Application-aware virtual paths insertion for NOCs

Majed ValadBeigi, Farshad Safaei *, Bahareh Pourshirazi

Faculty of ECE, Shahid Beheshti University G.C., Evin, 1983963113 Tehran, Iran

ABSTRACT

Network-on-chip (NoC) has rapidly become a promising alternative for complex system-on-chip architectures including recent multicore architectures. Additionally, optimizing NoC architectures with respect to different design objectives that are suitable for a particular application domain is crucial for achieving high-performance and energy-efficient customized solutions. Despite the fact that many researches have provided various solutions for different aspects of NoCs design, a comprehensive NoCs system solution has not emerged yet. This paper presents a novel methodology to provide a solution for complex on-chip communication problems to reduce power, latency and area overhead. Our proposed NoC communication architecture is based on setting up virtual source-destination paths between selected pairs of NoCs cores so that the packets belonging to distance nodes in the network can bypass intermediate routers while traveling through these virtual paths. In this scheme, the paths are constructed for an application based on its task-graph at the design time. After that, the run time scheduling mechanism is applied to improve the buffer management, virtual channel and switch allocation schemes and hence, the constructed paths are optimized dynamically. Moreover, in our design the router complexity and its overheads are reduced. Additionally, the suggested router has been implemented on Xilinx Virtex-5 FPGA family. The evaluation results captured by SPLASH-2 benchmark suite reveal that in comparison with the conventional NoC router, the proposed router takes 25% and 53% reduction in latency and energy, respectively besides 3.5% area overhead. Indeed, our experimental results demonstrate a significant reduction in the average packet latency and total power consumption with negligible area overhead.

© 2014 Elsevier Ltd. All rights reserved.

1. Introduction

System complexity in modern designs seems to make ASICs infeasible for myriad of applications according to their high development cost and time. Therefore, System-on-Chip (SoC) architecture has been introduced as a promising solution for many applications so that those SoC platforms include a large number of different IP-blocks communicating with each other. However, due to the increasing number of on-chip components and thus higher bandwidth demands on such large SoC platforms, IP blocks cannot be connected using point to point wired connections or shared buses. Hence, those communication mechanisms have been replaced by a flexible communication infrastructure in the form of a Network-on-Chip (NoC) [1–3].

In [4], three different interconnection fabrics have been compared to each other and it is demonstrated that NoC is faster, more predictable and more scalable than the two other options which are an AMBA AHB bus [4] and a $5 \times 5$ crossbar. However, an NoC must be flexible enough to be able to support a wide range of applications with different bandwidth demands and Quality of Service (QoS). Currently, this flexibility is provided by large packet-switched NoCs with an over-engineered total bandwidth capacity.

The use of an NoC to replace bus-based wiring has reached the advantages of better scalability at the architectural and physical levels, better performance under high loads, better decoupling of protocol-level and transport-level issues in the communication protocol stack, quicker design closure, more freedom in the design, more customizability and more streamlined design flows[5–8]. Moreover, in [9], some comparisons have been carried out between various on-chip communication schemes and the results show that NoCs are the most predictable mechanism in terms of the difference between the estimated and actual frequency.

There is no doubt that designing an efficient NoC architecture, while satisfying the application performance constraints are a complex process. The design issues span several abstraction levels, ranging from high-level application modeling to physical layout level implementation. The most important phases in designing NoCs are analyzing and characterizing application traffic, synthesizing the NoC topology for the application, mapping and binding
of the cores with the NoC components, finding paths for the traffic flows and reserving resources across the NoC, choosing NoC architectural parameters, such as the data width of the links, buffer sizes and frequency of operation, verifying the designed NoC for correctness and performance, building simulation, synthesis and emulation models for the NoC. Despite the fact that many researchers [10–16] have provided various solutions for different aspects of on-chip design, a comprehensive NoC system solution has not emerged yet.

