Automatic Hardware-Efficient SoC Integration by QoS Network on Chip

Evgeny Bolotin, Arkadiy Morgenshtein, Israel Cidon, Ran Ginosar and Avinoam Kolodny

Electrical Engineering Department, Technion - Israel Institute of Technology, Haifa 32000, Israel

Abstract— Efficient module integration in Systems on Chip (SoC) is a great challenge. We present a novel automated Network on Chip (NoC) centric integration method for large and complex SoCs. A Quality of Service NoC (QNoC) architecture and its design considerations are presented. Then we describe a chain of design automation tools that allows fast and hardware-efficient SoC integration using the QNoC paradigm. The tool-chain receives a list of system modules and their inter-module communication requirements and results in a complete system hardware and verification models for faster SoC fabrication and easier verification.

1. INTRODUCTION

The complexity of VLSI Systems on Chip (SoCs) is constantly growing. In current technologies a typical SoC comprises tens of system modules and according to technology projections [5][18] this number will grow to several hundreds in the near future. In order to shorten SoC design time and to overcome the growing design productivity gap, most of the SoC modules are purchased or reused from previous in-house projects. Therefore, the main SoC design challenge shifts to adaptation and fast and efficient integration of these Intellectual Property (IP) modules [18]. The main difficulty of integration is the need to cope with severe physical constraints of on-chip interconnect of the future technologies and to provide low cost and efficient communication among system modules. On the other hand, the integration process has to be easily realized to provide short time-to-market of an operational product.

On-chip global communication and module integration has traditionally been addressed by shared-bus structures [15][16][17] and direct inter-module connections. The non-scalability and hardware inefficiencies of these approaches are discussed in [1][2][3][11]. New generations of higher level abstraction methods and design automation tools are required to allow rapid and hardware- efficient SoC integration and consequently a shorter time-to-market [9][18]. To that end, a novel on chip interconnection approach, termed Network on Chip (NoC) was proposed in [1]-[14]. NoCs are shown to be very attractive solutions for the problem of global interconnect in deep sub-micron technologies, and for assuring Quality of Service (QoS) on chip communication [1][13]. In [1], we proposed a generic QoS based NoC architecture and design process termed QNoC which provides an efficient quality of service communication between SoC modules.

In this paper we describe an automated QNoC-centric SoC integration design flow, which allows faster and hardware-efficient system integration. We present a chain of Computer Aided Design (CAD) tools that provide an automatic SoC assembly, integration and verification. The QNoC centric tool-chain presents an attractive solution for the deep-submicron integration problems. Its major steps are: communication based module placement, QNoC topology generation, hardware efficient packet routing, QNoC cost minimization based on link load analysis and network simulations, and finally automatic hardware and verification models generation of the complete system.

The rest of this paper is organized as follows: Section 2 presents the basic QNoC architecture, Section 3 presents our vision of system integration with QNoC and elaborates about the set of design automation tools and their functionality. And finally Section 4 summarizes and discusses future work.

2. QNOC ARCHITECTURE

The QNoC architecture is based on a grid topology and wormhole packet routing. Links are assumed reliable (or made reliable using error correction) and credit based backpressure is applied between stages resulting in a loss-less data forwarding. Packets traverse the network along the shortest route, thus minimizing power dissipation and maximizing network resource utilization.

2.1. QNoC Topology

QNoC comprises routers interconnected by point-to-point links. Network topology can vary depending on system needs and module sizes and placement. Each system module is connected to a router (Figure 1) via a standard interface, whose bandwidth is adapted to the communication needs of that module. In addition, a module may be connected to the network through more than one interface.

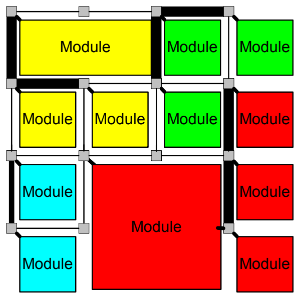


Figure 1. SoC example with QNoC custom topology - irregular customized mesh

The bandwidth of each inter-router link is similarly adjusted to accommodate the expected traffic at the specific link and fulfill overall system QoS requirements (in terms of bandwidth and end-to-end delay) for each class of traffic. Link and interface bandwidth are adjustable by changing the number of wires and the data frequency to achieve uniform utilization at each network link. Finally, unneeded resources such as ports, links, and buffers are trimmed where possible resulting in low cost, QoS based network.

