Mapping KPN Models of Streaming Applications on A Network-on-Chip Platform

Ashkan Beyranvand Nejad[1], Kees Goossens[1,2], Johan Walters[3], Bart Kienhuis[3,4]
[1]*Computer Engineering, Delft Unniversity of Technology, Delft, the Netherlands*
[2]*Corporate Research Department, NXP Semiconductors, Eindhoven, the Netherlands*
[3]*Compaan Design BV, Amsterdam, the Netherlands*
[4]*LIACS, Leiden University, Leiden, the Netherlands*
A.BeyranvandNejad@tudelft.nl, kees.goossens@nxp.com, jwalters@compaandesign.com, kienhuis@liacs.nl

Abstract

Designing complex parallel systems begins with behavioral modeling of applications. The more desired models should be at high abstract level of applications' behavior. One of this model is Kahn Process Network gives a partitioned model of applications as concurrently communicating processes. At the implementation level of applications, growing demands of running multiple applications on a single chip, introduced Network on Chip as an appropriate interconnection network between large number of IP cores for different applications. Therefore, design way from high abstract model of application to implementation level is becoming an issue. Here, in this paper we propose an approach for mapping KPN model of a streaming application on an architecture based on Network on Chip. We also provide an implementation of the applications using this approach as SystemC model of the architectures to be simulated.

1. Introduction

Traditionally, an application can be expressed as various models of behaviors. The most imperative model typically used by designers is a model of computation, most commonly C. On the other hand, running many tasks in parallel on different IP cores concurrently due to recent advances in systems on chip (SOC) design, pops up the needs of having parallel models of computation for applications. One of the most popular models of computations for specially streaming based applications like image processing, audio, etc., is Kahn Process Networks (KPN). The KPN considers a network of concurrent processes communicate through unbounded First-In-First-Out (FIFO) queue channels (pipes). Each process known as a node of the network, is a sequential program does blocking read on its inputs, executes its computational task and posts results to its outputs [1], [2]. Figure 1 shows a very simple application modeled as 3-node KPN.

Nowadays there is not only one application running on a single chip. Consequently, modern SOCs have become very complex in terms of large number of IP cores involve in running different applications. This has made the communication between those IPs a significant issue on a single chip. Network on chip (NoC) is a promising communication

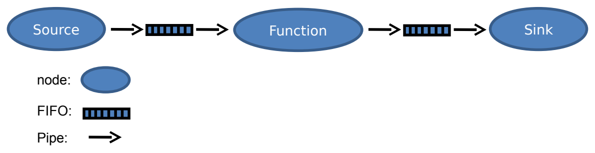

Figure 1. The KPN model of a simple streaming application

platform to solve this issue by providing not only modularity and scalability for designing new systems, but also it has raised up the abstraction level of system's designing up to the application level, i.e it can guarantee requirements such as delay and throughput at the demands of application level.

In this paper we present a method of mapping applications modeled as Kahn Process Networks (KPN) on a NoC. In order to achieve this, it is discussed that these four problems should be solved: 1) Nodes' functional implementation, 2) correspondence of communication pipes, 3) generation of appropriate NoC topology & configuration, and 4) corresponding elements for communication FIFOs. Our goal of implementation in this work is realizing SystemC model of NoC-based system for any given application modeled as KPN. For this purpose, an automatic design flow is proposed which starts from KPN model of application, generates SystemC model of correspondent NoC-based system and simulates it. Here, we use the Compaan design tools to derive KPN model of application from specified model in C [3]. However, we leave any RTL implementations as future works.

In [4], a framework for NoC-based multiprocessor SOC is presented. Its underlying model of computation is also KPN mapped on a NoC-based architecture. However, in this work communication pipes are software implemented as processor global FIFOs handled by primitives of an embedded operating system. In our work we propose our mapping approach based on using the already available NoC's resources (e.g. NI's FIFOs) as communication pipes in the KPN model of applications. Despite the realization of KPN pipes as dedicated hardware in [5] and [6], the proposed architecture is based on traditional bus interconnects which is hardly scalable and appropriate for concurrent parallel applications.However, our proposed approach here is based on taking advantages of the NoC as scalable and concurrent interconnection platform.

