Virtual Point-to-Point Connections for NoCs

Mehdi Modarressi, *Student Member, IEEE,* Arash Tavakkol, and Hamid Sarbazi-Azad

***Abstract*—In this paper, we aim to improve the performance and power metrics of packet-switched network-on-chips (NoCs) and benefits from the scalability and resource utilization advantages of NoCs and superior communication performance of point-to-point dedicated links. The proposed method sets up the virtual point-to-point (VIP) connections over one virtual channel (which bypasses the entire router pipeline) at each physical channel of the NoC. We present two schemes for constructing such VIP circuits. In the first scheme, the circuits are constructed for an application based on its task-graph at design time. The second scheme addresses constructing the connections at run-time using a light-weight setup network. It involves monitoring the NoC traffic in order to detect heavy communication flows and setting up a VIP connection for them using a run-time circuit construction mechanism. The proposed mechanism is compared to a traditional packet-switched NoC and some modern switching mechanisms and the results show a significant reduction in the network power and latency over the other considered NoCs.**

***Index Terms*—Multiprocessor system-on-chip (MPSoC), network-on-chip (NoC), performance evaluation, power consumption, virtual point-to-point (VIP).**

I. Introduction

AMONG DIFFERENT on-chip communication mechanisms, point-to-point connections where packets travel on dedicated pipelined wires which directly connect their source and destination nodes can yield the ideal performance and power results. A network-on-chip (NoC) [1]–[3] increases these ideal values by the delay and power related to the router pipeline stages (buffer read/write, route selection, arbitration and crossbar traversal). An analytical and experimental comparison of the power and performance metrics of dedicated point-to-point links and NoCs can be found in [4] and [5]. Poor scalability and considerable area overhead are the important drawbacks of dedicated links. In addition, predicting the delays of these links before the late stage in the design cycle (when the actual layout and routing are performed) is difficult [6]. In [7], the authors compare some on-chip communication schemes and show that NoCs are the most predictable mechanism in terms of the difference between the estimated and actual working frequency (obtained before and after placement and route).

In this paper, we introduce a packet-switched NoC which can also provide low-power and low-latency dedicated virtual point-to-point (VIP) paths between any two nodes by bypassing the pipeline of the intermediate routers. In this design, one virtual channel (VC) of each physical channel is designated to bypass the router pipeline stages. In this VC, the buffer is replaced by a register (1-flit buffer) which holds the flits arriving on the VC. Moreover, the switch allocator, VC allocator, and flow-control units are slightly modified in order to store VIP paths (by keeping the selected output port for each incoming VIP connection) and provide each VIP with its required bandwidth. The packets traveling on VIP connections do not find any channel busy along their path by prioritizing them over the packet-switched data and also not allowing the VIP connections to share the same links (i.e., each router port can be used by at most one VIP connection). Consequently, being constructed by chaining the VIP registers along the path, VIP connections can act as a dedicated pipelined-link. In other words, the flits only travel over the crossbars and links which cover the actual physical path between their source and destination nodes and skip through buffer read, buffer write and allocation operations.

However, unlike dedicated point-to-point links, which are physically established between the communicating nodes of a multicore chip (based on the communication pattern of the target application) and are fixed during the system lifetime, the VIP connections are dynamically reconfigurable and can be established based on the traffic pattern exhibited by the running application. It is an important feature, as several different applications are integrated onto today's multicore systems-on-chip (SoCs) and chip multiprocessors (CMPs) and communication characteristics can be very different across the applications. Besides, being established over conventional NoCs, VIPs benefit from the predictability and reusability of NoC architectures.

The fact that VIPs are not allowed to share the same link restricts the number of VIPs that can simultaneously exist in the network. However, in most of the applications implemented as a multicore SoC [8], [9] each core communicates with a few other cores (less than 3, on average). Similarly, CMPs, as a replacement for the traditional multi-chip multiprocessors, can benefit from both temporal and spatial communication locality of their typical workloads. Temporal locality represents the effect of temporal aggregation of the inter-core communications [10]. High-temporal locality suggests that during any given time period, inter-core communication occurs across a certain number of connections,

Manuscript received September 15, 2009; revised January 5, 2010. Date of current version May 21, 2010. This paper was recommended by Associate Editor A. Jantsch.

M. Modarressi is with the Department of Computer Engineering, Sharif University of Technology (SUT), Tehran, Iran (e-mail: modarressi@ce.sharif.edu).

A. Tavakkol is with Supercomputing Center, Institute for Research in Fundamental Sciences, Tehran, Iran (e-mail: arasht@ipm.ir).

H. Sarbazi-Azad is with the Department of Computer Engineering, Sharif University of Technology (SUT), Tehran, Iran, and also with the School of Computer Science, Institute for Research in Fundamental Sciences, Tehran, Iran (e-mail: azad@sharif.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2010.2048402

which can be called a *communication working set*. Spatial locality is determined by the distribution of the connections in the application and determines the size of the working set. It has been shown that in most of the CMP benchmarks, each node tends to have a small number of favored destinations for the messages it sends [11]. For example, the NASA advanced supercomputing (NAS) parallel benchmark suite exhibits very high-spatial locality and, therefore, contains small working sets [11]. Similar behavior has been reported for scientific [11] and commercial workloads [12]. As a result, we expect the VIP connections to be able to handle most of the on-chip traffic and deliver them to the destination node with near-ideal power consumption and delay. Nonetheless, if the VIP connections cannot cover all of the inter-core traffic, the VIPs are set up for higher volume communication flows to obtain more power and performance gains and the other flows will use the packet-switched network.

Our early results showed that VIPs are a promising on-chip communication mechanism [1], [14]. In this paper, we first introduce a static mechanism for VIP construction. In this scheme, the VIPs are constructed for an application based on its task-graph at mapping and route selection phases of the NoC design procedure. In multicore SoCs, mapping involves physically placing the cores in the network, while in CMPs with several identical processing nodes, mapping is accomplished by assigning each task to a network node. On starting execution of each application, the corresponding VIP connections are set up in the network. We then propose a dynamic circuit construction mechanism which uses a light-weight setup network in order to monitor the network traffic pattern and dynamically set up a VIP for the traffic flows with a communication volume higher than a predefined threshold. The setup network consists of a root node together with a simple network. This simple network provides the root node with current NoC state. The root node finds a VIP of minimum conflicts with already established VIP connections for high-volume communication flows. Finding the best path for setting up a VIP of a requesting communication flow is carried out in parallel with normal NoC operation, hence it does not accompany the long circuit setup-time of the traditional circuit-switched NoCs. We also investigate different aspects of dynamic and static VIP connections and evaluate their impact on NoC power and performance, compared to some efficient NoC designs including express virtual channels (EVC) and *speculative pipelined routers*.

The rest of this paper is organized as follows. Section II surveys some related work. Section III introduces the proposed router architecture. A static and a dynamic scheme for constructing VIP connections is presented in Sections IV and V, respectively. Section VI presents the experimental results, and finally, Section VII concludes this paper.

II. Related Work

The need for scalable on-chip communication architectures is pointed out in [2], [15]–[20]. Improving the power and performance metrics of packet-switched NoCs by integrating a second switching mechanism has been addressed in several previous works, such as hybrid circuit packet-switched networks [21], EVCs [4], and long range links [22].

Reducing the average packet hop count by physical bypass paths is introduced in [23]. The authors showed that their method reduces the NoC power consumption with 10% area overhead when running a multimedia application. However, our proposal reduces the average packet hop count by virtually bypassing the intermediate routers, and as will be shown in Section VI, it gives improvements close to this reconfigurable NoC, with a negligible area overhead.

Circuit-switching [24] has been used in a number of NoC architectures to reduce on-chip communication latency, when compared to packet-switched networks, since packets need not go through routing and arbitration once circuits are set up. However, this switching method often suffers from performance degradation due to circuit setup delay and poor resource utilization. The structure of VIP connections in this paper differs from the circuits in traditional circuit-switched NoCs in two ways: first, unlike circuit-switching, VIPs do not work based on time slot reservation, hence do not accompany the complex slot allocation procedure and poor resource utilization of circuit-switching. Second, in both static and dynamic VIP construction schemes proposed in this paper, the long setup time of the circuit-switching is removed.

Some methods try to benefit from the interesting properties of two switching methods (circuit and packet switching methods) by integrating them into a single NoC fabric [12], [18], [25], [26]. Our proposal also differs from these NoCs, since the main task of the packet-switched part of them is to carry out the circuit setup process rather than transferring the application data. They also give no specific algorithm to find a path for the circuits, while we develop an efficient algorithm to find the best path for a VIP. Message-passing asynchronous network-on-chip providing guaranteed services over OCP (MANGO) NoC [18], for example, provides connection-oriented guaranteed services (GS) as well as connectionless best-effort (BE) routing. The BE and GS routers are implemented separately and BE router is only employed to setup GS connections. Another hybrid NoC is proposed in [25], which support the GS traffic by a reconfigurable circuit-switched network. Applying the spatial division multiplexing (SDM) scheme, the link between two routers is divided into small 4-bit channels, or lanes, and each circuit is composed of several lanes. Again in this architecture, the packet-switched network is used to carry the configuration data of the circuit-switched sub-network. The SDM method also imposes some area overhead.

EVCs [4] allow packets to virtually bypass intermediate routers along their path. EVCs are restricted to connect nodes only along a single dimension and cannot turn from one dimension to another. Although VIPs are constructed on one VC of each router port, they are different from EVCs since the VIP connections aim to exploit the on-chip traffic locality and virtually provide the communicating nodes with dedicated point-to-point links and completely connect the source and destination of a communication flow, rather than providing shortcut paths to improve the performance of a packet-switched network.