In this paper we propose a novel flow control and microarchitecture design which tries to reduce the delay, energy and area overhead. The proposed method sets up the virtual paths between every two nodes by allowing the packets to bypass the intermediate routers along their paths. In this scheme, the connections are constructed for an application based on its task-graph at the design time. In this step, we propose an optimum tree construction algorithm and mapping mechanism to construct the paths and assign the priority according to the communication flow to each node. Further, we also try to optimize these connections by using a run time scheduling approach. In this step, the mechanism of buffer management, virtual channel allocation and switch allocation are optimized. In the proposed scheduling approach, the packets which belong to their nodes are observed based on their node priority. Then, a new substrate is presented based on a new priority ID to enable application-aware ranking in an NoC by coordinating the scheduling decisions made by the routers. By triggering this scheme we ensure that the packets use the network resources appropriately and thus a significant reduction in the power, latency and area overhead is achieved.

It is worthy to mention that our scheme includes techniques that ensure starvation freedom, resulting in a dynamic reconfigurable substrate that enables application-aware prioritization in on-chip networks. In this approach, a limited number of network packets periodically are grouped into batches. Each router ranks packets belonging to older batches, thereby ensuring that no application starves due to interference from higher-ranked applications. It should be noted that overlapping of multiple paths is allowed since all overlapping paths use network resources in their batches’ group.

The rest of this paper is organized as follows. Section 2 surveys the related work. Section 3 introduces the proposed router architecture. Section 4 presents the evaluation results, and finally Section 5 concludes the paper.

2. Related work

Packet-based Network-on-Chips (NoCs) are envisioned to be the solution for connecting tens to hundreds of components in a future many-core processor executing hundreds of tasks. In a packet-switched NoC architecture, in order to improve the bandwidth, application requirements are simultaneously satisfied by sharing physical channels between multiple flows. To achieve this end, messages are packetized and packets are sent on a hop by hop basis. However, since the links are shared, packets need to go through a complex router in a pipeline fashion to compete for output links at each intermediate router. Therefore, in comparison with point to point communications, in a packet-switched NoC, performance is degraded because of the increase in router complexity and thereby increasing communication latency and power consumption [14,15].

In order to improve the performance in packet-switched NoCs, several approaches have been developed throughout recent years. Most of these studies, such as EVCs [14] and long range links [16] have focused on benefiting from a second switching mechanism for lowering the power consumption and latency. Authors of [14] have proposed a method in which packets can bypass intermediate routers by using some virtual paths called EVCs. However, EVCs can only connect nodes along a single dimension and cannot turn from one dimension to another. In [16], inserting long-range links for application-specific grid-based NoC architecture has been presented. In this approach, performance of NoCs has been shown to improve by inserting of the long-range links following principles of small world graphs. Additionally, a methodology to reduce the average packet hop count by using the physical bypass paths is introduced in [17]. The authors illustrate that their approach decreases the NoC power consumption with 10% area overhead.

Circuit switching [18] is a mechanism which has been introduced to reduce communication latency. This switching method works based on constructing a pre-reserved circuit between two connecting nodes prior to their communication. Therefore, arbitration and routing are not needed after setting up the circuit. However, this method degrades the performance both in terms of latency by imposing an extra circuit setup time to the network, and an inefficient resource usage. Consequently, in some recent researches, benefiting from the properties of two switching methods (circuit and packet switching) has been proposed. The approaches which are based on both circuit switching and packet switching are proposed in [19–23]. These methods are applied in order to efficiently manage streaming and best-effort traffics generated by real-time applications. The techniques are used in circuit-switched sub-network to raise path diversity, thus improving the throughput and low resource utilization.

In contrast to the previous work, our approach uses optimization in both design and run time and thus significant reduction in latency and energy are accomplished. Moreover, the virtual paths aim to exploit the on-chip traffic locality and completely connect the source to the destination of a communication rather than providing shortcut paths. Further, in this mechanism, by inserting just a few numbers of application-specific virtual paths, traffic workloads can be significantly decreased. It is worthy to mention that unlike circuit switching our mechanism does not work based on time slot reservation and therefore does not accompany the complex slot allocation procedure and poor resource utilization of circuit switching. Besides, in the proposed virtual path construction mechanism of our scheme, the long setup time of other approaches such as circuit switching is reduced. Additionally, our approach reduces a number of hop counts by bypassing the intermediates router along the virtual paths and as will be shown in evaluation results, it gives significant improvements in energy and latency with negligible area overhead compared to above mentioned approaches. It is worthy to mention that applying the benefits of the run time optimization in the suggested reconfigurable method leads to optimize constructed paths dynamically and therefore outstanding improvements in the performance of the network are achieved.

