Energy-Optimized On-Chip Networks Using Reconfigurable Shortcut Paths

Nasibeh Teimouri[1], Mehdi Modarressi[1],
Arash Tavakkol[1,2], and Hamid Sarbazi-azad[1,2]

[1] Sharif University of Technology, Computer Engineering department, Iran
[2] School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran
n_teimouri@ce.sharif.edu,
modarressi@mehr.sharif.edu,{arasht,azad}@ipm.ir

Abstract. Topology is an important network attribute that greatly affects the power, performance, cost, and design time/effort of NoCs. In this paper, we propose a novel NoC architecture that can exploit the benefits of both application-specific and regular NoC topologies. To this end, a subset of NoC links bypass the router pipeline stages and directly connect remotely located nodes. This results in an NoC which holds both fixed connections between adjacent nodes and long connections virtually connecting non-adjacent nodes. These shortcut paths are constructed at run-time by employing a simple and fast mechanism composed of two processes: on-chip traffic monitoring and path reconfiguration. The former keeps the track of the changes in the on-chip traffic pattern and detects high-volume communication flows. The latter then adapts the shortcut paths to the current on-chip traffic pattern by constructing shortcut paths between the source and destination nodes of the high-volume communication flows. Supporting the shortcut paths imposes a trivial overhead on the area of a conventional packet-switched router. Experimental results reveal the effectiveness of the proposed technique in reducing the energy consumption and improving the performance of NoCs.

Keywords: NoC, Topology, Energy consumption, Performance, Reconfiguration, Short-cut path.

1 Introduction

With the advance in semiconductor technology in recent years, current application-specific multi-core system-on-chips (SoCs) have rapidly grown in size and complexity. Future SoCs will consist of complex integrated components communicating with each other at very high-speed rates. The microprocessor industry is also moving from single-core to multi-core and eventually will lead to many-core architectures containing tens to hundreds of identical cores arranged as chip multiprocessors (CMPs) [1]. The lack of performance scalability of bus-based systems and significant area overhead and design time/effort of point-to-point dedicated links have motivated the researchers to propose packet-switched Network-on-Chip (NoC) architectures to overcome complex on-chip communication problems [2]. However, although NoC solves some problems, e.g. scalability, the need for complex and multistage pipelined routers results in a high router-to-link energy/delay ratio and increases the communication delay and energy.

The choice of network topology is an important decision in designing an NoC. Different topologies have different average inter-node distances and total wiring lengths. Network topology also greatly affects the communication flows distribution. These characteristics, in turn, affect the power consumption and average packet latency of NoCs [3]. The mesh topology is the most popular topology proposed for regular tile-based NoCs, as it is regular and has low cost. Despite such favorable advantages for on-chip implementation, some packets may suffer from long latencies due to lack of short paths between distant nodes in a mesh NoC.

Application-specific topologies, on the other hand, are suitable options when the target application and its traffic pattern are known at design time. Several previous studies have tried to address the drawbacks of the mesh topology by modifying it based on the target application traffic characteristics [4][5][6]. For example, a semi-regular topology is presented in [4] which is obtained by inserting some physical long links between distant nodes in a mesh, based on the traffic pattern of the target application. Long-range links target the nodes with high traffic volume and are static and constructed at design time. The authors reported significant improvements over a conventional mesh. However, since placing the extra links are done for a single application when the application is known at design time, this method cannot be employed in modern complex multicore SoCs and CMPs which run several different applications, often unknown at design time. Furthermore, this NoC cannot fully exploit the reusability and predictability of regular topologies; physical design and optimization procedures must be repeated for each NoC as the long links are specific to that NoC.

In general, this is the problem of almost all application-specific optimizations; they give the best power and performance results for a single target application for which they are customized, but they does not necessarily work well for different applications with different traffic patterns. Not being reusable, they also lose the shorter design time/effort of regular NoCs.

