# PPNOCS: Performance and Power Network on Chip Simulator based on SystemC

**El Sayed M. Saad[1], Sameh A. Salem[1], Medhat H. Awadalla[1,2], and Ahmed M. Mostafa[1]**

**[1] Communication, Electronics and Computers Department, Faculty of Engineering, Helwan University, Helwan, Egypt**

**[2]Electrical and Computer Engineering Department, SQU University, Oman**

**Abstract**

As technology moves towards multi-core system-on-chips (SoCs), networks-on-chip (NoCs) are emerging as the scalable fabric for interconnecting the cores. Network-on-Chip architectures have a wide variety of parameters that can be adapted to the designer's requirements. This paper proposes a performance and power network on chip simulator (PPNOCS) based on SystemC to explore the impact of various architectural level parameters of the on-chip interconnection network elements on its performance and power. PPNOCS supports an arbitrary size of mesh and torus topology, adopts five classic routing algorithms and seven synthetic traffic patterns. Developers also can develop and verify their own network design by modifying the corresponding modules. Experiments of using this simulator are carried out to study the power, latency and throughput of a 4x4 multi-core mesh network topology. Results show that PPNOCS provides a fast and convenient platform for researching and verification of NoC architectures and routing algorithms.

***Keywords:*** *Network-on-Chip, Performance, Power, Simulation, SystemC.*

## 1. Introduction

Networks-on-chip [1] are critical elements of modern system-on-chip as well as multi-core designs. They consist of routers, links, and well-defined network interfaces. Packet-switched interconnection networks [2] facilitate communication between cores by routing packets between them. The structured and localized wiring of such a NoC design simplifies timing convergence and enables robust design that scales well with device performance.

One major difficulty that faces NoC architects is to select a communication network that suits a specific application or a range of specific applications with the constraints of cost, power and performance. Design decisions are typically made on the basis of simulation before resorting to emulation or implementation since it is cheap and flexible. To make a right decision on the network architecture, a simulation tool should enable to faster explore the architectural design space and assess design quality regarding performance, cost, and power.

SystemC [3] and Transaction Level Modeling (TLM) [4] have become quite popular and have found a relatively wide range of applications both in academia and industry [5]. SystemC is an extension of C++, in the form of a hardware-oriented library of C++ classes [6]. TLM is a library of functions built on the top of SystemC. In the TLM terminology, a transaction represents the information being exchanged between the different system modules. TLM is particularly interested in separating the computational component from the communication component. For this purpose, TLM provides constructs to efficiently model the inter-module communication such as channels, interfaces and ports, which are objects provided by SystemC.

This paper presents a performance and power network on chip simulator (PPNOCS) based on SystemC, to explore the impact of various architectural level parameters of the on-chip interconnection network elements on its performance and power. A general modularized NoC node structure is first realized under SystemC, and then connected to form the network. Users also can develop their own network topology and routing algorithm by modifying the corresponding modules. Then they can verify their design by loading different network traffic patterns to run the simulation.

The paper is organized as follows: Section 2 provides a brief overview of related work. The simulation platform is described in Section 3. Experimental results are discussed in Section 4. Finally, Section 5 concludes the paper.

## 2. Related Work

With the emergence of the NoC concept, researchers have realized the need to evaluate NoC systems. This has led to the use of existing network simulators, which have been adapted for on-chip communication networks [7]. Xu et al. employed the OPNET network simulator for simulation of on-chip network systems [8]. Such an approach leverages the already existing tool, which has had time to mature.

However, on-chip communication is different than traditional networks and parallel computer communication networks. NoC simulation environment must accurately reflect on-chip behaviors. Nostrum is another attempt of NoC simulation developed at KTH, Stockholm and it offers a packet switched communication platform based on the traditional OSI model of computer networks [9]. Initially, mesh topology is selected to prove the concept of Nostrum simulator. Recently, attempts have also been made to extend Nostrum to support both regular and irregular NoC topologies [10].

Many simulation tools have been developed to research the design of router architectures [11, 12] and NoC topologies [13] with varying area/performance [14] trade-offs for general purpose SoCs. Kogel et. al. [15] presents a modular exploration framework to capture performance of point-to-point, shared bus and crossbar topologies. The impact of varying topologies, link and router parameters on the overall throughput, area and power consumption of SoCs using relevant traffic models is discussed in [16]. Orion [17] is a power-performance interconnection network simulator that is capable of providing power and performance statistics. Orion model estimates power consumed by router elements (crossbars, FIFOs and arbiters) by calculating switching capacitances of individual circuit elements. Most of these tools do not allow for exploration of the various link level options of wire width, pitch, serialization, repeater sizing, pipelining, supply voltage and operating frequency.

In [18], Madsen et al. presented a NoC model which, together with a multiprocessor real-time operating system (RTOS) are used to model and analyze the behavior of a complex system that has a real-time application running on it. Mesh and torus are implemented in their design. Nurmi et al. [19] proposed a simulation environment by creating a library of pre-designed communication blocks that can be selected from a component library and configured by automated tools. From simulation point of view, these simulation tools are flexible to perform NoC design exploration. However, they are limited in topologies, and performance metrics [20].

In this paper, the proposed simulation platform is built from the ground up for Network-on-Chip simulation. The platform is built in SystemC, and takes advantage of the low-level modeling available in SystemC communication primitives, while leveraging the efficiency of C++ to achieve a balance between accuracy and performance. The main contributions of our simulation platform include the following:

- Explore the impact of various architectural level parameters of the on-chip interconnection network elements on its performance and power.
- Owing to the general NoC node structure and modularization modeling, users can extend the simulator with their own routing algorithm and network topology.
- PPNOCS provides a fast and convenient platform for researching and verification of various Network-on-Chip architectural designs.

