DYNAMIC RECONFIGURATION WITH HARDWIRED NETWORKS-ON-CHIP ON FUTURE FPGAS

Ronald Hecht, Stephan Kubisch, Andreas Herrholtz, Dirk Timmermann

Institute of Applied Microelectronics and Computer Engineering, University of Rostock
Richard-Wagner Str. 31, 18119 Rostock-Warnemuende, Germany
email: {ronald.hecht,dirk.timmermann}@uni-rostock.de

ABSTRACT

Due to their layered approach, Networks-on-Chip (NoC) are a promising communication backbone in the field of heterogeneous dynamically reconfigurable systems. In this paper a future FPGA architecture is discussed having a hardwired NoC as an additional high-level routing resource. Instead of implementing on-chip interconnection with valuable reconfigurable resources, on top of this architecture, cost-efficient statically and dynamically reconfigurable systems can be built. The concept of such an FPGA is explored by means of an abstract SystemC model. This model not only implements the NoC but also permits a tile based dynamic reconfiguration. It will be shown, that this approach advances the research on operating system support for dynamic reconfiguration in a new way.

1. INTRODUCTION

As Xilinx Virtex-4 devices now support a tile based dynamic reconfiguration, research in this field becomes more attractive then ever. But software and operating system support is not available at all or at low abstraction level only. Thus, developing efficient reconfigurable Systems-on-Chip (SoC) is still a challenging task. However, extending a general-purpose operating system in conjunction with a NoC as a homogeneous, scalable and fault-tolerant communication backbone seems to close the gap between research and commercial applicability [1, 2]. As shown in [3], building such a system with currently available FPGA technologies limits the number of reconfigurable tiles to about four. This makes the promising NoC approach questionable and inefficient. The development of operating system extensions is also primarily driven by the restrictions of the FPGA architecture. It is hardly possible to derive generalities in terms of scalability and performance. In prototypes bottlenecks often arise from the interface between the NoC and the host processor. Consequently, the processor and the operating system must be integrated into the FPGA [2, 4].

This illustrates only some of the difficulties when dealing with dynamically reconfigurable SoCs. Due to the restrictions of current FPGA families, one gets often lost in implementation details, rather than to thoroughly analyze and to define which features future FPGAs should provide to inherently support dynamic reconfiguration. The approach presented in this paper models and simulates such a system in order to explore future FPGA architectures, operating system support and application design. An abstract SystemC model of a dynamically reconfigurable FPGA containing a hard-wired NoC is shown. Together with a simplified model of the operating system first simulation results with sample applications running on the FPGA model are presented.

2. A FUTURE FPGA ARCHITECTURE

Many prototypes of dynamically reconfigurable SoCs have been presented in the past. Common homogeneous communication backbones, such as busses or packet-switched networks, are mandatory for a successful implementation. Unfortunately, they consume quite a lot of reconfigurable FPGA resources allowing only a few actually usable tiles. Furthermore the whole design process is still error-prone. Wires crossing reconfigurable areas must not be touched during reconfiguration. Even if this becomes easier with the tile-based dynamic reconfiguration of Virtex-4 – then wires can be routed around the tiles – the area consumption of the communication network and the design overhead emerging from location constraints is still an issue.

The main idea behind this paper is not to implement a NoC with valuable reconfigurable FPGA-resources, but to define a future FPGA providing this already on-chip. This hardwired NoC may be understood as an additional high-level routing resource allowing to “route packets not wires” [5] at high speed over long distances with minimal power consumption. Besides higher performance and lower costs compared to a soft-implemented NoC, it would tremendously simplify and secure the whole design-process. As the communication backbone would be already present, it will not be disturbed by dynamic reconfiguration. Figure 1 depicts the proposed architecture including already existent components, such as processors, transceivers (Transc), and

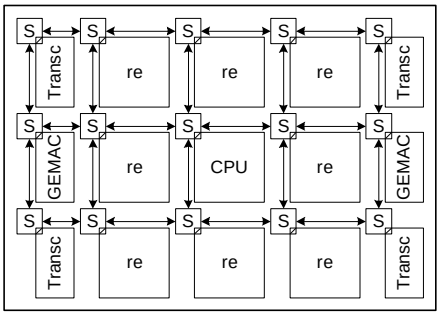

Fig. 1. A hardwired NoC on future FPGAs

Ethernet MACs (GEMAC). The suggested boundaries between the tiles are not meant to be strict. They should rather illustrate possible locations for reconfigurable cores. Modules larger or smaller than these areas are realizable with location constraints.