2.2. QNoC Service Levels

One of the main goals of NoC is the support of different QoS requirements. We identify four different types of communication requirements and define appropriate service levels (SL) to support them:

Signaling covers urgent messages and very short packets that are given the highest priority in the network to assure shortest latency. This service level represents interrupts and control signals and alleviates the need for dedicated wires.

Real-Time service level guarantees bandwidth and latency to real-time applications, such as streamed audio and video processing. This service is packet based; a maximal level of guaranteed bandwidth is allocated to each real-time link and should not be violated.

Read/Write (RD/WR) service level provides bus semantics and is designed to support short memory and register accesses.

Block-Transfer service level is used for the transfer of long messages and blocks of data, such as cache refill and DMA transfers.

A priority ranking is established among these service levels, where Signaling is given the highest priority and Block-Transfer the lowest. QNoC employs preemptive communication scheduling where data of a higher priority packet is always transmitted before that of a lower service level (a round-robin is employed within service levels). Additional service levels may be defined if desired. For instance, the RD/WR service level may be split into normal and urgent RD/WR sub-levels.

2.3. QNoC Communication

Packets carry routing information, command and payload. The command field identifies the payload, specifying the type of operation. The packet is divided into multiple flits following [22]. Flit transfer over the inter-router link is controlled by handshake.

2.4. QNoC Routers

Routers connect to up to five links, designed for planar interconnect to four mesh neighbors and to one SoC module. The router forwards packets from input to output ports. Every arriving flit is first stored in an input buffer. On the first flit of a packet, the router invokes a routing algorithm (see Section 2.5)to determine to which output port that packet is destined. The router then schedules the transmission for each flit on the appropriate output port. Each output port of a router is connected to an input port of a next router via a communication link. The output port maintains the number of available flit slots per each service level in the buffer of the next input port. The number is decremented upon transmitting a flit and incremented upon receiving a buffer-credit from the next router. When a space is available, the output port schedules transmission of flits that are buffered at the input ports and waiting for transmission through that output port (Figure 2). There are separate input buffers for each of the four service levels ("direct buffer mapping"). Relatively small buffers are allocated to each service level, capable of storing only a few flits, since buffers are relatively expensive in VLSI (see [2] for buffer-links area tradeoff study). Once a higher priority packet appears on one of the input ports, transmission of the current packet is preempted and the higher priority packet gets through. Transmission of the lower priority packet is resumed only after all higher priority packets have been serviced.

