J. Parallel Distrib. Comput. 123 (2019) 118–129

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Cost effective routing techniques in 2D mesh NoC using on-chip transmission lines

Dipika Deb [a,*], John Jose [a], Shirshendu Das [b], Hemangee K. Kapoor [a]

[a] Department of CSE, Indian Institute of Technology Guwahati, India
[b] Department of CSE, Indian Institute of Technology Ropar, India

HIGHLIGHTS

- Long distance on-chip communication degrades the performance of NoC in CMP.
- We propose a hybrid NoC architecture with high speed Transmission Line (TL).
- An adaptive routing technique (SBTR) is proposed to send long distance packets with TL.
- An improved hybrid mechanism called e-SBTR is also proposed in this paper.
- The proposed techniques are compared and combined with an existing technique.

ARTICLE INFO

Article history:
Received 5 September 2017
Received in revised form 2 September 2018
Accepted 13 September 2018
Available online xxxx

Keywords:
Hop count reduction
Chip multiprocessor
Hybrid NoC
Express Virtual Channel
Adaptive routing
Packet latency

ABSTRACT

Advancements in CMOS technology led to the increase in number of processing cores on a single chip. Communication between different cores in such multicore systems is facilitated by an underlying interconnect. Due to the limitations of traditional bus-based system Network on Chip (NoC) based interconnect is the most acceptable cost effective framework for inter-core communication. A packet in an NoC travels through a sequence of intermediate routers before arriving at its destination. As the size of NoC scales high, the average number of intermediate routers that a packet traverse also increases. This results in higher packet latency which degrades application performance. In this work, we introduce cost effective adaptive routing techniques that can forward long distance packets through specialized channels made of Transmission Line (TL). These extra TLs introduced in the chip reduce the diameter of the network thereby reducing average packet latency. We propose two novel router architectures; SBTR and e-SBTR that reduce packet latency by reducing the number of intermediate hops. We use PARSEC benchmark and SPEC CPU 2006 benchmark mixes to evaluate the performance of our proposed techniques. SBTR and e-SBTR reduce average packet latency by 7.9% and 25% respectively. Both the techniques also reduce average hop count by 8.13% and 27.6% respectively. We also observe that our proposed technique e-SBTR performs better than the state-of-the-art Express Virtual Channel technique in terms of packet latency and hop count respectively.

1. Introduction

With an increase in number of cores, computation has become faster but inter-core communication encountered with few performance issues. Chip Multiprocessor (CMP) which builds modern computer system demands an efficient on-chip interconnect as a communication backbone. For such systems, a shared point-to-point on-chip network-based interconnect called NoC [7] is used.

In recent multicore designs known as Tiled Chip Multiprocessor (TCMP), the cores are organized as tiles [4,21,13] where each tile consists of an out-of-order super-scalar processor, a private L1 cache and a slice of shared L2 cache (inclusive).[1] Fig. 1 shows a TCMP where each L2 cache slice is called as a bank and each cache block is statically mapped to a bank called as the home-bank of the block. The processor interacts with the L1 cache for instruction fetch, data load and store operations. When a requested word is not found in the L1 cache of a core the request for the block containing the word is forwarded to its L2 home-bank. The miss request packet has to travel through the network to reach the home-bank. The latency of the miss request and reply packet varies depending on how far the home-bank is situated from the requesting core. Each tile in TCMP is connected to NoC through a router and routers are connected using bidirectional RC links. In 2D

* Corresponding author.
E-mail addresses: d.dipika@iitg.ernet.in (D. Deb), johnjose@iitg.ernet.in (J. Jose), shirshendu@iitrpr.ac.in (S. Das), hemangee@iitg.ernet.in (H.K. Kapoor).

[1] Only two level of caches are considered with L2 cache as the last level cache.

https://doi.org/10.1016/j.jpdc.2018.09.009
0743-7315/

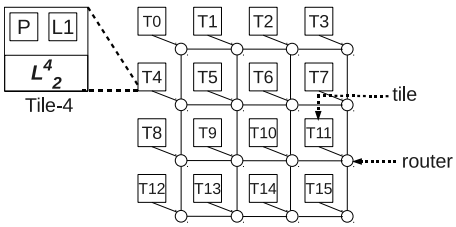


Fig. 1. A Tiled CMP in 2D mesh topology.

mesh NoC, the link delay is predictable but router delay is variable depending on dynamic congestion conditions.

The main contributor to NoC traffic is the cache miss and reply packets. Hence, the miss penalty is proportional to the packet latency. Reducing latency of long distance communication in NoC is an active research problem. Wireless NoC (WNoC) [14,24,25] and Photonic NoC (PNoC) [29,27,5,23] also focus on reducing long distance communication delay and provide high bandwidth and low power consumption. On-chip wireless communication [14] uses millimeter-wave and carbon nanotube antennas for long distance communication. Inductive coupling techniques can also be used to replace antennas. WNoC has many advantages when compared to the wired counterparts. But the wireless channel is lossy and hence reliability is a concern in WNoC [1,15].

Transmission Line (TL) is yet another promising long distance interconnect in NoC that provides high bandwidth and low power consumption. TL has an advantage of faster signaling than RC links [18]. Therefore, it is used as direct links between distant nodes on a chip with single cycle propagation delay [6,12]. TLs are an accepted form of interconnect due to its compatibility with existing CMOS technology [20] but it is wider than RC links and therefore occupies more area. In conventional NoC, the RC interconnect routes short distance packets efficiently but increases latency and power consumption for packets communicating between distant routers. Hence, designing a hybrid NoC using TLs for long distance communication and RC links for short distance communication is the main goal of this paper.

The range of communication footprint and the frequency of inter-core communication change from one phase of an application to another. For larger TCMPs the effect of these problems is more complex to deal with. Hence, better techniques are needed to route the traffic dynamically during runtime. In this work, TL is used as an interconnect for communication between distant routers. A novel adaptive routing mechanism is used to take dynamic decision of when to use the TLs. The major contributions of this paper are as follows:

1. A novel TL based routing called State Based Transmission Line Routing (SBTR) is proposed.
2. An improved hybrid mechanism e-SBTR is proposed by combining SBTR with the concept of the state of the art technique Express Virtual Channel (EVC) [22].
3. An experimental study is done to analyze the impact of the proposed techniques (SBTR, e-SBTR) in comparison with EVC and baseline NoC architecture. Synthetic traffic and real traffic consisting of multithreaded workloads like PARSEC [8] and multiprogrammed workloads of SPEC 2006 [16] benchmark mixes are used for comparison.

Miss penalty of L1 cache reduces if the average packet latency in the network reduces. For PARSEC workload, SBTR and e-SBTR reduce the average packet latency by 7.96% and 23.24%, respectively. Both the techniques also reduce the average hop count by 8.13% and 24.69%, respectively. Reduction of packet latency in SPEC 2006 benchmark mixes for SBTR and e-SBTR is by 7.22% and 25%, respectively while average hop count is reduced by 7.78% and 27.6%, respectively.

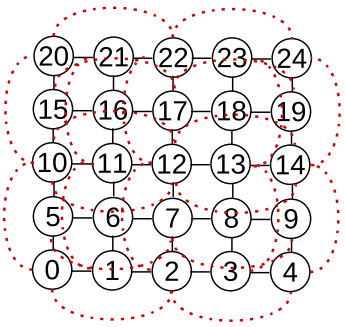


Fig. 2. An example of EVC in 5 × 5 2D mesh.

