SIMULATION SYSTEM FOR RUN-TIME RECONFIGURABLE NETWORKS-ON-CHIP

Carsten Albrecht, Roman Koch, Thilo Pionteck, and Erik Maehle

Institute of Computer Engineering
University of Lübeck
Ratzeburger Alle 160
23538 Lübeck
Germany

albrecht@iti.uni-luebeck.de(Carsten Albrecht)

Abstract

Complexity of System-on-Chip design has grown step by step in the last decade. Beside tremendous capacities of transistors developers have to deal with new features such as run-time reconfiguration. Run-time Reconfiguration, in particular, rapidly emerged and is currently provided by several FPGA series. Those features demand new methodologies for modelling and simulation, verification, and evaluation. Furthermore, the focus moves towards the interconnect structure that has to be scalable and adaptable to multiple requirements. To cope with the challenges of interconnect and system-on-chip design, hardware designers divide their systems into smaller sub-systems. A general approach for dividing up complete systems is tile-based modelling where tiles contain similar-sized groups of elements. This model is introduced for static as well as for dynamically reconfigurable systems. The one-to-one relation of the simulation model of a single tile and real tile on an FPGA is sketched and re-used for a tile-based model of run-time reconfigurable networks-on-chip. Additionally, its application to an existing adaptive network-on-chip, CoNoChi, supporting irregular topologies and changes of topology at run-time is shown. A complete simulation suite based on SystemC is introduced with the goal to establish a base for other application-specific derivatives.

Keywords: Tile-based Modelling, Network-on-Chip, Run-Time Reconfiguration, SystemC.

Presenting Author's Biography

Carsten Albrecht received his Diploma degree in Computer Science from the University of Lübeck in 2002. During his studies, he gained research experience at the University Claude Bernard of Lyon, France, in 1999 and work experience at Alstom Power Switzerland in 2000. Both placements set the focus to parallel processing applications. In 2002, he joined the Institute of Computer Engineering, University of Lübeck. First, he studied multi-threaded processor design and application-specific processors in the field of network computing. His current research topic is modelling, simulation and application-specific management of dynamically reconfigurable systems.

1 Introduction

The growing complexity of Integrated Circuits (ICs) and, years ago, the need to facilitate design reuse motivated the System-on-Chip (SoC) approach which is now state of the art. As SoC designs were incorporating an ever increasing number of modules comprising altogether several hundreds of millions of transistors efficient module interconnect structures became a crucial issue which has been widely discussed in literature during the last years. Arbitrary interconnection structures require different module interfaces and communication protocols. Furthermore, the design goals are spread from high bandwidth, low latency to small area or low power consumption. Usual interconnects have in common that they establish dedicated or bus-based links of modules. Unfortunately, bus-based systems do not scale very well with an increasing amount of IP (Intellectual Property) cores and may not meet the requirements of all systems [1]. Hence, a new communication paradigm, packet-based Network-on-Chip (NoC) [1, 2], was established and applied to Application Specific Integrated Circuit (ASIC) designs. Currently there exist a couple of research approaches for NoCs to be used in ASIC designs such as SPIN [3] and Nostrum [4] and commercial approaches such as Spidergon [5]. Their advantages over standard interconnects have been well examined for different application areas concerning power consumption, bandwidth, latency etc. Spidergon in particular is accepted as a suitable solution for SoC designs concerning applicability, scalability and performance [6].

Nowadays, Field Programmable Gate Arrays (FPGAs) become more and more popular to implement complete SoCs. Due to an increasing amount of gate equivalents and decreasing costs per device they growingly form a suitable platform in the field of e.g. media-stream processing. Moreover, NoCs become applicable on FPGAs on recent devices such as Xilinx Virtex-II [7], Virtex-4 [8], or Virtex-5 [9] which now provide this number of gate equivalents. Their presence makes NoC feasible for FPGA-based system-on-chip design.

Beside the growing gate capacity FPGAs provide the feature of being configurable with different configurations to be adapted to their application field or just changing conditions. In contrast to legacy FPGAs modern ones do not only support reconfiguration of the whole device, they even offer the opportunity to be reconfigured partially at run-time. So, they are able to form dynamically reconfigurable systems where hardware modules, parts of the system, may be exchanged during run-time. Areas that are not affected by reconfiguration can continue without interruption. This feature of modern FPGAs enables changes of area allocation and interconnect topology over time and, thus, makes system design and evaluation harder.