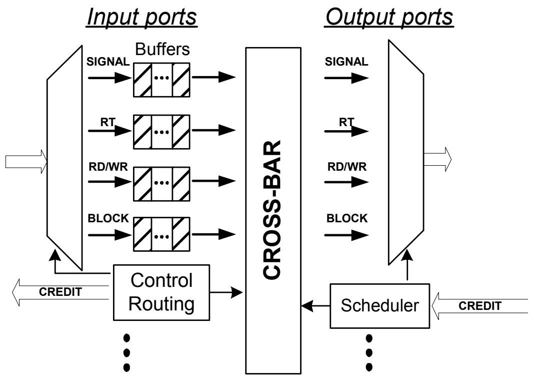


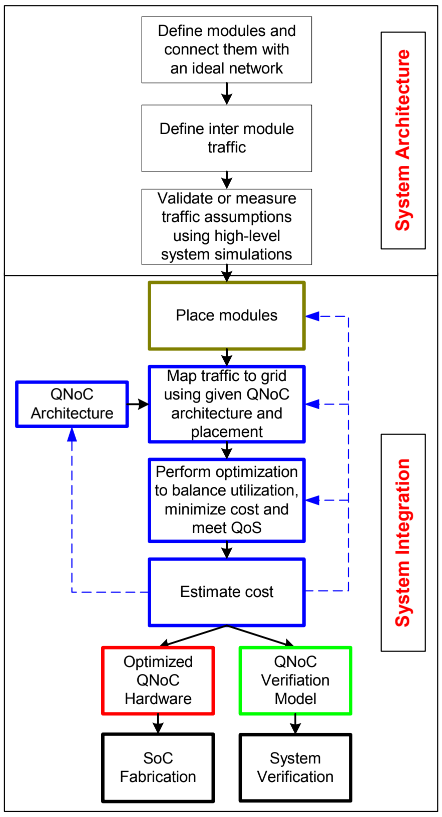
Figure 2. QNoC Router Architecture

2.5. QNoC Maze Routing

In full mesh routing is performed over fixed shortest paths, employing a simple X-Y routing discipline whereby each packet is routed first in an "X" direction and then along the perpendicular dimension or vice versa. However routing in irregular mesh (see Figure 1) recalls routing in a labyrinth. Some links are missing or might lead to a dead end. Therefore simple X-Y scheme cannot always be performed and QNoC maze-routing algorithm is applied. Maze routing algorithm employs simple "around the block" modification where needed. Around the block turns are performed by analyzing the second dimension target index or by small, predefined routing tables. These small routing tables are prepared in QNoC design time by a routing analyzer that is a part of a QNoC automatic generation tool chain (see next section). It analyzes all possible source destination paires in the system and calculates deadlock-free, shortest path routing with minimal deviation from the X-Y based path. Thus only turns that contradict with the X-Y regime are hard-coded in the routing tables, resulting in area efficient implementation. As a result network traffic is distributed non-uniformly over the mesh links, but each link's bandwidth is adjusted to its expected load, achieving an approximately equal level of link utilization across the chip (see next section).

3. SYSTEM INTEGRATION BY QNOC

SoC development flow can be divided into two major phases: system architecture definition and system integration and verification. In the system architecture phase, system requirements are analyzed and main system building blocks are defined. In the integration phase, blocks are designed and adapted and the whole system is assembled and verified. In practice, in order to reduce SoC design time only a few system modules are redesigned or developed from scratch, and the rest of the modules are purchased or reused from previous in-house projects. Therefore, the main SoC design effort is the adaptation and efficient integration of these IPs (see Figure 3). Integration includes placing the functional modules, planning and designing an interconnection architecture that will provide system communication needs and finally a complete verification of the resulting SoC at all levels (hardware and software). The main difficulty of integration is the need to cope with severe physical constraints of future technologies and to provide a low cost and efficient communication infrastructure between system modules. Nevertheless, the integration process has to be simple enough to provide a short time-to-market.



Define modules and connect them with an ideal network
Define inter module traffic
Validate or measure traffic assumptions using high-level system simulations
System Architecture
Place modules
QNoC Architecture
Map traffic to grid using given QNoC architecture and placement
Perform optimization to balance utilization, minimize cost and meet QoS
Estimate cost
System Integration
Optimized QNoC Hardware
QNoC Verifiation Model
SoC Fabrication
System Verification

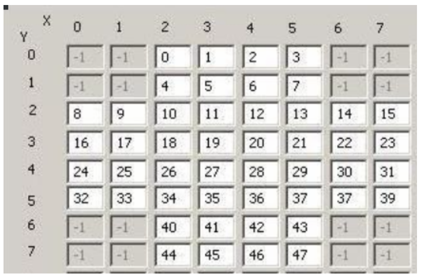
Figure 3. System development flow with QNoC

In [1] we proposed a generic QNoC design process in which a generic network architecture is customized and specific QNoC architecture is synthesized using a-priori known information such as a set of system modules and the given pre-characterized traffic patterns among them. This design process results in a cost-effective inter-module communication infrastructure.

In this section we describe more specifically how this design process is implemented by a chain of CAD tools.

The QNoC based SoC design flow is illustrated in Figure 3. In the system architecture phase we characterize and verify the inter-module traffic and system QoS requirements. The traffic assumptions and QoS requirements can be calculated and verified by high-level system simulations. In the Integration phase the effort shifts to designing and adapting the interconnection network to given traffic flows and QoS requirements and optimizing it for low cost in terms of area and power. Below we describe the main functionality of the future integration automation tools set that was partially implemented at the Technion:

QNoC Placement and Topology generator analyzes the system communication traffic among the modules and the module geometric constraints (shapes and sizes) and derives network topology and placement so as to minimize the overall area and the spatial traffic density that consequently optimize the power and area costs of the system. It also takes into consideration the physical layout constraints and pin-out limitations of the chip.