The concept of setup network is used in some works before, but our proposed setup network differs from them. A setup network was proposed in [12] which handles the construction and teardown of circuits and stores the switch configuration for active circuits. The main data network supports both circuit-switched and packet-switched traffic. The setup network is packet-switched and has three-stage pipeline routers. Constructing a new circuit may destroy some old circuits. Our setup network is simpler and finds the best path for constructing a VIP. In [27], a control network was used to propagate global congestion information and load balancing. A kind of run-time NoC adaption is proposed by Al-Faruque *et al.* in [28], [29]. In AdNoC architecture [28], for example, selecting the suitable output port for each packet and distributing the VC buffers among the router ports are done based on the current NoC state. Although AdNoC effectively increases the resource utilization, it requires some extra logic and may increase power consumption. Our proposed NoC aims at reducing the average communication power and latency by optimizing switching method for high-volume communication flows.

III. Proposed NoC Architecture

Although the packet-switched part of the proposed NoC is not restricted to specific switching and routing schemes, the NoC routers, in this paper, employ the wormhole switching mechanism with conventional five-stage pipelined routers [24] (Fig. 1(b)] which are slightly modified in order to support VIP connections.

A. Structure of VIP Connections

Fig. 1 displays the microarchitecture of the proposed router. The figure shows the implementation details of one input port and one output port of the router. As shown in the figure, in one virtual channel (VC 0) in each physical channel, the buffer is replaced by a register (1-flit buffer). These VCs are devoted for establishing VIP connections between any two given nodes. The switch allocator unit is also slightly modified in order to prioritize the VIP data over packet-switched flits. This router uses a multiplexer-tree-based crossbar fabric. Since a packet coming through an input port does not loop back, each multiplexer is connected to three input ports as well as the local PE.

Each VIP register has a signal called *full* which is set to 1 when the register has incoming flits to service. In the input ports, this signal is used to select the proper *select* signal of the multiplexer connecting one of the VCs to the crossbar input. If the register is empty ($full = 0$), a VC is selected based on the outcome of the routing function, virtual-channel allocation, and switch allocation units, just like in traditional packet-switched networks. Otherwise, *select* signal is set to 0 and the flit in the VIP register (VC 0) is directed to the crossbar input.

The *full* signal is also used to control the crossbar operation. Each output port contains a VIP allocator which, for each output port belonging to a VIP, determines (using a 2-bit register) the input port connected to this output port along that VIP. This logic takes the *full* signal of the input ports, as well as the output of the switch allocator unit, as input and allocates the output port to the input port of the VIP when it has an incoming flit in its VIP register to forward. Otherwise, the allocation is done according to the outcome of the switch-allocator unit, just like in traditional packet-switched NoCs.

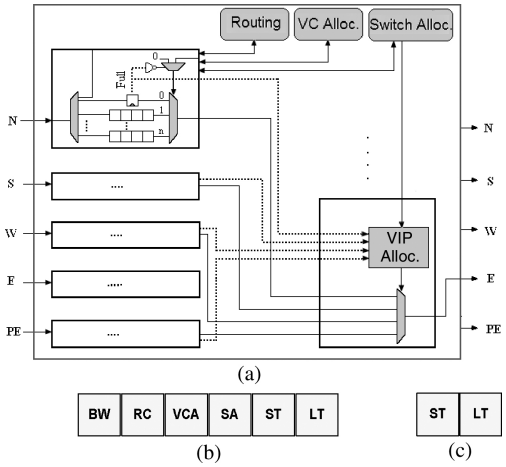


Fig. 1. (a) Proposed router architecture. (b) Pipeline stages of a canonical router. (c) Pipeline stages passed by VIP flits.

Supporting VIPs in a baseline packet-switched router involves adding a 1-flit buffer at each input port, a simple VIP allocator at each output port, and some 1-bit wires to propagate the *full* signals which has minimal impact on the baseline router architecture and imposes a negligible area overhead.

A VIP for a communication flow is established by appropriately setting the VIP allocators in each router in such a way that the VIP registers in each router are connected to proper output ports and then to the registers in the next router along the VIP. The source node of a communication flow selected for traveling on a VIP sends corresponding packets using VC number 0 of the output port through which the flits should be transferred. Since the VIP is already set up, the VIP allocator of the designated output port selects the injection port and connects it to the output, when the source node injects a new flit. When the flits reach the next node, they are de-multiplexed based on their VC identifier and the VIP flits (identified by VC identifier 0) are stored in the VIP register of the input port. Similarly, this register and the other registers in the intermediate routers along the path are connected to the proper output port and construct a VIP connection toward the destination.

As a result, at each hop, the VIP flits pass through two pipeline stages, switch traversal (ST) and link traversal (LT), as shown in Fig. 1(c), with VIP registers acting as staging registers. Since the VIP flits bypass the other router pipeline stages, the power consumption and delay related to buffer read and write, route calculation, and arbitrations are removed.

In the current implementation, the VIP data has the same flit structure as the packet-switched network. Nonetheless, the data traveling on VIPs can have a simpler format since they do not have to carry destination address and other fields required by a conventional packet-switched network [24]. The cores can even bypass the network adaptor and communicate through the protocol implemented in their core interface (for example OCP). This leads to further power and performance

improvements by removing the power and delay related to packetization and de-packetization of data. In this case, all required services (such as error control) are provided by the core interface protocol, just like in a dedicated physical link.

B. VIP Bandwidth Allocation

We have designed a simple end-to-end flow control mechanism between the nodes based on the traditional on/off flow control mechanism [24]. An *off* signal is sent by the downstream node of a VIP when the free buffers at its interface falls below the threshold T_{off}. If the number of free buffer rises above a threshold T_{on}, an *on* signal is sent. When the *off* signal is received at the VIP upstream node, it stops transmission of data over the VIP until an *on* signal is received. These signals are sent along the path from the source to destination of a VIP. This path is set up during the VIP construction process. The signals propagate to the sender along this path one hop per cycle. When an *off* signal is transmitted to the upstream node, there must still be sufficient buffering at the receiver to store all of the data items in transit, as well as all of the data items that will be injected during the time it takes for the signal to propagate back to the sender. As a result T_{off} must be at least $2 \times d$, where d is the VIP length (in terms of hop count).

Prioritizing the data traveling on VIP connections over packet-switched flits, however, may produce starvation at the intermediate routers along the VIP, if a VIP source node generates burst traffic in some consecutive cycles. To handle this scenario, we have modified the proposed flow control mechanism to avoid starvation at the intermediate routers. To this end, the *on* and *off* signals can also be asserted by the intermediate routers along a VIP.

We define two thresholds T_{vip} and T_{ps} for each link belonging to the path of a VIP which are set to some values proportional to the percentage of link bandwidth allocated to the VIP and packet-switched network. Each intermediate router counts the number of consecutive cycles on which the data of a VIP connection are serviced. If this number of cycles exceeds the threshold T_{vip} while there are packet-switched flits waiting to use the output port, it asserts the *off* signal to service some packet-switched flits and sends an *on* signal after T_{ps} cycles to allow the VIP source to resume sending data. T_{vip} and T_{ps} are two design parameters that can be set for each VIP separately. Without loss of generality, we set T_{ps} to the number of flits of a packet, which equals the number of cycles it takes for a packet to completely move from the current router to the next one. T_{vip} is then determined proportionally to its bandwidth usage and is set to

$$T_{vip} = a \times T_{ps} \times \frac{BW\_Usage(VIP)}{100 - BW\_Usage(VIP)}$$

where $BW\_Usage(VIP)$ is the percentage of link bandwidth devoted to the VIP and a is a constant factor which determines the burstiness allowed for the VIP traffic. If the traffic flow corresponding to a VIP is bursty in nature, by setting a to higher values, larger number of consecutive cycles can be allocated to VIP without being interrupted by the packet-switched network. In the current design we set a to 1.

This scheme of bandwidth allocation does not suffer from the resource utilization problems of TDMA-based circuit-switched NoCs, since it does not reserve NoC cycles (time slots) for the two NoC parts (packet-switched part and VIPs). Instead, each part uses the links when it has incoming flits to forward and the proposed flow control mechanism guarantees that each part is provided with the required bandwidth.

To remove redundant *off* signal transmissions, if multiple nodes are getting starved at the same time, a node which has already asserted the *off* signal blocks the *off* signals received from downstream nodes. Moreover, an *off* signal resets the counter (counting the number of serviced VIP flits) in all upstream routers along the VIP connection. A similar policy is applied for *on* signals.

IV. Static VIPs

Most multicore SoC programs have a small number of communication flows through which each core communicate with a small number of other cores. Moreover, the traffic pattern of such applications is known in advance.

In the static VIP construction scheme, the VIPs are exploited to improve the power and performance metrics of the NoC when running a specific target application with known traffic characteristics. Simply stated, for a given input application, our objective is to 1) physically map the cores of the application into different nodes of a mesh-connected NoC, 2) establish as many VIP connections as possible for the communication flows of the application, and 3) direct the flows for which a VIP could not be constructed through packet-switched network, in such a way that the total power consumption and latency of the NoC is minimized.

Each input application is described as a communication task-graph (CTG). The CTG is a directed graph $G(V, E)$, where each $v_i \in V$ represents a task, and a directed edge $e_{i,j} \in E$ characterizes the communication from v_i to v_j. The communication volume (bits per second) corresponding to every edge $e_{i,j}$ is also provided and is denoted by $t(e_{i,j})$.

The core mapping is accomplished in some steps by modifying NMAP (a simple and fast heuristic power-aware core mapping and route generation method presented in [8]) with respect to VIP construction as follows.

1) Partition the CTG vertices into two sets: *Mapped* (which contains the already mapped CTG vertices and is initially empty) and *Unmapped* (which includes the CTG vertices not yet mapped and initially contains all CTG vertices). Once a vertex is mapped it is moved to the *Mapped* set. Similarly, partition the NoC nodes into two sets: *Allocated* (including the nodes to which a CTG node has been already assigned) and *Unallocated* (which includes the free nodes). The *Allocated* members form a contiguous region defined as *Used Region*.
2) Map the core with the maximum communication demand onto one of the mesh nodes with maximum number of neighbors. The communication demand of a node is defined as the cumulative weight of the incoming and outgoing edges connected to the node.

