Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2012, Article ID 107821, 12 pages
doi:10.1155/2012/107821

Research Article

Intelligent On/Off Dynamic Link Management for On-Chip Networks

Andreas G. Savva,[1] Theocharis Theocharides,[1] and Vassos Soteriou[2]

[1] *Department of Electrical and Computer Engineering, University of Cyprus, 1678 Nicosia, Cyprus*
[2] *Department of Electrical Engineering and Information Technology, Cyprus University of Technology, 3036 Limassol, Cyprus*

Correspondence should be addressed to Andreas G. Savva, eep7sa4@ucy.ac.cy

Received 15 August 2011; Accepted 3 December 2011

Academic Editor: Sao-Jie Chen

Networks-on-chips (NoCs) provide scalable on-chip communication and are expected to be the dominant interconnection architectures in multicore and manycore systems. Power consumption, however, is a major limitation in NoCs today, and researchers have been constantly working on reducing both dynamic and static power. Among the NoC components, links that connect the NoC routers are the most power-hungry components. Several attempts have been made to reduce the link power consumption at both the circuit level and the system level. Most past research efforts have proposed selective on/off link state switching based on system-level information based on link utilization levels. Most of these proposed algorithms focus on a pessimistic and simple static threshold mechanism which determines whether or not a link should be turned on/off. This paper presents an intelligent dynamic power management policy for NoCs with improved predictive abilities based on supervised online learning of the system status (i.e., expected future utilization link levels), where links are turned off and on via the use of a small and scalable neural network. Simulation results with various synthetic traffic models over various network topologies show that the proposed work can reach up to 13% power savings when compared to a trivial threshold computation, at very low (<4%) hardware overheads.

1. Introduction

Power management is a crucial element in modern-day on-chip interconnects. Significant efforts have been made in order to address power consumption in networks-on-chips (NoCs) [1–6]. One of the most power-hungry NoC components are the links connecting the routers to each other and the processing elements (PEs) of the on-chip interconnection network (NoC). Recent data from Intel's Teraflop NoC prototype [7] suggests that link power consumption could be as high as 17% of the network power and could be even more given the types of links used as well as the size and pipelining involved in designing the link structure. These links, which can be designed with differential signals and low-voltage swing hardware using level converters as circuit-based optimizations for low power consumption, are almost active all the time, even when not transmitting useful data thus spending energy when no inter-router communication exits. While such traditional hardware design techniques have contributed towards reducing the power of these links, a system-level technique becomes necessary for more efficient power reduction, as the number of links increases with the scaling and increasing sizes of NoCs, and as application-specific knowledge becomes available. For example, power-aware-encoding techniques [8] such as Gray coding cannot be efficiently used, as the hardware cost in the encoder/decoder increases drastically as the system scales to a higher number of network components. As such, recent research focuses on turning links off and on in order to reduce power consumption, and has been adopted by several works [1, 6, 9–11], as certain links in the system are severely underutilized during a specific operational time frame [1]. Techniques such as DVFS (dynamic voltage and frequency scaling) applied to the link hardware, [9, 11] have been used to vary the link frequency and power according to link utilization; however, even when not data is sent across a link, static power is still being consumed, especially in multipipelined links with pipeline buffers in place. In addition, CMOS technology scaling is pointing towards an increased portion of the allocated power budget being consumed as static energy instead of dynamic energy; hence, switching on/off links instead of just selectively reducing their frequency/voltage levels offers better power saving advantages as links still do burn power even at lower (i.e., nonzero) voltage-frequency settings [12]. The majority of these on/off link dynamic power-management works employ traditionally a statically-computed threshold value on the link utilization, and based on that threshold value, the link is turned off for an amount of time and then is turned back on when the algorithm decides so. This of course is a pessimistic approach by nature, and imposes harder performance constraints. Recently, the use of control theory for managing candidate links for turning off has been proposed as an idea in [10], with promising results when compared to the statically-based approaches.

Motivated by the findings in [10], this paper proposes the use of artificial neural networks (ANNs) as a dynamic link power consumption management mechanism, by utilizing application traffic information. Based on their ability to dynamically be trained by variable scenarios, ANNs can offer flexibility and high prediction capabilities [13]. An ANN-based mechanism can be used to intelligently compute dynamically the threshold value used to determine which links can be turned off and on during discrete time intervals. The ANN receives link utilization data in discrete time intervals, and predicts the links that should be turned off or on based on the computed threshold. ANNs can be dynamically trained to new application information, and have been proven that they can offer accurate prediction results in similar scenarios [14]. ANNs can be efficiently designed in hardware provided they remained relatively small, through efficient resource sharing and pipelining. Furthermore, by partitioning the NoC, individual small ANNs can be assigned to monitor each partition independently, and in parallel monitor the entire network. This work also introduces topology-based directed training as a pretraining scheme, using guided simulation, which helps to minimize the large training set and the ANN complexity. This work extends our initial idea presented in [15] by several new contributions: (a) the ANN architecture has been redesigned, making it flexible and smaller through trade-off simulations involving the size and structure of the ANN and the offered power savings, (b) extended discussion on the architecture and its hardware implementation, (c) extended discussion on the simulation platform, synthetic traffic benchmarks and power modeling, and (d) extended results related to the power savings versus the performance penalty and the associated hardware overheads.

The rest of this paper is organized as follows. Section 2 discusses background and related work. In Section 3, we introduce the ANN-based approach for managing link power in NoCs. Section 4 presents the simulation framework and simulation results and analysis through various topologies and synthetic traffic, and Section 5 concludes the paper giving brief future research directives.

2. Background and Related Work

2.1. ANN Background and Motivation. An ANN is an information-processing paradigm that is inspired by the way biological neurons systems process information. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs learn by training and are typically trained for specific applications such as pattern recognition or data classification. ANNs have been successfully used as prediction and forecasting mechanisms in several application areas, as they are able to determine hidden and strongly nonlinear dependencies, even when there is a significant noise in the data set [14]. ANNs have been used as branch prediction mechanisms in computer architecture, as forecasting mechanisms in stocks [14], and in several other prediction applications. The ANN operates in two stages: the training stage and the computational stage. A neuron takes a set of inputs and multiplies each input by a weight value, which is determined in training stage, accumulating the result until all the inputs are received. A threshold value is then substracted from the accumulated result, and this result is then used to compute the output of the neuron based on an 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 the layers of an ANN. A neural network can be realized in hardware by using interconnected neuron hardware models, each of which is composed by multiplier accumulator (MAC) units, and a look-up Table (LUT) as a shared activation function. Some memory is also required to hold the training weights.

2.2. Related Work in Link Dynamic Power Management. Recently published research practice surveys such as [16] which outline the design challenges and lay the roadmap in future NoC design have emphasized the critical need to conduct research in NoC power management due to concerns of battery life, cooling, environmental issues, and thermal management, as a means to safeguard the scalability of general-purpose multicore systems that employ NoCs as their communication backbone. Link dynamic power management has been given significant attention by NoC researchers, as circuit-based techniques such as differential signals and low-voltage swing hardware using level converters do not seem to adequately address the power management problem [17, 18]. As such, there is a significant shift towards high-level techniques such as selective turning of links on and off. The challenge involved in those techniques includes the computation of the decision on whether a certain link is to be turned off, and when it will be turned back on. These decisions typically rely on information from the system concerning link utilization, and, so far, have been taken using a threshold-based approach. There have been attempts in dynamic link frequency and dynamic link voltage (DVFS) management with most using these thresholds as well.