## 3. Simulation Platform

A wormhole-router provides the necessary fine-grained flow control in terms of buffer and latency requirements, while the addition of virtual-channels aids in boosting performance and circumventing message-dependent deadlock [21]. Furthermore, Quality-of-Service (QoS) enhancements can be achieved by prioritizing the allocation of virtual-channels and switch bandwidth. For these reasons, PPNOCS implements the generic virtual-channel router shown in figure 1.

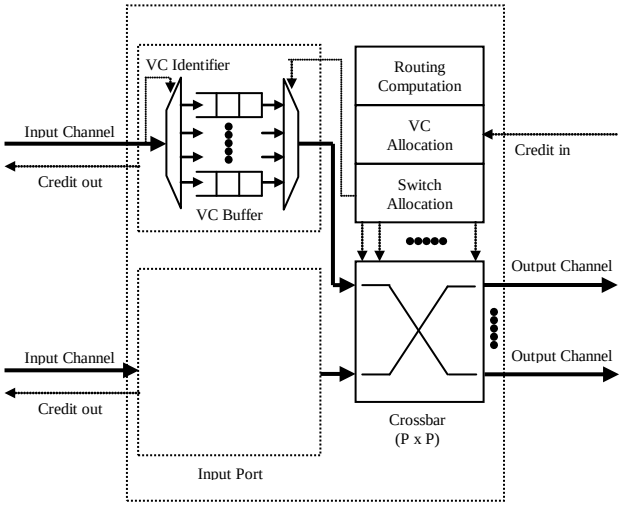



Fig. 1 Virtual-Channel Router

The router has P input ports and P output ports, supporting N virtual-channels (VCs) per port. Virtual-channel flow control exploits an array of buffers at each input port. By allocating different packets to each of these buffers, flits from multiple packets may be sent in an interleaved manner over a single physical channel. This improves both throughput and latency by allowing blocked packets to be bypassed.

### 3.1 PPNOCS Node Structure

To enable easy extensibility of the simulation platform, PPNOCS develop a modularized architecture for the generic router. Figure 2 shows the developed PPNOCS node structure.