Y \ X	0	1	2	3	4	5	6	7
0	-1	-1	0	1	2	3	-1	-1
1	-1	-1	4	5	6	7	-1	-1
2	8	9	10	11	12	13	14	15
3	16	17	18	19	20	21	22	23
4	24	25	26	27	28	29	30	31
5	32	33	34	35	36	37	37	39
6	-1	-1	40	41	42	43	-1	-1
7	-1	-1	44	45	46	47	-1	-1

Figure 4. Example of Custom QNoC after module placement, interconnecting 48 SoC modules – a snapshot of a QNoC Customizer Tool user interface

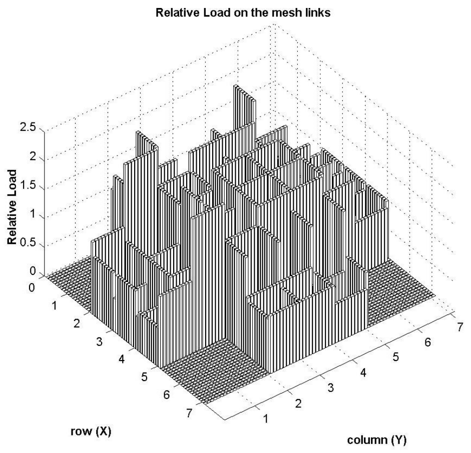


Figure 5. QNoC links relative load (Upper view) for the mesh illustrated in Figure 4 - Link Load Calculator output

QNoC Customizer comprises the *Maze-Router*, *Link Load Calculator* and *QNoC Network Simulator*. The *Customizer* receives a module placement and network topology from the placement tool (see example of placement in Figure 4) and inter-module traffic load and QoS requirements from system architecture phase. Routing paths between sources and destination are derived using the QNoC *Maze-Router* (see section 2.5 for details). Then QNoC architecture for the given SoC is finalized and customized. Architectural parameters, such as location of interconnecting links, each link capacity, number of ports at each router, buffer size, etc. are set according to the number of modules, their spatial placement and the QoS service levels to be supported. Once the routing algorithm is selected, communication paths between all pairs of modules can be determined and link bandwidth optimization can be performed. The required traffic is mapped onto the given topology according to the calculated routing. As parts of the grid are not fully utilized, some vertices and links can be eliminated as shown in Figure 1.Average traffic load at each link is simply calculated by the *Link Load Calculator* since routing is fixed and traffic patterns are known in advance. Relative traffic loads at all the links of the QNoC from Figure 4 are shown in Figure 5, where row and column coordinates represent x-y indices of the network routers. Link bandwidth is assigned proportionally to the calculated load (see Figure 5) at that link by varying the number of wires in a link or its frequency. In that way the tool calibrates the system resources so that average utilization of all links in the network is approximately equal. At this point, the *Load Calculator* provides only relative link bandwidths. To finalize the design, the QNoC can be simulated and analyzed more precisely by a *QNoC Network Simulator* (see [1] for an example) statistically assuring that a given percentage of packets at each service-level arrive to destination with end-to-end delay less or equal to the delay requirement (see Figure 6). Actual bandwidth is assigned to the links according to QoS requirements and the supporting simulation results. Further optimizations are performed: buffers and routers and links are trimmed where possible while maintaining the required QoS and minimize the overall cost function. The described steps are iterated if hardware cost of the resulting QNoC is too high, or if other QNoC architectures or topologies need to be investigated.

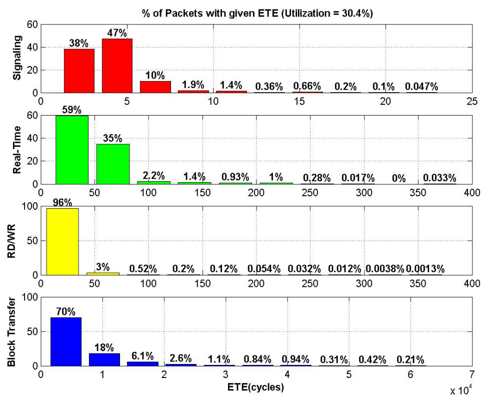


Figure 6. QNoC Simulation results: distribution off end-to-end (ETE) delay for each SL –used for links capacity allocation and statistically ensures that 99.9% of the packets are provided the required QoS.

Hardware Generator creates synthesizable VHDL description of the complete interconnection network based on the topology and QNoC parameters derived by *QNoC Customizer.* It includes appropriate parallelizer/serializer or high frequency transmitters/receiver circuitry at QNoC routers ports that is used for various link bandwidth support, routing tables etc. It also generates interface wrappers for easier connection of the system modules to the network. The tool is based on Perl software that creates the complete hardware description of the system using generic VHDL templates of QNoC components.

