On the Design of Hybrid Routing Mechanism for Mesh-based Network-on-Chip

Pooria M.Yaghini*, Ashkan Eghbal, Nader Bagherzadeh

Department of Electrical Engineering and Computer Science, University of California, Irvine
Irvine, CA 92697 USA
{*pooriam, aeghbal, nader*}*@uci.edu*

Abstract

Efficient on-chip communication is necessary for exploiting enormous computing power available on a many-core chip. Routing algorithms play a major role for the communication quality and performance of the on-chip interconnection networks. This paper proposes TagNoC, as an on-chip network router architecture with novel hybrid routing approach which reduces latency and power consumption at a fixed cost based on information redundancy. TagNoC is a hybrid routing approach which combines the source and distributed routing methods together. While eliminating packet routing in each router, TagNoC determines the forwarding output port in parallel with input buffering. For a marginal cost increase in header size, as compared to distributed routing techniques, routing latency can be hidden while eliminating power consuming portion of the routing, increasing router throughput and decreasing latency. The proposed TagNoC router is compared to baseline router with distributed routing implementation on a 16-node CMP mesh. Physical implementation of all routers is modeled using synthesized RTL, detailed area analysis, and accurate channel models. Performance evaluation is also carried out utilizing RTL simulation and detailed power analysis on both synthetic and application traffic is performed using post-synthesis gate-level simulation. The simulation results illustrate that TagNoC outperforms as compared to baseline distributed architecture and other source routing methods in terms of power, latency, and throughput.

Keywords: Network-on-Chip, Source Routing, Tag, Power, Distributed Routing

1. INTRODUCTION

The traditional high clock rate single core systems have been replaced by distributed many-core systems on a single die due to energy consumption and performance limits. Data transmission through a chip is considered more difficult since global interconnects are becoming the principal performance bottleneck for high performance systems [22, 24]. System-on-Chip (SoC) does not support future technologies as the number of cores on single dies increases. Network-on-Chip (NoC) has been proposed as a new interconnection architecture to support better modularity, scalability and higher bandwidth features [2, 7, 17, 19]. Power dissipation is becoming critical constraints of system design due to battery lifetime, cooling, and thermal budgets concerns [5]. It has been reported that network power for a many-core die in the future can be as high as 150W, assuming current network scale implementations [3]. Reducing energy consumed in NoC is of great importance for high performance and energy-efficient designs [4, 28, 35].

Routing is an integral part of NoC, and has a significant impact on the communication efficiency, especially in case of single-flit packets [12].

Routing algorithm can be implemented as source or distributed routing. In source routing method the complete path from source to destination is precomputed at source router and accumulating the exact router-to-router packet traversal information in the header. This information leads the packets to traverse through intermediate routers toward their destination, which has reasonable overhead for small networks. The header of the packet needs to contain at most k units of routing information for a network with a diameter of k. The header overhead becomes noticeable as the network size grows which is a big challenge for on-chip routing. However, the routing decision is made by the individual routers in distributed routing method, in which the header of a packet requires to contain only the destination address. The destination address is compared in each intermediate router to choose the appropriate channel to forward the packet. The router complexity of the latter scheme is higher than the former one though it imposes scalable information redundancy [9].

Although distributed routing is the most common technique in NoC, this paper TagNoC, as an hybrid routing method, showing superior performance along with saving area or power, by adding fixed number of bits (only one bit per dimension) which is called *Tag*.

Major contributions of this article are:

- Propose a new hybrid routing technique to reduce the intrinsic overhead of conventional source routing and per-hop running of distributed routing methods.
- Provide a detailed implementation of the proposed technique,TagNoC, with comprehensive scalability, performance, power, frequency, and area analysis.

*Corresponding author

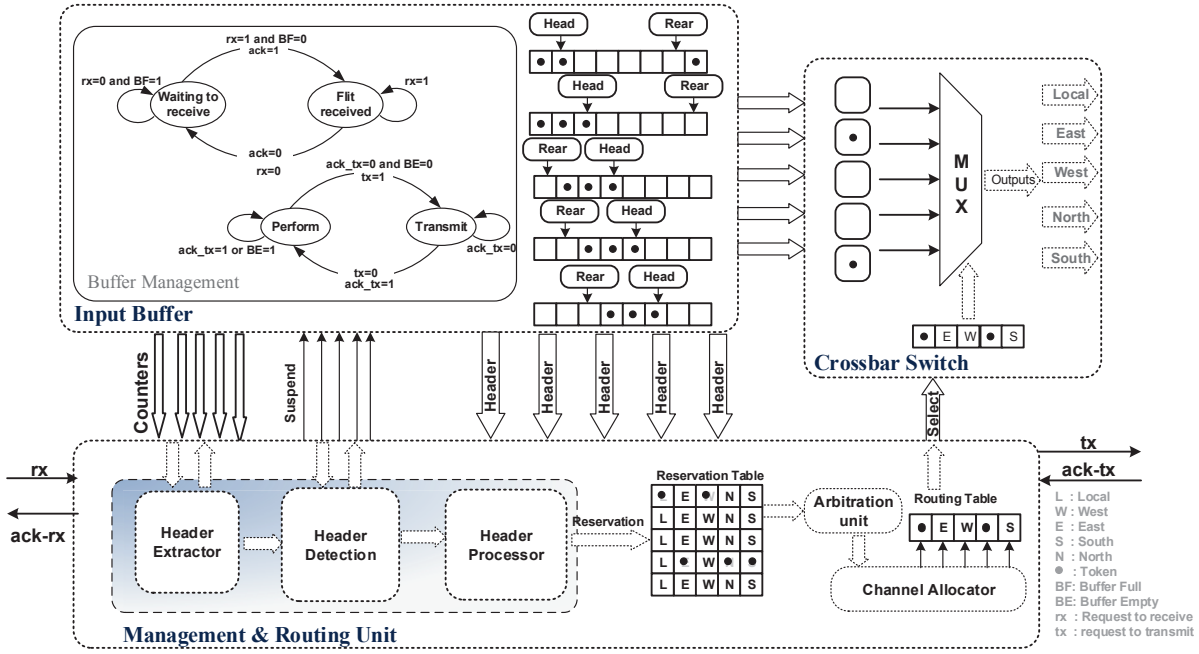


Figure 1: NoC router architecture for distributed routing

- Compare power and performance impact of baseline routing and source routing designs versus TagNoC routing by synthetic and scientific/commercial application workloads.

Overall, it is demonstrated that the improved router efficiency and elimination of unproductive link transitions enable the TagNoC router to outperform baseline router on power and latency basis, besides the marginal data redundancy in the header flit.

The remainder of this paper is organized as follows: Section 2 describes the related research work to this paper. NoC router architectures with the definition of source and distributed routing are illustrated in Section 3. In Section 4, the motivation of the proposed optimized hybrid Tag-based routing technique, called TagNoC architecture, is explained in details considering its scalability. Detailed evaluation methodology and power, performance, and area analysis of proposed approaches are explored in Section 5. Finally, Section 6 presents the conclusion remarks.

