A Holistic Approach Towards Intelligent Hotspot Prevention in Network-on-Chip-Based Multicores

Vassos Soteriou, *Member, IEEE*, Theocharis Theocharides, *Senior Member, IEEE*, and Elena Kakoulli, *Student Member, IEEE*

Abstract—Traffic hotspots, a severe form of network congestion, can be caused unexpectedly in a network-on-chip (NoC) due to the immanent spatio-temporal unevenness of application traffic. Hotspots reduce the NoC's effective throughput, where in the worst-case scenario, network traffic flows can be frozen indefinitely. To alleviate this problematic phenomenon several adaptive routing algorithms employ online load-balancing schemes, aiming to reduce the possibility of hotspots arising. Since most are not explicitly hotspot-agnostic, they cannot completely prevent hotspot formation(s) as their reactive capability to hotspots is merely passive. This paper presents a pro-active Hotspot-Preventive Routing Algorithm (HPRA) which uses the advance knowledge gained from network-embedded artificial neural network-based (ANN) hotspot predictors to guide packet routing in mitigating any unforeseen near-future hotspot occurrences. First, these ANN-based predictors are trained offline and during multicore operation they gather online statistical data to predict about-to-be-formed hotspots, promptly informing HPRA to take appropriate hotspot-preventive action(s). Next, in a holistic approach, additional ANN training is performed with data acquired after HPRA interferes, so as to further improve hotspot prediction accuracy; hence, the ANN mechanism does not only predict hotspots, but is also aware of changes that HPRA imposes upon the interconnect infrastructure. Evaluation results, including utilizing real application traffic traces gathered from parallelized workload executions onto a chip multiprocessor architecture, show that HPRA can improve network throughput up to 81 percent when compared with prior-art. Hardware synthesis results affirm the HPRA mechanism's moderate overhead requisites.

Index Terms—Multiprocessor interconnection, neural network hardware, on-chip network, ultra-scale integration

1 Introduction

State-of-the-art Chip Multiprocessors (CMPs) such as Intel's single-chip cloud computer (SCC) [12], employ networks-on-chips (NoCs) [5] to handle the high-throughput and low-latency communication demands among the various on-chip tiles. NoCs scale efficiently with topology sizing, and their properties of modularity, replicability, interoperability, ability to tolerate faults, and achieve energy efficiency, among other attributes, aid in quickly designing, testing, and verifying ultra high-performance multicore computing systems.

In general-purpose multicores traffic patterns of applications are inherently highly-varying and unpredictable across time and space, often described as bursty or "jittery," a phenomenon that can cause elevated traffic congestion along the NoC topology. These congestion occurrences, conventionally referred to as *hotspots*, can drastically reduce the performance of the NoC *temporally* at instances and *spatially* across a NoC topology. Conceptually, the cause of hotspots is the potential presence of a widening gap between packetized message production and consumption, where superimposed packet streams that compete in acquiring limited network resources, such as channels and buffers, causes prolonged contention among these routed packets [20]. Indeed, a single router module can cause a hotspot, that can in turn propagate across network regions, while hotspots can arise even with communication links of theoretically infinite bandwidth and with routers containing buffering queues of infinite capacity [2]. Further multicore system functional particularities can produce hotspots: lack of traffic balancing under oblivious routing, inefficient flow-control, non-optimal application mapping, needless application migration, and due to increased network resource acquisition requests that occur dynamically and unexpectedly [19].

Most NoCs employ wormhole flow-control (WFC) [6] to attain efficient message transportation; however, the spreading of in-transit flits (flow-control digit) in a pipelined mode across several routers, intensifies the detrimental effect of hotspots, as it produces backpressure at upstream buffers causing them to quickly fill-up in a domino-style fashion. Hence, a hotspot(s) can concurrently span several portions of the topology, causing further elevated message contention to propagate spatially across several routers, that may lead to eventual irreversible traffic blockage, eventually forcing the entire NoC to stall indefinitely. Hotspot formations are especially unpredictable in general-purpose multicores such as CMPs where application patterns are highly spatio-temporally variable [34].

Even under load-balancing adaptive routing functions substantial effective throughput degradation in a NoC can be observed [15]. The development of congestion-management techniques as a means to safeguard the

- *V. Soteriou and E. Kakoulli are with the Department of Electrical Engineering, Computer Engineering, and Informatics, Cyprus University of Technology, Lemesos, Cyprus. E-mail: vassos.soteriou@cut.ac.cy.*
- *T. Theocharides is with the Department of Electrical and Computer Engineering, University of Cyprus, Nicosia, Cyprus. E-mail: ttheocharides@ucy.ac.cy.*

Manuscript received 30 Apr. 2014; revised 27 Feb. 2015; accepted 1 Mar. 2015. Date of publication 19 May 2015; date of current version 10 Feb. 2016. Recommended for acceptance by J. D. Bruguera.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2015.2435748

scalability of NoCs and hence the performance sustainability of their hosting general-purpose CMPs, has been identified as a major research challenge in laying the roadmap of future multicore chip designs [19]. Current techniques such as dynamic congestion management in the form of adaptive routing protocols [6], [8], [13], [17], [18], [27] and application scheduling [19] in an attempt to sustain high NoC throughput in the presence of hotspot-causing traffic are not always sufficient; NoC congestion is a complex and unpredictable phenomenon.

In this paper we present a new congestion-preventive pro-active routing function, termed hotspot-preventive routing algorithm (HPRA). HPRA is not confined to *passively* measuring current network statistics, such as link and buffer utilization rates (i.e., normalized rates of usage and occupancy, respectively), in attempting to *reactively* balance-out traffic to sustain or even improve network throughput, a commonly exercised performance-targeted strategy employed in most existing routing algorithms [6]. Instead, HPRA *pro-actively* prevents the unanticipated formation of NoC hotspots that may occur in the near future during multicore operation. This pro-active hotspot prevention mode is achieved with the use of advance information sourced with the use of artificial intelligence (AI) principles that are utilized at runtime to continuously predict the formation of network traffic hotspots. AI principles are chosen because of their adaptability to changing traffic conditions and their ability to learn about small network spatio-temporal variations which can lead to online congestion, and hence build on improving their ability to forecast the next hotspot occurrence(s) in advance. Next, in extending the performance envelope even further, HPRA differentiates between hotspot-bound and non-hotspot-bound traffic online, as dictated by the ANN-based predictor. Under HPRA, the former traffic category can be temporarily throttled just before its network ingress, depending upon the current network status, so as to prevent further congestion buildup and attenuate hotspots, while the latter non-adversarial traffic category enjoys a ceaseless load-balanced flow in the NoC.

In particular, this paper extends our previous work [16] by integrating the HPRA algorithm alongside our original ANN hotspot prediction mechanism first proposed in [15]. This ANN mechanism received as inputs the buffer utilization rates as network load-level indicators, from each monitored NoC router, and performed pattern recognition, predicting the formation of hotspots throughout the on-chip network (OCN). HPRA, thus, is able to utilize this knowledge to efficiently reroute network messages, limiting the number of occurring hotspots, and subsequently improving the NoC's overall performance.

During this integration stage, the ANN training already in place, is fused with the utilization rates resulting from the application of HPRA across the on-chip network. The data fusion involved in training the ANN with pre-HPRA buffer utilization and post-HPRA buffer utilization rates, enables the ANN to recognize traffic patterns that are directly resulted by invoking the HPRA mechanism after a prediction for a hotspot has already been made; as such, the ANN is explicitly trained not to confuse such occurrences as new hotspots. The impact of this, while marginal with respect to performance, implies lessening the power consumed, due to the reduction in the false-positive predictions observed in our initial work [15]. Moreover, this paper comprehensively estimates all necessary hardware overheads that such a mechanism would require to be deployed. Furthermore, in cases where the HPRA mechanism ends up responsible for hotspot formation(s) due to its own message rerouting actions, the ANN is also trained to attempt to detect such cases and re-invoke HPRA accordingly. The direct impact of this specific contribution is the reduction of power consumption related to the HPRA algorithm by 7.8 percent, due to the lessening of false-positive predictions, thereby enabling HPRA to be invoked less frequently. Additionally, the base ANN hardware mechanism presented here has been redesigned with emphasis in reducing the hardware overheads, by performing an accuracy veruss overhead design-space exploration, the results of which have reduced the original ANN overheads by a further overall 4 percent, with negligible impact onto the hotspot prediction accuracy.

While our initial work featured training of the ANN using synthetic hotspot traffic (in one step), this work also features further training performed with data acquired after our hotspot prevention algorithm interferes (as the second step). Hence, this work presents a **holistic** hotspot forecasting and preventing mechanism, where the ANN mechanism not only provides the HPRA algorithm with forecasting data, but also is itself aware of the changes that the HPRA algorithm imposes upon the interconnect infrastructure. Details of the ANN mechanism can be found in our previous work [15], and as such, along with the HPRA algorithm, we only present the new training scheme and its associated benefits in relation to the ANN mechanism.

The proposed HPRA mechanism is designed with manageable hardware overheads that account to a 10.3 percent overall network overhead. The ANN mechanism is designed as an independent processing element (PE) that is embedded into the base NoC topology of a multicore system. Evaluation results across two adversarial synthetic traffic patterns, and application traffic traces gathered from the TRIPS CMP [25], show that HPRA can reduce network latency, and also improve network throughput by up to 81 percent, when compared with several existing state-of-the-art congestion-aware routing functions (see Section 5). In addition, the use of HPRA across two 2D mesh-based network sizes shows its scalability and versatility.

Following, Section 2 presents a survey of related work. Section 3 introduces ANNs, and outlines the proposed ANN architecture and the data it processes, and describes our ANN prediction mechanism and associated hardware optimizations as opposed to the ANN-based predictor originally proposed in [15]. Next, Section 4 provides details of HPRA's implementation. Following, Section 5 outlines our experimental setup and evaluates the performance of HPRA against prior-art, while Section 6 presents HPRA optimized hardware synthesis results. Finally, Section 7 concludes this paper.

2 Background and Related Work

Hotspot management schemes pertaining to the field of NoC-based multicores are classified as: (1) implicit approaches, focusing on network congestion reduction via

workload load-balancing, and (2) explicit approaches specifically geared towards reducing the negative impact of hotspots upon a NoC's performance. Both said categories include methodologies spanning traffic-agnostic or adaptive routing algorithms (ARAs), fair and efficient flow-control, and targeted application mapping [19]; here we mainly focus on the former field of study.

