A Hybrid Analog-Digital Routing Network for NoC Dynamic Routing

Terrence S.T. Mak\(^1\), Pete Sedcole\(^1\), Peter Y.K. Cheung\(^1\), Wayne Luk\(^2\) and K.P. Lam\(^3\)

\(^1\)Department of Electrical and Electronic Engineering
Imperial College London, London, UK

\(^2\)Department of Computing, Imperial College London, London, UK

\(^3\)Department of Systems Engineering and Engineering Management,
The Chinese University of Hong Kong, Hong Kong

Abstract

Dynamic routing can substantially enhance the quality of service for multiprocessor communication, and can provide intelligent adaptation of faulty links during run time. Implementing dynamic routing on a network-on-chip (NoC) platform requires a design that provides highly efficient optimal path computation coupled with reduced area and power consumption. In this paper, we present a hybrid analog-digital routing network design that enables efficient dynamic routing on an NoC architecture. The digital part provides accurate real-time traffic estimation using a temporal cost evaluation and adaptation scheme. The analog network, which is distributed within the digital communication network, provides an efficient implementation for the optimal routing algorithm with extremely low power consumption. Our results demonstrate the effectiveness of the hybrid analog-digital design, with a significant improvement in latency over the static routing for random hot spot traffics.

1 Introduction

Network-on-Chip (NoC) architectures have recently been proposed as a promising solution to the increasingly complicated on-chip communication demands and challenges [2]. Such architectures consist of a network of regular tiles where each tile can be an implementation of general-purpose processors, DSP blocks, memory blocks and embedded reconfiguration modules etc. Communication among these tile-based modules is following a packet-switch or circuit-switch scheme where messages are transmitted among the processing elements. Typically, there is a router at each tile to provide next-hub forwarding and switching, which is similar to that found on computer and data networks.

In such a NoC environment, routing of messages (or packets) becomes a critical issue, which determines the inter-processor communication performance. Routing provides a protocol for moving data through the NoC infrastructure and also determine the path of data transport. The selection of communication pathway would greatly affect the latency of packets transmitted from the source to the destination and therefore can have significant impact on the overall traffic flow in the network. An intelligent routing mechanism is required to effectively control the traffic flow and minimizing the transportation latency.

Dynamic routing (or adaptive routing) has been widely adopted in computer and data network design. Using the

![Figure 1. Latency versus network load for different routing schemes. The figure shows how the employment of dynamic adaptive routing scheme would significantly delay the network critical point. (Data adapted from [5] Fig. 5)](image-url)
on-line communication patterns and real-time information. Dynamic routing can effectively avoid hot spots traffics or faulty components, and reduce the chances that packets being continuously blocked. It has been shown in [5] that by applying the dynamic adaptive routing strategy in the random traffic network, the network saturation point (or critical point) can be significantly delayed. Several adaptive routing algorithms within the context of NoC have been proposed and their performance was reported. An implementation of wormhole adaptive Odd-Even routing was described [9]. In [15], a minimal routing with partially adaptive protocols was investigated. Consensus results were found from these investigation that there is noticeable improvement on the packets latency and traffic load balancing. However, when comparing to its static routing counterpart, implementation of dynamic routers requires significantly more resources, dissipates more power and may introduce extra delay to packets in the network. This is due to extra logics are required for implementing the optimal path computation and which is usually computationally expensive. Although it has been shown that dynamic routing can provide a better quality of service in packet switching, it is still calling for a more hardware-economical design methodology for NoC applications.

In the last two decades, analog current-mode CMOS circuits have been widely studied and successfully implemented for a wide spectrum of applications from analog signal processing and communication [17] to low-power bio-implantable electronics (or neuromorphic circuits) [14]. The ultra-low power consumptions and small circuit area enable analog circuit design to realize computational intensive algorithms and operating control systems in many time-critical applications. In this paper, we propose a hybrid analog-digital on-chip network capable of implementing dynamic routing decisions. In the network, data transport is in the digital domain. The transport layer includes an evaluation of the latency which is used by a low-power analog control network to make routing decisions. The novel contributions of this paper are as follows: 1) A routing scheme for on-chip networks, which provides efficient evaluation of network congestion and minimal latency packet-forwarding at run-time. 2) A high-performance and low-power hybrid analogue-digital network for on-chip dynamic routing.