In this paper, we introduce a novel NoC architecture which can realize application-specific long links or shortcut paths in regular mesh NoCs. To keep the cost of the proposed NoC equal to a conventional packet-switched one, the shortcut paths are constructed by exploiting the current NoC resources. To this end, using the Spatial-Division-Multiplexing (SDM) scheme, each n-bit NoC link is divided into two $n/2$-bit parallel sublinks. One of the sublinks permanently connects two adjacent routers connected to its two endpoints, just like in a conventional mesh topology. The other sublink of each link, however, is devoted to establish shortcut paths between frequently communicating source-destination pairs by bypassing the intermediate routers. The packets traveling on this sub-network do not pass through the router pipeline stages of the intermediate routers, so the routing and arbitration and also the power-hungry buffering operations are omitted.

The SDM, also known as Link-Division-Multiplexing (LDM), is an alternative for the Time-Division Multiplexing (TDM). TDM is the dominant scheme for sharing network links among several circuits in circuit-switched NoCs. Unlike 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 allocats a sub-set of the link wires to a given circuit for the whole connection lifetime. The results in [7] show that the SDM removes the scheduling complexity and the memory needed for storing the switch configuration for every cycle in TDM. It also removes the power consumed at each cycle for changing the switch configurations. SDM has already been used to provide certain QoS levels [8], guarantee the required throughput of virtual circuits in application-specific NoCs [9], and low-power hybrid packet/circuit-switching [6].

Our routers can be considered as 16-port routers (8 input and 8 output ports, in addition to the ports to the local core) each of which of *n*/2-bit wide. 8 ports of these 16 ports are fixed and connected to adjacent routers according to the mesh topology, and the remaining 8 ports are connected to some other nodes according to the irregular pattern exhibited by the running application(s).

Since these irregular links are constructed over the structured NoC components, they benefit from the advantages of both application-specific and standard topologies. Reconfigurability is the key point of our approach; shortcut paths can be always customized based on the current on-chip communication characteristics. In addition, our approach imposes negligible area overhead to the NoC.

A rather similar approach has been applied in the hybrid circuit/packet-switched NoC presented in [6]. However, in [6], the circuits (which are equivalent to our shortcut paths) are set up on a per packet basis in a single cycle and held throughout the packet's duration. This involves a fast control network capable to set up the circuits in a single cycle. Our method constructs the shortcut paths for a communication flow, instead of individual packets, in order to reduce the frequency of the circuit construction procedure invocation and also remove the need for fast real-time path construction. Our work also differs from some other NoCs combining packet and circuit switching [16], since the main role of the packet-switched part of most of these proposals is to set up circuits, rather than carrying the application messages. They also do not propose algorithms for efficient circuit construction, while our proposal includes an efficient algorithm to find the best path for shortcut path construction.

In this paper, we first introduce the proposed router architecture and then, present a shortcut path construction mechanism to absorb the packets of high-volume communication flows and deliver them to the destination nodes with near-ideal power and delay. This procedure involves monitoring the on-chip communication in order to detect frequently communicating nodes and reconfiguring the shortcut paths in favor of them.

The rest of the paper is organized as follow. Section 2 presents the proposed NoC architecture. The run-time shortcut path construction algorithm is introduced in Sections 3. Section 4 presents experimental results and finally, Section 5 concludes the paper.

2 The Proposed NoC

2.1 Resource Partitioning Mechanism

As mentioned in Section 1, in this work, we propose dividing the entire NoC links into a reconfigurable and a fixed set of links using the SDM scheme. Applying SDM in a packet-switched network leads to splitting the links into two parts and increases the average packet latency, because the number of flits of a packet is increased. This effect is particularly more noticeable in low traffic when the packets rarely face contention in using the shared resources. SDM, however, allows having several links in parallel in the same direction and therefore increases the number of distinct paths in that direction. In medium and heavy traffic loads, this increase in the path diversity reduces the *head of line (HOL)* blocking and can compensate the latency overhead of SDM and even improve the average latency and throughput of the network.

An in-depth analysis of the effects of splitting the wide links into smaller parallel links on network latency and throughput is given in [10], where 512-bit wide NoC links are split into different numbers of sub-links with equal widths. Using two different network sizes and packet sizes up to 1248 bits, authors in [10] show that splitting the links into two sub-links increases the throughput by 50-60% with some negative effect on the average message latency under low-traffic loads.