The rest of the paper is organized as follows. Section 2 presents the related work done and some background works required for this paper. Section 3 provides the motivation behind this work and Section 4 describes our proposed architectures. Section 5 describes about the experimental setup and simulation methods. Section 6 analyzes the performance of the proposed architectures. Section 7 analyzes the hardware aspect of the proposed architectures with respect to baseline and finally the paper is concluded in Section 8.

2. Background and related work

Transmission Line (TL) [10,20,17,3] is a high speed interconnect that acts as a dedicated wire connecting two distant nodes. It reduces the number of hops for long distant packets by bypassing the intermediate routers. TLs are wider than RC wires and hence offer less resistance and propagation delay of signals through them. At larger wire diameters the inductance of the wire increases. The high operating frequency along with the increase in inductance forces us to take wave nature of signaling into consideration. Hence, TLs are fast interconnects that operate at RLC range. Carpenter et al. [10,11] have proposed TL as a feasible NoC interconnect. The authors used TL to design an on-chip bus that connects all the nodes in the system. TLs are also used to reduce the communication latency between L2 cache banks and the cache controller [6,18]. A broadcast interconnect has been designed using TL in [28]. All the above mentioned literature shows that TLs can be considered as a viable design alternative for long distance on-chip communication.

Another major design alternative to reduce packet latency for short distance packets ($hops \leq 3$) is Express Virtual Channel (EVC) [22]. EVCs are not extra physical links like TLs. In EVC, switch allocation across few routers is pre-configured so that the routing and Virtual Channel (VC) allocation delay is avoided in the intermediate routers. Hence, packets passing through EVC have only link delay at every hop. When the number of packets trying to win an EVC is high, it leads to congestion at EVC source. For the existing EVC design, long distance packets will have to hop along multiple intermediate EVCs. But congestion at EVC sources might increase the latency of such packets. Moreover, a packet can use EVC along one dimension only. If there is a change in dimension the packet must hop at an intermediate router. In Fig. 2 the dotted lines represent a 2-hop EVC where the EVC acts as a bypass over two adjacent hops. Other approaches to improve long distance communication are long range physical RC links [26], Optical links [27,2,29] and wireless communication [14,24,25] over conventional NoC.

3. Motivation

In TCMP (shown in Fig. 1) the cache miss request and reply packet travel through the underlying NoC. The miss penalty which impacts the processor stall time at the source core is proportional to the round-trip network latency of these packets. Packet latency in NoC can be calculated using Eq. (1). It mainly consists of Link

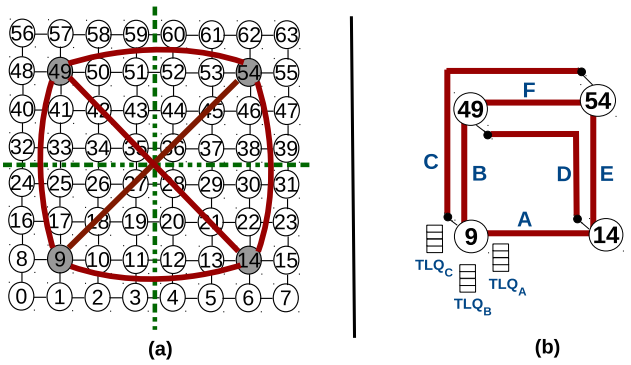


Fig. 3. (a) Logical view of TLs in a 8 × 8 2D mesh. (b) Physical layout of TL on-chip as 2D planar structure with six TLs.

Delay (L_d) and Router Delay (R_d). Link delay is the time taken to traverse the links whereas router delay includes the amount of time a router takes to process an incoming packet and the queuing delay encountered due to port conflicts.

$$Latency = \sum_{i=1}^{H} R_{d_i} + H \times L_d \quad (1)$$

where H = number of hops a packet will take as per the routing algorithm, L_d = link delay, R_{d_i} is the delay in the intermediate router i.

In an un-congested network (zero load), a router requires n cycles (n-staged pipeline router) to process a packet whereas a link takes only one cycle to forward the packet to its immediate downstream router. But in a congested network, a packet might be queued for more than n cycles due to port conflicts or lack of credit in the downstream router. As the number of cores increases, average packet hop also increases. With a traditional 2D mesh topology, it is not possible to reduce the hop count using conventional routing algorithms. Conventional NoC uses RC interconnects as metal links connecting two routers. RC links suffer from higher latency as well as power consumption for long distance communication. Therefore, long distance packet suffers from multihop drop. TL provides high bandwidth and consumes less power but the area overhead of TL is larger than that of RC links. So only few links can be placed on the chip. Moreover, TLs are not the best choice to connect between two adjacent routers due to the fact that the TL delay is much lower than the router pipeline delay. Therefore, we are motivated to analyze a hybrid NoC with TL as an alternative interconnect for long distance packets and RC links for short distance packets. We propose two hybrid NoC architectures to reduce the hop count. In the first approach (SBTR) TLs are added over the conventional NoC. The second approach (e-SBTR) incorporates the merits of EVC design to our first approach.

4. Proposed techniques

We use a 64-core TCMP organized as 8 × 8 2D mesh for the baseline architecture. For our proposed work the network is divided into four quadrants, each of size 4 × 4. There are two types of routers: Normal Router and TL Router (TLR). We keep the center router of each quadrant as TLR and the position of TLRs are pre-fixed and known to all the routers in the network. TLRs are connected to high-speed TLs as shown in Fig. 3(a) and each TLR has three TLs connected to it. The two ends of a non-diagonal TL are at 5 hops distance away. The experimental justification for taking five hop distance is discussed in Section 7. As shown in the figure, one of the advantage of such TL design is that this arrangement is scalable even for larger NoC. Similar to an 8 × 8 2D mesh, a $2^n \times 2^n$, ($n \geq 3$) 2D mesh can also be divided into four quadrants where each quadrant is of size $2^{(n-1)} \times 2^{(n-1)}$. Therefore, TLs can be laid out in an identical fashion in each quadrant as in 8 × 8 mesh. The hybrid NoC has a 2D planar structure which is maintained by avoiding crossing of TLs as shown in Fig. 3(b). Each TL consists of two wires to make it bidirectional and is a 2-tuple $\langle p, q \rangle$. Here, p and q are the two TLRs to which the TL is connected, i.e. there is one TL from p to q and the other from q to p. Apart from normal RC links where communication between a pair of routers is through multiple parallel wires (equal to the flit size), here there is a single link through which a flit is sent serially at high bandwidth rate. Fig. 3 shows six bi-directional TLs: $A\langle 9, 14\rangle$, $B\langle 9, 49\rangle$, $C\langle 9, 54\rangle$, $D\langle 14, 49\rangle$, $E\langle 14, 54\rangle$ and $F\langle 49, 54\rangle$.

Any packet that flows through TL is called as *TL-candidate*. Algorithm 1 helps in identifying *TL-candidates* in the network. A packet is divided into multiple flits: head flit, body flit and tail flit. Apart from the normal fields in the head flit, a boolean field *isTL* and a 6-bit field *t_dst* has been added. *isTL* is set to true for a valid *TL-candidate*. When *isTL* is true, the flit has to be forwarded to the nearest TLR whose address is stored in *t_dst*. Algorithm 1 explains the procedure to calculate *t_dst* and *isTL* for each packet operation. The *t_dst* field stored in the flit helps in identifying the TLR during TL routing.