It should be noted that in [24], it has been observed that storing a packet in a buffer consumes far more energy than transmitting it. Therefore, in our scheduling mechanism, we provide an approach to dynamically manage the buffer for transmitting packets appropriately and therefore more power saving can be achieved.

3. The proposed NoC architecture

In this section we present the proposed NoC architecture. To do so, first, system model and some basic assumptions are described. Second, we explain the proposed methodology and finally, the presented architecture is clarified.

3.1. System model and basic assumptions

The system of interest consists of a set of nodes interconnected by a mesh network. The nodes of the network (referred to as PEs)
are populated with processing and/or storage elements that communicate with each other via the network. Moreover, in this paper, we employ the conventional five-stage pipelined router [18] which is illustrated in Fig. 1 and 2 as a baseline NoC router. The router has input and output ports corresponding to the four neighboring directions and the local processing element (PE) port. The major components, which constitute the router, are the input buffers, routing computation (RC), virtual channel allocator (VA), switch allocator (SA) and crossbar switch.

Given the very limited buffer space in resource-constrained on-chip networks, routers tend to consider wormhole flow control, which relaxes the constraints on buffer size as compared to store-and-forward and virtual cut through.

In order to remove the serialization delay, we have benefited from the lookahead routing (LA) [25]. The route of a packet is determined one hop in advance, thereby enabling flits to compete for VCs immediately after the buffer write stage.

3.2. The proposed methodology

The proposed methodology is exploited to improve the power and performance metrics of the NoC when running a specific target application. To reach this, the high volume traffics are selected and virtual paths are reserved between their sources and destination nodes, if there are sufficient free resources in the network. It is worthy to mention that by assigning the high volume traffics through the virtual paths a large number of packets can take advantage of the low-power and low-latency communication provided by these paths. The key point in this approach is that the optimization processes are applied in both the design and run time.

In the design time, for a given application, our objective is:

(a) The tree construction algorithm is applied based on the determined costs for each communication flow.
(b) Physically map the cores of the application into different nodes of a mesh-connected NoC.
(c) Establish as many virtual paths as possible for the communication flow of the application.

Additionally, the following operations are triggered in the run time.

(a) The flit scheduling mechanism is preformed to manage the buffer while a new flit is arrived.
(b) In order to remove the starvation of low ranked applications, the batching scheme is applied.
(c) To optimize the VA and SA, the scheduling approach is triggered.

3.2.1. The design time optimization

In order to prioritize the nodes for a given application, our proposed mechanism performs a primary ranking in the design time. This ranking is determined by using the tree construction algorithm. In this algorithm each input application is described as a communication task-graph which is depicted by WDG. WDG is a weighted directed graph \( G(V, E) \), where each vertex \( v_i \) represents a task, and a directed edge \( e_{ij} \in E \) characterizes the communication edge from \( v_i \) to \( v_j \). The communication volume corresponding to an edge \( e_{ij} \) is also provided and is denoted by \( C(e_{ij}) \). Besides, the number of edges attached to each node are defined by \( \lambda(v_i) \).

This parameter helps us to consider a proper priority for each node for assigning it to one of the end points of a virtual path. To clarify, the following steps describe the proposed methodology processes in the design time.

**STEP 1:** In this step, a cost \( P(j) \) is assigned to each node determining its priority. This quantity can be computed as

\[
P(j) = \sum_{i} C(e_{ij}) \times \lambda(v_i)^{1/2} \quad \forall i, j, p, q \in V
\]

It should be noted that Eq. (1) shows the normalized value of the cumulative communication volume of the incoming edges connected to a node.
STEP 2: In this step, the optimized tree prescribing the nodes priorities is constructed. This tree is built by selecting the node \( v_m \) with maximum \( P(m) \) as the root node. Then, the rest of nodes can be chosen in an ascending order based on their cost for expanding the tree. This process is illustrated in Fig. 1. The tree finds the best results to map the cores appropriately. This approach finds the best mapping obtained by pair-wise swapping of vertices, invoking Eq. (1). The worst-case computational complexity of the entire algorithm is \( O(EV) \).