It could be argued that a pre-routed NoC to merely support dynamic reconfiguration is too expensive. However, nearly every FPGA-design today looks more or less like a SoC. In these designs, many FPGA-resources are already dedicated to SoC busses and FIFOs. A hardwired Network-on-Chip would supersede these structures and thus actually reduce costs. On the other hand, a packet or message based inter-module communication with common interfaces improves design reuse and facilitates a building block based design especially in the case of network processing [6]. We conclude that architectural support for dynamic reconfiguration should be understood as an additional property of NoCs rather than the only motivation for it.

2.1. Configuring the NoC

Apart from packet switches and their physical links, the NoC also comprises elements to be configured at start-up or during runtime. Routing tables matching physical and logical addresses as well as virtual point-to-point connections between cores have to be set up. If dynamic reconfiguration is not intended for the system, all these parameters should be adjustable by the FPGA's configuration bitstream. Actually, this approach does not conflict with current FPGA technologies, as the identified components are basically RAMs or registers.

During dynamic reconfiguration, routing tables and virtual point-to-point connections must be altered at runtime. Two alternatives may be followed. One possibility is to configure the NoC through the internal or external configuration port of the FPGA. For this purpose, it must be allowed to have dedicated access to the configuration data of the NoC components. This may be realized by a block-based reconfiguration already present in the Virtex-4 family. On the other hand, special NoC messages and protocols may implement a management plane to be used for configuration. This concept is followed within this paper as it is expected to be faster due to the high bandwidth of the NoC. Implementing an individual configuration NoC [1] seems to be inefficient as it will mostly remain dormant.

2.2. A Model as an Explorator

NoCs are a rather new field of research. Protocol stacks have not been standardized and commercial products have to be build with conventional SoC design yet. Thus, the discussed FPGA architecture is far from fabrication. Instead of implementing another limited prototype based on current FPGA technologies, the proposed FPGA is abstractly modelled to explore its architecture and implications. One intention is to obtain an executable specification for the vendor. The model can be used to refine the lower layer NoC protocols, the NoC topology and to specify the configuration methods. A second motivation for such a model is to use it as a virtual FPGA to study software and operating system support for NoC based dynamically reconfigurable systems. Utilizing an abstract model as a virtual FPGA means to integrate it into a runtime system, as it would be a real FPGA. In that case, the operating system manages the NoC and reconfigurable tiles of the model. In contrast to prototype implementations based on currently available FPGAs, this approach is not limited by the amount of reconfigurable resources or design obstacles earlier discussed.

At the moment, it is not entirely clear how to simply describe, implement, and debug applications for dynamically reconfigurable systems. A central dilemma is the mixture of sequential and parallel components. Here the FPGA model will help to experiment with use cases and will aid hardware/software co-design and debugging. Independence from a specific technology will speed up the whole design process and allows for considering various design paradigms and implementation methods. In this case, the FPGA model will act as an executing platform for dynamically reconfigurable designs.

Assuming the FPGA with a NoC to be existent in the future, it must be possible to emulate reconfigurable modules by their software counterparts, further on called decelerators. This is the case when all tiles are exhausted or no FPGA is present in the system. For this purpose, the model can be extended for an unlimited number of tiles to execute the decelerators. At that point, the model acts as an integral part of the virtual hardware management of the operating system.

3. MODELLING THE DYNAMICALLY RECONFIGURABLE SYSTEM

To use the model as a virtual FPGA in a reconfigurable runtime system, low-level details may be omitted. It is neither

necessary to emulate the real configuration bitstream nor the behaviour of each CLB or other basic building blocks of today's FPGAs. The approach presented here abstracts the FPGA as an array of individually reconfigurable tiles. Each tile can be configured with an accelerator – the hardware implementation of a core. It is possible to load an accelerator in every tile. This contrasts to current FPGAs not allowing to relocate bitfiles. Prototype implementations hold a bitfile for each tile to circumvent this. A fundamental part of the FPGA model is the NoC permitting accelerators and processors to communicate with. The model of the network is not cycle-accurate, but can be constrained with bandwidths and delays. From the operating system and application point of view, these abstractions entirely fulfil the model to be a dynamically reconfigurable FPGA.

