A Methodology for Design, Modeling, and Analysis of Networks-on-Chip

Jiang Xu, Wayne Wolf
EE, Princeton University
{jiangxu, wolf}@princeton.edu

Joerg Henkel
CS, University of Karlsruhe
henkel@informatik.uni-karlsruhe.de

Srimat Chakradhar
NEC Laboratories America, Inc.
chak@nec-labs.com

Abstract --- *In this paper, we present an architecture-level methodology for modeling, analysis, and design of networks-on-chip (NoC), and we tested it through two NoC designs. Our methodology 1) can be used to design both regular-topology NoC and application-specific NoC, 2) uses real communication traces to guide design and performance analysis, 3) is based on circuit-level models and floorplan to estimate power and area, and 4) uses widely available tools, OPNET, Design Compiler, and SPICE, which make the methodology ready to use. The methodology can quickly and accurately estimate the performance, power, and area of a NoC at architecture level, and it can efficiently model different types of NoCs. The methodology can be easily incorporated into a traditional design flow. Using this methodology, we designed a bus-based NoC and a crossbar-based NoC for a high-performance embedded video SoC design in a 0.13um technology. We analyzed their performance, power, and area in detail. We find that the crossbar-based NoC not only has 62.5% higher performance but also consumes 82% less power compared to the bus-based NoC. The crossbar-based NoC uses 88% less silicon area and 123% more metal area than the bus-based NoC. This methodology also helped to refine the crossbar-based NoC design and save additional 57% power. Our study shows that interconnections dominate power and area in NoC designs, and only comparing the logic circuits of NoCs will give wrong conclusions.*

I. INTRODUCTION

As an effective way to reduce cost, improve reliability, and produce versatile products, system-on-chip (SoC) design not only implements function units, but also emphasizes cooperation among function units to improve performance and reduce cost. On-chip communication subsystem decides how effective the cooperation is, and it gradually grows from ad-hoc interconnections into sophisticated networks-on-chip (NoC). NoC was introduced by several researchers [1] [5] [6] [7]. Many on-chip communication architectures were developed based on buses, for example, AMBA from ARM [9], CoreConnect from IBM [13], MicroNetwork from Sonics [10], and Wishbone from Silicore [20]. Other on-chip communication architectures were inspired by multi-computer networks and telecommunication networks [12] [15] [23] [24] [25].

Several NoC design methodologies were introduced in previous work. Kumar presented a design methodology specifically for a 2-D mesh type of NoC [3]. Siegmund presented a methodology to model and synthesize NoCs for SystemC designs [11]. Jalabert proposed a tool to instantiate NoCs based on user-defined topologies [4]. Goossens proposed a methodology to provide both guaranteed and best-effort services in NoC [2]. These methodologies are confined to either specific NoC topologies or customized NoC technologies and components. Power and area are considered in the NoC methodologies, but not accurate. A methodology, which can efficiently design any types of NoCs and give accurate performance, power, and area results, can help designers to choose different NoCs for different applications.

In this paper, we propose a general architecture-level methodology for modeling, analysis, design of networks-on-chip (NoC). Compared to previous work, our methodology can be used to design any types of NoCs. It helps to efficiently design a NoC and accurately analyze all three import characters, performance, power, and area (both the metal area and silicon area). Our methodology can be easily incorporated into the traditional design flow. It adapts OPNET [19] and uses Design Compiler and SPICE. We tested the methodology by two NoC designs, a bus-based NoC and a crossbar-based NoC, which were designed for a high-performance embedded video system-on-chip.

The next section gives an overview of the methodology. Corresponding steps in the methodology are described in detail and illustrated in sections III through X. Section XI shows the time spent on each step and draws the conclusions about our methodology.

II. NOC DESIGN METHODOLOGY

In this section, we give an overview of the methodology, and more details will be illustrated by examples in the following sections. Our methodology starts with the computation architecture design (figure 1). Computation architecture describes the behaviors of the designed system and the computation units used to implement the system functions. One method of computation architecture design is using two models---behavior model and computation architecture model. The behavior model is partitioned and mapped to the computation architecture model.

