Winning with Pinning in NoC

Ahmed Abousamra, Rami Melhem  
University of Pittsburgh  
Computer Science Department  
{abousamra, melhem}@cs.pitt.edu

Alex Jones  
University of Pittsburgh  
Electrical and Computer Engineering Department  
akjones@ece.pitt.edu

Abstract—In Chip Multiprocessors (CMPs), on-chip interconnect carries data and coherence traffic exchanged between on-chip cache banks. Reducing communication latency is critical for improving the performance of applications running on CMPs. Communication latency is affected by network design, cache organization, and application design. Previously proposed techniques for reducing router latency using express virtual channels or hybrid circuit switching effectively reduce communication latency. However, our analysis of communication traffic of a suite of scientific and commercial workloads on a 16-core cache-coherent CMP showed low utilization of circuits due to repeated establishment and tear down of circuits. In this paper, we explore circuit pinning, an efficient way of establishing circuits that promotes higher circuit utilization, adapts to changes in communication characteristics, simplifies network control, and allows smarter routing techniques due to the stability of configured circuits. Comparison with state of the art packet switched and hybrid circuit switched interconnects across different cache organizations demonstrates the benefits of our technique.

I. INTRODUCTION

CMP systems rely on the on-chip interconnect to facilitate communication between different cores, cache banks, memory controllers, and shared on-chip functional units. Latency of the communication over the interconnect has a significant effect on CMP system performance. This effect will continue to increase as the technology scales down enabling many more cores to be put on a chip.

Previous research has attempted to reduce communication latency by a variety of ways. Many designs have been proposed to reduce global hop count: Flattened butterfly topology [10] uses high radix routers to enrich connectivity; Concentrated mesh [2] shares each router among multiple nodes; Hybrid Ring/Mesh interconnect [4] breaks the 2D mesh interconnect into smaller mesh interconnects connected by a global ring; Hybrid Mesh/Bus interconnect [6] uses buses as local interconnects and uses a global mesh interconnect to connect the buses; and in 3D stacked chips, a low-radix and low-diameter 3D interconnect [18] connects every pair of communication points in at most 3 hops. Another approach for reducing communication latency is reducing router latency: express virtual channels [11] and hybrid circuit switching interconnect [8] reduce router latency by allowing part of the traffic to bypass the router pipeline. Schemes that reduce hop count can be combined with schemes that reduce router latency to further improve communication latency.

Reducing router latency is achieved in [8], [11] by having preconfigured, complete or partial, circuits between source and destination nodes. However, there is a latency cost for establishing a circuit, which makes a circuit only effective if that cost can be amortized through enough circuit reuse. In an ideal setting, circuits can be simultaneously established between every pair of communicating source and destination nodes. However, this would require a huge amount of wiring that is not practical given the limited die area. Hence, the best alternative is to simultaneously establish the most important circuits that would achieve the greatest improvement in communication latency given the available network resources.

Our analysis of communication traffic of a suite of scientific and commercial workloads on a simulated 16-core CMP using hybrid circuit switching interconnect [8] showed two interesting points: (1) circuit utilization is limited, and (2) the average time between sending two consecutive packets from the same source to the same destination is large, which explains the low circuit utilization; often, a circuit is not there to be reused as it gets torn down to allow other conflicting circuits to be established. These findings prompted the exploration of circuit pinning, an alternative that aims at keeping circuits in place, thus promoting their reuse. No resources are wasted when the pinned circuits are not utilized since those resources are also used for normal packet switching.

Circuit pinning provides another advantage: stability of the configured circuits, which allows for effective partial-circuit routing, in which partial as well as complete circuits are used, thus further improving circuit utilization.

In this paper we explore the benefits of circuit pinning and describe how circuits are established and reconfigured over time to cope with changes in communication characteristics. The paper is organized as follows. Section II briefly describes the previously proposed hybrid circuit switched interconnect. Sections III and V explain the details of circuit pinning and partial circuit routing. Section IV describes implementation assumptions. Simulation methodology and results are presented in Sections VI and VII. Finally, Section VIII concludes the paper.