Function of normal routers:

Whenever a packet is generated, the source router decides whether to use TL or not, using Algorithm 1. The algorithm takes the head flit (f) as input and checks whether it is a *TL-candidate* or not. The algorithm uses two functions: (a) *get_distance*(s, d) and (b) *get_nearest*(s, $TL\langle p, q\rangle$). The first function returns the number of cycles required to transmit a flit from s to d using XY routing in zero load condition. In such ideal condition, the router takes 2 cycles and the link takes 1 cycle to transmit a packet from an upstream router to a downstream router. The second function returns the id of the TLR (p or q) which is nearest to s.

Algorithm 1: Deciding TL-candidates.

```
1  isTL_Candidate(f) begin
       Input: f ⟵ the flit.
2      dist = 0: integer value.
3      list⟨TL⟩ TLS: list of all TLs in NoC.
4      dist_XY = get_distance(f.src, f.dst)
5      dist_TL = dist_XY + 1 //initialising larger than dist_XY
6      tl_index = −1 //initialising a -1.
7      for i ← 1 to TLS.size do
8          s′=get_nearset(f.src,TLS[i])        // nearest TLR from
           source.
9          d′=get_nearset(f.dst,TLS[i])        // nearest TLR from
           destination.
10         if s′ ≠ d′ then
11             dist= get_distance(f.scr, s′) // source to start of
               TL.
12             dist+ = get_distance(d′, f.dest)        // TL end to
               destination.
13             dist+ = 1                               // TL delay.
14             if dist_TL > dist then
15                 dist_TL = dist
16                 tl_index = i
17     if dist_TL < dist_XY then
18         s′ =get_nearest(f.src, TLS[tl_index]);
19         f.t_dst = s′
20         isTL = true
21     else
22         f.t_dst = −1
```

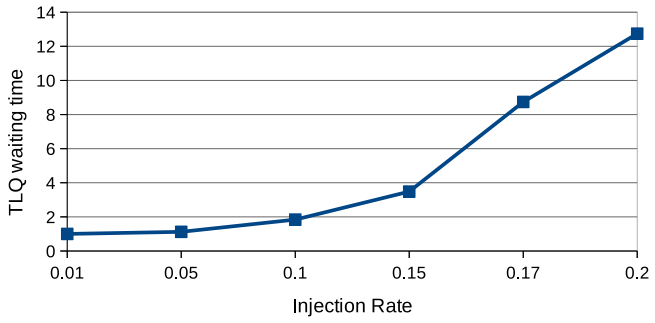


Fig. 4. Network load vs TLQ waiting time (in cycles) in uniform traffic.

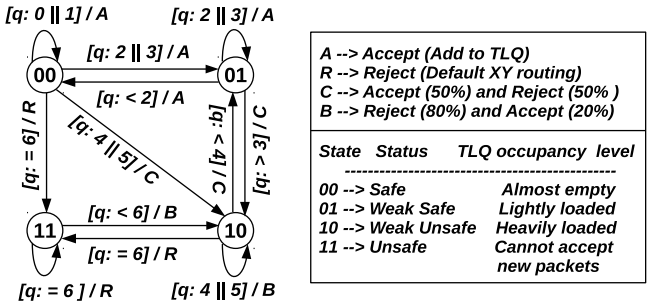


Fig. 5. Finite state machine for SBTR. Here, the term [q: T_1 || T_2] means that the TLQ size is either T_1 or T_2.

Line 4, in Algorithm 1 uses *get_distance*($f.src, f.dst$) and stores the value in $dist_{XY}$. Here $f.src$ and $f.dst$ are the source and destination of the flit f, respectively. Lines 7 to 16 calculate the minimum number of cycles ($dist_{TL}$) required to transmit the flit (from $f.src$ to $f.dst$) through TL. For each TL, the end nearest to $f.src$ is considered as s' (line 8) and the end nearest to $f.dst$ is considered as d' (line 9). The cycles required to communicate through a TL are calculated in lines 11 to 13 as $dist = get\_distance(f.src, s') + get\_distance(d', f.dst) + 1$. Here, one cycle is required for the flit to propagate through the TL. At the end of the loop, the $dist_{TL}$ stores the minimum cycles required to transmit f through TL and the field *tl_index* holds the *TL id* through which the flit should flow. If the value of $dist_{TL}$ is less than $dist_{XY}$, the flit is considered as *TL-candidate* and *isTL* is set as true. The nearest TLR of TLS[*tl_index*] is considered as $f.t\_dst$.

Example. Consider a packet $P(0, 63)$ in Fig. 3. The source router (0) executes Algorithm 1 and calculates $dist_{XY}$ as 42 cycles and $dist_{TL}$ as 15 cycles if the packet uses TL C⟨9, 54⟩. So the head flit set its *isTL* field as 1 and *t_dst* as 9; all the flits will be forwarded to TLR 9 to flow through the TL C⟨9, 54⟩. Note that in this technique only the TLR information is appended and not the complete path in the packet. This logic is incorporated in all the routers because all the cores in the network may generate cache miss packets.

Functions of TL router:

Each of the TLR has three *TLQs* one for each TL as shown in Fig. 3(b). If a *TL-candidate* after reaching the TLR finds the link busy then it waits in the corresponding TLQ. When the link becomes free, flits from TLQ flow through the link. Experimentally we found that as network load increases, number of flits waiting in TLQ increases as shown in Fig. 4. As most of the flits choose expressways to travel long distances, the size of TLQs may grow rapidly. If no bound is implemented on the queue size, it may grow indefinitely and increase the waiting time of *TL-candidates* in the TLQ. Hence, we propose SBTR to handle high queuing delay. SBTR dynamically controls whether an incoming flit coming to TLR can be allowed to enter the TLQ or not. Apart from normal routing operations, TLR has two additional tasks to perform when a *TL-candidate* arrives.

1. Inserting flits into TLQ (taken care by SBTR technique) and
2. Transmitting flits through TL (TL routing).

4.1. State based transmission line routing (SBTR)

The SBTR uses a finite state machine (FSM) as shown in Fig. 5. The FSM controls the routing of *TL-candidates* efficiently by restricting TLQ size to six. It consists of four states which indicate the TLQ occupancy level. Each edge of the FSM is denoted by X/Y where X is the probable TLQ size and Y is the action to be taken on a newly arriving flit. Initially, the state of all the TLQs is *00* which indicates that the TLQ size is at most one. Based on the occupant level of the queue there can be four possible actions:

- *A*: Accept new flits into TLQ.
- *R*: Reject new flits from entering into TLQ.
- *C*: Choice. TLQ will accept flit with a probability of 0.5.
- *B*: Choice. TLQ will accept flit with a probability of 0.2.

When flits from TLQ are forwarded to TL, the queue size decreases and accordingly the state of the FSM changes. The transition from one state to another can be triggered by two events: (a) arrival of a *TL-candidate* in TLR or (b) movement of a flit from TLQ through TL. The state transition table is shown in Table 1. Few transitions in the FSM are prohibited. The prohibition is done so that when the TLQ size sharply decreases, the state of the TL should not become safe at once. The decrease in the TLQ size must propagate slowly in the FSM whereas the increase in the size of TLQ must propagate fast in the FSM. This mechanism is called as *"Fast Increase slow Decrease"*.