STEP 3: In this step, in order to map the core wisely, the tree nodes are labeled based on their priorities. Furthermore, the tree edges are directed to their higher priority end node. Also, the proper tree links for benefiting from virtual paths are selected based on nodes priority and their label. Fig. 3b shows the constructed tree for the sample graph which is depicted in Fig. 3a.

STEP 4: In this step, the nodes of the optimized tree are mapped into the mesh network. To do so, the root node of the optimized tree (the node with the maximum communication volume) is mapped onto one of the mesh nodes with maximum number of neighbors. Afterwards, the tree nodes connected to the already mapped tree nodes are selected based on their communication volume. The core mapping is accomplished by modifying NMAP, a simple and fast heuristic power-aware core mapping and route generation method [26], by considering the \( T(S,E) \) optimized tree from tree construction algorithm, as follows.

1. Cluster the \( T(S,E) \) vertices into two sets: \( Mapped \) (which contains the already mapped \( T(S,E) \) vertices and is initially empty) and \( Unmapped \) (which includes the \( T(S,E) \) vertices not yet mapped and initially includes all \( T(S,E) \) vertices). Since a vertex is mapped it is moved to the \( Mapped \) set.
2. Map the core with the lowest priority label (the root node of the optimized tree) onto one of the mesh nodes with maximum number of neighbors. The priority label of a node is determined by the tree construction algorithm which is illustrated in Fig. 3b.
3. Repeat the following steps until the \( Unmapped \) set become empty (i.e., all nodes are mapped).
   i. Select the core \( v_i \) that has a connected edge in the \( T(S,E) \) with the \( Mapped \) members that has the lowest priority compared to the other cores.
   ii. Examine all \( Unmapped \) mesh nodes adjacent to the nodes of \( T(S,E) \) for placing it.
      a. Select the \( T(S,E) \) edges between the selected core \( v_i \) and the \( Mapped \) member in order of their priority label.
      b. Examine the possibility of constructing a virtual path for edge along the one of the shortest path between its source and destination nodes. Map the core with low priority to be directed on virtual paths. More precisely select the node with maximum \( P(m) \).

Moreover, after finding a route for all \( T(S,E) \) edges, virtual paths are established by using the routing table of the router. Fig. 3c displays the mapping mechanism and virtual path construction in a \( 3 \times 3 \) mesh for a sample graph. It should be noted that constructing virtual paths based on \( T(S,E) \) of an application on an NoC for a given mapping suffer from a problem. In this case the physical placement of the cores is fixed and the problem is to find optimum virtual paths for the \( T(S,E) \)edges during the run time. This problem can be solved by the scheduling optimization which is described in the next step in order to optimize the constructed paths dynamically.

3.2.2 The run time optimization

In this section, we are trying to provide an efficient mechanism to improve the process of buffer management, VA and SA in order to reduce the power and improve the performance. By providing this mechanism the flit scheduling is performed appropriately and no flit suffer much time in the buffer and it is served immediately. The main contribution of this mechanism is based on transmitting flits immediately instead of keeping them in the buffer and thus more power saving is achieved. We will discuss about these improvements in the evaluation results.

In this scheme, we define a Control Decision Block (CDB) to manage the incoming flits. CDB includes two components which are referred to as CDB control table and Central Decision Logic (CDL). The former is employed to keep the flits information while the latter is used to periodically gather the information from each incoming flit and store them in the CDB control table. Table 1 illustrates the definition of the flit information and their functions.
It is worthy to mention that CDB periodically updates its control table in each cycle.

As we mentioned in the table, for each incoming flit in the buffer, we consider some information in the CDB control table. The Flit-Num is obtained according to the flit location in the buffer while the Priority is assigned based on the priority number of a node which flit is sent from. We also rank each flit to avoid starvation based on its determined Batch-ID. Suppose a flit wants to go to the specific VC, in each cycle this VC is assigned to the new flit and thus the mentioned flit should wait much time to go to the VC. To prevent this starvation, we propose an approach to rank each flit based on the batching mechanism which is described in STEP 5. Additionally, RPI is a Relative Priority Identifier to rank each flit in order to assign a suitable VC for it which is explained in STEP 6. The VC-ID determination is obtained based on RPI of each flit and it is described in STEP 7. The detailed description of the proposed optimization approach is as follow.