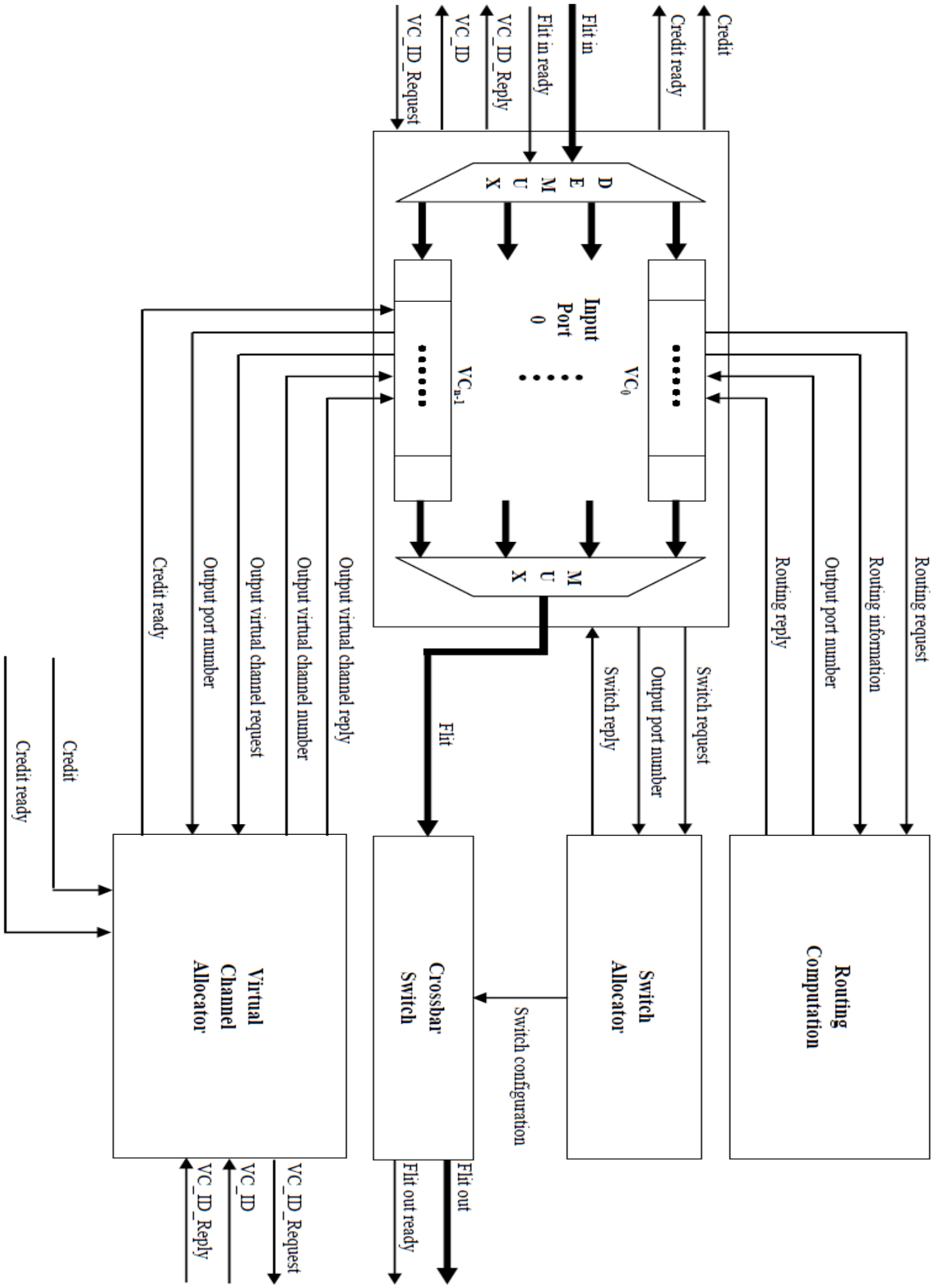



Fig. 2 PPNOCS node structure

In PPNOCS, a packet is divided into flow control digits or flits. A flit is the basic unit of bandwidth and storage allocation used by most flow control methods. The position of a flit in a packet determines whether it is a head flit, body flit, or tail flit. A head flit is the first flit of a packet and carries the packet's routing information. A head flit is followed by zero or more body flits and a tail flit. In a very short packet, there may be no body flits. In the following, a brief description of each module in the PPNOCS node structure is given.

### 3.1.1 Input Port Module

Input port module consists of a set of virtual channel modules. Each virtual channel consists of a FIFO buffer. The user can determine the number of virtual channels and the depth of each buffer in terms of flits. Any flit arrives at the input port contains a virtual channel identifier (VC_ID) which determines in which virtual channel buffer it will be stored.
All of the flow control mechanisms that use buffering need to know the availability of buffers at the downstream nodes. Then the upstream nodes will determine when a buffer is available to hold the next flit to be transmitted. This type of buffer management provides backpressure by informing the upstream nodes when they should stop flit transmission because all of the downstream flit buffers are full. Three types of low-level flow control mechanisms are in common use today to provide such backpressure: credit-based, on/off, and ack/nack [22]. PPNOCS implements the credit-based flow control mechanism. With credit-based flow control, the upstream router keeps a count of the number of free flit buffers in each virtual channel downstream. Then, each time the upstream router forwards a flit, thus consuming a downstream buffer, it decrements the appropriate count. If the count reaches zero, all of the downstream buffers are full and no further flits can be forwarded until a buffer becomes available. Once the downstream router forwards a flit and frees the associated buffer, it sends a credit to the upstream router for incrementing the buffer count.

### 3.1.2 Routing Computation Module

When a head flit of a new packet arrives at the input port, a routing request along with the routing information is sent to the routing computation module. According to the routing algorithm a set of valid output ports is produced and sent back to the input port module. The number of outputs produced by the routing computation module will depend on the routing algorithm. If more than one output produced, the selection function randomly select one of these outputs. PPNOCS implements five routing algorithms:

**XY Routing Algorithm:** XY routing is a dimension ordered routing which routes packets first in x- or horizontal direction to the correct column and then in y- or vertical direction to the receiver. XY routing suits well on a network using mesh or torus topology. Addresses of the routers are their xy-coordinates. XY routing never runs into deadlock or livelock [23]. Figure 3 shows an example of XY routing.

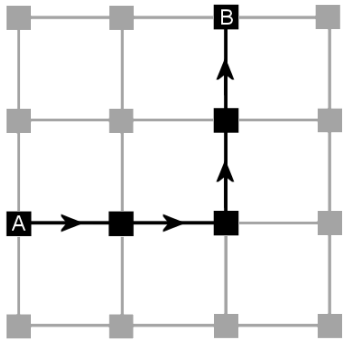



Fig. 3 XY routing from router A to router B

**West-First Routing Algorithm:** A west-first routing algorithm prevents all turns to west. So the packets going to west must be first transmitted as far to west as necessary. Figure 4 shows the allowed turns in the west-first routing.

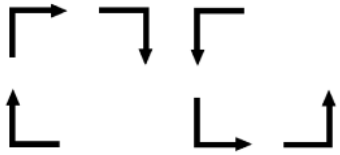


Fig. 4 Allowed turns in west-first routing

**North-Last Routing Algorithm:** Turns away from north are not possible in a north-last routing algorithm. Thus the packets which need to be routed to north must be transferred there at last. Figure 5 shows the allowed turns in the north-last routing.

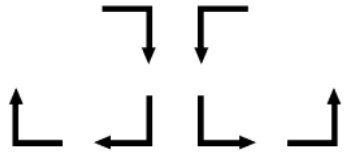


Fig. 5 Allowed turns in north-last routing

**Negative-First Routing Algorithm:** Negative-first routing algorithm allows all other turns except turns from positive direction to negative direction. Packet routings to negative directions must be done before anything else [24]. Figure 6 shows the allowed turns in the negative-first routing.

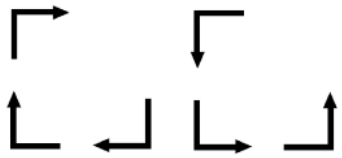


Fig. 6 Allowed turns in negative-first routing

**Fully Adaptive Routing Algorithm:** Fully adaptive routing algorithm uses always a route which is not congested. The algorithm does not care although the route is not the shortest path between sender and receiver [22].

### 3.1.3 Virtual Channel Allocation Module

After selecting a specific output port for the packet, the input port module sends a virtual channel request along with the output port number to the virtual channel allocation module. Then, the virtual channel allocator module sends a virtual channel request to the specified input port module in the downstream router. After receiving the VC_ID from the downstream router, the virtual channel allocator module sends it back to the input port module.

