An Efficient Dynamically Reconfigurable On-chip Network Architecture

Mehdi Modarressi[1,2], Hamid Sarbazi-Azad[1,2], Arash Tavakkol[2]
[1]Computer Eng. Dept., Sharif University of Technology, Tehran, Iran
[2]IPM School of Computer Science, Tehran, Iran
modarressi@ce.sharif.edu, azad@sharif.edu, arasht@ipm.ir

ABSTRACT

In this paper, we present a reconfigurable architecture for network-on-chips (NoC) on which arbitrary application-specific topologies can be implemented. The proposed NoC can dynamically tailor its topology to the traffic pattern of different applications at run time. The run time topology construction is realized through a light-weight control network. It involves monitoring the network traffic in order to detect heavy communication flows and configuring the topology in order to reduce the hop count between their source and destination nodes, while the NoC connectivity is preserved. In this paper, we first introduce the proposed reconfigurable topology and then address the problem of run-time topology reconfiguration. Experimental results show that this architecture effectively improves the performance of NoCs and reduces the power consumption over the existing conventional NoCs.

Categories and Subject Descriptors

B.4.3 [**INPUT/OUTPUT AND DATA COMMUNICATIONS**]: Interconnections (Subsystems)—*topology*

General Terms

Design

Keywords

NoC, performance, power, reconfigurable, topology.

1. INTRODUCTION

Application-specific optimization is one of the most effective approaches to improve power/performance metrics of Network-on-Chips (NoC) [1]. Customizing the network topology for a given application is an important application-specific NoC optimization method which dramatically affects the power consumption and average latency of NoCs, when running that application. Like other application-specific optimization methods, existing methods generate a customized topology and mapping for a single target application, provided that the application and its traffic characteristics are known at the design time [2]. However, several different applications may be integrated onto a single complex multicore System-on-Chips (SoC). In addition, NoC-based Chip Multiprocessors (CMPs) are generally applied to run different applications which may not be known at design time.

DAC'10, June 13–18, 2010, Anaheim, California, USA.
Copyright 2010 ACM 1-58113-000-0/00/0010…$10.00.

Generally, an NoC that is designed to run exactly one application does not necessarily meet the design constraints of the other applications. Furthermore, the traffic pattern of a single application may vary significantly over time.

In this paper, we address the need for dynamic topology adaption by introducing a reconfigurable NoC architecture which enables the network topology to dynamically match the communication pattern of the currently running application.

The reconfiguration of the proposed architecture is achieved by inserting several simple switches in the network which allows dynamically changing the inter-node connections and implementing the topology in which the number of intermediate routers between the source and destination nodes of high volume communication flows is reduced by bypassing as many intermediate routers as possible. This can lead to considerable performance improvement since the power/latency of the router's pipeline stages has a significant contribution to the total NoC power/latency.

Our early evaluation results using a static design-time topology generation method showed the effectiveness of this architecture in reducing NoC power consumption [3]. In this work, we show the ability of our reconfigurable NoC architecture in handling dynamic traffic patterns by developing a run-time topology reconfiguration mechanism. To guarantee the connectivity of the network when the topology is reconfigured, a set of links with fixed connections are used along with the reconfigurable links. This set of links, called the *fixed sub-network* (or Fnet), is used to provide permanent connections among nodes based on a given topology, e.g. mesh, here. The second set, called the *reconfigurable sub-network* (or Rnet), on the other hand, is devoted to establish application-specific connections between the nodes with high-volume communication demands. The Rnet is still a packet-switched network, but its topology is customized for the traffic flows travel over it.

To keep the cost of the proposed NoC equal to a traditional packet-switched one, the NoC links and resources are divided between the two sub-networks using *Spatial-Division Multiplexing* (SDM). Unlike *Time-Division Multiplexing* (TDM) where at each time slot, all the wires of a link are dedicated to transmission of data from a single source, the SDM technique allocates a sub-set of the link wires to a given circuit for the whole connection lifetime.