In our method, by establishing shortcut paths in one of the sub-networks, we can more fully mitigate the negative effects of the increased packet size (in terms of the number of flits) of using SDM and even offer better performance results. From the energy consumption point of view, our method has a great advantage over a conventional NoC, since it provides power-efficient shortcut paths for a portion of the packets. In a canonical packet-switched NoC with pipelined routers, passing each hop involves traversing different router pipeline stages [11]. The proposed shortcut paths can be considered as virtual pipelined links, on which the flits only pass through *link traversal* and *crossbar traversal* pipeline stages at each intermediate hop and skip over other stages and save their corresponding power consumption. In addition, as the SDM circuits remain unchanged for some time intervals, the crossbar does not switch between different input and output ports frequently (unlike a crossbar in a packet-switched network). Consequently, the control line (or select line) of the crossbar has no activity and hence, consumes no dynamic power. Our study using the Orion power library [19] shows that the control line accounts for 15-20% of the total crossbar power consumption.

2.2 NoC Architecture

Fig. 1 illustrates the overall technique employed by the proposed NoC architecture where the *n*-bit wide network resources (links, buffers, and crossbar) are divided into two parallel *n*/2-bit sub-networks. One of the *n*/2-bit sub-networks forwards the packets according to the conventional packet-switching scheme and the other sub-network is used to construct shortcut paths. Let us refer to the former sub-network as PS (packet-switched sub-network) and the latter as SC (shortcut sub-network). As shown in the figure, the total buffering space in a router remains the same as that of the original conventional NoC. The crossbar switch structure is also the same as in a conventional packet-switched router. However, the switch allocator (arbiter) is divided into two parts to handle the SC and PS flits separately. The SC allocator of every output port (Shortcut Path Register in Fig. 1) is simply made by a register which determines which input port is connected to that output port, if the output port is a part of a shortcut path.

The input ports, however, are slightly modified in order to separate the SC and PS sub-networks and provide different paths for packets to travel over and switch between the sub-networks.

Fig. 1 also shows the functional view of the internal structure of an input port and highlights different paths that packets may take. Simply stated, PS packets are buffered and go through the router pipeline stages at each hop. SC packets, on the other hand, bypass the pipeline stages in the intermediate nodes of a shortcut path, but are buffered at the endpoint node of the shortcut path.

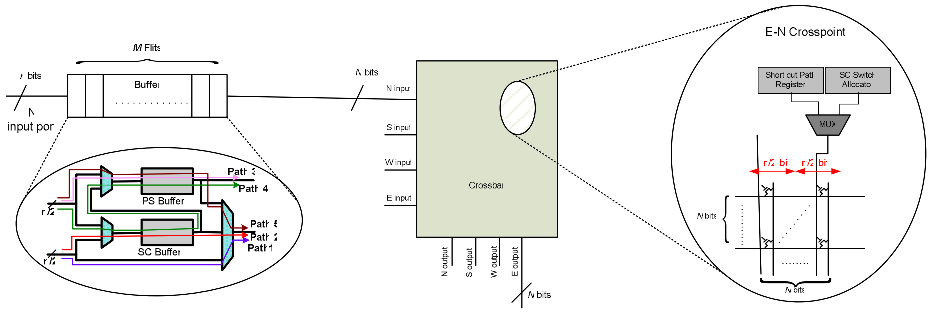

Fig. 1. The router architecture and the details of the input port and crossbar

If the SC part of a link is not currently used by a shortcut path, it will be used to establish a regular connection between its upstream and downstream nodes. In this case, the link is controlled by a regular switch allocation unit corresponding to that SC port (SC Switch Allocator in Fig. 1) by appropriately setting the multiplexer of the port (MUX in Fig. 1).