### 3.1.4 Switch Allocation Module

Each flit waiting in a virtual channel buffer and has available space in the downstream buffer can send a switch request to the switch allocator module. PPNOCS implements 5 × 5 (5-input × 5-output) input-first separable allocator. In an input first separable allocator, arbitration is first performed to select a single request at each input port. Then, the outputs of these input arbiters are input to a set of output arbiters to select a single request for each output port. The result is a legal matching, since there is at most one grant asserted for each input and for each output. The switch allocator module sends a switch reply for each input port module wins in the arbitration. If multiple VCs in the input port have been requested the same output port which is granted by the switch allocator, then they will be serviced in a Round Robin (RR) fashion. Upon granting the switch allocation requests, the switch allocation module sends a switch configuration signals to the crossbar switch module.

### 3.1.5 Crossbar Switch Module

Flits that have been granted passage on the crossbar are passed to the appropriate output ports.

### 3.2 Traffic Patterns

Application-driven workloads can be too cumbersome to develop and control [22]. This motivates the inclusion of synthetic workloads, which capture the salient aspects of the application-driven workloads, but can also be more easily designed and manipulated. Synthetic workloads are divided into three independent aspects: traffic patterns, injection processes, and packet length.

Traffic pattern is the spatial distribution of messages in interconnection networks. This message distribution is represented with a traffic matrix, where each matrix element $\lambda_{s,d}$ gives the fraction of traffic sent from node s destined to node d. Table 1 lists some common static traffic patterns used to evaluate interconnection networks [22].

Table 1: Network traffic patterns

| Name | Pattern |
|---|---|
| **Random** | $\lambda_{sd} = 1/N$ |
| **Bit complement** | $d_i = \neg s_i$ |
| **Bit reverse** | $d_i = s_{b-i-1}$ |
| **Bit Rotation** | $d_i = s_{i+1 \bmod b}$ |
| **Shuffle** | $d_i = s_{i-1 \bmod b}$ |
| **Transpose** | $d_i = s_{i+b/2 \bmod b}$ |

PPNOCS supports seven synthetic traffic patterns (Uniform Random, Hotspot, Bit Reversal, Bit Complement, Bit Rotation, Shuffle, and Matrix Transpose).

**Random traffic**: In which each source is equally likely to send to each destination is the most commonly used traffic pattern in network evaluation. Random traffic is very benign because, by making the traffic uniformly distributed, it balances load even for topologies and routing algorithms that normally have very poor load balance. Some very bad topologies and routing algorithms look very good when evaluated only with random traffic [22].

**Hotspot Traffic:** In hotspot traffic pattern, there's a particular node that will receive more traffic than other nodes. In PPNOCS, the hotspot is specified along with the percentage of the traffic dedicated to it.

**Bit Reversal, Bit Complement, Bit Rotation, Shuffle, and Matrix Transpose:** These are called Bit permutation patterns, in which each bit $d_i$ of the b-bit destination address is a function of one bit of the source address, $s_j$ where j is a function of i. permutation traffic patterns stresses the network topology or the routing algorithm because each source s sends all of its traffic to a single destination.

Injection process determines the average number of packets it injects per cycle (injection rate). The most common injection processes used in network simulations is the Bernoulli process [22]. For a Bernoulli process with rate r, the injection process A is a random variable with the probability of injection a packet equal to the process rate,

P(A = 1) = r. PPNOCS implements the Bernoulli process and the user can specify the packet injection rate before running the simulation. Also, the packet length in terms of flits can be specified.

### 3.3 Architecture Parameters

This section summarizes the different architectural parameters that can be configured before running the simulation. Table 2 shows a brief description for each parameter that can be specified by PPNOCS.

Table 2: Network Architectural Parameters

| Parameter Name | Description |
|---|---|
| Topology | 2D Mesh or 2D Torus |
| DimX | Number of columns |
| DimY | Number of rows |
| NUM_INPUTS | Number of input and output ports |
| VC_NUM | Number of virtual channels in each input port |
| VC_BUFFER_SIZE | Number of buffers for each virtual channel |
| PACKET_INJECTION_RATE | Injection rate (< =1) |
| TRAFFIC_DISTRIBUTION | Uniform Random, Hotspot, Bit Reversal, Bit Complement, Bit Rotation, Shuffle, and Matrix Transpose |
| ROUTING_ALGORITHM | XY, West-First, North-Last, Negative-First, and Fully Adaptive |
| PACKET_SIZE | Number of flits in the packet |
| WARM_UP_TIME | The number of clock cycles after which the simulator starts to collect statistics |
| SIMULATION_TIME | The number of clock cycles that have to be simulated. |

### 3.3 Performance Metrics

A standard set of performance metrics can be used to compare and contrast different NoC architectures. The performance metrics evaluated by PPNOCS include throughput and packet latency [22].

#### 3.3.1 Throughput

Throughput is the rate at which packets are delivered by the network for a particular traffic pattern. It is measured by counting the packets that arrive at destinations over a time interval for each flow (source-destination pair) in the traffic pattern and computing from these flow rates the fraction of the traffic pattern delivered [22]. Throughput, or accepted traffic, is to be contrasted with demand, or offered traffic, which is the rate at which packets are generated by the Intellectual Property (IP) block. Throughput can be defined as follows [16]:

$$\frac{\text{Total number of packets received at their destinations}}{\text{(Number of IP blocks) x (Total Time in Cycles)}}$$

#### 3.3.2 Packet latency

Transport latency is defined as the time (in clock cycles) that elapses from between the occurrence of head flit injection into the network at the source node and the occurrence of the tail flit reception at the destination node [25].
In order to reach the destination node from some starting source node, flits must travel through a path consisting of a set of routers and interconnects [19]. Depending on the source/destination pair and the routing algorithm, each packet may have a different latency [19]. Therefore, for a given packet $P_i$, the latency $L_i$ is defined as:

$L_i$ = receiving time (tail flit of $P_i$) − sending time (head flit of $P_i$)

Let F be the total number of packets reaching their destination IPs and let $L_i$ be the latency of packet $P_i$, where i ranges from 1 to F. The average packet latency, $L_{avg}$, is then calculated according to the following equation [19]:

$$L_{avg} = \frac{\sum_{i=1}^{F} L_i}{F}$$

### 3.4 Power Model

In PPNOCS, a power model at flit level is proposed depending on the results obtained from the Intel 80-core teraflop chip [26]. To explain the proposed model, consider a source IP injects a head flit into the write port of the input port module. The virtual channel module writes the flit into the tail of the FIFO buffer and emits a buffer write event, which triggers the buffer power model to compute buffer write power $P_{write}$. After the routing module determines the output port to which the head flit will be sent, a request is sent to the switch allocator module for the desired output port. The allocator module performs the required arbitration and generates an arbitration event, which signals the arbiter power model to compute arbitration power $P_{arbiter}$. Assuming the request is granted, the arbitration result is sent to the config port of the crossbar module. A grant signal is also sent to the grant port of the virtual channel module, leading to the read port of the buffer module activated. The flit is then read, emitting a buffer read event, which causes the buffer power model to compute buffer read power $P_{read}$. The flit next

traverses the crossbar, from input port to the output port. The crossbar module emits a crossbar traversal event and the crossbar power model computes traversal energy $P_{crossbar}$. Finally, the flit leaves the router and traverses the link. The link module emits a link traversal event, which calls the link power model to compute link traversal power $P_{link}$.

The total power consumed by this head flit at this node and its outgoing link is as follows:

$$P_{flit} = P_{write} + P_{arbiter} + P_{read} + P_{crossbar} + P_{link} \quad (1)$$

Let:

$$P_{buffer} = P_{write} + P_{read} \quad (2)$$

Then by substituting Eq. (2) in Eq. (1):

$$P_{flit} = P_{buffer} + P_{arbiter} + P_{crossbar} + P_{link} \quad (3)$$

The Intel 80-core teraflop chip recently introduced by Intel [26] is a good example of an aggressive NoC prototyping effort. The Teraflops Processor architecture contains 80 tiles arranged in a 8 x 10 2D array and connected by a mesh network that is designed to operate at 5 GHz. A tile consists of a processing engine connected to a five-port router, which forwards packets between tiles. The communication power is significant at 28% of each processing tile's total power. As shown in Figure 7, clocking power, 33%, is the largest component of router power, with the FIFO buffers the second largest component at 22%. Power due to physical links, crossbar switch, and arbiter come next at 17%, 15% and 7%, respectively.

According to the proposed power model, it is required to compare and contrast different NoC architectures in terms of power consumed in buffers, links, crossbar, and arbiter. So, in PPNOCS, it is considered that a unit power ($U_p$) is consumed by a flit to traverse from the input port to the output port of the router and leave through the outgoing link. $U_p$ is then divided between buffering, arbitration, crossbar traversal, and link traversal according to the power ratios presented by the Intel 80-core teraflop chip and shown in Figure 7.

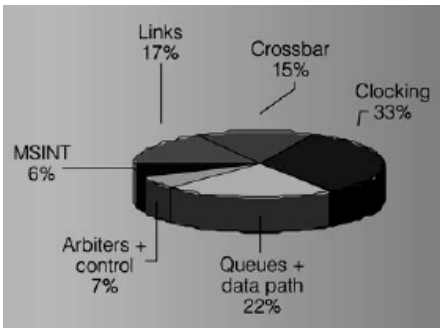



Fig. 7 Router power breakdown at 4 GHz, 1.2 V, and 110 C

## 4. Experimental Results

There are three potential ways of using PPNOCS for rapid exploration of network microarchitectures.

- The architect may wish to explore the impact of two application traffic patterns on specific network microarchitecture.
- The architect may wish to trade-off two configurations of microarchitecture, exploring their effect on network power and performance. This involves setting the network architectural parameters for the two configurations. Given a specific traffic pattern of the targeted application, the architect can feed the traffic pattern and configurations into two different instances of PPNOCS, and obtain their power and performance numbers.
- The architect may develop new network microarchitecture and wish to explore its impact on power and performance, evaluating it against a base microarchitecture. Owing to the general NoC node structure and modularization of PPNOCS, the architect can extend the simulator easily with their own microarchitectures.

In the experiments, a 16-node network organized as a 4 x 4 mesh is implemented, as shown in Figure 8. Each router has five physical bidirectional ports (north, south, east, west, and injection/ejection). Each simulation is run for a warm-up phase of 1000 cycles and simulation phase of 10000 cycles.

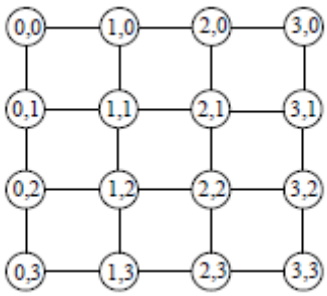



Fig. 8 A 4×4 2D mesh network

### 4.1 Exploring the effect of different traffic patterns

In this experiment, a 4×4 mesh NoC with XY routing is implemented, and 6 different traffic patterns to run the simulation is loaded. The packet length is two flits and each input port has 4 virtual channels with FIFO buffer

depth equal to 4 packets. The result of this simulation is shown in Figure 9.