The schemes under category (1) are equally applied to off- and on-chip interconnection networks; survey works such as [6], [19] cover the extensive related literature. Load-balancing selection functions which mainly account for de-centralized online statistical gathering to direct routing decisions, aim at exploring the path diversity which is inherent in NoC topologies, mostly in the form of 2D meshes, distributing uneven traffic among alternative lightly-loaded network topology-spanning routing paths. Their main goal is to help sustain the effective network throughput at reasonable levels under adversarial (i.e., highly unevenly distributed) spatio-temporal traffic patterns, but do not specifically manage traffic hotspot formations occurring in the on-chip network.

We next focus upon load-balancing routing functions applicable to the on-chip domain. The balanced adaptive routing protocol (BARP) [17] uses hybrid local-global network congestion information to distribute traffic evenly among the shortest routing paths to avoid congestion emergencies. Next, DyXY [18] provides adaptive routing based on congestion conditions measured in the topological proximity of a NoC router, enforcing unique shortest paths. The work by Gratz et al. [10] aims to enhance load-balancing, using a technique that measures traffic levels in regions beyond the locality of adjacent network routers to direct a regional congestion-aware routing scheme. Next, [23] proposes global congestion awareness (GCA) where global link state and congestion data is "piggybacked" in packet headers among routers, so as to guide routing efficiently react to changing overall network conditions. Qian et al. [22] propose a traffic-aware routing algorithm, that, in tandem with hub (regional) routers and express channels, tunes routing in accordance with online congestion metrics. Further, the work in [28] presents a routing algorithm that supports multiple concurrent applications in a congestion propagation NoC, where destination-based adaptive routing (DBAR) leverages both local and global network information to lead towards accurate congestion estimation. Next, the authors in [24] propose a minimal Destination-based Adaptive Routing (DAR) strategy where every node estimates the delay to every other network node, with routing decisions based on these per-destination delay metrics. Following, [20] proposes a memory-less switch for NoCs that misroutes packets to non-ideal routes during instances of elevated localized congestion levels. DyAD [13] adopts buffer occupation statistics and accordingly exchanges flags between neighbors to signal congestion and hence act accordingly (deterministically or adaptively) to increase NoC throughput. Finally, [31] uses source routing with QoS session establishment and global traffic monitoring to adaptively direct packets choose the least congested path(s). All above schemes merely *react* to already elevated message delivery delays, either locally (e.g., [13], [18], [22]), semi-globally (e.g., [10]), globally (e.g., [23], [31]), hybrid locally-globally (e.g., [17], [28]), or with the use of overlay networks (e.g., [10], [24]) or dedicated hardware (e.g., [20], [22]) to exchange route load data among router nodes, to lead the NoC adjust its response before further delays take effect; however, unlike HPRA, they all exploit no advance information about potential hotspots so as to *proactively* direct NoC congestion counter-response(s).

Most of the explicit hotspot management efforts under category (2), both in the off- and on-chip interconnect domains, focus on reducing the number of routed packets destined to highly utilized routers. All such schemes assume that the spatial locations of hotspots are either (a) *known a priory* and hence the hotspot management techniques *pro-actively* aim at reducing possible hotspot formations, or (b) they spatially and temporarily *respond reactively* to reduce the possibility of near-future hotspot occurrences. Various such hotspot prevention works fall under the domain of large-scale interconnected computers. The scheme in [11] discards hotspot-bound packets which are later re-transmitted to their destinations, while the technique in [1] reacts temporally and throttles hotspot-destined traffic at router injection ports. Next, the work in [33] presents a global-knowledge-based, self-tuned, congestion control technique that prevents network saturation at high loads, while [26] presents an analytical model of fully-adaptive wormhole routing in k-ary n-cubes in the presence of hotspot traffic. As NoC microarchitectures are resource-limited, most of the hotspot-alleviating schemes applicable to off-chip networks are unsuitable for NoCs. In the NoC field, hotspot prevention mostly concerns the use of adaptive routing which aims in guiding packets bypass hotspots, such as the work in [7] which sends control packets up-front to gather data about various routing paths, update routing tables, and then direct traffic towards lightly-used paths. Lastly, Walter et al. [34] use an end-to-end credit-based allocation technique to regulate hotspot-destined traffic flow.

3 ANNs: Information Processing and Prediction

An artificial neural network (ANN) is an information processing paradigm, inspired by the way biological neuron systems process information, and is composed of a large number of densely interconnected nodes (neurons) [14]. Neurons work in unison to solve specific problems. A neuron takes a set of inputs and performs a multiplication and accumulation operation, between each input, and its associated weight value, which is determined during the training stage. The results are accumulated until all the inputs are processed. If the accumulated result meets a preset threshold value (also determined during training), then it is used to compute the neuron output, based on the so-called *activation function*. The neuron output is then propagated to the neurons of the next layer which perform the same operation with the newly set of inputs and their own weights. This is repeated for all ANN layers.

ANNs learn by training, and are typically trained for specific usage in applications such as pattern recognition or data classification. They have also been successfully used as prediction and forecasting mechanisms in several areas, as

they are able to determine hidden and strongly non-linear dependencies, even with significant noise found in the data set [14]. Hence, their ability to identify data patterns renders them ideal for forecasting scenarios, where they can be used to "observe" data and the associated data changes within preset time intervals (i.e., discrete-time sampling), and identify a situation that is about to emerge based on the observed data patterns. ANNs have already been adopted in computer architecture, as branch prediction mechanisms, and in several areas such as forecasting mechanisms in stocks and other prediction applications [14]. While this mechanism could be implemented as a subroutine in the host operating system or middleware, we choose its integrated hardware implementation to facilitate real-time computation of our forecasting algorithm; this is vital to a holistic approach, as the hardware-based ANN mechanism can provide immediate results related to the status of the underlying NoC interconnect to the HPRA mechanism, which can then also react in real-time. As we are targeting a comprehensive response within the order of a few hundred cycles, the hardware implementation ensures that both the forecasting and HPRA responses are completed within this time constrain. Also, a neural network can be realized in hardware by using interconnected neuron hardware computation engines, each of which is composed by a multiplier-accumulator (MAC) unit and a custom computational unit which implements the activation function. Training weights are held in memory, but overall, the complexity of the hardware lies in the interconnection structure; fortunately, the discrete operation of the ANN allows for effective resource-sharing and pipelining, hence for small ANNs the neuron complexity is not an issue [14] and the interconnect can be addressed by efficient dataflow and time-division multiplexing protocols.

In our previous work [15], we introduced the concept of using ANNs as a hotspot forecasting mechanism that can be effectively integrated within the underlying NoC infrastructure using dedicated hardware. In contrast to other forecasting mechanisms, ANNs offer significantly better accuracy when dealing with non-linear forecasting models, something also observed in NoC hotspots. We illustrated how a generic ANN could be implemented in hardware, and how ANNs can be designed to surveil NoC traffic and predict hotspot formations, and provided the necessary implementation details. Moreover, we illustrated how a base ANN engine that monitors a 4×4 mesh NoC, can be designed to monitor a variety of NoC sizes and topologies, and how it can scale to facilitate large NoCs, through a combination of base ANNs and coordinated voting mechanisms. As such, all relevant details, trade-offs and design parameters such as hardware overheads vs. prediction accuracy, ANN scalability issues, ANN configuration and size, can be found along with further discussion related to the details of the ANN architecture in our previous works [15], [16]; here we only provide the specific implementation details concerning the NoC targeted in this work (8×8 mesh). As the focus in this work is the routing algorithm, we only detail how the base ANN mechanism can be further optimized; we present how piece-wise linear approximation (PWL) [32] can be used instead of dedicated on-chip RAM, further reducing overheads without compromising the accuracy of the ANN predictions.

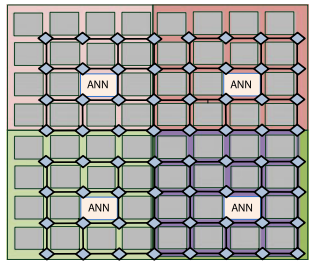


Fig. 1. ANN monitoring system: Four base ANN modules each monitoring a 4×4 region in an 8×8 two-dimensional mesh NoC topology.

HPRA, therefore, utilizes an ANN system, implemented in hardware, and can accurately detect and inform the underlying interconnect system about potential hotspots, providing the router node Cartesian coordinates of all the predicted hotspots in the network. Hence, HPRA uses the forecast given by the ANN, to *pro-actively* regulate network traffic, in an effort to prevent hotspots from forming. The ANN can subsequently be trained and configured to monitor the network traffic in discrete time intervals, and detect *pro-actively* whether a traffic pattern observed indicates a high probability of a hotspot forming within the network. The ANN has two operating modes: (1) off-line training (in two steps, where the weights are being computed), and (2) on-line prediction (where the computed weights are programmed into the ANN which uses the real-time traffic to predict hotspot formations). The ANN designed in this work adopts to an 8×8 mesh architecture; thus, the overall network uses four base ANN engines as Fig. 1 shows, and a sequence of OR-based voting mechanisms for the border neurons (shown in Fig. 2). We refer readers to our previous work [15] that explains how these parameters are chosen. The base ANN thus samples the average buffer utilization (ABU) values for each router port, for all routers monitored by the base ANN in 50-cycle intervals, and forecasts potential hotspot formations across the routers that it monitors. Given that hotspots are typically a situation spanning across more than one router, and given that the base ANN monitors a 4×4 region, the border routers between each region are considered to be hotspots if at least one of their neighboring routers is also considered to be a hotspot. The ANN outcome is propagated through dedicated links, belonging to an overlay network that is separate from the underlay data network, to all routers within the monitored region during each discrete interval, and it is used by the hotspot-preventive HPRA mechanism. The same links as the ones used to carry the free virtual channel count and average buffer utilization metrics needed for HPRA's routing decisions (a total of 6 bits in our case; see Section 4 and Section 6 for details) are also used to carry the utilization rates from each router to the ANN during each interval, thus minimizing the overall hardware overheads.

3.1 ANN Hardware Optimizations

One of the relatively large sub-units in terms of area for each ANN neuron module is the memory required to store the activation function (hyperbolic tangent in our case). In our initial approach [15], we used a shared SRAM-based look-up table (LUT). However, after exploring the impact of the accuracy of the activation function output on the overall

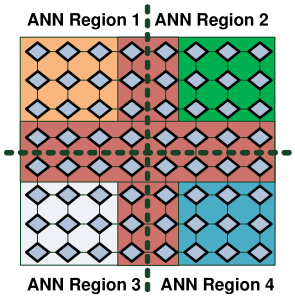


Fig. 2. ANN monitoring fragmented into four quadrants each covering a 4×4 region in an 8×8 2-dimensional mesh NoC topology. The boundary routers (dark red region) of each quadrant participate in the voting process.