The rest of this paper is organized as follows. Section 2 explains the NoC platform as our target interconnection architecture. In section 3, our approach of mapping KPN to a NoC-based system with its different aspects are discussed, continuing to Section 4 describes SystemC implementation of approach. In Section 5 we propose a design flow for our method, following the description of an experimental example in Section 6. Finally, we conclude our paper in Section 7.

2. Target NoC Platform

The target NoC platform used in this paper to map KPN model of an application on, is the Æthereal NoC platform [7]. Its basic building blocks are Routers (Switches) and Network Interfaces (NIs) (Figure 2.a). The Routers and NIs are connected together by bidirectional communication links, forming a mesh-like topology. Different IP blocks (Cores) are connected to the NoC via Network Interfaces. NI is responsible to translate communication protocol of various IPs to the interface protocol understandable by the network, and vice versa. The buffered end-to-end flow control policy used by the Æthereal platform has made the architecture of the NI very important for us. The NIs have different number of ports according to the number of logical communication channels (request or response) should be connected to the corresponding Router. Architecture of a NI with one port is illustrated in Figure 2.b. There is a dedicated variable size queue for each channel, where the sizes can be determined at the NoC design time.

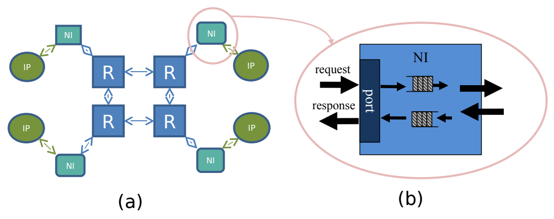

Figure 2. a) Mesh-like topology of the NoC, b) The schematic architecture of a Network Interfaces

Data transformed by the NIs into the format of packets should be routed by Routers. A Packet contains routing path information and destination address. Regarding to applications running on the network, there are two kinds of packet require different quality of services to be offered; Best Effort (BE) and Guaranteed Throughput (GT). In order to achieve application's requirements, the GT packets have higher priority than the BE ones to be routed by Routers. Therefore, the GT packets are routed immediately and there is no buffering of them inside the Routers; while there are buffers (queues) for the BE packets that might need to wait to be routed.

Later in section 3 it is shown how we use the available queues in the NoC as a novel approach in our work.

3. Mapping Approach

In order to map a KPN model of an application to the system consists of a NoC as interconnection network between different IP cores, there are four main procedures distinguished. These procedures are closely related to the features of the KPN model. In the KPN model, there are number of nodes running sequential computational tasks, communication links between nodes form a communication configuration, and First-In-First-Out (FIFO) queues for storing intermediate data. Considering the KPN model shown in Figure 1 and NoC-based system illustrated in Figure 3, the four processes can be enumerated as follows:

1) implementation of nodes' computational functionality
2) Physical or logical correspondence of communication pipes
3) Generation of appropriate NoC topology & configuration
4) Corresponding elements for communication FIFOs

Depending on a task performed by each node, there can be a dedicated hardware IP core or a processor element correspondence to that node (process) runs the sequential program. However, our goal of implementation in this work is not realizing hardware; in Section 4 we will explain our implementation's method of every node's function in SystemC to deal with the aforementioned Step 1.

The automatic design flow of the Æthereal platform [8] provides topology generation and appropriate configuration of a NoC architecture by declaring our model's IP cores (nodes) and logical communication links between them in the XML format. Thus, this has already solved above mentioned procedures 2 & 3 as the NoC generation and the communication pipes. It is more explained in Section 5 where the design flow is detailed.

KPN considers unbounded size for all the FIFO queues in the model of an application. However, in reality it is not possible to have such unlimited size FIFOs. On the other hand, it is discussed in Section 2 that there are some queues in the Routers and NIs of the Æthereal platform. These FIFOs which are there to fulfill application requirements by NoC, can be also used as communication FIFOs between nodes in KPN model of the application. Since NIs' queue is there for both BE and GT traffic and Router queues are for temporary storing of only BE data, the NI's queue is in interest of our mapping approach. The novelty of our approach is using NoC's NI dedicated FIFOs for communication channels as communication FIFOs in KPN model. A node which is correspondent to an IP posts its produced data to the target IP core over the NoC and it would be stored in the target NI's queue. IF the target NI's queue is full, end-to-end flow control policy stops sending data. In this case the data is kept in the source NI's queue till there would be available space. The space becomes free by consumption of data already available in target queue by target node (IP core).