Congestion at TLR is controlled using a back pressure mechanism. When a TLQ reaches the unsafe state (state 11) TLR applies back-pressure about the status of TLQ to its 2-hop neighbors. The back-pressure information is sent as control packets using the conventional credit links used in NoC. When a back pressure information reaches the 2-hop neighbors of TL router, they stop sending packets towards TL router for m cycles. Here, m corresponds to the window size. Fig. 7(a) shows the variation in average packet latency on changing the window size. From the figure it can be seen that preventing packets from the neighbor to the TL router helps in stabilizing the TLQ. A higher value of window size may result in withdrawing too many long distance packets in using TL while a lower value of window size may result in an unstable

Table 1
State transition table for SBTR.

States	Queue size	Next state	Action
00	0 or 1	00	A
00	2 or 3	01	A
00	4 or 5	10	C
00	6	11	R
01	<2	00	A
01	2 or 3	01	A
01	>3	10	C
10	<4	01	C
10	4 or 5	10	B
10	6	11	R
11	<6	10	B
11	6	11	R

Table 2
Latency in cycles and path proposed for different proposed techniques.

P(Src, Dest)	XY routing (default)	SBTR (2-hop EVC)	EVC (SBTR with 2-hop EVC)	e-SBTR
P_1 (0 , 63)	42 cycles [0-1-2-3-4-5-6-7-15-23-31-39-47-55-63]	15 cycles [0-1-9-54-55-63]	30 cycles [0-2-4-6-7-23-39-55-63]	15 cycles [0-1-9-54-55-63]
P_2 (1 , 60)	30 cycles [1-2-3-4-12-20-28-36-44-52-60]	15 cycles [1-9-54-53-52-60]	22 cycles [1-2-4-20-36-52-60]	13 cycles [1-9-54-52-60]
P_3 (33 , 22)	21 cycles [33-34-35-36-37-38-30-22]	12 cycles [33-41-49-14-22]	15 cycles [33-34-36-38-22]	10 cycles [33-49-14-22]
P_4 (38 , 41)	18 cycles [38-37-36-35-34-33-41]	12 cycles [38-46-54-49-41]	14 cycles [38-36-34-33-41]	10 cycles [38-54-49-41]

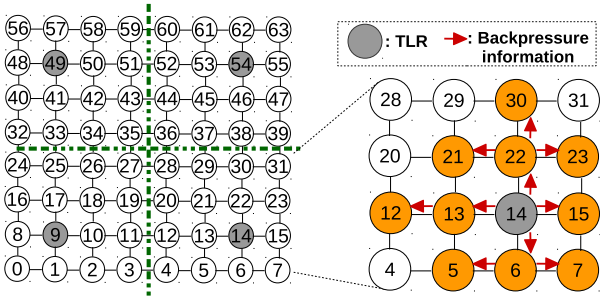


Fig. 6. Back pressure mechanism in TLR.

TLQ. Hence, the window size should be kept at an optimal size for better network performance. Experimentally we found the optimal window size to be 4 as shown in the figure.

As shown in Fig. 6, in a quadrant of size 4 × 4, 2-hop distance from the center router almost covers the quadrant. The neighboring routers after receiving the unsafe information set up a timer window for 4 cycles. During this timer window the neighboring routers will not forward any *TL-candidates* to TLR and the *isTL* flag is reset thereby permitting them to flow using normal XY routing. Such flits are called as rejected flits. In the meantime, flits already stored in TLQ make progress through the TL and the queue size reduces. After the timer expires the *TL-candidates* can be sent towards the TLR again. Fig. 7(b) shows the distribution of flits generated by PARSEC workloads in 8 × 8 mesh network using SBTR. It can be observed that the flits flowing through TLs help in reducing the latency of long distance packets. However reducing the latency of rejected and normal flits may further improve the network performance and this is the main motivation behind e-SBTR.

4.2. e-SBTR

Normal flits can gain an advantage by using EVC while long distance flits use TLs. Therefore we propose e-SBTR, a combination of SBTR and EVC. Normal flits may require more time to travel through the vicinity of TLRs due to port conflicts with *TL-candidates*. Consider a *TL-candidate* $f_1(33, 55)$ and a normal flit $f_2(25, 57)$ in Fig. 3. Let us assume that f_1 uses TL $F\langle 49, 54\rangle$ and f_2 uses XY routing. To reach the TLR 49, f_1 follows the same path as that of f_2. Hence, f_2 will experience some additional delay on its way which would not have been the case if f_1 uses default XY routing [33-34-35-36-37-38-39-47-55]. Therefore, if f_2 uses an EVC that bypasses TLR 49 then neither f_1 nor f_2 will experience mutual port/path conflicts. Packets that get rejected at TLRs would have used default XY routing in SBTR but in e-SBTR, it can also flow through EVC. Thus the latency of all types of flits (*TL-candidate*, rejected and normal flits) in the network decreases.

An additional advantage of e-SBTR is that if *TL-candidates* upon arriving at TLR identify a high TLQ size on the shortest path TL, then it can consider to wait in other TLQs based on the queue size. Consider a flit $f_3(0, 63)$ in Fig. 8 whose shortest TL is $C\langle 9, 54\rangle$. When the flit reaches the foot of TLR (9), two other TLQ choices are also available i.e. TLQ for $TL\{A\langle 9, 14\rangle + E\langle 14, 54\rangle\}$ or $TL\{B\langle 9, 49\rangle + F\langle 49, 54\rangle\}$. In e-SBTR, *TL-candidates* after reaching the TLR can dynamically decide to wait on the TLQ with least queue size. Fig. 9 shows the flowchart for e-SBTR. Table 2 gives a comparison study of four flits P_1, P_2, P_3, P_4 and their paths in SBTR, EVC and e-SBTR techniques.[2]

4.3. Router micro-architectures

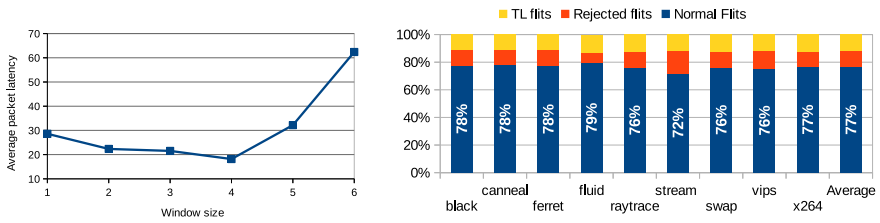
As described in Section 4, our proposed techniques have two types of routers: normal router and TLR. A normal router has five input/output ports one for each of the four neighbors (north, south, east, west) and one port for its local core. Fig. 10(a) shows a normal router where every incoming port is connected to a virtual channel. The route computation module calculates the outgoing port considering the destination of the flit. VC allocation stage reserves buffer for a packet in the downstream router and switch (SW) allocation stage selects an input port of the crossbar to reach the desired output port in the router. After SW allocation process, in the next cycle the flit traverses the crossbar and travels through the link to reach the downstream router.