As shown in figure 2, our framework connects the FPGA model to a processor configuring the tiles and communicating with them using the NoC. A certain switch whose interface is exported establishes the connection to the NoC. The full FPGA model is still under development and thus not integrated in a real operating system yet. A simplified model comprising the basic building blocks to manage dynamic reconfiguration simulates this. Consequently, the whole system runs in a SystemC simulation. As depicted, the processor executes a scheduler, which manages the reconfigurable resources, and a dispatcher which configures the tiles and routing tables. An IPCore registry keeps track of all cores currently in use by applications.

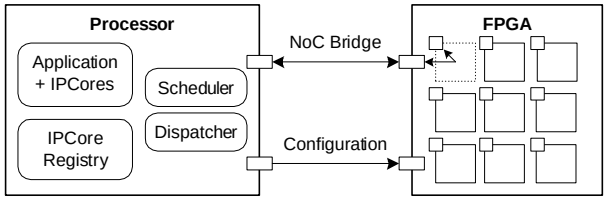

Fig. 2. SystemC simulation environment

3.1. The NoC Model

Designing a NoC mainly bases on the definition of protocol layers. Based on the ISO/OSI model or the TCP/IP protocol stack, five abstraction layers can be identified: the physical, datalink, network, transport and application layer. Additionally, hierarchical addresses allow for fast routing and fixed virtual connections supporting dynamic reconfiguration. Physical or geographical addresses distinguish tiles, logical addresses IP cores, and ports processes on IP cores or processors.

The concept behind the NoC model presented here is based on SystemC channels. Each protocol is implemented as an individual channel. Interconnecting their interfaces, stacks the protocols. The entire class library may be understood as a collection of building blocks, which can be assembled to every possible NoC topology and application. As illustrated in figure 3, a protocol channel has interfaces to upper- and lower-layer protocols. Attached protocols can access its services through these interfaces. Services of upper and lower layer protocols are accessed through the ports of the channel.

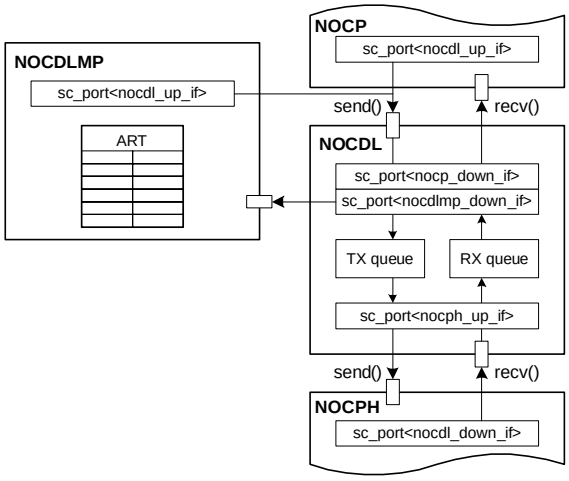

Fig. 3. Modelling network protocols with channels

3.2. Modelling dynamic Reconfiguration

Dynamic reconfiguration on SoC level is basically swapping hardware modules at runtime. Going into more detail, this implies a controlled interruption of the accelerator and a shut down of the virtual point-to-point connection to other processes. The current state of the core may be saved to resume its execution later or to emulate it by its decelerator. Finally, the physical link has to be disconnected. Ideally, the application and the user should not notice any reconfiguration at all except for an improved or reduced performance.

Most of the work is done by the NoC protocols as each layer is assigned certain responsibilities to support dynamic reconfiguration. But some issues are left associated with the creation and destruction of hardware modules and their physical link to the network. Because SystemC does not allow dynamic port binding, a new port class has been implemented circumventing this limitation. Additionally, it refuses access to unconnected interfaces during simulation. This port class has to be used for all internal connections of accelerators.

As a replacement of the SystemC class sc_module, the new class scr_module has been specifically designed for reconfigurable cores. In contrast to sc_module, it is allowed to create objects of this class during simulation. Processes

within such modules are spawned at runtime using the extensions of SystemC 2.1. Additionally, this class features an interface to interrupt and to start its operation. The same interface is used to extract and to restore the state. Using scr_module as the base class, an accelerator class has been derived. This class integrates some parts of the transport layer and the application specific message protocol. A decelerator class containing the software implementation of the core and also the application protocol represents the software counterpart. An IPCore class aggregates both classes and implements the inverse application protocol. This class represents the stub used by an application to communicate with the accelerator or decelerator. In contrast to [7], this approach is completely application independent and may be used for all types of accelerators or interface cores. Additionally, it is well suited for real dynamically reconfigurable systems or prototypes.