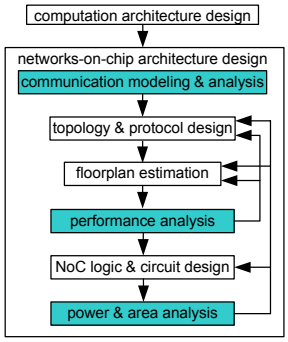

Fig. 1. A methodology for NoC

NoC architecture design follows the computation architecture design, which partitions a behavior model and maps it onto an computation architecture model. The first step is communication modeling and analysis. Communication patterns can be gotten from communication traces in the computation architecture simulation, and it shows the communication requirements, which are the key of NoC design. Communication analysis extracts the communication patterns of the application and computation architecture. The pattern includes communication types, information sizes, and communication frequencies among all the computation units. Communication pattern information is used to direct the design of protocol and topology.

The second step is topology and protocol design based on the communication requirements. The aim of this step is to use the least network resource to fulfill the communication requirements showed by the communication pattern. A matured telecommunication system design and simulation environment, OPNET [19], is adapted for this step.

To estimate the delay on each interconnection in term of clock cycles and analyze power and area of a NoC design, the chip floorplan should be estimated in the third step to get some design requirements for interconnections. In NoC design, instead of logic circuits, interconnections dominate power consumption and area.

The fourth step, performance analysis, simulates the NoC architecture with the communication traces from the computation architecture in OPNET. The performance results help to compare

different design choices and refine the designs. The minimum system frequency will be gotten from the performance. If no design meets the performance requirements or has too small performance margin, we have to go back to previous steps.

In the fifth step, logic design can help increase the accuracy of power and area analysis, but the interconnection design is the centre of the analysis. We use Design Compiler [22] to synthesize logic circuits and use Cadence SPICE [21] as the interconnection design and simulation environment. It is also possible to automate the interconnection design by building an interconnection library, which simplifies both the design task to look up a table and the floorplan estimation.

The last step is power and area analysis. Worst-case power to transmit 1-bit information is measured for each interconnection using SPICE. Worst-case power of logic circuit is estimated based on the number of transistors. OPNET records all the activity of NoC. By summing the power of each activity, we get the worst-case power of the NoC. Silicon and metal area usages are gotten by adding up all the design parts. If the power or the area do not meet the requirements, we will go back to choose different circuits or redesign the floorplan. In the worst case, we have to redesign the topology and protocol.

To test our methodology, we designed two NoCs for a high-performance embedded video system-on-chip (SoC). The following sections illustrate each step of the methodology.

III. COMPUTATION ARCHITECTURE DESIGN

Computation architecture describes the behaviors of the designed system and the computation units used to implement the system functions. The computation architecture design defines the requirements with which the communication architecture would comply. One method of computation architecture design is using two models---behavior model and computation architecture model--and partitioning and mapping the behavior model to the computation architecture model (figure 2a).

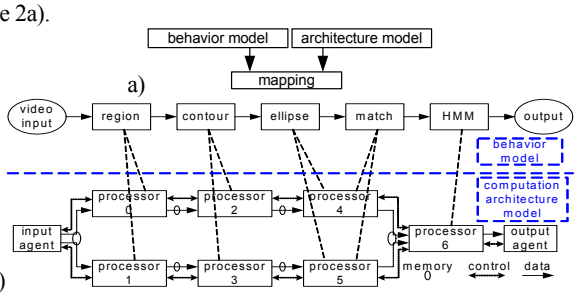

Fig. 2. Computation architecture design, a) method, b) example

The Smart Camera system [8] is an embedded video processing application that can process 150 frames per second. To deliver such a high performance a dual-pipeline computation architecture is designed (Figure 2b). Each video frame will go through 5 processing stages, which are region, contour, ellipse, match, and HMM. Processors P0, P2, P4, and P1, P3, P5 form the two pipelines, and processor P6 works for both pipelines. Based on the workload, processor P0 handles region, P2 handles contour, P4 handles both ellipse and match, and P6 handles HMM for the both pipelines. Since the Smart Camera system will be put into a compact space along with other parts, low power consumption will help to reduce the heat-related cost, and smaller chip area can reduce the manufacture cost.

IV. COMMUNICATION MODELING AND ANALYSIS

Communication trace (figure 3a) recorded from computation architecture simulation is a good source for communication analysis. Each computation unit has its own trace. A trace has entries to record network accesses. An entry includes access interval (between current and the last access) in term of number of clock cycle, source, destination, operation type, address, data size, and other special information related to a particular computation unit. Those traces give the communication pattern for a specific application and will help to design a suitable NoC in the next step.