Varying number, size and shape of exchangeable modules as well as their communication requirements such as bandwidth and latency have an impact on the communication architecture. Using a fixed communication network with fixed interconnection ports for hardware modules generally leads to an inefficient usage of FPGA logic resources. Thus, flexible networks should enhance area utilisation while meeting the performance requirements. Hence, the evaluation of more flexible communication infrastructures and their comparison to fixed ones are important to detect and quantify the benefit. Current NoCs proposed for ASIC design do not support topology changes at run-time so that their topology has to be determined at design time and cannot be adapted to a changing SoC structure. Therefore, the drawbacks of bus-based interconnects are not completely removed for dynamically reconfigurable SoCs by static NoCs.

So, scalable and topology-adaptive NoCs for those systems seem to be a suitable solution. There are different approaches [10] but only few are scalable and support irregular topologies. The simulation suite presented here provides a test bed selectively on transaction or even on cycle-accurate level for tile-based FPGA designs using packet-switched NoCs as on-chip system interconnect for run-time reconfigurable systems. The simulator is based on the idea to divide the array of logic resources into tiles. These tiles build the atomic exchangeable units for all reconfiguration processes. They may contain basic functionalities and permanent interconnects to adjacent tiles which are bound to larger possibly irregular shapes implementing a functionality on their tile compound [11]. For the simulator, the C++ class library SystemC [12] is used as simulation core engine. SystemC brought out by a pool of companies and research institutions backs the top-down design methodology so that each module can be iteratively developed and supports system design on an arbitrary level. Today, it is applied in almost all areas of hardware development.

The paper is organised as follows: In the next section the basic model and its application in hardware design for standard and run-time reconfigurable systems is explained. Section 3 shows the simulation suite with system partitioning and application-specific parts. Then, Section 4 presents an application example utilising the simulation system and, finally, the paper is summarised.

2 Tile-based Modelling

2.1 Static Tile-based Models

Competing with the growing number of transistors is a crucial issue in actual hardware design. Especially, verification and evaluation of systems and sub-systems is difficult. For this reason systems are broken down into a set of few components which allows to assemble the complete system by using an arbitrary number of each element. A frequently used method is generating tiles of same size and interface structure but providing different application-specific functionalities. Because of the unified interfaces tiles can be easily combined to form larger systems. Above all, system verification and evaluation can also be broken down to verification and evaluation of each element of the building set. These tasks might cause less effort than dealing with the complete system at once.

Furthermore, placement of modules with unified shapes and interfaces does not cause spatial waste. Additionally, cost of wiring is reduced by just connecting neighbouring tiles. Long wires across the die with a decelerating impact on the overall system speed are avoided. Systems based on the tile-based modelling approach have also shown good ratios concerning power consumption and computing power. With regard to power efficiency Synchroscalar [13], for example, a tile-based architecture for digital signal processing, outperforms standard ASIC implementations as well as standard digital signal processors. Another approach brought out by Intel is an 80-core programmable processor [14] that is regularly built up by utilising a single type of tile. All tiles include a router to establish an on-chip network by connecting all neighbours in 2D-mesh. This multi-core processor achieved more than 1 Teraflop while the power consumption with 62 W is rather low.

In general tile-based approaches deal with rectangular tile shapes. Other shapes that provide similar features for loss-less placement and unified interfacing can be applied, too. In [15] a chip design based on hexagonal-shaped modules to increase connectivity is presented. Because of its hexagonal structure the architecture is called HoneyComb. Basically, a HoneyComb architecture consists of three different types of tiles: a datapath tile, a memory tile, and an input/output tile. All tiles include a router to establish any link between tiles. The input/output tile simply installs an interface to the usual SoC world by establishing a certain bus interface. The datapath tiles include functional elements for data processing and the memory tiles simply store data.

Decomposing a system into modules and even smaller parts such as tiles can be seen as a functional and partially data decomposition. In terms of parallel processing the idea of re-using the tile-based model to execute the simulation in parallel by distributing the tiles suggests itself. Unfortunately, SystemC as a basic simulation core does not support parallel execution. This issue is addressed but not yet solved by the community. Thus, with regard to parallel computing the model can be taken as a contribution for the future.

2.2 Dynamic Tile-based Models

Beside customised ASIC designs tile-based modelling can be applied to FPGA designs as well. As shown in Figure 1, FPGAs also provide a regular structure with long wires to establish connections across the whole device and logic blocks implementing functions with inputs and outputs. Additionally, each block contains a small memory. Unfortunately, irregularity is often introduced by application-specific function blocks or simply hard-wired components such as embedded processors. Dividing an FPGA into tiles delivers multiple tile types depending on the device and granularity. So, a tile can be a single logic block or may span over a couple of logic blocks so that at least all contain the same resources.