2. Related Work

We discussed the TagNoC architecture to exploit opportunities for having better performance with lower energy consumption using a better way of routing packets in NoC. To the best of our knowledge, there is no routing algorithm for NoC combining both source and distributed routing models together and TagNoC is the first hybrid routing approach for NoC. However, there are couple of routing algorithm classes which can benefit from TagNoC and in this section we relate our contributions to prior studies on these source and distributed routing algorithms.

2.1. Distributed Routing

Various distributed routing algorithms have been proposed for NoC platforms. An adaptive distributed routing algorithm for 2D mesh NoC is presented, which is based on Dynamic XY (DyXY) [21]. A distributed routing algorithm is presented without any routing table in [13, 34]. In [10] a distributed routing algorithm for a 3D NoC is introduced.

Turn Model routing scheme based on wormhole switching mechanism provides deadlock and live-lock freedom in the mesh topology [6, 14] which are the most popular routing models utilized in NoC architectures. Three well-known turn models are Negative-First (NF), West-First (WF), and North-Last (NL) [9] with six allowed turns. XY routing algorithm is also a popular dimension order routing algorithm in which four turns are avoided in order to prevent deadlocks. TagNoC supports all turn model and dimension order routing algorithms including XY, NF, WF and NL.

Adaptive routing algorithms [31, 40, 32] are not supported in the simple form of TagNoC since there is no routing component inside the TagNoC routers. The Odd-Even model is one of the most popular partial adaptive wormhole routing algorithms in 2D mesh on-chip interconnection network [6] without virtual channels. Unlike the turn model which prohibits certain turns in all locations of the network, in the Odd-Even model some turns are restricted. Although adaptive routing algorithms cannot be implemented with TagNoC, Odd-Even model as a partial adaptive routing algorithm, is supported in this approach. The reason is that the routing rules in Odd-Even model are fixed and pre-defined, so can be computed as source routing prior to having the packets traversed in network.

2.2. Source Routing

Recent studies have explored various aspect of source routing for NoC [18, 26, 23, 20].

Distributed routing algorithm in NoC is commonly used, but source routing algorithm in NoC has not been fully considered because of the large overhead needed to store path information in the header. Furthermore, source routing will provide limited path adaptivity in the presence of faulty links or congestion in the network. In [18], a fault-tolerant source routing algorithm was proposed that demonstrated 50% more fault tolerance as compared to the conventional algorithms. Source routing is not suitable for networks with variable dimension and topology. However, it can be efficiently encoded with small number of bits in a NoC, since the topology and number of nodes are fixed at run-time. According to [26], authors have decreased the control data overhead by employing two bits encoding scheme every hop all the way to the destination. The router design is significantly simplified by the help of encoded source routing technique. This scheme shrinks the overhead imposed by the source routing to a satisfactory level, but still the number of extra bits needed to be pushed into the header is a function of network dimension. With increasing growth of number of cores in many-cores, this method does not provide a promising approach.

A class of source routing switches that can be used efficiently in arbitrary network topologies has been presented in [23]. The size of routing tables in the network interface increases when source routing is used. An exploration and synthesis of low-overhead configurable source routing tables for network interfaces is presented in [20], resulting in up to 15 times reduction of area overhead in routing tables.

3. NoC Router Architectures

NoC is the dominant infrastructure for communication in highly massive many-core systems.

Our baseline NoC router is implemented with five bidirectional ports, to support mesh and torus topologies in Verilog. It is composed of three main components including an input buffer, a management and routing unit, and a crossbar switch [12, 38], as shown in Figure 1. The shaded sub components in management and routing unit component of Figure 1 are the focus of this paper. It should be noted that discussions and cost analysis are performed based on a 2D mesh topology.

The input buffer is responsible for storing incoming packets while there is free space. By using a circular buffer scheme it was possible to optimize the buffering operation. Buffer management subcomponent is composed of two state diagrams for receiving, storing and transferring data packets, as illustrated in Figure 1. To comply with the wormhole switching technique, the buffer size is chosen to be less than the packet size.

The management and routing unit component is the central unit which includes header extractor and header processor for executing the routing algorithm, arbitration unit and the routing table. Once the trailer flit is recognized by the routing component, the corresponding output port is released. The grant and activation signals are disabled as the trailer flit of a packet is transmitted to its desired output port. Each of the input channels can reserve one of the output ports when the routing process is accomplished. An arbitration unit locks the dedicated output channel until the end of packet transmission. It is also responsible for assigning output channels evenly among input ports by implementing a round robin algorithm. The routing unit grants the requested input port and triggers the selected output multiplexer to establish a connection, once the routing process is accomplished successfully. Such grant and activation signals are disabled as the trailer flit of a packet is transmitted to its desired output port.

The third component of NoC is the crossbar switch. Control signals of the routing unit select one of the output ports for an incoming header flit. By implementing the wormhole switching, all of the remaining flits of a packet follow the header flit. If a header is blocked, then the following flits are blocked, as well. Once a packet transmission is completed, the switch is unlocked to serve other input channels. NoC architectures discussed here are distinguished by their implementation of routing algorithm. All of theses source routing methods are extended version of the baseline routing method. First, None Encoded Address (NEA) method is introduced as the naive source routing techniques. Also Encoded Address (EA) and Optimized Encoded Address (OEA), as efficient source routing algorithms for NoC, are explained in the following subsections. In Subsection 4.3, they are compared against proposed TagNoC approach in terms of header size overhead, affecting the scalability and reliability of the system. Power consumption, area overhead, and latency comparisons of them and proposed TagNoC approach are reported in Section 5.

3.1. NEA

NEA technique is considered as the very first source routing approach. In this method, the routing process will be done in the source router prior to injecting a packet into the network. The header flit contains the coordinates of all intermediate routers from source toward its destination instead of just the coordinates of the destination node. The elimination of routing decision in the intermediate routers results in less latency and power consumption, at the cost of increasing header flit size. However, NEA is not scalable and imposes a large amount of data overhead, as discussed in Subsection 4.3. Header flit size for this method depends on the network size since the necessary number of bits for distinguishing each node is a function of the number of routers per each dimension of the network. For example in an 8×8 network, 3 bits are needed for each intermediate router while 4 bits are needed in a 16×16 network. Compressing the overhead bits in the header flit is critical and it is considered for the following techniques.