Traces are also used in performance analysis, where they control the communication behaviors of corresponding computation units. To accelerate the design and simulation, statistical traces can be generated from either the recorded trace or the behavior model (figure 3b). In our designs, we used Poisson distributions. Compared to using recorded traces, we got a 2~3X speedup in performance analysis and about ±5% inaccuracy.

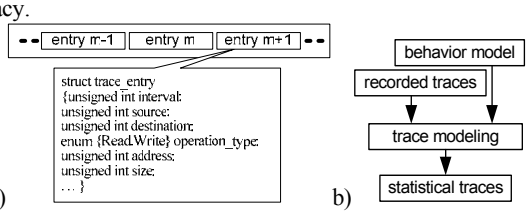

Fig. 3. a) Recorded communication trace, b) Trace modeling

V. TOPOLOGY AND PROTOCOL DESIGN

We suggest using the method in figure 4 to design topology and protocol. From the communication trace, requirements for a NoC can be derived. Some requirements can be fulfilled directly: the connectivity among computation units, the maximum bandwidth, and shortest delay. Other requirements can only be found out by simulation, for example the average bandwidth, average delay, and maximum delay. However a well established protocol tends to give more insides even before running simulations. This approach also helps us reuse NoC components.

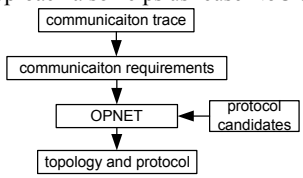

Fig. 4. Topology and protocol design method

OPNET is used as the design environment in this step, and it is also used as the NoC simulator in performance analysis. OPNET can model many common phenomena in communication systems. It is also a packet-based simulator, and this feature is particularly useful for most NoC designs. OPNET also has some disadvantages, and some adaptations are required to use OPNET for NoCs. First, it is developed for simulating telecommunication systems and some on-chip features are not well supported. For example, the smallest time unit is the second, where on-chip communication architectures need nanosecond or even picosecond, and the smallest distance unit is meter instead of micrometer. Second, OPNET assumes asynchronous communication, so for a synchronous system, designers have to explicitly design a clock scheme and a distribution network. We made the following adaptations in OPNET. In link models, 1) disable the propagation delay pipeline stage and 2) disable the error model. In transmitter and receiver models, set data rate high enough to eliminate the effects of transmission delay. (We set data rate to 10^6 bps, which introduces 1 μs transmission delay.) In all node models, state transitions should be on clock edges. For the clock, 1) use one second to represent one clock cycle and 2) build a clock bus to synchronize the system. We've shown some of these changes in a previous work [14].

We chose to design a bus-based NoC and a crossbar-based NoC for the Smart Camera system. In the bus-based system, a bus connects all nodes. A node may be a processor, the input agent, the output agent, or the on-chip memory. The arbiter decides who can use the bus based on a priority list and the status of its neighbor processing stages and of its own. If the previous stage finishes a frame and there is enough space to store the result in its own share of memory, a processor can process the finished frame from previous stage. In crossbar-based system, packet switching is used. The crossbar and a control unit form a switch. The control unit schedules arriving packets to be sent and resolves the competing requests for the same port. The scheduling is based on the node priorities and their status. Packets are used to transfer data and other information among nodes. There are 4 types of packets, read packet, write packet, read response packet, and switch response packet.

VI. FLOORPLAN ESTIMATION

We estimate the floorplan to get the length of each interconnection. The length can decide the delay on the interconnection in term of clock cycles. In NoC design, instead of logic circuits, interconnections dominate power consumption and area. While performance of a NoC design can be estimated at architecture level, power consumption and area of a NoC design can not be accurately estimated without interconnections design. Unlike logic circuits, interconnections have simple structures and are easy to design if the requirements are known. The requirements for interconnections include the length, the number and positions of input drivers and output drivers, and the working frequency. Length and position can be given based on an estimation of the chip floorplan, where only sizes of function units and their relative positions are needed. The IP reuse can make the estimation more accurate.

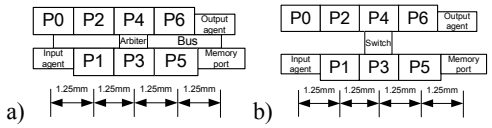

Fig. 5. Floorplans, a) bus-based system, b) crossbar-based system

We use Plasma core [18] for each processor. A Plasma core needs about 1M transistors to implement. Approximate floorplans for the bus-based and crossbar-based system are shown in figure 5. Instead of the best cases, average cases are considered in this step. We use the average length between two units, which are started and ended in the middle of the units. The bus is about 5mm long and the arbiter is put in the centre. The interconnections in crossbar-based system are 2.5 mm and 1.25 mm. The on-chip memory surrounds the processors and is not shown in the figure.