Among the proposed techniques, some approaches use software-based management techniques such as the one in

[17], which proposes the use of reducing energy consumption through compiler-directed channel voltage scaling. This technique uses proactive power management, where application code is analyzed during static compilation time to identify periods of network inactivity; power management calls are then inserted into the compiled application code to direct on/off link transitions to save link power. A similar approach was also taken in [19] for communication power management using dynamic voltage-scalable links. Both of these techniques, however, have been applied to highly predictive array-intensive applications, where precise idle and active periods can be extracted. Hence, run-time variability, applicable to NoCs found in general-purposed multicore chips, has not been examined. Further the work in [20] proposes software-hardware hybrid techniques that extend the flow of a parallelizing compiler in order to direct run-time network power reduction. In this paper, the parallelizing compiler orchestrates dynamic-voltage scaling of communication links, while the hardware part handles unpredicted online traffic variability in the underlying NoC to handle unexpected swings in link utilization that could not be captured by the compiler for improved power savings and performance attainability.

Low-level, hardware-based techniques that determine on/off periods and manage the voltage and frequency, exhibit however better energy savings as they can shorten the processing time required for a decision whether to turn a link off or on to be made. The most commonly used power management policies deal with adjusting processing frequency and voltage (dynamic voltage scaling—DVS). The works in [5, 18] present DVS techniques that feature a utilization threshold to adjust the voltage to the minimum value while maintaining the worst case execution time. In [21], the authors propose that the dynamic voltage scaling is performed based on the information concerning execution time variation within multimedia streams. The work in [22] proposes a power consumption scheme, in which variable-frequency links can track and adjust their voltage level to the minimum supply voltage as the link frequency is changed. Furthermore, [11] introduces a history-based DVS policy which adjusts the operating voltage and clock frequency of a link according to the utilization of the link/input buffer. Link and buffer utilization information are also used in [9], which proposes a DVS policy scheme that dynamically adapts its voltage scaling to achieve power savings with minimal impact on performance. Given the task graph of a periodic real-time application, the proposed algorithm in [9] assigns an appropriate communication speed to each link, which minimizes the energy consumption of the NoC while guaranteeing the timing constraints of real applications. Moreover, this algorithm turns off links statically when no communications are scheduled because the leakage power of an interconnection network is significant. In general on/off links have, in most cases, been more efficient than DVFS techniques, as links, even if operating at a lower voltage, still consume leakage and dynamic power [1, 6]. These works therefore present a threshold-based technique that turns links off when there is low utilization, using a statically computed threshold. Given that static computation by nature is pessimistic, dynamic policies have been proposed. Research work in [23] proposes a mechanism to reduce interconnect power consumption that combines dynamic on/off network link switching as a function of traffic while maintaining network connectivity, and dynamically reducing the available network bandwidth when traffic becomes low. This technique is also based on a threshold-based on/off decision policy. Next, the work in [24] considers a 3D torus network in a cluster design (off-chip interconnection network) to explore opportunities for link shutdown during collective communication operations. The scheme in [25] introduces the Skip-link architecture that dynamically reconfigures NoC topologies, in order to reduce the overall switching activity and hence associated energy consumption. The technique allows the creation of long-range Skip-links at run-time to reduce the logical distance between frequently communicating nodes. However, this is based on application communication behavior in order to extract such opportunities to save energy. Finally the related work in [26] explores how the power consumed by such on-chip networks may be reduced through the application of clock and signal-gating optimizations, shutting power to routers when they are inactive. This is applied at two levels: (1) at a granular level applied to individual router components and (2) globally at the entire router.

Run-time link power management has recently gained ground in research to address the leakage issues as well. As links become heavily pipelined to satisfy performance constraints, link buffers and pipeline buffers contribute significantly in leakage power consumption. As such, the problem becomes significant with the increased on-chip NoC sizes, impacting both the power consumption as well as the thermal stability of the chip. Dynamic link management techniques have therefore been proposed; the work in [2] proposes an adaptive low-power transmission scheme, where the energy required for reliable communications is minimized while satisfying a QoS constraint by varying dynamically the voltage on the links. The work in [27] introduces ideas of dynamic routing in the context of NoCs and focuses on how to deal with links or/and routers that become unavailable either temporarily or permanently. Such techniques are a little more complicated than a threshold-based approach, and inhere performance overheads during each dynamic computation. As such, the work in [10] introduces the idea of an intelligent method for dynamic (run-time) power management policy, utilizing control theory. A preliminary idea of a closed-loop power management system for NoCs is presented, where the estimator tracks changes in the NoC and estimates changes in service times, in arrival traffic patterns and other NoC parameters. The estimator then feeds any changes into the system model, and the controller sets the voltage and frequency of the processor for the newly estimated frequency rate. Motivated by the promising results presented in [10], and the potential performance benefits of dynamic threshold computation techniques, this work proposes a dynamic, intelligent, and flexible scheme based on ANNs for dynamic computation of the threshold that determines which links can be turned off or on.

3. ANN-Based Threshold Computation Methodology

3.1. Static Threshold Computation for On/Off Links. The first step in realizing the proposed ANN methodology is to establish a framework for comparing whether an intelligent management is comparable to the nonintelligent case, not only in terms of energy savings, but also in terms of throughput and hardware overheads. As such, a trivial case, where a simple threshold mechanism was used to determine whether or not a link would turn off or back on, was first implemented using an NoC simulation framework and the Orion power models [28] (explained later in Section 4.1). The mechanism chooses an appropriate threshold based on which the links turn on and off. This trivial algorithm takes as input the link utilizations of all the links in the experimental NoC system, and outputs control signals based on a statically defined threshold; based on this threshold, the algorithm then decides which links are turned off and then back on. The statically-defined threshold was computed based on simulation observations from different synthetic traffic models and based on the observed power savings and throughput reduction when compared to a system without the mechanism. Figure 1 shows the real-time power savings for four synthetic traffic models, observed over a 4 × 4 NoC.

This method was introduced in [1], and the results presented therein as well as the experiments with our framework indicate that such mechanisms can be quite effective. However, a run-time mechanism, which can potentially benefit from real-time information stemming from the network, can potentially outperform this method. Such mechanism is described next. Furthermore, [1] uses an open-loop mechanism, prone to oscillations that potentially can limit both the attainable performance and also the power savings, as power is still used during the transition [10].

3.2. Mechanism Overview. The ANN-based mechanism can be integrated as an independent processing element in the NoC (PE), potentially located in a central point in the network for easy access by the rest of the PEs, and each base ANN mechanism can be assigned to monitor an NoC partition. Such cases are shown in Figure 2(a). Each base ANN mechanism monitors all the average link utilization rates within its region. These values are processed by the ANN, which computes the threshold utilization value for each link within its region, during each interval. The threshold value is then used to turn off any links in the region that exhibit lower utilization. Links which have been turned off remain off for a certain period of time. Experiments in related work [1, 10] indicate that such time should be within a few hundred cycles, as longer periods tend to create a vast performance drop-off (as the network congestion increases due to lack of available paths), whereas shorter periods do not incur worthy power savings. The proposed ANN mechanism uses a 100-cycle interval, during which all new utilization rates are received. This interval was chosen based on existing experiments in [1], which shows that a 100-cycle interval incurs better performance to power savings. The interval, however, is a system parameter, which can also be taken into consideration by the system training, and involves future work. During the interval span, the ANN computes and outputs the new threshold, which is then used by the link control mechanisms in each router to turn off underutilized links. The links remain off for another 100 cycles, and turn back on when a new threshold is computed. During the 100-cycle interval, links which are off, do not participate in the computation of the next threshold; instead, they are encoded with a sentinel value that represents them being fully utilized, so they are not kept off in two subsequent intervals. This reserves fair path allocation within the network.