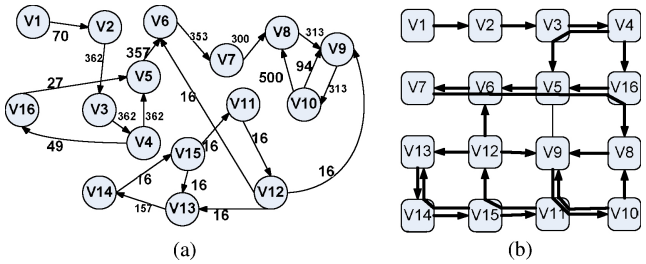


Fig. 2. (a) VOPD CTG and (b) its mapping on a 4×4 mesh. Each arrow is a VIP between two nodes.

3) Repeat the following steps until the *Unmapped* set becomes empty (i.e., all nodes are mapped).
   a) Select the core v_i that communicates most with the *Mapped* members.
   b) Examine all unallocated mesh nodes adjacent to the boundary nodes of the *Used Region* for placing it. Considering only the boundary nodes for placing the new core extends the *Used Region* contiguously. For each selected mesh node.
      i) Select the CTG edges between the selected core v_i and the *Mapped* members in order of their communication volumes.
      ii) Examine the possibility of constructing a VIP for the edge (i.e., the existence of sufficient unused resources) along one of the shortest paths between its source and destination nodes. Map the core onto the node which allows more traffic to be directed on VIPs. More precisely, select the node which maximizes

$$\sum_{\forall e \in E \text{ connected to } v_i} t(e) \times VIP(e)$$

where *VIP(e)* is obtained by

$$VIP(e) = \begin{cases} 1, & if \text{ a VIP is reserved for } e \\ 0, & \text{otherwise} \end{cases}$$

if more than one node provide the same amount of traffic with VIPs, select the node x which minimizes

$$\sum_{\forall v_j \in V \text{ connected to } v_i} t(e_{i,j}) \times dist(x, M(v_j)) \times VIP(e_{i,j})$$

where $dist(a, b)$ is the Manhattan distance between NoC nodes a and b, $M(v_j)$ is the NoC node into which the CTG vertex v_j is mapped, and $t(e_{i,j})$ is the communication volume of edge $e_{i,j}$. $VIP(e_{ij})$ is defined as above. The relation indicates that the node with minimum cumulative product of the VIP volumes and VIP hop counts is selected.
      iii) The edges for which a VIP could not be found should be handled by the packet-switched network. Select unmapped edges in order of their communication volumes. Then, for each edge consider all shortest paths between its source and destination nodes and select the path which has the minimum overlap with VIPs. The path should not violate the bandwidth of the network links (to avoid congestion). More formally, the bandwidth constraint of each NoC link l_k must be satisfied as

$$\forall l_k \, BW(l_k) \geq \sum_{\forall e_{i,j} \in E} X^k(i, j)$$

where $BW(l_k)$ is the bandwidth of link l_k and $X^K(i, j)$ is obtained by

$$X^k(i, j) = \begin{cases} t(e_{i,j}), & \text{if } l_k \in \text{path}(e_{i,j}) \\ 0, & \text{otherwise} \end{cases}$$

where $path(e_{i,j})$ represents the set of links onto which CTG edge $e_{i,j}$ [with volume $t(e_{i,j})$] is mapped either on a VIP or on a packet-switched part.

After finding a route for all CTG edges, VIP connections are constructed by setting the VIP allocators, while packet-switched routes (if any) are set up by appropriately setting the routing table of the routers. For example, Fig. 2 displays the CTG and VIP connections constructed in a 4×4 mesh for video object plane decoder (VOPD) [8] application. As shown in the figure, all of the communication flows generated by this application are directed through VIPs.

There is also a rather similar VIP construction problem: constructing VIP connections based on the CTG of an application on an NoC with a given mapping. In this case, the physical placement of cores is fixed and the problem is reduced to find a VIP connection or packet-switched route for the CTG edges. This problem can be solved by the described algorithm by simply removing the steps which deal with the physical placement of the cores.

The algorithm described above also work for homogeneous CMPs, where we allocate each CMP node to one of the CTG tasks, rather than physically map the cores into a mesh-based network. However, these two operations are equivalent from the view point of our algorithms and hence, the same algorithms can be applied in both cases.

V. Dynamic VIPs

This section proposes a run-time dynamic VIP construction scheme. This approach is useful in the cases where it is not possible to know in advance the exact communication pattern of running applications. Constructing a VIP using this procedure may take a few clock cycles. The key point, however, is that the proposed procedure is performed in parallel with normal NoC operation as a background process, i.e., during searching for a path for VIP construction, the packets of the requesting traffic flow still keep on traveling on the packet-switched network. Therefore, unlike the traditional circuit-switched networks, this setup latency does not degrade the network performance.

A. Problem of Dynamic VIP Construction

Setting up VIP connections at run-time, as mentioned before, involves 1) monitoring the on-chip traffic, and

2) developing an algorithm that dynamically reconfigures the VIP connections based on the NoC state extracted from the monitored data. To formulate the problem and the proposed algorithm, we need to define some new terms.

1) *Communication Flow Volume:* The volume of a communication flow between nodes s and d is defined as the number of flits that are originated form s and destined to d during a given interval.

2) *Communication Flow Weight:* The weight of a communication flow between nodes s and d is defined as the value assigned to each flow which is proportional to its communication volume and the Manhattan distance between s and d.

3) *VIP Weight:* The VIP weight is defined as the value assigned to each VIP connection which is equal to the weight of the communication flow for which the VIP is constructed.

4) *VIP Path:* The set of consecutive links which form a VIP from its source to destination nodes.

5) *Current VIP Set:* Represented by *VIPS*, is defined as the set containing all of the pre-existing VIP connections in an NoC at any given time. Each $vip_{i,j} \in VIPS$ between nodes i and j is characterized by its path $P(vip_{ij})$, weight $W(vip_{ij})$, and length (hop count) $L(vip_{ij})$.

6) *Current Communication Flow Set:* Represented by *CFS*, is defined as the set containing all of the communication flows in an NoC at any given time. Each $cf_{i,j} \in CFS$ between nodes i and j is characterized by its communication volume $V(cf_{ij})$, weight $W(cf_{ij})$, and whether this communication flow is directed over a VIP or the packet-switched network. The latter is obtained by

$$IsVIP(cf_{i,j}) = \begin{cases} 1, & \text{if } vip_{i,j} \in VIPS \\ 0, & \text{otherwise} \end{cases}$$

taking $cf_{i,j} \in CFS$ as input, the function returns *true* if there is a VIP between nodes i and j, otherwise returns *false*.

7) *Shortest Path Area:* The shortest path area between nodes s and d is defined as the set of all the NoC nodes and links which are located along one of the shortest paths between these two nodes.

Formally, the problem of dynamic VIP construction can be stated as follows: *given* an $n \times m$ mesh-connected VIP-enabled NoC, the current VIP set *VIPS*, the current set of communication flows *CFS* which may vary in time, *find* a new configuration for VIP connections in response to a change in *CFS* such that

$$MAX\left\{\sum_{\forall cf_{i,j} \in CFS} W(cf_{i,j}) \times IsVIP(cf_{i,j})\right\}.$$

Updating the *CFS* and *VIPS* sets involves monitoring the on-chip traffic and setting the new weight for each VIP and traffic flow. Once the weights are updated, our algorithm sorts all $cf_{x,y} \in CFS$ which satisfies $W(cf_{x,y}) > T$ and $IsVIP(cf_{x,y}) = 0$ in decreasing order, where the weight threshold T is a parameter defined in order to determine high-volume communication flows: each $cf_{x,y}$ weighting higher than T is designated for VIP construction. Afterwards, for every selected $cf_{x,y,}$ in the order, the algorithm finds a VIP connection $vip_{x,y}$ within the shortest path area between x and y with minimum cost $MinCost(vip_{x,y})$ defined as

$$MinCost(vip_{x,y}) = \sum_{\forall vip_{i,j} \in VIPS} W(vip_{i,j}) \times Conflict(vip_{x,y}, vip_{i,j})$$

where $Conflict(vip1, vip2)$ is defined as

$$Conflict(vip1, vip2) = \begin{cases} 0, & \text{if } P(vip1) \bigcap P(vip2) = \phi \\ 1, & \text{otherwise} \end{cases}$$

the relations indicate that the algorithm should minimize the cost of VIP construction which is defined as the cumulative weight of the VIPs that have to be torn down in order to make room for the new VIP. So, if there are not sufficient free resources for constructing a new VIP (as a VIP connection cannot share a common port with other VIPs), our algorithm is allowed to tear down old VIPs.

If $W(cf_{x,y}) > MinCost(vip_{x,y})$, the newly found VIP $vip_{x,y}$ will be set up in the NoC and all VIPs $vip_{i,j}$ with $Conflict\,(vip_{x,y}, vip_{i,j}) = 1$ will be torn down, otherwise the VIP construction request is rejected and no VIP will be constructed. The procedure should then notify the source of a torn down VIP that it has to send data through packet-switched network. The traffic flows corresponding to the torn-down VIPs can request for constructing a new VIP at the next invocation of the procedure (or round) provided that their flow weight is still above the given threshold.

Due to the varying nature of the on-chip traffic, this procedure must be called periodically in order to adapt the VIPs to the current NoC state. On starting at specific times, this procedure continues until all VIP construction requests are serviced. Then, it will be inactive until the next invocation.