This paper is organized as follows: In section 2, the background theories are presented. Implementation and the architecture of the hybrid analog-digital routing network are described in section 3. Results are presented in section 4. Section 5 evaluates the results and concludes the paper.

2 Methods

Minimal cost (or shortest path) computation is fundamental among different dynamic routing implementation strategies, such as wormhole, store-and-forward and virtual cut-through [4]. The basic idea is that the routing algorithm always chooses the least congested route toward the destination. The least congested route can be found based on the shortest path computation where the path cost is the runtime congestion conditions. Since the traffic intensity and conditions are changing at run-time, the dynamic routing algorithm should be able to discover the congestions and perform shortest path computation at the same time. We propose a temporal cost evaluation and adaptation scheme, which provides an evaluation of congestions at run-time and that the information will be used for shortest path computation and minimal latency packet switching adaptively. The background of shortest path computation and the temporal cost evaluation and adaptation scheme are described below.

2.1 Background

The shortest path problem can be described as follows: Given a directed graph \( G = (V,A) \) with \( N = \{V\} \) nodes, \( M = \{A\} \) arcs, and a cost associated with each arc \( (i,j) \in A \), which is denoted as \( C_{i,j} \). The arc cost can be defined subjected to different applications, and in our case the cost is the delay for a packet traveling from node \( i \) to node \( j \). The total cost of path \( p = (v_0,v_1,...,v_k) \) is the sum of the costs of its constituent edges: \( w(p) = \sum_{i=1}^{k} C_{i-1,i} \). The shortest path of \( G \) from \( i \) to \( j \) is then defined as any path \( p \) with cost \( w(p) \) is \( \min \{w(p) : i \rightarrow j\} \).

Consider an \( N \)-node single-destination shortest path problem, where the network is defined as a set \( V \) of nodes \( \{v_i, i = 1,...,N-1\} \) and a set \( A \) of arcs \( \{a_i\} \) with nodes \( \{v_j, j \neq i\} \) with costs \( C_{i,j} \) and \( v_N = K \) is the destination node. The Bellman-Ford algorithm [1, 7] defines a recursive procedure in step \( k \), to find the new estimate \( \tilde{V}^{(k)}(v_i) \) for the expected shortest path cost from \( v_i \) to the destination \( K \) using the previous estimates in step \( k - 1 \), known as dynamic programming.

\[
\tilde{V}^{(k)}(v_i) = \min_{a \in A(v_i)} [C_{i,j} + \gamma \cdot \tilde{V}^{(k-1)}(v_j)] \tag{1}
\]

where \( \gamma \) is the discount factor that determines the importance of estimates that is far away from the current node \( v_i \). Thus, the estimate of the path cost from the current node \( v_i \) to the designation node \( K \) by taking \( k \) steps can be expressed as

\[
\tilde{V}^{(k)}(v_i) = C_{i,1+i} + \gamma \cdot C_{1+i,1+i+2} + \gamma^2 \cdot C_{i+2,i+3} + \ldots + \gamma^k \cdot C_{K-1,K} \tag{2}
\]
Thus, Eq. 1 is the recursive form of expected delay evaluation from Eq. 2. In addition, the optimal decisions at node \( i \), \( \hat{a}(v_i) \), that leads to the shortest path can be readily obtained from the argument of the Bellman equation as follows

\[
\hat{a}(v_i) = \arg \min_{a \in A(v_i)} \left[ C_{i,j} + \gamma \cdot V^{(k-1)}(v_j) \right]
\]  

(3)

where \( A(v_i) \) represents the set of arcs that are emerging from node \( v_i \).

### 2.2 Temporal Cost Evaluation

From the above shortest path formulation, the path cost has been assumed to be a constant value during the computation. This is applicable for the case of static routing, in which the path costs are estimated at pre-fabricated time and optimal routing and scheduling are determined based on these estimated costs. In the dynamic routing case, we allow the cost to be evolved according to the congestion of the network. Therefore, the cost will not be a static value, and it will capture the congestion situation of paths at runtime. A method to obtain an approximate of this cost at run-time is necessary.

