Chapter 5
Automatic Generation of Memory Interfaces for ASIPs

David Kammler
RWTH Aachen University, Germany

Ernst Martin Witte
RWTH Aachen University, Germany

Anupam Chattopadhyay
RWTH Aachen University, Germany

Bastian Bauwens
RWTH Aachen University, Germany

Gerd Ascheid
RWTH Aachen University, Germany

Rainer Leupers
RWTH Aachen University, Germany

Heinrich Meyr
RWTH Aachen University, Germany

ABSTRACT
With the growing market for multi-processor system-on-chip (MPSoC) solutions, application-specific instruction-set processors (ASIPs) gain importance as they allow for a wide tradeoff between flexibility and efficiency in such a system. Their development is aided by architecture description languages (ADLs) supporting the automatic generation of architecture-specific tool sets as well as synthesizable register transfer level (RTL) implementations from a single architecture model. However, these generated implementations have to be manually adapted to the interfaces of dedicated memories or memory controllers, slowing down the design-space exploration regarding the memory architecture. To overcome this drawback, the authors extend RTL code generation from ADL models with the automatic generation of memory interfaces. This is accomplished by introducing a new abstract and versatile description format for memory interfaces and their timing protocols. The feasibility of this approach is demonstrated in real-life case studies, including a design space exploration for a banked memory system.

INTRODUCTION
Nowadays, the market for multi-processor system-on-chip (MPSoC) solutions is expanding dramatically. Often, the development of such an MPSoC includes the design of new processor architectures which are tailored to a particular application. A common technique to develop these application-specific instruction-set processors (ASIPs) is the use of an architecture description language (ADL) (Fauth, Van Praet, & Freericks, 1995; Hadjiyiannis, Hanono, & Devadas, 1997; Halambi et al., 1999; Hoffmann, Meyr, & Leupers, 2002; Leupers

DOI: 10.4018/978-1-4666-0912-9.ch005
Automatic Generation of Memory Interfaces for ASIPs

Automatic Generation of Memory Interfaces for ASIPs (Marwedel, 1998; Mishra & Dutt, 2008; Rajesh & Moona, 1999). Present ADL tool suites enable the automatic generation of a fully synthesizable hardware description language (HDL) model of the architecture on register transfer level (RTL) (Basu & Moona, 2003; Hadjiyiannis & Devadas, 2003; Mishra, Kejariwal, & Dutt, 2003; Schliebusch, Meyr, & Leupers, 2007). Optimizations and standard processor features like debug mechanisms are supported by the automatic generation process making the generated model suitable for the final implementation. This decreases development time drastically, thereby allowing the designers to concentrate on the actual architectural features on the high abstraction level of ADLs rather than spending time on detailed modeling on RTL. Accesses to memories and busses are usually modeled as abstract function calls in order to e.g., transfer the address or data. Neither the definition of the interface pins nor a highly accurate description of the timing protocol are required on this level. However, both are mandatory for an accurate implementation on RTL. Adding this low-level information to the ADL model by specifying the pins and their usage directly is no option. This would cause an overhead and lower the abstraction level of the ADL model inadequately, making it complex, hard to maintain and prohibit a fast design-space exploration. Especially, attaching different memory types to the processor would result in many changes to the model, thus slowing down dramatically the exploration of memory architectures which is of special importance for the development of tailored ASIPs. Slower accesses via more complex shared buses (e.g., AMBA) are not directly targeted in this work as depicted in Figure 1. This approach, however, still allows the connection to shared memories utilizing either more than one memory port or a dedicated arbiter as shown in the upper half of the figure.

Their pins as well as the timing protocol including the usage of address/data busses with fixed timing or handshake mechanisms. Keeping the MID separate from the ADL model has several advantages:

Orthogonalization of processor model and memory-interface description: Separating the ADL model of the processor from the MID allows for a fast design-space exploration of the ASIP on ADL level and a pin-accurate implementation on RTL at the same time with the same ADL model.

Reuse of MIDs: Once specified, MIDs for a dedicated memory or memory system can be reused for other architectures or if several identical memories are attached to a single processor.

Rapid exploration of different memory architectures: Different memory architectures, including their physical parameters (e.g., area, timing, etc.), can be explored easily by selecting different MIDs for the HDL-code generation.

Maintaining simulator performance: Since the model is not extended with pin level details, the simulator performance is not affected. Nevertheless, the MID can still be used by other tools of the ADL tool suite if required.

Independence of memory vendors: Designers can more easily switch to other vendors with comparable memories by simply modifying or replacing MIDs. Changes to the ADL model in order to adapt the interface are not required.

The MID is defined so that the description of a wide range of real-world memory interfaces is possible. Even memory systems with cache hierarchy are supported as long as an HDL description is available. Focus is put on on-chip memories, which are tightly coupled with the processor cores, as this is usually the preferred option for fast hardware accelerators, such as most ASIPs. Slower accesses via more complex shared buses (e.g., AMBA) are not directly targeted in this work as depicted in Figure 1. This approach, however, still allows the connection to shared memories utilizing either more than one memory port or a dedicated arbiter as shown in the upper half of the figure.