A Virtual Channel Network-on-Chip for GT and BE traffic

Nikolay Kavaldjiev, Gerard J. M. Smit, Pierre G. Jansen, Pascal T. Wolkotte

Department of EEMCS, University of Twente, the Netherlands
{n.k.kavaldjiev, g.j.m.smit, p.t.wolkotte, p.g.jansen}@utwente.nl

ABSTRACT[*]

This paper presents an on-chip network for a run-time reconfigurable System-on-Chip. The network uses packet-switching with virtual channels. It can provide guaranteed services as well as best effort services. The guaranteed services are based on virtual channel allocation, in contrast to other on-chip networks where guarantees are provided by time-division multiplexing. The network is particularly suitable for systems in which the traffic is dominated by streams.

We model the data traffic in the system and simulate the behaviour of the network with this model. The results show that the network is capable of handling the system traffic and can provide the required guarantees.

I. INTRODUCTION

Advances in silicon technology bring, among others, two problems that chip designers have to face – a high design complexity and a signal integrity problem [1],[2],[3]. The first problem is the concern that the complexity of a system that fits on a single chip is getting so high that the time needed to design a completely new system using the current design methods and tools is becoming impractical long. For that reason it is foreseen that future System-on-Chip (SoC) will be based mostly on pre-designed IP blocks relying on extensive IP reuse. To be practical, such a design methodology needs to be complemented with a unified and simple solution for interconnecting and integrating IP blocks in a system. Currently on-chip buses offer such a solution, but since the bus bandwidth does not scale with the number of IP cores on the chip it will soon become a system bottleneck.

The second problem, the signal integrity problem, is due to the fact that with the technology scaling transistors get smaller and faster while wires get thinner and slower. Wire delay becomes proportional to the wire length and a few long wires on a chip can degrade the performance of the entire chip. Thus, the on-chip interconnects become a limiting factor for SoC performance and their physical parameters must be taken into account at an early design stage. The current approach of ad-hoc on-chip wiring becomes questionable especially for the global (inter IP) on-chip interconnects, which are longer and slower.

A possible approach for coping with the two problems is to use a Network-on-Chip (NoC) [2][3], which is a light weight communication network built on the chip. This network replaces the traditional on-chip bus and provides a scalable high-bandwidth solution for interconnecting the increasing number of IP blocks. On the other hand, when a regular network topology is used, the global on-chip wires are short and well structured which helps coping with the signal integrity problem.

The performance of a network strongly depends on the characteristics of the data traffic generated by the system. Therefore a NoC cannot be evaluated separately, but has to be considered in the context of its operational environment. In this paper we present a network-on-chip solution for a run-time reconfigurable SoC used in mobile multimedia devices. The dynamic nature of such a system requires a flexible NoC solution. Since many of the system applications have real-time requirements, the system and the network have to be predictable. Network predictability is achieved by providing guaranteed network services. The proposed network provides guaranteed throughput (GT) and best effort (BE) services.

We discuss the system and the applications running on it and construct a model of the data traffic in such system. The model is then used to evaluate the network. We perform simulations of the network behaviour the results of which show that the network can handle the system traffic and can provide the guarantees requested by the applications

The paper is organized as follows: Section II gives an overview of the SoC where the proposed network is used. Section III presents the network-on-chip and explains how guaranteed services are provided. Section IV presents simulation results showing that the network can handle the system traffic and can provide the required traffic guarantees. Section V presents related work.

II. SYSTEM OVERVIEW

The network proposed in this paper is used in a SoC for a specific application domain – mobile multimedia devices. In such a system streaming applications dominate the demand

[*] This research is supported by the research program of the Dutch organisation for Scientific Research NWO (project number 612.064.103) and the EU-FP6 project 4S (IST 001908)

for computation and communication capacity and high performance has to be achieved with a limited power budget. The operational environment requires for a dynamic system capable to adapt it self at run-time.