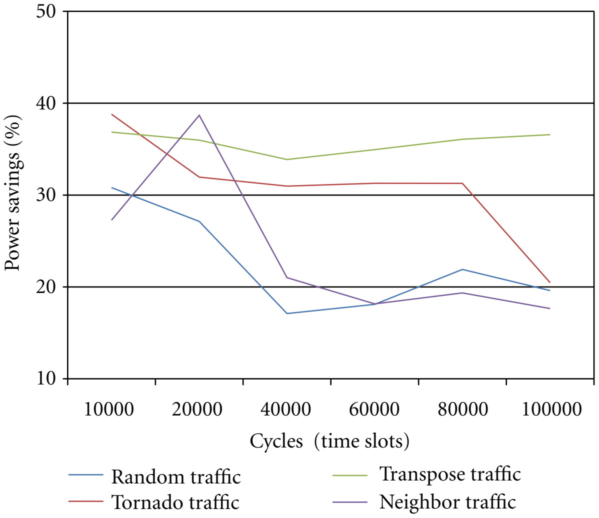

FIGURE 1: Power savings of a trivial threshold case compared to no on/off links case.

Each ANN-based mechanism follows a fully connected multilayer perceptron model [13, 14], consisting of one hidden layer of internal neurons/nodes and a single output-layer neuron. The activation function used in this work is the hyperbolic tangent function, which is symmetric and asymptotic, henceforth easy to implement in hardware as a LUT [29]. Furthermore, the specific function has been extensively used in several ANNs and its accuracy has been very good [13]. The ANN system is shown in Figure 2(b). The number of internal neurons was chosen to be the half of the summation of the input and output neurons [13]. The input neurons depend on the number of links that the system receives as feedback. As such, the size of the ANN depends on the number of inputs to the system. The output neuron chooses the corresponding threshold that best matches the pattern observed through the hidden layer neurons and outputs the threshold value to the link controller.

The neuron computation involves computing the weighted sum of the link utilization inputs. An activation function is then applied to the weighted sum of the inputs of the neuron in order to produce the neuron output (i.e., *activate* the neuron). Equation (1) shows how the output of a neuron is calculated.

Function which calculates the output of a neuron:

$$f(x) = K\left(\sum_i w_i g_i(x)\right). \qquad (1)$$

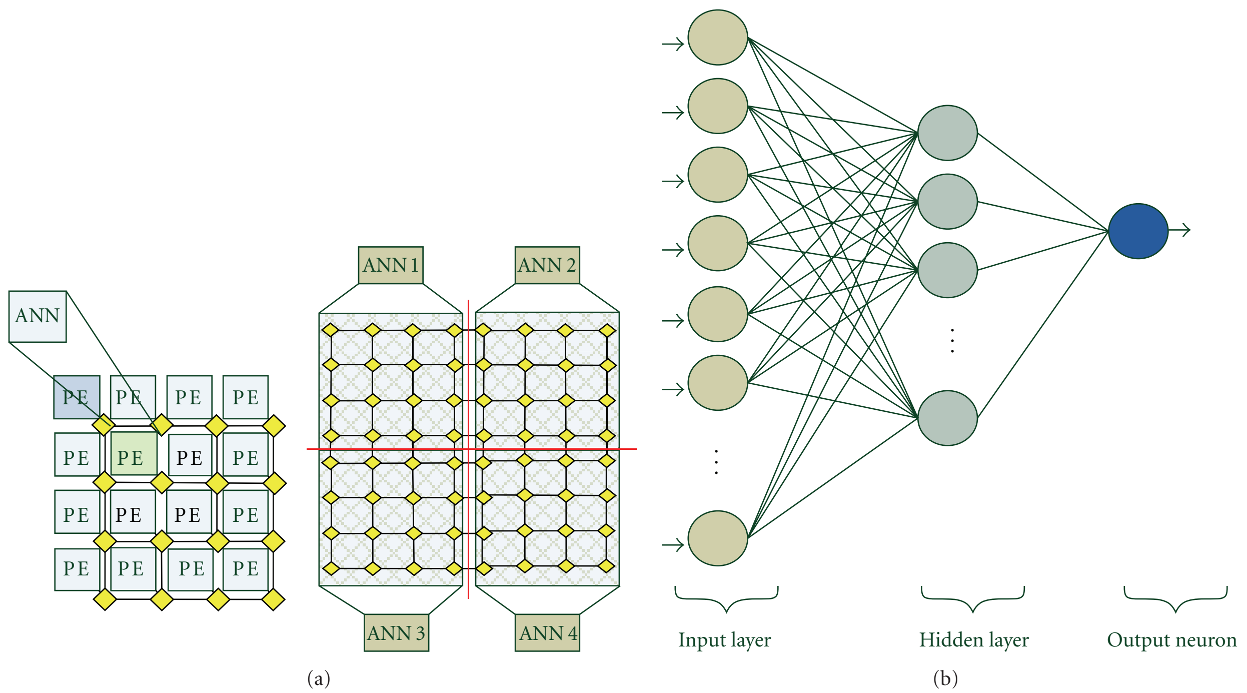

Figure 2: (a) ANN predictor with NoCs and an 8×8 network partition into four 4×4 networks with their ANNs; (b) Structure of the neural network.

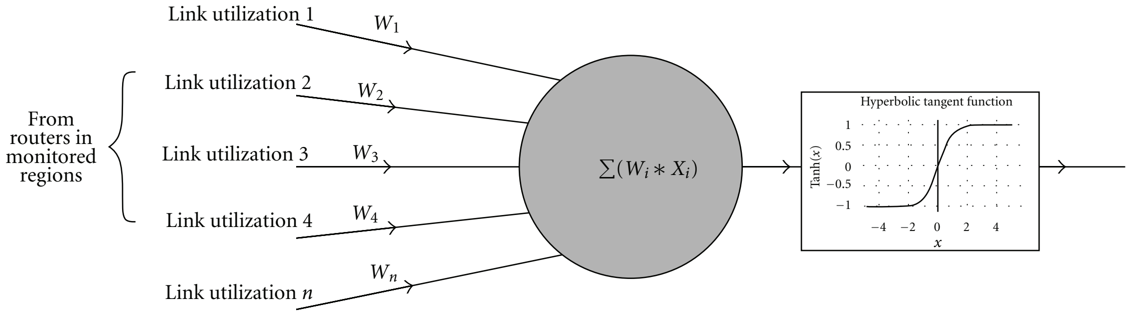

Figure 3: Neuron computations.

where K represents the activation function which is the hyperbolic tangent, w represents the weights which apply to the link utilization inputs which are represented by $g(x)$ input function. The overall procedure is shown in Figure 3.