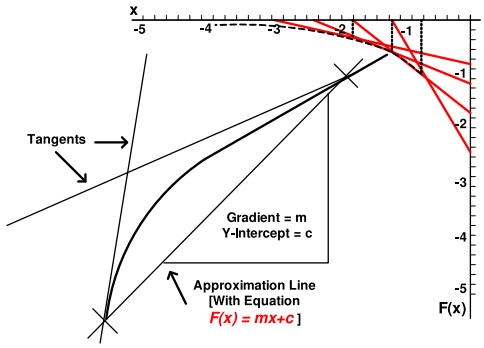


Fig. 3. Piece-wise linear function example.

accuracy of the ANN, we determined that instead of a dedicated LUT, we could implement the activation function using piece-wise linear approximation instead [32].

PWL approximates a non-linear function (i.e., quadratic) using a series of linear tangents to the function [3], [4]. The objective is to estimate $F(x)$ around x by treating the function section around x as a straight line instead. The function to be approximated is sampled every n points, and at each point, the function is approximated using the linear tangents that intersect with that particular line. Two neighboring points form a sampling interval line (Fig. 3), within which all values of the x-axis yield the approximate value of the function that now lies on the approximation line instead of the quadratic function. Each function segment is approximated using the tangents at the two points that construct each segment using simple geometric triangle properties (Fig. 3 magnified region). Therefore, as the equation of a straight line can be defined as $F(x) = mx + c$, where m is the gradient of the line, and c is the y-intercept, what we need to store for each approximation line, are the values of m and c. When the input (x) arrives, it is multiplied with the gradient (m) and the result is added to the y-intercept (c), which yields the corresponding approximate output of the function, that lies on the approximation line.

As shown in Fig. 3, $F(x)$ therefore becomes linear for each segment, and can then be represented by $F(x) = mx + c$. Only the values of m and c need to be stored to perform the approximation in hardware, with few registers used to store and retrieve these values. The remaining hardware operations involve the addition and the multiplication involved in the linear equation. Moreover, choosing tangent gradients as powers of two eliminates the need for multiplication, making the operation a left shift that further simplifies the implementation. Using one PWL unit to approximate the activation function instead of an SRAM-based LUT, reduces the overall ANN hardware mechanism overheads from 4.2 percent reported in [16] to 3.1 percent. It must be emphasized that the prediction accuracy of the ANN mechanism was negligible (less than 0.12 percent). The new approach also saves power, reducing the computation wattage required by almost 7 percent (mostly leakage power).

3.2 ANN Training with HPRA

In our initial approach in proposing the use of an intelligent mechanism for dynamic NoC hotspot forecasting, we utilized synthetic traffic benchmarks, and used buffer utilization rates in discrete time intervals, and supervised learning to train the ANN. We then used the trained ANN on "regular" traffic traces, both synthetic and those of real-world applications (TRIPS [25], see Section 5.3), and measured its forecasting accuracy on the hotspots that were predicted on those traces. However, in a holistic scenario such as the one presented in this paper, the ANN will be used in conjunction with the HPRA mechanism. Therefore, upon implementing the HPRA algorithm (along with the ANN with the initial training data), we optimize the ANN training in order to reflect the impact of the HPRA mechanism in the newly resulted buffer utilization data. As HPRA comes into effect, obviously the previous system behavior (that was reflected in the original training) is no longer representative; instead, the HPRA mechanism effectively changes the behavior of the system to control and possibly prevent hotspots from being formed.

There are two issues stemming from this. First, as HPRA attempts to prevent a hotspot from happening, the resulting buffer utilization rates that the ANN receives, might be misjudged for an about-to-be hotspot formation (since the resulting data patterns were never before seen by the ANN). Second, while HPRA is proven to be effective, it is not guaranteed that the adjustments it makes inside the network will not result in a hotspot formed somewhere else that is the direct result of HPRA. While the initial ANN training should cover these cases (as a hotspot formation, for an ANN, is merely an observed pattern, independent of what led to that pattern), there is always the possibility that a small number of hotspots formed directly because of HPRA adjustments might not have been adequately characterized in the training data. Therefore, to address these two cases, we retrained our ANN mechanism with buffer utilization metrics obtained after HPRA was applied in our simulation environment. This two-step approach (training without and with HPRA) ensures that the original training data that characterizes hotspots, is now fused with data that characterizes the events stemming from HPRA's inference on the interconnect. An alternative, in order to avoid hotspot formations directly resulted because of HPRA intervention (and not because they were initially going to be formed), is switching the HPRA algorithm off for some time. However, it must be emphasized that this will not solve the problem, as HPRA responds to information provided from the ANN; therefore, shutting down HPRA, even for small amounts of time, implies that the

TABLE 1
Prediction Accuracy Results for Synthetic Hotspot Traffic (See Section 5.2 for Model Details) and Transpose Traffic Patterns (See Section 5.1), Percentage of False Positive Hotspot Predictions, and Percentage of the Correctly Identified Hotspots with At Least 50 Cycles of Advance Prediction Using the Original Approach in [15] and the Optimized Approach Using Training Sets Fused with post-HPRA_b Data

	Training Without HPRA‡			Training Fused With Post-HPRA_b		
Normalized Throughput	Prediction Accuracy	False Positives	Hotspots Predicted 50 Cycles Ahead	Prediction Accuracy	False Positives	Hotspots Predicted 50 Cycles Ahead
	Hotspot Traffic					
0.46	95%	4.3%	91%	96%	2.4%	94%
0.64	95%	5.3%	91%	96%	2.7%	93%
0.82	91%	5.8%	88%	94%	3.0%	90%
0.98	89%	6.2%	84%	92%	3.2%	89%
	Transpose Traffic					
0.55	92%	5.2%	90%	95%	4.1%	93%
0.67	91%	7.0%	88%	94%	4.1%	91%
0.79	91%	6.8%	83%	92%	4.7%	91%
0.92	90%	5.9%	80%	90%	4.9%	91%

The ‡ symbol refers to the original training carried out in [15].

TABLE 2
Prediction Accuracy Results for the TRIPS SPEC CPU2000 Benchmarks [25], Percentage of False Positive Hotspot Predictions, and Percentage of the Correctly Identified Hotspots with At Least 50 Cycles of Advance Prediction Using the Original Approach in [15] and the Optimized Approach Using Training Sets Fused with post-HPRA_b Data

	Training Without HPRA‡			Training Fused With Post-HPRA_b		
TRIPS Benchmark	Prediction Accuracy	False Positives	Hotspots Predicted 50 Cycles Ahead	Prediction Accuracy	False Positives	Hotspots Predicted 50 Cycles Ahead
ammp	90%	8%	82%	91%	5%	85%
applu	84%	5%	80%	88%	3%	81%
apsi	71%	7%	65%	71%	6%	66%
art	78%	5%	78%	80%	4%	79%
bzip2	83%	6%	80%	90%	5%	81%
crafty	85%	4%	81%	88%	4%	81%
equake	89%	7%	82%	90%	6%	82%
gap	78%	7%	69%	80%	5%	70%
mcf	88%	9%	84%	88%	6%	85%
mesa	75%	6%	70%	80%	4%	72%
parser	81%	8%	74%	83%	7%	75%
swim	83%	3%	75%	85%	3%	77%
twolf	82%	8%	80%	83%	6%	82%
vortex	75%	7%	68%	81%	6%	75%
vpr	92%	9%	88%	95%	5%	90%
wupwise	90%	8%	86%	91%	5%	88%

The ‡ symbol refers to the original training carried out in [15].

overall balancing process will be thrown off, as HPRA is designed to operate continuously.

Another advantage of this revised (in relationship to our previous work [15]) methodology, is the lessening of false-positive predictions, as shown in Tables 1 and 2. The amount of false-positives is reduced, which implies that the HPRA algorithm, normally invoked whenever the ANN makes a hotspot prediction, will be invoked fewer times (but without any impact onto the performance) as the ANN will be more reliable. The impact of a misprediction (i.e., a missed hotspot) on the performance, while important, is not as vital as a false-positive. After all, if a hotspot is formed without it being predicted (i.e., without taking proactive measures since HPRA would not be invoked), then the NoC would behave exactly as it would without having the congestion-aware mechanism in place. The algorithm itself, and the information regarding the utilization in each router, use dedicated wires that transmit utilization rates and HPRA control signals, and would not be affected by a missed hotspot; instead, as the utilization rates regarding the newly formed (and unpredicted) hotspot rise, this trend would indicate its presence eventually (albeit after the hotspot has been formed), and HPRA would start taking corrective measures. On the other hand, a false-positive might alter the behavior of the application traffic (taking corrective action when not necessary) and potentially result in hotspots for which the ANN was never trained for, thus concurrently reducing the overall prediction accuracy. Typically, classifier training in such cases is biased towards the positive forecast (higher accuracy) than the negative forecast (false-positives), as the opposite nullifies the classifier benefits; what is important though, is to keep the false-positive occurrences as few as possible. In our case, the amount of false positives was reduced significantly by retraining the ANN predictor with NoC traffic traces resulting from post-HPRA operation.

There is, however, a fine line in training the ANN with post HPRA traces, as this might lead to either too strict, or too loose characterizations of a hotspot. While HPRA is reacting to an ANN forecast related to a specific hotspot, and starts diverting traffic elsewhere, a new situation might arise that may, or may not essentially lead to a new hotspot(s). However, if we force the ANN to essentially recognize all situations stemming from HPRA, and their data patterns are seemingly appearing to characterize a new hotspot, as actual hotspot candidates, then obviously the holistic approach will become an endless cycle, where HPRA will react to divert traffic away from a hotspot, while the ANN will inform it that HPRA is about to create a new hotspot, and vice versa. On the other hand, if we relax the hotspot formation patterns too much, then the overall accuracy of hotspot prediction will be reduced, essentially degrading the beneficial impact of our approach. To address this, we fine-tuned the ANN training method by utilizing as training data not only synthetic traces from both transpose and hotspot traffic, both at various hotspot formation rates (at moderate, medium, medium-high and high packet injection rates, shown in normalized form with respect to the network's saturation throughput in the leftmost column of Table 1) [15], but also training data from traces of the TRIPS benchmarks [25] that we used to evaluate HPRA (specifically, the `vpr` and `ammp` applications). We focused specifically on patterns that were observed right after the HPRA process was invoked, in an effort to steer the ANN to recognize the specific changes in behavior from the original training data.