A. *System architecture*

The architecture we envision for a mobile multimedia system is shown in *Figure 1*. It is heterogeneous SoC consisting of multiple processing elements (PEs), represented as rectangles. Each PE has its own local data and code memory and can work independently. Several PEs are general purpose processors shared between the control oriented tasks running in the system. Most PEs are domain specific processors, like accelerators, DSPs etc., optimized to perform fast and efficiently computationally intensive algorithms in a specific application domain, e.g baseband processing, image, video and audio processing etc. These PEs are programmable, but since highly optimised, they are not very flexible. Predicting the performance of a task running on a single-task processor is easier than on a multi-task processor. Instead of complex multi-task processors, for the computationally intensive tasks we prefer to use many simple and optimised single-task processors, thus gaining efficiency and predictability. On the other hand, in contrast with ASIC solutions, we maintain programmability which prolongs the system life and therefore reduces its cost, while keeping the high performance and efficiency. An example of a domain specific PE is the processing tile proposed by Heysters et al [4]. For 0.13 μm technology the area of the tile is 2 mm^2. On a typical chip of 14x14 mm near 100 such PEs can be fitted. The next technology generations will allow hundreds of domain specific PEs to be fitted on a single chip.

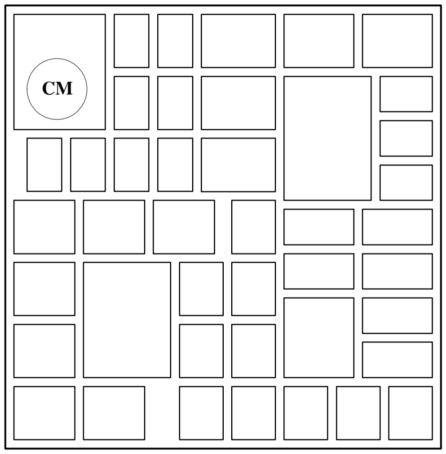

Figure 1 System architecture

The system may contain also some large memories blocks providing common storage space or some peripheral blocks that interface with external devices. All the components in the system are interconnected by an on-chip network which transports data between them.

B. *System operation*

To provide the flexibility and adaptability required by the application domain the system should be dynamic, reconfigurable at run-time. The system we envision is centralized. A central authority, which we call *configuration manager* (CM), is in charge of the management of the system resources, e.g. PEs, network, communication channels, shared memories etc. CM runs as a high priority task on one of the general purpose processors. By sending control messages on the network to PEs the CM can control and configure each PE separately while the rest of the system is running. Using this mechanism the CM can start, stop and adapt applications at run-time. Once started, an application runs independently and does not require the attention of the CM. To facilitate the application starting process, at compile-time the applications are represented as a set of sequential *processes* that can run in parallel exchanging data on *communication channels*. At run-time when starting an application the CM allocates PEs for the processes and network routes for the communication channels of the application. The set of running application in the system changes dynamically, therefore the system state changes over time. But for the duration of an application the PEs and the network routes remain statically allocated to the application. Considering applications like baseband processing, audio/video (de)coding, etc. the application duration is from seconds to hours. Hence, changes in the system state do not occur very often.

The centralized system organization is suitable for run-time reconfigurable systems, because it keeps central information about the system state and decisions connected with the system reconfiguration can be taken and deployed fast and centrally.

C. *Streaming applications*

The system described above is targeted at the domain of mobile multimedia devices. Many computationally intensive applications in that domain deal with processing of data streams; for example, base-band processing for wireless communications, audio/video (de)coding, image processing etc. When represented as a set of communicating processes, streaming applications typically have a simple pipeline structure. Our experience with applications is in the domain of base-band processing for wireless communications, like HiperLAN/2, UMTS, Bluetooth and DRM [5][6]. A generalized applications structure observed in this domain[†] is shown in *Figure 2*. Two parts can be distinguished in this structure: a *processing part* and a *control part*.

[†] A similar observation for media processing applications is made by Dally et al [7].

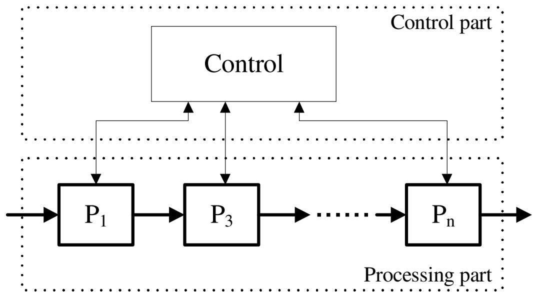

Figure 2 Generalized structure of a streaming application

The processing part does the actual stream processing. It has a simple pipeline structure. The data from the stream flow through the pipe and each process applies some transformation on it. The processes constructing the pipe (denoted as P_1, P_2, …, P_n) typically implement DSP algorithms, e.g. FFT, DCT, FIR etc. These algorithms are small, computationally intensive and specific for the application domain. Therefore, they are suitable to run on domain specific PEs. The communications between the processes in the pipeline carry the main data stream and therefore require high throughput. Because the pipe usually works under real-time constraints, the communications there are also real-time, hence requite GT services. Therefore the communication traffic generated by the processing part of the applications is high throughput GT traffic.