3.3. ANN Training and Operation. The training stage can be performed off-line, that is, when the NoC is not used, and the training weights can be stored in SRAM-based LUTs for fast and on-line reconfiguration of the network. The network is trained using application traffic patterns, off-line, using the back-propagation ANN training algorithm [14]. In our experiments, we used synthetic traffic patterns and the Matlab ANN toolbox; the weight values were then fed to the simulator as inputs, where the actual prediction was then implemented and simulated. The operation of the ANN can be potentially improved, by categorizing the applications that a system will practically run. As such, for each application category (and subsequently traffic patterns with certain common characteristics), the ANN can be trained with the corresponding weights. Each training set can then be dynamically loaded during long operation intervals, where the system migrates to a new application behavior.

3.4. Intelligent Threshold Computation—ANN Size and NoC Scalability Issues. While ANNs are heavily efficient in predicting scenarios based on learning algorithms, they require careful hardware design considerations, as their size and complexity depend on the number of inputs received as well as the number of different output predictions (classes) that they have to do. NoCs consist of a large number of links which grow exponentially as the size of the NoC grows. Therefore, receiving link utilization and having to determine the threshold that controls which links are candidates for turning off and on would require an exponentially scalable ANN. As such, we devise a preprocessing technique, which identifies, based on simulation and observations, the set of candidate links for turning off and on, eliminating links which are almost always utilized. This depends obviously on the chosen network topology (e.g., in a 2D mesh topology, links that are likely to be less busy include links which are at the edges of the mesh, whereas central links are usually more active and can be left on all the time), so that the ANN mechanism can handle the output decision in a more manageable way. This can be aided by intelligent floor planning and placement of processing elements inside the NoC as well, but is beyond the scope of this paper. Through various synthetic traffic simulations, for each given NoC topology, the average utilization values for each link through various phases in the simulation are computed, and the links with the highest utilization values are always assumed that they will be on. Obviously this step reduces a little the effectiveness of the ANN, but it is necessary to minimize the size and overheads of the ANN both in terms of performance and in terms of hardware resources. This step has to be done for a given topology, prior to the ANN training. However, both steps (determining the links that the ANN will use, as well as the ANN training) can be done off-line, during the NoC design stage. The ANN training can also be done repeatedly whenever new application knowledge becomes available that might alter the on-chip network traffic behavior. This particular property of ANNs provides a comparative advantage against a statically computed threshold, making the NoC flexible under any application that it is required to facilitate. It must be stated that the number of links that will be considered as likely candidates for on/off activity (i.e., the ones which do tend to have low utilization during the pretraining stage) impact both the size of the ANN itself and the overall size of the mechanism (which involves logic that sends the appropriate control signals). Through the two steps, pretraining and training, each ANN can be trained and configured independently to satisfy its targeted NoC structure (topology and number of monitored links).

Furthermore, large NoCs can be partitioned into smaller regions. As such, a base ANN architecture can be assigned to monitor each region, and all the link utilizations of the routers of the NoC partition arrive at the ANN which is responsible for that region. The size of this NoC region, however, depends on two major factors; the incurred power savings that the corresponding base ANN offers, which depend on its ability to process and evaluate the input information, and the resulting ANN size and hardware overheads (and subsequently power consumed within the ANN) which grow exponentially as the size of the NoC region grows. Choosing a small NoC region will likely result in a small ANN, but will result in smaller savings since the ANN will not have enough information to compute a good threshold value. On the other hand, a large NoC region will provide the ANN with much more information and potentially result in a much better threshold value, but its size and overheads would reduce the power savings making the ANN ineffective. As such, we experimented with several NoC regions and base ANNs, comparing their hardware overheads (a product of the ANN power consumption and the gate count required to implement each ANN in hardware) and responding savings incurred with the computed threshold.

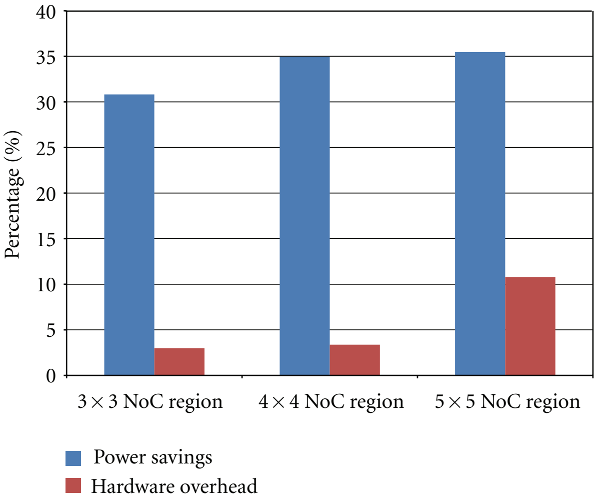

Figure 4: Power savings versus CMOS hardware overheads corresponding to various sizes of ANN monitoring regions in an NoC.

Figure 4 shows a comparison between hardware overheads (power × gate count) and power savings in the cases of 3 × 3, 4 × 4, and 5 × 5 ANN sizes for monitoring regions in an NoC. Results show that computation over a 4 × 4 NoC region offers satisfactory power savings and significantly less ANN overheads when compared to a 5 × 5 NoC region. A 3 × 3 NoC region does not provide enough information to the ANN in order to make accurate predictions. Based on these observations, we designed the base ANN system to monitor 4 × 4 NoC regions.

3.5. Base (4 × 4) Artificial Neural Network Operation. The ANN mechanism is responsible to compute for all the link utilizations the minimum values during each interval. Based on these values, the ANN calculates an optimal threshold. Figure 5 shows the procedure of the ANN mechanism for a 4 × 4 NoC partition. The ANN mechanism receives all the average link utilizations from all the links of the 4 × 4 NoC partition. These values are fed to the ANN in order to calculate an optimal threshold. Each router contains a control hardware monitor that measures the average link utilization for each of the four links in each router, and this value is sent to the ANN every n cycle (where n is the size of the time interval). If a router fails to transmit the values at a single interval, its value is set to sentinel value, which shows that its buffers are fully utilized. This mechanism acts also as a congestion information mechanism because links which are heavily active are not candidates to be turned off. The ANN uses the utilization values to find the threshold which will determine if a link is going to be turned off or on for the next n-cycle interval. As said earlier, we used 100-cycle intervals [1] (i.e., $n = 100$) in our simulations.