3.2. EA

EA [26] is the second source routing approach which decreases the overhead of the NEA method. In this method, output port of each intermediate router is represented by only two bits to support all the other ports except the incoming input port in

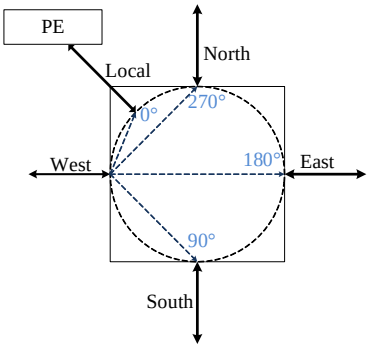


Figure 2: Port forwarding directions

Dimension	Code	Rotation
Before turn	00	180°
	01	90°
	10	270°
	11	0°
After turn	00	180°
	01	90°
	10	270°
	11	0°

Table 1: EA rotational codes

Dimension	Code	Rotation
Before turn	00	180°
	01	90°
	10	270°
	11	0°
After turn	0	180°
	1	0°

Table 2: OEA rotational codes

a 5-port NoC router. The information overhead of this method is two bits per hop in the header flit, which is provided by Network Interface (NI) component. EA method utilizes a coding in which each incoming flits has four possible options to choose its output port channel, which are recognized by $0°$, $90°$, $180°$, and $270°$ turns in our representation. based on our definition, the local port is always recognized by $0°$. The other output ports are determined based on the counterclockwise relative rotation with respect to the position of input channel. One of the unique aspects of our idea is the rotation concept. Since in regular (non-fault tolerant) dimension-order routing, packets coming from one port are not forwarded to the same port (meaning 0 or 360 degree rotation), we used $0°$ for local port. For the case when the incoming header flit arrives from the east port, the north output port will be recognized by $90°$.

Using the header content, router decides how many rotation degrees are necessary to forward the incoming header flit to the expected output port. Table 1 demonstrates the idea of using two bits in order to implement the EA technique. In EA method the appropriate output port is chosen based on the header flit code and encoding table which are shown in Table 1. In this example the header flit selects the east output port, if its corresponding header flit code is "00."

EA routing technique needs two bits for each intermediate nodes regardless of the size of the network, while in NEA method the header size depends on the size of the network. The size of header flit in EA routing is still large by imposing two bits for each intermediate node.

3.3. OEA

OEA is the third source routing approach in which the header flits are compressed. In the EA technique, a header flit passes through all the intermediate routers in one dimension and then moves to the other dimension to reach its destination. The idea of OEA is to use the same encoding technique as EA method until the header changes its dimension on its way toward destination. Then the header flit has only two options, choosing the local port or traversing in the same direction. In OEA routing techniques one bit per hop is employed to navigate the header toward destination, once the header turns to the other dimension. This optimization is because of the property of the dimension order routing algorithms in which only one turn is permitted. Table 2 demonstrates the encoding table before and after header flit turns, where the difference between OEA and EA methods is highlighted once the header flit turns in the vertical direction. OEA enhances the additional overhead of the EA method by 25%. It also saves more switching power by having only one transition from "0" to "1" in the header flit after the header turns to move forward in the second dimension.

4. TagNoC Approach

The proposed hybrid routing method is presented in this section. The necessity of this technique is provided first and then it will be introduced in details in following subsections.

4.1. Motivation

The transition to a NoC design methodology empowers computer architects to optimize interconnection substrates in a modular, scalable, and easily-quantifiable manner [7]. As designers keep on optimizing NoC structures, enabling superior performance in the high bandwidth environment which they operate, multiple design trends have emerged. In some of the designs [27, 39, 15], router datapaths have been increasingly simplified through the reduction or elimination of buffering resources and pipeline stages. At the same time, router control logic complexity has increased as architects have added a huge logic to optimize performance. On the other hand, designs with pipeline architectures [30] are trying to increase the operating frequency of NoC routers. With the increase in number of cores, emerging routing algorithms are getting more complex [11]. The complexity of routing algorithm architectures aggregates as the number of connected cores in NoC increases to support both functionality and performance [29]. In both trends, eliminating routing latency helps shortening the clock period or the cycle counts.

The primary objective of TagNoC architecture is to propose an innovative scheme which overlaps the function of output port determinism by a new coding-based *Tag* architecture, router efficiency increases with saving the routing latency. The following sections elaborate the primary mechanism which enables this as well as the key architectural components necessary for implementation. TagNoC is a micro-architectural level method which deterministic routing algorithms of NoC can utilize it. TagNoC inherits the routing-specific characteristics of the target routing algorithm. For instance, TagNoC is deadlock and

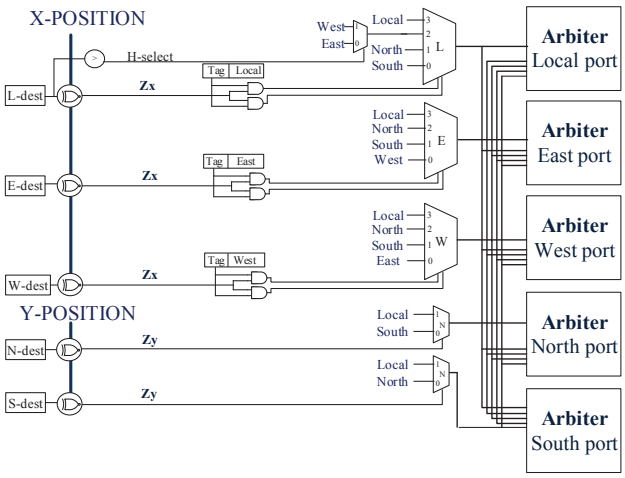


Figure 3: TagNoC architecture for a 5-port NoC router

Table 3: TagNoC control bits and corresponding rotation

(a) Horizontal direction

Z_x	**Tag[1]**	**Tag[0]**	**Rotation**
0	×	×	180°
1	0	1	90°
1	1	0	270°
1	1	1	0°

(b) Vertical direction

Z_y	**Tag[1]**	**Tag[0]**	**Rotation**
0	×	×	180°
1	×	×	0°

live-lock free, if the routing algorithm which is implemented with TagNoC has these properties as well. Turn model routing scheme, based on wormhole switching mechanism, are the most popular routing models utilized in NoC architectures. TagNoC supports all turn model routing algorithms including four well-known ones like XY, NF, WF and NL. Since XY routing algorithm is the reference model for most of the NoC related research, in this paper TagNoC coding values and corresponding multiplexer selectors are designed for this algorithm. However, they can be easily modified for any turn-based routing model based on their turn direction.