The control part of the application comprises all the application tasks dealing with the application organisation and control, run-time adaptation and reconfiguration. Because of the reactive nature of these tasks they run not so often, require lighter computation and therefore are more suitable to run on shared general purpose PEs. The data traffic between the control and the processing part consists of infrequently exchanged short control messages, like event notifications, status and parameters exchange. These are not real-time communications, hence require BE services. Therefore the communication traffic generated by the control part is low throughput BE traffic. As a rough estimate for the applications we consider (base band processing), we observe that 90% of the traffic is GT traffic and 10% is BE traffic.

D. *Traffic*

In summary, the traffic in the stream-processing system we envision for mobile multimedia devices consists of: i) 90% high throughput GT traffic coming from streaming data, ii) 10% low throughput BE traffic coming from short control messages.

The configuration messages sent by the CM to the PEs in the system also contribute to the BE traffic in the system. The size of these configuration messages depends on the configuration space of the PEs. For example the total configuration space of the processing tile proposed by Heysters et al [4] is 2.6KB. Although this size is not small, configuration messages are generated infrequently, only when a new application is started, and therefore do not contribute significantly to the system traffic - our estimate is for less then 0.1% of the system traffic.

III. Network

Here we present the on-chip network we propose for interconnecting the PEs in the system described in the previous section. It is a packet switching virtual channel network that provides GT as well as BE services.

The topology of the network is a 2-dimensional mesh - each PE is equipped with a router (its local router) and all the routers are connected in a mesh topology. Such a regular network topology keeps the global on-chip wires short and well structured which allows for speed and power optimization. A PE is connected only to its local router. The routers connect to each other (and to the PEs) through parallel full-duplex *network channels*.

The data in the network are transported on virtual channels [8]. Virtual channels (VCs) are logical channels that share the same network channel. A network channel is time shared between its VCs on a cycle-by cycle basis in round-robin fashion, but cycles are allocated to a VC only when it has data to transmit. The VCs are separately buffered in FIFOs at the router inputs. In our network, each network channel is shared between 4 VCs, this number being motivated by the trade-off between performance and buffer area of a virtual channel router studied by Dally [8].

Sharing a network channel on a cycle-by-cycle basis means that the VCs equally share the network channel bandwidth. In our design the bandwidth is shared only between the VCs that currently have data to transmit; the VCs that are idle do not consume bandwidth. Thus the minimal throughput of a VC is determined by the maximal number of VCs that can simultaneously transmit data. Therefore, knowing the number of VCs that are used on a network channel, we can give a lower bound on the throughput for these VCs. If on a physical channel of bandwidth *b*, *v* VCs are used, then each of these VCs is guaranteed throughput at least

$$TH_{\min} = \frac{b}{v} \quad (1)$$

This is the worst case throughput – when all the *v* VCs constantly transmit data. Therefore, whatever traffic load is applied to the *v* virtual channels their throughput will not go below TH_{min} (provided that the remaining VCs are idle). Since in our network the number of VCs per network channel is 4, the possible values that TH_{min} can take are *b*, *b/2*, *b/3* or *b/4*.

In traditional virtual channel networks VCs are allocated to packets dynamically by the routers. The number of VCs on a network channel that are currently used depends on the current traffic and cannot be determined. Therefore no throughput bounds can be given for a VC. In contrast, in our network VC are statically allocated to communication channels. The allocation is done centrally and the number of occupied VCs on a network channel is known. Therefore, we can give a lower bound on the VCs throughput.

On its route from source to destination a communication channel traverses a sequence of VCs, one per network channel, which are statically allocated to the communication channel for the duration of the application. Thus a communication channel traverses the networks on a chain of VCs. The minimal throughput of the chain is determined by the VC with minimal throughput bound. Since we can give a lower bound on the VCs throughput, we can give a lower bound on the throughput of communication channels.