Intuitively, the cost function explicitly models the latency (waiting time) that packets will spend on a particular node. If the latency is large, it implies that the congestion is likely to happen at this node and vice versa. Therefore, by measuring the average waiting time that packets spend on the node can provide a good approximation for the path congestion conditions. Thus, we denote the temporal cost function \( C_{i,j}(k) \) for the path cost from node \( i \) to \( j \) and which is the average waiting time of \( k \) previous packets. This cost function can then be used for computing the shortest path. Specifically, the waiting time for the \( k \)-th packet is denoted as \( N_{i,j}(k) \). Based on the exponential smoothing method [8], the average of \( k \) previous packets’ waiting time is as follows:

\[
C_{i,j}(k) = (1-\alpha)N_{i,j}(k) + (1-\alpha)N_{i,j}(k-1)\alpha + \\
(1-\alpha)N_{i,j}(k-2)\alpha^2 + ... + \\
(1-\alpha)N_{i,j}(1)\alpha^{k-1}
\]  

(4)

\[
C_{i,j}(k+1) = (1-\alpha) \cdot N_{i,j}(k) + \alpha \cdot C_{i,j}(k)
\]  

(5)

where \( \alpha (0 < \alpha < 1) \) is called the smoothing constant that determines the number of samples to be considered.

This function can be readily converted into a recursive form (in Eq. 6), which can be easily implemented in hardware.

An alternative way to obtain this estimate is by using Little’s formula [12], in which the expected waiting time \( W = L/\lambda \) where \( L \) is the expected number of customers in the queue and \( \lambda \) is the mean arrival rate of packets into the queue. In this case, \( \lambda \) can be obtained by measuring the inter-arrival time. Also \( L \) can be obtained by measuring the number packets in the FIFO waiting to be sent. However, the division and several multiplications can be expensive in hardware. But this approach can reduce the size of the FIFO required. It is of the designer’s disposal for choosing either approach for computing the expected waiting time.

### 2.3 Distributed Shortest Path Computation

The shortest path algorithm can be implemented using a distributed parallel approach [3], which is originally proposed with the motivation of computational speed-up. The on-chip network provides distributed processing elements and communication channels, which are similar to the computer networks. Distributed shortest path computation can be realized by taking advantage of the distributed environment to provide efficient shortest path computation. In the following, we will present a distributed shortest path computation network and mapping of NoC communication tasks to this network for dynamic routing. A mesh network topology will be used throughout the paper for illustrating the idea. However, the methodology is not limited to the mesh topology and simple modification can be made for tackling other topologies, such as torus and Butterfly Fat-Tree (BFT).

Suppose a communication task graph and their mappings to a tiled-based architecture are given as in Fig. 2(a). Nodes represent the tiles or processing elements, dotted lines represent the communication tasks and arrows represent the possible routes for transporting packets to the destination. A distributed shortest path computation network can be readily constructed based on the network from figure (a). The corresponding shortest path computation network is shown in Fig. 2(b). The nodes and the topology of the network is the same as the original architecture. As the Bellman recursive equation (Eq. (1)) is implemented at each node, the arrows here represent the direction of the information flow. The output from node \( i \) is \( \tilde{V}(v_i) \), which is the expected cost to the destination from node \( i \). The inputs for node \( i \) are \( \tilde{V}(v_j) \), for all node \( j \) are the neighbors of node \( i \). When each node is performing the computation as stated in Eq. (1), the expected cost \( \tilde{V}(v_i) \), for all \( i \), will converge to the optimal value with a speed of \( O(\log_2(N-1)) \) where \( N \) is the total number of nodes [11].

Note that this methodology enables shortest path computation for multiple-sources single-destination communications tasks. In other words, for each shortest path compu-
Figure 2. (a) A 4 × 4 tile-based mesh architecture where nodes are the tile/PE and edges are the communication channels. The dotted lines are examples of the communication tasks. (b) The corresponding shortest path computation network for the network in (a) and node t16 is the destination node for the communication tasks.

3 Implementation

3.1 Hybrid Analog-Digital Routing Network

Fig. 3 shows the design of a router, which enables dynamic routing. Basically, the router design is similar to conventional routers, which were depicted in the context of NoC [9]. There are two additional blocks that implement the dynamic routing algorithm. They are the congestion monitor and the controller for the switch. The congestion monitor implements the temporal cost evaluation algorithm that compute the estimate of average latency that packets will spend in this router for each of the queues. Specifically, a time tag, which used for measuring the waiting of a packet, will be added to each of the packet when they enter the queue. At the switch, the time tag will be examined and compared to the current time, so that the time spend that the packet has spent in the router can be obtained.