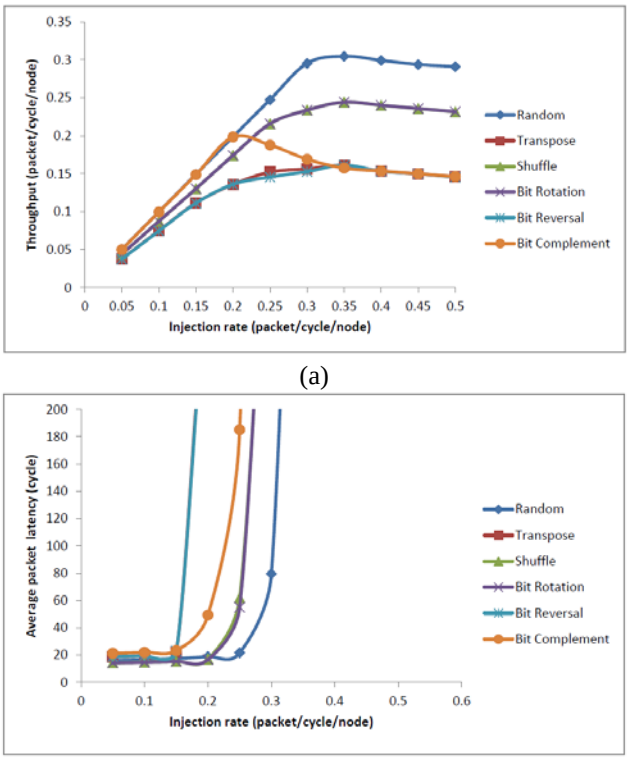



Fig. 9 Results of different traffic patterns

As shown in figure 9, Random traffic pattern gives better throughput and average packet latency according to its uniform and balanced distribution of load.

### 4.2 Exploring the effect of different configurations

This set of simulations is based on 4X4 mesh with XY routing, packet size equal two flits and under uniform random traffic. In this experiment, four different router configurations are simulated and compared:-

- Wormhole router with 64-flit input buffer per port (WH64).
- Virtual-channel (VC) router with 2 VCs per port and 16-flit input buffer per VC (VC_2_16).
- Virtual-channel router with 4 VCs per port and 4-flit input buffer per VC (VC_4_4).
- Virtual-channel router with 8 VCs per port and 8-flit input buffer per VC (VC8_8).

Figure 10 shows results obtained from simulating these routers in PPNOCS. Figure 5(a) shows VC_8_8 out-performing WH64, despite having the same total buffer size per input port, saturating at a higher packet injection rate of 0.35 packets/cycle/node. However, this performance improvement is achieved at the expense of higher power consumption, as indicated by Figure 11.

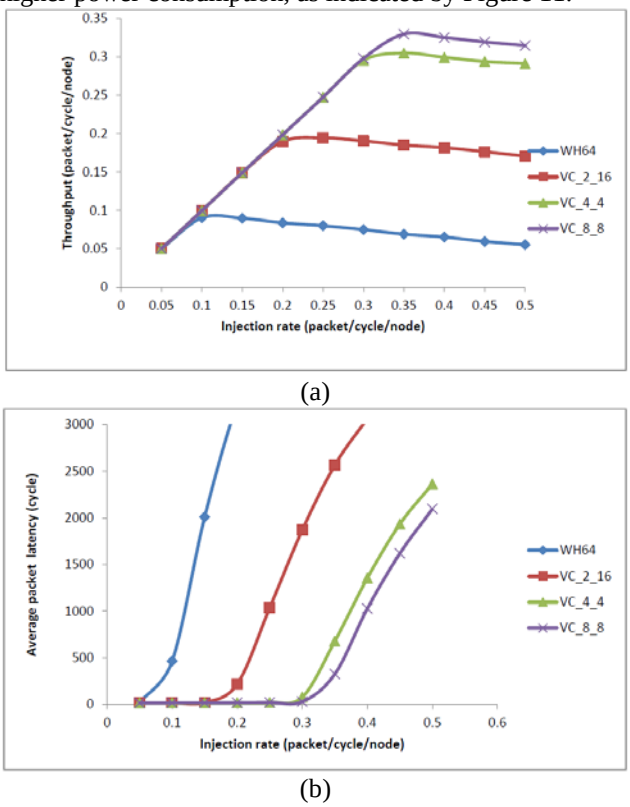



Fig. 10 Results of different configurations

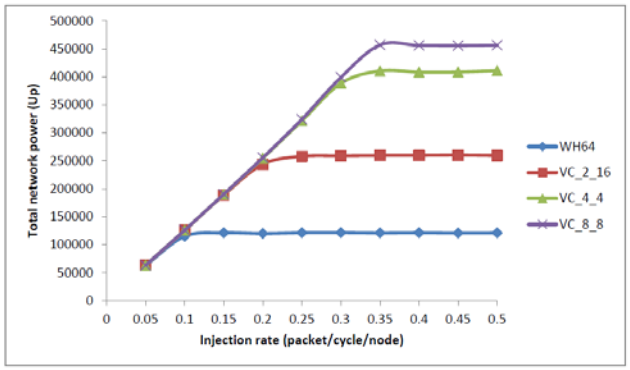



Fig. 11 Results of different configurations on power consumption

Beyond packet injection rate of 0.1 packets/cycle/node, VC_8_8 starts to consume more power than WH64, since it is still able to absorb the higher packet injection rate, so network activity continues to increase. For all configurations, total network power levels off after saturation, since the network cannot handle a higher packet injection rate, so the switching activity of the network remains constant.