4.2. Architectural Details

Our proposed architecture of the routing component, is composed of XNOR gates comparison circuit and a multiplexer for each port (Figure 3). This method needs a negligible logic circuit inside the NI to generate the two *Tag* bits. Once the header has settled in the input buffer, X(Y) portion of destination address of the header flit is compared with the coordinates of the current router using XNOR gates. The output of comparison gates are Z_x and Z_y are used to determine if the flit has reached its expected horizontal or vertical coordinates, respectively. If the Z_x output of XNOR circuit for the X-POSITION is '0', then the packet traverses with $180°$ rotation. It means that if the input port is west (south) bound, the output port will be east (north) bound or vise versa, as shown in Table 3a. The Z_x will be '1' once header reaches the router with the same X-dimension of its destination address . Subsequently, *Tag* bits are examined to choose the appropriate direction based on the values of Table 3a. For instance, if the packet has entered from the east and the *Tag* is "01", then the south bound output port is chosen with $90°$ rotation. The Z_y will be checked after the header has turned and it will be set to '1' once the header has arrived at its destination, as shown in Table 3b. *Tag* bits will not be considered after the Z_y is equal to '1'. This method is flexible to support all the turn model based routing algorithms. The header of the packet is provided in the NI of the core and is forwarded to the connected router to navigate the whole packet towards the destination. A *Tag* is computed, just one time regardless of the network size, and appended to the coordinates of the destination router in the header of the packet.

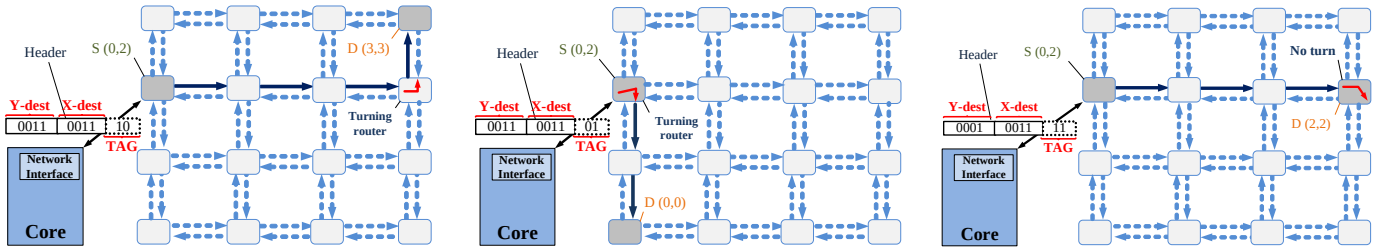
This operation is performed once the header settles in input buffer at a cost of simple comparison, bypassing the conventional routing function.

Figure 4 shows a partition of a 16-node 2D mesh, in which TagNoC routing method is applied for three scenarios. It illustrates the usage of *Tag* bits while packets need to turn up, down, or do not turn once they reach the X-dimension of the destination. This point is shown as turning point in Figure 4a, Figure 4b and as no turn in Figure 4c. The intermediate routers before the turning point only compare the X-dimension of destination with their own information. Once the packet reaches the turning (no turn) point, *Tag* bits are examined to decide if the packet has arrived to its destination or it needs to turn. The Y-dimension are compared after the packet has turned. In Figure 4a the Z_x signal gets '1' once the router reaches the turning point and then it turns. The *Tag* which is "10" makes the header to turn $270°$ as shown in Figure 4a. In Figure 4b the Z_x signal gets '1' at the source router, meaning source and destination have the same Y-dimension. Then the header turns $90°$ as indicated for the "01" *Tag* value in Table 3a. For Figures 4b and 4c, Z_y values are checked after the turn as described before. In Figure 4c, source and destination routers are on the same row; once Z_x gets the value of '1' there is no turn according to the "11" values of *Tag* as depicted in Figure 4c.

4.3. Scalability

NoC has emerged as a solution to address the communication demands of future many-core architectures due to its scalability, reusability, and parallelism in communication infrastructure [36]. Routing algorithm plays a major goal in scalability of NoC. The number of enforced bits in the header flit by a routing algorithm is the limiting factor. The size of header flit influences the phit[1] size which directly impacts the bandwidth of the network. If the size of phit is large, the limiting factor in

[1]phit (physical unit) is a unit of data that is transferred on a link in a single cycle.



(a) turn-up at turning point (b) turn-down at turning point (c) no-turn at turning point

Figure 4: TagNoC in a 16-node 2D mesh - Various position of Source and Destination

wire routing or bandwidth efficiency parameters are not satisfied.

Table 4: Header size comparison of proposed methods

Approach	Header size	Header bits
Baseline	$(\lceil \log_2 X \rceil + \lceil \log_2 Y \rceil)$	8 bits
NEA	$(\lceil \log_2 X \rceil + \lceil \log_2 Y \rceil) \times ND$	144 bits
EA	$2 \times ND$	36 bits
OEA	$2 \times (X-1) + (Y-1)$	27 bits
TagNoC	$(\lceil \log_2 X \rceil + \lceil \log_2 Y \rceil) + 2$	10 bits

Another critical factor in NoC is communication reliability. Any fault occurrence on the header flit may result in a misrouting [12] of the whole packet. The probability of fault occurrence on header flit grows as the header flit size increases, assuming there is a same BER[2] for both data and control flits. The extra routing information bits in the header flit which limits the scalability and decrease the reliability of NoC is a big challenge in source routing methods.

The required number of routing bits in header flits for all discussed approaches are presented in Table 4 where X and Y represent the horizontal and vertical dimensions of the network, respectively. The third column of the Table 4 demonstrates the corresponding header size of each method for a 10×10 NoC. *ND* term represents network diameter which is $(X+Y-2)$ for a mesh topology.

In EA method, two bits are needed for each intermediate node from the source to destination (see Section 4.) Comparing to NEA, EA method is more scalable but still depends on the network dimension. In comparison with EA, the header flit in OEA method requires even fewer bits. The reason is that in OEA method, two bits are needed for each horizontal movement while after the turn, only one bit is enough for each vertical movements. In TagNoC, regardless of NoC dimensions, only two extra bits (*Tag*) are used so the number of header flit bits for TagNoC technique is considerably smaller than the rest of proposed approaches especially for larger networks.

Figure 5 compares header size for all the source routing algorithms with XY algorithm and TagNoC approach for different

[2]The Packet Error Rate (PER) is the number of received data packets with errors divided by the total number of received packets. A packet is faulty, if at least one bit is erroneous. This assessment depends on the Bit Error Rate (BER) as a standard measure of the performance of a channel in the presence of fault.

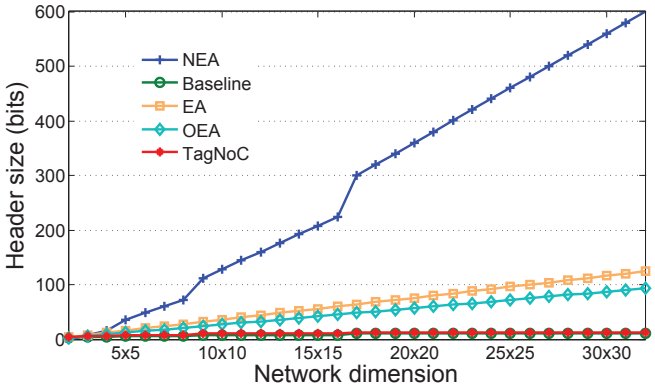


Figure 5: Routing overhead bits in the header flit vs. number of nodes in each dimension for different routing schemes

number of cores. In this figure, it is demonstrated that the size of header flit in TagNoC grows logarithmically which is almost the same as the baseline distributed XY routing with negligible constant amount of overhead.