3.3 Hotspot Modeling and Classification

We generate and subsequently utilize synthetic hotspot traffic, described in detail next, containing numerous (but of finite number) hotspots precisely pre-set in both their time and network position of occurrence (i.e., router node Cartesian coordinates) to evaluate experimentally the effectiveness of our ANN-based predictor. Successful hotspot

predictions are considered the ones actually predicted at least 50 cycles prior to their occurrence [15]. Hotspot classification (and identification) using our synthetic hotspot model is obviously straightforward, as the hotspots were intentionally set to occur in both time and space. However, this is not quite apparent when considering any alternative math model-based synthetically generated traffic that is not configured to contain hotspots explicitly, or real application traffic traces gathered from parallelized workload executions onto CMP architectures, such as synthetic transpose traces (see Section 5.1 for a detailed model description) and the TRIPS CMP benchmarks (see Section 5.3 for architectural details), utilized in our experimental evaluation of Section 5; here, hotspots arise mostly as by-products, i.e., as a sole consequence, of the behavioral patterns contained in said traffic traces or CMP workloads. As such, a hotspot classification model is in need. In particular, a rigorous mathematical model to identify the presence of NoC hotspot formations precisely, in any traffic pattern, and hence quantify them in space-time, is required to be developed. Such a hotspot classification model is beyond the scope of this paper and is left as future work. Hence, to create a heuristic-based hotspot classification model, following we establish, in detail, a traffic hotspot quantification model based upon our proposed hotspot-generating model.

Under our hotspot-generating traffic model two hotspots of 800-clock cycle duration randomly occur at two distinct routers in a 64-node 8×8 mesh-connected 2D NoC topology, within every window of 3,000 clock cycles. Hence 3.125 percent of the routers act as hotspots during 26.67 percent of the time. Multiplying these two values one can find a *rough average* of network space-time hotspot occurrences, resulting to a value of 0.83 percent under our synthetic model. We used this traffic profiling scheme for the transpose and TRIPS traces in identifying their contained hotspots. Using our simulation infrastructure we measured the input buffer utilization, a direct indication of traffic congestion levels, at every input port of each router in the NoC individually over a moving window (MW) of 300 cycles for the entirety of each transpose trace and TRIPS benchmark, and collected all data in a database. This MW cycle duration, empirically set with an aim to provide the best possible hotspot prediction accuracy, is set just long enough to capture traffic "spikes" (i.e., short-term intense hotspots) avoiding the effects of "smoothing" over longer sampling periods of statistics collection, while short enough to test the effectiveness of our predictor in filtering-out and recognizing short-term hotspots from the traffic "jitter," hence minimizing the number of possibly unpredicted hotspots. As the MW length may affect the process of statistical sampling, formally deriving its optimal length according to the specific spatio-temporal behavior of traffic under consideration, is beyond the scope of this work, and has ramifications for future studies. The router's (x,y) coordinates and the starting cycle of measurement for each corresponding buffer utilization data point were simultaneously recorded along. We then ranked the buffer utilization data of each such trace or benchmark in decreasing order, and considered the top 0.83 percent of highest measurements to be hotspot occurrence indicators due to their high-ranking network resource (buffer) demands. We note that a single hotspot does not necessarily have to span exactly 300 cycles; it can rather span several consecutive 300-cycle moving windows, or even exist during fewer than 300 cycles. For example, a hotspot of a 500-cycle duration spans 201 consecutive such 300-cycle buffer utilization moving windows. Thus, neither does the aforementioned figure of 0.83 percent highest buffer utilizations denote that exactly 0.83 percent of the application's space-runtime signifies NoC hotspot occurrences, as several hotspots can co-exist in time albeit at distinct router node planar coordinates, nor it is inferred that precisely 0.83 percent of the aggregate NoC traffic is hotspot-bound.

During our ANN-based predictor experiments, we compare the ANN-based predictor results, which likewise report the cycle of each hotspot occurrence (a 50-cycle advance prediction is considered to be successful [15]) and its router spatial location ((x, y) coordinates) in the NoC, against the original simulator-created sorted buffer utilization database so as to determine the accuracy of our ANN-based predictor for our synthetic transpose traces and TRIPS benchmark traces. Timely, missed, and falsely reported hotspot predictions are each accounted for. Tables 1 and 2 (they display the changes in accuracy) show the impact onto the ANN accuracy when comparing the initial training (with synthetic traces) and training data fused with post HPRA utilization rates. The accuracy changes are evaluated both over synthetic hotspot and transpose traces at various normalized (with respect to the saturation point) injection rates (Table 1), and 16 TRIPS benchmark traffic traces (Table 2); Tables 1 and 2 show the overall prediction accuracy, the percentage of falsely predicted hotspots, and the percentage of hotspots predicted at least 50 cycles ahead of their occurrence (giving enough time to HPRA to actually prevent their formation). For the synthetic hotspot traces the holistic approach, over the original approach, improves the prediction accuracy, the percentage of false predictions, and the 50-cycle ahead prediction accuracy by an average of 2.1, 47.4 and 3.2 percent respectively. The corresponding average numbers for the transpose traffic are 1.8, 28.5 and -1.3 percent respectively; a few prediction points actually show slight degradation. Finally, the corresponding average numbers for the TRIPS benchmark traces are 2.2, 25.3 and 2.3 percent. Overall, the ANN prediction accuracy is improved, albeit, on average, by only a few percentage points. The holistic approach also exhibits improved prediction rates for the hotspots predicted at least 50 cycles ahead, and reduces the number of falsely predicted hotspots as well.

4 Hotspot-Preventive Routing Algorithm: Microarchitecture and Variants

Conventional reactive-adaptive routing algorithms [6] (see Section 2) account for network load state to guide them online in dispersing the spatio-temporally unevenly distributed traffic of workloads in a best-effort load-balancing mode across the NoC topology to achieve higher uniformity in network resource usage in an attempt to sustain or improve the network's effective throughput [10], [17], [18]. However this pursuit is merely *reactive* to such traffic imbalances as conventional adaptive routing schemes use *current* or *historical* network statistical metrics, such as link usage

rates and buffer utilization (i.e., normalized router input queue occupancy), that have been gathered over short windows of elapsed time, to direct routing, instead of *preventing* congestion *a priory*; they are hence solely load imbalance *counteractive*. HPRA, our proposed scalable, progressive, and adaptable congestion-preventive routing function, overcomes these "near-sighted" limitations in alleviating network congestion by instead responding *pro-actively* to the near-future presence of highly adversarial traffic, preventing the unanticipated formation of NoC hotspots that may occur dynamically in the near future across the network topology. This pro-active hotspot prevention mode is achieved with the use of advance information sourced with the use of AI principles that are utilized during network operation that are applied in the form of NoC-embedded hardware ANNs to continuously predict possible future hotspot formations with relatively good accuracy (see Section 3; also refer to our previous work [15]). This advance hotspot ANN-sourced knowledge is used to guide packet routing across the NoC topology in an effort to mitigate or prevent any unforeseen near-future occurrences of hotspots. We note that here we assume that hotspots occur at destination nodes, when several routers steer traffic towards a subset of remaining routers.

HPRA works on two main axes: (1) it continuously balances-out traffic that is not destined towards routers that are predicted to become hotspots, or currently act as hotspots, and (2) as stated earlier, it utilizes advance information from NoC-embedded ANNs that report a priory near-future possible hotspot formations in an effort to mitigate them. To achieve these two modes it differentiates traffic into two categories: (1) traffic that is Non-Hotspot-router-Destined (NonHSD), and (2) traffic that is Hotspot-router-Destined (HSD), as determined by the independent ANN hardware processing engines embedded into the NoC (see Section 3 for details).

HPRA takes advantage of network path diversity, an attribute integral to 2D mesh-based NoCs, to load-balance non-adversarial traffic (i.e., NonHSD) across the NoC topology. Specifically, NonHSD traffic is routed through and is continuously balanced spatially across the topology based on real-time utilization statistics that are correlated well with downstream congestion and that are inexpensive to compute. In particular, the aggregate count of free virtual channels (VC_free), an indicator of congestion levels (i.e., the greater the aggregate count of free VCs spanning a network path suggests a lower level of network congestion), along the perimeter of the virtual minimum rectangle, outlined by the source-NonHSD destination node pair, is used to choose the least congested progressive routing path, towards the same NonHSD node.

There are two such sub-modes of operation, dubbed as HPRA_a and HPRA_b, as Figs. 4b and 4c demonstrates. Under HPRA_a, the mechanism at each router aggregates and propagates the VC_free count of downstream routers to adjacent (neighboring) upstream routers, measured independently along both the X or Y dimensions tangential to the source router. Here, downstream VC_free counts flow backwards (upstream) to the source router, using a low-bandwidth overlay (superimposed) network to propagate VC_free information, avoiding in-band signaling or all-to-all communication that can contribute to congestion, as Fig. 4b shows [10]. Each packet is *progressively* directed towards one of the two candidate routes, *once* at the source, onto which the VC_free count is the greatest, an indication of smaller congestion levels, similar to the operation of the O1TURN routing function [27] (O1TURN, however, is merely oblivious to traffic levels, where either dimension is randomly chosen once at the source). Like O1TURN, HPRA does not require lookup tables; the decision among the two possible XY or YX paths is purely algorithmic, carried out at the injecting router, where a packet routes along its first dimension first (either horizontally along X or vertically along Y) until it is exhausted, before switching to the next dimension until ejection; this constrains hardware overheads. When the same packet reaches the opposite corner router in a progressive manner, it starts to route along the remaining dimension. If for example HPRA_a first selects to route along the X dimension, once this path is exhausted, it then progressively traverses the Y dimension. As this scheme is carried out on a packet-by-packet basis, packets belonging to the same scheme may arrive out-of-order, hence a re-order buffer (ROB) at the ejection port reorders the arrival of packets, maintaining dependencies and synchronization (see Fig. 4a). When the X or Y offsets are zero,

