# A Design Methodology for Application-Specific Networks-on-Chip

JIANG XU and WAYNE WOLF
Princeton University
JOERG HENKEL
University of Karlsruhe
and
SRIMAT CHAKRADHAR
NEC Laboratories America, Inc.

With the help of HW/SW codesign, system-on-chip (SoC) can effectively reduce cost, improve reliability, and produce versatile products. The growing complexity of SoC designs makes on-chip communication subsystem design as important as computation subsystem design. While a number of codesign methodologies have been proposed for on-chip computation subsystems, many works are needed for on-chip communication subsystems. This paper proposes application-specific networks-on-chip (ASNoC) and its design methodology. ASNoC is used for two high-performance SoC applications. The methodology (1) can automatically generate optimized ASNoC for different applications, (2) can generate a corresponding distributed shared memory along with an ASNoC, (3) can use both recorded and statistical communication traces for cycle-accurate performance analysis, (4) is based on standardized network component library and floorplan to estimate power and area, (5) adapts an industrial-grade network modeling and simulation environment, OPNET, which makes the methodology ready to use, and (6) can be easily integrated into current HW/SW codesign flow. Using the methodology, ASNoC is generated for a H.264 HDTV decoder SoC and Smart Camera SoC. ASNoC and 2D mesh networks-on-chip are compared in performance, power, and area in detail. The comparison results show that ASNoC provide substantial improvements in power, performance, and cost compared to 2D mesh networks-on-chip. In the H.264 HDTV decoder SoC, ASNoC uses 39% less power, 59% less silicon area, 74% less metal area, 63% less switch capacity, and 69% less interconnection capacity to achieve 2X performance compared to 2D mesh networks-on-chip.

Categories and Subject Descriptors: B.7.0 [**General**]; C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems; C.5.4 [**VLSI Systems**]

General Terms: Algorithms, Design, Performance, Theory

Additional Key Words and Phrases: Application-specific, networks-on-chip, architecture, regular topology, methodology

This research was supported by NEC with additionl support by NSF grant CCF0325119.
Authors' addresses: Jiang Xu and Wayne Wolf, Department of Electrical Engineering, Princeton University, Princeton NJ 08544; email: jiangxu@princetion.edu; Joerg Henkel, University of Karlsruhe,Germany; Srimat Chakradhar; NEC Laboratories America, Inc,Princeton NJ 08540.

## 1. INTRODUCTION

As an effective way to reduce cost, improve reliability, and produce versatile products [Wolf 2001], system-on-chip (SoC) 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. While a number of HW/SW codesign works have been proposed for on-chip computation subsystems, a lot of works are still needed for on-chip communication subsystems. In this paper, we propose a design methodology for application-specific networks-on-chip (ASNoC).

### 1.1 Previous Work

Two challenges gradually move on-chip communication subsystem from bus and adhoc interconnection to more sophisticated NoC. The migration toward NoC is propelled by reduced feature sizes in each generation of process technology. On one hand, smaller transistors allow more and better functions on a single chip. This results in more computations as well as more on-chip communications. On the other hand, reduced feature sizes make on-chip communication more difficult. Global interconnection delays increase exponentially, as NAND gate delays decrease exponentially [Sematech 2004]. On-chip communication will need not one but multiple clock cycles [Cong 2001]. Cross-talk noise becomes significant and poses great threats on signal integrity [Kuhlmann and Sapatnekar 2001]. Facing these two challenges, state-of-art communication network theories and technologies are used to systemically study and design more sophisticated and efficient on-chip communication subsystems.

Many on-chip communication architectures were developed based on on-chip buses, for example, AMBA from ARM [Flynn 1997], CoreConnect from IBM [Hofmann and Drerup 2002], MicroNetwork from Sonics [Wingard 2001], and Wishbone from Silicore [Silicore]. For high-performance SoCs, bus architectures often fail to deliver required throughput. For large SoCs with multiply IPs (intellectual property), bus architectures are not power-efficient and need large chip areas. The root of these issues is that bus architectures use shared medium. Hierarchical bus architectures are proposed to improve the throughput and power efficiency [Ryu et al. 2001]. Those improvements are brought by switching technologies used in the hierarchical bus architectures.

Regular-topology NoC is proposed as on-chip communication architectures primarily using switching and routing technologies [Benini and De Michelli 2002; Dally and Towles 2001; Goossens et al. 2003; Sgroi et al. 2001; Hemani and Jantsch 2000]. A two-dimensional (2D) folded torus NoC is proposed in Dally and Towles [2001]. Two-dimensional mesh NoC, such as CLICHÉ, Nostrum, Eclipse, and aSoC, are presented in Kumar et al. [2002], Millberg et al. [2004], Forsell [2002], and Liang et al. [2000] respectively. RAW is a multiprocessor system based on a 2D mesh NoC [Taylor et al. 2002]. SoCIN uses a 2D mesh or torus [Zeferino and Susin 2003]. SPIN has a fat-tree topology [Adriahantenaina et al. 2003]. Octagon has a fixed topology [Karim et al. 2002]. Proteo uses a ring topology [Siguenza-Tortosa and Nurmi 2002]. All those NoCs have either regular or fixed topologies. Compared to on-chip buses, regular-topology NoC

has much higher throughput and better scalability. 2D mesh topology is preferred in most studies, because of its simplicity and corresponding tile-based floorplan.