Fig. 10(b) shows the internal structure of TLR which contains an additional component, TL Candidate checker (TLCc) to check if a flit is a *TL-candidate* or not. If a flit is *TL-candidate* then the flit skips the route computation stage and enters into the VC stage. The FSM as shown in Fig. 5 is implemented in TL allocator and based on the TLQ occupancy level, the flit enters into the TLQ. In e-SBTR the TLRs are never an EVC source/sink. Hence an EVC latch is added at the input buffer to bypass the route computation stage and VC units. This EVC latch will reduce the complexity of the TLR. Therefore, EVC source, sink, and bypass nodes are similar to that of [22]. Fig. 10(c) shows a TLR in e-SBTR.

4.4. Deadlock and livelock

In TL routing, a *TL-candidate* passes through a maximum of three phases: *pre-TL* phase, *TL-link* phase and *post-TL* phase. In pre-TL phase, the flit reaches TLR using deterministic XY routing. In TL-link phase the flit passes through TL which is equivalent to a flit flowing one hop distance in a normal routing scenario. In the post-TL phase, the flit passes from the other end of the TL to the destination. The first phase and third phase use deterministic XY routing which is a deadlock free routing mechanism. In

[2] A packet may have multiple flits. The table shows the time required to transmit a flit of the packet.



(a) Window size vs Avg. packet latency. (b) Flit distribution in SBTR.

Fig. 7. Congestion control mechanism and flit distribution in SBTR.

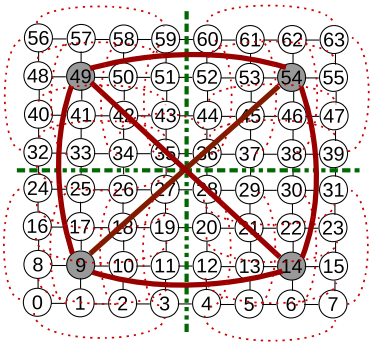


Fig. 8. e-SBTR: Combination of SBTR with EVC.

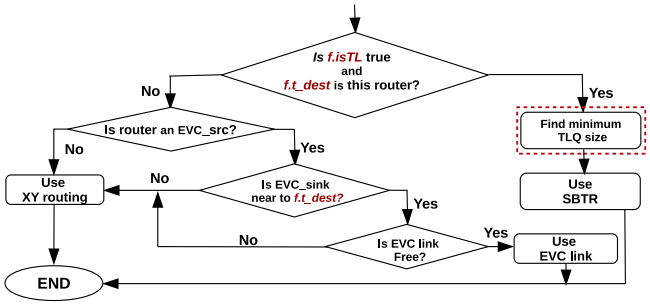


Fig. 9. Flow chart for e-SBTR.

TL-link phase the flit has only one productive output queue where it gets stored and subsequently moves through the TL. Since the flit cannot hold or occupy any other TLQ therefore, the condition of "circular wait" is completely avoided. This makes sure that deadlock is avoided in our proposed TL routing.

Another scenario that can cause deadlock is by some rejected *TL-candidates*. A rejected *TL-candidate* will use XY routing to reach its destination from the point of rejection (TLR or its neighbors). In this case, the packet will have two phases in its travel. Phase 1: source router to the point of rejection using XY routing and phase 2: point of rejection to destination. At the point of rejection, there can be some cases where a packet encounters a transition from YX to XY direction leading to a deadlock. To avoid such scenario, a packet is re-injected into the local port virtual channel. Appropriate change is made on the packet header in source and destination fields to reflect this re-injection and necessary flow control. Since the packet is re-injected, it is a movement from local port VC to output direction. So even though the packet movement may look like a YX transition, it is physically implemented by forwarding through local port VC. Hence, it is not a turn violation as per XY routing policy. TLRs and its 2-hop neighbor routers are designed with this minor modification to ensure a smooth transition from phase 1 to phase 2.

Our TL routing is also free from livelock scenarios. A source router according to Algorithm 1 sets its *isTL* field for *TL-candidates*. The *isTL* field is reset either after traversing TL or during TL rejection (congestion control). Note that only a source router can set the *isTL* field. So after a packet is untagged, the flit can never become a *TL-candidate* again. Hence, a *TL-candidate* can travel through TL just once. Moreover, all flit uses the deterministic XY routing either to reach TLR or to reach destination thereby making it livelock free.

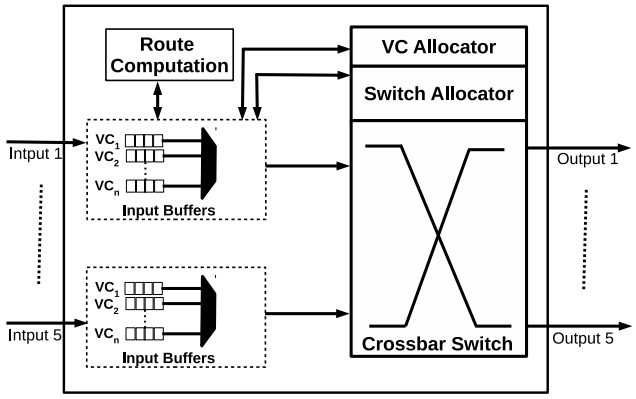
Table 3
Simulation parameters.

Gem5 configuration.	
Processor	64, x86 cores with out-of-order execution
L1 cache per core	64KB (private), 4-way associative, 32B block size
L2 cache per core	512KB (shared), 16-way associative, 64B block size
Coherence protocol	MESI CMP protocol
Booksim configuration.	
Topology	8 × 8 2D Mesh
Routing algorithm (default)	XY routing
No. of ports per router	5
No. of VCs per port	4
VC buffer size:	4
Flit size/channel width	128-bit
Packet size	1 flit request, 5(1 head + 4 body) flit reply
Link length: RC wire	5 mm
Link length: TL wire	25 mm

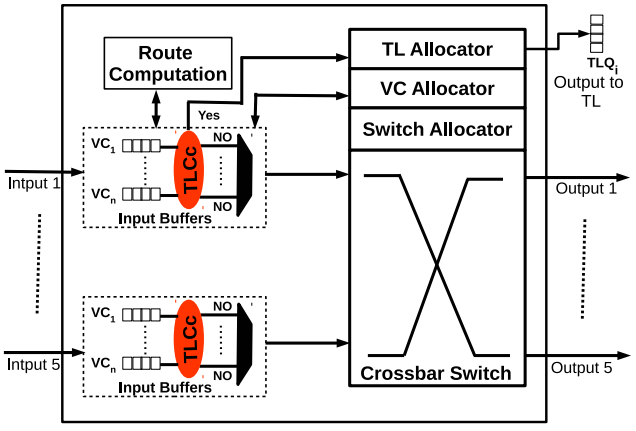
5. Experimental setup and workloads

We use Booksim 2.0 [19], a cycle accurate NoC simulator which can simulate various aspects of NoC. The network is organized as 8 × 8 2D mesh topology with XY as default routing algorithm. We model the network with virtual channel flow control mechanism, wormhole XY routing and finite input buffering. To implement the proposed technique, the routing algorithm and router micro-architecture have been completely modified. Transceiver circuit is incorporated at every TLR. In the baseline architecture, router encounters 2 cycles delay and link requires 1 cycle delay to process a flit. We use uniform pattern for synthetic traffic because it generates a good mix of both short as well as long distance packets. The normal TLs (A, B, E and F) take one cycle and diagonal TLs (C and D) take two cycles each to transfer a flit. We evaluate our proposed techniques using PARSEC benchmark suite, SPEC 2006 benchmark mixes and the synthetic traffic generators. To run SPEC and PARSEC benchmarks we model the processor and cache architecture in Gem5 [9] simulator. All the workloads are executed on 64-core TCMP to generate the traces of L1 misses which are responsible for the network traffic. We use the traces as input to Booksim 2.0 for analyzing the network performance. The simulation parameters are summarized in Table 3.