Consequently, we have two parallel *n*/2-bit ports between the two adjacent nodes that make the bandwidth between them doubled. The packets then use the SC and PS parts through paths 2 and 3 in Fig. 1. If the SC port is used by a shortcut path and is an intermediate node along the path, the packets take path 1 in Fig. 1 and bypass the router and head to the next router through the port determined by Short-cut Path Register in Fig. 1. However, if the current node is the endpoint node of a shortcut path, the flits are directed to the buffer at the SC port (path 2 in Fig. 1).

Once the flits are buffered in either PS or SC buffers, they should pass the pipeline stages of the router. For each header flit, the routing logic performs *route computation* (RC) to determine the proper output port of the packet. The NoC uses an adaptive minimal routing algorithm. Applying a minimal routing scheme, a packet is only allowed to take ports along the shortest paths toward the destination (at most two directions and 4 ports). The algorithm checks the corresponding output ports and assigns one of the free ports to the packet. The algorithm, in particular, checks whether there exists a free SC shortcut path originating from the current node and destined to some node along the route towards the packet's destination; these ports are prioritized over PS ports as they shorten the packet path. If two free shortcut paths are found, the one which allows more intermediate nodes to be skipped over is selected.

As a result packets may switch between the SC and PS sub-networks several times to reach their destinations. Moving from SC to PS and from PS to SC sub-networks is done through a part of path 4 and path 5 in Fig. 1, respectively.

Each port has two virtual channels. We guarantee deadlock freedom by employing the well-known escape channel scheme [11]. One of the VCs of the PS part adopts a deadlock-free routing algorithm (here, XY routing) and is used as the escape channel, when a deadlock is detected.

Another issue in the proposed architecture is that when a SC port is used by a shortcut path, its buffer is not used, as packets skip over it. In this case, we can add the unused buffers to the neighboring PS port to increase the packet-switched network buffering capacity and hence, increase its performance. In this case, the PS packets take path 4 in Fig. 1. Note that Fig. 1 shows the functional view of the proposed buffer merging approach. The actual implementation of the reconfigurable buffers can be easily realized by several methods proposed in the literature, e.g. the one presented in [12], with negligible overhead.

The area of the proposed NoC is rather the same as a conventional NoC. The only source of overhead is the multiplexers used in the input ports and the extra routing and allocation logic needed to handle the PS and SC packets separately. We modeled the area of a conventional and the proposed router using the analytical NoC area models presented in [13]. We also used the area values in [14] to set the parameters of the model. The model shows that the area overhead for a 64-bit 6×6 NoC with 8-flit deep buffers is less than 2%.

3 The Reconfiguration Procedure

This section presents a run-time algorithm to adapt the shortcut paths to the current on-chip traffic pattern. Our approach relies on constructing shortcut paths for high-volume traffic flows in order to improve the average performance and power of the network. The idea behind selecting the source and destination nodes of high-volume communication traces for shortcut path construction is that by providing shortcut paths for such flows, more packets can use these paths and we obtain more power and performance gains.

Constructing a shortcut path takes a few clock cycles. However, the proposed procedure is performed as a background process in parallel with normal NoC operation. Therefore, unlike the traditional circuit-switched networks, path setup latency does not degrade the network performance.

Simply stated, the whole procedure is as follows. The NoC monitors the traffic pattern and all source-destination pairs which communicate at a rate higher than a predefined threshold are considered for shortcut-path construction. An algorithm then finds a shortcut path between the two nodes along one of the shortest paths between them. If there are not sufficient free resources for constructing a new shortcut path, the algorithm is allowed to tear down old shortcuts, but it minimizes the cumulative weight of the shortcut paths that have to be torn down in order to make enough resources for the new shortcut. To achieve this goal, we need the following steps.

Monitoring. Each node keeps track of the communication flows it originates by storing the number and the destination node addresses of the packets it sends. Periodically, at specific times, the m most-significant bits of the register holding the traffic volume are considered as the new weight of the flow. The weight of each existing shortcut path, defined as the communication rate passing over them, is also calculated by its upstream node.