NoC design methodologies and tools were introduced in previous work. SUNMAP is a tool for selecting a standard NoC from a library based on different requirements [Murali and De Micheli 2004]. Hu and Marculescu [2003] presented a method to map IPs onto a 2-D mesh network. Lahiri et al. [2004] presented a methodology to map an application to a given communication architecture. Daveau et al. [1997] presented a methodology to select protocols and generate an interface in HW/SW codesign. XpipesCompiler is a tool for instantiating NoCs based on user-defined topologies [Jalabert et al. 2004]. Orion is a power and performance simulator for interconnections [Wang et al. 2002]. Zhu et al. [2004] proposed a framework for modeling and simulating on-chip communications. Kumar et al. [2002] presented a design methodology for CLICHÉ NoC. Goossens et al. [2003] proposed a methodology to provide guaranteed services in NoCs. Siegmund and Muller [2003] presented a methodology to model and synthesize protocols for NoCs. Pinto et al. [2002] presented a method to synthesize topologies for both the on-chip and off-chip communication systems. Ho and Pinkston [2003] presented a NoC design methodology based on well-behaved communications. Xu et al. [2005] proposed a general design, modeling, and analysis methodology for any type of NoC. REGULAY is a physical planning tool for homogenous multiprocessor networks Ye and De Micheli [2003].

### 1.2 Contributions and Paper Overview

Regular-topology NoC is inspired by general-purpose multicomputer networks. However, most SoC are heterogeneous and application-specific/domain-specific. In SoC, computation nodes (unites) often have very different communication requirements and geometry sizes. For example, an embedded processor requires much more communications bandwidth and area than an USB controller. Even in a homogeneous multiprocessor system, processors running different processes have very different communication requirements. With the help of application-specific mapping and tuning for a SoC application, regular-topology NoC can improve performance and reduce power consumption compared to on-chip buses.

There are several limitations to use regular-topology NoC for heterogeneous application/domain-specific SoC. First, communication locality is poorly supported in regular-topology NoC. Second, different communication requirements of function units are treated equally. Third, regular-topology NoC has abundant network resource, but the utilization is low. Fourth, different-sized function units do not fit well in floorplans preferred by regular network topologies.

We proposed application-specific networks-on-chip (ASNoC) to solve the limitations of regular-topology NoC. A methodology to automate the ASNoC design is also developed. ASNoC is a hierarchical NoC. In ASNoC, there is no fixed topology for all the applications. Instead, topology is decided by communication requirement, floorplan, and switch design. A conceptual ASNoC architecture is shown in Figure 1; an ASNoC example is shown in Figure 2. The lowest level

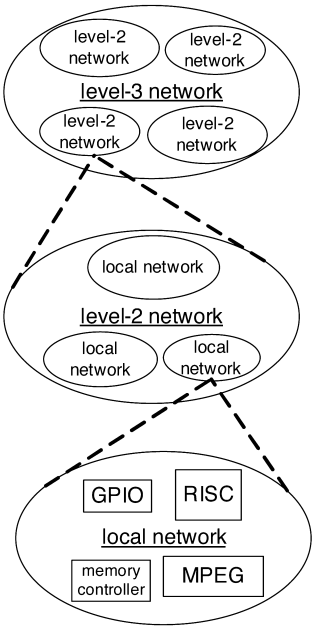


Fig. 1. Conceptual architecture of ASNoC.

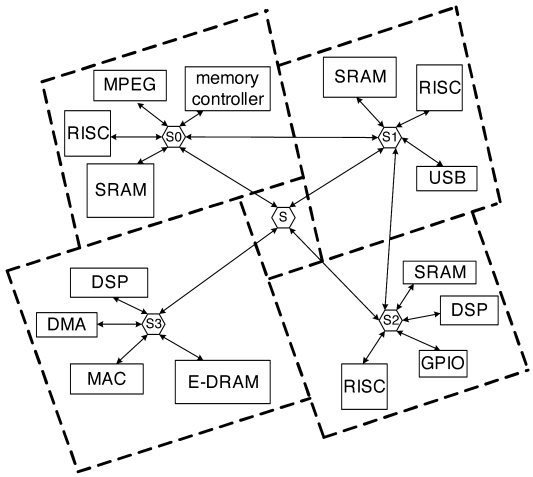


Fig. 2. ASNoC example.

of hierarchy is called local network (or level-1 network). A local network holds multiple closely cooperated function units. For example, the RISC, MPEG decoder, SRAM, and external memory controller are closely cooperated and in the same local network in Figure 2. Which function units are in a local network are decided by their cooperation relationship. The maximum number of function units in a local network is limited by chip floorplan. Multiple local networks are connected by a level-2 network. For example, there are four local networks

in Figure 2 and they are in a level-2 network. The maximum number of local networks in a level-2 network is limited by the switch design. If a SoC application has a large number of function units, such as several hundreds, more hierarchies will be needed.

By using network hierarchies, ASNoC tries to maximize communication locality. Irregular topology can treat different communication requirements of function units differently and increase network utilization. Besides hierarchy and irregular topology, floorplan estimation is used to consider the sizes of function units and reduce interconnection length, and protocols are selected to reduce network latency.

The design methodology for ASNoC can automatically generate an optimized hierarchical ASNoC for an application. It also generates a corresponding distributed shared memory, along with the ASNoC. The methodology uses either recorded or statistical communication traces for cycle-accurate performance analysis with different advantages. Standardized network component library and floorplans are used to estimate power and area. We adapt an industrial-grade network modeling and simulation environment, OPNET, which makes the methodology ready to use. The methodology can be easily integrated into current HW/SW codesign flow.

The methodology is used to generate ASNoC for a H.264 HDTV decoder SoC and Smart Camera SoC. ASNoC and 2D mesh NoC are compared in detail in performance, power, and area. The results show that the ASNoC provide substantial improvements in power, performance, and cost compared to 2D mesh NoC. In the H.264 HDTV decoder SoC, the ASNoC uses 39% less power, 59% less silicon area, 74% less metal area, 63% less switch capacity, and 69% less interconnection capacity to achieve 2X performance compared to 2D mesh NoC.

The next section gives an overview of the methodology. Corresponding steps in the methodology are detailed and illustrated in sections 2 through 8. Section 9 briefly describes the standardized network component library. Section 10 concludes this paper by showing the time spent on each step.