On the other hand, optimal path computation is implemented using analog circuit. Since for a standard mesh or torus on-chip network, there are usually no more than four directions that a packet can choose as the next forwarding hub. In other words, if the optimal path computation is realized using analog circuit, there is no need to implement a high resolution Analog-to-Digital (ADC), as the directions are only one or two bits information. By taking the advantage of this fact, the analog circuit becomes a suitable candidate for implementing the minimum value network. As can be seen from Fig. 3 label (2), the analog optimal path computation outputs the directions to the control through ADC. Label (5) and (6) are the inputs of the cost estimates from the digital congestion monitors of neighbor nodes. These cost estimates are used for the shortest path computation.
3.2 Analog Implementation of the Shortest Path Computation Network

A continuous-time extension of the recursive Bellman-Ford algorithm can then be formulated from Eq.(1), to give [11]

\[
\frac{dV_{v_i}(t)}{dt} = -\tau_i V_{v_i}(t) + \min_{a \in A(v_i)} [C_{i,j} + \gamma \cdot V_{v_j}(t)]
\]  

(7)

where \(V_{v_i}(t)\) represents the approximated \(\hat{V}_{v_i}\) in the continuous time, and \(\tau_i\) is a first-order lag constant which is implementation-dependent. Notice that when \(dV_{v_i}(t)/dt\) is approximated by the discrete-time finite difference \(V_{v_i}(t + 1) - V_{v_i}(t)\) and \(\tau_i = 1\), Eq.(2) reduces back to Eq.(1).

Eq.(2) provides a natural setting for parallel computational circuits, without the need of imposing sequential constraints as in the traditional digital Bellman-Ford equation (Eq.(1)).

3.2.1 Current-Mode Shortest Path Computation

An analog voltage-mode implementation using the binary relation inference network (BRIN) [11] for solving the dynamic programming was previously reported. In [13], a current-mode implementation for implementing Eq. 2 was presented. The current mode design is with a simpler and more effective current-mode design using only lossertake-all (LTA or min) circuits [6, 16] previously developed for neural-fuzzy systems to solve the shortest path problem. By extending the previous design, we introduce the digital-analog conversion (DAC) and analog-digital conversion (ADC) to provide an interface for communication between the analog network and the digital computational units.

Fig. 4(a) shows a typical mesh network architecture with \(N^2\) nodes. Node \(S_{N,N}\) is the destination node and the arrows represent the possible route to \(S_{N,N}\). A analog distributed shortest path computation network can be constructed following the topological configuration of network in Fig. 4(a). Fig. 4(b) is the analog current-mode realization of the the shortest path computation network. The arrow represents the analog current direction and the topology of the analog circuit is constructed followed the original mesh network. Each computational unit implements the Eq. (2). Labels 2, 5, and 6 represent the interfacing with the digital router and labels 1, 3 represent the interconnections between different analog computational units. This figure signifies that the analog computational network is continuously computing the shortest path while digital keeps feeding the new cost estimates into the network.

3.2.2 The Analog Computational Unit

The right hand side of Eq.(1), in which a minimum operation is performed over all possible actions from node \(v_i\), provides the necessary framework for using the LTA in a dynamic programming unit for solving an \(N\)-node directed graph problem. The circuit schematic is shown in Fig. 5. It comprises of three blocks, which are the digital-analog conversion (DAC), analog-digital conversion (ADC) and a Loser-take-all (LTA) circuit. Inputs from labels 5 and 6 are digital inputs, which are voltages of either zero or Vdd. Current mirror Q1, Q2, Q3 and Q4 are for amplifying the reference current to convert the digital input into analog current inputs. Specifically, the output current from this DAC would be \(\sum_{i=0}^{4} q_i 2^i\) for inputs (5) and \(\sum_{i=0}^{4} p_i 2^i\) for inputs (6). Current mirrors Q1-Q4 amplifying the current accordingly based on the geometric parameters \(W\) and \(L\) modification during circuit layout. Inputs 3 and 4 are current inputs, which are coming from other computational units (see Fig. 4(b)). There are two outputs from the minimum circuits. The \(V_{out}\) outputs the minimum current, which represents the expected latency. This will become the input of (3) and (4) for other computational units. The transistor \((F_0)\) gives the output \(\max(A-B,0)\). In this case, if \(A > B\), the output will be a non-zero value, otherwise, the output will be zero. Based on output from \((F_0)\), we can readily convert it into digital signal by using a typical algorithmic ADC. The output \((D_0)\) is a digital signal for the router controller, which
Figure 4. (a) A typical mesh network with $N^2$ nodes. Node $S_{NN}$ is the destination node and the arrows represent the possible route to $S_{NN}$. (b) A computational network that corresponds to the network from (a). The arrow represents the analog current direction and the topology of the analog circuit is constructed following the original mesh network. Each computational unit implements the Eq. (2). Labels 2, 5, and 6 represent the interfacing with the digital router and labels 1, 3 represent the interconnections between different analog computational units.