3.6. Base (4 × 4) Artificial Neural Network Hardware Architecture. One of the main advantages of ANNs is their simple hardware implementation when the number of neurons remains small and the activation function remains simple

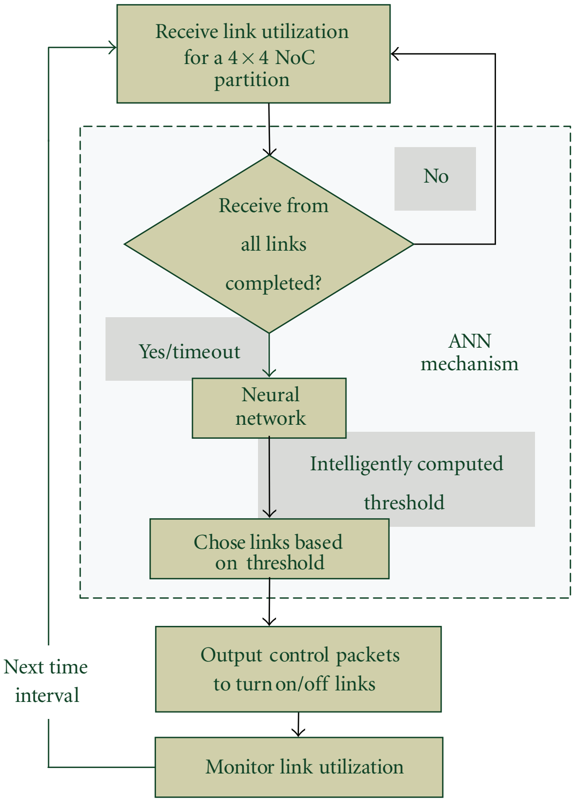
Receive link utilization for a 4 × 4 NoC partition
Receive from all links completed?
No
Yes/timeout
ANN mechanism
Neural network
Intelligently computed threshold
Chose links based on threshold
Next time interval
Output control packets to turn on/off links
Monitor link utilization

Figure 5: Main steps of a 4 × 4 ANN predictor.

[13]. The neuron operation can be designed efficiently in hardware since it can be modeled as a multiply-accumulate operation. The ANN hardware implementation depends on the number of hidden layer neurons; each neuron is implemented as a multiplier-accumulator (MAC) unit, with the accumulators being multiplexed for each neuron, so that the number of multipliers is minimized. The base ANN hardware architecture is shown in Figure 6. Utilization values for each link arrive and sorted through an input coordination unit, which distributes the values to each of the appropriate multipliers. The multipliers receive these values and through a shared weights memory, the corresponding weight. The weights and inputs product is then accumulated in the corresponding accumulator, with the entire process controlled via a finite-state machine controller. Each neuron has an assigned storage register, to enable data reuse; when one layer of neurons is computed, their outputs are stored inside a corresponding register. As such, the same hardware is reused for computing the next layer (i.e., from input layer to hidden layer and from hidden layer to output layer). When each neuron finishes its MAC computation, the result is then computed through the activation function LUT and propagates to the output neuron.

An ANN monitoring a 4×4 region in a torus topology, for example, receives 64 different inputs; if we are to assume that each router transmits a packet with its own link utilization during each interval, and if we also assume one packet per cycle delivered to the ANN during each interval, then, during each cycle, the ANN will receive at most 4 input

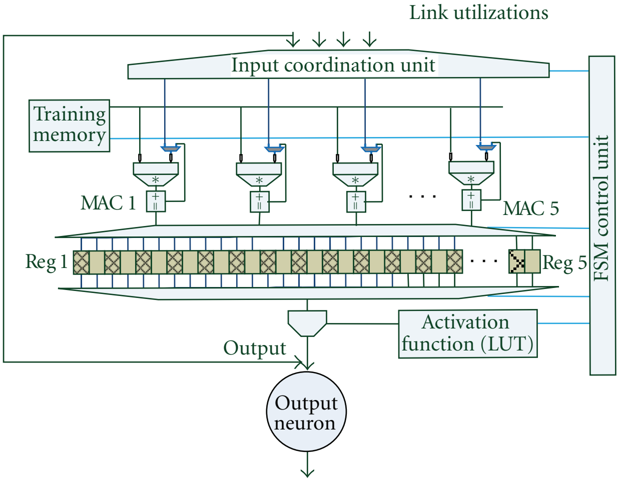

Figure 6: ANN hardware architecture and its hardware realizations.

values. Hence, if we use pipelined multipliers, we need only 4 multipliers for each ANN to achieve maximum throughput. The ANN therefore remains small and flexible, regardless of the size of the network it monitors. Furthermore, an ANN monitoring a 4 × 4 NoC partition receives 16 packets (one for each router); as such, it requires $16m$ cycles (where m is the cycle delay of each multiplier), plus 16 cycles for each accumulator, plus one cycle for the activation function plus one cycle for the output neuron, to output the new threshold (total of $16m + 18$ cycles). The overall data flow and architecture is shown in Figure 6.

3.7. ANN Hardware Optimization and Trade-Offs. In order to make the ANN architecture simpler and smaller we studied how the number of neurons of the hidden layer affect the total power savings of the system. Given that the 4 × 4 ANN monitors 16 routers, we need at least 8 input neurons [14]. Having eight neurons at the input layer of the ANN means that the hidden layer should have five neurons (based on the rule of thumb that a satisfactory number of the hidden layer neurons equals to half the number of input neurons plus one neuron) [14]. Three different ANNs were developed with five, four, and three neurons at the hidden layer, respectively. Figure 7 shows the power savings for these ANNs under the use of four different traffic patterns (Random, Tornado, Transpose, and Neighbor). Using four neurons therefore (instead of five), in the hidden layer exhibits the best power savings for all the traffic patterns. In addition, we studied how the bit representation of the training weights affects the threshold computation and subsequently the total power savings. Figure 8 shows how the bits used in representing the training weights influence the power savings of the system. As we can see 24, 16, 8, and 6 bits show similar power savings, but these savings are significantly reduced when 4 bits are used, due to reduced training accuracy. Based on the above, we selected the weight bit representation from 6 bits, which made the multiplier-accumulation hardware very small, requiring a 6-bit port for each weight and a 5-bit port for the utilization values.

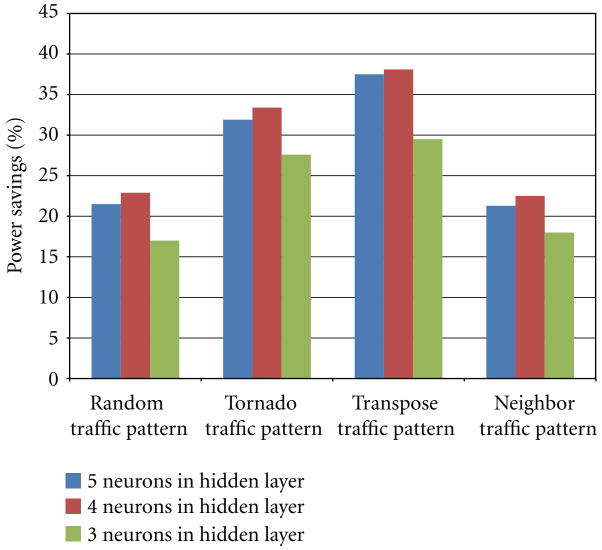

Figure 7: Power savings for five, four and three neurons in the hidden layer of the ANN.

4. Simulation and Results

4.1. Experimental Setup. In order to evaluate the ANN-based on/off link prediction mechanism, we developed a simulation framework based on the Java-based cycle-accurate *gpNoCsim* simulator (general-purpose simulator for network-on-chip architectures) [30]. The framework enables simulation of multiple topologies, utilizing dimension-ordered XY routing algorithm with virtual channel support and 4-stage pipelined router operation. The simulated router supports 64-bits flit width, 2 virtual channels per link and two buffers per virtual channel. The routers used are all the same, and we assume wormhole flow control. The framework supports various synthetic and user-defined traffic models. We experimented with a 4×4 mesh topology, and mesh and torus 8×8 topologies. Simulations are done over a range of 200,000 cycles, with a warm-up period of 100,000 cycles. In the 8×8 topologies, we partitioned the NoC into four regions of 4×4 routers/links, where each ANN-based model was assigned as responsible for monitoring. The ANN-based models monitored all links in their corresponding partition, all links were candidates for off/on, and all ANN results related to the size and operation of the ANN are given based on these architectural details.