Moreover, sizing KPN queues should be done in such a way that occurring deadlock in running application be avoided. This can be perform by doing static analysis on KPN model of application; or starting system implementation with large enough size of FIFOs to find out the exact consumed queues' space after performing SystemC simulation and back annotate them. Since the NoC design flow (Section 5) assigns the queue sizes to fulfill its own requirements, the starting point should be some values at least equal to or more than calculated ones by the flow.

Figure 3 shows a conceptual view of NoC-based system corresponds to the KPN model illustrated in Figure 1. There are three IP cores resemble three nodes of the KPN. Logical communication channels are set-up over the network, where there are dedicated source and target NIs for every link.

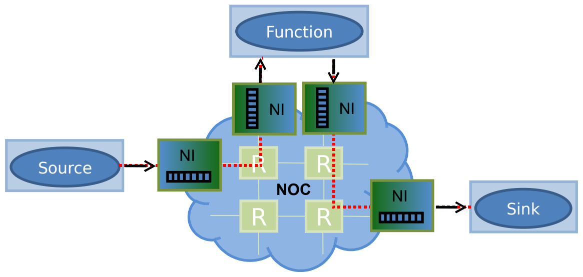

Figure 3. The NOC-based model of a simple streaming application

4. SystemC Implementation

Generally computational function of each KPN node is performed in two distinguished tasks: Communication and Computation. The communication task is the same for all nodes, except that the number of input and output ports for each node might be different. Every node should wait for the data to be available at all its input ports (blocking read) and sends out the result from its output ports (unblocking write) to the FIFOs. In other words, there is two interface phases of reading and writing, while in between computation task is performed. Every node has its own function implementing a specific sequential program.

A SystemC module has been implemented as a generic wrapper for every node to perform communication phases to the NoC regardless of its computational function. This wrapper is instantiated as IP cores interfacing with the NoC. Therefore, there is a one-to-one mapping between KPN nodes and SystemC wrappers (procedure 1 in Section 3). The wrapper is parameterized for every node according to its number of in/output ports. However, it should also executes the specific function of every node. For this purpose, every KPN node's computational functionality is implemented as a C++ function code to be called inside its own wrapper module at run time. To make it feasible, the function should be compiled as a Shared Object Library (.so files in Linux) at the design time. During the time of running SystemC model of the application, the wrapper executes its function by dynamically loading corresponding library, passing input data to and receiving back the results from that. This procedure is illustrated as a flow chart in Figure 4.

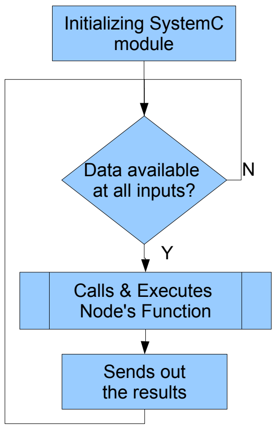
Initializing SystemC module
Data available at all inputs?
N
Y
Calls & Executes Node's Function
Sends out the results

Figure 4. The functional flow chart of the generic wrapper

5. Design Flow

The Æthereal design flow is an automated tool flow generates appropriate NoC architecture for any specified applications. An application is identified in two design spaces. First, its architecture, i.e the number and type of the IP cores it has, and second, the communication between the IPs and their requirements (e.g. bandwidth, latency and throughput) should be declared. These are provided respectively as *architecture & communication* XML format files.