Given the lower bound TH_{min} on a communication channel throughput, the maximal latency T_{max} of a message of length L bits passing that channel (in a wormhole manner) is [18]

$$T_{\max} = N * t_r + \frac{L}{TH_{\min}} \tag{2}$$

where, N is the hop count or the number of network channels traversed by the message, and t_r is the delay of the message per router or the time that the head of the message spends waiting in a router. In our design t_r is at most v clock cycles (v has the same meaning as in equ. (1)). Let T_c be the network clock period and w the network channel width. Taking into account equation (1) and having in mind that $b=w/T_c$, then equation (2) is transformed to

$$T_{\max} = \left(N + \frac{L}{w} \right) * v * T_c \tag{3}$$

This is the maximal latency that a message of length L bits, travelling distance N hops experiences on a GT channel with $TH_{min}=b/v$.

In our network VCs are allocated by using source routing - a network address specifies not only the destination PE, but also the route to be followed. The route is described as e sequence of network channels and their VCs to be taken. Packets destination address is carried by the packet header. When a packet header is send, it is handled by the network and transported to the destination PE following the route described by address. Along its way in the network the header reserves the VCs it traverses. The data form the packet body follows the reserved VCs and reaches the destination PE without need for additional control information. At the end, the packet tail releases the reserved VCs.

Each PE contains a table with addresses of the PEs to which it has to send data. The table is loaded by the CM during the PE configuration. When starting an application the CM uses a central routing function to find routes for all the communication channels of the application. The routing function keeps track of the state of the network. It knows exactly which VCs in the network are used and which are free. Thus, the routing function can predict what the minimal throughput of a given route will be. Further more, it can look for a route with specified throughput bound. In this way GT services are provided to communication channels in our network.

When routing a communication channel requiring GT service, the routing function looks for a route with the specified throughput bound. This route should not violate the guarantees give to already routed communication channels. When a route is found its VCs are reserved and no other channels can use them for the duration of the application. The data of the communication channel are transported on the route in packets. Since the VCs on the route are used only by the communication channel the packets can be of any length without disturbing the other traffic in the network. Thus, a message directly fits into a single packet and there is no need of messages splitting and reassembling. A packet of unlimited length can be seen as a *connection* opened between the source and destination. The connections are a convenient way for transporting streams - the prevailing traffic type in the system. Instead of sending the data from the stream in separate packets, a connection is opened. The packet header is sent only once and then the stream is continuously sent as a packet body. This reduces the network control overhead because a header is sent only once instead for every stream item.

When routing communication channels requiring BE services, the routing function uses shared VCs. These VCs can be shared between several BE communication channels. Sharing entails packet blocking, which makes the throughput prediction difficult. Therefore no guarantees are given. On BE communication channels data are sent only on packet-by-packet basis. Since the VCs for BE connections are statically allocated and shared, a behaviour similar to that of a wormhole network should be expected. The wormhole networks do not perform well for intensive traffic consisting of long packets. But as we saw in the previous section, the BE traffic in our system is of low intensity and consists mainly of short messages (several bytes), therefore no performance problems are expected.

A network router has been implemented and synthesised for 0.13 μm technology. The router has an area of 0.18 mm^2 and can operate at a maximal frequency of 500 MHz [9]. Almost half of the router area is consumed by the buffers. More implementation details can be found in Kavaldjiev et al [10] and results for the energy consumption are presented by Wolkotte et al [11].

IV. Simulation

To validate the network a cycle-accurate simulation is performed. A mesh network of size 6-by-6 is simulated using a traffic model with the characteristics presented in *Section II.D*.

A. Setup

The communication traffic pattern used in the simulation is derived as follows. The nodes of a directed graph with ring topology are scattered over the nodes of a 6-by-6 mesh network. The graph contains 36 nodes and each graph node is placed on a separate network node. The edges between the placed graph nodes define the communication channels between the network nodes.

The ring communication pattern is representative for the GT traffic of the streaming applications, because a large ring graph can be seen as a serial connection of many short pipeline graphs. We consider random scattering as worst case strategy for mapping applications on the system. In a real system the application mapping will be done such that the communication locality is maximized, which leads to better network conditions.