Transferring the traffic information. At specific times, this information is sent to a root processor in order to reconfigure the shortcut paths based on the current on-chip traffic. Some previous work [15][16], use a dedicated control network for control messages. However, as the messages are small and the shortcut reconfiguration procedure is invoked very infrequently, we get the data network to carry control packets with negligible performance overhead. Each node first sends the current weight of the shortcut paths starting from it to the root processor to update its information about the current state of the existing shortcut paths. It is followed by sending the shortcut construction requests (composed of the weight and the destination of the candidate flows) for the traffic flows weighting higher (in terms of the communication rate) than a predefined threshold to the root node. To avoid congestion at the root node, the nodes send their information one by one in different times. To this end, first the processor at the address (0,0) sends the weight of its shortcut paths to the root and then sends a signal (token) to the next processor (at the address (0,1), for example) to allow it to send its shortcut weights. This procedure continues until the last processor in the order (the processor at the address (n-1,n-1), for example) sends its information. It then signals processor (0,0) to get it to start sending the shortcut path requests. The requests are then sent one by one to the root processor.

Once the new configuration of the shortcut paths are calculated by the root processor, it sends the new configuration data to the NoC routers in order to set up new shortcut paths. The information delivered to each node includes the values of the SC allocators, as well as the values control the select line of the multiplexers of the input ports to construct appropriate data-paths in the input ports.

Shortcut Construction. As mentioned before, this algorithm is run on one of the NoC nodes. Since most current SoCs contain a central configuration processor that configures the system [17], we assume that the task of managing shortcut connections is assigned to this processor. Being simple and infrequently called, the algorithm does not impose considerable performance overhead to the root processor. Moreover, since the shortcut construction job is performed in parallel with NoC normal operation, there is no need for real-time path construction and the algorithm can be given a lower priority than the current tasks of the root processor.

Once the new weight of the current shortcut paths and the requests for new shortcuts are delivered to the root processor, the shortcut path reconfiguration is started. Our algorithm, for every selected request between nodes x and y, finds a shortcut path $sc_{x,y}$ over one of the shortest paths between x and y with minimum cost. The cost of a shortcut path is defined as

$$Cost(sc_{x,y}) = \sum_{\forall sc_{i,j} \in SC} W(sc_{i,j}) \times \varphi(sc_{x,y}, sc_{i,j}) \tag{1}$$

where $\varphi(sc_{i,j},sc_{k,l})$ is defined as

$$\varphi(sc_{i,j}, sc_{k,l}) = \begin{cases} 0, & if\ sc_{i,j}\ and\ sc_{k,l}\ share\ a\ common\ link \\ 1, & otherwise \end{cases} \tag{2}$$

As our algorithm is allowed to tear down existing shortcut paths, the relation indicates that the algorithm should minimize the cumulative weight of the shortcut paths that should be destroyed in order to have sufficient free resources to establish the new shortcut path.

If *the weight of* $sc_{x,y}$ *is greater than* $Cost(sc_{x,y})$, $sc_{x,y}$ will be set up in the NoC and all shortcut paths $sc_{i,j}$ with $\varphi(sc_{x,y},sc_{i,j})=1$ will be torn down, otherwise the shortcut path construction request is rejected. The traffic flows corresponding to the torn-down shortcut paths can request for constructing a new shortcut path at the next shortcut construction round, provided that their corresponding flow weight is still above the threshold.

Our algorithm serves the received shortcut path construction requests in the decreasing order of their weights. Constructing the shortcuts in this order guarantees that no newly constructed shortcut is torn down by a consequent shortcut path request at the same round, since a shortcut cannot destroy a shortcut with greater weight.

We use the communication rate of between the network source-destination pairs as the criteria for detecting high-traffic paths. However, once the shortcut paths are set up, each packet (not necessarily the packets of the flow based on which the shortcut path is constructed) which finds some shortcuts useful for shortening its path can use them.