Time was divided into 100-cycle intervals [1]; at the end of each interval, all routers in the NoC partition transmit their average utilization data for that span (computed via a counter and LUT-based multiplication with the reciprocal of the interval). A time-out mechanism equal to the expected delay of each router towards the ANN mechanism is imposed to maintain reasonable delays. The ANN receives one packet from each router with four utilization values, one for each port. The ANN then proceeds to compute the new threshold which is transmitted to each router through a control packet. Each router then turns off each link, depending whether or not its utilization value is above or below the new threshold. The router continues operation until the end of the new interval. It must be repeated that when a link is turned off or on, an extra 100-cycle penalty is inserted into the simulation to indicate the impact on the network throughput.

While the savings could significantly be improved using a more intelligent routing algorithm than the trivial XY dimension-ordered algorithm (DOR), we experimented with the XY algorithm and induced blocking when a link was off and the output buffer of that link fully utilized. In the case that a packet arrives in a router, and its destination link is off, the packet resides in the buffer until the link is turned back on. While this incurs a performance penalty, it is necessary to maintain correctness; this is a pessimistic approach obviously—a better routing algorithm would likely yield much better throughput and power savings, but is beyond the scope of this paper.

In order to study the power savings and the throughput of the dynamic ANN-based prediction algorithm for turning links on/off, we compare this to a static threshold-based algorithm and to a system without any on/off mechanism. Prior to discussing the simulation results, we first explain the power modeling followed in the experiments.

4.2. Power Modeling. We adopted the Orion power models for the dynamic power consumption of each router [28]. Router and link hardware were designed and synthesized in Verilog and Synopsys Design Compiler in order to obtain the leakage power values. We used a commercial CMOS 65 nm library, and a sequence of random input vectors, for several thousand cycles, and measured the leakage power of each router and link, through all computation cycles and combinations of events. The leakage values are then fed into the simulator, along with the Orion models for active power, and the overall power is computed. In addition, we take into consideration the start-up power consumed when we turn a link back on.

4.3. Simulation Results and Discussion. Using synthetic traffic patterns with varied injection rates (Random, Tornado, Transpose, and Neighbor) [31, 32], we first evaluated the power savings of the ANN-based mechanism when compared to the same system without any on/off link capability, and when compared to a system that employs a statically determined threshold. The traffic patterns, for which we experimented, are a superset of the patterns used to train the ANN; we measured power savings and the impact of the throughput on all the traffic patterns. However, in order to compute the power savings in the torus network, we follow the guided-training approach as described in Section 3.3, and we measure link utilizations in all possible partitions of the torus network to compensate for the toroidal links. The link utilizations with the least values (from all the link utilizations, from all the partitions of the torus network) are then passed through the ANNs. Figure 9 shows the comparison when targeting 8×8 mesh and torus NoCs. The power savings of the ANN-based mechanism are better than the savings in the cases of statically determined threshold

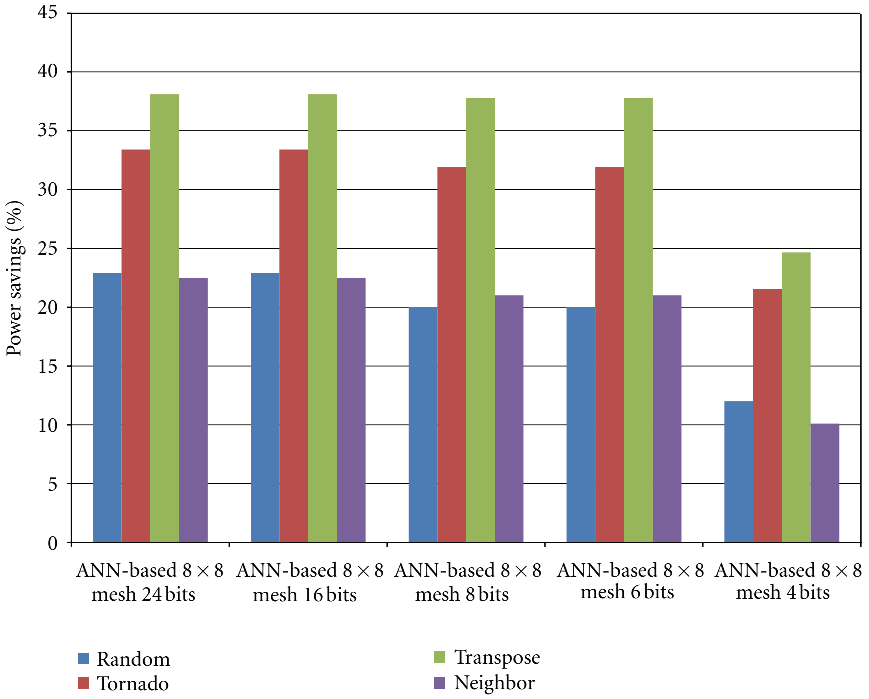

Figure 8: Power savings for different training weight bit representations.

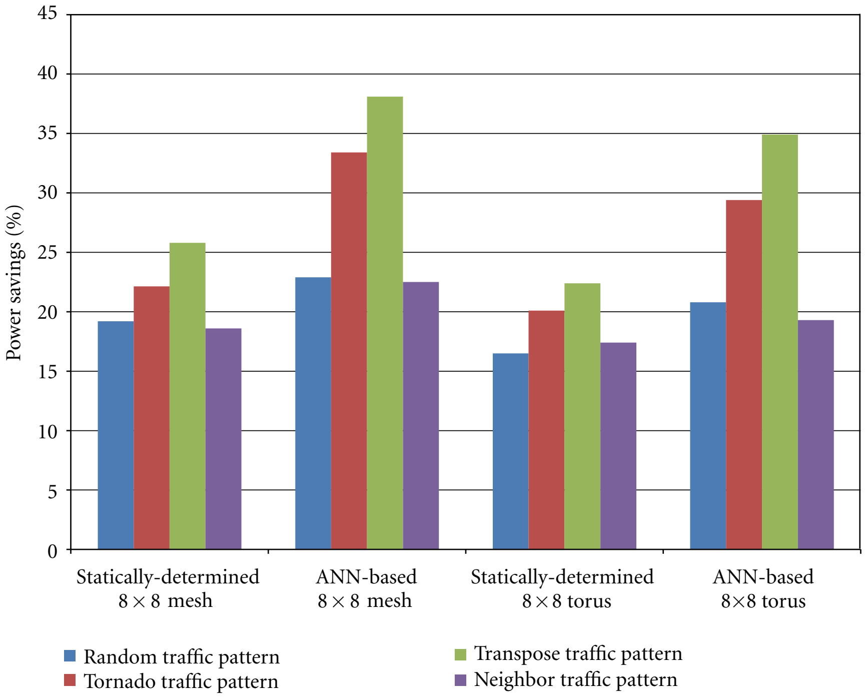

Figure 9: Power Savings for 8 × 8 mesh and 8 × 8 torus networks for the ANN-based technique, static threshold technique and no on/off technique.

and the case without any on/off links. The ANN-based mechanism can identify a significant amount of future behavior in the observed traffic patterns; therefore, it can intelligently select the threshold necessary for the next timing interval.