Each node in the network generates both types of traffic, GT and BE, transmitted on two separate communication channels following the ring communication pattern (a communication channel in the ring graph is substituted by one GT and one BE channles). Although the ring pattern is not the most realistic traffic pattern for BE traffic, we use it because the purpose of the BE traffic in this simulation is only to disturb the GT traffic and to create heavy traffic conditions. We shall see that what ever the BE traffic load is the GT traffic cannot be disturbed. During the simulation the GT traffic is kept constant while the BE traffic is gradually increased to the point of network saturation. Statistics are collected for the message latencies. The aim is to show that the throughput guarantees given by the network to the GT traffic are not violated by any traffic condition.

The GT traffic generated by a network node has the characteristics of the traffic generated by a HiperLAN/2 receiver [5] – a typical high throughput baseband processing application. Every 4 µs a new message of size 256 Bytes is generated, which equals 512 Mbit/s average throughput per node or 18.4Gbit/s total aggregated throughput for the 36 nodes in the system. The BE traffic generated by a node consists of packets with 10 Bytes payload. The generation period is gradually reduced in order to increase the BE traffic intensity.

We assume that GT message latency is at most 1/3 of the message generating period 4 µs or <1.3 µs. In this way in a real system a PE will spent at most 1/3 of the time transmitting messages, at most 1/3 of the time receiving messages and the rest of the time will be for message processing. On 16 bit network channels (w = 16) this upper bound on the message latency can be achieved if the clock period T_c = 3 ns and the throughput requested by the GT channels is $b/3$ (v=3). The maximal message distance in a 6-by-6 mesh network is N_{max}=10 hops. The length of the GT messages is L=8*256B=2048 bits. According to equation (3) the maximal GT message latency is 414 clock cycles or 1.242 µs. For the simulation, the routing of the communication channels in the network is done such that at most 3 VC are used per network channel (v=3). Therefore, with this simulations setup we expect no GT message latency to exceed the given latency bound of 414 clock cycles. According to equation (1) the minimal throughput of a VC and hence of a communication channel in the network is $TH_{min}=b/3\approx1.8$Gbit/s.

B. Simulation and results

Figure 3 presents the simulation results. The graph shows how the latency of the GT and BE messages depends on the offered BE load. The offered BE load is given per PE as all the PEs generate equal amount of data. For the GT packets the mean and the maximal latency are given. The horizontal line represents the 414-cycle latency guarantee for the GT packets.

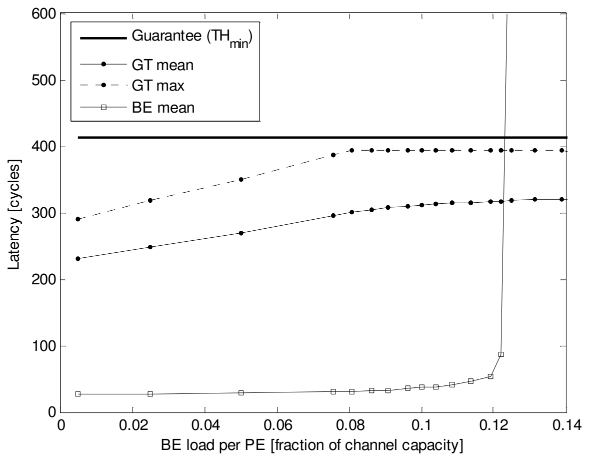

Figure 3 Message delay of the GT and BE traffic vs. BE load for 6-by-6 network

When the offered BE load is low the latency of the GT packets is smaller than the guaranteed latency. The reason is that the GT traffic utilizes the bandwidth unused by the BE traffic. The latency of the GT packets is higher than the latency for the BE traffic because the GT packets are larger than the BE packets. With the increase of the BE load the latency of the GT traffic increases too and at some point it saturates. Further increase of the BE load increases the GT mean latency but the GT maximum latency does not increase and never exceeds the guaranteed latency. The GT maximum latency never reaches the latency bound, because the guarantee given by equation (1) is for worst case conditions when all v VCs transmit data all the time, while in our simulation setup the GT channels transmit data only 1/3 of the time. Thus, even beyond the point of network saturation for the BE traffic there is no GT packet that experience latency higher than 414 cycles – the packet latency is bounded according to the given guarantees.