Utilising the modelling technique on FPGAs leads to reconfigurable system designs where reconfiguration is

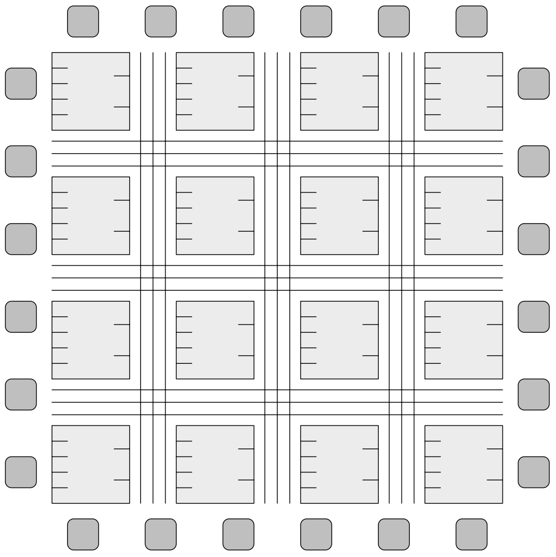
Fig. 1 Generic FPGA architecture.

performed on coarse-grained level by exchanging tiles. Modern FPGAs such as the Virtex-4 family allow tile-based replacement of configuration data. Additionally, these devices can be reprogrammed even partially from inside so that a SoC can adapt itself autonomously. Although these features are now technically matured the tool chain does not include support for developing, simulating, and verifying reconfigurable SoCs at once. Currently, each system configuration must be developed to create differential configuration data to change the system partially later on.

The tile-based modelling and simulation approach for run-time reconfigurable systems based on SystemC is introduced by [11]. Existing approaches for modelling run-time reconfigurability with SystemC are discussed and the method for tile-based models is motivated and derived by current FPGAs. SystemC is widely used for modelling and evaluating hardware designs on a high abstraction level. Nowadays, different tools allow to refine the system model e.g. using VHDL to map it to the target device while keeping the test method implemented in SystemC so that there is a reduced risk of faults compared to test methods reimplemented in other languages on lower levels. As well as SystemC supports nearly all levels of abstraction in hardware design, the tile-based model can be applied to all of these. It is targeted to cycle-accurate transaction-level models [11]. Actually, SystemC designs are not run-time reconfigurable but the work of [11] enlarges the SystemC tool suite by this feature to bridge the gap from usual SoCs to run-time reconfigurable ones.

In the simulation suite presented here, the simulation models of tiles are basically the same as usual SystemC modules. Each type of tile has to be modelled once and the complete system is assembled of multiple instances of any type. Modelling run-time reconfigurable tiles in SystemC reuses the concept of unified interfaces. But the internal functions are moved to polymorphic func-

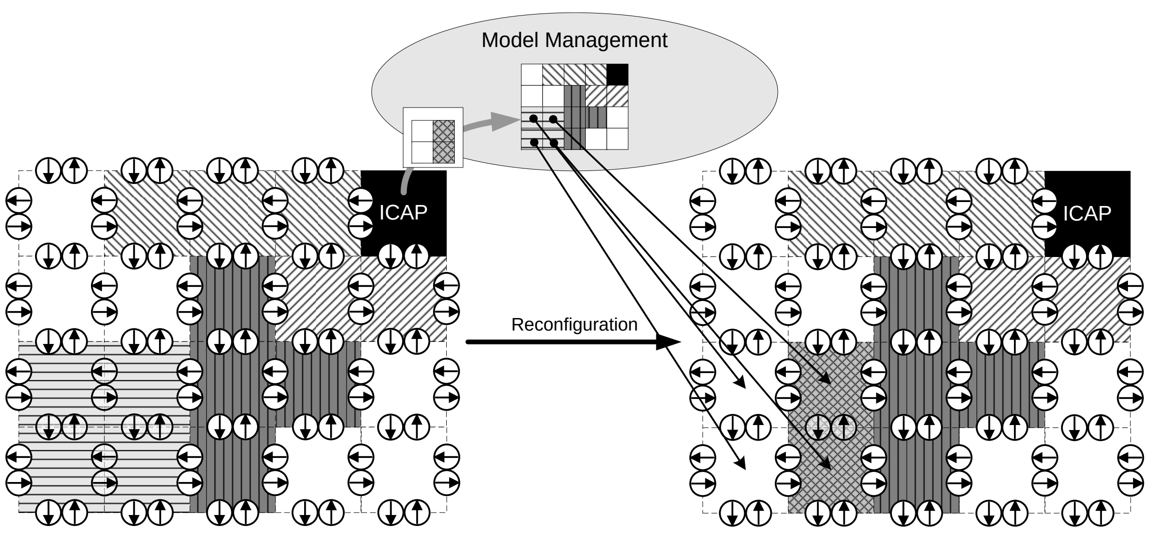

Fig. 2 Reconfiguration of Tile Array.