It is interesting to note that VC_8_8 dissipates approximately the same amount of power as WH64 before saturation. Intuitively, since virtual-channel flow control is a more complicated protocol, requiring more complex hardware, we would expect a virtual-channel router to be more of a power hog than a wormhole router. The interpretation of this is that the power consumed by buffers, links and the crossbar switch is the dominant power consumption in a network node.
Figure 12 shows the power consumed by each component of the router. From these results, it can be verified that the power consumption of the buffer and crossbar components of the router is much more than the power consumed in arbitration.

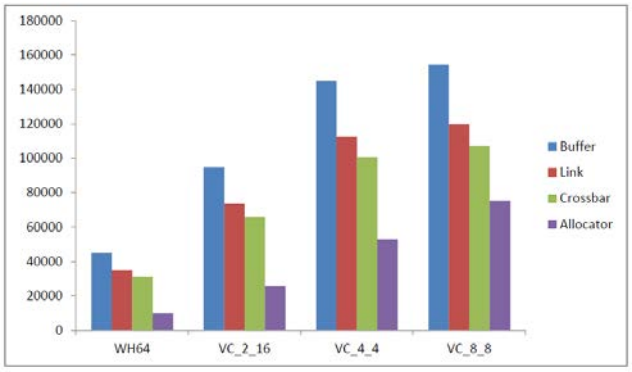


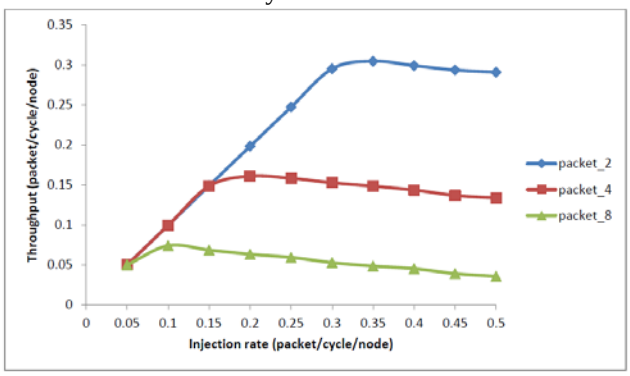

Fig. 12 Power consumption of different router components

### 4.3 Exploring the effect of different packet length

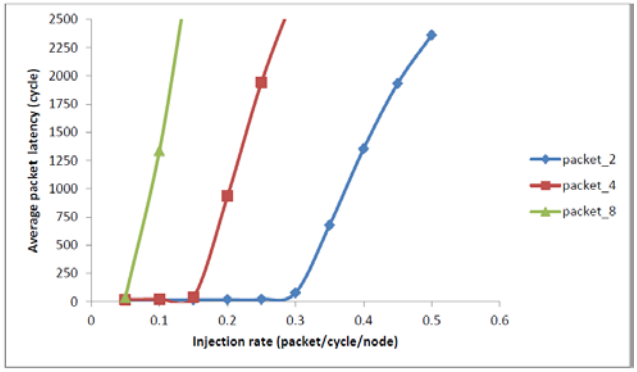
This set of simulations is based on 4X4 mesh with XY routing under uniform random traffic. Figure 13 shows the throughput and average packet latency for packet length equal to 2, 4, and 8 flits.
The throughput increase linearly when the injection rate is low. However, with the injection rate increasing, the confliction encountered in the network limits the increase of the throughput. Figure 13 shows that the average packet latency for 8 flits/packet is larger than that for 2 flits/packet. This is due to two reasons. First, longer packets will take more time to receive. Second, longer packets will cause more confliction at intermediate routers on the path from the source to the destination.

### 4.4 Exploring the effect of different routing algorithms

This set of simulations is based on 4X4 mesh with packet size equal two flits and under uniform random traffic. Table 3 lists the total number of received packets for the XY, West-First, North-Last, Negative-First, and Fully Adaptive routing algorithms for injection rates of 0.1 packets/cycle/node (under saturation), 0.3 packets/cycle/node (saturation), and 0.5 packets/cycle/node (above saturation). These results collected for the 10000 cycles simulation time.




(a)




(b)

Fig. 13 Results of different packet length

Table 3: Total number of received packets

| Routing Algorithm | Injection Rate | | |
|---|---|---|---|
| | 0.1 | 0.3 | 0.5 |
| **XY Routing** | 15960 | 47226 | 46543 |
| **West-First Routing** | 16051 | 43662 | 42310 |
| **North-Last Routing** | 15992 | 44040 | 41915 |
| **Negative-First Routing** | 15933 | 43042 | 40653 |
| **Fully Adaptive Routing** | 15996 | 43546 | 41131 |

From the above table it can be seen that deterministic XY routing is faster than the other three partially adaptive algorithms. Partially adaptive algorithms can potentially speed up the time to deliver individual packets, but globally the results point out to poorer performance than the XY algorithm. Glass and Ni [27] suggested that reducing the number of turns that a message takes may reduce blocking and hence improve performance. This can be justified because adaptive routing has a trend to

concentrate the traffic at the center of the network, increasing in this way the number of blocked paths.

## 5. Conclusions

In this paper, a performance and power network on chip simulator (PPNOCS) based on SystemC has been proposed. As demonstrated, PPNOCS is a general NoC simulation and verification platform with high extensibility. Using PPNOCS, the impact of various architectural level parameters of the on-chip interconnection network elements on its performance and power can be explored. Owing to the general NoC node structure and modularization modeling, developers can develop their own routing algorithm and network topology in such a way that they can use either traffic patterns provided by PPNOCS or their own traffic pattern. Then, the simulation and design verification can be applied. By going through simulation experiments using five classic routing algorithms, the practical usage of PPNOCS is verified. Also, the impact of different traffic patterns, routing algorithms, virtual channel configurations, and packet length on the network performance and power is evaluated. As shown, PPNOCS provided a fast and convenient platform for researching and verification of NoC architecture and routing algorithm.