The GT traffic offered to the network per PE is 512 Mbit/s or 0.09 of the channel capacity w/T_c=5.3Gbit/s. In Section II.D for baseband processing applications the BE traffic is estimated to be 10% of the traffic while the remaining 90% is GT traffic. Thus, the intensity of the BE traffic expected in a real system per PE is about 57 Mbit/s or 0.01 of the channel capacity, which means that the network will operate in the very left part of the graph. According the simulation results the network saturates when the BE load per PE reaches about 0.12 of the channel capacity or 640 Mbit/s - more then ten times the expected BE load.

To estimate the network resource utilisation we count the number of used VCs. From the total number of 480 VCs in the network 52% are used. *Figure 4* shows how the utilized VCs are distributed over the physical channels. Most of the physical channels have two VCs utilized. There is no physical channel with one VC utilized. The reason is that in order to simplify the routing in our setting the BE and GT channel between a pair of nodes are routed together – they follow the same route but on different VCs. Thus, a physical channel is either not used (0 VCs occupied) or used for both BE (one shared VC used by all passing BE communication channels) and GT traffic (1 or 2 VCs occupied by separate GT communication channels).

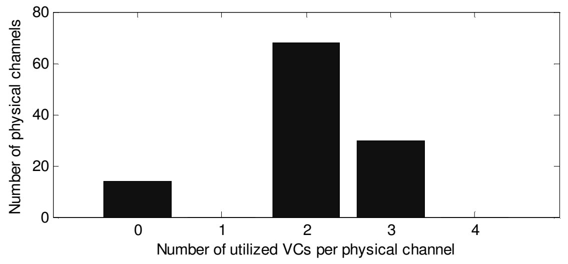

Figure 4 Utilization of the VC in the network load for 6-by-6 network

V. Related work

The RAW processor is a parallel architecture that exploits the applications instruction level parallelism [12]. For interconnecting its processing components it incorporates two types of networks dynamic and static that handle the different classes of traffic. The dynamic network is a dimension-ordered wormhole network, while the static network implements time-division multiplexing. In our solution both types of traffic are handled by a single virtual channel network.

Ethereal is a packet switching NoC solution based on time-division multiplexing (TDM) [13]. It provides both guaranteed and best-effort services and is targeted at general multimedia SoCs. The network is complemented by a network interface [14] providing different communication abstractions to the system components.

aSOC is a framework for on-chip communications in heterogeneous tiled architectures [15][16]. The proposed network implements a kind of advanced time-division multiplexing that can handle efficiently more complex communication patterns than the traditional TDM do. Instead of a simple timetable, each router there has a sequencer that allows more irregular switching behaviour. TDM (as used in aSOC and Ethereal) requires recomputation of a schedule for all the communications in the network even when only one communication link changes which makes the system rather static. We avoid this by using virtual channels instead of TDM. We can add links incrementally without affecting the performance of already allocated links.

Wolkotte et al [9] proposes a circuit switching network which benefits small area and low energy consumption, while providing more flexibility than the traditional circuit switching solutions. Each network channel is divided into four lanes. Switching can be done at different granularity – from single lane to a whole channel. A network interface hides the real channel size from the application. Disadvantage of the circuit switching solution is that it does not support sharing and BE traffic. An additional network is required for configuring the switches and for carrying the BE traffic.

The SPIN network is a wormhole network that uses adaptive routing and is based on a fat tree topology [17]. It is targeted at general multiprocessor systems. While good in performance the topology used for this network is not natural for plane layout. Thus it is not clear whether it can help in structuring the global on-chip wiring.

VI. Conclusion

This paper presented a network-on-chip for a run-time reconfigurable multi-processor system-on-chip. The system is used in mobile multimedia devices where most of the intensive applications are stream-processing applications. A model of the data traffic in the system is constructed and the network is simulated with this traffic model. Considering the communication locality, the simulated network conditions are worst case conditions, since in a real system they will be relaxed by increasing the communication locality. Nevertheless, the network manages to transport the system traffic and is able to provide the requested guaranteed services. The maximum message latency in the network never exceeds the guaranteed latency bound.