B. VIP Construction Approach

In our previous work [13], we proposed a distributed algorithm for dynamic VIP construction. We refer interested readers to [13] for more details of this algorithm. In this paper, we propose an alternative approach for dynamic VIP construction which works in a centralized manner. In this approach, a root process is added to the setup network which keeps track of the network state, characterized by the path and weight of VIP connections and communication flows. The root process runs Dijkstra's algorithm to find a VIP path with minimum cost for a requesting communication flow. Since, most current SoCs contain a central root or configuration process (or processor) that configures the system [26], we assign the task of managing VIP connections to this processor. Being simple, fast, and infrequently invoked, the algorithm does not impose considerable overhead to the root processor. Moreover, since the VIP construction job is performed in parallel with NoC normal operation, there is no need for real-time response and the algorithm can be given a lower priority than the current tasks of the root processor. Centralized algorithms have been widely used in state-of-the-art NoCs for conducting run-time management tasks such as mapping, slot-allocation, resource management, and so on [30]–[32]. Reducing the number of communication transactions was our main motivation to move toward a centralized algorithm. In the centralized algorithm,

since the path computation is done by a single processor rather than by transactions among network nodes, as in the distributed algorithm proposed in [13], the communication transactions are reduced to sending the new weight of the pre-existing VIPs and VIP construction requests to the root node. This considerably reduces the overall traffic produced for dynamic VIP construction. We obtained more than four times lower traffic compared to the previous distributed algorithm for the scenarios considered in Section VI-B.

Nonetheless, centralized algorithms generally suffer from a single point of failure, poor scalability, and communication hot-spot around the root node. However, we do not expect the centralized VIP management scheme to be a bottleneck of our system due to the infrequent reconfiguration and appropriately scheduling the communication between the setup network nodes and the root node (Section V-C). As will be shown in Section VI, the proposed centralized procedure can efficiently manage the VIPs in up to 64-node NoCs. However, it may not scale well when the NoC size increases. In this case, we can divide the NoC nodes into some clusters. Each cluster has a local root which manages the VIPs inside the cluster. A global root then handles the VIPs spanning over several clusters.

C. Setup Network

The setup network is composed of a root node and a very simple network which is used by the root node in order to collect network state and VIP construction requests and send appropriate control data for setting up and tearing down VIPs. The size of the setup network is the same as the size of the main data network and each node in the setup network corresponds to a node in the data network with the same address. Each node is simply composed of an internal switching fabric and limited buffering space. It also has a simple controller which directs the packet toward the root node and from the root node to other nodes. The controller has an interface to the main data network through which it receives the current state of the NoC. In addition, the interface can configure the VIP allocator units in the data network in order to set up or tear down VIPs. Due to the limited space, we refer interested readers to [33] and [13] for more details on the setup network implementation.

Having a small bit-width and a simple internal structure, we expect that the area of the setup network to be negligible compared to the main data-network. We have implemented a 3-bit wide setup network node in VHSIC hardware description language (VHDL) in order to assess its area. The synthesis result shows that each node is implemented with around 390 gates, thus confirming our expectation that its area overhead is negligible.

As the main data network is 64-bit wide and the setup network is 3-bit wide, it increases the wires of each link by 6%. The Orion II area estimation function reports the area of our 64-bit routers and a 1 mm^2 links to be 0.089 mm^2 and 0.048 mm^2, respectively. According to these area numbers, the area of an 8×8 NoC for example (with 112 bidirectional links and 64 routers) will be increased by less than 2%, due to the additional setup network links.

D. Dynamic VIP Construction Procedure

The description of the proposed dynamic VIP construction procedure (see Section V-A) indicates that it should handle providing the root node by the NoC state (updated VIP weights, ...), finding a new VIP configuration by the root node, constructing new VIPs, and destroying old VIPs. Here, we describe how the procedure accomplishes these jobs.

The whole procedure relies on monitoring the current on-chip traffic pattern. Traffic monitoring is simply done at each node in a distributed manner. Each node keeps track of the communication flows it originates by storing the number and destination of the packets it sends in a table. The weight of each flow is an n-bit value proportional to the flow volume.

Periodically, at specific times, the n most-significant bits of the register holding the traffic volume are considered as the new weight of the flow. This value is multiplied by the traffic flow length, defined as the Manhattan distance between the source and destination nodes of the flow; this is because as the flow length increases, the contribution of the flow in the entire on-chip traffic increases. If a VIP has already been set up for the flow, the VIP weight is set to the weight of its corresponding traffic flow.

E. Collecting the NoC State

The NoC state is characterized by the current weight of each VIP, the current weight of each communication flow, and the average weight of the NoC flows. Collecting the required data from the NoC by the root node involves two different communication transactions over the setup network: broadcasting a request from the root node to the setup network nodes, and sending the required information from the setup network nodes to the root. A header is added to the setup network messages to indicate which target data must be collected by each transaction. Setup network nodes, in turn, send the response messages with the same header. Having a bit-width of 3, setup network supports eight distinct headers.

For the first transaction, broadcasting a request or a data from the root to the network, the root node sends the message on its row along both the E and W directions. Each setup network node at the same row of the root node, on detecting the message, sends it to the nodes in its column along both the N and S directions. Fig. 3(a) shows the path through which the messages are broadcasted. For the second transaction, however, since the setup network nodes are simple, we adopt a scheduling scheme in order to remove conflict among the messages destined to the root. According to this scheduling, the setup network is divided into four quadrants and the nodes inside each quadrant first direct their messages to the node at the row or column in which the root node is located and then to the root node.

Fig. 3(b) demonstrates the path through which each node sends its messages to the root node. Each node accepts the message of its previous node only when it has finished sending its message to the next node along the assigned path. A conflict may occur at the nodes in the same row/column of the root node, when one message from their row and one message from their column reach these nodes at the same cycle. For example,

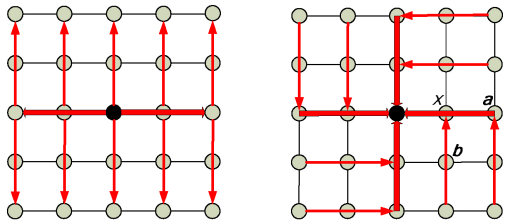


Fig. 3. (a) Path for broadcasting requests from the root to the setup network nodes. (b) Paths for sending data to the root node. Black node is the root node.

at node x in Fig. 3(b), a conflict may happen between messages approaching from nodes a and b. To solve this conflict, our setup network prioritizes the messages traveling along rows over the conflicting messages received from columns.

These two transactions are used for updating the weight of the existing VIPs in the root, collecting required information to update the value of the VIP weight threshold, and requesting VIP construction. The proposed scheduling schemes work correctly for any given location of the root node, but it is more beneficial to map the root node into one of the central nodes, since it allows more parallelization in data transmission.

F. Finding a Proper VIP Path for a Requesting Flow

This process starts after the previous step, when all VIP weights are updated and communication flows weighting higher than the VIP weight threshold request for a VIP. Receiving the requests, the process sorts them in a decreasing order of weights. Servicing the requests in this order assures that no newly constructed VIP is torn down by a consequent VIP at the same round, since a VIP cannot destroy a VIP with greater weight.

Then, for each request, in the order, a path with the minimum cost is found. We apply Dijkstra's algorithm to find a path with minimum cost for the VIPs. There are two considerations related to the implementation of the Dijkstra's algorithm for our specific problem. First, we must design an appropriate scheme for constructing a directed weighted graph out of the current configuration of VIP connections (to be used as the input of Dijkstra's algorithm). Second, while Dijkstra's algorithm does not guarantee that the path with minimum weight is also a minimal path in terms of hop count, VIPs must be constructed along one of the shortest paths between the source and destination nodes of the communication flow.

G. Constructing the Graph

For an $n \times n$ NoC, we construct a *path graph* $(PG) = (Q, R)$, where $|Q| = n^2$ and each $q_i \in Q$ represent a node, each of which corresponding to one of the NoC nodes, and a directed edge $r_{i,j} \in R$ characterizes the communication from q_i to q_j.

In our mesh-connected NoC, each NoC node is connected by a directed link to at most four adjacent routers, one in each cardinal direction. If the link between an NoC node v_i and its adjacent node v_j is not used by a VIP connection, we consider a directed link $r_{i,j}$ with the weight of 0 between q_i and q_j, their corresponding nodes in PG. Otherwise, an edge with a weight equal to the weight of the VIP passing over it is added to PG, between q_i and all of the downstream links along the VIP. This is because any node within the path of a VIP vip_i can construct a new VIP to any downstream node along vip_i with the cost of $W(vip_i)$ by only tearing down vip_i. For example, in Fig. 4 which displays a sample 3×3 NoC and its corresponding PG, the minimum cost for constructing a VIP from nodes 4 to 2 is 5 which can be obtained by destroying $VIP1$. If more than one edge can be created between two nodes in PG (for example between nodes 4 and 2 through $VIP1$ and $VIP2$), the edge with minimum weight is considered in PG (the edge with the weight of 5 between PG nodes 4 and 2).

H. Finding a Path in the Shortest Path Area

The other consideration is that the path found for constructing a VIP between any two nodes s and d must be laid out inside the shortest path area between s and d. Since the VIPs are constructed along one of the shortest paths between two nodes, they can span at most along two directions determined by their source and destination addresses. Hence, each VIP belongs to one of the four possible quadrants: North-East, North-West, South-East, and South-West. Similarly, the PG edges are grouped into the same four quadrant sets (North-East, North-West, South-East, South-West) according to the address of their source and destination nodes. A 2-bit flag determines the corresponding quadrant of the edges of PG, which is set when the edge is added to PG.

When the algorithm is initiated to service a VIP construction request from the NoC node s to the NoC node d, the algorithm starts finding the shortest path from the PG node q corresponding to the NoC node s. The algorithm then 1) only considers the graph nodes corresponding to the NoC nodes belonging to the shortest path area between nodes s and d, and 2) only considers the graph edges belonging to the same quadrant as the requesting VIP. The first condition guarantees that the found path only consists of the nodes inside the shortest path area, while the second condition assures that one of the shortest paths will be selected by directing the algorithm to extend the found partial-path only toward the destination node.