4 Results and Discussion

4.1 Hybrid Network Evaluation

To evaluate the performance and network convergence of the hybrid analog-digital routing network, we have implemented the design using CMOS transistors using the AMI 0.50 micron model. Networks of size $3 \times 3$ and $4 \times 4$ were studied. By using the SPICE simulation, we can obtain the network response curve by measuring the current outputs from each of the analog computational units and the digital outputs from the ADCs. Fig. 6(a) shows the network convergence for a $3 \times 3$ network. The output, which is labeled (1) from Fig. 5, was measured and the output from four different units that were changing with time are shown in the figure. The curves show the evolution of currents from the network responding to the changes of the digital input (the latency estimate) at $t = 500$ ns. It can be observed that the analog responses to the digital inputs and converges to a steady state in 100 ns. The lower panel shows the relative error of the output signals relative to the numerical solutions, which was obtained from the algorithmic computation using Matlab. This result show that the error for the shortest path computation is reduced to less than 5% in less than 100 ns. Fig. 6(b) shows the results for a $4 \times 4$ network simulation. The output from nine different units that were changing with time are shown in the figure. A similar observation can be made that the network converges to the optimal shortest path in around 150 ns.

In order to compare our mixed-signal and a purely digital implementation for power consumption and hardware area, a digital counterpart of the analog circuit was implemented. Ripple carried adders was used for the addition circuit and the minimize circuit was implementing by using a subtractor and a multiplexer. The same number of bits as the DAC were used in the digital circuits. As the digital circuit was also using the same technology model and implemented using CMOS transistors, the fair comparison for the power consumption and area can be obtained. We compared the area by counting the number of transistors used between the two different circuits. The area comparison result is shown in Fig. 7(a). We compared the area of the analog circuit and its digital counterpart for different network size varying from 4 to 16. The analog circuit area is generally much smaller than its digital counter part. For smaller network, such as 4-node network, analog area is around half of the digital design (64 versus 124 for analog and digital respectively). For a large network such as 16-node network, analog area is only 20.6% of the digital area (453 versus 2192 for analog and digital respectively).

Comparing circuit area based on transistor count may not be accurate. A transistor in an analog circuit is gen-
eral much larger than that in a digital circuit. However, this is after partially compensated the reduced routing found in analog circuits. Therefore, the area results presented above is probably optimistic in favour of the analog implementation. More accurate comparison will be presented after the circuits are implemented as physical layout.

We also compared the power consumption between the analog and its digital counterpart. The power analysis was performed using SPICE, in which the average power dissipation in conjunction with the transient analysis. The circuit was injected with a set of random test vectors and was allowed to run for one section for the transient analysis with $V_{dd} = 3V$. The analog design generally consumes significantly less power than its digital counterpart. Even for a small network, 4-node network, the analog circuit consumes 19% of the power that the equivalent digital circuit consumes (0.95 mW versus 5 mW for analog and digital circuits respectively). For a larger network, 16-node network, the analog circuits consumes 23% of the power that the digital circuit consumes (12.7 mW versus 54.8 mW for analog and digital circuits respectively).

In general, the analog design consumes less area and power when comparing to its digital counterpart. This advantage is more significant for a large network, such as 16-node network. From our experiments, an analog design

\footnote{The average power $P$ for a time interval $(t_1, t_2)$ is computed by using the trapezoidal rule approximation to evaluate the integral $P(t_1, t_2) = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} P(\tau) d\tau$.}

with a 4-bit DAC is well robust for handling the minimal network computation. Further analysis on the affects of interconnect parasitic and substrate noise, which are the major sources of noise in the digital circuit, on the analog circuit will be investigated in the future.