3.1 Shortcut Reconfiguration Algorithm

We apply Dijkstra's algorithm to construct shortcut paths with minimum cost. To this end, for the input of Dijkstra's algorithm, we construct a directed weighted graph out of the current shortcut paths. For an $n \times n$ NoC, we construct a *Shortcut Graph* $G=(V,E)$, where $|V|=n^2$ and each $v_i \in V$ is corresponding to one of the NoC nodes. Each directed edge $e_{ij} \in E$ is also corresponding to each NoC link. If the link between an NoC node v_i and its adjacent node v_j is used by a shortcut path, the weight of e_{ij} is set to the weight of that shortcut path, otherwise is set to 0. At each period, the received weight of the existing shortcut paths is used to update the edge weights. We use some simple techniques for graph generation in order to obtain the precise value of a path cost and also increase the speed of Dijkstra's algorithm, but omit the details of these techniques due to the limited space.

Dijkstra's algorithm does not guarantee that the path with minimum weight is also a minimal path in terms of hop count, but a shortcut path should be constructed over a shortest path between the source and destination nodes of the corresponding communication flow.

Being laid out along one of the shortest paths between two nodes, the shortcut paths are allowed to span along at most two cardinal directions according to their source and destination addresses, hence the shortcut paths can be classified into four classes: North-East, North-West, South-East, and South-West. Similarly, the G edges are grouped into the same four quadrant sets according to their source and destination addresses. To find a path for a request, the algorithm only considers the graph edges belonging to the same quadrant as the requesting shortcut path. Obviously, this guarantees that the algorithm only explores the shortest paths between the source and destination nodes to find the path with the minimum cost.

When the algorithm finds a path for a request, the path cost is compared to the weight of the requesting flow. If the cost is less than the flow weight, G is updated by storing the path of the new shortcut path and removing the overlapping shortcut paths. Appropriate signals are then sent to the nodes along the new and destroyed shortcut paths to set up the new shortcut path and destroy the conflicting ones.

The reconfiguration is actually done when all in transient packets of the current shortcut paths are delivered to the destination node.

4 Experimental Results

In this section, we evaluate the impact of the proposed NoC architecture on the energy consumption and performance of NoCs under some realistic and synthetic benchmarks.

The results are compared against a mesh NoC with speculative 4-stage pipelined routers. We cannot compare our work to long-links presented in [4], as they are built statically at design time, while we target dynamic traffic patterns where it is not possible to know in advance the exact communication pattern of a running application.

For the two NoCs, simulations are performed for a 64-bit wide system with speculative 4-stage pipelined routers, 2 virtual channels per port, and 8-flit deep buffers. In all of the simulations, each node sets the weight threshold to the average weight of its communication flows and issues shortcut requests for the flows weighting higher than this threshold. We have evaluated the proposed NoC architecture using Xmulator, a fully parameterized simulator for interconnection networks [18]. The Orion power library [19] integrated to Xmulator calculates the energy consumption of the NoCs. The energy 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. We evaluate the effectiveness of the proposed NoC under the traffic traces generated from SPLASH-2 benchmarks [20] in addition to synthetic traffic patterns.

As mentioned before, the shortcut construction process has no impact on NoC performance, as it is done in parallel with the NoC operation. It also has negligible impact on NoC energy consumption due to its simple algorithm and infrequent invocation. However, we include the energy consumption of the algorithm in the reported values by calculating the number of logic and arithmetic operations it needs and estimating the energy consumption of each operation.

As mentioned before, we expect to improve the NoC energy consumption more than the packet latency, because increasing the flits of a packet has a negative impact on the latency. If we relax the NoC area constraint (set the bit-width of the sub-networks to a value greater than n/2), we can achieve better latency results, as well. This work, i.e., exploring the trade-off between the cost and performance of the proposed NoC is left for future work.

4.1 Synthetic Traffic Results

Most multi-core SoC and scientific CMP workloads, exhibit a high degree of temporal and spatial communication locality [5][16]. As a result, we use a synthetic traffic with similar characteristics to SoC and scientific CMP workloads. To this goal, we use *n-hot flow* synthetic traffic patterns presented in [5] to evaluate the effects of the shortcut path connections on typical CMP workloads. In the *n-hot flow* traffic pattern, each node sends a considerable portion of the generated packets (60% in our experiments) to exactly *n* destination node and the remaining traffic to other randomly chosen nodes, while the hot destination nodes of each source node are selected randomly. Due to the limited space, we only use the 3-hot flow traffic pattern (n=3). Each packet is 8-flit long. Each simulation runs for 500,000 cycles and the network initiates the shortcut path construction procedure every 50,000 cycles.