II. PRELIMINARIES

A. Hybrid Circuit-Switched Interconnect

A hybrid circuit-switched interconnect [8] supports circuit-switched (CS) and packet-switched (PS) traffic. In packet-switching, when the head flit of a packet arrives at an input port of a router, it is first decoded and buffered in its input virtual channel (VC) buffer in the buffer write (BW) stage.
of the router. Second, it goes through the route computation (RC) stage during which routing logic computes the output port for the packet. Third, the head flit arbitrates for an output virtual channel (VC) in the virtual channel allocation (VA) stage. After arbitration succeeds and an output VC is allocated, the head flit competes for the switch input and output ports during the switch allocation (SA) stage. Finally, the head flit proceeds to traverse the cross bar in the switch traversal (ST) stage, followed by traversing the link in the link traversal (LT) stage. Body and tail flits of the packet skip the RC and VA stages since they just follow the head flit.

Several techniques can be applied to shorten the critical path through the router stages. Lookahead routing, where route computation occurs in the BW stage, removes RC from the router critical path. Aggressive speculation [12], [13] allows VA and SA stages to occur simultaneously, where a head flit is allowed to enter SA stage assuming it will succeed in allocating an output VC in the VA stage, but is not allowed to enter ST stage if it fails in allocating an output VC, at which case the head flit will have to go through the VA and SA stages again. Switch and link traversal can be performed together in one stage. Note that further reduction of the router critical path to only one stage is possible under low loads through aggressive speculation and bypassing [1].

Circuit-switching, on the other hand, works by configuring a circuit from a source tile $S$ to a destination tile $D$. A circuit is configured by recording at each intermediate router which input port should be connected to which output port during the SA stage. To allow intermingled circuit- and packet-switched flits, an extra bit, called circuit field check (CFC) is added to each flit to indicate whether the flit is circuit- or packet-switched. The CFC bit is checked when the flit enters the router. If the CFC bit is set, the flit is allowed to bypass directly to the switch traversal stage, otherwise it is buffered in the appropriate virtual channel buffer and routed as packet switched. CS flits have higher priority than PS flits. The switch allocator receives signals from the input ports indicating the presence or absence of incoming CS flits, and accordingly determine which input ports can send PS flits to the next router.

### B. On-Demand Circuit Configuration

One way of configuring circuits is to establish them as they are needed, which is the approach taken in [8]. A dedicated setup network is used to establish circuits. The setup network uses packet-switching to route single flit messages, and thus does not need virtual channels.

The on-chip data interconnect is comprised of one or more interconnect planes. To send a data packet, $p_i$, from tile, $S$, to tile, $D$, either $p_i$ is sent on an already established circuit from $S$ to $D$ on one of the interconnect planes, or if there is no such circuit, one of the interconnect planes is chosen to establish the circuit. $S$ sends a circuit setup request on the setup network specifying the destination $D$ and the chosen data plane on which to establish the circuit. $S$ does not wait for the circuit to be established, but rather sends the data packet immediately behind the circuit setup request. When $S$ wishes to subsequently send another packet, $p_j$, to $D$, it can be sent on the established circuit if it is still in place, i.e., if the circuit is not torn down during the time between the two packets $p_i$ and $p_j$ are sent.

When an existing circuit, $C_{old}$, from $S$ to $D$ is torn down at an intermediate router, $R_k$, to allow a new conflicting circuit, $C_{new}$, to be established, $R_k$ asserts a reconfiguration signal at the input port, $ip_j$, of $C_{old}$ so that an incoming CS packet at $ip_j$ is buffered and routed as packet-switched (the CFC bit is reset). In addition, a circuit removal notification packet is injected on the setup network and sent to $S$ to notify it of the removal of the circuit.

### III. Hybrid Circuit-Switched Interconnect with Pinned Circuit Configuration

Instead of configuring circuits on-demand as described in section II-B, we propose to periodically configure circuits and keep the new circuit configuration stable for an interval of time, then repeat the process. We are motivated by our analysis of communication traffic of a suite of scientific and commercial workloads from the SPLASH-2 [16] and PARSEC 1.0 [3] benchmarks on a simulated 16-core cache-coherent CMP using four planes of hybrid circuit switching interconnect with on-demand circuit configuration. We found the percentage of flits traveling on circuits to be low (Fig. 1) indicating low circuit utilization. We also found that the average time between sending two consecutive packets from the same source to the same destination can range from hundreds to thousands of processor cycles (Fig. 2), which explains the low circuit utilization; often, a circuit is not there to be reused as it gets torn down to allow another conflicting circuit to be established. Having a stable circuit configuration would promote circuit reuse and avoid circuit thrashing. However, periodic configuration of circuits is necessary to cope with changes in communication characteristics.