## 2. DESIGN METHODOLOGY OVERVIEW

In this section, we give an overview of the design methodology for ASNoC; more details will be illustrated by examples in the following sections. The inputs of the methodology are a behavior model and corresponding computation architecture (Figure 3). The output is a hierarchical ASNoC along with a distributed shared memory. The behavior model captures the functions of an application. The computation architecture describes computation nodes (units) used to implement those functions. The behavior model is partitioned and mapped to the computation architecture.

The first step of the methodology is the communication analysis. A mapped behavior model is used to obtain communication traffic patterns among computation nodes. The patterns include communication types, frequencies, timings, and information sizes. Those patterns are recorded by communication traces. Communication requirements can be extracted from the traces and used to

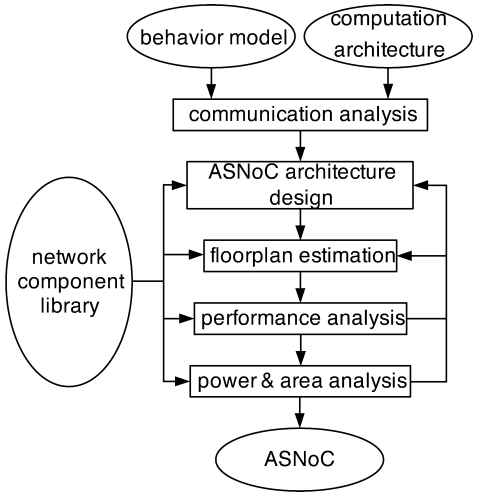


Fig. 3. ASNoC design methodology.

design ASNoC. Optionally, statistical traces can be modeled from the recorded traces or the behavior model. They can be used to accelerate performance analysis.

The second step is the ASNoC architecture design based on the communication requirements. The design uses a recursive method to generate a hierarchical ASNoC. A distributed shared memory is also generated along with the NoC. The NoC is composed of network components from a standardized component library. The library includes network interfaces, switches, and interconnections. Network components have adjustable features for different applications. Each network component has a cycle-accurate behavior model, a circuit model, and a layout model.

The third step estimates the chip floorplan to analyze and reduce the interconnection length. This length will be used to analyze performance, power, and area of a NoC design. Only the floorplan is estimated and placement and routing in each computation node is not involved.

The fourth step is the performance analysis. An industrial-grade network modeling and simulation environment, OPNET [OPNET], is adapted for this step. Cycle-accurate ASNoC models are built from the network component library. Simulations are based on the either recorded communication or statistical traces. The application performance results help to compare different design choices and refine the NoC designs. If no design meets the performance requirements or has too small performance margin, we have to go back to previous steps.

The last step is the power and area analysis. The network component library gives the power of each type of activity for each network component. OPNET records the numbers and types of activities of each network component. By summing the power of activities of all the network components, we obtain the power of the NoC. The network component library also gives the areas for each network component. Silicon and metal area of the NoC design are obtained by

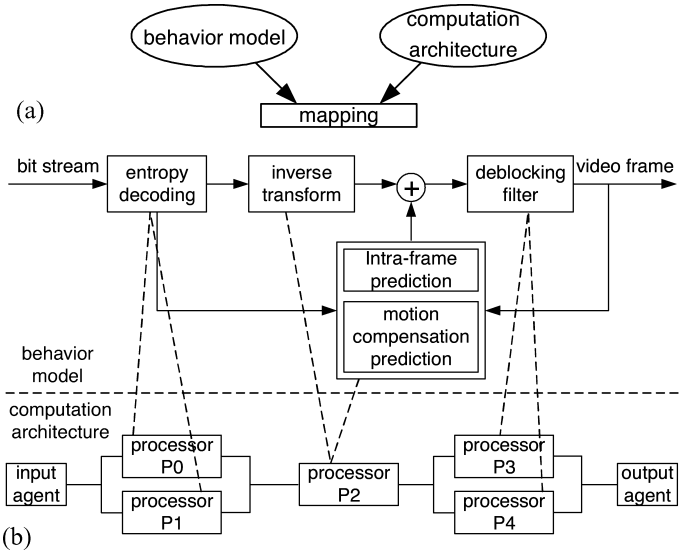


Fig. 4. Behavioral model and computation architecture (a) mapping, and (b) H.264 HDTV decoder.

summing the areas of all network components. If the power or the areas do not meet the requirements, we have to go back to previous steps.

The following sections will detail the design steps. Using the methodology, we designed ASNoC for two high-performance SoC. We will use one SoC design to illustrate each design step. Although the example we use is a homogeneous SoC, the methodology works for heterogeneous SoCs.

## 3. BEHAVIOR MODEL AND COMPUTATION ARCHITECTURE

Behavior model and computation architecture are the inputs of the design methodology for ASNoC. The behavior model captures the functions of an application using languages such as C or SystemC. The computation architecture describes a group of computation nodes (units), which are used to implement the functions and their connectivity. The behavior model is partitioned and mapped to the computation architecture (Figure 4a). The computation architecture could be heterogeneous or homogeneous. It is not necessary that all computation nodes are implemented at this time. In the computation architecture, a distributed memory is used at the beginning, and it will be converted to a distributed shared memory during the ASNoC design.

Using the methodology, we designed ASNoC for two high-performance SoCs. One is Smart Camera SoC. The smart Camera system [Wolf et al. 2002] is an embedded video processing application that can process 150 frames per second. The other is an H.264 HDTV decoder SoC, which is the latest video compression standard [Wiegand et al. 2003]. It is a candidate for HDTV broadcasting. The H.264 HDTV decoder is used to illustrate each step of the design methodology.