**STEP 5:** In this step we introduce the mechanism of batching in order to prevent the starvation of low ranked packets. In this approach, network packets are grouped into finite-size batches. A packet belonging to an older batch is always prioritized over a packet in a younger batch. As a result, reordering of packets across batches is not possible within a router. In this approach, batching starts a new batch whenever a certain number of N flits have been injected into the current batch. That is, with batching, the first N injected packets are assigned the batch number 0, the next N packets the batch number 1, and so on. The batch number is depicted by batch IDs. By assigning this approach, we consider the packet with old-batch ID and we ensure starvation-freedom.

**STEP 6:** We define a Relative Priority Identifier (RPI) to each packet in order to rank them wisely. This value is computed as

\[ RPI = P \times \text{BatchID} \]  

in which, \( P \) is the priority of the packet which is calculated as mentioned in STEP 1. Each packet with a lowest \( RPI \) has a high priority. Note that if two arbitrary packets have the same \( RPI \), we choose the one that has a lower \( P \). Once \( N \) new flits have been injected, CDB notices the network routers to increment their batch IDs. At each input port, a router maintains a pool of free flit buffers. When a new flit arrives, CDB stores the flit information in its control table and computes \( RPI \) of it. In each cycle, CDB checks its table and updates the information which belongs to the flits.

**STEP 7:** In this step, VCs are assigned based on the proposed VA method. In this mechanism two sets of VCs are considered. The former one with 1-flit buffer is depicted by High Priority VC (HPVC), serves as a VC which is responsible for carrying higher priority packets through a single hop while the latter, which is denoted as Regular Priority VC (RPVC) is used by the rest of the packets. HPVC has a signal which is set to 1 when it has incoming flit to service. It uses this signal to direct the flit in its buffer to the crossbar. On the other hand, when the HPVC buffer is empty, RPVCs are selected in an ordered way based on its RPI in the CDB table and the outcome of RC, VC, and SA just like the traditional packet-switched network.

### Table 1
A summary of CDB control table information.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flit-Num</td>
<td>The flit number for tracking it in the buffer</td>
</tr>
<tr>
<td>Priority</td>
<td>The determined priority for each flit</td>
</tr>
<tr>
<td>Batch-ID</td>
<td>The flit ranks in the buffer</td>
</tr>
<tr>
<td>RPI</td>
<td>The relative priority for each flit</td>
</tr>
<tr>
<td>VC-ID</td>
<td>The type of VC which is assigned to each flit</td>
</tr>
</tbody>
</table>

3.3. The NoC architecture

Fig. 4 shows the microarchitecture of the proposed router. As shown in the figure, one VC buffer in each physical channel is replaced by 1-flit buffer which is depicted by HPVC. These VCs are used for establishing the virtual paths between any given nodes. Moreover, Fig. 5 illustrates CDB mechanism and its control table. CDB is applied to dynamically manage the arriving flits at each input port. When a new flit arrives, CDB sends the buffer write signal to DeMux-1 and assign this flit into the buffer pool.

In CDB, CDL is triggered to periodically gather information from each incoming flit and store the information of the flit in CDB control table. It is also determine the \( RPI \) for each flit based on its priority and \( \text{batch-ID} \). Additionally, in order to implement the batching mechanism CDB at each input port keeps a local copy of the \( \text{batch-ID} \) register containing the current (injection) batch number. Note that \( \text{batch-ID} \) is updated after every \( N \) flits that have been injected. Moreover, a new \( RPI \) is computed at each cycle. For this purpose, CDB at each input port has additional logic and a set of hardware counters to measure the ranking metric value that will be used by CDL to compute \( RPI \) of the flits. The counters are reset at the beginning of each cycle. At the end of each cycle, CDB
forms the CDB control table containing the flits metric value (s) measured during that cycle.