The proposed network is suitable for carrying the traffic in a real-time stream-processing system. The network provides guaranteed as well as best effort services. Data steams requiring guaranteed services are efficiently handled by network connections which reserve network resources, while in the same time best effort traffic is handled by allowing network resource sharing. Thus the same network handles both types of traffic. The network requires minimum configuration, (loading address tables), which is done partially and only where and when it is required (only for the nodes which need to be configured). Therefore the network is suitable for dynamic systems where fast reconfiguration is required.

References

[1] T. Whitney and G. Neville-Neil, "SoC: Software, Hardware, Nightmare, Bliss", *ACM Queue*, vol. 1, April 2003, pp. 24-31.

[2] W. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks", *DAC*, June 2001, pp. 684-689.

[3] L. Benini, G. De Micheli, "Networks on Chips: A New SoC Paradigm.", *IEEE Computer*, vol. 35, January 2002, pp. 70-78.

[4] P. Heysters., G. Smit, E. Molenkamp, "A Flexible and Energy-Efficient Coarse-Grained Reconfigurable Architecture for Mobile Systems", *Journal of Supercomputing*, vol. 26, November 2003, pp. 283-308

[5] G. Rauwerda, P Heysters, G. Smit, "Mapping Wireless Communication Algorithms onto a Reconfigurable Architecture", *Journal of Supercomputing*, vol. 30, December 2004, pp 263-282.

[6] P. Wolkotte, G. Smit, L. Smit, "Partitioning of a DRM receiver", *Proceedings of the 9th International OFDM-Workshop*, September 2004, pp. 299-304.

[7] W. Dally, P. Hanrahan, M. Erez, T. Knight, F. Labonté, J. Ahn, N. Jayasena, U. Kapasi, A. Das, J. Gummaraju, I. Buck, "Merrimac: Supercomputing with Streams", *Proceedings of the ACM/IEEE Supercomputing Conference (SC2003)*, November 2003, pp. 35.

[8] W. Dally, "Virtual-channel flow control", *IEEE Transactions on Parallel and Distributed systems*, vol. 3, March, 1992, pp. 194-205.

[9] P. Wolkotte, G. Smit, G. Rauwerda, L. Smit, "An Energy-Efficient Reconfigurable Circuit Switched Network-on-Chip", *Proceedings of the 12th Reconfigurable Architectures Workshop (RAW 2005),* April 2005.

[10] N. Kavaldjiev, G. Smit, P. Jansen, "A Virtual Channel Router for On-chip Networks", *Proceedings of the IEEE International SOC Conference,* September 2004, pp. 289-293.

[11] P. Wolkotte, G. Smit, N. Kavaldijev, J. E. Becker, J. Becker, "Energy Model of Networks-on-Chip and a Bus", *Proceedings of the International Symposium on System-on-Chip,* Tampere, Finland, November 2005

[12] M. Taylor, W. Lee, Saman Amarasinghe, and Anant Agarwal, "Scalar Operand Networks", *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, February 2005, pp. 145-162.

[13] E. Rijpkema, K. Goossens, A. Radulescu, J. Dielissen, J. van Meerbergen, P. Wielage, E. Waterlander, "Trade Offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Networks on Chip", *Proceedings of Design, Automation and Test Conference in Europe*, March 2003, pp. 350-355.

[14] A. Radulescu, J. Dielissen, S. Pestana, O. Gangwal, E. Rijpkema, P. Wielage, K. Goossens, "An Efficient On-Chip Network Interface Offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Programming", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol* 24, January 2005, pp. 4 - 17.

[15] J. Liang, A. Laffely, S. Srinivasan, and R. Tessier, "An Architecture and Compiler for Scalable On-Chip Communication", *IEEE Transactions on VLSI Systems*, vol. 12, July 2004, pp. 711-726.

[16] J. Liang, S. Swaminathan, and R. Tessier, "aSOC: A Scalable, Single-Chip Communications Architecture", *Proceedings of the IEEE International Conference on Parallel Architectures and Compilation Techniques*, October 2000, pp 37-46.

[17] A. Andriahantenaina, H. Charlery, A. Greiner, L. Mortiez, C. Zeferino, "SPIN: a Scalable, Packet Switched, On-Chip Micro-network", *Proceedings of the Design Automation and Test in Europe Conference (DATE'2003) Embedded Software Forum*, March 2003, pp. 70-73.

[18] W. Dally, , B. Towles, "Principles and Practices of Interconnection Networks", Morgan Kaufmann, 2003.