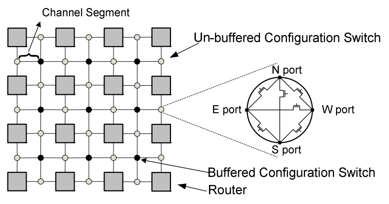
Improving the power and performance metrics of packet-switched NoCs by integrating a second switching mechanism has been addressed in several previous works [4][5][6]. For example, the long-range links in [6] are constructed to provide shortcut path between the source and destination nodes of high-volume communication flows at design-time. The reconfigurability is, however, the key advantage of our work over long-range links as it allows the shortcut paths to be constructed dynamically based on the traffic pattern at run-time. Besides, being established over conventional NoCs, our architecture still benefits from the predictability and reusability of regular NoC architectures.

In this paper, we first introduce the proposed NoC architecture and then propose a run-time topology reconfiguration algorithm and show its effectiveness in improving the NoC power and performance metrics.

2. THE PROPOSED NOC

2.1 Reconfiguration Mechanism

The system under consideration is composed of $m \times n$ nodes arranged as a 2D mesh network. In the proposed NoC architecture, however, the routers are not connected directly to each other, but connected through simple switch boxes, called *configuration switches* (see Figure 1). Each square in Figure 1 represents a network node which is composed of a processing element and a router, whereas each circle represents a configuration switch. Figure 1.a shows the internal structure of a configuration switch. It consists of some simple transistor switches that can establish connections between incoming and outgoing links. In this figure, for the sake of simplicity, only a single wire connection is depicted between each two ports of a configuration switch. However, there are two connections between each two ports of a configuration switch in order to route the incoming and outgoing sub-links of bidirectional links independently. Actually, the internal connections are implemented by a multiplexer at each output port of the switch. We refer interested readers to [3] and [7] for more details of this architecture, including its architectural features, area overhead, and reconfiguration latency.



Channel Segment
Un-buffered Configuration Switch
N port
E port
W port
S port
Buffered Configuration Switch
Router

Figure 1. The reconfigurable NoC architecture

2.2 Network Partitioning

As mentioned in Section 1, in this work, we propose to partition the NoC into two sub-networks using the SDM scheme. This proposal aims to benefit from the decreased average packet path length (hop count) of the reconfigurable sub-network, while preserve the network connectivity via the fixed sub-network.

Applying SDM in a packet-switched network allows having several links in parallel in the same direction and therefore increasing the number of distinct paths in that direction. This increase in the path diversity reduces the *head of line (HOL)* blocking and can improve the average latency and throughput of the network. On the other hand, splitting the links into two parts may increase the average packet latency as the number of flits of a packet is increased.

An in-depth analysis of the effect of SDM on the NoC latency and throughput in [8] shows that splitting the links into two sub-links increases the throughput by 50-60% with a negligible negative effect on the average message latency. Thus, SDM is a suitable bandwidth multiplexing scheme for our NoC, since in addition to throughput, by selectively employing the sub-networks, we can also improve the power and latency of the NoC.

Figure 2 illustrates the partitioning scheme, where network resources (links, buffers, crossbar, and configuration switches) in an *N*-bit wide NoC are divided into 2 parallel *N*/2-bit sub-networks. The first set of sub-links (the upper part) form Fnet and the other part is devoted to Rnet. The total buffering space in a router remains the same as that in the original conventional NoC. The crossbar switch structure is the same as in a conventional packet-switched router, but the switch allocator (arbiter) is divided into two separate Fnet and Rnet allocators.

Although the proposed NoC is not restricted to a specific switching scheme and routing algorithm, the NoC routers, in this study, adopt a conventional wormhole switching mechanism. However, the route computation and switch allocation schemes are adapted to the new features of the proposed architecture.

In this architecture, packets, on arriving at an input port, are buffered in R_Buff (in Figure 2) if they are arrived from Rnet, and in F_Buff, if they are arrived from Fnet. To route a packet, the router first checks if there exists a long link in the Rnet originating from the current node destined to some node along the route towards the packet's destination. As a packet is only allowed to use the shortest paths, routing logic checks the (at most two) output ports along the shortest paths toward the destination for Rnet links. Between the two Rnet ports, the one which allows more intermediate nodes to be skipped over is selected. In parallel, the default routing algorithm (x-y routing) is used to select the proper output port in Fnet. The header then requests for the selected Fnet and Rnet (both, if any) partitions of the corresponding output ports in the *switch allocation* stage of the router pipeline. On successful allocation of the requested Rnet port, the flit withdraws its Fnet request. If the flit wins more than one ports at the same cycle, it selects the Rnet port. In the current implementation, deadlock freedom on Rnet is guaranteed using the well-known *escape channel* method [9] which involves using at least two virtual channels per port.