System Verification Tool provides QNoC verification models to enable complete system verification platform. It creates reference designs for QNoC modeling for hardware verification simulations such as [19][20]. In addition it can provide higher level models of the derived QNoC for higher level system hardware-software co-verification[21].

4. SUMMARY

In this paper we presented an automated and hardware efficient QNoC centric SoC integration. We describe the basic QNoC architecture and its design considerations. Then we showed how by a chain of CAD tools we can hide the complexity and allow quick and efficient SoC integration and verification. Several components of the tool-chain (such as *QNoC Simulator*, *Link Load Calculator,* parts of the *Hardware Generator and Maze Router*) are already developed and the rest are being researched and constitute a part of our future work.

REFERENCES

[1] E. Bolotin, I. Cidon, R. Ginosar and A. Kolodny, "QNoC: QoS Architecture and Design Process for Networks on Chip", Journal of Systems Architecture, special issue on Network on Chip, vol. 50, February, pp. 105-128, 2004.
[2] E.Bolotin, I.Cidon, R. Ginosar and A. Kolodny, "Cost Considerations in Network on Chip", Integration - the VLSI Journal, special issue on Network on Chip, 2004.
[3] William J. Dally and Brian Towles, " Route Packets, Not Wires: On-Chip Interconnection Networks", DAC 2001, Las Vegas, Nevada, USA, June, 2001.
[4] John Dielissen, Andrei Radulescu, Kees Goossens, and Edwin Rijpkema, "Concepts and Implementation of the Philips Network-on-Chip". IP-Based SOC Design, Grenoble, November, 2003.
[5] Jian Liu , Li-Rong Zheng and Hannu Tenhunen, "Interconnect intellectual property for Network-on-Chip (NoC)", Journal of Systems Architecture special issue on Network on Chip, vol. 50, February 2004.
[6] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, A. Sangiovanni-Vincentelli, "Addressing the System-on-a-Chip Interconnect Woes Through Communication-Based Design", Design Automation Conference, June, 2001.
[7] Luca Benini, Giovanni De Micheli, "Networks on Chips: A New SoC Paradigm", IEEE Computer, no. 35, vol. 1, pp. 70-78, 2002.
[8] Shashi Kumar, Axel Jantsch, Juha-Pekka Soininen, Martti Forsell, Mikael Millberg, Johny Oberg, Kari Tiensyrja and Ahmed Hemani, "A Network on Chip Architecture and Design Methodology", Proceedings of the IEEE Computer Society Annual Symposium on VLSI, 2002.
[9] J. Soinen, H. Heusala, "A design Methodology for NoC based Systems", Networks on Chip, Kluwer Academic Publishers, Boston, pp. 19-38, 2003.
[10] Ahmed Hemani, Axel Jantsch, Shashi Kumar, Adam Postula, Johnny Oberg, Mikael Millberg, Dan Lindqvist, "Network on a Chip: An architecture for billion transistor era", In Proceeding of the IEEE NorChip Conference, November, 2000.
[11] Pierre Guerrier , Alain Greiner,"A generic architecture for on-chip packet-switched interconnections", Proceedings of Design, Automation and Test in Europe Conference and Exhibition., pp. 250 –256, 2000.
[12] E.Rijpkema, K. Goosens and P.Wielage, " A Router Architecture for Networks on Silicon", Proceedings of Progress, 2nd workshop on embedded systems, 2001.
[13] K. Goossens, J. van Meerbergen, A. Peeters, and P. Wielage, "Networks on Silicon: Combining Best-Effort And Guaranteed Services", DATE 2002, Design automation and test conference, March, 2002.
[14] Paul Wielage and Kees Goossens, "Networks on Silicon: Blessing or Nightmare?", Euromicro Symposium On Digital System Design (DSD 2002), Dortmund, Germany, September, 2002.
[15] "AMBA Specification", Arm Inc, May, 1999.
[16] "The CoreConnect Bus Architecture", IBM, 1999.
[17] Drew Wingard, " MicroNetwork-based integration of SOCs", In Proceedings of the 38th Design Automation Conference, June, 2001.
[18] The International Technology Roadmap for Semiconductors (ITRS) 2003 edition. Design Chapter.
[19] Specman Elite by Verisity (www.verisity.com)
[20] OpenVera by Synopsis (www.synopsis.com)
[21] SystemC (www.systemc.org)
[22] W. J. Dally, "A VLSI Architecture for Concurrent Data Structures", Kluwer Academic Publishers, 1987.