Further, by using the obtained information from CDB control table, CDB assigns the VC-ID to the flit based on its RPI. If the VC-ID is HPVC, it sends buffer read signal to Mux-1 to let the flit traverse to HPVC. On the other hand, if the VC-ID is RPVC, the flits move to one of the RVC’s VCs based on the ascending order of their RPI. Note that CDB uses Flt-Num from its table for tracking the buffer assigned to all flits corresponding to the different input VCs.

While HPVC has incoming flit to service, it sets its signal to 1. This signal is used to connect one of the VCs to the crossbar input. When this signal is set to 1, the select signal of Mux-3 is set to 0 and Mux-2 directs HPVC to the crossbar input. On the other hand, if HPVC signal is set to 0, the flits which are located RPVC’s VCs traverse the router stage based on CDB control table and outcome of RC and SA. It is just like the traditional packet-switched network as we mentioned above.

It should be noted that in this design no modifications need to be made in switch crossbar design.

4. Evaluation

In this section we evaluate the proposed mechanism. To do so, we first present the evaluation methodology. Then, we briefly describe the traffic patterns used for comparison purposes. Finally, results and analysis are drawn.

4.1. Evaluation methodology

To evaluate the proposed on-chip communication scheme, we have implemented the NoC architecture using Xmulator, a fully parameterized simulator for interconnection networks [27]. The simulator is augmented with Orion power library [28] to calculate the power consumption of the networks. Further, in Orion library the process feature size and working frequency of the NoC is set to 65 nm and 150 MHz, respectively. Table 2 summarizes the common router features and network parameters for synthesis and simulation. We have also performed a large number of simulation experiments in order to make the evaluation results independent of the relative positions of the faults.

Moreover, the determined routers were developed in structural VHDL code and synthesized using Xilinx ISE toolkit. The platform used for implementing the proposed methodology is Xilinx Virtex-5 FPGA family.

4.2. Traffic patterns

We consider two different traffic patterns when evaluating the network behavior: synthetic patterns and traces [18]. Synthetic patterns are widely used because they allow evaluating the network in the most generic way. When we use them, every node has the same traffic injection rate. We evaluate the complete range of traffic injection rate, from low levels up to saturation point. The synthetic traffic patterns used are Hotspot [18].

- For hotspot, 90% of the sources (selected randomly) inject traffic to the same destination (selected randomly), the rest of end nodes inject traffic to random destinations. This traffic pattern allows to model the situation when one or more end nodes are frequently accessed by the remaining end nodes (a disk server, for instance).

On the other hand, to assess the performance of the proposed mechanism, we perform simulations with some benchmark application task-graphs. The benchmark suite includes some existing SoC designs which have been widely used in the literature: multiwindow display (MWD) with 12 cores, video object plane decoder (VOPD) with 16 cores, H.263 encoder (H.263) and MP3. These benchmarks employ the same cores and they have been run on a 4 x 4 mesh-based NoC. The task-graph-based traffic generation approach is introduced and used in [29].

Moreover, traces are based on capturing the traffic when running real applications. Traces contain the source, destination, injection time and the size of the transmitted packets. They allow obtaining results in more realistic scenarios and let us compare them with the results obtained by the synthetic patterns. The traces are collected from the execution of FFT, LU, BARNES, RADIX, OCEAN, RAYTRACE WATER-NSQUARED and WATER-SPATIAL applications from SPLASH-2 [30] suite in shared-memory multiprocessors. These types of applications are widely used when simulating multiprocessor systems on engineering and scientific computations.

4.3. Results and analysis

In this section, we first compare our proposed router with conventional NoC router [18] as a baseline router to find the considerable understating of the NoC improvements in our approach by considering different evaluation parameters. Then the proposed router is compared with EVC [14] and conventional NoC router [18] by studying various benchmarks to show the significant reduction in power and latency in our design compared to the other approaches.