3.3. Simulation Experiments

To verify the implementation of the NoC, the dynamic reconfiguration and their management, some simulation experiments have been done. The SystemC verification library (SCV) was used here, in particular data introspection and constrained randomization. To verify each NoC protocol layer, complex constraints were used to generate network traffic ranging from large data bursts to single management and signalling messages. As the NoC model is intentionally not cycle-accurate, no timing deviations from the adjusted bandwidths and delays have been investigated.

Additionally, two use cases where applied to the dynamically reconfigurable system model. As shown in figure 4, a simple AES crypto core has been implemented based on the methodology presented above. This core allows encryption and decryption of packets. Together with these operations, setting keys and encryption modes are encapsulated in a NoC application protocol. During dynamic reconfiguration, keys are saved and restored respectively. A second application example was a consumer-producer arrangement. Here, both cores were relocated between tiles and replaced by their software counterparts during execution. By this example, the concept of interruption and state save was successfully verified.

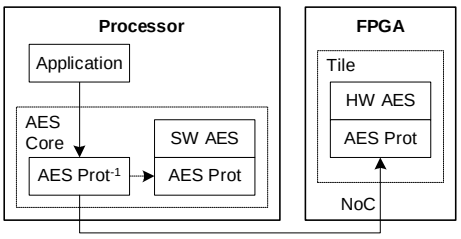

Fig. 4. Simulation of an AES Core

4. CONCLUSION AND OUTLOOK

Motivated by the success of NoCs in the field of dynamically reconfigurable systems, a vision of a future FPGA architecture was drawn. In contrast to implementing the communication backbone with valuable reconfigurable logic, this FPGA will already have a hardwired NoC as an additional high-level routing resource. This network will not only inherently support dynamic reconfiguration but will also replace the on-chip busses currently built with reconfigurable logic. To explore this architecture, a new approach was followed. Instead of assembling another prototype, the envisaged FPGA is modelled with SystemC at a high level of abstraction. Basic sample applications were implemented to run on this model demonstrating NoC-based dynamic reconfiguration. This strategy allows abandoning the limitations and design obstacles of currently available FPGAs and frees the mind for design considerations of future FPGAs, their software support, and application development. Possibly it will motivate to fabricate such an FPGA in the future.

Besides an executable specification, it is planned to use the model as a virtual FPGA in an operating system managing reconfigurable logic as a shared resource. Following this idea, research on such operating systems will advance in a novel way, as it is not restricted by prototype limitations. It will be possible to evaluate run-time placement and scheduling strategies with real applications. Thus, implementing more use cases and studying application design paradigms for dynamically reconfigurable systems are one of the most concerns for prospective work.

5. REFERENCES

[1] T. Marescaux et al, “Networks on chip as hardware components of an os for reconfigurable systems.” in *FPL 2003*, Lisbon, Portugal, 2003, pp. 595–605.

[2] R. Hecht, D. Timmermann, S. Kubisch, and E. Zeeb, “Network-on-chip basierende laufzeitsysteme für dynamische rekonfigurierbare hardware.” in *ARCS Workshops*, Augsburg, Germany, 2004, pp. 185–194.

[3] T. Marescaux et al, “Interconnection networks enable fine-grain dynamic multi-tasking on fpgas.” in *FPL 2002*, Montpellier, France, 2002, pp. 795–805.

[4] M. Hübner et al, “Scalable application-dependent network on chip adaptivity for dynamical reconfigurable real-time systems.” in *FPL 2004*, Leuven, Belgium, 2004, pp. 1037–1041.

[5] W. J. Dally and B. Towles, “Route packets, not wires: on-chip interconnection networks.” in *DAC 2001*, Las Vegas, NV, USA, 2001, pp. 684–689.

[6] G. J. Brebner, P. James-Roxby, E. Keller, and C. Kulkarni, “Hyper-programmable architectures for adaptable networked systems.” in *ASAP 2004*, 2004, pp. 328–338.

[7] A. Schallenberg, F. Oppenheimer, and W. Nebel, “Designing for dynamic partially reconfigurable FPGAs with SystemC and OSSS.” in *FDL 04*, Lille, France, 2004.