tions or objects so that they are able to change their behaviour or type, respectively, at run-time. The relationship between SystemC tiles and FPGA tiles is generally one-to-one. The size of a SystemC tile can be chosen arbitrarily whereas FPGA tiles are constrained by device limits such as frames, slices, or logic blocks. E.g. the Xilinx Virtex-4 [8] and the Virtex-5 [9] series offer a fine reconfiguration granularity. In contrast to the Virtex-II series where a frame always spans the complete height of a column, the height of a frame of these architectures is a multiple of 16 or 20 configurable logic blocks (CLBs), respectively. Thus, the smallest unit that can be reconfigured consists of a tile with a width of one CLB and a height of 16 or 20 CLBs.

The designer of the model is responsible to take area consumption into account and to keep the tiles within the targeted range of functionality. In case of implementing functional units that span over multiple tiles, these tiles are compounded in the SystemC model and a master tile is selected that executes the functionality. The others are only used for interconnects while their functional capabilities are switched off.

Figure 2 depicts the exchange of tiles within a tile-based simulation model. A global management systems keeps the configuration data and can be accessed via the internal configuration access port (ICAP, the term is taken from Xilinx FPGAs). The ICAP receives a packet with addresses and new configuration data for certain tiles and passes it to the global model management which represents the configuration memory. Utilising its global perspective the management calls the reconfiguration functions of the tiles affected in this process. The new configurations are ready after a certain period of time. This period represents the time consumption or costs of reconfiguration. The reconfiguration costs rc are approximately determined by a linear function of the number of affected tiles n:

$$rc(n) = b_{ICAP}^{-1} \times (s_{tile} \times n + h)$$

where b_{ICAP} is the bandwidth of the ICAP, s_{tile} is the configuration stream size of a single tile and h is a constant value for header, footer and control information in the configuration stream independent of the number of tiles. So, $rc(n)$ delivers the time the reconfiguration process consumes.

3 Simulation

The NoC simulator is based on the tile-based SystemC model introduced in Section 2.2. Any simulated topology is embedded in a two-dimensional array of tiles. Each tile has rectangular shape with the same amount of input and output ports per edge. In contrast to other simulations with similar goals using tiles incorporating a router combined with a functional element [16], the tiles here are targeted to comprise either a router or a dedicated NoC interface with a functional unit. This may lead to irregular, non-grid topologies of the NoC.

As shown in Figure 3, the SoC software simulation model is divided into a static and a reconfigurable part. These two basic parts generally build up the complete run-time reconfigurable system. The static part is built up as a usual SoC design. It mainly contains permanently needed application-specific features such as interfaces or management components. The reconfigurable part is a two-dimensional array of tiles which are SystemC modules executing polymorphic functions [11]. There are components which have to be part of both, static and reconfigurable, part to keep consistency within the on-chip interconnect structure. So, the NoC links and the modules are available in the static part, too. Thus, they are elements of both system parts. The implementation of those NoC elements is encapsulated in C++ objects so that they can be re-used in any SystemC module. In Figure 3 the components and their classification within the simulation system are sketched. A set of essential NoC functions provided by tiles and static part is shortly described in the following:

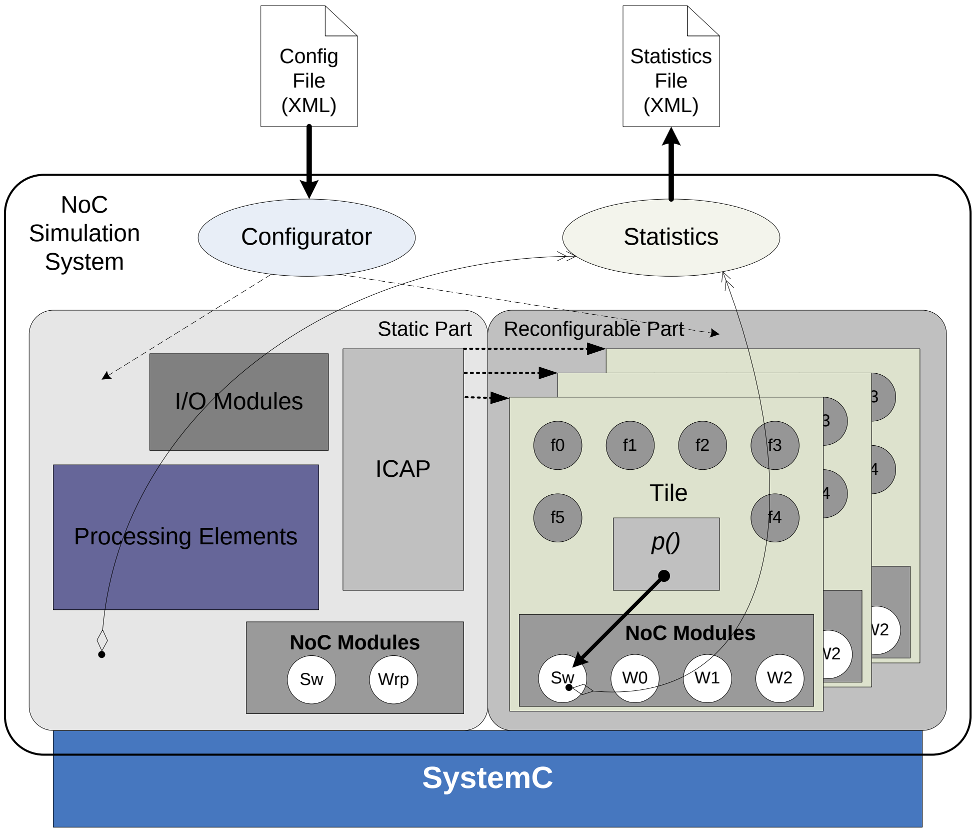

Fig. 3 Architecture of Simulation System

- SWITCH. The switches, see module *Sw* in Figure 3, perform their routing decision by look-ups. The routing tables needed have to be centrally prepared and sent via NoC.
- LINK. Because all components are distributed among an array it is not necessarily the case that communication neighbours are placed in adjacent tiles. So, functionally unused tiles can connect neighbous directly and loss-less. They are noted as *Wx* in Figure 3.
- INTERFACE. Each reconfigurable function unit (RFU) connected by the NoC needs a unified interface to be attached to the NoC. In Figure 3, they are included in the RFU processes denoted by *fx*.
- WRAPPER. If data is injected from outer world into the SoC the data has to be packetised and attributed with the protocol information. For this purpose a wrapper, see module *Wrp* in Figure 3, has to be made available at least as a static module.

Additionally, arbitrary functions can be included in a tile to complete a customised simulation model. All functions incorporated within a tile must be added to the reconfiguration procedure so that it is callable by a unique identifier that is used in the whole system.

The setup of a complete simulation is externally defined using an XML-based configuration file. It contains all simulation-model parameters: the global ones such as system clock speed, the set-up of the static part including modules and their specific parameters. Furthermore, the reconfigurable part is defined by the dimension of the tile matrix $n \times m$. As well as the static modules, all used RFUs must be defined and parametrised. Beside the functionality of an RFU its latency and throughput is mandatory for correct simulation of the NoC behaviour. Depending on the management strategy at least an initial configuration of the reconfigurable part is mandatory. Applying on-line placement for RFUs and NoC components the resulting configurations are hardly predictable and cannot be and, of course, do not need to be previously defined. Limiting the system to few configurations and reducing the management to find the most suitable one for the moment a set of all valid configurations must be defined. The management strategies must be implemented on their own, only the basic technique of computing the difference of two configurations and creating the corresponding configuration stream is provided. Arbitrary protocols are difficult to support but it is possible to define the packet part which is used for look-up in the switches, routers and NoC interfaces.

The simulation is driven by data to be processed by the system. This data has to be packetised. Thus, even if there is a continous, non-discrete data flow from outside, the data is re-organised by the wrapper as mentioned above. Combining these two steps the simula-

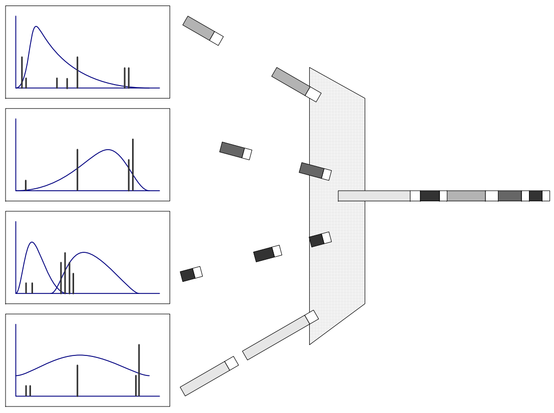

Fig. 4 Multiplexed packet, header (white box) and payload (grey-shaded boxes), generation using various distributions for bandwidth (continuous) and packet size (histogram).

tion can generally be fed by a packet generator. Creating packet streams synthetically is a rather crucial issue. Here, combinations of standard techniques are provided. First, single streams or flows can be generated using different distributions such as Poisson or Uniform distribution. Second, because of the bursty nature of traffic the b-model [17] is included and finally, third, several flows are multiplexed to get a combined stream of different packet flows. Figure 4 shows the packet generation process by creating several flows with own data distributions, continuously depicted, and packet-size distributions, histogram. All flows are multiplexed to retrieve a single flow for the simulation.