VII. PERFORMANCE ANALYSIS

After topology and protocol design and floorplan estimation, different NoC designs can be simulated in OPNET. C is used to model the behavior of each NoC component. Communication traces will be used to control the communication behavior of each computation node. And the simulation results help to choose potential NoC designs or revise a design. Simulation speed is very important for performance analysis, which needs to simulate billions of clock cycles to get meaningful application-level results. OPNET has a good simulation speed.

We simulated each NoC design for $3x10^8$ clock cycles, and the results are shown in table 1. We define the system performance as the number of frames being processed in a fixed time. The results show that the crossbar-based system has 62.5% higher performance over the bus-based NoC. Using statistical traces, the results have about ±5% difference compared to using recorded traces. However, the simulation has a 2~3X speedup by using statistical traces. Because the system need to process 150 frames per second, we use the following formula to determine the minimum system frequencies.

$$Minimum \quad system \quad frequency(Hz) = \frac{3\times10^8 \times 150(frame/\sec)}{processed \quad frames(frame)}$$

Table 1. Performance analysis result

	bus-based	crossbar-based
processed frame in $3x10^8$ cycles	56	91
min. system frequency (MHz)	804	495
performance improvement	--	62.5%
processed frame using statistical traces	54	95
difference	3.6%	4.4%

VIII. NOC LOGIC AND CIRCUIT DESIGN

Compared to the interconnections, NoC logic only requires very little power and area. People can choose to only design the interconnection circuit without the NoC logic. The behaviors of logic circuits are defined by topology and protocol design. The behaviors should be represented in a hardware description language and then synthesized. We used Verilog to represent the logic circuits and synthesized them using Design Compiler. Other languages and synthesized tools can also be used.

Interconnection design requires a good wire model and transistor model (figure 6). The design uses SPICE to find out the proper sizes and number of transistors according to the requirements. Lengths are given by the floorplan estimation, and working frequency is given by performance analysis. Interconnection design is easily automated by building a table of interconnections for the target technology, which simplifies the design to table lookup. Interconnection can use different technologies, such as low-swing, differential, and wave-pipelining [16]. Based on performance analysis, we choose 1GHz system frequency for both the bus-based and crossbar-based system. We choose 130nm aluminum technology, which has a 1.5v power supply. We use the typical wire dimensions from the Berkeley Predictive Technology Model [17]. We model a wire as a fine-grained lumped RLC network, and consider the coupling capacitance. Since the coupling inductance has a significant effect at 130 nm technology, mutual inductances are considered up to the 3rd neighboring wires. Interconnections were designed and simulated in Cadence SPICE.

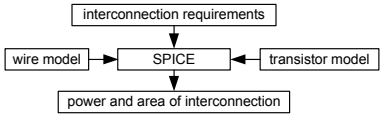

Fig. 6. Interconnection design method

The bus-based NoC includes the arbiter and interconnections, which are composed by input driver chains, transmission gates, metal wires, and output drivers. The bus interconnections use global metal layer, and the arbiter control interconnections use intermediate metal layer. The crossbar-based NoC has four parts, the control unit, links between the nodes and the switch, input buffers, and the crossbar. The interconnections in links are on the global metal layer.

IX. POWER ANALYSIS

A Power

The power of a NoC design is calculated as the following formula. Each part of NoC is designed and simulated separated. The number of each type of activity is recorded during OPNET performance analysis. The power of each type of activity is measure in SPICE simulation. The total power is the sum of power of all types of activity on all NoC parts. Worst-case power is used for each NoC part. For logic circuits, the worst case happens when all the transistors switch at the same time. For interconnections, all the input patterns are simulated, and the one using the most power is the worst case.

$$Power(J) = \sum_{activity \quad types} No. \quad of \quad activities \times power \quad per \quad activity(J)$$

Table 2. Power of bus-based NoC

arbiter	2.76 pJ/cycle
control bus	5.79 pJ/bit
address bus	5.56 pJ/bit
data bus	5.57 pJ/bit
1.25mm arbiter control	0.35 pJ/bit
2.5mm arbiter control	0.71 pJ/bit

Table3. Power of crossbar-based NoC

control unit	4.61 pJ/cycle
crossbar	0.30 pJ/bit
input buffer	0.014 pJ/bit
link (1.25mm)	0.67 pJ/bit
link (2.5mm)	1.28 pJ/bit

Tables 2 and 3 list the power for each part in the bus-based and crossbar-based NoCs respectively. In the bus-based NoC, bus interconnections are the most power-hungry. Considering the number of interconnections in the bus, the arbiter uses less than 1% power of all the bus interconnections. Arbiter control interconnections also consume relative little power. In the crossbar-based NoC, considering the number of interconnections, the control unit consumes less power than the interconnections.