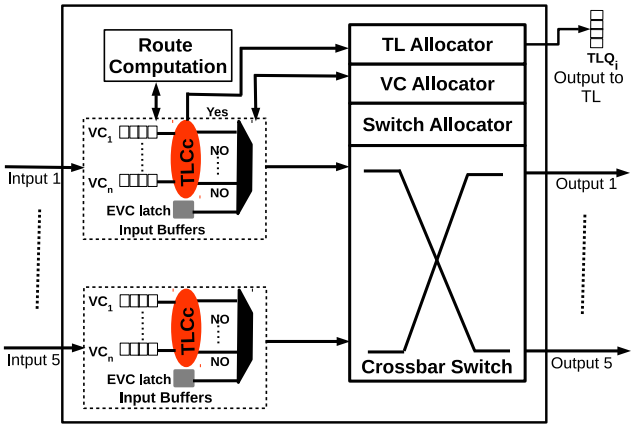
SPEC 2006 benchmarks used for our simulations are summarized in Table 4. We classify the benchmarks into three categories



(a) Normal Router.



(b) TL Router (SBTR).



(c) TL Router (e-SBTR).

Fig. 10. Router architectures of normal routers and TL routers.

i.e. low, high and medium based on their Misses Per Kilo Instruction(MPKI). We categorize these workloads into 6 mixes (M1 to M6) based on the percentage of network injection intensity. Each PARSEC workload consists of 64 multithreaded instances of the given application. Table 5 categorizes the PARSEC benchmarks based on the working set size of L2 cache.

6. Performance analysis

We compare our proposed techniques (SBTR, e-SBTR) with 2-hop EVC and the baseline architecture.

Table 4
Details of SPEC CPU 2006 workload constituents.

Low MPKI (MPKI: 0–5)	*calculix, gromacs, h264ref, gobmk;*
Medium MPKI (MPKI: 5–25)	*bzip2, gcc, gobmk;*
High MPKI (≥25)	*gobmk, hmmer.nph3, lbm, mcf , leslie3d;*

Mix of benchmarks	M1	M2	M3	M4	M5	M6
Percentage of low MPKI	100	0	0	50	0	50
Percentage of medium MPKI	0	100	0	0	50	50
Percentage of high MPKI	0	0	100	50	50	0

Table 5
PARSEC benchmark classification.

Low intensity:	*blackscholes (black), raytrace, swaptions(swap).*
Medium intensity:	*fluidanimate (fluid), streamcluster (stream), vips.*
High intensity:	*canneal, ferret, x264.*

Table 6
Average hop count reduction (%) over baseline in PARSEC benchmark.

	black	canneal	ferret	fluid	raytrace	stream	swap	vips	x264	Avg
SBTR	6.52	5.08	7.00	6.80	8.28	11.11	11.01	8.65	8.77	8.13
EVC	15.05	4.99	7.18	16.43	17.86	18.91	19.19	16.65	8.5	13.86
e-SBTR	22.72	21.90	23.53	24.36	25.66	27.11	27.15	24.86	24.88	24.69

6.1. Effect on average number of hops in PARSEC benchmarks

The main motive of this paper is to cut down the latency of long distance packets by reducing the average number of hops (H) as described in Eq. (1). Fig. 11 shows the normalized average hops used by the packets to reach their destination. Table 6 shows the reduction of hop count in SBTR, EVC and e-SBTR over baseline for different benchmarks. It can be observed that for all PARSEC benchmarks the average number of hops traversed by a packet is reduced in the proposed techniques over baseline as well as EVC. As compared to the baseline architecture SBTR, EVC and e-SBTR reduce hop count by 8.13%, 13.86% and 24.7%, respectively. SBTR performs better than baseline due to the usage of TLs. But only long distance packet gets the advantage of using TLs. e-SBTR has lower hop count than SBTR as it uses TLs for long distance packets and EVC for packets above two hops. Compared to EVC, e-SBTR reduces average hops by 10.83 percentage points.

6.2. Effect on packet latency in PARSEC benchmarks

Packet latency is the time taken by a packet to reach its destination from the time it is injected into the network. As the number of hops reduces, the latency of a packet also reduces. In this section, the analysis of average latency is shown separately for each category of PARSEC benchmarks.

Case I : Low and medium network intensive benchmarks

In low and medium network-intensive benchmarks, the applications are more computation intensive rather than communication intensive. So the amount of packets injected in the network is less. Fig. 12(a) shows the average packet latency normalized over baseline in low intensive benchmarks. In EVC, the average packet latency reduces by 11.38% while SBTR and e-SBTR achieve a reduction of 8.3% and 24%, respectively. In Fig. 12(b), EVC shows a reduction of average packet latency by 11.36% while SBTR and e-SBTR achieve a reduction of 8.46% and 24.2%, respectively for medium intensive benchmarks.

Increase in TLQ occupancy affects the performance of SBTR. The performance gap between SBTR and EVC in this scenario is very low. The reason behind this behavior is the small amount of network traffic which can be easily handled by both the techniques.

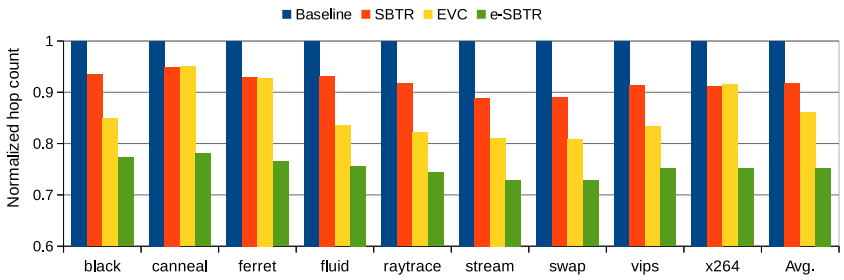


Fig. 11. Normalized average hop count in PARSEC benchmark.

Table 7
Average packet latency reduction (%) over baseline in PARSEC benchmarks.

	Low intensive			Medium intensive			High intensive			
	black	raytrace	swap	fluid	stream	vips	canneal	ferret	x264	Avg
SBTR	6.35	7.94	10.59	6.53	10.64	8.22	4.94	7.73	8.37	7.96
EVC	9.45	11.46	13.25	10.26	13.07	10.76	3.39	8.94	8.79	9.93
e-SBTR	21.672	24.41	25.93	23.13	25.84	23.64	19.34	21.88	23.38	23.24

When comparing EVC with e-SBTR it can be observed that e-SBTR performs better. The behavior of e-SBTR shows that both short as well as long distance packets are benefited altogether unlike in SBTR and EVC acting alone.