![Fig. 1. % of flits traveling on circuits from source to destination](image)

### A. Circuit Set up and Pinning

We assume an $N$-tile CMP in which a tile consists of a processor, private L1 (instruction and data), an L2 bank, a

---

1 Figures 1 and 2 were produced using the simulator described in section VII with SNUCA L2 and 1MB L2 bank size.
directory to maintain coherence, and a network interface card (NIC). The NIC is the interface point between the tile and on-chip interconnect. It is responsible for sending and receiving packets. In case there are multiple interconnect planes, the NIC decides on which plane to send the next packet. In our proposed circuit pinning scheme, the NIC keeps statistics about the communication between its tile and other tiles. Specifically, an NIC at tile \( i \), \( 0 \leq i < N \), tracks the number of packets sent from tile \( i \) to every tile \( j \), \( j \neq i, 0 \leq j < N \). Thus, each NIC has \( N - 1 \) storage elements to maintain the number of packets sent to each unique destination and a single adder per interconnect plane for updates. The number of bits, \( b \), required for each storage element depends on the length of the time interval during which statistics are gathered and the clock frequency of the interconnect. With \( N \) NICs in the system, storage overhead complexity is \( O(bN^2) \). For our simulations, 16 bits per storage element were sufficient to keep track of the number of sent messages per time interval. Thus, in a 256-tile CMP, the total storage overhead would be 512 bytes per tile.

Our goal is to maximize the percentage of on-chip data traffic that travels on circuits to improve network latency. Unfortunately, due to limited die area, we cannot establish a circuit between every pair of communicating tiles. Thus, assuming temporal locality of communication, we can divide time into intervals, \( T_1, T_2, \ldots \), each of equal length, \( t_p \). This allows the gathering, during each time interval \( T_i \), of communication statistics at each tile, \( S \), on the number of packets sent to every other tile, \( D \neq S \). Based on the gathered statistics, we attempt to establish circuits from \( S \) to its most frequent tile destinations. The new circuit configuration is kept stable for the duration of the next time interval, \( T_{i+1} \).

Assume that setting up the new circuits takes time \( t_s \). During \( t_s \), the new circuit configurations will only be recorded in the routers of the data interconnect planes but will not be activated, i.e., the old circuit configuration will remain in effect during \( t_s \). To ensure that the transition to the new circuit configuration does not cause incorrect routing, we specify a period of time, \( t_f \), to flush in-flight CS packets out of the interconnect. During \( t_f \), all tiles cease to send new CS packets, and only send PS packets. After \( t_f \) passes, the new circuit configuration is activated and NIC statistics counters are all re-initialized to zero. The new circuit configuration is kept stable until the end of the time interval. In the following section we present two algorithms for setting up circuits.

### B. Circuit Configuration Algorithms

1) **A Centralized Algorithm:** This algorithm uses a centralized controller to which all NICs are connected for handling the configuration of the circuits that will be active during the next time interval, \( T_{i+1} \). At the end of a time interval, \( T_i \), every NIC sends to the centralized controller the list of \( N - 1 \) other tiles, i.e., message destinations, ordered by most important to least important. We define the most important destination of an NIC to be the one the NIC sent the most number of packets. Similarly, we define the least important destination to be the one the NIC sent the least number of packets. In an implementation this can be accomplished with one additional storage cell and a comparator per node using a hardware implementation of bubble sort.

Assuming there are \( k \) data planes, the centralized controller performs \( k \) iterations. In each iteration, the controller attempts to create the next important circuit for every NIC on one of the \( k \) data planes. If the controller fails to create a circuit for an NIC, \( N_i \), due to conflicting resources - it attempts to create the next important circuit of \( N_i \).