The obtained minimum cost is then compared to the weight of the requesting flow. If the cost is less than the flow weight, the following actions should be made: 1) store the path of the new VIP, 2) calculate the T_{vip} and T_{ps} for the new VIP based on the communication volume of the requesting flow, 3) determine the overlapping VIPs which must be torn down, and 4) update PG with the new VIP configurations.

I. Tearing Down Conflicting VIPs and Setting up New VIPs

Once a new VIP is constructed, we have to set up it in the data network and destroy old VIPs which have at least one overlapping link with the new VIP. The path of the new VIP, as well as the path of the VIPs to be destroyed, is returned by Dijkstra's algorithm. In this step, the root node first sends a message over the setup network to all setup network nodes within the path of the VIPs to be destroyed containing the identifier of the port which must be released. The ports of

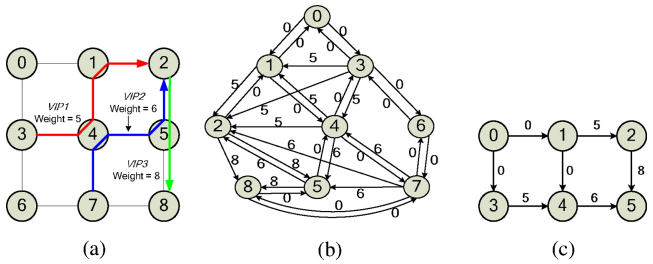


Fig. 4. (a) Sample VIP configuration, (b) its corresponding *PG*, and (c) subset of *PG* nodes and edges involved in constructing a VIP between nodes 0 and 5.

data network nodes are released by their corresponding setup network nodes by clearing their associated VIP allocators. When this message is delivered to the source node of a VIP, it resets the VIP flag associated to the traffic flow and directs the traffic through the packet-switched network, hereafter. The VIP is actually destroyed when all intransient data of the VIP is delivered to the destination node.

Then the new VIP is set up by sending a message to the setup network nodes along the VIP path. The message contains the identifier of the output port assigned to the VIP and the T_{vip} and T_{ps} values. The setup network then sets up the new VIP in the data network by configuring the VIP allocators in the nodes along the path and setting a flag at the source node indicating that the traffic of the corresponding flow should be directed through the established VIP.

VI. Experimental Results

In this section, we first evaluate the impact of static VIPs on power and performance of NoCs under some multicore SoC benchmarks. We then provide the simulation result for dynamic VIP construction scheme under some synthetic and realistic traffic loads.

A. Static VIPs

To evaluate the performance of static VIPs, we perform simulations with some benchmark application task-graphs. The benchmark set includes some existing SoC designs which have been widely used in the literature: multiwindow display (MWD) [9] with 12 cores, video object plane decoder (VOPD) [8] with 16 cores, global system for mobile (GSM) encoder [34], and H.263 encoder (H.263) [35]. Different tasks of H.263 encoder and GSM encoder are mapped onto 12 and 16 cores, respectively, and we set their input rates so that to scale the rate of their communication flows to the order of several mega bits per second.

In addition to the conventional five-stage routers mentioned before, two more efficient NoC designs are selected to be compared to the performance of our proposed VIP-enabled NoC. The first design speculatively performs virtual-channel allocation (VA) and switch allocation (SA) in parallel resulting in four-stage routers (including BW, route computation, SA + VA, and ST stages). In this scheme, the routers speculate that a waiting packet will succeed in output VC allocation stage. Non-speculative requests (flits which have already been allocated a VC) are prioritized over speculative ones to avoid performance degradation may be caused by unsuccessful speculations. The second NoC architecture further reduces the pipeline depth to three stages (including BW, SA+VA, and ST stages) by applying look-ahead routing, whereby the route computation for the current node is performed in the previous router. Since VIPs cover most of the on-chip traffic, reducing the router pipeline stages of the proposed VIP-enabled NoCs will not result in considerable performance improvement, hence our proposal employs the popular five-stage routers.

We have evaluated the proposed NoC architecture using Xmulator, a fully parameterized simulator for interconnection networks [36]. The Orion power library [37] is integrated to our simulator to calculate the power consumption of the networks.

In the simulations, the packet-switched part of our proposed NoC and the considered conventional NoCs use the same parameters: 64-bit wide flits, 2 VCs per port, 16-flit deep buffers, and 8-flit packets. The power results reported by Orion are based on an NoC in 65 nm technology and the working frequency of the NoC is set to 150 MHz. In simulation experiments, packets are generated with exponential distribution and the communication rates between any two nodes are set to be proportional to the communication volume between them in the task-graph. This task-graph-based traffic generation approach is introduced and used in [38]. Mapping and VIP construction in VIP-enabled NoCs are performed by the algorithm explained in Section IV, while in all of the conventional designs the mapping and route generation is performed by the baseline NMAP algorithm [8].

B. Impact of VIPs on Performance and Power Consumption

Fig. 5 compares the average packet latency of a VIP-enabled NoC with different conventional NoC architectures for the selected applications. As shown in the figure, VIPs consistently provide the lowest latency for different benchmarks compared to other considered NoCs. We see that as the pipeline is shortened, the overall performance improvement lessens; but it should be noted that while speculative allocation can shorten pipelines, they only work at very low network loads, lengthening the router pipeline instead at moderate to high loads when speculation frequently fails. The VIPs, however, can work consistently in all network situations. Fig. 5 also exhibits the power consumption of the considered NoCs. As the figure indicates, the power results follow the same trend as the latency results, with VIPs outperforming other NoCs. Obviously, the reduction in dynamic power is higher than the reduction in total power (dynamic + leakage), since VIPs do not improve the leakage power.

In summary, the proposed VIPs clearly outperform the other three NoC designs across all benchmarks, providing an average improvement of 36% in latency and 12% in power consumption over conventional NoCs with five-stage pipelined routers, 27% in latency and 20% in power over NoCs with four-stage pipelined routers, and 20% in latency and 23% in power over NoCs with three-stage pipelined routers.

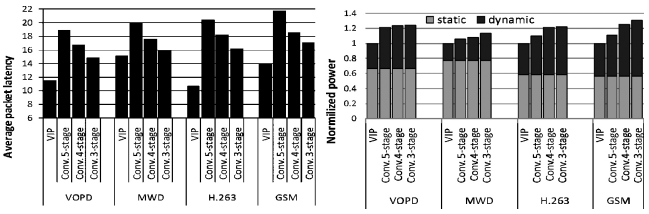


Fig. 5. Average packet latency (cycles for 8-flit packets) and normalized power consumption (with respect to the VIP power) of different NoC architectures

C. Impact of VIPs on NoC Adaption

We now evaluate the ability of VIP-enabled NoCs to adapt to a new application with different traffic pattern by keeping the mapping fixed and changing the on-chip traffic pattern over the one which is used for application-to-NoC mapping.

The H.263 encoder is a member of the multimedia system (MMS) benchmark suit [35] which also contains three other programs, H.263 decoder, MP3 encoder, and MP3 decoder. These applications use the same set of intellectual property (IP) cores. Here, we assume that an NoC whose mapping is performed for H.263 encoder is employed to run the other three MMS applications. For the GSM benchmark, we assume that a GSM decoder application is run on an NoC whose mapping is performed for GSM encoder. Similarly, for the first two SoCs which have a single CTG, we synthesize two additional CTGs for each of them, called X-60% and X-30%, where X is the name of the application. The X-30% and X-60% task-graphs are generated by replacing the source and destination nodes of the edges of task-graph X with other randomly chosen nodes (i.e., moving the edges of the base task-graph to other positions) with a probability of 30% and 60%, respectively. The NoC mapping for these synthetic applications are carried out based on the baseline application. In all of the benchmarks, the packets are forwarded using X-Y routing scheme. The other simulation parameters are set the same as the previous section.

Table I shows the average message latency for different applications. For comparison, Table I also reports the average packet latency for the case where mapping is performed for each individual benchmark. The latency values are normalized with respect to the latency of VIP-enabled NoC to gain a better understanding of the results. As can be seen in the table, VIPs not only outperform the other equivalent NoC designs, but also yield better latency than the NoCs with mapping optimized for the current application in most considered cases. The reason is that the latency increase due to longer packet path is completely compensated by shorter per hop latency.

For the MWD and VOPD benchmarks, when the difference between the synthetic applications and the original ones is increased, more power and performance gains are obtained. This is due to the fact that in these cases, the mapping is performed for an application whose traffic pattern is different (by 30% and 60% in our experiments) from the traffic pattern of the current application, resulting in more latency and power consumption. However, since VIPs can virtually connect the source and destination nodes of communication flows, the increase in average packet latency of the VIP-enabled NoCs, when changing the on-chip traffic, is relatively small and remains within 7% (on average) of the latency of the baseline application (based on which the mapping is done).

As mentioned before, the contribution of the static power in the total power consumption of VIPs remains the same as the conventional NoCs, hence the improvement in total power consumption is less than the improvement obtained for the latency and dynamic power. As Table I indicates, VIPs reduce the total NoC power consumption by 20%, on average, over other NoCs.

D. Dynamic VIPs: Synthetic Traffic Results

As discussed in Section I, most of the realistic CMP workloads exhibit a high degree of temporal and spatial communication locality, so we now use a synthetic traffic with similar characteristics to commercial and scientific CMP workloads. To this goal, we introduce three synthetic traffic patterns to evaluate the effects of the VIP connections on the CMP workloads. In the first pattern, 1-hot flow traffic pattern, each node sends 80% of the generated packets to exactly one destination node and the remaining 20% to other randomly chosen nodes. The other two traffic patterns, 2-hot flow and 3-hot flow traffic patterns, each node sends 20% of the generated packets equally randomly to the network nodes; in 2-hot traffic pattern, the remaining 80% of the traffic of each source node is sent to exactly two other nodes (40% to each destination node), while in 3-hot flow traffic pattern, this 80% traffic generated by each node is destined to three specific destination nodes equally. The favored destination nodes of each source node are selected randomly.