Simulation traces are basically reduced to adding needed information to the packets so that the complete path of any packet can be reconstructed. Based on the packet trace, component performance information can be gained by filtering all packets that passed the component in the requested period of time. The packet trace is generated by formatting packet and collected monitoring information using an XML scheme and dumping it into a file. The data is only stored when a packet is terminated by any reason. The trace file is evaluated and statistics of internal model components, system behaviour and performance measures are generated after a simulation run. Thus, stability measures such as confidence intervals are currently not controlled at run-time. Sufficient run-time cycles of the simulation must be determined by estimations and test runs.

Beside the analysis of the NoC features and performance the simulator provides a means to develop and evaluate strategies to detect the point of reconfiguration and procedures to perform the reconfiguration. System reconfigurations generally base on events generated in- or outside of the system. For instance, a monitor surveys performance data and internally triggers reconfiguration. Decisions therefor are made by various algorithms which have to be tailored to application-specific purposes. Thus, the simulation can be utilised to determine an application-specific subset of monitoring data and a suitable reconfiguration management algorithm.

4 Sample Simulation Model: CoNoChi

Based on the previously described methodology a topology-adaptive dynamically reconfigurable NoC called *CoNoChi* (Configurable Network-on-Chip) [18] was modelled, simulated and functionally analysed.

4.1 Components

CoNoChi comprises virtual cut-through switches with four equal full-duplex links. The links can either be used for connections to adjacent switches or for connecting an RFU to a switch. The switches contain routing tables so that forwarding decisions are made by look-up. The interfaces buffer incoming and outgoing packets, collect performance data if needed, and are able to re-target packets in case they are misrouted after a reconfiguration. The protocol wrapping is included in the dispatcher that performs mapping of incoming data to RFUs by table-based look-ups. Adaptability enables to implement only the minimal number of switches required for a given configuration of hardware modules, minimizing both overall latency and area requirements of the network.

The design of *CoNoChi* is oriented at the reconfiguration capabilities of the Virtex-4 FPGA series, i.e. the smallest unit that can reasonably be reconfigured during runtime is a CLB column consisting of 16 CLBs. This leads to a tile size of 16x16 CLBs which is sufficient to contain a complete *CoNoChi* network switch with 32 bit link width running at about 75 MHz.

4.2 Management

The protocol is three-layered. It contains a physical, logical and application-specific layer. It is designed to support the current physical topology and a sort of overlay network that might be organised in an application-specific manner to bind processes to logic, not physical addresses. Logical addresses can be mapped to other physical ones in case another component takes over. The application-specific layer is used for control of the target RFU by indicating the type of data and if present the mode how to process it.

Further on, the central management instance is the reconfiguration manager. It computes the routing tables for each switch and sets up the dispatcher and evaluates the collected monitoring data to adapt the system to the current load situation.

4.3 Reconfiguration

Depending on the topology, switches can be added to or removed from the network without stalling the NoC. The NoC only contains switches which are currently needed for operation. When RFUs are inserted or removed by dynamic reconfiguration the number of switches and their locations have to be adapted by the same mechanism as well. To guarantee full connectivity of the NoC special precautions have to be taken in order not to isolate parts of the NoC. So, first inner nodes of the NoC graph can only be removed if the NoC