Next, we measure the impact of the throughput in each mechanism; while having no on/off mechanism obviously yields a higher throughput, the ANN-based technique shows better throughput results compared to statically determined threshold techniques. Figure 10 shows the throughput

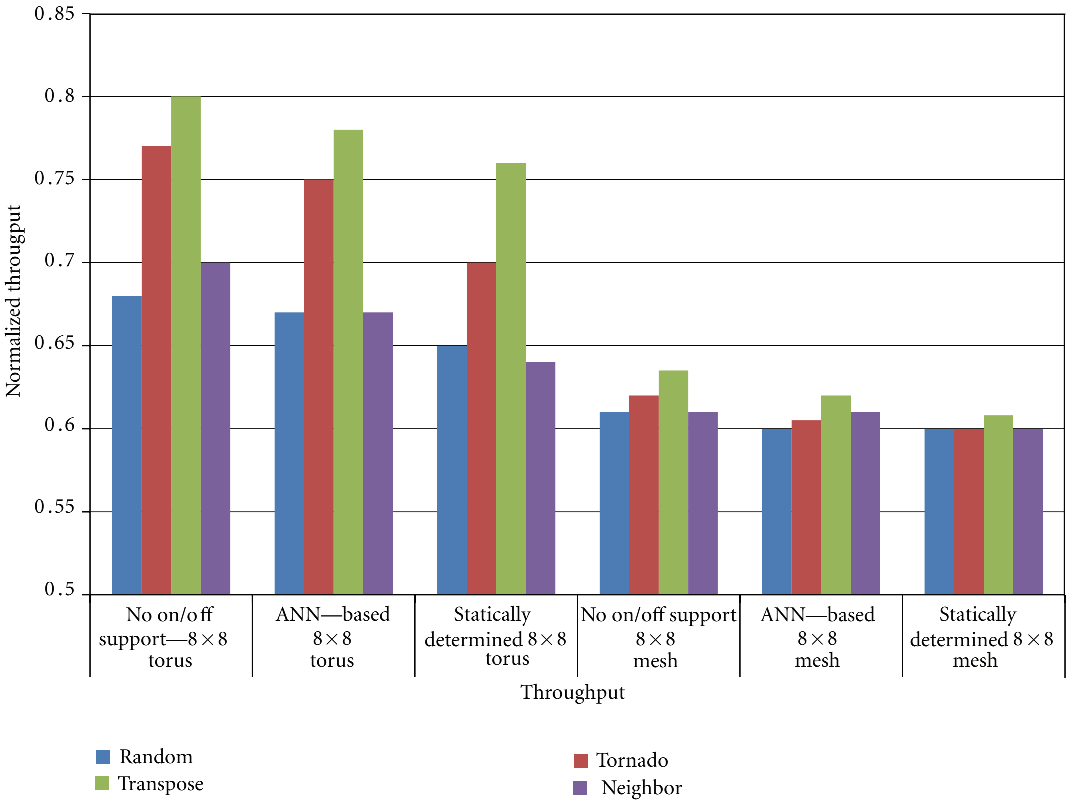

FIGURE 10: Average network throughput comparisons for 8 × 8 mesh and torus networks.

TABLE 1: Power savings/hardware overhead comparisons.

Related work	Characteristics	Power savings comparing to no algorithm	Hardware overhead
[1]	8 × 8 2D mesh topology, Uniform traffic	~37,5%—turning on/off 2 links per router	N/A
[11]	8 × 8 2D mesh topology, Pareto distribution—0.5 packet injection rate	~30%	500 equivalent logic gates per router port Delay ignored
Proposed ANN-based technique	8 × 8 2D mesh topology and 8 × 8 torus topology,uniform traffic	Up to ~40%—turning on/off links based on ANN prediction	4% of the NoC hardware for a complete 4 × 4 Mesh NoC

comparisons for an 8 × 8 mesh and an 8 × 8 torus network. The throughput values are normalized based on the number of the simulation cycles.

Figure 11 represents the normalized energy consumed in a 8 × 8 mesh network. We observe that the energy consumed using the ANN mechanism is less than the cases of statically-computed threshold and without on/off link management algorithm. The ANN exhibits a reduction in the overall energy, because of a balanced performance-to-power savings ratio, when compared to not having on/off links or when compared to static threshold computation.

Figure 12 presents the average packet delay in packets per cycle for the 8 × 8 mesh, when the ANN-based mechanism is used compared to the cases where no on/off mechanism is used and the statically computed threshold case. The ANN-based mechanism incurs more delay, but we believe that the delay penalty is acceptable when compared to the associated power savings.

4.4. ANN Hardware Overheads: Synthesis Results. To compute the hardware overheads of the proposed scheme, the ANN-based mechanism for one 4 × 4 NoC region was synthesized and implemented targeting a commercial 65 nm CMOS technology. The ensuing synthesized ANN-based controller and the associated hardware overheads in each router consume approximately 4 K logic gates (for comparison purposes, an NoC router similar to the one used in our simulation [29] consumes roughly 21 K gates),

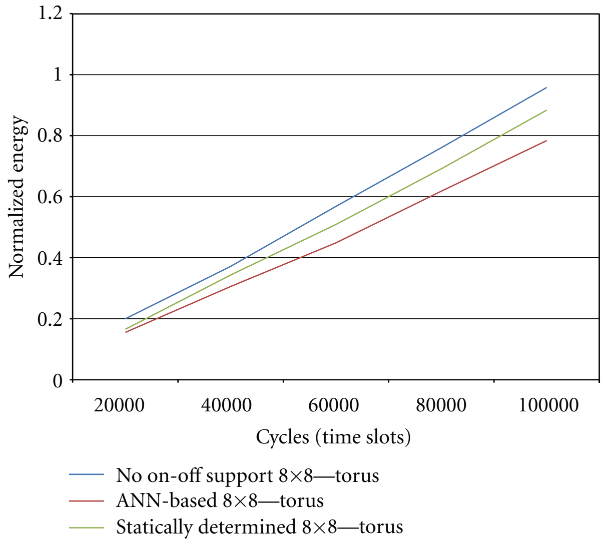

Figure 11: Energy consumption for a 8 × 8 mesh network.

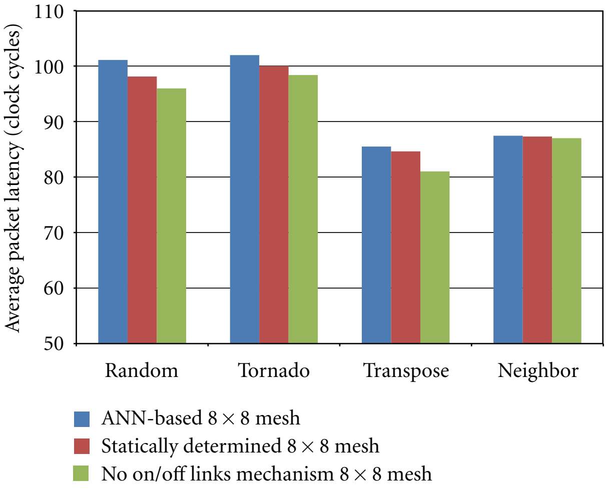

Figure 12: Average packet latency for the cases where ANN-based mechanism is used, when trivial case is used and when there is no on/off mechanism.

bringing the estimated hardware overhead for an 4 × 4 mesh network to roughly 4% of the NoC hardware.