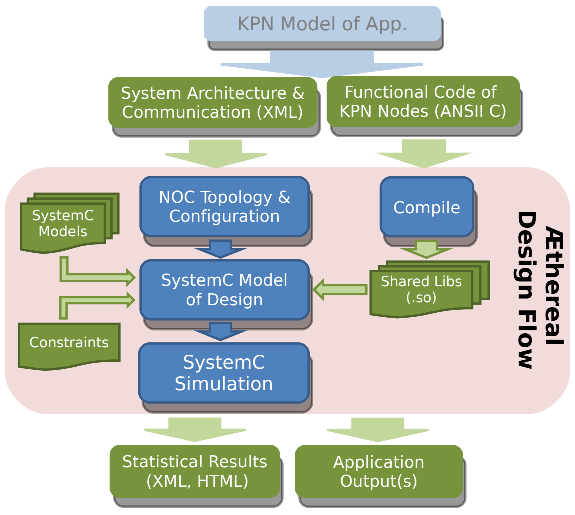

Figure 5. Mapping Design Flow

Our contributions in this section is to extend and modify the Æthereal design flow to:

1) Provides declaration of application KPN model in above mentioned XML files
2) Adds our generic wrapper module to its SystemC module library
3) Generate Shared Object Libraries automatically from functional codes of KPN nodes

Figure 5 shows complete overview of the Æthereal modified design flow. It can also simulate the SystemC model of the design and gives us some statistical results of the simulated architecture, e.g. maximum queue's occupancy for each channel, data delivery delay over channels.

6. Experimental Example

As an application example we have chosen the Sobel algorithm for image's edge detection, modeled in C. KPN model of this application is derived by using Compaan Design tools. We run our design flow for this example to generate the NoC-based system architecture of this application. Generated network consists of 5 nodes' wrapper modules, 4 Routers and 16 NIs. Having simulated the SystemC model of the application for input data image size of 255*255 pixels, the time taking to process this image is equal to 6m seconds at 500 MHz clock cycle. The real time of simulation took about 3.5 seconds on a PC.

7. Conclusion

The work presented here is a straightforward approach from application modeled as Kahn Process Network to be mapped on a NoC-based system architecture. The NoC platform used in this project is the Æthereal NoC which our method taking advantages of its already available resources in NIs. Our goal of implementation to realize the SystemC model of the application is fulfilled by introducing an automatic design flow. However, further realization of the system based on this method in RTL and hardware implementation is left as our future work.

Acknowledgment

We would like to acknowledge that this work is funded as part of the *TSAR* European project coded as 2A718 by MEDEA+.

References

[1] G. Kahn, "The semantics of a simple language for parallel programming," in *Information Processing '74: Proceedings of the IFIP Congress*, J. L. Rosenfeld, Ed. New York, NY: North-Holland, 1974, pp. 471–475.

[2] E. A. Lee and T. M. Parks, "Dataflow process networks," *Proceedings of the IEEE*, vol. 83, no. 5, pp. 773–801, 1995.

[3] B. Kienhuis, E. Rijpkema, and E. Deprettere, "Compaan: deriving process networks from matlab for embedded signal processing architectures," in *CODES '00: Proceedings of the eighth international workshop on Hardware/software codesign*, 2000, pp. 13–17.

[4] G. Sassatelli, N. Saint-Jean, C. Woszezenki, I. Grehs, and F. Moraes, "Architectural issues in homogeneous noc-based mpsoc," in *Rapid System Prototyping, 2007. RSP 2007. 18th IEEE/IFIP International Workshop on*, May 2007, pp. 139–142.

[5] T. Stefanov, C. Zissulescu, A. Turjan, B. Kienhuis, and E. Deprettere, "System design using kahn process networks: The compaan/laura approach," in *DATE '04: Proceedings of the conference on Design, automation and test in Europe*. Washington, DC, USA: IEEE Computer Society, 2004, p. 10340.

[6] C. Zissulescu, T. Stefanov, B. Kienhuis, and E. F. Deprettere, "Laura: Leiden architecture research and exploration tool," in *FPL*, 2003, pp. 911–920.

[7] K. Goossens, J. Dielissen, and A. Rădulescu, "The Æthereal network on chip: Concepts, architectures, and implementations," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 21–31, Sept-Oct 2005.

[8] K. Goossens, J. Dielissen, O. P. Gangwal, S. G. Pestana, A. Rădulescu, and E. Rijpkema, "A Design Flow for Application-Specific Networks on Chip with Guaranteed Performance to Accelerate SOC Design and Verification," in *Proc. of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, Mar. 2005, pp. 1182–1187.