Our proposed controller is similar to the controller for a time division multiplexed (TDM) crossbar designed in [7], which receives as input an \( N \times N \) request matrix specifying the required circuits to establish between \( N \) possible source and destination nodes. To establish a circuit from \( S \) to \( D \), the controller checks that the output port on \( S \) and the input port on \( D \) are available, i.e., are not already assigned to other circuits. This is done by checking two matrices storing availability information of input and output ports. The time slots - on which circuits are established - of the TDM crossbar [7] correspond to the interconnect planes comprising our on-chip data network. In our controller, the check for input and output ports availability is replaced by checking the availability of all links on the network path from \( S \) to \( D \), where a link is defined by a pair of router input-output ports. Our controller has matrices storing availability of links on each of the \( k \) data planes. Searching the matrices of all the \( k \) planes is done in parallel. For each data plane, search indicates whether the circuit can be established or not. If the circuit can be established on more than one plane, we choose to establish it on the least numbered data plane.

2) **A Distributed Algorithm:** Another alternative is using the setup network and allow tiles to simultaneously establish circuits. Qiao and Melhem [14], and Yuan et. al [19] studied distributed circuit reservation for optical networks using time division multiplexing, and wavelength division multiplexing. We present a similar distributed two phase circuit reservation algorithm: A tile sends a circuit reservation (CR) message (one control flit) on the setup network. The source tile indicates in the CR message the list of possible data planes on which the circuit maybe established. Each router on the setup network...
network tracks the status of the input and output ports on the corresponding routers of the data interconnect planes. At the beginning of the algorithm, all input and output ports of the routers of all data interconnects are marked as *available*. When a circuit is established, the status of the ports on the circuit path is changed to *unavailable*. The port status is set to *reserved* while a circuit is being established. A port marked *reserved* on plane $i$ would eventually be marked *unavailable* if the circuit is established on plane $i$, or marked *available* otherwise.

At each router, the circuit establishment algorithm: (1) Identifies the input and output ports required by a CR message. (2) Waits until the status of each of the required ports is resolved to either *available* or *unavailable*. A CR message that will wait should be moved to the end of the buffer to avoid blocking other messages. (3) Chooses the available input and output port pairs on the same plane and changes their status to *reserved*. (4) Updates the list of planes in the CR message and passes the message to the next router. If a CR message cannot proceed, i.e., cannot reserve a complete circuit on at least one plane, the router drops the CR message and injects a circuit free (CF) message, which travels the same path as the dropped CR message but in reverse, to free reserved ports. If a CR message reaches its destination and succeeds in reserving a complete circuit on at least one plane, one of those planes is chosen to establish the circuit. The destination router injects a circuit confirmation (CC), which - similar to the CF message - travels the same path as the CR message in reverse, to confirm the establishment of a circuit on the chosen plane and free the reserved ports on the other planes. Note that multiple rounds of the algorithm are executed until each tile exhausts its list of connections to establish.

IV. IMPLEMENTATION ASSUMPTIONS

In this section, we briefly describe some details regarding the implementation of Hybrid Circuit Switching. The goal is to clarify and unify the implementation for both on-demand and pinning circuit configuration schemes for fair comparison.

A. Router Design

When a circuit, $C_{old}$, is broken at an intermediate router, $R_k$, due to the establishment of a new conflicting circuit, $C_{new}$, a CS packet, $p$, traveling on $C_{old}$ will have to become packet-switched starting at $R_k$. This requires buffering the flits of $p$ in one of the virtual channel buffers at the input port, $ip_j$, through which $p$ enters $R_k$. Since CS packets bypass the router VA stage and hence are not associated with virtual channel id, we assume that CS packets travel on a special channel and add to each input port of the router a special buffer, which we call $VC_{CS}$, to store the flits of CS packets that become packet-switched, as shown in Fig.3.

B. Delayed Circuit Reconfiguration at a Router

Given that only the head flit of a packet contains the routing information, i.e., destination tile and possibly the source tile, while the body and tail flits follow the same route of the head flit, breaking a configured circuit, $C_{old}$ - to configure a new conflicting circuit $C_{new}$ - at an intermediate router, $R_k$, while a CS packet is traversing $R_k$, would require adding the routing information to body and tail flits. Additionally, it would complicate the router design since some non-head flits would require going through the virtual channel allocation stage. Therefore, if a CS packet, $p$, traveling on $C_{old}$, is traversing $R_k$, we delay breaking $C_{old}$ until the tail flit of $p$ is seen.