## References

[1] L. Benini et al., "Networks on Chips: A New SoC Paradigm", Computer, vol. 35, no. 1, Jan. 2002, pp. 70-78.
[2] W.J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks", Proc. 38th Design Automation Conf., ACM Press, 2001, pp. 681-689.
[3] http://www.systemc.org/, “Open systemc initiative.”
[4] Cai L, Gajski D, "Transaction-level modeling in system level design", CECS technical report (03-10), Center for Embedded Computer Systems, Information and Computer Science, University of California, Irvine, March 2003.
[5] Sandro Penolazzi, "A System-Level Framework for Energy and Performance Estimation of System-on-Chip Architectures", Ph.D. thesis, KTH School of Information and Communication Technology, Stockholm, Sweden, 2011.
[6] Thorsten Grotker, System Design with SystemC, Kluwer Academic Publishers, Norwell, MA, USA, 2002.
[7] Gul N. Khan and V. Dumitriu, "A Modelling tool for simulating and design of on-chip network systems", Embedded Hardware Design-Microprocessors and Microsystems, Vol. 34, No. 3-4, pp. 84-95, 2010.
[8] J. Xu, W. Wolf, J. Henkel, S. Chakradhar, "A design methodology for application specific networks-on-chip", ACM Transactions on Embedded Computing Systems, 2006, pp. 263–280.
[9] M. Millberg, E. Nilsson, R. Thid, S. Kumar, A. Jantsch, "The nostrum backbone – a communication protocol stack for networks on chip", in: Proceedings of the 17th International Conference on VLSI Design, 2004, pp. 693–696.
[10] L. Papadopoulos, S. Mamagkakis, S. Catthoor, D. Soudris, "Application – specific NoC platform design based on system level optimization", in: IEEE Computer Society Annual Symposium on VLSI, March 2007, pp. 311–316.
[11] K. Lee, S.-J. Lee, and H.-J. Yoo, "Low-power network-on-chip for high-perforamance soc design", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 14, pp. 148–160, Feb. 2006.
[12] S. E. Lee, J. H. Bahn, and N. Bagherzadeh, "Design of a feasible on-chip interconnection network for a chip multiprocessor (cmp)", in Proc. Of, Computer Architecture and High Performance Computing. Intl. Symp. on, pp. 211–218, 2007.
[13] F. Karim et. al., "An interconnect architecture for networking systems on chips", IEEE Micro, vol. 22, pp. 36–45, Oct. 2002.
[14] Rehan Maroofi, Vilas Nitnaware, and Shyam Limaye, "Area Efficient Design of Routing Node for Network-on-Chip", International Journal of Computer Science Issues (IJCSI), Vol. 8, Issue 4, No 1, July 2011.
[15] T. Kogel et. al., "A modular simulation framework for architectural exploration of on-chip interconnection networks", in Proc. of, Hardware/Software Codesign and System Synthesis, 2003, pp. 338–351.
[16] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures", IEEE Transactions on Computers, vol. 54, pp. 1025–1040, Aug. 2005.
[17] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: A power performance simulator for interconnection networks", in Proc. of, MICRO 35, 2002.
[18] J. Madsen, S. Mahadevan, K. Virk, and M. Gonzalez, "Network-on-chip modeling for system-level multiprocessor simulation", Proc. 24th IEEE Real-Time Systems Symp. (RTSS), 2003, pp. 265-274.
[19] D. Siguenza-Tortosa and J. Nurmi, "VHDL-based simulation environment for proteo NoC", Proc. 7th IEEE Int'l High-Level Design Validation and Test Workshop, 2002, pp. 1-6.
[20] Xinan Zhou, "Performance evaluation of network-on-chip interconnect architectures", M.S. thesis, Department of Electrical and Computer Engineering, University of Nevada, Las Vegas, 2009.
[21] Robert Mullins, Andrew West, and Simon Moore, "Low-latency virtual-channel routers for on-chip networks". In Proceedings of the International Symposium on Computer Architecture, 2004.
[22] W.J. Dally, B. Towles, "Principles and Practices of Interconnection Networks", Morgan Kaufmann, 2004.
[23] M. Dehyadgari, M. Nickray, A. Afzali-kusha, Z. Navabi, "Evaluation of Pseudo Adaptive XY Routing Using an Object Oriented Model for NOC", The 17th International Conference on Microelectronics, December 2005.
[24] H. Kariniemi, J. Nurmi, "Arbitration and Routing Schemes for On-chip Packet Networks", Interconnect-Centric Design for Advanced SoC and NoC, Kluwer Academic Publishers, 2004, pp. 253–282.
[25] P. P. Pande, C. Grecu, A. Ivanov, and R. Saleh, "Design of a switch for network on chip applications", Proc. Int'l Symp. Circuits and Systems (ISCAS), 2003, pp. 217-220.

[26] S. Vangal et al., "An 80-tile 1.28TFLOPS network-on-chip in 65 nm CMOS", in Proc. Solid-State Circuits Conf., Feb. 2007, pp. 98–589.
[27] Glass, C.; Ni, L., "The Turn Model for Adaptive Routing. Journal of the Association for Computing Machinery", v. 41(5), Sep. 1994, pp. 874-902.