Fig. 2 shows the average packet latency and energy per flit of the proposed NoC and other NoC design as a function of the injection rate (packet/node/cycle) under hot-flow synthetic traffics in a 6×6 mesh NoC. As we expect, in low traffic rates, the conventional speculative NoC outperforms our NoC, as the contention probability is low and the path diversity provided by SDM cannot mitigate the negative effect of the increased packet length. However, as shown in the figure, the proposed shortcut paths improve the latency of the NoC over other considered NoC design at medium and high traffic loads.

As a result, by setting the link bandwidth (by static or dynamic voltage/frequency scaling methods) in such a way that the links work under medium traffic loads, we can benefit from the energy and performance efficiency of shortcut paths, as well as better NoC resource utilization.

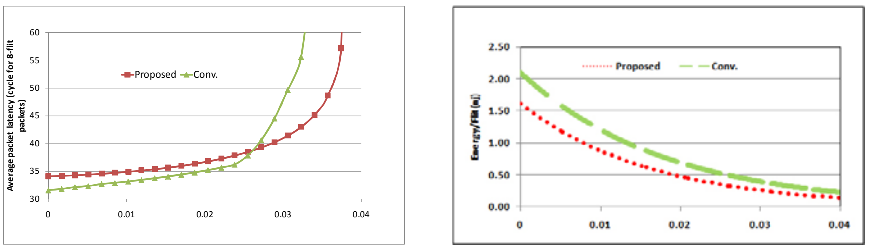

Fig. 2. The average packet latency (cycle) and energy per flit (nJ) of the proposed and a conventional 6×6 NoC under 3-hot flow traffic for different injection rates (packet/node/cycle)

4.2 SPLASH-II Results

We then evaluate the effectiveness of the proposed shortcut paths under the traffic traces generated from SPLASH-2 benchmarks. Based on the results of the previous experiments, we set the bandwidth of the NoC for each individual benchmark in such a way that the links work under a medium traffic load.

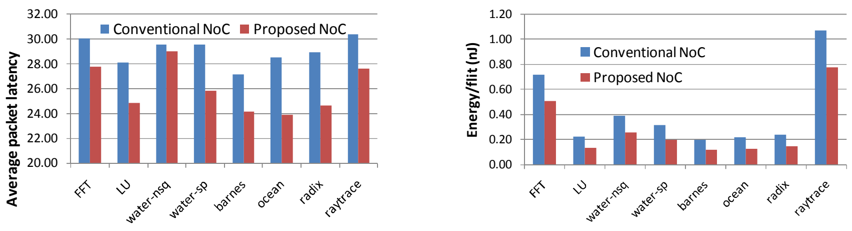

Fig. 3. The average packet latency (cycles) and energy per flit (nJ) for the SPLASH-2 programs

Fig. 3 compares the latency and energy per flit results obtained by the proposed and conventional NoCs across 7 SPLASH-2 traces, on a 7×7 NoC. We set the period of the topology reconfiguration procedure to 100,000 cycles, while each simulation runs for 5,000,000 cycles.

As the figure shows, on average, our proposal outperforms the conventional NoC by 36% when considering the energy consumption. Following the same trend, the packet latency offered by the NoC with shortcut paths outperforms the conventional NoC by 8%, on average.

The shortcut paths give their best improvement when a large portion of the packets are transmitted along several (preferably small number of) high-volume traffic flows.

5 Conclusion

In this paper, we proposed an NoC architecture in which some adaptable long links can be established between frequently communicating nodes. The proposed NoC architecture holds the benefits of both application-specific and regular standard topologies. We then presented a reconfiguration procedure in order to adapt the NoC topology to the current on-chip traffic characteristics. The entire procedure relies on monitoring the on-chip traffic and changing the shortcut paths in response to a change in the on-chip traffic. Using a centralize management approach, we showed that our proposal can effectively improve the NoC energy and performance metrics over a conventional packet-switched NoC with the same cost. Reconfigurability is the key advantage of our work over previously proposed methods, in that our NoC adapts its shortcut paths to the traffic pattern exhibited by the current running application.