Fig. 6 displays the average packet latency of the conventional NoC architecture and the proposed NoC architecture under hotspot traffic for different traffic injection rates in a 4 x 4 mesh network. As seen in the figure, although our methodology establishes scheduling mechanism in the run time and it consumes time to perform this operation, our approach provides the lower latency compared to the conventional NoC. Besides, according to Fig. 7, although our design adds extra module like CDB to the architecture, its efficient operations reduce the total power consumption. Further, as the number of favored destinations of a source node increases the effect of the proposed approach decreases. The reason is that the number of high-volume connections is increased

<table>
<thead>
<tr>
<th>Topology</th>
<th># of Ports</th>
<th># of VCs</th>
<th>Flit size</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 x 4 Mesh</td>
<td>5</td>
<td>4</td>
<td>128-bit</td>
</tr>
<tr>
<td>Routing</td>
<td>Packet length</td>
<td>Buffer size</td>
<td>Link latency</td>
</tr>
<tr>
<td>Look-ahead</td>
<td>8-Flit</td>
<td>16-Flit</td>
<td>1 cycle</td>
</tr>
</tbody>
</table>

Please cite this article as: M. ValadBeigi, et al., Application-aware virtual paths insertion for NOCs, Microelectron. J (2014), http://dx.doi.org/10.1016/j.mejo.2014.02.010
The proposed mechanism

while the network resources that can be used by the virtual paths are fixed. Therefore, more power saving is achieved.

Figs. 8 and 9 compare the latency and power consumption obtained by the presented NoC with the conventional NoC architecture across eight SPLASH-2 traces, on a 7 x 7 mesh. The results are normalized to the results given by the proposed approach. On average, our approach outperforms the conventional NoC by 25% considering the packet latency. It also provides a superior achievement in the total power consumption by 53% improvement. It is worthy to mention that managing the buffer in the proposed approach, which relies on packet transmission instead of consuming, has significant impacts on reducing the power consumption.

Moreover, Table 3 compares the energy of various components in the presented NoC with those of the conventional NoC architecture across eight SPLASH-2 traces, on a 7 x 7 mesh. On average, our approach outperforms the conventional NoC by 15% considering the total energy components. We also compare the energy of the each component used in the presented methodology and conventional NoC. As can be seen from the table, it is evident that CDB unit imposes a minimal energy on the overall design, while providing full operations to manage the buffers and optimize VA/SA preformation. It should be noted that by providing CDB control decision logic and control table, the VA and SA mechanisms are optimized and thus, they consume minimal energy in comparison with the VA and SA of the conventional NoC router.

Further, Table 4 illustrates the average number of available flits in the input buffer for different injection rates for the hotspot traffic. As we mentioned in the methodology description, our approach is based on transforming flits appropriately instead of consuming them. As the injection rates of the network traffic increase, the number of flits in the network raised. Hence, the number of access requests to the nodes (input buffers) also increases. Our approach handles these huge numbers of flits efficiently and no saturation, which happens when the number of requests to access node (buffer) is higher than the available free buffers, will occur. To clarify, as can be seen in the table, the determined size for keeping flits in the input buffer is more than the obtained buffer usage in the simulation parts. Therefore, the results prove that the suggested design transmits the packet efficiently and by considering each injection rate a few number of flits remain in the buffer. Hence, low hardware and more power saving is achieved. Moreover, the energy reduction percentages of similar components in both the proposed method and conventional NoC are presented. As depicted in the table due to input buffer, SA and VA components, the proposed methodology experience a significant reduction in terms of energy in those elements compared to conventional NoC.

Table 5 also provides accurate area analyzes based on the implemented design on FPGA Virtex-5. We compare our proposed design with conventional NoC router. As can be seen in the table, the extra logic in the proposed router takes only 3.5% more area overhead than the conventional NoC router. However, in comparison to the area of EVC, the proposed approach area is 2.5% less than the EVC design.

In this step, we compare our design with EVC and the conventional NoC under different application specific benchmarks to reveal the significant improvements of our methodology in comparison with other alternatives. Fig. 10 compares the latency of the proposed NoC with the conventional and EVC-based NoC architectures for the selected applications. As shown in the figure, the proposed approach consistently provides the lowest latency for different benchmarks compared to other considered NoCs. The latency values are normalized with respect to the latency of the proposed NoC to gain a better understanding of the results. For the MWD and VOPD benchmarks, when the difference between the

Please cite this article as: M. ValadBeigi, et al., Application-aware virtual paths insertion for NOCs, Microelectron. J (2014), http://dx.doi.org/10.1016/j.mejo.2014.02.010
synthetic applications and the original ones is increased, more latency gains are acquired. 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. In summary, the proposed method clearly outperforms the other two NoC designs across all benchmarks, providing an average improvement of 38% and 12% in terms of latency over the conventional NoC and EVC-based NoC, respectively. These achievements are related to optimization in both the design and run time and also provide a profitable path to connect the source and destination with optimized scheduling mechanism in each router.

