since these statistics refer to the System model, we have not reported configurations which change only for network parameters. The first four metrics relate to the complexity of the SystemC model, i.e., the number of sc_modules, sc_ports, processes, and code rows. The fifth metric measures the performance of the System in clock cycles for each transmitted packet; since the clock signal is not present at TL3 and TL2, in the first three rows of the table the value of this metric is set to 1 clock cycle. The last metric reports the CPU time spent to simulate 10 s of transmission on Intel Xeon at 3.06 GHz with 4 GB of RAM and Linux operating system. Values for this metric have been obtained by running the SystemC executable model with the `time` command and summing up user time with system time. The CPU time of the concurrent network simulation is about 5 s in case of packets of 1 kbyte and about 34 s in case of packets of 100 byte.

5. CONCLUSIONS

A new methodology has been presented to combine the traditional Transaction-Level Modeling with system/network co-simulation. The result is a two-dimension design space in which one dimension addresses the exploration of system configurations and TLM refinement of the SystemC model while the other represents the space of network configurations represented by NS-2, a well-known network simulator. Feedbacks coming from one domain contribute to refine the model in the other domain. During the design process the network model is used both to drive architecture exploration and to build testbenches for the validation of the system after each refinement step; also network models can be optimized according to the design choices on the system part. The potentialities of the proposed approach have been shown through a case study related to the design of a Voice-over-IP client.

6. ACKNOWLEDGMENTS

The authors would like to thank Dr. Maura Turolla, of Telecom Italia Lab, for providing specifications about the application scenario.

7. REFERENCES

[1] A. Dahan et al. Combining system level modeling with assertion based verification. In *Proc. IEEE ISQED*, pages 310–315, Mar. 2005.

[2] N. Bombieri, A. Fedeli, and F. Fummi. On PSL properties re-use in SoC design flow based on Transaction Level Modeling. In *Proc. IEEE MTV*, 2005.

[3] L. Cai and D.Gajski. Transaction Level Modeling: An Overview. In *Proc. IEEE Int. Conf. on Hardware/Software Codesign & System Synthesis*, pages 19–24, Oct. 2003.

[4] D.S. Brahme et al. The Transaction-Based Verification Methodology. Technical Report CDNL-TR-2000-0825, Cadence Berkeley Labs, Aug. 2000.

[5] F. Fummi et al. A timing-accurate modeling and simulation environment for networked embedded systems. In *Proc. ACM Design and Automation Conf. (DAC)*, pages 42–47, Jun. 2003.

[6] F. Fummi et al. Heterogeneous co-simulation of networked embedded systems. In *Proc. IEEE Design Automation and Test in Europe Conf. (DATE)*, pages 168–173, Feb. 2004.

[7] T. Grotker, S. Liao, G. Martin, and S. Swan. *System Design with SystemC*. Springer, 2002.

[8] S. McCanne and S. Floyd. NS Network Simulator – version 2. *URL: http://www.isi.edu/nsnam/ns*.

[9] S. Pasricha, N. Dutt, and M. Ben-Romdhane. Extending the Transaction Level Modeling approach for fast communication architecture exploration. In *Proc. ACM Design and Automation Conf. (DAC)*, pages 113–118, 2004.

[10] A. Rose, S. Swan, J. Pierce, and J.-M. Fernandez. Transaction Level Modeling in SystemC. Technical report, Cadence Design Systems, Inc, 2004.

[11] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. *RFC 3550*, July 2003.

[12] H. Zhu and I. Chlamtac. Performance analysis for IEEE 802.11e EDCF service differentiation. *IEEE Transactions on Wireless Communications*, 4(4):1779–1788, July 2005.