Each simulation runs for 5 000 000 cycles. The randomly-selected favored destination nodes of each source node are changed every 1 000 000 cycles to evaluate the ability of dynamic VIPs to adapt to dynamic traffic patterns. The setup network initiates the VIP construction procedure every 500 000 cycles. Since the hot flows are chosen randomly, we repeat each simulation run 10 times and every point in the plots represents the average of values obtained in each run.

The results are compared against a conventional NoC with speculative four-stage pipelined routers and a NoC employing dynamic EVC. The $l_{\max}$ of the dynamic EVC is set to 2 [4]. In addition to the VC devoted to bypass the routers, the EVC and VIP enabled routers use two VCs per port. The conventional NoC has 2 VCs per port, as well. For all NoCs, simulations are performed for a 64-bit wide system, speculative four-stage pipelined routers, 16-flit deep buffers, and 8-flit packets. The power results reported by Orion are based on an NoC implemented in 65 nm technology and the working frequency of the NoC is set to 2 GHz.

1) *Performance Results:* Fig. 6(a)–(c) shows the average packet latency of the proposed NoC and other NoC designs as a function of the injection rate (packet per node per cycle) under hot-flow synthetic traffics in a 6×6 mesh NoC. As shown in the figure, the proposed VIPs improve the latency of the NoC compared to other considered NoC designs under all considered traffic patterns. The reduction in latency over the conventional NoC before the saturation point of the conventional NoC (injection rate = 0.02) is 43% for

TABLE I
TOTAL POWER CONSUMPTION AND AVERAGE PACKET LATENCY OF THE VIP-ENABLED NOC AND OTHER CONSIDERED NOCS

	Baseline-Application Mapping							Optimized Mapping					
		Conv. Five-Stage		Conv. Four-Stage		Conv. Three-Stage		Conv. Five-Stage		Conv. Four-Stage		Conv. Three-Stage	
	VIP	Power	Latency	Power	Latency	Power	Latency	Power	Latency	Power	Latency	Power	Latency
VOPD-30%	1	1.38	1.69	1.44	1.47	1.48	1.38	1.08	1.33	1.09	1.18	1.2	1.02
VOPD-60%	1	1.58	1.84	1.58	1.6	1.61	1.6	1.08	1.35	1.1	1.19	1.16	1.08
MWD-30%	1	1.19	1.56	1.23	1.37	1.24	1.18	1.09	1.5	1.12	1.33	1.18	1.13
MWD-60%	1	1.15	1.72	1.15	1.5	1.15	1.29	0.99	1.57	1.11	1.38	1.17	1.2
H263 dec.	1	1.31	2.05	1.32	1.72	1.32	1.35	0.86	1.15	1.06	1.03	1.07	0.99
MP3 enc.	1	1.38	2.74	1.44	2.26	1.48	1.83	1.08	1.19	1.09	1.07	1.2	0.9
MP3 dec.	1	1.58	1.89	1.58	1.64	1.61	1.39	1.08	1.1	1.1	0.98	1.16	0.87
GSM dec.	1	1.19	1.68	1.23	1.41	1.24	1.36	1.09	1.28	1.12	1.09	1.18	1.01

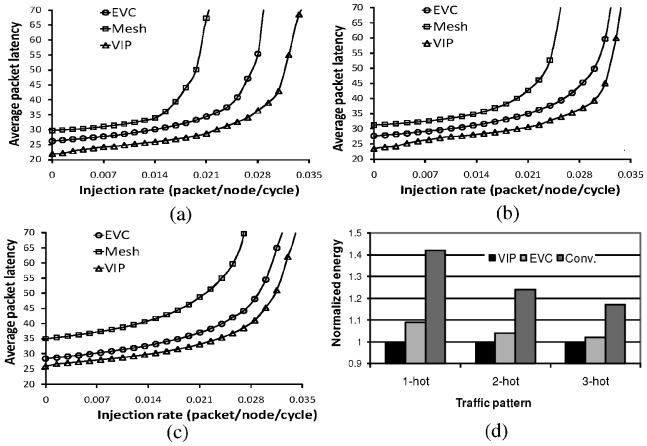


Fig. 6. Average packet latency (cycles for 8-flit packets) under (a) 1-hot flow, (b) 2-hot flow, and (c) 3-hot flow traffic loads, and (d) their normalized energy in a 6×6 NoC.

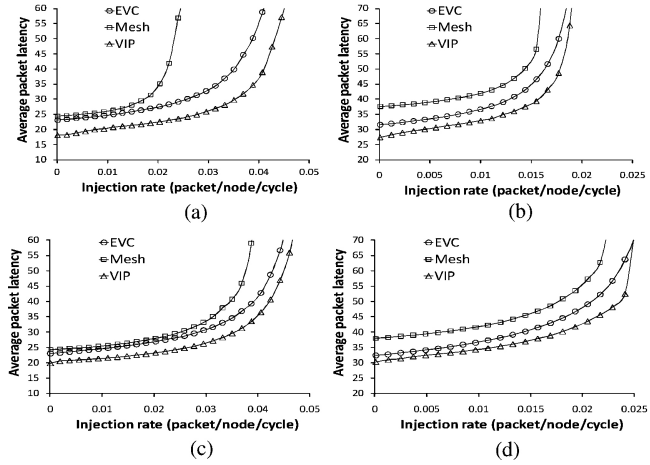
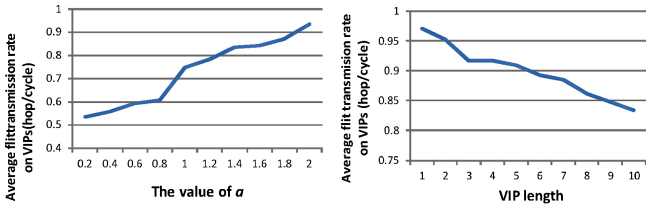


Fig. 7. Average packet latency (cycles for 8-flit packets) under (a) 1-hot flow in a 4×4 NoC, (b) 1-hot flow in a 8×8 NoC, (c) 3-hot flow in a 4×4 NoC, and (d) 3-hot flow in a 8×8 NoC.

1-hot flow, 31% for 2-hot flow, and 28% for 3-hot flow traffic. Compared to EVCs, the obtained reduction is 16%, 13%, and 6% under the 1-hot flow, 2-hot flow, and 3-hot flow traffic, respectively. Moreover, the VIPs significantly push the NoC throughput by around 30% over the conventional NoC. It also gives slightly better throughput than EVCs (up to 8% under 1-hot flow traffic). However, as we move from 1-hot flow traffic to 3-hot flow, the effect of the proposed approach reduces. This is because the number of high-volume connections is increased while the network resources that can be used by VIP links are fixed. Thus, the NoC cannot provide a VIP connection for some traffic flows and larger portion of the traffic is directed through the packet-switched network. However, even in 3-hot traffic pattern, the VIP connections can handle a significant portion of on-chip traffic and improve the performance over the two other considered NoCs.

2) *Energy Results:* Fig. 6(d) plots the energy consumption (in terms of energy per flit) of the three NoC designs at a traffic load just before the saturation point of the conventional NoC. The selected injection rate is 0.02 for all traffics. The numbers in the figure are normalized to the energy consumption of the VIPs. The energy of the VIP includes the NoC energy and the energy related to the VIP construction process and NoC traffic monitoring. We see from the figure that VIPs and EVCs outperform the conventional NoC. This is mainly due to the fact that EVCs and VIPs forward a significant portion of the on-chip traffic over short-cut paths, thereby the power-hungry buffering operations are removed.

3) *Sensitivity to the NoC Size:* To investigate the effect of the NoC size on the power and performance, we repeat the experiments for two different NoC sizes: 4×4, and 8×8. Fig. 7 reports the average packet latency of the VIPs as a function of the packet injection rate compared to EVCs and the conventional NoC for different network sizes. The results are reported for 1-hot and 3-hot flow traffic patterns. The figure shows that the VIPs perform very well under 1-hot flow traffic in all NoC sizes. Under 3-hot flow traffics, however, although the VIPs continue to deliver better performance than the EVCs and conventional NoCs, the gap between EVCs and VIPs reduces, especially for the 8×8 NoC. The reason is that in larger NoC sizes, the average length of VIPs increase. This situation, in turn, increases the conflict among VIPs and therefore, the number of flows covered by VIPs reduces. From the results reported in Figs. 6 and 7, we can conclude that traffic pattern has a first order impact on the VIP performance, i.e., under the same traffic pattern, VIPs deliver a relatively consistent improvement over the other methods for all of the considered NoC sizes. VIPs are advantageous when the on-chip traffic is grouped into several communication flows. They give better gains when the number of such flows decreases.



(a) (b)

Fig. 8. (a) Sensitivity of data communication rate on VIPs to a. (b) VIP length.

4) *Sensitivity to the Flow Control Parameters:* Now, we investigate the effects of the proposed flow-control mechanism presented in Section III-B on the NoC performance. We select a 6×6 NoC under 2-hot flow traffic pattern for this purpose. We first evaluate the effect of the parameter a (in T_{vip} defined in Section III-B) on the average flit transmission rate (hops per cycle) traveling on VIPs. In order to assess the effects of flow control mechanism in the simulation experiments, the packets are generated according to a self-similar process with a Hurst parameter of $h = 0.8$ to generate bursty traffic on VIP connections. Fig. 8(a) shows the average flit transmission rate on VIPs (hops per cycle) for different values of a. As can be seen in the figure, larger values of a give higher flit transmission rates on VIPs.