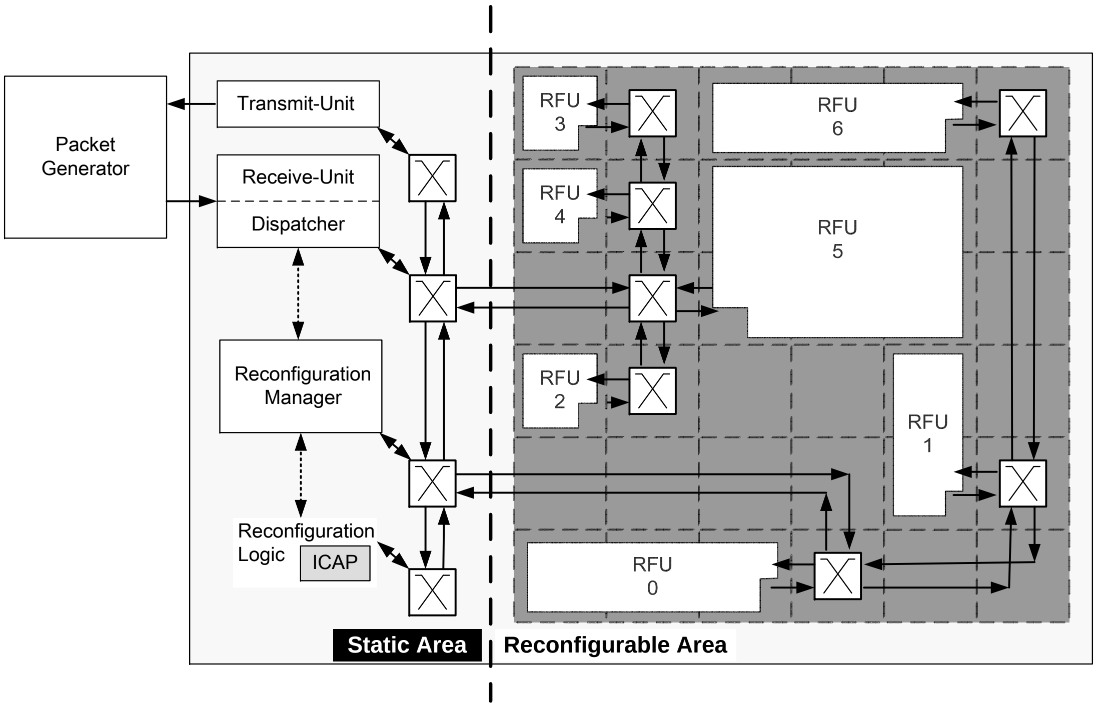

Fig. 5 Sample Simulation Model

graph is not separated in two graphs. Additionally, a link can only be removed if there exists another way to connect both involved switches. If both conditions are satisfied, first the routing tables and the dispatcher are adapted to the temporary topology with removed switch and/or links, then the affected part is reconfigured and the routing tables and the dispatcher are set up to the new established configuration. A detailed description of certain cases is beyond the description of the simulation itself and can be found in [19].

4.4 Application

CoNoChi is applied to the architecture of a dynamically reconfigurable network co-processor DynaCORE [20]. Figure 5 shows a complete simulation model with a static part including system I/O using a receive and transmit unit, the ICAP, and the reconfiguration manager which cares about NoC routing and detecting the need for reconfiguration with regard to system load and data processing requirements. The system information is collected in the NoC interfaces and send via the NoC. The configuration data as well is sent via the NoC to the ICAP. The decisions are based on monitoring data, the current system configuration and possible successor configurations. If the estimated gain of a reconfiguration is smaller than the reconfiguration cost the system would not be improved. Thus, the reconfiguration should not take place. The dispatcher assigns incoming data packets to RFUs by table look-up. Further on, it performs protocol wrapping for incoming packets. The model is currently used to test and develop algorithmic details and architectural aspects of the management modules.

5 Conclusion

High-level simulation of complete complex SoCs including their interconnect is a crucial issue for hardware design. Especially, the evolution trends towards NoC and run-time reconfigurability demand new methodologies and tools. The simulation system presented here bases on a tile model. It is highly configurable so that models can be adapted without re-writing the complete code. Furthermore, the simulation results are reduced to packet traces. These traces can be evaluated at a post-processing step to gain any internal system information about performance and load. The simulation suite applies SystemC because it allows detailed modelling of hardware. As an example for NoCs, *CoNoChi* is applied here. Its set-up and algorithms are sketched in round terms to demonstrate a topology-adaptive NoC with promising features and but also high resource requirements for management. All in all, the simulator described here provides a suitable system to analyse NoC efficiency and reconfiguration strategies in SoC designs.

6 Acknowledgement

This work was funded in part by the German Research Foundation (DFG) within priority programme 1148 under grant reference Ma 1412/5.

7 References

[1] Pierre Guerrier and Alain Greiner. A Generic Architecture for On-Chip Packet-Switched Interconnections. In *DATE '00: Proceedings of the Conference on Design, Automation and Test in Eu-*

rope, pages 250–256, New York, NY, USA, 2000. ACM Press.

[2] Luca Benini and Giovanni De Micheli. Networks on Chips: A New SoC Paradigm. *IEEE Computer*, 35(1):70–78, January 2002.

[3] Adrijean Adriahantenaina, Herve Charlery, Alain Greiner, Laurent Mortiez, and Cesar Albenes Zeferino. SPIN: A Scalable, Packet Switched, On-Chip Micro-Network. In *DATE '03: Proceedings of the Conference on Design, Automation and Test in Europe: Designer's Forum*, pages 70–73, Washington, DC, USA, 2003. IEEE Computer Society.

[4] Mikael Millberg, Erland Nilsson, Rikard Thid, Shashi Kumar, and Axel Jantsch. The Nostrum Backbone - A Communication Protocol Stack for Networks on Chip. In *VLSI Design*, pages 693–696, 2004.