5. Evaluation Methodology

The source routing algorithms and TagNoC approach which were discussed in Section 4 and baseline XY routing method are compared with each other in terms of performance, throughput, power consumption, area overhead, and frequency. NEA method is not considered for this evaluation because of its high overhead. The experimental framework is introduced first and experimental results are provided in following subsections.

5.1. Experimental Setup

A 16-node, 4×4 mesh network with 16-bit interconnection links is implemented in Verilog to obtain precise experimental results. The accuracy of the NoC model is verified by tracing the traverse of packets through the network with the help of SystemVerilog language for different traffic patterns. Both synthetic and application-level traffic analysis are considered in order to investigate the efficiency of the proposed hybrid routing approach under different performance constraints.

Routers contain five 128-bit buffers per input port and operate with the frequency of 1 GHz. Selected signals for the simulation are profiled after the simulator is warmed up and

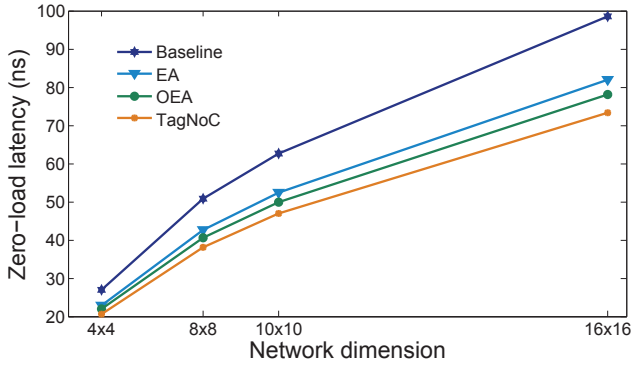


Figure 6: Zero-load latency versus different network size

the majority of routers are involved in packet transmissions as source or intermediate nodes in order to report accurate results.

Header flit arrival time is recorded until the packet leaves the network in order to measure performance. Throughput is defined as the rate at which packets are delivered by the network for a particular traffic pattern [8]. Performance, power and area of each architecture is obtained by a combination of cycle-accurate RTL router simulation, Verilog synthesis, and performing post-synthesis gate-level simulation. The Verilog synthesis is done by Synopsys Design Compiler with TSMC 40nm standard cell library and post-synthesis gate-level simulation is done by Synopsys PrimeTime in order to extract experimental results. Further details on synthetic and application performance modeling are presented in Subsection 5.2. Each network is designed to operate at its maximum frequency with the assumption that processor tiles operate on their own frequency domain asynchronously with respect to the interconnection network, similar to recent industry many-core chips [16].

5.2. Performance Analysis

Zero-load network latency and both synthetic and application-level traffic patterns are used in order to analyze impacts of proposed router architecture on the NoC. The zero-load network latency is widely used as a performance metric for traditional interconnection networks. Zero-load latency gives a lower bound on the average latency of a packet through the network in which a packet never contends for network resources with other packets [8]. Zero-load latency in this experiment is implemented by the injection rate of 0.1%. Zero-load latency does not depend on type of the generated traffic patterns as there is no contention among routers to access the interconnection resources. The impact of proposed architectures on zero-load latency for different number of nodes under the uniform random traffic pattern is plotted as shown in Figure 6 for baseline, EA, and TagNoC methods. All EA, OEA, and TagNoC methods require the same number of clock cycles with different clock cycle time since they have different routing unit architectures. Figure 6 shows that as the network size increases, the latency for all methods increases. The performance of source routing methods scales much better than the baseline architecture for a fixed network size. In a 100-node, 10×10 mesh network the zero-load latency of TagNoC technique is 26% less than baseline method.

Table 5: System configuration parameters

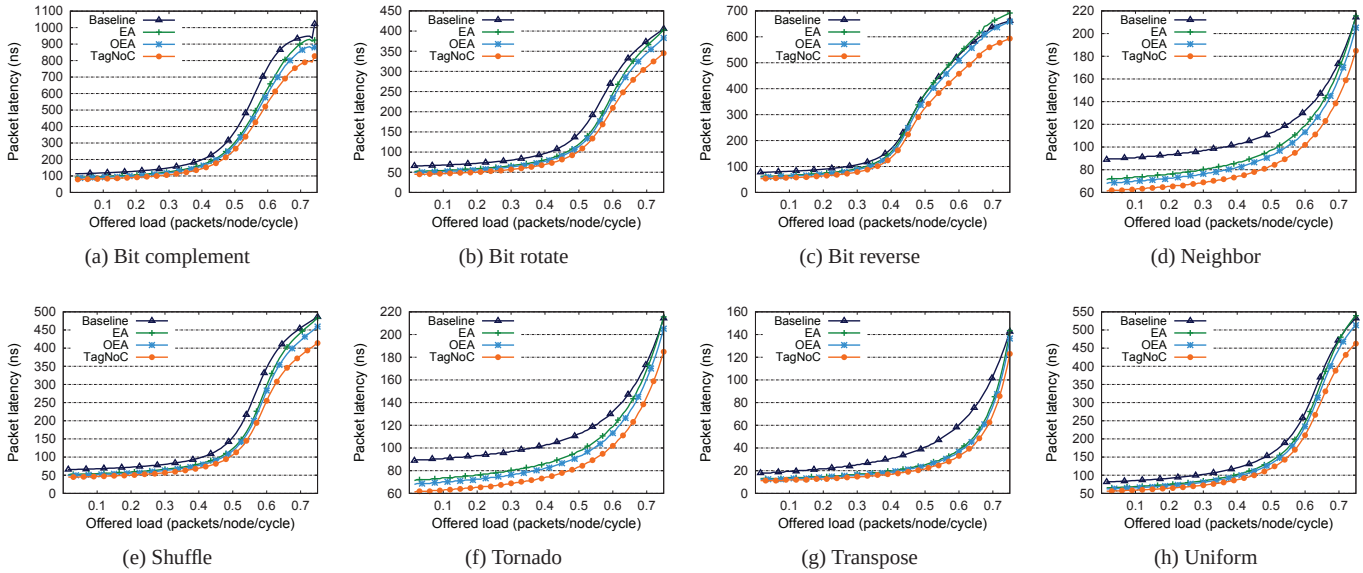
Parameter	Value
Cores	16
Topology	4x4 mesh
Processor	SPARC
L1 Cache I/D	64KB, 2-way, 3-cycle access
L2 Cache	Shared, 6-cycle bank access
Cache Coherence Protocol	MESI
Memory Access Latency	220 cycles
Packet Size	22 bytes
Flit Size	16 bits
Buffer Depth	8, 16-bit entries/port
Switching Scheme	Wormhole
Routing Algorithm	Dimension Ordered Routing