This parameter can be set appropriately (based on the degree of burstiness of the traffic or bandwidth requirement of the application) when a certain degree of quality of service should be provided for a given application. In this paper, we have focused on reducing the average energy and latency of the NoC. Extensive exploration of the effects of flow control mechanism parameters on VIP enabled NoCs is left for future work, when considering quality of service issues.

Since control signals of the flow control mechanism travel one hop per cycle, the proposed end-to-end on/off mechanism may require large buffers at the downstream node of VIPs, when the channel latency is high or the VIP is long (long VIPs can be formed in NoCs with large diameters). However, since the VIPs are constructed between the source and destination nodes of a traffic flow, they are directly connected to the buffers of the network interface at the destination node, which generally has larger buffering space than the router ports. This larger buffering space is mainly due to the need for reordering the packets of a single message [24]. Furthermore, as the data in the network interface buffers are consumed by the processing cores, the buffers are very unlikely to incur overflow (thus, changing the state of *on/off* signals), if the operating frequency of the cores are higher than the frequency of the NoC.

Fig. 8(b) plots the average flit transmission rate on VIPs in a 6×6 NoC, as a function of VIP length. Here, the statistics for all VIPs of a certain length are averaged. The value of a is set to 1 and the buffer depth at the network interface is set to 64 flits (8 packets). Again, the traffic is generated according to a self-similar process with Hurst parameter $h = 0.8$. As shown in this figure, the resulted curve is almost linear revealing that the VIP length has a slight impact on flit transmission rate over VIPs. Therefore, we can conclude that the impact of the used flow control mechanism on the performance of long VIPs (that can be formed in NoCs with large diameters) is negligible.

E. Dynamic VIPs: Stanford Parallel Applications for Shared-Memory (SPLASH) Results

Now, we evaluate the effectiveness of the proposed VIPs under the traffic traces generated from SPLASH-2 benchmark suit [39]. The parameters of the three NoCs are set the same as the previous section. We set the period of VIP construction procedure for each benchmark separately in such a way that the procedure is invoked 10 times during the execution of the benchmark. Fig. 9 compares the latency and energy obtained by VIPs, EVCs, and conventional NoCs across eight SPLASH-2 traces, on a 7×7 NoC. The results are normalized to the results given by VIPs. On average, VIPs outperform the EVCs by 5% and the conventional NoC by 34% when considering the packet latency. For the programs like *WATER-Nsquard*, *OCEAN*, and *WATER-Spatial,* where the traffic is distributed rather evenly across the nodes, the EVCs provide better improvement than the VIPs over the conventional NoCs.

The VIPs, on the other hand, outperform the EVCs when a large portion of the packets are transmitted along several (preferably small number of) high-volume traffic flows. This traffic pattern, as shown before, can be efficiently handled by the VIPs. The *RADIX* traffic is an example of such traffic pattern for which the VIPs give the largest improvement, with 27% reduction in latency over EVCs and 64% over the conventional NoC. Following the same trend, the energy consumption offered by VIPs outperforms EVCs by 2% and conventional NoCs by 11%, on average.

1) *Impact of the VIP Weight Threshold:* In these experiments, the VIP weight threshold is set to $4 \times W_{\text{avr}}$, where W_{avr} is the average weight of all of the NoC communication flows. Obviously, by decreasing the threshold, more communication flows will be eligible to request a VIP. However, more eligible flows will not necessarily lead to more VIPs. The main reason is that, being saturated by VIPs with higher rates, the NoC may not have enough free resources to accept all of the new requests with lower weights. Table II outlines the average number of VIP construction requests the root node receives at each round of VIP reconfiguration for two different threshold values. As shown in this table, increasing the VIP weight threshold to $4 \times W_{\text{avr}}$ results in a significant decrease in number of requesting flows, whereas the loss in the percentage of the traffic covered by VIPs is small. Further increase in the threshold, however, brings about more than 10% drop in average traffic coverage, hence setting the threshold to $4 \times W_{\text{avr}}$ results in a more appropriate trade-off between the VIP performance and the number of requests to be handled by the dynamic VIP construction procedure.

2) *Impact of the Frequency of Updating VIPs Configuration:* The frequency of invoking the procedure of the dynamic VIP reconfiguration is another important parameter in the proposed method. In general, by increasing the frequency, the VIPs will better adapt to the on-chip traffic pattern, but at the price of more activity of the setup network. Fig. 10 plots the increase in the amount of the traffic covered by VIPs when the frequency of the dynamic VIP reconfiguration is

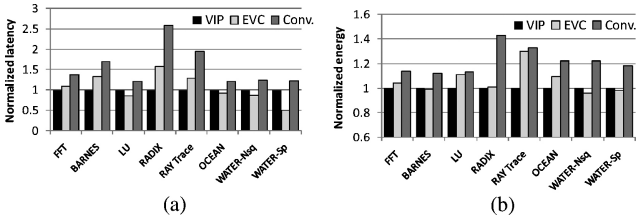


Fig. 9. (a) Normalized average message latency and (b) normalized energy consumption in the conventional, EVC-based, and VIP-based NoCs for SPLASH-2 programs.

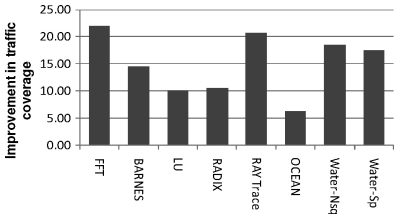


Fig. 10. Increase in the amount of the traffic covered by VIPs when the frequency of the VIP construction is doubled.

TABLE II
AVERAGE NUMBER OF VIP CONSTRUCTION REQUEST FOR DIFFERENT VIP WEIGHT THRESHOLDS

Benchmark	Threshold = T	Threshold = $4 \times T$	Coverage loss (%)
FFT	266	58	7.8
BARNES	69	42	1.3
LU	293	12	11.2
RADIX	40	18	0.4
RAYTrace	46	14	0.9
OCEAN	211	82	6
WATER-Nsquard	347	58	9.6
WATER-Spatial	276	34	11.9

doubled (from 10 to 20 calls during the benchmark execution time). The programs with high degree of temporal traffic variation benefit from higher frequencies. For the programs with relatively unchanged traffic pattern during the simulation time (like *OCEAN*), on the other hand, a low frequency is more appropriate. Clearly, the suitable frequency depends on the program behavior; so, for the future work we intend to develop a mechanism for setting the frequency at run-time.

F. Dynamic VIPs: Multicore SoC Results

To evaluate the effect of the proposed dynamic VIP construction mechanism on multicore SoCs, we use the MMS programs, namely H.263 encoder, H.263 decoder, MP3 encoder, and MP3 decoder. These benchmarks use the same cores and we run them on a 4×3 mesh-based NoC one at a time in consecutive periods with packets generated with exponential distribution and the communication rates between any two nodes are set to be proportional to the communication volume between them in the task-graph. The cores are mapped using the baseline NMAP based on a CTG generated by integrating the CTG of all four benchmarks into a single CTG.

The results are compared to a conventional NoC with four-stage pipelined routers using X-Y routing. The conventional NoC uses the same mapping as the VIP-enabled one. Both NoCs work at 150 MHz with 64-bit wide flits, 2 VCs per port, 16-flit deep buffers, and 8-flit packets. The same parameters are set for the packet-switched part of the VIP-enabled NoC. In this experiment, it is assumed that it is not possible to know in advance the exact sequence in which different applications may come into the system. The running application is changed randomly (every 1 000 000 cycles, on average) and the VIP construction procedure initiates every 500 000 cycles. The results of realizing the proposed dynamic VIP construction method for the considered applications during 10 000 000 cycles of execution, with application being randomly changed during the simulation show 18% reduction in power consumption and 33% reduction in the average message latency (from 27.41 cycles to 18.22 cycles). It means that the dynamic VIP construction scheme can adapt the NoC to the on-chip traffic pattern for multicore SoC programs as well, and its results are comparable to the results offered by the static VIP construction scheme.

VII. CONCLUSION

In this paper, we presented a packet-switched router architecture that can results in lower power consumption and packet latency by dedicating VIP connections between the source and destination nodes of heavy communication flows. This is achieved by means of a subset of VCs which bypass the pipeline of the intermediate routers along the path. Amongst different potential applications of the VIP connections, we focused on power and latency reduction and developed an application-specific NoC design methodology which statically constructs VIPs for an application. Simulation results using some multicore SoC benchmarks showed that, compared to some conventional NoC designs, the proposed architecture noticeably improves the NoC performance and power consumption. We also addressed run-time construction of VIPs using a light-weight setup network. Since in multicore SoCs and CMPs, each node tends to have a small number of favored destinations for the messages it sends, most of the on-chip traffic is directed through the VIP connections and considerable power and performance improvement can be achieved. Simulating the proposed NoC architecture using synthetic and realistic benchmarks with different NoC parameters confirmed such improvements and showed that the VIPs outperform the two efficient methods reported in the literature, i.e., EVCs and speculative pipelined routers.

REFERENCES

[1] J. Owens, W. J. Dally, R. Ho, D. N. Jayasimha, S. W. Keckler, and L. S. Peh, "Research challenges for on-chip interconnection networks," *IEEE Micro*, vol. 27, no. 5, pp. 96–108, Sep.–Oct. 2007.

[2] L. Benini and G. De Micheli, "Networks on chip: A new paradigm for systems on chip design," *IEEE Comput.*, vol. 35, no. 1, pp. 70–78, Jan. 2001.

[3] Y. Hoskote, S. Vangal, A. Singh, N. Bokar, and S. Bokar, "A 5-GHz mesh interconnect for a teraflops processor," *IEEE Micro*, vol. 27, no. 5, pp. 51–61, Sep. 2007.

[4] A. Kumar, L. S. Peh, P. Kundu, and N. K. Jha, "Express virtual channels: Toward the ideal interconnection fabric," in *Proc. Int. Symp. Comput. Architect. (ISCA)*, 2007, pp. 150–161.