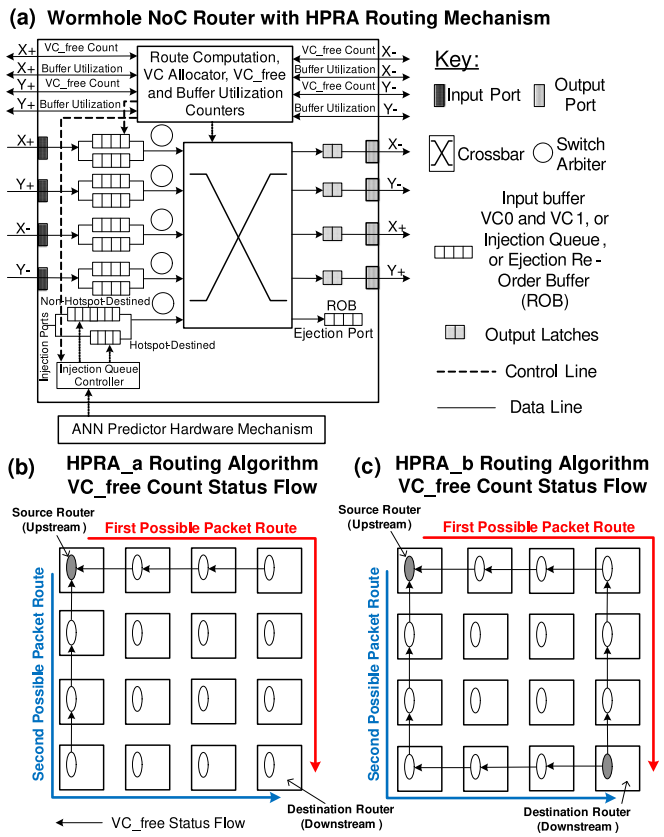


Fig. 4. HPRA mechanism added to a base wormhole flow-control NoC router microarchitecture: (a) base NoC router with two VCs (more VCs per port can be added accordingly) at each input port with HPRA augmentations and ANN processing element predictor, (b) HPRA_a routing algorithm with the two possible progressive packet routes and free VC (VC_free) status flow (only in the Y or X dimensions towards the source router) in the packet flow's reverse path direction, and (c) HPRA_b routing algorithm with the two possible progressive packet routes and free VC status flow (XY or YX towards the source router) in the packet flow's reverse path direction.

the progressive nature of HPRA dictates a straight-line route traversal, accordingly.

HPRA_b also uses the same overlay network to propagate VC_free information among routers. In HPRA_b, as opposed to HPRA_a, the two entire progressive paths along the perimeter of the minimum rectangle are each accounted separately for their VC_free counts so as to capture a larger region of congestion information, as Fig. 4c shows; these two paths are XY and YX, i.e., first traversing the X dimension and then the Y dimension, or the reverse, respectively. At each router the propagated VC_free values along the current progressive dimension are aggregated with those of the orthogonal dimension, similar to the "Fanin" RCA variant [10]. The overlay network assumes links of 5-bit width to encode and propagate VC_free information in an 8×8 mesh network, where the product of it's 14-hop longest path and an assumed two VCs per input router port dictates a maximum of 28 free VCs. For routers with larger VC counts per port, or for larger topologies, during each step of free_VC aggregation at each intermediate router, a shifting to the left technique could be used before adding the previous values at each router, hence reducing both the number of bits needed, and also weighing down the far-away collected statistics which are less significant and may contain noise [10]. The overhead of the overlay network was taken into account in computing HPRA's CMOS area and power overheads in Section 6. HPRA_b offers a slightly better performance advantage versus HPRA_a as the results of Section 5.1 show, due to the extended spatial congestion measurement gathering that is fed back to the source routers, albeit at a slightly higher cost as compared to HPRA_a.

In categorizing traffic as HSD and NonHSD, HPRA offloads hotspot router receivers and their surrounding network paths, while allowing non-hotspot-causing traffic (i.e., NonHSD) to ceaselessly reach their destination(s) in a load-balanced mode. HSD and NonHSD traffic are each housed into a distinct injection queue at each router (two injection queues per router), so that each such traffic flow can be separately monitored and controlled. Fig. 4a shows these two injection queues that are managed by a controller, i.e., the "injection queue controller." The HSD traffic injection queue is set at half the size of the NonHSD traffic injection queue (see Section 6), as experimental measurements showed that even under the worst-case NoC operating conditions, i.e., just below saturation point, the NonHSD traffic is at least twice as much as the HSD traffic. These buffer capacities ensure that no further delays are incurred between any processing element and its associated network interface queue, which, in case it fills up, can cause propagated backpressure to stall the pipeline of the PE and hence reduce the system's overall performance. The traffic categorization mechanism works as follows: when a PE residing in a CMP tile is about to inject a packet into the network, the packet's destination coordinates are checked. If they match the coordinates of any hotspot router, or of an-about-to-become-a-hotspot router (as determined by the ANN predictors, 50 cycles in advance, as computed in Section 3 and in our previous work [15]), it is housed into the HSD first-come first-served (FCFS) injection queue; otherwise, the packet is categorized as NonHSD and placed into the non-hotspot-router-destined FCFS queue, as would normally happen in a base router architecture. In the latter case the NonHSD routing load-balancing rules outlined earlier are carried out, and the NonHSD traffic flow is governed by regular wormhole flow-control [6]. To avoid deadlocks, VCs at inter-router ports are atomic and they cannot each be shared among HSD and NonHSD traffic. Finally, once the NonHSD queue is granted permission to inject a packet(s), the packet (s) is allowed to be network-injected in its entirety avoiding possible segmentation in case the NonHSD queue is immediately re-throttled; this scheme eliminates deadlocks.

This traffic differentiation essentially monitors the amount of HSD traffic that is destined to flow to router hotspot receivers, so that no further congestion is caused. With this scheme in place, the risk of further severely congesting and even stalling indefinitely the entire network as a hotspot(s) spatially spreads across the NoC topology, is moderated. The advance prediction of possible hotspot formation (s) gained from the ANN prediction engine(s) is utilized to either *partially* or *completely* throttle HSD traffic, gradually allowing some or all packets of such HSD traffic to reach their destinations. To determine whether HSD traffic should be throttled altogether, or allowed to be partially or completely injected into the network, a second statistical indicator is utilized, that of the average buffer utilization at the destination hotspot router, a calculated average of the aggregation of all its flit-occupied VC buffers at all its input ports. This information is fed back to the source router (refer to the "Buffer Utilization" signals in Fig. 4a; only one overlay link is used, which can be set to a logic value of "0" when the occupancy is below a threshold (read next), or set to a logic value of "1" when the average is equal/above this threshold). Only the source router is allowed to throttle HSD traffic, with throttling applied at the source router's HSD injection queue. The ABU metric is utilized at the source router to determine whether it should send HSD flits to the hotspot destination router; when the current ABU value at this hotspot destination router exceeds a threshold set at 50 percent (this value was determined via extensive empirical evaluation to be the most optimal; other values can be set accordingly), no HSD traffic from any source router is allowed to be injected into the network; otherwise HSD traffic is injected into the network, until this threshold limit is again exceeded. ABU is monitored on a cycle-by-cycle basis, enabling fine-grained HSD injection or throttling decisions.

Last, in case HSD traffic is de-throttled and concurrently NonHSD traffic exists in its own injection queue, both at the same source router, while a hotspot still exists or is predicted to persist, the Injection Queue Controller (see Fig. 4a) prioritizes the injection of HSD traffic over NonHSD traffic, as the injection of the former category has already suffered a delay at injection (during throttling). If the same hotspot is predicted as eradicated then the injection of either NonHSD or HSD traffic is randomly carried out on a per-flit basis. The entire above process is repeated until the same hotspot again starts to build-up or is predicted to occur in the near future.

5 Experimental Setup and Results

To evaluate HPRA's performance we implemented a detailed cycle-accurate simulator that supports k-ary 2D

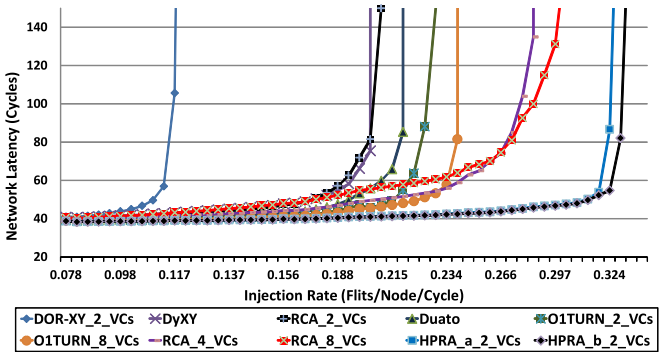


Fig. 5. Latency-throughput curves for various routing algorithms under synthetic transpose traffic applied to an 8×8 2D mesh-connected NoC.

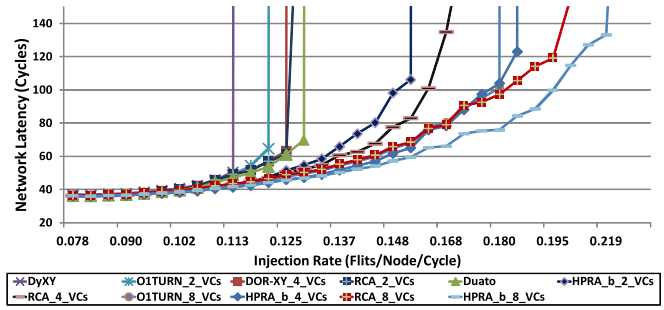


Fig. 6. Latency-throughput curves for various routing algorithms under synthetic hotspot traffic applied to an 8×8 2D mesh-connected NoC.

mesh topologies with four-stage pipelined routers, each with multiple 5-flit buffered VCs. The pipeline stages comprise: (1) routing computation to compute the next-hop route of in-transit packets, (2) VC arbitration to allocate VCs at the downstream router, (3) switch arbitration to allocate per-flit switch bandwidth at the crossbar to traversing flits, (4) crossbar (switch) traversal, and lastly a one-cycle physical link traversal consumed by each flit. In the transpose and hotspot synthetic traffic patterns utilized (see Sections 5.1 and 5.2, respectively), packets are composed of five 32-bit flits, while the TRIPS CMP applications [25] comprise two packet types of relevant sizes-Section 5.3 outlines their particularities. Also, each router is assumed to be capable of receiving data out-of-order, with a re-order buffer incorporated at the router's ejection port re-ordering received flits (see Fig. 4a). Simulation duration was taken to be half a million cycles for synthetic traffic patterns, and for the entire length of each TRIPS benchmark (Section 5.3). In-network transit delay is taken to span the elapsed time between a head flit's network ingress at either injection buffer, HSD or NonHSD, at its source router node, and network egress of the tail flit that belongs to the same packet at its destination node; this includes the queuing delay, the router pipeline and link delays, and any incurred contention-related delays. The network latency metric is calculated by aggregating over all such in-network transit delays of all packets and then averaging over the total number of transported packets during a simulation run.