5.2.1. Synthetic Performance

Spatial distribution of messages in interconnection networks are considered using traditional synthetic traffic pattern to evaluate the latency of baseline and proposed routing approaches. These synthetic traffic patterns include Bit_complement, Bit_rotate, Bit_reverse, Neighbor, Shuffle, Tornado, Transpose, and Uniform which are explained in more details in [1, 33]. Packets with variable data flit sizes are injected to routers in this experiment for each of traffic patterns as the NoC designs are expected to support multiple applications. Figure 7 illustrates the average packet latency versus offered load for all the approaches under running synthetic traffics. Latency and offered load are reported in units of nanoseconds and packet/cycle/node, respectively. The performance of the EA, OEA, and TagNoC routing algorithms are better than the baseline method for the majority of synthetic traffics except in Bit_reverse traffic for higher injection rate in which baseline has better performance than EA method. The turning point of injection rate for the synthetic traffic patterns is between 0.35-0.55 packets/node/cycle (Figure 7.) With higher injection rates than the turning point $\approx$ 0.45 packets/node/cycle the average latency of all approaches under running synthetic traffics rises sharply until the network is saturated. However, in TagNoC approach the network saturated at higher injection rate as compared with other techniques, providing a better performance for all synthetic traffics.

5.2.2. Application Performance

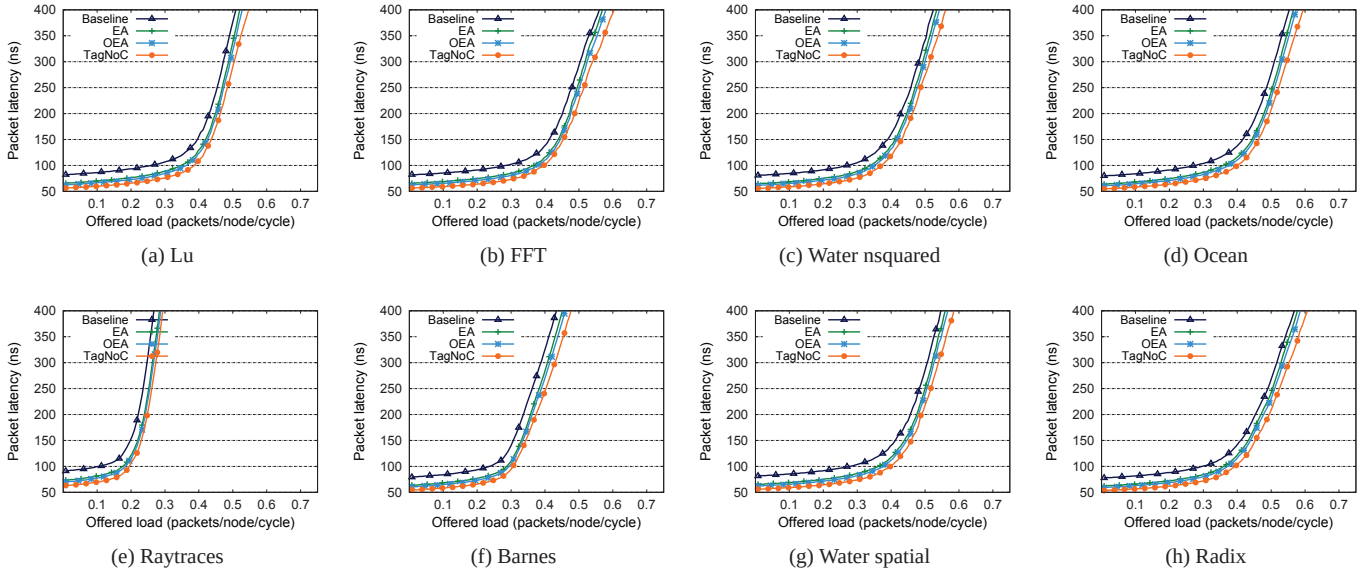
Application traces are obtained from the GEMS simulator [25] using the SPLASH-2 application benchmark suites [37]. Full simulation system parameters are reported in Table 5. The average packet latency versus offered load for the baseline, the TagNoC, and the source routing approaches under running SPLASH-2 application traffics are plotted in Figure 8. The reported average packet latency for SPLASH-2 traffics demonstrates similar results of performance reports as re-



(a) Bit complement (b) Bit rotate (c) Bit reverse (d) Neighbor

(e) Shuffle (f) Tornado (g) Transpose (h) Uniform

Figure 7: Average packet latency under synthetic traffic as a function of aggregate offered load for mesh network of 16 nodes



(a) Lu (b) FFT (c) Water nsquared (d) Ocean

(e) Raytraces (f) Barnes (g) Water spatial (h) Radix

Figure 8: Average packet latency under SPLASH-2 application traffic as a function of aggregate offered load for mesh network of 16 nodes

ported for synthetic traffics, although the turning point of diagrams for SPLASH-2 application traffics is between the offered load of 0.2-0.4 packets/node/cycle. TagNoC architecture has a better performance for all application traffics (Figure 8.) This is because packets are buffered for shorter time through intermediate routers and can be quickly routed, resulting in a network being able to tolerate higher injection rates. Better performance is reported for TagNoC method as it has a simpler routing decision circuit; resulting in shorter clock cycle time as compared to the others. Such a technique would increase the efficiency of the NoC to support cores with higher frequencies.

5.3. Power Analysis

Figure 9a shows dynamic power consumption under a 1 GB/s/node *fft* traffic load for a 16-node network with the injection rate of 0.5 packets/node/cycle. NoC with TagNoC router architecture is reported as the lowest power consumer as comparing to other approaches since it eliminates extra comparison circuits. However, total dynamic power consumption of the baseline, the TagNoC, and other source routing techniques do not increase since they all employ the same buffer, crossbar, and arbiter components. Dynamic power consumption of the management and routing unit of different approaches are compared separately in order to highlight improvements of TagNoC technique (Figure 9b.) The extra power consumption of NI to

generate the extra routing information bits is reported in Table 6 in terms of micro-watt (μW), which are negligible due to their tiny logic, as expected. It should be noted this extra power consumption is imposed once per packet only during packet injection into the router.

Table 6: NI power overhead for extra routing information computation

Power consumption (μW)	TagNoC	EA	OEA
Dynamic power	0.69	9.066	10.86
Leakage power	18.2	210.94	186.9
Internal power	2.42	23.2	20.43
Total	21.37	243.18	218.23

As it is shown in Figure 9b EA/OEA methods as source routing algorithms consume considerably less power than the baseline distributed routing. This is due to the elimination of routing logic in EA/OEA approaches. However, at router level it can be observed that EA/OEA methods consume slightly higher power than the baseline. The reason behind this observation is that in EA/OEA methods the number of header bits imposed by routing algorithm is more than baseline method which leads to more power consumption. It is shown that TagNoC architecture consumes power approximately 83% and 27% less than the baseline router and EA/OEA architectures.