Joint Model [JVT] is used as the behavior model of our H.264 HDTV decoder (Figure 4b). The main profile and a progressive HDTV sequence with a resolution of 1920 X 1088 and 523 frames are used. In the entropy-decoding stage,

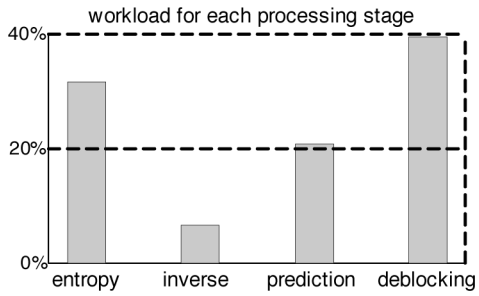


Fig. 5. Workload for each processing stage in H.264 HDTV decoder.

the input video stream is interpreted and various syntax elements are demultiplexed. Syntax elements of the video stream related with residual macroblocks are processed by the inverse transform stage. Syntax elements related with intraprediction macroblocks and motion-compensated prediction macroblocks are processed by the intraframe prediction and motion-compensation stage with reference to previous decoded frames or fields. The deblocking filter stage reduces artifacts introduced by the coding process at block boundaries. Figure 5 shows the workload of each processing stage in terms of the percentage of the whole system.

The decoder behavior model is partitioned and mapped to a computation architecture (Figure 4b) based on the workload of each processing stage. Input and output agents help to organize the input video stream and decoded video frames. The entropy-decoding stage is implemented by two processors, P0 and P1. The inverse transform stage and the intraframe prediction and motion-compensation prediction stage are implemented by processor P2. The deblocking filter stage is implemented by two processors, P3 and P4. We target a 130-nm technology to implement the H.264 HDTV decoder SoC. We use Plasma core [Opencore] for each processor.

## 4. COMMUNICATION ANALYSIS

The first step of the methodology is the communication analysis. A partitioned behavior model can be simulated to obtain communication traffic patterns among computation nodes. If high-level abstractions of computation nodes are available, more detailed traffic patterns can be obtained by simulating the mapped behavior model on the computation architecture. The communication patterns include communication types, frequencies, timings, and information sizes. Those patterns are recorded by communication traces (see Figure 6).

There is a communication trace for each computation node. A trace includes multiple entries and each entry records a network access. An entry records the interval between previous network and current network access, the source and destination of the access, the operation type, the size of transferred information, and the memory address if it is a memory access. The network access interval uses a unit of clock cycle. Compared to exact network access time, access interval is more suitable to describe the interactive communication behaviors of computation nodes. The change of one network access time because of blocking

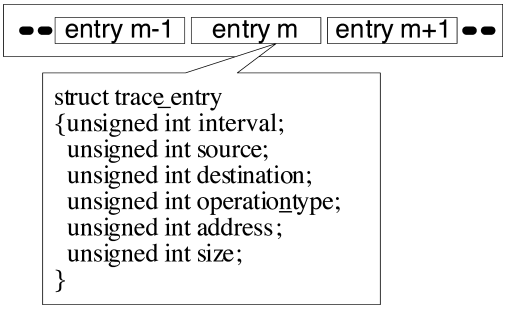


Fig. 6. Recorded communication trace.

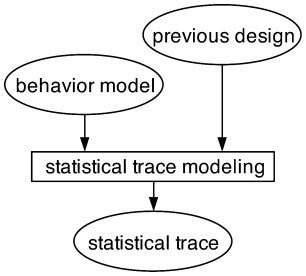


Fig. 7. Statistical trace modeling.

will affect the exact timing of all following network accesses, but this change has much less impact on the following network access intervals. Communication requirements can be extracted from the traces and used to design ASNoC. For example, the average communication traffic in terms of bits per second can be obtained from the traces. The traces are also used in performance analysis, where they control the communication behaviors of corresponding computation nodes.

Besides recorded communication traces, statistically modeled communication traces (statistical traces) are also very useful. Statistical traces can be modeled from previous design or behavior model (Figure 7). In trace modeling, probability distributions are selected to capture the communication behavior of each node. They can be used to accelerate performance analysis. In our study, Poisson distributions are selected. We obtain a 2–3X speedup in performance analysis and an error within 5% by using statistical traces instead of recorded traces. Statistical traces can help to accelerate the overall HW/SW codesign. Statistical traces from a similar design can be used to predict the communication behavior of a node even before behavioral simulations. Statistical traces also make it possible to analytically study NoCs.

## 5. ASNOC ARCHITECTURE DESIGN

The second step is the ASNoC architecture design. The design is based on a communication graph G extracted from the computation architecture and the traces (Figure 8). Communication graph $G = (V, E)$ is a weighted graph, $v \in V$ is a computation node, $e \in E$ is a connection between nodes, and the weight of

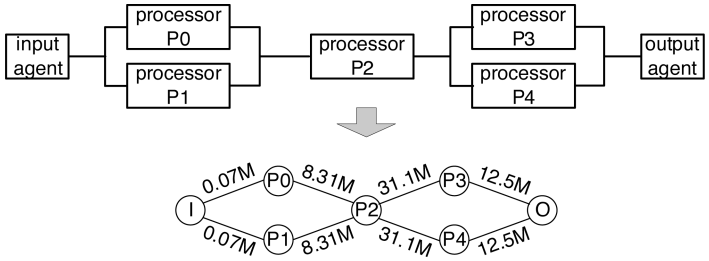


Fig. 8. Computation architecture and communication graph.

an edge $w(e)$ is the average communication traffic in a fixed period of time. A distributed memory is used in the computation architecture. A local memory stays with the corresponding computation node.

We use a recursive method to generate a hierarchical ASNoC. Generally, the number of hierarchies depends on the number of computation nodes. The lowest hierarchy level is local networks. Network hierarchies maximize communication locality using multiple network levels instead of a nonhierarchical network. The NoC architecture is generated by following algorithm.

