From TLM to FPGA:

Rapid Prototyping with SystemC and Transaction Level Modeling

Wolfgang Klingauf, Robert Günzel
Technical University of Braunschweig, Dept. E.I.S., Germany
[klingauf, guenzel]@eis.cs.tu-bs.de

Abstract

We describe a communication-centric design methodology with SystemC that allows for efficient FPGA prototype generation of transaction level models (TLM). Using a framework comprising of well-defined communication protocols and synthesizable communication wrappers, the process of refining the TLM specification of a HW/SW system to its synthesizable implementation can be systematically automated. We look at how to map the TLM communication channels of both HW and SW components to virtually any target platform and illustrate our approach on refining an example HW/SW system from its TLM specification to a Virtex-II Pro FPGA implementation.

1. Introduction

As embedded systems become more and more ubiquitous, the thorough verification of the system functionality is often not possible until a fully workable prototype has been built. This is particularly true for networked embedded systems in a highly distributed and heterogeneous environment, such as an in-vehicle network. Based on the transaction level modeling paradigm, recently various methodologies have been proposed to enable fast design space exploration with TLM. However, most techniques focus on communication analysis [1, 4]. First approaches for TLM-based MPSoC synthesis have as well been presented [5], but several design steps such as HW/SW interface generation still require a large manual effort.

In this paper, we outline a methodology that adopts the TLM design paradigm for efficient rapid prototyping of MPSoC. The goal is to guide the system engineer through a highly systematic design process, which on the one hand enables fast communication architecture exploration and early embedded software development, and on the other hand fills the communication refinement gap between TLM and synthesizable RTL.

2. Communication refinement

When proceeding through TLM abstraction layers towards a synthesizable implementation, the system engineer has to perform a variety of refinement steps. While for fast design space exploration at a high level of abstraction, communication is modeled by simple send and recv method calls, in the less abstract TLM models transactions split up into various phases. By providing a TLM modeling fabric to the system engineer which clearly associates each TLM abstraction layer to a well-defined communication protocol [2], we can automate these refinement tasks using generic algorithms.

Figure 1. Mixed-mode TLM model of a HW/SW system with CASM

The refinement process starts with the insertion of one or more communication architecture simulation models (CASM) into the TLM design (figure 1). CASMs are functional models of a target bus or NoC architecture, allowing for fast but accurate communication architecture exploration. In our design method, they also function as a base for subsequent automated TLM-to-RTL refinement steps. In contrast to the point-to-point TLM channels, CASMs provide m:n communication facilities via multiple TLM ports which implement a generic communication protocol. As we show in [3], the injection of CASMs into the SystemC code of a design can be accomplished by simple algorithms.

HW refinement. When an appropriate system configuration has been identified, all TLM initiator and target ports of each HW module at TLM level have to be merged into a single master/slave port which can be connected to the CASM without the need for wrappers. This requires the splitting of transactions into se-
veral handshake phases with respect to the CASM interface protocol, and the generation of target addresses for communication routing in the target system. This also can be treated automatically (see [3]).

The result is a fully refined SystemC source code with automatically generated send and receive loops reflecting the original high-level TLM API calls. To complete the refinement procedure, the designer’s sole task is to migrate data processing into these automatically generated loops, thus altering his processing entities from object processing to stream processing. Thanks to the systematic design flow, verification of results can be done ad hoc by mixed-mode simulation without the need for testbench re-engineering.

**SW refinement.** In contrast to the HW refinement procedure, for TLM ports of SW components no refinement is necessary at all. The TLM API calls stay unchanged and can be directly mapped to operating system functions by a hardware abstraction layer (HAL). Moreover, quality of service requirements annotated to the TLM ports can be taken into account (e.g. bandwidth and priority demands).

### 3. Prototype generation

When the lowest TLM level of abstraction has been reached, an FPGA-based prototype is ready to be generated. Due to their standardized communication ports, all HW modules can be directly connected to the target communication architecture by so called *accessors*. Accessors are generic wrappers comprising a CASM protocol adapter on the one side and a technology adapter which connects to the target communication architecture on the other side. The technology adapters implement a common data and control protocol and allow for clock domain crossing between fast HW modules and slower buses or NoCs and vice versa.

For HW/SW communication in the prototype, all processor cores in the system are equipped with CPU adapters. They implement a HW/SW handshake protocol which is used by the HAL to pose as virtual TLM channels to the SW modules. In a manner analogous to the accessors, our CPU adapters also use the technology adapters to connect to the on-chip communication architecture.

### 4. Case Example

To analyze the practicability of the described design methodology and its rapid prototyping capabilities, we implemented a HW/SW system with various image manipulation algorithms such as raster rotation and JPEG compression. After a behavioural model has been specified with C++, the system was partitioned into HW and SW SystemC modules. The Virtex-II Pro FPGA fabric was selected as a target architecture, and an IBM CoreConnect CASM and a PowerPC HAL were inserted into the system, using the algorithms introduced in section 2.

To proceed towards a synthesizable design, the TLM ports of the HW modules were first replaced by OCP-TL1 ports and then by OCP RTL signals, using the algorithms described in [3]. As a result, data processing had to be switched from block to stream processing, which implicitly led to an RTL model of computation. By the use of wrappers, the resulting designs were verified by comprehensive mixed-mode simulations at all intermediate levels of abstraction.

![Communication architecture model of the case example](image2)

**Figure 2. Communication architecture model of the case example**

Finally, a model as displayed in figure 2 was obtained and mapped to the communication architecture of the target FPGA by using CoreConnect accessors, a CPU adapter for PowerPC and a C++ HAL for the SW module. The resulting design was immediately synthesizable with standard EDA tools and successfully performed on the Virtex-II Pro FPGA architecture.

### 5. Conclusion

In this paper, a design methodology was presented which combines the world of rapid prototyping with the transaction level modeling approach. The system engineer is guided through a straight-forward development process which both enjoys all advantages of the TLM paradigm and considerably facilitates rapid prototyping. Our future work will focus on generic HW/SW interface architectures.

### 6. References