5.4. Frequency Results

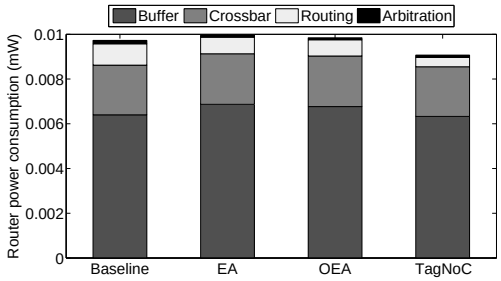
Table 7 summarizes the minimum clock cycle time for baseline and other methods, extracted using Synopsys Design Compiler. Due to shifting the header bits, EA and OEA routing circuits need higher clock cycle time in comparison with baseline method as shown in Table 7. TagNoC is reported as the fastest technique by 2%, 12%, and 6% less clock period comparing to baseline, EA, and OEA techniques.

5.5. Area Analysis

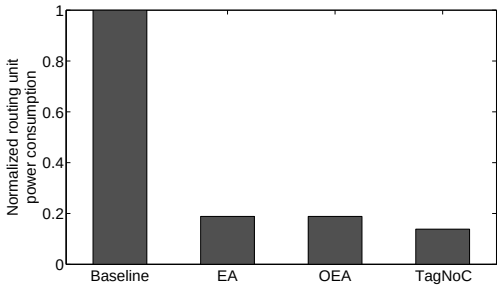
Area values of routing logic for all the baseline, EA, OEA, and TagNoC architectures are illustrated in Figure 10. Since they do not impact the buffer, arbiter, and crossbar components, there is no difference between them. The area of input buffer, crossbar switch, and arbiter for all the architectures are $9471\mu m^2$, $520\mu m^2$, and $1065\mu m^2$, respectively. As it is depicted in Figure 10, TagNoC routing logic occupies 53% less area than routing unit of baseline distributed router. In total, TagNoC router architecture incurs $138\mu m^2$ (2%) less silicon area than baseline router due to elimination of conventional routing unit. Overall, the total TagNoC router saves area by elimination of routing logic. The reason that EA and OEA methods seem to occupy approximately the same amount of logic as baseline is that these paradigms in implementation need to have a shift register to drop the current routing bits and push the rest to the front. So next router can process the correct routing information bits. This shifter imposes some area overhead.

Table 7: Router clock periods

Architecture	Clock period
Baseline	0.7466ns
EA	0.8196ns
OEA	0.7801ns
TagNoC	0.7328ns



(a)



(b)

Figure 9: (a) shows network dynamic power consumption under 1GB/s/node *fft* traffic, (b) illustrates the total power consumption for routing unit component of router architectures (power values are normalized with respect to the Baseline method)

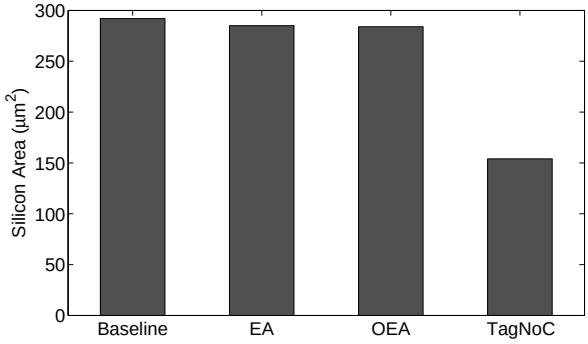


Figure 10: Routing logic area results (μm^2.)

6. Conclusion

Routing unit is one of the critical components of a NoC router architecture. Distributed routing method has been utilized in the past, but it requires a complex circuit which does not scale proportionally as the number of cores increases. Source routing

algorithm has been proposed to mitigate the hardware overhead of distributed routing techniques for larger networks. Three source routing algorithms for a NoC have been discussed as reference models for comparison, which are all based on turn-based model technique. Source routing algorithms result in a large amount of information overhead to the packet's header. The main goal of some of these approaches is to decrease the information overhead of source routing algorithm with compression while minimizing the routing unit circuit of intermediate router. This paper proposed TagNoC, as an on-chip network router architecture with novel hybrid routing approach which reduces the latency and power consumption at a fixed cost of information redundancy. TagNoC is a hybrid routing approach which combines the source and distributed routing methods together. In this work, TagNoC is compared with a baseline distributed XY routing algorithm and described source routing methods in terms of performance, power consumption, throughput, maximum frequency, and area overhead. TagNoC is the most efficient method which benefits from advantages of both source routing and distributed routing algorithms. In a 10×10 mesh network zero-load latency of TagNoC technique is around 33% less than baseline method. TagNoC has the best performance as compared to baseline and other proposed methods under running synthetic and application traffics for different injection rates while it only imposes two extra bits comparing to baseline. Also TagNoC is reported as the fastest technique by 2%, 12%, and 6% less clock period comparing to baseline, EA, and OEA techniques. The reported dynamic power consumption of the TagNoC router architecture is around 10% less than the baseline router architecture for a 4×4 mesh network, while occupying 53% less routing logic area than baseline distributed router. As a future work, TagNoC approach will be extended to support adaptive routing and other topologies like torus.

References

[1] J. H. Bahn and N. Bagherzadeh. A generic traffic model for on-chip interconnection networks. In *International Workshop on Network-on-Chip Architectures*, pages 22–29, 2009.

[2] L. Benini and G. De Micheli. Networks on chips: a new soc paradigm. *Computer*, 35(1):70–78, 2002.

[3] S. Borkar. Thousand core chips: A technology perspective. In *Proceedings of the 44th Annual Design Automation Conference*, pages 746–749, 2007.

[4] S. Borkar. The exascale challenge. In *VLSI Design Automation and Test (VLSI-DAT), 2010 International Symposium on*, pages 2–3, 2010.

[5] D. Brooks, R. Dick, R. Joseph, and L. Shang. Power, thermal, and reliability modeling in nanometer-scale microprocessors. *Micro, IEEE*, 27(3):49–62, 2007.

[6] G.-M. Chiu. The odd-even turn model for adaptive routing. *IEEE Trans. Parallel Distrib. Syst.*, 11(7):729–738, July 2000.

[7] W. Dally and B. Towles. Route packets, not wires: on-chip interconnection networks. In *Design Automation Conference, 2001. Proceedings*, pages 684–689, 2001.

[8] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[9] M. Danashtalab and M. Palesi. Basic concepts on on-chip networks. In M. Palesi and M. Daneshtalab, editors, *Routing Algorithms in Networks-on-Chip*, pages 1–18. Springer New York, 2014.