Table 4. Average power consumption of processing one frame

	bus-based NoC	crossbar-based NoC
process one frame	334 uJ	137 uJ
overhead	43%	44%

Table 4 shows the average power consumed by the two NoCs to process a frame. The crossbar-based NoC uses 59% less power than the

bus-based NoC to process a frame. In addition to data, some overhead information transferred by the NoCs is used to facilitate data transfer. The overheads are about the same in both NoCs. The overhead is 43% in the bus-based NoC and 44% in the crossbar-based NoC.

B Additional Power Saving

A detailed analysis on the power consumption helps us to save more power. In the bus-based NoC, the position of each node only affects power consumption on the arbiter control interconnections, which accounts for less than 0.4% of total power consumption, and the power consumed by the bus is not affected by the node position. However, in the crossbar-based NoC, the node position directly affects the power consumption. Table 5 shows the power consumption for read operation between different node positions. The rows are request node positions, and the columns are the target node positions. We mark the positions from the left to right in figure 6 as 0 to 4. For example, the input agent and processor P0 are at position 0, the processors P3 and P4 are at position 2, and so on. Based on the table 5, we refine the floorplan as figure 7. Since all the processors are the same, we only change the functions of some processors and move the memory port to the centre. The refined floorplan only needs 59uJ to process a frame, which saves additional 57% power. And this also makes the crossbar-based NoC uses 82% less power than the bus-based NoC. A better floorplan need to move the switch more left, and the exact position is decided by the communication traffic, which is affected by the Smart Camera algorithm.

Table 5. Power consumption between different positions

position	0	1	2	3	4
0	164pJ	130 pJ	94 pJ	130 pJ	164 pJ
1	130 pJ	96 pJ	59 pJ	96 pJ	130 pJ
2	92 pJ	59 pJ	22 pJ	59 pJ	92 pJ
3	130 pJ	96 pJ	59 pJ	96 pJ	130 pJ
4	164 pJ	130 pJ	94 pJ	130 pJ	164 pJ

Table 6. Silicon and metal area

	bus-based	crossbar-based
silicon area	0.51mm^2	0.064mm^2
metal area	0.35mm^2	0.78mm^2

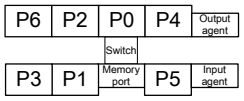

Fig. 7. Refined floorplan

X. AREA ANALYSIS

We estimate silicon area and metal area in the bus-based NoC and crossbar-based NoC in table 6. The silicon area is the sum of all the logic circuit, interconnection drivers, and transmission gate. For the crossbar-based NoC, the silicon area also includes the input buffers. The metal area includes all the interconnections, except the interconnections in logic circuits. The crossbar-based NoC uses only 12.5% the silicon area of the bus-based NoC. Logic circuits only account for 0.7% of the silicon area in the bus-based NoC and 12% of the silicon area in the crossbar-based NoC. The crossbar-based NoC use 123% more metal area than the bus-based NoC. Since our crossbar design is not compact, the crossbar-based NoC could use less metal area.

XI. RESULTS AND COMPARISON

Table 7. Approximated times spent on each step (person*day)

	bus-based	crossbar-based
comm. modeling & analysis	7	
topology & protocol design	3	8
floorplan estimation	2	2
performance analysis	7	10
logic circuit design	3	5
NoC logic & circuit design	7	9
power & area analysis	3	3
total	32	43

Table 7 lists the times we spent on each steps. Because the communication analysis, performance analysis and interconnection designs involve a lot of modeling and simulation tasks, those steps are most time consuming. Compared to other methodologies, our methodology 1) uses real communication trace to guide design and performance analysis, 2) is based on circuit-level model to estimate power and area, and 3) use widely available tools, OPNET, Design Compiler, and SPICE. These differences lend our methodology several advantages. First, the methodology can efficiently design different types of NoCs. Second, it can quickly and accurately estimate the performance, power, and area of a NoC at architecture level. Third, it uses communication trace which improves the speed and quality of design and analysis. Fourth, it uses widely available tools which make it ready to use.