C. CFC bit versus lookahead signal

As mentioned in section II-A, a CFC bit is required to differentiate CS and PS flits. Since a CS flit takes one cycle to traverse a router, a packet consisting of multiple flits cannot be delayed at any intermediate router unless it will become packet switched. Therefore, CS flits have higher priority than PS flits. Consequently, when a router detects a CS flit traveling on circuit, $C_i$, and in order to allow that CS flit to traverse the router, it may have to preempt up to two PS packets that were allocated the crossbar input and output ports of $C_i$ at the SA stage. An alternative to the CFC bit is to send a lookahead signal one cycle in advance of sending a CS flit so that the next router knows that there is an incoming CS flit. This allows more efficient switch allocation since only the input ports with no incoming CS packets can participate in the SA stage. For a fair comparison we assume the use of lookahead signal for all hybrid-circuit-switched interconnects.

V. ROUTING ON PARTIAL CIRCUITS

Stable circuit configurations provide an opportunity to further improve circuit utilization by using partial circuit routing. For example, assume tile $S$ wishes to send a packet, $p$, to tile $F$ but there is no circuit from $S$ to $F$ (we will use route $SF$ to denote the route from $S$ to $F$). Further assume there is an established circuit, $C_{SD}$, from $S$ to some node $D$, where routes $SF$ and $SD$ share path, $SK$, (note that we may have $K = D$ or $K = F$ or $K \notin \{D, F\}$). In this case, $p$ can traverse the shared route $SK$ as a CS packet on the circuit $C_{SD}$. After existing $C_{SD}$ at $K$, the packet is routed to its
destination, $F$, on the PS network, if $K \neq F$. Stability of configured circuits allows the NIC at each tile $S$ to compute for each circuit, $C_{SD}$, originating at $S$, the destinations that can partially use it. This computations need only be done once at the end of each round of circuits configuration, then used when sending packets for the duration of the circuits pinning interval.

Partial circuit routing is used unintentionally with on-demand circuit configuration when a packet is sent on a broken circuit before the sending tile receives notification of the circuit removal. However, use of partial circuits can be planned. To enable it, we add to the lookahead signal a unary counter specifying the number of remaining links to be traversed as a CS packet. In a 2D square mesh tiled CMP of $N$ processors, the maximum number of hops to reach a destination is $2\sqrt{N} - 1$. Thus, the unary counter would consist of $2\sqrt{N} - 1$ bits. Only one bit of the counter will be set to 1 while the rest are 0s. The router examines the least significant bit (LSB) of the received counter. If LSB is 0, then the incoming packet should be routed as a CS packet and the counter bits are shifted right one bit and sent to the next router. If, on the other hand, the LSB is 1, then the incoming packet will be buffered in the $VC_{CS}$ buffer of the input port of the router and will be routed as a PS packet. Note that to send a CS packet all the way to the destination of a circuit, the unary counter bits should all be set to 0s.

A possible disadvantage with partial circuit routing is that it may reduce the percentage of packets traveling on complete circuits from source to destination. When a partially circuit routed packet, $p_k$, becomes packet switched at an input port, $ip$, of some router, $R_i$, $p_k$ is written to $ip$’s $VC_{CS}$ buffer. Similar to the express virtual channel buffer management technique [11], if there is not enough free space in the $VC_{CS}$ buffer to accept another full packet, $R_i$ sends a stop signal to $op$, the output port on the previous router, $R_{i-1}$, connected to $ip$. The stop signal indicates that $R_i$ cannot accept any more CS packets at $ip$. When flits are sent out of the $VC_{CS}$ buffer and there is enough free space to accommodate at least one more full packet, $R_i$ sends a resume signal to $op$ indicating it can now receive CS packets at $ip$. Thus, a stop signal temporarily disables a link of a circuit, rendering the rest of the circuit links unusable until the link is re-enabled by a resume signal. During the time a circuit link is disabled, other CS packets will become packet switched when they reach disabled circuit links. However, simulation results show that the benefits gained from partial circuit routing greatly outweigh this possible disadvantage.