4.5. Comparison with Related Works. Lastly, we briefly give a comparison with relevant related works that follow dynamic threshold techniques in Table 1. When compared to both [1, 11], the ANN-based prediction yields better power savings than having no prediction mechanism, while still maintaining lower hardware overheads. We must note that while [10] was the motivating idea behind our paper, it presented only a preliminary implementation of the idea, without enough information about hardware overheads and power savings in order to make an informed comparison.

5. Conclusions

This paper presented how an ANN-based mechanism can be used to dynamically compute a utilization threshold, that can be in turn used to select candidate links for turning on or off, in an effort to achieve power savings in an NoC. The ANN-based model utilizes very low hardware resources, and can be integrated in large mesh and torus NoCs, exhibiting significant power savings. Simulation results indicate approximately 13% additional power savings when compared to a statically determined threshold methodology under synthetic traffic models. We hope to expand the results of this paper to further explore dynamic reduction of power consumption in NoCs using ANNs and other intelligent methods.

References

[1] V. Soteriou and L.-S. Peh, “Dynamic power management for power optimization of interconnection networks using on/off links,” in *Proceedings of the 11th Symposium on High Performance Interconnects*, pp. 15–20, 2003.

[2] F. Worm, P. Thiran, P. Ienne, and G. De Micheli, “An adaptive low-power transmission scheme for on-chip networks,” in *Proceedings of the 15th International Symposium on System Synthesis*, pp. 92–100, October 2002.

[3] S. Kumar, A. Jantsch, J.-P Soininen et al., “A network on chip architecture and design methodology,” in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, pp. 117–124, 2002.

[4] L. Benini and G. De Micheli, “Networks on chips: a new SoC paradigm,” *Computer*, vol. 35, no. 1, pp. 70–78, 2002.

[5] K. Govil, E. Chan, and H. Wasserman, “Comparing algorithms for dynamic speed-setting of a low-power CPU,” in *Proceedings of the 1st Annual International Conference on Mobile Computing and Networking*, pp. 13–25, November 1995.

[6] V. Soteriou and L. S. Peh, “Exploring the design space of self-regulating power-aware on/off interconnection networks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 3, pp. 393–408, 2007.

[7] Y. Hoskote, S. Vangal, S. Dighe et al., “Teraflops Prototype Processor with 80 Cores,” Microprocessor Technology Labs, Intel Corporation.

[8] W.-C. Cheng and M. Pedram, “Low power techniques for address encoding and memory allocation,” in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC ’01)*, pp. 245–250, 2001.

[9] D. Shin and J. Kim, “Power-aware communication optimization for networks-on-chips with voltage scalable links,” in *Proceedings of the 2nd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS ’04)*, pp. 170–175, September 2004.

[10] T. Simunic and S. Boyd, “Managing power consumption in networks on chips,” in *Proceedings of the Design, Automation and Test in Europe*, pp. 110–116, 2002.

[11] L. Shang, L.-S. Peh, and N. K. Jha, “Dynamic voltage scaling with links for power optimization of interconnection networks,” in *Proceedings of the International Symposium on High Performance Computer Architecture*, pp. 91–102, 2003.

[12] Semiconductor Industry Association, “International Technology Roadmap for Semiconductors,” 2009, http://www.itrs.net/Links/2009ITRS/Home2009.htm.

[13] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: a tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[14] R. Schalkoff, *Artificial Neural Networks*, McGrow-Hill, 1997.

[15] A. Savva, T. Theocharides, and V. Soteriou, "Intelligent on/off link management for on-chipnetworks," in *Proceedings of the IEEE Annual Symposium on VLSI*, pp. 343–344, 2011.

[16] 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 Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 1, pp. 3–21, 2009.

[17] G. Chen, F. Li, M. Kandemir, and M. J. Irwin, "Reducing NoC energy consumption through compiler-directed channel voltage scaling," in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '06)*, pp. 193–203, June 2006.

[18] T. Pering, T. Burd, and R. Brodersen, "Voltage scheduling in the IpARM microprocessor system," in *Proceedings of the 2000 Symposium on Low Power Electronics and Design (ISLPED '00)*, pp. 96–101, July 2000.

[19] F. Li, G. Chen, and M. Kandemir, "Compiler-directed voltage scaling on communication links for reducing power consumption," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '05)*, pp. 455–459, November 2005.

[20] V. Soteriou, N. Eisley, and L.-S. Peh, "Software-directed power-aware interconnection networks," *ACM Transactions on Architecture and Code Optimization*, vol. 4, no. 1, pp. 274–285, 2007.

[21] E. Y. Chung, L. Benini, and G. De Micheli, "Contents provider-assisted dynamic voltage scaling for low energy multimedia applications," in *Proceedings of the International Symposium on Low Power Electronics and Design*, pp. 42–47, August 2002.

[22] J. Kim and M. Horowitz, "Adaptive supply serial links with sub—1V operation and per-pin clock recovery," in *Proceedings of the International Solid State Circuits Conference (ISSCC '02)*, pp. 1403–1413, 2002.

[23] M. Alonso, S. Coll, J. M. Martínez, V. Santonja, P. López, and J. Duato, "Power saving in regular interconnection networks," *Parallel Computing*, vol. 36, no. 12, pp. 696–712, 2010.

[24] S. Conner, S. Akioka, M. J. Irwin, and P. Raghavan, "Link shutdown opportunities during collective communications in 3-D torus nets," in *Proceedings of the 21st International Parallel and Distributed Processing Symposium (IPDPS '07)*, pp. 1–8, March 2007.

[25] C. Jackson and S. J. Hollis, "Skip-links: a dynamically reconfiguring topology for energy-efficient NoCs," in *Proceedings of the 12th International Symposium on System-on-Chip 2010 (SoC '10)*, pp. 49–54, September 2010.

[26] R. Mullins, "Minimising dynamic power consumption in on-chip networks," in *Proceedings of the International Symposium on System-on-Chip (SoC '06)*, pp. 1–4, November 2006.

[27] M. Ali, M. Welzl, and S. Hellebrand, "A dynamic routing mechanism for network on chip," in *Proceedings of the 23rd NORCHIP Conference*, pp. 70–73, November 2005.

[28] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: a power-performance simulator for interconnection networks," in *Proceedings of the International Symposium on Microarchitecture*, pp. 294–305, 2002.

[29] E. Kakoullit, V. Soteriou, and T. Theocharides, "An artificial neural network-based hotspot prediction mechanism for NoCs," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI '10)*, pp. 339–344, July 2010.

[30] H. Hossain, M. Ahmed, A. Al-Nayeem, T. Z. Islam, and M. M. Akbar, "gpNoCsim—a general purpose simulator for network-on-chip," in *Proceedings of the International Conference on Information and Communication Technology (ICICT '07)*, pp. 254–257, March 2007.

[31] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, San Francisco, Calif, USA, 2004.

[32] J. Duato, S. Yalamanchili, and L. M. Ni, *Interconnection Networks: An Engineering Approach*, Morgan Kaufmann, San Francisco, Calif, USA, 2003.