For future work, we will consider using the TDM scheme for resource partitioning between the two sub-networks. In addition, improving the proposed architecture in order to offer adaptable sub-network bit-width can be considered as another future work in this line.

References

1. Owens, J., Dally, W.J., Ho, R., Jayasimha, D.N., Keckler, S.W., Peh, L.S.: Research challenges for on-chip interconnection networks. IEEE Micro 27, 96–108 (2007)
2. Benini, L., De Micheli, G.: Networks-on-Chip: a new paradigm for systems on chip design. In: Date on design, pp. 418–419 (2002)
3. Jantsch, A., Tenhunen, H.: Networks on Chip. Kluwer Academic Publishers, Dordrecht (2003)
4. Ogras, U., Marculescu, R.: Application-specific network-on-chip architecture customization via long-range link insertion. In: IEEE/ACM International Symposium on Computer Aided Design, San Jose (2005)
5. Modarressi, M., Tavakkol, A., Sarbazi-Azad, H.: Virtual Point-to-Point Connections in NoCs. IEEE on Computer-Aided Design for Integrated Circuits and Systems 29, 855–868 (2010)
6. Modarressi, M., Sarbazi-Azad, H., Arjomand, M.: An SDM-Based Hybrid Packet-Circuit-Switched On-Chip Network. In: DATE, pp. 566–569 (2009)
7. Marchal, P., Verkest, D., Shickova, A., Catthoor, F., Robert, F., Leroy, A.: Spatial Division Multiplexing: a Novel Approach for Guaranteed Throughput on NoCs. In: CODES+ISSS, pp. 81–86 (2005)
8. Morgenshtein, A., Kolodny, A., Ginosar, R.: Link Division Multiplexing (LDM) for Network-on-Chip Link. In: 24th IEEE International Symposium on Electrical and Electronics Engineers (2006)
9. Wolkotte, T., Smit, G.J.M., Rauwerda, G.K., Smit, L.T.: An Energy-Efficient Reconfigurable Circuit Switched Network-on- Chip. In: Reconfigurable Architecture Workshop, RAW (2005)
10. Gomez, C., Gomez, M.E., Lopez, P., Duato, J.: Exploiting Wiring Resources on Interconnection Network: Increasing Path Diversity. In: PDP, pp. 20–29 (2008)
11. Dally, W.J., Towles, B.: Principles and practices of interconnection networks. Morgan Kaufmann, San Francisco (2004)

12. Petrov, P., Orailoglu, A.: Performance and Power Effectiveness in Embedded Processors: Customizable Partitioned Caches. In: IEEE International Symposium on Computer Aided Design of Integrated Circuits and Systems, vol. 20 (2001)
13. Balfour, J., Dally, W.J.: Design tradeoffs for tiled CMP on-chip networks. In: The International Conference of Supercomputing, pp. 178–189 (2006)
14. Booksim NoC simulator, `http://nocs.stanford.edu/booksim.html`
15. Modarressi, M., Sarbazi-Azad, H., Tavakkol, A.: An Efficient Dynamically Reconfigurable On-chip Network Architecture. In: DAC, pp. 166–169 (2010)
16. Jerger, N.E., Lipasti, M., Peh, L.: Circuit-Switched Coherence. Computer Architecture Letters 6, 5–8 (2007)
17. Goossens, K., Dielissen, J., Radulescu, A.: The Æthereal Network on Chip: Concepts, Architectures, and Implementations. Design and Test of Computers 22, 414–421 (2005)
18. Xmulator NoC Simulator, `http://www.xmulator.org`
19. Kahng, A., Li, B., Peh, L., Samadi, K.: ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration. In: DATE, France, pp. 423–428 (2009)
20. SPLASH-2, `http://www.flash.stanford.edu/apps/SPLASH`