4.2 Dynamic Routing under Random Traffic

To evaluate the performance gains that can be achieved with the temporal cost adaptation and the hybrid routing network, a higher level simulation platform is required. Simulink has become a popular platform for rapid prototyping and several Simulink tool boxes were developed that can synthesis the cycle-accurate model into VHDL and EDIF netlist for hardware realization, such as Xilinx System Generator [18]. We have developed a cycle-accurate interconnection packet-switching network model based on Simulink. The router is implemented based on fixed-point cycle accurate model and can be readily translated into synthesizable VHDL for further implementations. The analog part of the routing network was implemented by using the Euler integration method by converting the differential equations into differences equations, which can be readily implemented using a cycle-accurate simulator.

We simulate several mesh networks with different routing schemes and design parameters under random hot spot traffic patterns. Under each load and configuration, net-
work of sizes $4 \times 4$ are simulated, which uses X-Y static routing and temporal cost adaptation dynamic routing, respectively. In X-Y routing, the packet follows the row first, then moves along the columns towards the destination [4]. The efficiency of the two different types of routing is evaluated through latency curves. Similar to other work in the literature, we assume that the packet latency spans the instant when the first packet is created, to the time when the last packet is ejected to the destination node, including the queuing time at the source. In this case, the processors generate messages at the time intervals chosen with exponential distribution. Similar to the work presented in [9], hot spot traffic pattern is used in the simulation.

Fig. 9(a) shows the frequency versus packet latency for the static and dynamic routing from the same simulation model. The traffic load injected into the network is smaller than the critical traffic load. Both distributions are having the peak on latencies between zero and 500 cycles. However the distribution for the static routing appears to have a much longer tail. The variance of the packet delay is much larger than the one with dynamic routing, which has all packets arrived in 2000 cycles or less. The dynamic routing scheme provides a better quality of service than the static routing in random traffic. In fig. 9(b), the latency-throughput figures under the hot spot random traffic pattern is shown. The average latency is measured for different traffic injection loads, which has been normalized. The robustness of different network designs that tolerates traffic loading can be found in this figure. For traffic load less than 0.2, both static and dynamic routing networks provide a stable service for packets and the latency is roughly 50 cycles. When the traffic load is larger than 0.2, the latency from a static routing network begins to increase and when the traffic load equals to 0.4, the latency increases sharply and that the network is saturated. For the dynamic routing network, the latency increases slowly even for traffic load larger than 0.2. The latency increases rapidly after traffic load is larger than 0.5. The network with dynamic routing has a much larger tolerance to the random traffic loadings.

It has been shown that dynamic routing can provide significant improvement on latency for random traffics. In addition, our results show that a network with dynamic routing can provide an overall improvement on traffic flow and the average waiting time for the entire network can be reduced.

Fig. 8 shows the evolution of average waiting time for all queues in the network over 20000 cycles. The average waiting time is an important estimate indicating that the congestion status and traffic flow over the network. The larger of the average waiting time implies that the traffic congestion is more serious in the network. In the figure, the static routing result shows a much higher average waiting time implying that there are more congested nodes in the network.
Figure 7. The comparison between the analog and its digital counterpart on hardware area and power consumption for the optimal path computation. (a) The transistor count comparison. (b) Power consumption comparison.

For the network using dynamic routing, the average waiting time is reduced significantly. The results suggest that introduction of dynamic routing may provide an effective flow control for random traffics.

5 Discussion and Conclusion

A hybrid analog-digital routing network for network-on-chip dynamic routing has been presented. The hybrid design methodology, by taking the advantages of low-power analog computation and accuracy of digital computation, can significantly reduce the hardware area and power dissipation for on-chip dynamic routing while provide effective run-time dynamic routing. Our results show that the hybrid design takes only 20% of the area and consumes 23% of the power of that the digital equivalent requires. The hybrid approach provides a hardware economical solution for implementing on-chip dynamic routing. The implementation of hybrid routing network is based on a temporal cost evaluation and adaptation scheme, in which packets are forwarding to a path with the minimum expected waiting time. Our results show that with the adaptation scheme, average latency and overall average packets waiting time can be reduced significantly.

The proposed design methodology enables hardware cost-effective dynamic routing on chip and would substantially enhance the quality of service for on-chip communication tasks. The proposed dynamic routing scheme can be readily adapted into other hybrid routing schemes including hybrid adaptive/deterministic routing to achieve the best performance and design tradeoff and judiciously customized to match the given application traffic patterns.
Figure 9. The comparison between dynamic routing using temporal cost adaptation and static X-Y routing. (a) The frequency distribution of packet delays. (b) The average latency versus normalized traffic injection load.

References