XII. CONCLUSIONS

In this paper, we presented an architecture-level methodology for design, modeling, and analysis of networks-on-chip (NoC), and we tested it through two NoC designs. Our methodology is based on widely available tools, which make the methodology ready to use. Our methodology not only can help to efficiently design and model NoC but also can accurately estimate the performance, power, and area of NoC. The methodology can be easily incorporated into traditional design flow. Using the methodology, we designed a bus-based NoC and a crossbar-based NoC for a high-performance embedded video SoC. This methodology helped refine the crossbar-based NoC design and saved additional 57% power. Our methodology shows that interconnections, not logic circuits, dominate power and area in NoC designs, and only compared the logic circuits of NoCs will give wrong conclusions.

REFERENCES

[1] L. Benini, G. De Micheli, "Networks on chip: a new paradigm for systems on chip design", *Design, Automation and Test in Europe Conference and Exhibition*, 2002

[2] K. Goossens, J. van Meerbergen, A. Peeters, and P. Wielage, "Networks on Silicon: Combining Best-Effort And Guaranteed Services" *Design, Automation and Test in Europe Conference and Exhibition*, 2002

[3] S. Kumar, A. Jantsch, Juha-Pekka Soininen, M. Forsell, M. Millberg, J. Öberg, K. Tiensyrjä, A. Hemani, "A network on chip architecture and design methodology", *IEEE Computer Society Annual Symposium on VLSI*, 2002

[4] A. Jalabert, S. Murali, L. Benini, G. De Micheli,"XpipesCompiler: a tool for instantiating application specific networks on chip", *Design, Automation and Test in Europe Conference and Exhibition*, 2004

[5] W. Dally, B. Towles, "Route packets, not wires: on-chip interconnection networks", *Design Automation Conference*, 2001

[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*, 2001.

[7] A. Hemani, A. Jantsch, etc, "Network on chip: An architecture for billion transistor era", *IEEE NorChip Conference*, 2000

[8] W. Wolf, B. Ozer, T. Lv, "Smart cameras for embedded systems," *IEEE Computer*, 35(9), September 2002, pp. 48-53

[9] D. Flynn, "AMBA: enabling reusable on-chip designs", *IEEE Micro*, 17(4), 1997

[10] D. Wingard, "MicroNetwork-based integration for SOCs", *Design Automation Conference*, 18-22 June 2001

[11] R. Siegmund, D. Muller, "Efficient modeling and synthesis of on-chip communication protocols for network-on-chip design", *IEEE International Symposium on Circuits and Systems*, 2003

[12] J. Liang, S. Swaminathan, R. Tessier, "aSoC: a scalable, single-chip communications architecture", International Conference on Parallel Architectures and Compilation Techniques, 2000

[13] R. Hofmann, B. Drerup, "Next generation CoreConnect processor local bus architecture", *Annual IEEE International ASIC/SOC Conference*, 25-28 , 2002

[14] J. Xu, W. Wolf, Joerg Henkel, Srimat Chakradhar, Tiehan Lv, "A Case Study in Networks-on-Chip Design for Embedded Video", *Design, Automation and Test in Europe Conference and Exhibition,* 2004

[15] M. Millberg, E. Nilsson, R. Thid, S. Kumar, A. Jantsch, "The Nostrum backbone-a communication protocol stack for Networks on Chip", VLSI Design, 2004

[16] J. Xu, W. Wolf, "Wave Pipelining for Application-specific Networks-on-Chip", *Compilers, Architecture, and Synthesis for Embedded System*, 2002

[17] http://www-device.eecs.berkeley.edu/~ptm/interconnect.html

[18] www.opencores.org

[19] www.opnet.com

[20] www.silicore.net

[21] www.cadance.com

[22] www.synopsis.com

[23] M. Taylor, "The Raw prototype design documentation" v5.00, 2003

[24] A. Adriahantenaina, H. Charlery, A. Greiner, L. Mortiez, C.A. Zeferino, "SPIN: a scalable, packet switched, on-chip micro-network", *DATE* 2003

[25] M. Forsell, "A scalable high-performance computing solution for networks on chips", *IEEE Micro*, Volume: 22 , Issue: 5 , 2002