[10] F. Dubois, A. Sheibanyrad, F. Petrot, and M. Bahmani. Elevator-first: A deadlock-free distributed routing algorithm for vertically partially connected 3d-nocs. *Computers, IEEE Transactions on*, 62(3):609–615, 2013.

[11] M. Ebrahimi, H. Tenhunen, and M. Dehyadegari. Fuzzy-based adaptive routing algorithm for networks-on-chip. *Journal of Systems Architecture*, 59(7):516 – 527, 2013.

[12] A. Eghbal, P. M. Yaghini, H. Pedram, and H. R. Zarandi. Designing fault-tolerant network-on-chip router architecture. *International Journal of Electronics*, 97(10):1181–1192, 2010.

[13] J. Flich, S. Rodrigo, and J. Duato. An efficient implementation of distributed routing algorithms for nocs. In *Networks-on-Chip, 2008. NoCS 2008. Second ACM/IEEE International Symposium on*, pages 87–96, 2008.

[14] C. Glass and L. Ni. The turn model for adaptive routing. In *Computer Architecture, 1992. Proceedings., The 19th Annual International Symposium on*, pages 278–287, 1992.

[15] M. Hayenga and M. Lipasti. The nox router. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44 '11, pages 36–46, New York, NY, USA, 2011. ACM.

[16] J. Howard, S. Dighe, S. Vangal, G. Ruhl, N. Borkar, S. Jain, V. Erraguntla, M. Konow, M. Riepen, M. Gries, G. Droege, T. Lund-Larsen, S. Steibl, S. Borkar, V. De, and R. Van Der Wijngaart. A 48-core ia-32 processor in 45 nm cmos using on-die message-passing and dvfs for performance and power scaling. *Solid-State Circuits, IEEE Journal of*, 46(1):173–183, Jan 2011.

[17] A. Jantsch and H. Tenhunen, editors. *Networks on Chip*. Kluwer Academic Publishers, Hingham, MA, USA, 2003.

[18] Y. B. Kim and Y.-B. Kim. Fault tolerant source routing for network-on-chip. In *Defect and Fault-Tolerance in VLSI Systems, 2007. DFT '07. 22nd IEEE International Symposium on*, pages 12–20, 2007.

[19] S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani. A network on chip architecture and design methodology. In *VLSI, 2002. Proceedings. IEEE Computer Society Annual Symposium on*, pages 105–112, 2002.

[20] I. Loi, F. Angiolini, and L. Benini. Synthesis of low-overhead configurable source routing tables for network interfaces. In *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, pages 262–267, 2009.

[21] P. Lotfi-Kamran, A. Rahmani, M. Daneshtalab, A. Afzali-Kusha, and Z. Navabi. Edxy a low cost congestion-aware routing algorithm for network-on-chips. *Journal of Systems Architecture*, 56(7):256 – 264, 2010.

[22] Z. Lu and A. Jantsch. Trends of terascale computing chips in the next ten years. In *ASIC, 2009. ASICON '09. IEEE 8th International Conference on*, pages 62–66, 2009.

[23] L. Ma and Y. Sun. On-chip network design automation with source routing switches. *Tsinghua Science and Technology*, 12(1):77–85, 2007.

[24] R. Marculescu, U. Ogras, L.-S. Peh, N. Jerger, and Y. Hoskote. Outstanding research problems in noc design: System, microarchitecture, and circuit perspectives. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 28(1):3–21, 2009.

[25] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood. Multifacet's general execution-driven multiprocessor simulator (gems) toolset. *SIGARCH Comput. Archit. News*, 33(4):92–99, Nov. 2005.

[26] S. Mubeen and S. Kumar. Designing efficient source routing for mesh topology network on chip platforms. In *Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on*, pages 181–188, 2010.

[27] R. Mullins, A. West, and S. Moore. Low-latency virtual-channel routers for on-chip networks. In *Proceedings of the 31st Annual International Symposium on Computer Architecture*, ISCA '04, pages 188–, Washington, DC, USA, 2004. IEEE Computer Society.

[28] A. Munir, S. Ranka, and A. Gordon-Ross. High-performance energy-efficient multicore embedded computing. *Parallel and Distributed Systems, IEEE Transactions on*, 23(4):684–700, 2012.

[29] P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh. Performance evaluation and design trade-offs for network-on-chip interconnect architectures. *Computers, IEEE Transactions on*, 54(8):1025–1040, 2005.

[30] S. Park, T. Krishna, C.-H. Chen, B. Daya, A. Chandrakasan, and L.-S. Peh. Approaching the theoretical limits of a mesh noc with a 16-node chip prototype in 45nm soi. In *Proceedings of the 49th Annual Design*

Automation Conference, DAC '12, pages 398–405, New York, NY, USA, 2012. ACM.

[31] J. Pontes, M. Moreira, F. Moraes, and N. L. V. Calazans. Hermes-aa: A 65nm asynchronous noc router with adaptive routing. In *SOC Conference (SOCC), 2010 IEEE International*, pages 493–498, 2010.

[32] I. Pratomo and S. Pillement. Gradient-an adaptive fault-tolerant routing algorithm for 2d mesh network-on-chips. In *Design and Architectures for Signal and Image Processing (DASIP), 2012 Conference on*, pages 1–8, 2012.

[33] A.-M. Rahmani, A. Afzali-Kusha, and M. Pedram. A novel synthetic traffic pattern for power/performance analysis of network-on-chips using negative exponential distribution. *Journal of Low Power Electronics*, 5(3):396–405, 2009.

[34] S. Rodrigo, S. Medardoni, J. Flich, D. Bertozzi, and J. Duato. Efficient implementation of distributed routing algorithms for nocs. *Computers Digital Techniques, IET*, 3(5):460–475, 2009.

[35] S. W. Son, K. Malkowski, G. Chen, M. Kandemir, and P. Raghavan. Reducing energy consumption of parallel sparse matrix applications through integrated link/cpu voltage scaling. *J. Supercomput.*, 41(3):179–213, Sept. 2007.

[36] H. Tenhunen and A. Jantsch. *Networks on chip*. Kluwer Academic Publishers, 2003.

[37] S. Woo, M. Ohara, E. Torrie, J. Singh, and A. Gupta. The splash-2 programs: characterization and methodological considerations. In *Computer Architecture, 1995. Proceedings., 22nd Annual International Symposium on*, pages 24–36, 1995.

[38] P. M. Yaghini, A. Eghbal, H. Pedram, and H. R. Zarandi. Investigation of transient fault effects in synchronous and asynchronous network on chip router. *Journal of Systems Architecture*, 57(1):61 – 68, 2011.

[39] G. Yuan, A. Bakhoda, and T. Aamodt. Complexity effective memory access scheduling for many-core accelerator architectures. In *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pages 34–44, 2009.

[40] J. Zhang and H. Gu. A partially adaptive routing algorithm for benes network on chip. In *Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on*, pages 614–618, 2009.