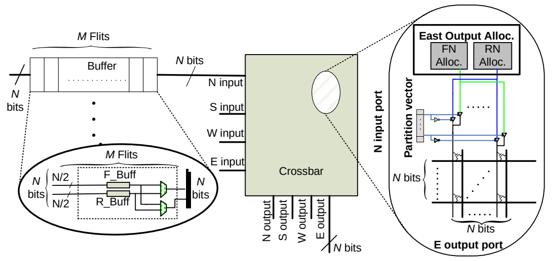


Figure 2. The proposed router architecture

Although this section assumes that the link-width is equally divided between the Fnet and Rnet sub-networks, this architecture is capable to assign different number of wires to each sub-network. This can be achieved by using a bit-vector register, called the *partition vector (PV),* which determines the bit-width assigned to each sub-network. Setting a bit in PV to 1 indicates that the corresponding bit in the NoC link is assigned to the Fnet, otherwise it belongs to the Rnet. PV is used in the network interface of each node to determine the flit width of packets. Setting the flit width to an appropriate value can be easily carried out by shift registers. Clearly, the flit width should be long enough to guarantee that the routing and other required information can be embedded into a single flit (the header flit). Moreover, PV is used at the crossbar to direct the flits to appropriate output port wires (Figure 2), according to the decision made by each arbiter (Fnet allocator and Rnet allocator).

Reconfigurability is the key point of the proposed partitioning method, in that the link-width of Fnet and Rnet can be changed at run-time by simply changing the bits of the PV register. The network can even be converted to a fully customized or fully fixed packet-switched network, based on the current traffic pattern.

3. TOPOLOGY RECONFIGURATION ALGORITHM

This section proposes a scheme for topology reconfiguration which dynamically changes the Rnet connections in response to a change in the current on-chip communication pattern. Here, it is assumed that each task is non-migratory and already mapped onto some node. The whole procedure relies on monitoring the traffic generated by each node in order to detect high-volume communication flows. A mechanism then selects the best Rnet path for the detected flows. The path must include only the routers and configuration switches along the shortest paths between the source and destination nodes of the requesting flow.

Finding a new topology using this procedure may take a few clock cycles. However, this procedure is done in parallel with normal network operation and packets do not wait for the new Rnet configuration to be completed; they continue traveling on the current Fnet and Rnet configuration. Therefore, this setup latency does not degrade the network performance.

Traffic Monitoring. Traffic monitoring is simply done at each node by storing the number and the destination node addresses of the packets the node sends. The weight of each flow is an *m*-bit value which is set to the *m* most-significant bits of the register holding the traffic volume, 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 length of a flow increases, the contribution of the flow in the entire on-chip traffic increases.

Once the weights are periodically updated at some specific times, each node sends the weight of its flows weighting higher than a threshold to a global arbiter. The threshold of each node is defined as the average communication volume of the flows originating from it. The arbiter then sorts the communication flows in order of their communication volume and tries to build a path with minimum cost for each flow in the order. We calculate the cost of a path as the cumulative cost of the routers and configuration switches it includes. According to the power and latency analysis presented in [3], we assign a cost of 1 to a configuration-switch and a cost of 5 to a router.

Initially, in the topology selection algorithm, the internal connections of all configuration switches are un-configured. Finding a route may involve configuring the switches which are not yet configured in order to bypass some intermediate routers and make a shorter connection between two given nodes. The algorithm can configure the un-configured internal connections of the configuration switches, but not the connections that have already been configured at previous iterations of the algorithm.