Case II : Heavy network intensive benchmarks

Heavy benchmarks are communication intensive. Fig. 12(c) shows the latency analysis of heavy benchmarks. In SBTR due to higher network load, congestion around TLR increases. Hence, in this case the time required for packet movement around the vicinity of TLR increases. Moreover, the EVC link also gets congested and therefore the average packet latency reduction in SBTR and EVC is almost same (7%). e-SBTR on the other hand reduces the average packet latency over baseline by 21.54% and over EVC by 15.58%. The motive behind separate analysis of benchmarks to case I & case II is to show the dynamic behavior of e-SBTR over SBTR and EVC at higher network traffic.

Table 7 summarizes the latency reduction over baseline in all PARSEC benchmarks. It can be observed that EVC performs slightly better than SBTR. But, e-SBTR performs significantly better than that of EVC. On an average, e-SBTR reduces packet latency by 13.31 percentage points as compared to EVC. At higher network load, TLQ occupancy rate will be high and congestion around the vicinity of TLR also increases. This in turn increases the time taken by packets to flow through the network. In e-SBTR, the port conflicts between packets are reduced due to which the network can handle more load. e-SBTR handles network traffic dynamically by giving choices for short distance packets to use either XY routing or EVC and long distance packets to take up a TLQ that has lesser size. Thus congestion around TLRs of e-SBTR is reduced dynamically resulting in lower latency values.

6.3. Effect on packet latency and average hop count in SPEC 2006 workload

The SPEC 2006 benchmarks are multiprogrammed and hence each core runs independent SPEC applications. But since the underlying CMP is tile-based (refer Fig. 1) where the L2 cache is shared, a core may need to communicate with other tiles for accessing a block from its home-bank. Hence, even though the benchmarks are multiprogrammed there is tile to tile communication in the network. This section describes the performance of our proposed techniques using SPEC CPU 2006 benchmark mixes. The mixes show different behavior as the combination of MPKI value (low, medium and high) varies.

Table 8
Average hop count reduction (%) over baseline in SPEC 2006 benchmark mixes.

Benchmark	M1	M2	M3	M4	M5	M6	AVG
SBTR	9.10	6.71	9.13	6.69	6.80	8.30	7.78
EVC	11.05	9.26	10.00	11.50	8.83	8.49	9.85
e-SBTR	33.73	25.37	27.86	25.73	25.87	27.16	27.60

Table 9
Average packet latency reduction (%) over baseline in SPEC 2006 benchmark mixes.

Benchmark	M1	M2	M3	M4	M5	M6	AVG
SBTR	10.75	2.93	6.87	7.47	7.00	8.42	7.22
EVC	14.19	3.30	10.20	9.46	7.08	10.65	9.09
e-SBTR	26.02	20.13	22.12	23.86	23.41	25.36	25.01

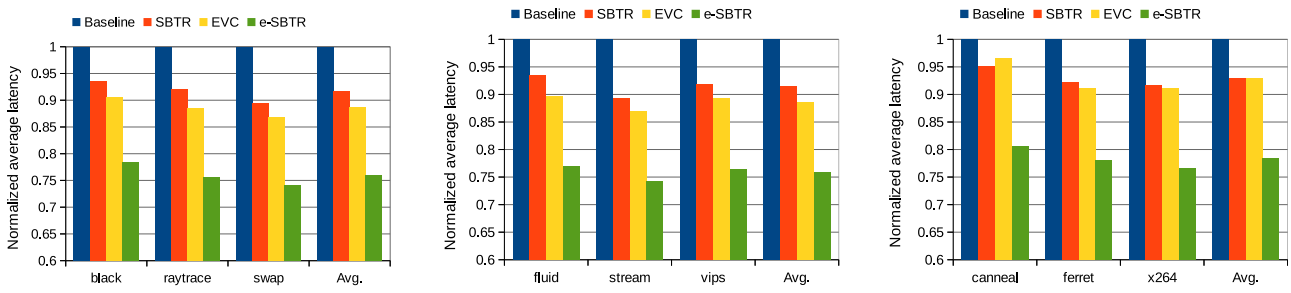
Table 8 shows the reduction of hop count in proposed techniques over baseline. Fig. 13(a) shows that the average hop count over baseline for SBTR, EVC and e-SBTR reduces by 7.78%, 9.85% and 27.6%, respectively. It can be observed that in e-SBTR hop reduction is significantly better than EVC. Fig. 13(b) shows that SBTR, EVC and e-SBTR reduce average packet latency by 7.22%, 9.09% and 25%, respectively (see Table 9). Since the average hop count is minimum for e-SBTR, average packet latency is also minimum for e-SBTR.

6.4. Effect on packet latency in synthetic traffic

In average packet latency versus injection-rate graph, lower latency and wider saturation point indicate better network performance. Increase in injection rate (flit/cycle) increases the number of packets in the network. Hence, packet requires more cycle to reach their destination. Further increase in injection rate congests the network at which the network reaches saturation point. Beyond saturation point, packets can hardly make any progress and the latency of packets increases exponentially. Fig. 14 shows the performance of various techniques in synthetic traffic. We can observe that e-SBTR achieves up to 25% latency reduction as compared to the baseline. Initially at low injection rate, e-SBTR shows very low average packet latency curve. At higher injection rate, e-SBTR curve is much lower than that of the baseline, SBTR and EVC. Moreover the saturation point of e-SBTR is higher than that of other techniques. This indicates that e-SBTR is able to handle more network load efficiently.

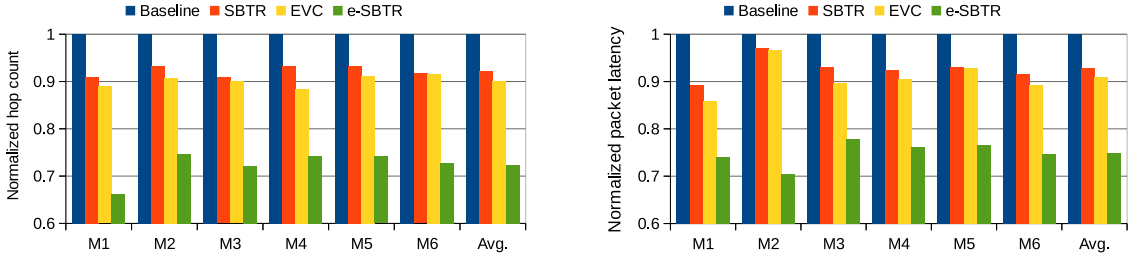
6.5. Effect on packet latency and hop count due to increase in packet size

Fig. 15 shows the performance of e-SBTR when packet size increases from 2 flits to 8 flits. From the normalized packet latency graph it can be seen that in e-SBTR the network performance



(a) Low intensity. (b) Medium intensity. (c) Heavy intensity.

Fig. 12. Average packet latency over baseline using PARSEC benchmarks.



(a) Normalized hop count. (b) Normalized packet latency.

Fig. 13. Performance of SPEC 2006 benchmark in various techniques.