5.1 Results with Synthetic Transpose Traffic

We first evaluate the effectiveness of HPRA using synthetic transpose traffic, an adversarial form of traffic as it overloads the two orthogonal virtual diagonals of the network topology. We utilize a uniform random injection process, with each router having the same probability of injecting, as well as ejecting a packet, into/from an 8×8 mesh topology. We note that the ANN-based predictions used to identify the location and time of occurrence of hotspots in the 8×8 mesh NoC are the original single-run trained pre-HPRA predictions which do not include the fused post-HPRA results (see Section 3.2).

Fig. 5 compares the sustainable throughput performances of both HPRA_a and HPRA_b (see Section 4 for details) against dimension-order routing (DOR-XY), Duato's fully adaptive Protocol [8], the near-optimal O1TURN adaptive routing algorithm [27], the DyXY congestion-oriented adaptive routing algorithm [18], and the regional congestion awareness (RCA) technique [10] which aims in enhancing global load-balancing in adaptive routing; we compare against the *1D* version of RCA, a locally adaptive routing scheme that uses the *VC congestion* metric, independently aggregated and propagated along each network dimension (row and column) that bound a packet's currently routing quadrant in a 2D mesh-connected NoC. Note that for readability, the left to right, then down, flow of the graph markers follow the left to right flow of the curves; the same format applies to Fig. 6. We note that RCA [10] and O1TURN [27] were originally designed to work optimally with a whopping eight VCs per physical input router port, as compared to our lightweight HPRA approach which uses just two VCs per port (at minimum), hence requiring significantly fewer overheads both in router buffering and in the arbitration mechanism's design complexity. Nevertheless, we also run additional versions of O1TURN and RCA, with both two and four VCs per port as well, for comparison purposes.

The performances of the two HPRA schemes, as depicted in Fig. 5, show improvement in attainable throughput, in some cases dramatic, versus current state-of-the-art adaptive routing schemes, with no sacrifice in latency; HPRA outperforms DOR-XY, the DyXY congestion-adaptive routing [18], and Duato's routing Protocol [8] (all with 2 VCs), significantly. HPRA_a and HPRA_b (each with two VCs) also outperform, again in terms of the sustainable throughput, the O1TURN algorithm [27], which uses eight VCs, by 35.5 and 37.2 percent, respectively. Lastly, HPRA_a and HPRA_b outperform the RCA congestion-aware algorithm with two VCs by 63.5 and 65 percent, respectively, and RCA with eight VCs by about 13 and 14 percent, respectively. HPRA_b is marginally better than HPRA_a, as it considers the aggregated congestion levels along the entire span of the two alternative routing paths (see Section 4).

5.2 Results with Synthetic Hotspot Traffic

To determine the effectiveness of HPRA in eradicating hotspots we utilize traffic generated with our hotspot-generating model (see Section 3.3) in which well-defined spatially-located hotspot formations of relatively short duration, centered at two distinct randomly chosen destination routers, are directed to receive an abnormal amount of traffic (concurrent hotspot occurrences). Since the selection of hotspot placements is random, two closely located

hotspots may give rise to an aggregated hotspot, or "super hotspot," with HPRA called to mitigate them. Note that the ANN-based HPRA scheme can handle any number of hotspots; the rationale of using just two hotspots in our 64-node NoC (8×8 mesh network) is to allow few intensified hotspot formations in order to stress-test HPRA, where using relatively many hotspots would debilitate their individual intensities.

Our hotspot model uses the uniform random traffic (URT) model as a base, except when for short pre-specified and periodically occurring time intervals just two arbitrarily selected nodes each receive with equal probability 10 percent (20 percent of total traffic, combined) of the aggregate network traffic from any remaining (non-hotspot) sender nodes; the remaining 80 percent of aggregate traffic remains as URT traffic. In particular, we define time frame windows (TFW) of 3,000 cycles, during any time which the two hotspots can occur simultaneously for a duration of 800 cycles. These values were set empirically. No other hotspots can occur within a TFW, and during the rest of the duration of the TFW the traffic behaves purely as URT. Under URT, all router nodes in a NoC have an equal probability of sending and receiving a packet to/from another node per unit time; thus the traffic is evened-out across the NoC topology, and the attainable network throughput, during this time with hotspots being absent, is the maximum that can be achieved both practically and theoretically [6]. The injection rate during the URT phase, and during the hotspot phase, is of constant periodicity, i.e., the injection rate is steady and is set at a pre-defined value. This model stress-tests the ability of the ANN-based predictor in making correct hotspot occurrence predictions in time-space, as hotspots occur frequently and unexpectedly in both time and space. Also, since the hotspots are of relatively long duration (800 cycles out of every 3,000 cycles), our hotspot model also tests the ability of HPRA to handle hotspot traffic while sustaining high network throughput under such adversarial conditions.

Fig. 6 compares the sustainable throughput performance of HPRA_b (see Section 4 for details) against deterministic DOR-XY routing with four VCs per input port, Duato's fully adaptive Protocol [8], the O1TURN [27] (with two or eight VCs per input port) and DyXY [18] adaptive routing algorithms, and the RCA algorithm [10] using its *1D* version, as used previously in our transpose traffic experiments of Section 5.1. The results of HPRA_a closely track (albeit being slightly less-performing) those of HPRA_b and are not shown for readability purposes. We use three versions of RCA and HPRA_b, both with two, four, or eight VCs. The ANN-based predictions used to identify the location and time of occurrence of hotspots in the 8×8 mesh NoC are the original single-run trained pre-HPRA predictions which do not include the fused post-HPRA results (see Section 3.2).

The performance of HPRA_b, as depicted in Fig. 6, shows improvement in attainable throughput, in some cases dramatic, versus current state-of-the-art adaptive routing schemes, with no sacrifice in latency; HPRA_b outperforms DOR-XY with four VCs per input port, DyXY routing [18], and Duato's Protocol [8], substantially. HPRA_b outperforms, also in terms of the sustainable throughput, the O1TURN algorithm [27] with two VCs by approximately 9.6, 40, and 81 percent when HPRA_b utilizes two, four and eight VCs per input port, respectively; HPRA_b also outperforms O1TURN by 18 percent when both algorithms use eight VCs per input port. Last, HPRA_b with two VCs outperforms the RCA congestion-aware routing algorithm with the same VC count per input port by approximately 16.1 percent, by 11.8 percent when both algorithms utilize four VCs at each of their input port, and by 8.8 percent when both algorithms utilize eight VCs at each of their input port. The conclusion, here, is that HPRA_b is relatively better performing as opposed to RCA, when fewer (and more reasonable) numbers of VCs are in use. Given the resource-constrained environment of NoCs, i.e., it is more ideal to use fewer VCs for buffering, HPRA_b is the preferred choice vs. O1TURN or RCA in handling hotspot traffic.

5.3 Results with TRIPS CMP Applications

The TRIPS [25] prototype CMP consists of two large, tiled, distributed processing cores, with each core containing an execution array of ALU tiles, distributed register file tiles and partitioned local L1 cache tiles interconnected via a 5×5 mesh network. The two cores are interconnected by a second, 4×10 2D mesh network to a shared, distributed static-NUCA L2 cache. The main, processor-core, 5×5 wormhole-routed NoC is the Operand mesh-connected Network, abbreviated as OPN. The two multi-tile processor cores communicate through the on-chip secondary cache system using an embedded 4×10 wormhole-routed mesh network with four virtual channels per port and DOR-YX routing, abbreviated as on-chip network [9]. The OCN is optimized for cache line-sized transfers with support for other memory-related operations, acting as an inter-core fabric for the two processors, the L2 cache, DRAM, I/O and DMA traffic [25].

We used a set of 16 benchmarks from the SPEC CPU2000 Suite [30] gathered from the cycle-accurate TRIPS processor simulator to generate OCN requests, compiled using the Scale compiler [29] (appropriately modified for TRIPS's use). There are two memory system transaction types, writes and reads, consisting of request and reply packets. Write transactions from the processor to the L2 bank consist of a five-flit request packet containing the evicted L1 dirty cache line, answered with a one-flit packet acknowledgment to the processor; while read transactions consist of a single flit read request packet from the processor, replied by the L2 bank with a five-flit packet containing the requested cache line.

The TRIPS 4×10 mesh OCN was blanketed using three base 4×4 mesh-covering ANNs as in our previous work [15]; an OR-based voting engine was used for the results from each ANN that affect the overlapping routers as Fig. 7 shows. The ANN-based predictions used to identify the location and time of occurrence of hotspots in the 4×10 mesh OCN are the original single-run trained pre-HPRA predictions which do not include the fused post-HPRA results (see Section 3.2). Fig. 8 compares latency results for various TRIPS OCN [25] benchmarks using HPRA_b with two VCs and various other routing algorithms: deterministic DOR-XY (note that the benchmark latencies of the `applu`, `apsi`, `art`, `mgrid`, `twolf`, and `vpr`, all running using DOR-XY routing, are over 44 cycles and are clipped

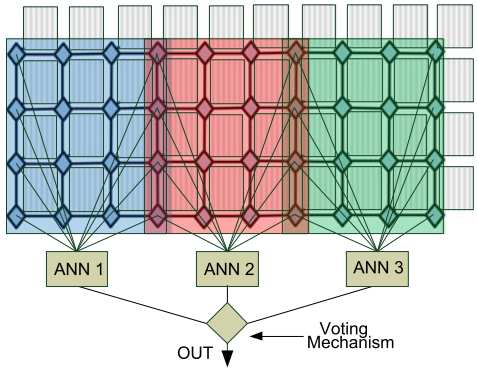


Fig. 7. Scaling the base 4×4-NoC area-covering ANN to map the TRIPS 4×10 mesh OCN NoC [25]. Three base ANNs were used (ANN1-ANN3), where an OR-based voting engine was used for calculating the results from each ANN that affect the overlapping routers (two darker red regions).