Fig. 11 also exhibits the power consumption of the considered NoCs. The results are normalized to the results given by the proposed NoC. As the figure indicates, the power results follow the same trend as the latency results, with the proposed method outperforming other NoCs. On average, the proposed NoC outperforms the conventional NoC by 54% and EVC-based NoC by 43% when considering the power consumption.

To sum up, Table 6 illustrates the conclusions we can extract from the performed evaluations. As an overall conclusion it can be seen that in terms of latency and power, our proposed approach has superior performance over the conventional NoC and EVC-based NoC.

Table 3 Components energy of the proposed methodology (TPM) and the conventional NoC (Conv.) for splash-2 programs. The energy reduction percentage for similar components are also presented.

<table>
<thead>
<tr>
<th>Components</th>
<th>Energy (µJ)</th>
<th>TPM</th>
<th>CONV.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input buffer</td>
<td>0.321</td>
<td>0.153</td>
<td>0.132</td>
</tr>
<tr>
<td>CDB</td>
<td></td>
<td>0.154</td>
<td>0.145</td>
</tr>
<tr>
<td>VA</td>
<td></td>
<td>0.146</td>
<td>0.141</td>
</tr>
<tr>
<td>SA</td>
<td></td>
<td>0.134</td>
<td>0.138</td>
</tr>
</tbody>
</table>

Table 4 The average number of available flits in the input buffer with different injection rates.

<table>
<thead>
<tr>
<th>Injection rate</th>
<th>0.01</th>
<th>0.02</th>
<th>0.03</th>
<th>0.04</th>
<th>0.05</th>
<th>0.06</th>
<th>0.07</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input buffer</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 5 FPGA resource usage for the proposed methodology (TPM), the conventional NoC (Conv.) and EVC based NoC.

<table>
<thead>
<tr>
<th>Logic utilization</th>
<th>Conv.</th>
<th>TPM</th>
<th>EVC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of slice registers</td>
<td>11428</td>
<td>11775</td>
<td>11.905</td>
</tr>
<tr>
<td>Number of slice LUTs</td>
<td>7968</td>
<td>8436</td>
<td>8685</td>
</tr>
<tr>
<td>Number of fully used LUT-FF pairs</td>
<td>4928</td>
<td>5068</td>
<td>5224</td>
</tr>
<tr>
<td>Number of bonded IOBs</td>
<td>644</td>
<td>652</td>
<td>669</td>
</tr>
<tr>
<td>Number of BUFG/BUFGCTRLs</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Overall</td>
<td>3.5</td>
<td>2.5</td>
<td>2.6</td>
</tr>
</tbody>
</table>

Fig. 10. Normalized latency in the conventional, EVC-based and the proposed NoC for different benchmarks.

Fig. 11. Normalized power consumption in the conventional, EVC-based and the proposed NoC for different benchmarks.

5. Conclusions

In this paper, a novel flow control mechanism and router microarchitecture design has been presented. It has shown that the power consumption and packet latency is reduced by dedicating virtual paths between the source and destination to allow the packet to bypass the node along their paths. In this mechanism two optimization processes are applied in the both design and run time. Afterwards, amongst different potential applications of the virtual paths, we focused on developed an application-specific NoC design methodology to reduce the power, latency and area overhead.

The presented design is evaluated with various workloads in the different network size and also is implemented on Xilinx.
Virtex-5 FPGA family. Evaluation results captured by SPLASH-2 benchmark suite show that the proposed router performs well by taking 25% and 53% reduction in latency and energy, respectively besides 3.5% area overhead based on the implemented design on FPGA Virtex-5. It is also provides a significant level of configurability with lower hardware overhead, latency and energy compared to alternative designs.

In the future, we have plans to present an analytical model for 2-D mesh networks to analyze accurate power and latency. Moreover, we want to extend the present work to include inserting virtual paths for different objective functions such as fault-tolerance and QoS criteria.

References