VI. SIMULATION METHODOLOGY

In this paper we compare four interconnect designs:

a) Packet Switched Interconnect (PSI): We simulate a cycle accurate packet switched interconnect. In our simulation we use one plane of the PSI with a 64 byte link width. All control and data messages are one flit long. For routers, we simulate a 3-stage pipeline: BW, VA+SA, ST+LT (see section II-A). Each input port has 4 virtual channel buffers, each buffer capable of storing 5 flits.

b) Hybrid Circuit Switched Interconnect with On-Demand Circuit Configuration (CSID): In our simulation we use 4 planes of the CSID with a 16 byte link width, for an aggregated 64 byte link width across the 4 planes. Control messages are one flit long, while data messages are 4 flits long. A PS packet goes through a 3-stage pipeline, while a CS packet traverses the router in one cycle. Each input port has 5 virtual channel buffers, each buffer capable of storing 5 flits. Four of the virtual channel buffers are used for PS packets, while the fifth one is used for buffering an incoming CS packet that would become packet switched until it reaches its destination.

c) Hybrid Circuit Switched Interconnect with Pinned Circuit Configuration (CSIP): The design of CSIP is similar to CSID, except that, circuits are established every preset time interval instead of on-demand. After circuit establishment, they are pinned until it is time to reconfigure the circuits, as described in Section III. In simulation we set $t_f = 50ns$ and $t_p = 100000ns$. We simulate the centralized circuit configuration algorithm\footnote{The authors of [7] report that their controller takes $t_s = 760ns$ on an FPGA to configure a set of non-conflicting circuits for a system of 16 processors, which is the same system size in our simulations.} described in section III-B1, but we chose $t_s = 8000ns$ to be large enough to allow for the distributed circuit configuration algorithm.

d) Hybrid Circuit Switched Interconnect with Pinned Circuit Configuration and Partial Circuit Routing (CSIPR): This is similar to CSIP but with the additional use of partial circuit routing as described in Section V.

All four interconnects use X-Y routing, and we employ a critical word first approach, in which a stalling instruction can proceed as soon as the first word of the requested memory block is received.

We implemented the interconnects’ simulators on top of Simics [15], which is a full system simulator. We configured Simics to simulate tiled CMP consisting of 16 SPARC 2-way in-order processors, each clocked at 2 GHz, running Solaris 10 operating system, and sharing a 4 GB main memory with 55 ns (110 cycles) access latency. The processors are laid out in a $4 \times 4$ mesh. Each processor has a 32 KB (divided equally between instruction and data) private 4-way set associative L1 cache with 64 byte cache blocks (access latency: 1 cycle). Interconnect routers are clocked at 1 GHz. We use benchmarks from the Splash-2 [17] and Parsec 1.0 [3] suites. We simulate the parallel section of each benchmark. Benchmark input parameters are listed in Table I. To promote communication locality, we enforced thread binding to processors for all benchmarks except for canneal, because we found it would require extensive code changes than the other benchmarks.

Since cache organizations as well as cache sizes affect the communication traffic on chip, we demonstrate the benefits of circuit pinning through a variety of configurations. We simulate two cache organizations: distributed shared L2 (SNUCA) [9] and private L2 [5]. For the SNUCA L2, the physical
memory address space is statically mapped to L2 banks in granularity of cache block. For the private L2, a distributed directory is used for maintaining data coherence. We used 16-way set associative directory banks (access latency: 8 cycles). We set the directory bank size so that it has a number of entries twice the number of cache blocks of an L2 bank\(^3\).

The L1 and L2 caches are write-back and maintain the inclusion property. We maintain cache coherence through a MESI protocol. Given that the size of available cache on chip may vary depending on how much die area is allocated to cores and caches, we show results for two L2 bank sizes: (1) 16-way set associative 1 MB (access latency: 15 cycles), and (2) 8-way set associative 256 KB (access latency: 8 cycles). An L2 cache block is 64 bytes, and an L2 bank is located at each tile.

### VII. Simulation Results

This section presents simulation results for 16 possible system configurations using: the 4 interconnects PSI, CSID, CSIP, and CSIPR, the 2 cache organizations: SNUCA L2 and Private L2, and two L2 bank sizes: 1 MB and 256 KB.