[5] Marcello Coppola, Riccardo Locatelliand Guiseppe Maruccia, Lorenzo Pieralisi, and Alberto Scandurra. Spidergon: A Novel On-Chip Communication Network. In *Inernational Symposium on System on Chip*, page 15, Tampere, Finland, November 2004. IEEE.

[6] Luciano Bononi and Nicola Concer. Simulation and Analysis of Network on Chip Architectures: Ring, Spidergon and 2D-Mesh. In *DATE '06: Proceedings of the Conference on Design, Automation and Test in Europe*, pages 154–159, 3001 Leuven, Belgium, 2006. European Design and Automation Association.

[7] Virtex-II Pro and Virtex-II Pro X FPGA User Guide. Technical Report, 2005.

[8] Virtex-4 Family Overview. Data Sheet, 2005.

[9] Virtex-5 FPGA Configuration User Guide, UG191, Version 2.2. User Guide, February 2007.

[10] Thilo Pionteck, Carsten Albrecht, Roman Koch, Erik Maehle, Michael Hübner, and Jürgen Becker. Communication Architectures for Dynamically Reconfigurable FPGA Designs. In *14th Reconfigurable Architectures Workshop*, Long Beach, USA, 2007. IEEE Computer Society.

[11] Carsten Albrecht, Thilo Pionteck, Roman Koch, and Erik Maehle. Modelling Tile-Based Run-Time Reconfigurable Systems Using SystemC. In Ivan Zelinka, Zuzana Oplatkova, and Alessandra Orsoni, editors, *21st European Conference on Modelling and Simulation*, ISBN 978-0-9553018-2-7, pages 509–514, Prague, Czech Republic, 2007. European Council for Modelling and Simulation.

[12] IEEE Standard SystemC Language Reference Manual. IEEE 1666-2005, 2006.

[13] John Oliver, Ravishankar Rao, Paul Sultana, Jedidiah Crandall, Erik Czernikowski, Leslie W. Jones IV, Diana Franklin, Venkatesh Akella, and Frederic T. Chong. Synchroscalar: A Multiple Clock Domain, Power-Aware, Tile-Based Embedded Processor. In *ISCA '04: Proceedings of the 31st annual international symposium on Computer architecture*, page 150, Washington, DC, USA, 2004. IEEE Computer Society.

[14] Sriram Vangali, Jason Howard, Gregory Ruhi, Saurabh Dighe, Howard Wilson, James Tschanz, David Finan, Priya Iyerl, Arvind Singh, Tiju Jacob, Shailendra Jain, Sriram Venkataraman, Yatin Hoskotel, and Nitin Borkar. An 80-Tile 1 .28TFLOPS Network-on-Chip in 65nm CMOS. In *Proceedings of 2007 IEEE International Solid-State Circuits Conference*, Piscataway, NJ 08854 USA, February 2007. IEEE Solid-State Circuits Society.

[15] Alexander Thomas. Design of a Dynamic Reconfigurable Multi-Grained Hardware Architecture with Adaptive Runtime Routing. In *FPL*, pages 745–746, 2005.

[16] Lilian Bossuet, Wayne Burleson, Guy Gogniat, Vikas Anand, Andrew Laffely, and Jean-Luc Philippe. Targeting Tiled Architectures in Design Exploration. In *International Parallel and Distributed Processing Symposium (IPDPS'03)*, pages 172–179, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[17] Mengzhi Wang, Ngai Hang Chan, Spiros Papadimitriou, Christos Faloutsos, and Tara Madhyastha. Data Mining Meets Performance Evaluation: Fast Algorithms for Modeling Bursty Traffic. In *18th International Conference on Data Engineering*, pages 507–516, Los Alamitos, CA, USA, 2002. IEEE Computer Society.

[18] Thilo Pionteck, Roman Koch, and Carsten Albrecht. Applying Partial Reconfiguration to Networks-on-Chips. In *Proceedings of 2006 International Conference on Field Programmable Logic and Applications*, Madrid, Spain, 2006. IEEE.

[19] Thilo Pionteck, Roman Koch, and Carsten Albrecht. A Dynamically Reconfigurable Packet-Switched Network-on-Chip. In *Proceedings of the DATE 2006*, 2006.

[20] Carsten Albrecht, Jürgen Foag, Roman Koch, and Erik Maehle. DynaCORE - A Dynamically Reconfigurable Coprocessor Architecture for Network Processors. In *Proceedings of the 14th Euromicro Conference on Parallel, Distributed and Network-Centric Processing*, pages 101–108, Montbéliard-Sochaux, France, February 2006. Euromicro, IEEE Computer Society.