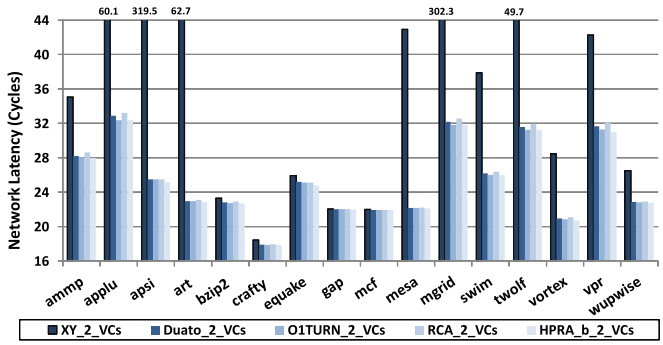


Fig. 8. Latency results for sixteen TRIPS OCN [25] benchmarks using DOR-XY routing and the following adaptive and/or load-balancing routing algorithms: Duato's Protocol [8], O1TURN [27], RCA [10], and HPRA_b.

in Fig. 8), Duato's Protocol [8], the near-optimal O1TURN adaptive routing algorithm [27], and the RCA algorithm [10] using its *1D* version (as used in Section 5.1 and Section 5.2). Despite the fact that under all TRIPS benchmarks HPRA_b with 2 VCs shows better NoC sustainable performance, this improvement is marginal; HPRA_b is on average 0.93, 0.37, and 1.57 percent better-performing than Duato's Protocol (two VCs), O1TURN (two VCs) and RCA (two VCs), respectively. HPRA_b also outperforms DOR-XY by a whopping 177.5 percent, mainly due to the `applu`, `apsi`, `art`, `mgrid`, `twolf`, and `vpr` performing relatively poorly under DOR-XY routing. The best performance gains are with `vpr` at 2.20 percent when compared to Duato's Protocol, `equake` and `apsi` both at 1.38 percent when compared to O1TURN adaptive routing, and `vpr` at 3.52 percent when compared to RCA congestion-aware routing. The reason for these small improvements, except in the case of DOR-XY deterministic routing, is that the TRIPS applications contain very light hotspots, and the 4×10 mesh OCN, due to its small size in the horizontal dimension (spanning just three hops) offers little opportunity in exploiting path diversity; hence, HPRA cannot offer substantial performance advantage as compared to when using the reported routing functions.

5.4 Discussion: HPRA versus Existing State-of-the-Art, and the Role of a Prediction Mechanism

While any adaptive routing algorithm may utilize the prediction results produced by the ANN-based predictor, such an ARA will need to support appropriate maneuvering in proactively steering traffic away from hotspots that may lie ahead. If the above flexibility is lacking, then the ANN-based hotspot predictor(s) may not be fully utilized. For instance, Duato's ARA [8] (an instrumental ARA, on which various other adaptive high-performance ARAs are based on, and utilized in our experimental evaluation of Section 5) concentrates mostly on providing a deadlock-free routing environment, where a contributory "escape virtual channel" acts towards establishing such deadlock-avoidance. Though HPRA is also deadlock-free, Duato's ARA is oblivious to any statistical gathering that HPRA performs in steering traffic around hotspots, hence Duato's ARA will have to be appropriately modified.

Another state-of-the-art ARA, the O1TURN [27] semi-ARA (also utilized in our experimental evaluation of Section 5), is a theoretically near-optimal worst-case throughput ARA for mesh networks, where its oblivious-to-congestion-levels load-balancing nature helps attain high network throughput. It is, however, a source-based minimal-hop routing function with no regard to network-level statistical gathering to direct its routing decisions, which dictates that it has to be modified to both accept online traffic-level statistics and also consider a prediction algorithm to match the behavior of HPRA.

Lastly, DyXY [18] and RCA [10] are adaptive minimal-hop and progressive routing algorithms which use online statistics to guide their routing decisions in alleviating network congestion, where DyXY uses localized utilization statistics as an indication of network congestion, while RCA utilizes "regional" congestion statistics considered at far-ahead routers-hence RCA's performance is often greater than that of DyXY as RCA makes improved routing decisions based on greater spatial coverage. However, they are both merely *reactive* to congestion levels, where already-occurring congestion, albeit possibly light, is corrected, unlike HPRA, which is *proactive*, as it avoids having congestion instances altogether by reacting in advance to hotspot prediction directives provided by the network-embedded ANN engines.

HPRA therefore, combines all the good attributes from various ARAs, in addition to traffic throttling and consideration of ANN-based hotspot predictions: (1) it is deadlock-free as Duato's algorithm, (2) it is minimal as DyXY, O1TURN and regional congestion awareness routing, as it only allows either XY or YX routing, and (3) it gathers statistical data as in DyXY and RCA. Given that HPRA performs better, under both the synthetic traffic patterns that were tested, and the TRIPS [25] real workload traces, as compared to the four aforementioned routing algorithms (Duato, DyXY, O1TURN and RCA), it may be unfair to these algorithms (and any other ARAs) which have not been modified to be used with a hotspot predictor to directly compare them with HPRA when it comes to them getting a mechanism such as our prediction engine. However, one of the possible future work opportunities is how existing and well-performing NoC ARAs can benefit from an intelligent forecasting mechanism.

6 Hardware Synthesis Results

The overall interconnection system, including the optimized PWL approximation ANN-based prediction engines (Section 3.1) and the HPRA hardware (Section 4), was implemented in Verilog and synthesized using synopsys design vision, using a commercial 65 nm CMOS technology library at 1 V power supply voltage, targeting 500 MHz as operating frequency. For the purposes of comparison, we also synthesized a three-stage pipelined router with two VCs per input port comprising (1) routing computation, (2) parallelized VC arbitration and speculative switch allocation, and (3) switch (crossbar) traversal, where each such stage is executed within one clock cycle. The speculative switch allocation prioritizes non-speculative requests over speculative ones, while prioritized matrix arbitration, look-ahead routing and credit-based wormhole flow-control [21] are utilized. We also synthesized hardware deployed in each processing node, to receive data packets out-of-order (i.e., an ejection port re-order buffer of 8-flit capacity, and control logic), and included those as overheads in our computations as well. A 100-flit FCFS NonHSD traffic injection queue was constructed. The Hotpot-router-Destined (HSD) traffic injection queue was set at a 50-flit depth, half the size of the NonHSD traffic injection queue (see Section 4), as experimental measurements showed that even under the worst-case NoC operating conditions, i.e., just below the saturation point, no more than 45-flit buffering to support HSD traffic throttling was required. These queue capacities ensure that no further delays are incurred between any traffic-sourcing PE and its associated network packet-injecting queue, which, in case it fills up, can cause propagated backpressure that can stall the pipeline of the PE and hence reduce the system's overall performance. The ANN hardware overheads and scalability analyses, and their related impact upon the ANN prediction accuracy and performance were investigated in detail in our previous work [15]; here, we comprehensively provide the total hardware overheads for both the targeted NoC and its associated ANN and HPRA hardware.

For a targeted 8×8 mesh NoC, synthesis results indicate a CMOS hardware overhead of 3.1 percent for the ANN prediction system (all hardware related to the ANN, including the new PWL approximation units (see Section 3.1)), and an overhead of 7.5 percent for the HPRA hardware, the majority of which lies in the extra buffer space required in the injection ports of each router. These hardware results also include the 6-bit dedicated links (comprising the overlay network, see Sections 3 and 4) between routers to carry buffer utilization data and ANN prediction results back to the routers needed/produced by the ANN engine (see Section 3), that are time-multiplexed to also carry VC_free count and average buffer utilization metrics (five for the former and one for the latter, see Section 4). The overall CMOS hardware overhead, thus, is 10.6 percent. We note that the reported hardware overheads are not directly comparable to existing hotspot-managing routing algorithms, however, as none of the related works perform such a comparison; further, the overheads for each algorithm depend heavily on the selected network sizes, topologies and the synthesis methodology chosen, hence complicating direct comparisons. Instead, the reported HPRA overheads (3.1 percent) are in comparison to a basic DOR-XY pipelined wormhole router and its associated network overheads.

Next, we used Synopsys' PrimePower to estimate power consumption. Assuming a 30 percent switching activity probability, the synthesized ANN engine described above (with the PWL unit instead of an SRAM-based look-up table), consumes an estimated 0.008 mW when computing one hotspot prediction, a negligible overhead. Another advantage of using PWL over SRAM is that the resulting unit replaces SRAM logic with gates instead, reducing the overall area, while it also lessens power consumption. The power consumption number reported does not include the power consumed in transmitting the utilization rates from each router, but it includes the computation of these values at each router. However, the extra packetized messages involved in the ANN computation are a very small fraction of the on-chip network power consumption, as each router transmits one packet every 50 cycles (as in our case for each 4×4 mesh sub-area, see Section 3; the number of cycles and the time frame that utilization rates are collected can be adjusted depending on each specific system as explained before). Therefore, in our experimental NoC, if we assume that each router will route at least one packet per cycle (i.e., fully utilized routers), the control packets will be $1/50$th of the total number of NoC packets. This however is an extreme scenario; obviously, system-level simulation which includes cycle-accurate NoC power modeling is required to compute the impact on the power consumption of the ANN-induced communication overheads. Overall, the estimated power requirements of the ANN predictor can therefore be considered negligible when compared with existing real architectures such as Intel's single-chip cloud computer [12].

An important contribution of this paper is the reduction in the overall power consumption of HPRA, stemming from the lessening of false positives (resulting from the new training fused with the HPRA operation data). As the HPRA algorithm is less frequently invoked, the overall power savings for the system, when comparing the holistic training of the ANN, versus the training of the ANN using only synthetic data, is 8.9 percent. While the accuracy benefits might seem initially insignificant, the power savings are quite important, as HPRA is a relatively frequently triggered online system optimization algorithm, and any wrongful invocation (directly resulted due to false positives) is extremely costly in terms of power.

The base router was estimated at 351 mW, whereas the modified router (with the injection buffers) consumes 383 mW, hence the power overheads of the overall hotspot prevention system (that includes the ANN and the dedicated hardware within each router) are estimated at approximately 9 percent. Both area and power results demonstrate that the ANN-based predictor together with the hotspot-preventive routing algorithm are a feasible hardware engine in a NoC-based multicore chip. However, as HPRA improves the overall system performance by reducing the average packet delay and extending the effective throughput of the on-chip network, it is anticipated that the overall energy savings will be more in the case of a hotspot-preventive mechanism present in a NoC-based multicore processor. This is left as future work.