ASNoC_architecture_generation ($G(V, E)$)
{
1 $K=\lceil |G|/M\text{-}2 \rceil$, $M$ is 6 for the first loop and the maximum number of ports of switch for other loops
2 Find $P_n=\{G_1, G_2, \ldots, G_K\}$ by $K$-way partitioning. The elements of $P_n$ are disjoint, $G_i$ $1 \le i \le K$ is an induced subgraph of $G$, the order of $G_i$ is $2,3,\ldots,M\text{-}2$, $\{V_1(G_1), V_2(G_2), \ldots, V_K(G_K)\} = V(G)$, and $n \in N$
3 For each $P_n$, find cost $C_n=\alpha^*\sum w(e_i)+\beta^*\sum w(e_j)$,
$e_i \in \{E_1(G_1), E_2(G_2), \ldots, E_K(G_K)\}$,
$e_j \notin \{E_1(G_1), E_2(G_2), \ldots, E_K(G_K)\}$ and $e_j \in E(G)$.
$\alpha$ is the communication cost in a local network in term of cycle,
$\beta$ is the cost between local networks
4 Select $P_n$ with the lowest cost
5 If $P_n$ has more than $M$ elements and can't be connected by a switch, then collapse the elements to vertices in $G$ to get $G'$ and go to step 1
6 Use a switch to connect nodes in subgraph $G_i$, connect the subgraphs using direct interconnections and switches in the network component library
7 Combine local memories of computation units in subgraph $G_i$ as a distributed shared memory
8 Return the ASNoC architecture
}

Because it is difficult to find the exact costs of ASNoC before implementation, we choose multiple candidates instead of just one. The order of a subgraph decides the number of computation nodes in a local network. The order of subgraph $G_i$ is limited by M, the maximum number of ports of a switch. ASNoC is composed of network components from a standardized component library (Figure 9). The library includes network interfaces, switches, and interconnections.

A distributed shared memory is also generated along with the NoC by combining local memories of all the nodes in a local network. Although memory can

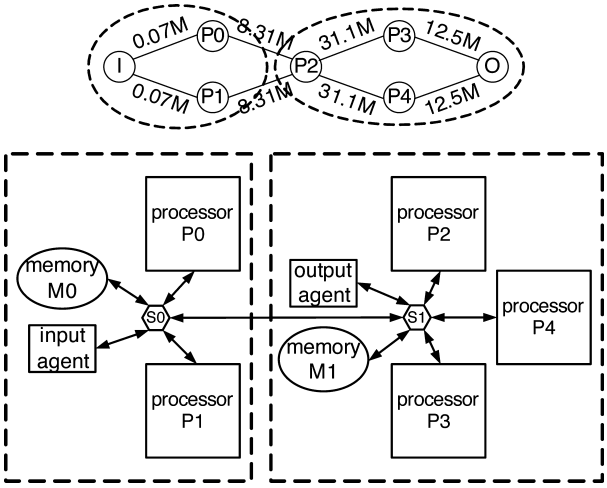


Fig. 9. Partition result and the ASNoC architecture.

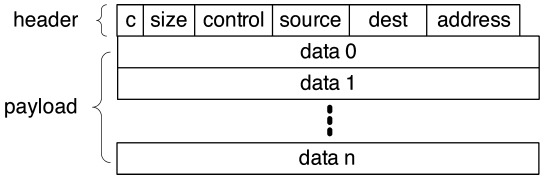


Fig. 10. Packet template.

be designed in the computation architecture, it is better designed along with NoC by considering the communication costs of memory accesses.

The packet format is customized based on a template (Figure 10). In the packet header, field sizes are chosen based on application's requirements. Small packet header reduces power consumption in a NoC. The c field is for checking the integrity of the packet. The size field shows the size of payload after the header. The control field holds the control information. The source field shows the origin of the packet. The destination field shows the destination of the packet. The address field gives the memory address for a memory access. Deterministic routing and cut-through flow control are used.

## 6. FLOORPLAN ESTIMATION

The third step estimates and designs chip floorplan. Chip floorplan gives the Manhattan length of each interconnection. The interconnection lengths decide the interconnection delays in term of clock cycles. In NoC, interconnections dominate the power and area. Interconnection lengths decide the power and area of NoC. In this step, only global and intermediate interconnections are designed and there is no placement and routing inside computation nodes involved. We estimate the slicing floorplan using a method similar to [Yuen and Young 2003]. The network hierarchies are used as the clustering constraints. A chip floorplan is shown in Figure 11.

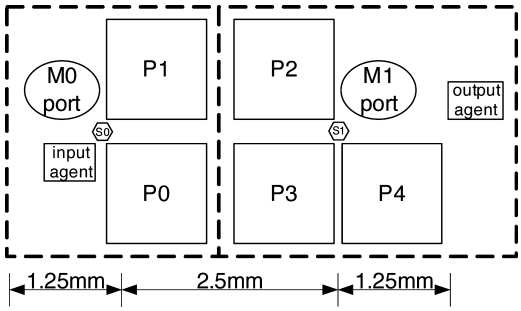


Fig. 11. Chip floorplan.

## 7. PERFORMANCE ANALYSIS

The fourth step is the performance analysis of the ASNoC. The application performance, instead of network performance, is analyzed in this step. Network performance is measured by network capacity, utilization, end-to-end delay, etc. Application performance has different metrics depending on application types. For example, the performance of a MPEG2 decoder is measured by the time to decode one frame or the number of frames decoded in 1 s. Compared to network performance, application performance requires simulating many more chip clock cycles. In the H.264 HDTV decoder, we simulated 5 billion chip clock cycles.

An industrial-grade network modeling and simulation environment, OPNET [OPNET], is adapted for this step. Compared to other network simulators, such as NS2, OPNET is faster and more stable. OPNET can model many common phenomena in communication systems. It is also a packet-based simulator, which is particularly useful for NoC designs. OPNET also has some limitations and some adaptations are required to use OPNET for simulating NoCs. First, it is developed for simulating computer networks and telecommunication networks 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. 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 make following adaptations in OPNET. In link models, (1) disable propagation delays pipeline stage and (2) disable 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-$\mu$s transmission delay.) In all node models, state transitions should be on clock edges. For the clock, (1) use 1 s to represent one clock cycle and (2) build a clock bus to synchronize the system.