#### A. Average Flit Latency

Figures 4 - 7 show the average flit latency (AFL) of the four interconnects normalized to the flit latency for the PSI interconnect. With respect to AFL, order of the interconnects from best to worst is: CSIPR, CSIP, CSID, and PSI. The AFL of CSIP is worse than CSID only in Blackscholes and Swaptions (in private L2 - 1MB). According to [3], Blackscholes has a small working set and low degree of data sharing among threads. This was reflected in our simulation runs of Blackscholes where we noticed the number of messages communicated on the interconnects to be significantly small compared to the other benchmarks. This led to less thrashing of circuits on CSID, where the on-demand circuit configuration policy adapted better to communication characteristics than the pinning policy. This is also confirmed by Blackscholes having

---

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>BENCHMARKS DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parsec Benchmarks</strong></td>
<td></td>
</tr>
<tr>
<td>Benchmark</td>
<td>Input Parameters</td>
</tr>
<tr>
<td>Blackscholes</td>
<td>16384 options.</td>
</tr>
<tr>
<td>Swaptions</td>
<td>32 swaptions, 10K simulations.</td>
</tr>
<tr>
<td>Canneal</td>
<td>16 15000 2000 200000 nets.</td>
</tr>
<tr>
<td>Bodytrack</td>
<td>sequence B 2 4 2 2000 5 0 16.</td>
</tr>
<tr>
<td>Fluidanimate</td>
<td>16 5 in 25K fluid.</td>
</tr>
<tr>
<td><strong>SPLASH-2 Benchmarks</strong></td>
<td></td>
</tr>
<tr>
<td>Benchmark</td>
<td></td>
</tr>
<tr>
<td>Barnes</td>
<td>32K particles.</td>
</tr>
<tr>
<td>Ocean</td>
<td>514x514.</td>
</tr>
<tr>
<td>Radiosity</td>
<td>Large room model.</td>
</tr>
<tr>
<td>Raytrace</td>
<td>Car input.</td>
</tr>
<tr>
<td>LU</td>
<td>-n1024 -p16 -b16.</td>
</tr>
</tbody>
</table>

---
the highest percentage of flits traveling on complete circuits in Fig. 1.

B. Execution Time

Benchmarks execution times normalized to PSI execution times are depicted in Fig. 8 - 11. Different communication latencies due to the different interconnects can affect the execution path of multi-threaded benchmarks causing variations in the number of executed instructions among the different simulated system configurations. For most benchmarks, CSIPR provides better system performance than CSIP, and system performance using either CSIPR or CSIP is better than using PSI or CSID. Fig. 9 and 11 show that Radiosity exhibits some anomalous behavior. We found that in system configurations with Private L2 cache organizations, Radiosity simulations execute many more system instructions when the CSIP and CSIPR interconnects are used than when the PSI and CSID interconnects are used. The result is the increase of the execution time of the corresponding system configurations using CSIP and CSIPR interconnects. This increase in system instructions may be due to synchronization structures (e.g. locks).

C. Circuits Utilization

The charts in Fig. 12 - 15 show the percentage of flits traveling on complete circuits from source to destination.

Percentage of flits traveling on partial circuits are shown in Fig. 16 - 19. These figures indicate that circuit utilization improves with pinned circuits configuration policy and use of partial circuit routing. Applying partial circuit routing in CSIPR puts more pressure on circuits. As a result, a slightly higher percentage of flits travel on complete circuits in CSIP than CSIPR. However, because of the extensive use of partial circuits, the overall performance of CSIPR is better than CSIP.

VIII. CONCLUSION

The significance of on-chip networks will continue to grow as technology advances and many more cores are placed on chip. Our pinned circuit configuration policy, attempts to exploit communication locality, where there are pairs of frequent communication points, to improve communication latency, while coping with changes in communication characteristics through periodic reconfiguration. Combining partial circuit routing technique with pinned circuit configuration further boosts the utilization of circuits, achieving better communication latency. Our simulations demonstrate the potential benefits of these techniques. On average, the pinned circuit configuration policy alone improves flit latency over the on-demand circuit configuration policy by 10%, with partial circuit routing adding another 10% for a total of 20% improvement over the on-demand circuit configuration policy. These improvements in communication latency translate into improved execution times.
Fig. 12. SNUCA L2 (1MB) - % Flits using complete circuits

Fig. 13. Private L2 (1MB) - % Flits using complete circuits

Fig. 14. SNUCA L2 (256KB) - % Flits using complete circuits

Fig. 15. Private L2 (256KB) - % Flits using complete circuits

Fig. 16. SNUCA L2 (1MB) - % Flits using partial circuits

Fig. 17. Private L2 (1MB) - % Flits using partial circuits

REFERENCES


Fig. 18. SNUCA L2 (256KB) - % Flits using partial circuits

Fig. 19. Private L2 (256KB) - % Flits using partial circuits