Path Selection Algorithm. We employ a simple and tiny set-up network (or control network) along with the main data network to configure the paths in the Rnet. The size of the setup network is the same as the size of the main data network. Each node in the setup network corresponds to a node (router or configuration switch) in the data network with the same address. Having a small bit-width and a simple internal structure, the area of the setup network is negligible, compared to the main data-network. It also consumes negligible power due to its infrequent activity and small bit-width. To calculate the minimum cost for an Rnet path for a flow, the cost of the path is propagated from the source node toward the destination node, along all of the shortest paths. In this way, initially, the source node sends its cost as the initial partial path cost to its neighboring nodes along the shortest path towards the destination. Each node (router or configuration switch) receives the partial path cost (the cost of the path from the source node to the current node), adds it to its cost, and propagates the result to the next nodes towards the destination, until the final cost is received at the destination node. If the current node is a router, it sends the cost to its neighboring configuration switches along the shortest paths. Similarly, configuration switches send the updated cost to appropriate neighboring routers or configuration switches. However, the current configuration switch may be already configured in previous steps of the algorithm, in such a way that the port through which the cost reaches the switch is connected to some output port. In this case, the cost is sent along the direction determined by the current switch configuration, provided that the direction is along the shortest path towards the destination. Starting from the source node, the cost values are forwarded to the destination node one hop per cycle. The intermediate nodes may receive two costs from the neighboring nodes at the same cycle. Each node selects the minimum cost for propagation (and ignores the other one.

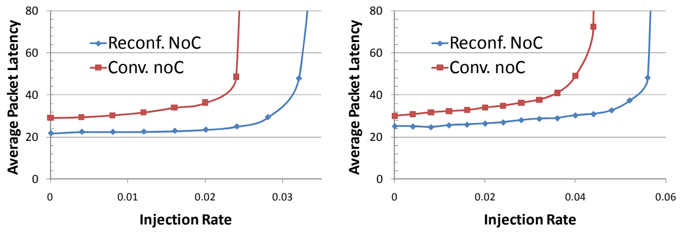


Figure 3. The average packet latency (cycles for 8-flit packets) for the proposed and conventional NoCs, under 1-hot flow (a), 2-hot flow (b), and 3-hot flow in a 6×6 NoC

A node also ignores a received cost, if the link between the current node and the node from which the cost is received does not have enough bandwidth to accept the requesting communication flow.

Each node keeps the track of the path with the minimum cost by storing the direction from which the minimum cost is received. Once the final cost reaches the destination node, indicating that an Rnet path with the minimum possible cost is found for the flow, the destination node sends back an acknowledgement signal toward the source node along this path. The connection is then established in the NoC by configuring all according configuration switches within the path. Afterwards, the algorithm continues with the next flow, until all requesting flows are serviced. Since the path exploration is carried out within the shortest path area, the worst case happens when the selected path (if any) includes the same number of routers as a shortest path in the Fnet.

4. TOPOLOGY RECONFIGURATION

4.1 Synthetic Traffic

As shown in [10], 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 use the three synthetic traffic patterns presented in [10], to evaluate the effects of the presented architecture on the CMP workloads. In the first pattern, the 1-*hot flow* traffic pattern, each node sends 80% of the generated packets to exactly one destination node and the remaining 20% packets to other randomly chosen nodes. In the other traffic pattern, the 3-*hot flow* traffic pattern, each node sends 20% of the generated packets equally randomly to the network nodes and the remaining 80% are sent to three specific nodes. The hot destination nodes of each source node are selected randomly from other network nodes.

Each simulation runs for 1,000,000 cycles. The randomly-selected hot destination nodes of each source node are changed every 200,000 cycles and the reconfiguration procedure is initiated every 100,000 cycles.

We have evaluated the proposed NoC architecture using Xmulator, a fully parameterized simulator for interconnection networks [11]. The Orion power library [12] integrated to Xmulator calculates the power consumption of the NoCs.

The results are compared against a conventional mesh NoC with speculative 4-stage pipelined routers employing an adaptive routing algorithm. For all NoCs, simulations are performed for a 128-bit wide system, speculative 4-stage pipelined routers, 2 virtual channels per port, 8-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 500 MHz. The link-width of the Rnet and Fnet are set to 96 and 32 bits, respectively.

Performance Results. Figure 3 shows the average packet latency of the proposed NoC and the conventional NoC 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 architecture improves the latency of the NoC. The reduction in latency over the conventional NoC (before the saturation point of the conventional NoC) is 28% for the 1-hot flow, and 13% for the 3-hot flow traffic. Moreover, our NoC significantly push the NoC throughput by around 25% over the conventional NoC. As we move from the *1-hot flow* traffic to the *3-hot flow*, increasing the number of favored destinations of each source node results in decreasing the advantages of the proposed approach. This is because the number of high-volume connections is increased while the network resources that can be used by long links are fixed. Thus, the Rnet cannot provide long connection for some traffic flows and larger portion of the traffic is directed through the packet-switched network.