In the network component library, each network component has a cycle accurate. ASNoC models are built on the network component library. OPNET uses C/C++ to model networks. Simulations are driven by either recorded communication or statistical traces. Communication traces are used to control the communication behavior of each computation node. Statistical traces can

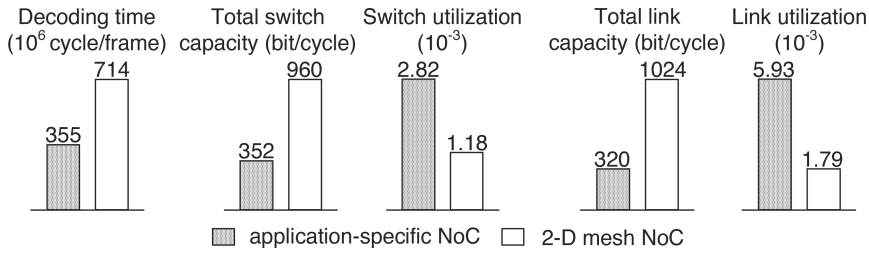


Fig. 12. Application and network performances of ASNoC and 2D mesh NoC.

accelerate the simulations. We observed a 2–3X speedup in our case studies. The application performance results help to compare different design choices. OPNET can give detailed performance of each network component and help to refine the NoC designs. If no design meets the performance requirements or has too small performance margin, we have to go back to previous steps.

Total switch capacity is defined as the maximum number of bits that can be processed by all the switches in 1 s. Total interconnection capacity is defined as the maximum number of bits that can be transferred by all the interconnections in a second. The switch utilization and interconnection utilization are obtained from OPNET simulations. The switch utilization is defined by

$$U_s = \frac{\sum B_s}{\sum C_s},$$

where $B_s$ is the number of bits switched by a switch in one second and $C_s$ is the capacity of a switch in 1 s. The switch utilization $U_s$ is the ratio of total number of bits switched by all the switches to total capacity of all switches. Similarly, the interconnection utilization is defined by

$$U_l = \frac{\sum B_l}{\sum T_l},$$

where $B_l$ is the number of bits transferred by an interconnection in 1 s. and, $T_l$ is the throughput of an interconnection. The interconnection utilization $U_l$ is the ratio of total number of bits transferred by all the interconnections to total throughput of all the interconnections

We compared the application performance of the ASNoC with a regular-topology NoC, 2D mesh NoC. 2D mesh NoC uses exactly the same group of computation nodes and memory as the ASNoC. In 2D mesh NoC, we optimized the mapping positions of computation nodes and memory for performance. In the H.264 HDTV decoder, the application performance is measured by the number of clock cycles needed to decode one frame. The results show that the ASNoC uses about 50% less decoding time than 2D mesh NoC, which is a 2X speedup (Figure 12). In term of the network performance, the ASNoC has only 37% total switch capacity and 31% total interconnection capacity of those of 2D mesh NoC, respectively. However, the ASNoC has 2.4X switch utilization and 3.3X interconnection utilization of those of 2D mesh NoC, respectively. It shows that the ASNoC can better use the limited network resources than 2D mesh NoC. We get the same conclusion in the other case study, Smart Camera SoC.

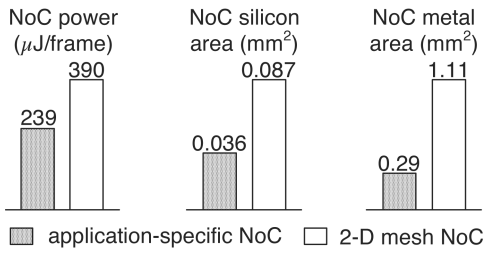


Fig. 13. Power and area of ASNoC and 2D mesh NoC.

## 8. POWER AND AREA ANALYSIS

The last step is power and area analysis. The network component library gives power of each type of activity for each network component $E_{ij}$. Power is obtained from SPICE simulation of network components. OPNET records the numbers and types of activities of each network component $A_{ij}$. By summing power of activities of all network components, we obtain power of NoC, $P = \sum \sum A_{ij} * E_{ij}$. The network component library also gives silicon area and metal area for each network component, $A_{Si}$ and $A_{Mi}$. Areas are based on the layout design of each network component. The total silicon and metal area of NoC are obtained by summing areas of all network components, $A_S = \sum A_{Si}$ and $A_M = \sum A_{Mi}$. If power or area does not meet requirements, we have to go back to previous steps.

We compared power and area of ASNoC with 2D mesh NoC in the H.264 HDTV decoder SoC design (Figure 13). Results show that ASNoC uses 39% less power, 59% less silicon area, and 74% less metal area than 2D mesh NoC. This is because ASNoC has less network resources—switches and interconnections—than 2D mesh NoC. We obtain the same conclusion in the study of Smart Camera SoC.

## 9. NETWORK COMPONENT LIBRARY

A standardized network component library is used throughout the design. The library includes switches, interconnections, and network interfaces. Each network component has three models: behavior, circuit, and layout model. The models give the cycle accurate behavior, power, and areas for each network components. We designed the circuit and layout of each component in the library. The network component was simulated in Cadence Spectre [Cadence].

Currently we use pipelined input/output buffered crossbar switches (Figure 14). Other types of switch can be added if required. Interconnections can use different technologies, such as low-swing, differential, and wave-pipelining [Xu and Wolf 2003]. We model a wire as a fine-grained lumped RLC network and consider the coupling capacitance. Since mutual inductances have significant effects at deep submicron processes, they are considered up to the third neighboring wires. All the design is based on a 130-nm aluminum process. More advanced processes can also be used. We use BISM3 model for the transistors and the typical wire dimensions from the Berkeley Predictive Technology Model [Cao et al. 2000].