[5] H. G. Lee, N. Chang, Ü. Y. Ogras, and R. Marculescu, "On-chip communication architecture exploration: A quantitative evaluation of point-to-point, bus, and network-on-chip approaches," *ACM Trans. Design Automat. Electron. Syst.*, vol. 12, no. 3, pp. 1–20, Aug. 2007.

[6] A. Pullini, F. Angiolini, S. Murali, D. Atienza, G. Micheli, and L. Benini, "Bringing NoCs to 65 nm," *IEEE Micro*, vol. 27, no. 5, pp. 75–85, Sep.–Oct. 2007.

[7] F. Angiolini, L. Benini, P. Meloni, L. Raffo, and S. Carta, "Contrasting a NoC and a traditional interconnect fabric with layout awareness," in *Proc. Design, Automat. Test Eur. (DATE)*, 2006, pp. 1–6.
[8] S. Murali and G. De Micheli, "Bandwidth-constrained mapping of cores onto NoC architectures," in *Proc. Design, Automat. Test Eur. (DATE)*, 2004, pp. 896–901.
[9] K. Srinvasan and K. Chatha, "A low complexity heuristic for design of custom network-on-chip architectures," in *Proc. Design, Automat. Test Eur. (DATE)*, 2006, pp. 130–135.
[10] Z. Ding, R. Hoare, A. Jones, D. Li, S. Shao, S. Tung, J. Zheng, and R. Melhem, "Switch design to enable predictive multiplexed switching in multiprocessor networks," in *Proc. Int. Parallel Distrib. Process. Symp. (IPDPS)*, 2005, pp. 100.1–100.10.
[11] E. Bilir, R. Dickson, Y. Hu, M. Plakal, D. Sorin, M. Hill, and D. Wood, "Multicast snooping: A new coherence method using a multicast address network," in *Proc. Int. Symp. Comput. Architect. (ISCA)*, 1999, p. 270.
[12] N. E. Jerger, M. Lipasti, and P. Li-Shiuan, "Circuit-switched coherence," *IEEE Comput. Architect. Lett.*, vol. 6, no. 1, pp. 5–8, Jan. 2007.
[13] M. Modarressi, H. Sarbazi-Azad, and A. Tavakkol, "Low-power and high-performance on-chip communication using Virtual Point-to-Point connections," in *Proc. IEEE/ACM Int. Symp. Netw. Chip*, 2009, pp. 203–213.
[14] M. Modarressi, H. Sarbazi-Azad, and A. Tavakkol, "Virtual point-to-point links in packet-switched NoCs," in *Proc. IEEE Comput. Soc. Symp. Very Large Scale Integration*, 2008, pp. 433–437.
[15] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in *Proc. Design, Automat. Test Eur. (DATE)*, 2000, pp. 250–256.
[16] A. Jantsch and H. Tenhunen, *Network on Chip*, Norwell, MA: Kluwer, 2003.
[17] A. Jalabert, S. Murali, L. Benini, and G. De Micheli, "XpipesCompiler: A tool for instantiating application specific networks-on-chip," in *Proc. Design, Automat. Test Eur. (DATE)*, 2004, pp. 884–889.
[18] T. Bjerregaard and J. Sparsø, "A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip," in *Proc. Design, Automat. Test Eur. (DATE)*, 2005, pp. 1226–1231.
[19] X. Ru, J. Dielissen, C. Svensson, and K. Goossens, "Synchronous latency-insensitive design in æthereal NoC," in *Proc. Future Interconnects Netw. Chip Workshop Design, Automat. Test Eur. (DATE)*, Mar. 2006, pp. 22–27.
[20] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch, "Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network-on-chip," in *Proc. Design, Automat. Test Eur. (DATE)*, 2004, pp. 890–895.
[21] M. Modarressi, H. Sarbazi-Azad, and M. Arjomand, "An SDM-based hybrid packet-circuit-switched on-chip network," in *Proc. Design, Automat. Test Eur. (DATE)*, 2009.
[22] U. Ogras and R. Marculescu, "Application-specific network-on-chip architecture customization via long-range link insertion," in *Proc. Design Automat. Conf. (DAC)*, 2005.
[23] M. Stensgaard and J. Sparsø, "ReNoC: A network-on-chip architecture with reconfigurable topology," in *Proc. Int. Symp. Netw. Chip*, 2008, pp. 55–64.
[24] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Network*, San Mateo, CA: Morgan Kaufmann, 2004.
[25] P. T. Wolkotte, G. J. M. Smit, G. K. Rauwerda, and L. T. Smit, "An energy-efficient reconfigurable circuit switched network-on-chip," in *Proc. Reconfig. Architect. Workshop*, 2005, pp. 155–162.
[26] K. Goossens, J. Dielissen, and A. Radulescu, "The æthereal network on chip: Concepts, architectures, and implementations," *IEEE Design Test Comput.*, vol. 22, no. 5, pp. 414–421, Sep.–Oct. 2005.
[27] P. Gratz, B. Grot, and S. W. Keckler, "Regional congestion awareness for load balance in networks-on-chip," in *Proc. High Performance Comput. Architect. (HPCA)*, 2008, pp. 203–214.
[28] M. A. Al Faruque, R. Krist, and J. Henkel, "ADAM: Run-time agent-based distributed application mapping for on-chip communication," in *Proc. Design Automat. Conf. (DAC)*, 2008, pp. 760–765.
[29] M. Faruque, T. Ebi, and J. Henkel, "Run-time adaptive on-chip communication scheme," in *Proc. Int. Conf. Comput. Aided Design (ICCAD)*, 2007, pp. 26–31.
[30] L. Smit, G. Smit, J. Hurink, H. Broersma, D. Paulusma, and P. Wolkotte, "Run-time mapping of applications to a heterogeneous reconfigurable tiled system on chip architecture," in *Proc. Int. Conf. Field Programmable Logic Applicat. (FPL)*, 2004, pp. 421–424.
[31] V. Nollet, T. Marescaux, P. Avasare, D. Verkest, and J.-Y. Mignolet, "Centralized run-time resource management in a network-on-chip containing reconfigurable hardware tiles," in *Proc. Design, Automat. Test Eur. (DATE)*, 2005, pp. 234–239.
[32] C. Chou, U. Y. Ogras, and R. Marculescu, "Energy and performance-aware incremental mapping for networks-on-chip with multiple voltage levels," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 10, pp. 1866–1879, Oct. 2008.
[33] M. Modarressi, A. Tavakkol, and H. Sarbazi-Azad, "VIP paths in NoCs," Dept. Comput. Eng., Sharif Univ. Technol., Tehran, Iran, Tech. Rep. TR-HPCAN10-1, 2010.
[34] M. Schmitz, "Energy minimization techniques for distributed embedded systems," Ph.D. dissertation, School Electron. Comput. Sci., Univ. Southampton, Southampton, U.K., 2003.
[35] J. Hu and R. Marculescu, "Energy and performance-aware mapping for regular NoC architectures," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 4, pp. 551–562, Apr. 2005.
[36] *Xmulator Simulator*, 2010 [Online]. Available: http://www.xmulator.org
[37] A. Kahng, B. Li, L. Peh, and K. Samadi, "ORION 2.0: A fast and accurate NoC power and area model for early-stage design space exploration," in *Proc. Design, Automat. Test Eur. (DATE)*, 2009, pp. 423–428.
[38] J. Hu and R. Marculescu, "Application specific buffer space allocation for networks on chip router design," in *Proc. Int. Conf. Comput. Aided Design (ICCAD)*, 2004, pp. 354–351.
[39] *SPLASH-2*, 2008 [Online]. Available: http://www.flash.stanford.edu/apps/SPLASH/

Mehdi Modarressi (S'08) received the B.S. degree from Amirkabir University of Technology, Tehran, Iran, in 2003, and the M.S. degree from Sharif University of Technology (SUT), Tehran, Iran, in 2005, both in computer engineering. He is currently working toward the Ph.D. degree from the Department of Computer Engineering, SUT.

He is visiting the Parallel Systems Architecture Laboratory, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, from Jan. 2010 till Aug. 2010. His current research interests include different aspects of high-performance computing and computer-aided design with a particular emphasis on network-on-chip architectures.

Arash Tavakkol received the B.S. and M.S. degrees in computer engineering from Sharif University of Technology, Tehran, Iran, in 2005 and 2008, respectively.

Since 2008, he has been with Supercomputing Center, Institute for Research in Fundamental Sciences, Tehran, Iran. His current research interests include network-on-chips, interconnection networks, and performance evaluation.

Hamid Sarbazi-Azad received the B.S. degree in electrical and computer engineering from Shahid-Beheshti University, Tehran, Iran, in 1992, the M.S. degree in computer engineering from Sharif University of Technology (SUT), Tehran, Iran, in 1994, and the Ph.D. degree in computing science from the University of Glasgow, Glasgow, U.K., in 2002.

He joined the Department of Computer Engineering, SUT, in 2002, where he is currently an Associate Professor. Since 2003, he heads the School of Computer Science, Institute for Research in Fundamental Sciences, Tehran, Iran. His current research interests include high-performance computer architectures, network-on-chips and system-on-chips, parallel and distributed systems, performance modeling/evaluation, graph theory and combinatorics, and wireless/mobile networks. He has published more than 250 refereed conference and journal papers on these topics.

Dr. Sarbazi-Azad received the Khwarizmi International Award in 2006, the TWAS Young Scientist Award in Engineering Sciences in 2007, and became a Distinguished Researcher of SUT in 2004, 2007, and 2008. He has served as the Editor-in-Chief of the *CSI Journal on Computer Science and Engineering* since 2006. He has been an Editorial Board Member, and the Guest Editor for several special issues on high-performance computing architectures and networks in related journals.