Energy Results. We measure the energy consumption of the NoC in terms of energy per flit (including the setup network energy) at the traffic generation rate just before the saturation point of the conventional NoC. Our proposed NoC outperforms the conventional NoC, with 22% energy reduction for the *1-hot flow* and 9% reduction for the *3-hot flow* traffic loads. This is because the proposed NoC forwards a significant portion of the on-chip traffic over short-cuts, thereby the power-hungry buffering operations are removed. We omit the power diagrams due to the limited space.

4.2 SPLASH Traffic

Now, we evaluate the effectiveness of the proposed NoC under the traffic traces generated from SPLASH-2 benchmark suit [13].

The parameters of the NoCs are set like in the previous section. We set the period of the topology reconfiguration procedure for each benchmark separately in such a way that the procedure is invoked 5 times during the execution of the benchmark. Figure 4 compares the latency and energy obtained by the proposed and conventional NoCs across the SPLASH-2 traces, on a 7×7 NoC. The results are normalized to the results given by our NoC. On average, the proposed architecture outperforms the conventional NoC by 19% and 12%, when considering the packet latency and energy, respectively.

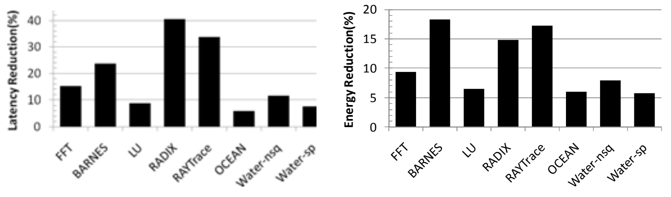


Figure 4. The reduction in average packet latency (a) and energy consumption (b) of the proposed NoC over the conventional NoC

5. CONCLUSIONS

We proposed a dynamically reconfigurable architecture for NoCs on which arbitrary application-specific topologies can be implemented. The reconfigurability of the proposed NoC architecture allows it to dynamically tailor its topology to the traffic pattern of different applications at run time. The network connectivity in our NoC is guaranteed by partitioning the entire links into two sets and keeping the connection of the links inside one of the sets fixed. Simulation results showed that, compared to a conventional NoC, the proposed NoC architecture consumes less power and reduces the average communication latency.

6. 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, 2007, pp. 96-108.

[2] T. Bjerregaard, and S. Mahadevan, "A survey of research and practices of network-on-chip", *ACM Computing Surveys*, Vol. 38, No. 1, 2006, pp.1-51.

[3] M. Modarressi, and H. Sarbazi-Azad, "Power-aware mapping for reconfigurable NoC architectures", In *Proc. of ICCD*, 2007, pp. 417-422.

[4] M. Modarressi, H. Sarbazi-Azad, M. Arjomand, "An SDM-Based Hybrid Packet-Circuit-Switched On-Chip Network", in *Proc. of DATE*, 2009.

[5] A. Kumar, et. al, "Express virtual channels: towards the ideal interconnection fabric", in *Proc. of ISCA*, 2007, pp.150-161.

[6] U. Ogras, and R. Marculescu, "Application-specific network-on-chip architecture customization via long-range link insertion", in *Proc. of DAC*, 2005.

[7] M. Modarressi, H Sarbazi-Azad, "A High-Performance and Low-Power Reconfigurable Network-on-Chip Architecture", Chapter 13 in *Dynamic Reconfigurable Network-on-Chip Design: Innovations for Computational Processing and Communication*, IGI Global Pubs., 2010.

[8] C. Gomez, et al., "Exploiting Wiring Resources on Interconnection Network: Increasing Path Diversity", *in Proc of 16th Euromicro PDP*, 2008.

[9] W. J. Dally, and B. Towles, *Principles and practices of interconnection networks*, Morgan Kaufmann Publishers, 2004.

[10] M. Modarressi, H. Sarbazi-Azad, A. Tavakkol, "Low-power and High-Performance On-Chip Communication Using Virtual Point-to-Point Connections", in *Proc. of NoCS*, 2009, pp. 203-213.

[11] Xmulator NoC Simulator, 2008, from http:// www.xmulator.org, 2009.

[12] 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. of DATE*, France, 2009.

[13] SPLASH-2, http://www.flash.stanford.edu/apps/SPLASH/.