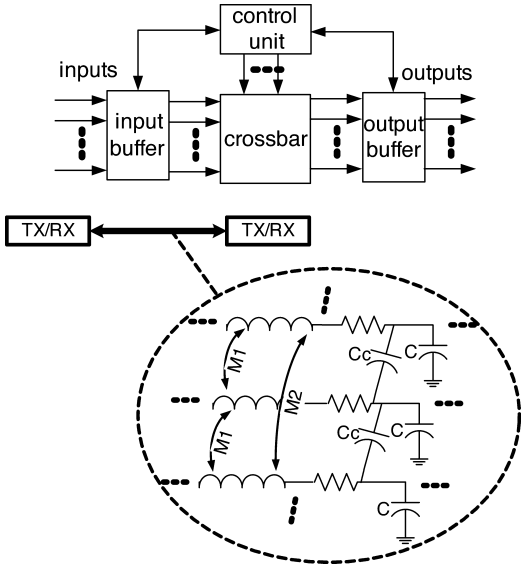


Fig. 14. Switch and interconnection design.

## 10. CONCLUSIONS

ASNoC and its design methodology are described in this paper. Our methodology can automatically generate all aspects of ASNoC for different applications. Some previous methodologies use standard topologies and vary a few parameters. Kumar et al. [2002] has a design methodology only for CLICHÉ NoC. Hu et al. [2003] considers the mapping in a 2D mesh network. Lahiri et al. [2004] can map an application to a given communication architecture. Murali and De Micheli [2004] selects a standard NoC from a library based on different requirements. Some other methodologies can synthesize a part of NoC. Daveau et al. [1997] can select protocols and generate interface. Siegmund and Muller [2003] can model and synthesize protocols. Jalabert et al. [2004] can generate NoCs from user-defined topologies. Our methodology can generate hierarchical topologies with distributed shared memories. Some other methodologies can generate nonhierarchical topologies. Pinto et al. [2002] can synthesize topologies for both the on-chip and off-chip communication systems. Ho and Pinkston [2003] can generate topologies for well-behaved communications.

In summary, compared to previous works, our methodology (1) can automatically generate optimized hierarchical ASNoC for different applications, (2) can generate a corresponding distributed shared memory along with an ASNoC, (3) can use both recorded and statistical communication traces for cycle-accurate performance analysis, (4) is based on the standardized network component library and floorplan to estimate power and area, (5) adapts an industrial-grade network modeling and simulation environment, OPNET, which makes the methodology ready to use, and (6) can be easily integrated into current HW/SW codesign flow.

Table I. Estimated Times Spent on Each Step (Person/Day)

| | H.264 HDTV decoder | Smart Camera |
|---|---|---|
| Communication analysis | 12 | 7 |
| ASNoC architecture design | 4 | 5 |
| Floorplan estimation | 1 | 1 |
| Performance, power and area analysis | 8 | 6 |
| Total | 25 | 19 |

Using this methodology, we generated ASNoC for a H.264 HDTV decoder SoC and Smart Camera SoC. We compared ASNoC with 2D mesh NoC in performance, power, and area in detail. The comparison results show that the ASNoC provide substantial improvements in power, performance, and cost compared to regular-topology NoC. In the H.264 HDTV decoder SoC, the ASNoC uses 39% less power, 59% less silicon area, 74% less metal area, 63% less switch capacity, and 69% less interconnection capacity to achieve 2X performance compared to 2D mesh NoC.

Table I lists the estimated time we spent on each design step. The whole design requires about 25 person day for the H.264 HDTV decoder. Communication and performance analysis involve large simulations and consume most of the design time.

## ACKNOWLEDGMENTS

Authors would like to thank the reviewers and editors for their helpful comments.

## REFERENCES

ADRIAHANTENAINA, A., CHARLERY, H., GREINER, A., MORTIEZ, L., AND ZEFERINO, C. A. 2003. SPIN: A scalable, packet switched, on-chip micro-network. *Design, Automation and Test in Europe Conference and Exhibition*.

BENINI, L. AND DE MICHELI, G. 2002. Networks on chip: A new paradigm for systems on chip design. *Design, Automation and Test in Europe Conference and Exhibition*.

CADENCE. www.cadence.com

CAO, Y., SATO, T., SYLVESTER, D., ORSHANSKY, M., AND HU, C. 2000. New paradigm of predictive MOSFET and interconnect modeling for early circuit design. *IEEE Custom Integrated Circuits Conference*. 201–204.

CONG, J. 2001. An Interconnect-centric design flow for nanometer technologies. *Proceedings of the IEEE 89*, 4, 505–528.

DALLY, W. AND TOWLES, B. 2001. Route packets, not wires: On-chip interconnection networks. *Design Automation Conference*.

DAVEAU, J.-M., MARCHIORO, G. F., BEN-ISMAIL, T., AND JERRAYA, A. A. 1997. Protocol selection and interface generation for HW-SW codesign *IEEE Transactions on Very Large Scale Integration Systems, 5*, 1.

FLYNN, D. 1997. AMBA: Enabling reusable on-chip designs. *IEEE Micro, 17*, 4.

FORSELL, M. 2002. A scalable high-performance computing solution for networks on chips. *IEEE Micro, 22*, 5.

GOOSSENS, K., DIELISSEN, J., MEERBERGEN, J., POPLAVKO, P., RADULESCU, A., RIJPKEMA, E., WATERLANDER, E., AND WIELAGE, P. 2003. Guaranteeing the quality of services in networks on chip. *Networks on Chip*, A. Jantsch and H. Tenhunen, eds. Kluwer Acad. Publ., Boston, MA.