7 Conclusions and Future Work

This paper presented the HPRA which uses advance congestion-level knowledge based on artificial intelligence principles embedded in the system hardware, in the form of ANNs, to *pro-actively* guide packet routing across the NoC in an effort to mitigate any unforeseen near-future occurrences of traffic hotspots. These ANNs were first trained offline and during multicore operation they gather online statistical data to predict about-to-be-formed hotspots, promptly informing HPRA to take appropriate action(s) in preventing hotspot formation(s). Next, in a holistic approach, further ANN training was performed with data acquired after HPRA interfered, so as to improve hotspot prediction accuracy; hence, the ANN mechanism did not only predict hotspots, but was also aware of changes that HPRA imposed upon the interconnect infrastructure. Evaluation results indicate that a post-HPRA data-trained ANN-based predictor can forecast hotspot formation(s) with an accuracy up to 94 percent. HPRA also extends network throughput by up to 81 percent when compared with several existing state-of-the-art congestion-aware routing functions, while requiring a moderate 10.3 percent hardware overhead. Future work involves the derivation of a formalized mathematical-based hotspot classification model that could prove handy in precisely quantifying the intensity of hotspots in space-time, to replace the heuristics-based hotspot classification model utilized here, so as to optimally guide the ANN-based hotspot preventive mechanism of HPRA.

Acknowledgments

This work falls under the Cyprus Research Promotion Foundation's Framework Programme for Research, Technological Development and Innovation 2009-10 (ΔΕΣΜΗ 2009-10), co-funded by the Republic of Cyprus and the European Regional Development Fund, and specifically under Grant ΠΕΝΕΚ/0311/06. The authors would like to thank Dr. Paul Gratz for helping with the TRIPS CMP OCN applications.

References

[1] E. Baydal, P. Lopez, and J. Duato, "A family of mechanisms for congestion control in wormhole networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 9, pp. 772–784, Sep. 2005.

[2] P. Bogdan and R. Marculescu, "Quantum-like effects in network-on-chip buffers behavior," in *Proc. ACM/EDAC/IEEE Design Automation Conf.*, Oct. 2007, pp. 266–267.

[3] S. H. Cameron, "Piece-wise linear approximations," IIT Res. Inst., U.S. Government Res. & Develop. Rep., Tech. Rep. CSTN-106, vol. 67, Feb. 1966.

[4] S. H. Cameron, *Piece-Wise Linear Approximations*, P.N. Publishers, ISBN: B00ACXWSBO, 1966.

[5] W. J. Dally and B. Towles, "Route packets not wires: On-chip interconnection networks," in *Proc. ACM/EDAC/IEEE Design Automation Conf.*, Jun. 2001, pp. 684–689.

[6] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Mateo, CA, USA: Morgan Kaufmann, 2004.

[7] M. Daneshtalab, A. Sobhani, A. Afzali-Kusha, O. Fatemi, and Z. Navabi, "NoC Hot spot minimization using antnet dynamic routing algorithm," in *Proc. IEEE Int. Conf. Appl.-Specific Syst., Archit. Process.*, Dec. 2006, pp. 33–38.

[8] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, no. 12, pp. 1320–1331, Dec. 1993.

[9] P. Gratz, C. Kim, R. McDonald, S.W. Keckler, and D. Burger, "Implementation and evaluation of on-chip network architectures," in *Proc. IEEE Int. Conf. Comput. Design*, Oct. 2006, pp. 477–484.

[10] P. Gratz, B. Grot, and S. W. Keckler, "Regional congestion awareness for load balance in networks-on-chip," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit.*, Feb. 2008, pp. 203–214.

[11] W. S. Ho and D. L. Eager, "A novel strategy for controlling hot-spot congestion," in *Proc. IEEE Int. Conf. Parallel Process.*, Aug. 1989, pp. 14–18.

[12] J. Howard, S. Dighe, S. R. 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. K. 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," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 173–183, Jan. 2011.

[13] J. Hu and R. Marculescu, "DyAD-Smart routing for networks-on-chip," in *Proc. ACM/EDAC/IEEE Design Automation Conf.*, Jun. 2004, pp. 260–263.

[14] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *IEEE Comput. Mag.*, vol. 29. no. 3, pp. 31–44, Mar. 1996.

[15] E. Kakoulli, V. Soteriou, T. Theocharides, "Intelligent hotspot prediction for network-on-chip-based multicore systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 3, pp. 418–431, Mar. 2012.

[16] E. Kakoulli, V. Soteriou, and T. Theocharides, "HPRA: A proactive hotspot-preventive high-performance routing algorithm for networks-on-chips," in *Proc. IEEE Int. Conf. Comput. Design*, Oct. 2012, pp. 249–255.

[17] P. Lotfi-Kamran, M. Daneshtalab, C. Lucas, and Z. Navabi, "BARP-A dynamic routing protocol for balanced distribution of traffic in NoCs," in *Proc. ACM/IEEE Design, Automation Test Eur. Conf. Exhib.*, Mar. 2008, pp. 1408–1413.

[18] M. Li, Q.-A. Zeng, and W.-B. Jone, "DyXY-A proximity congestion-aware deadlock-free dynamic routing method for network on chip," in *Proc. ACM/EDAC/IEEE Design Automation Conf.*, Jul. 2006, pp. 849–852.

[19] R. Marculescu, U. Y. Ogras, L.-S. Peh, N. E. Jerger, and Y. Hoskote, "Outstanding research problems in NoC design: System, microarchitecture, and circuit perspectives," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 1, pp. 3–21, Jan. 2009.

[20] E. Nilsson, M. Millberg, J. Oberg, and A. Jantsch, "Load distribution with the proximity congestion awareness in a network on chip," in *Proc. IEEE/ACM Design, Automation Test Eur. Conf. Exhib.*, Mar. 2003, pp. 11126–11127.

[21] L.-S. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit.*, Jan. 2001, pp. 255–266.

[22] Z. Qian, P. Bogdan, G. Wei, and C.-Y. Tsui, "A traffic-aware adaptive routing algorithm on a highly reconfigurable network-on-chip architecture," in *Proc. IEEE/ACM/IFIP Int. Conf. Hardware/Softw. Codesign Syst. Synthesis*, Oct. 2012, pp. 161–170.

[23] M. Ramakrishna, P. V. Gratz, and A. Sprintson, "GCA: Global congestion awareness for load balance in networks-on-chip," in *Proc. ACM/IEEE Int. Symp. Netw.-on-Chip*, Apr. 2013, pp. 1–8.

[24] R. S. Ramaujam and B. Lin, "Destination-based adaptive routing on 2D mesh networks," in *Proc. ACM/IEEE Symp. Archit. Netw. Commun. Syst.*, Oct. 2010, pp. 19–31.

[25] K. Sankaralingam, R. Nagarajan, R. McDonald, R. Desikan, S. Drolia, M. S. Govindan, P. Gratzf, D. Gulati, H. Hanson, C. Kim, H. Liu, N. Ranganathan, S. Sethumadhavan, S. Shariff, P. Shivakumar, S. W. Keckler, and D. Burger, "Distributed microarchitectural protocols in the TRIPS prototype processor," in *Proc. IEEE/ACM Int. Symp. Microarchit.*, Dec. 2006, pp. 480–491.

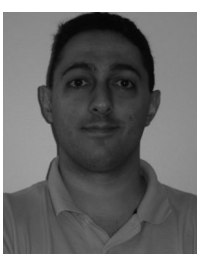
[26] H. Sarbazi-Azad, M. Ould-Khaoua, and L. M. Mackenzie, "An analytical model of fully-adaptive wormhole-routed k-Ary n-Cubes in the presence of hot spot traffic," *IEEE Trans. Comput.*, vol. 50, no. 7, pp. 623–634, Jul. 2001.

[27] D. Seo, A. Ali, W.-T. Lim, and N. Rafique, "Near-optimal worst-case throughput routing for two dimensional mesh networks," in *Proc. IEEE/ACM Int. Symp. Comput. Archit.*, Jun. 2005, pp. 432–443.

[28] M. Sheng, N. E. Jerger, and Z. Wang, "DBAR: An efficient routing algorithm to support multiple concurrent applications in networks-on-chip," in *Proc. IEEE/ACM Int. Symp. Comput. Archit.*, Jun. 2011, pp. 413–424.

[29] A. Smith, J. Burrill, J. Gibson, B. Maher, N. Nethercote, B. Yoder, D. Burger, and K. McKinley, "Compiling for EDGE architectures," in *Proc. IEEE/ACM Int. Symp. Code Generation Optim.*, Mar. 2006, pp. 185–195.
[30] Standard Performance Evaluation Corporation, SPEC CPU2000 Benchmark Suite. [Online]. Available: http://www.spec.org, 2000.
[31] L. P. Tedesco, T. Rosa, F. Clermidy, N. Calazans, and F. G. Moraes, "Implementation and evaluation of a congestion aware routing algorithm for networks-on-chip," in *Proc. IEEE Symp. Integr. Circuits Syst. Design*, Sep. 2010, pp. 91–96.
[32] T. Theocharides, G. Link, E. Swankoski, N. Vijaykrishnan, M. J. Irwin, and H. Schmit, "Evaluating alternative implementations for LDPC decoder check node function," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI Design*, Feb. 2004, pp. 77–82.
[33] M. Thottethodi, A. R. Lebeck, and S. S. Mukherjee, "Self-tuned congestion control for multiprocessor networks," in *Proc. Int. Symp. High-Perform. Comput. Archit.*, Jan. 2001, pp. 107–118.
[34] I. Walter, I. Cidon, R. Ginosar, and A. Kolodny, "Access regulation to hot-modules in wormhole NoCs," in *Proc. ACM/IEEE Annu. Symp. Netw.-on-Chip*, May 2007, pp. 137–148.

Vassos Soteriou (S'03-M'08) received the BS and PhD degrees in electrical engineering from Rice University, Houston, TX, in 2001, and Princeton University, Princeton, NJ, in 2006, respectively. He is currently an assistant professor at the Department of Electrical Engineering, Computer Engineering, and Informatics, Cyprus University of Technology. He received the Best Paper Award at the 2004 IEEE International Conference on Computer Design. His research interests lie in multicore computer architectures, and on-chip networks. He is a member of the IEEE.



Theocharis Theocharides (S'01-M'05-SM'11) is an assistant professor at the Department of Electrical and Computer Engineering, University of Cyprus. His research focuses on the broad area of intelligent embedded systems design, with emphasis on the design of reliable and low power embedded and application-specific processors, media processors and real-time digital artificial intelligence applications. He serves on the editorial boards of *IEEE Design & Test Magazine* and on the TPC of several IEEE-sponsored and co-sponsored conferences. He is a senior member of the IEEE.

Elena Kakoulli (S'10) received the master's degree in computer science from the University of Cyprus. She is currently working toward the PhD degree in computer engineering at the Department of Electrical Engineering, Computer Engineering and Informatics, Cyprus University of Technology. Her research interests lie in the fields of computer architecture and on-chip interconnection networks (OCINs), with emphasis on routing algorithms, performance enhancements, and hybrid photonic-electronic OCINs. She is a student member of the IEEE.