HEMANI, A. AND JANTSCH, A. 2000. Network on chip: An architecture for billion transistor era", *IEEE NorChip Conference*.

HO, W. H. AND PINKSTON, T. M. 2003. A methodology for designing efficient on-chip interconnects on well-behaved communication patterns. *International Symposium on High-Performance Computer Architecture*.

HOFMANN, R. AND DRERUP, B. 2002. Next generation CoreConnect processor local bus architecture. *Annual IEEE International ASIC/SOC Conference*. 25–28.

HU, J. AND MARCULESCU, R. 2003. Energy-aware mapping for tile-based NoC architectures under performance constraints. *Asia and South Pacific Design Automation Conference*.

JALABERT, A., MURALI, S., BENINI, L., AND DE MICHELI, G. 2004. XpipesCompiler: A tool for instantiating application specific networks on chip. *Design, Automation and Test in Europe Conference and Exhibition*.

Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Joint Model Reference Software, iphome.hhi.de/suehring/tml/

KARIM, F., NGUYEN, A., AND DEY, S. 2002. An interconnect architecture for networking systems on chips. *IEEE Micro, 22*, 5.

KUHLMANN, M. AND SAPATNEKAR, S. S. 2001. Exact and Efficient Crosstalk Estimation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 20*, 7, 858–866.

KUMAR, S., JANTSCH, A., SOININEN, JP., FORSELL, M., MILLBERG, M., ÖBERG, J., TIENSYRJÄ, K., AND HEMANI, A. 2002. A network on chip architecture and design methodology. *IEEE Computer Society Annual Symposium on VLSI*.

LAHIRI, K., RAGHUNATHAN, A., AND DEY, S. 2004. Design space exploration for optimizing on-chip communication architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 23*, 6.

LIANG, J., SWAMINATHAN, S., AND TESSIER, R. 2000. aSoC: A scalable, single-chip communications architecture. *International Conference on Parallel Architectures and Compilation Techniques*.

MILLBERG, M., NILSSON, E., THID, R., KUMAR, S., AND JANTSCH, A. 2004. The Nostrum backbone-a communication protocol stack for Networks on Chip. *International Conference on VLSI Design*.

MURALI, S. AND DE MICHELI, G. 2004. "SUNMAP: A tool for automatic topology selection and generation for NoCs", Design Automation Conference.

Opencore. www.opencores.org

Opnet. www.opnet.com

PINTO, A., CARLONI, L. R., AND SANGIOVANNI-VINCENTELLI, A. L. 2002. Constraint-driven communication synthesis.*Design Automation Conference*.

RYU, K. K., SHIN, E., AND MOONEY, V. J. 2001. A comparison of five different multiprocessor SoC bus architectures. *Euromicro Symposium on Digital Systems*, Design.

SEMATECH. 2004. *International Technology Roadmap for Semiconductors*

SGROI, M., SHEETS, M., MIHAL, A., KEUTZER, K., MALIK, S., RABAEY, J., AND SANGIOVANNI-VINCENTELLI, A. 2001. Addressing the system-on-a-chip interconnect woes through communication-based design. *Design Automation Conference*.

SIEGMUND, R. AND MULLER, D. 2003. Efficient modeling and synthesis of on-chip communication protocols for network-on-chip design.*IEEE International Symposium on Circuits and Systems*.

SIGUENZA-TORTOSA, D. AND NURMI, J. 2002. Proteo: A New Approach to Network-on-Chip. *International Conference on Communication Systems and Networks*.

SILICORE. www.silicore.net

TAYLOR, M. B., KIM, J., MILLER, J., WENTZLAFF, D., GHODRAT, F., GREENWALD, B., HOFFMAN, H., JOHNSON, P., LEE, JAE-WOOK, LEE, W., MA, A., SARAF, A., SENESKI, M., SHNIDMAN, N., STRUMPEN, V., FRANK, M., AMARASINGHE, S., AND AGARWAL, A. 2002. The Raw microprocessor: A computational fabric for software circuits and general-purpose programs. *IEEE Micro, 22*, 2.

WANG, H., ZHU, X., PEH, L., AND MALIK, S. 2002. Orion: A power-performance simulator for interconnection networks. *IEEE MICRO*.

WIEGAND, T., SULLIVAN, G.J., BJONTEGAARD, G., AND LUTHRA, A. 2003. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 560–576.

WINGARD, D. 2001. MicroNetwork-based integration for SOCs. *Design Automation Conference*.

WOLF, W. 2001. *Computers as Components: Principles of Embedded Computing System Design*. Morgan Kaufman Publishers.

WOLF, W., OZER, B., AND LV, T. 2002. Smart cameras for embedded systems. *IEEE Computer, 35*, 9, 48–53.

XU, J. AND WOLF, W. 2003. A Wave-Pipelined On-chip Interconnect Structure for Networks-on-Chips. *Hot Interconnects*.

XU, J., WOLF, W., HENKEL, J., AND CHAKRADHAR, S. 2005. A methodology for design, modeling, and analysis of Networks-on-Chip. *IEEE International Symposium on Circuits and Systems*.

YE, T. AND DE MICHELI, G. 2003. Physical planning for on-chip multiprocessor networks and switch fabrics. *IEEE International Conference on Application-Specific Systems, Architectures, and Processors*

YUEN, W. S. AND YOUNG, E. F. Y. 2003. Slicing floorplan with clustering constraint. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 22*, 5.

ZEFERINO, C. A. AND SUSIN, A. A. 2003. SoCIN: A parametric and scalable network-on-chip. *Symposium on Integrated Circuits and Systems Design*.

ZHU, X., QIN, W., AND MALIK, S. 2004. Modeling operation and microarchitecture concurrency for communication architectures with application to retargetable simulation. *International Conference on Hardware/Software Codesign and System Synthesis*.

Received February 2005; revised June 2005; accepted September 2005