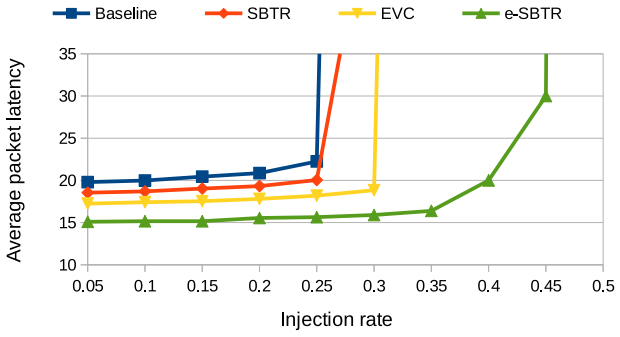


Fig. 14. Average packet latency in uniform traffic.

initially increases with increase in packet size. In 2-flit packet, the combination of normal and *TL-candidates* in the network is less. Hence the network resources like routers, RC links and TLs are free for most of the time. Network traffic increases as packet size increases from 4-flit to 8-flit. Increase in network traffic is handled well by the underlying network which is better explained by the reduction in hop count. The normalized hop count graph shows that among all the packet sizes, the hop count of packets with 8-flit scenario is less. This indicates that e-SBTR helps in decreasing the average number of hops that a packet uses to travel in the network. Further increase in packet size congests the network and hence both normal and *TL-candidates* require more time to make progress due to higher occupancy rate of TLQs.

6.6. Flit distribution in e-SBTR for PARSEC and SPEC benchmark

In addition to the above advantages, e-SBTR reduces the waiting time of *TL-candidates* at the TLQ. This is done by providing all the TLs at TLR as their potential choices. It reduces the queuing delay of packets. Figs. 16 and 17 show flit distribution in SBTR and e-SBTR for PARSEC and SPEC 2006 benchmarks. In both the figures, the normal flit includes both normal and rejected flits. From the figures it can be observed that in e-SBTR, the average amount of flits passing through TL is around 30% for PARSEC and 34% for SPEC benchmark. This is almost twice than that of SBTR and EVC. Thus, e-SBTR not only reduces packet latency and average hop count but also reduces the number of *TL-candidates* that gets rejected at TLR. So by adding choices of other TLs to a *TL-candidate* result in better load handling in the network.

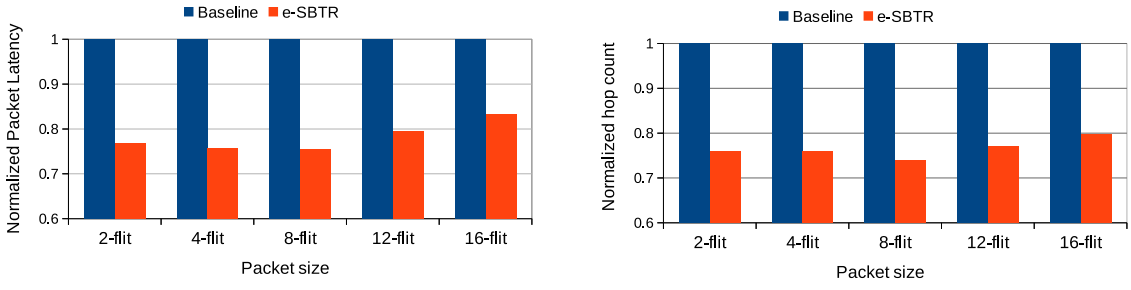
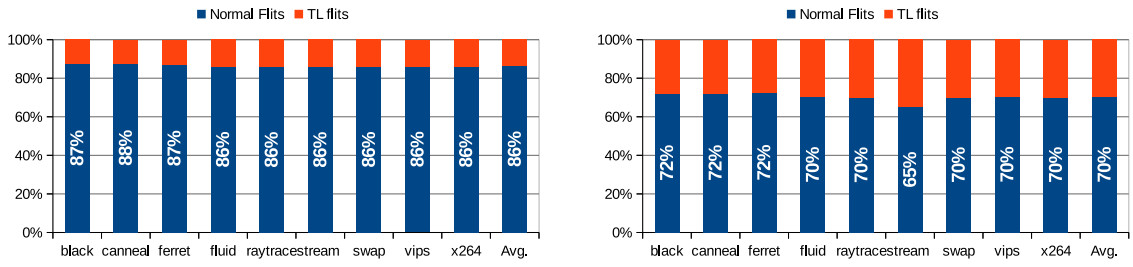
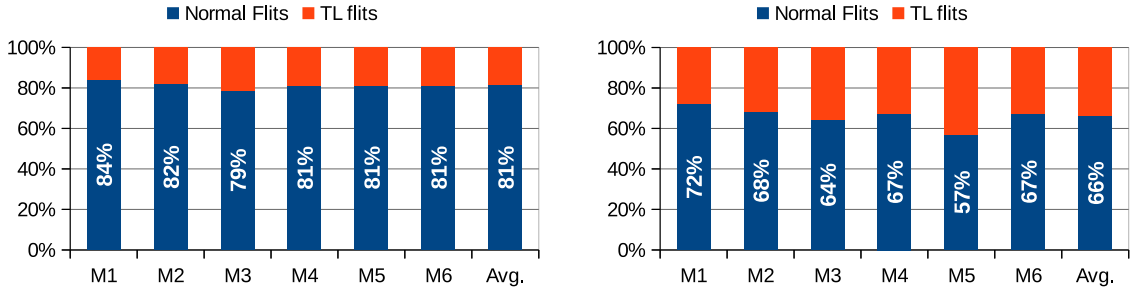


Fig. 15. Normalized performance of e-SBTR for various packet sizes with packet latency on the left and hop count on the right.



(a) SBTR. (b) e-SBTR.

Fig. 16. Fractional distribution of flits in PARSEC benchmark.



(a) SBTR. (b) e-SBTR.

Fig. 17. Fractional distribution of flits in SPEC 2006 benchmark.

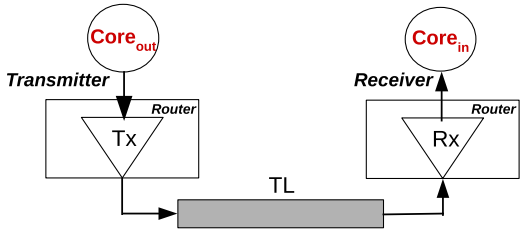


Fig. 18. Transceiver circuits for communication through a TL.

Table 10
Normalized area and critical path latency overhead over baseline router.

Technique	Area	Latency
SBTR	1.005	1.045
EVC	1.02	1.002
e-SBTR	1.025	1.055

7. Hardware analysis

We model the proposed TLR that consists of de/serializer and TLQ in Verilog to analyze the hardware overheads. The synthesized TLR design is implemented and verified on Diligent Zedboard Zynq-7000 evaluation and development kit using Xilinx Vivado 2016.2. The Verilog design was synthesized with 90 nm CMOS library using Synopsys Design Compiler to obtain router area and critical path latency. The normalized results with respect to the baseline are shown in Table 10.

7.1. Area overhead and power consumption for TL

The signal across a TL is transferred as an electromagnetic wave with the speed of light. A transmitter and receiver help in transmitting a packet serially across TL as shown in Fig. 18. The transmitter and the receiver serialize and deserialize the flit. At 90 nm technology, a transceiver circuit requires total area of 750 μm^2 (Transmitter: 200 μm^2 and Receiver: 550 μm^2) [10]. In our experiments, we use cache block of size 64B and the bandwidth of normal RC links is 128 bits/cycle. Hence, a block is divided into 1 control flit(head) and 4 body flits.

For a pair of TL made of Coplanar Strip, the data rate is around 26.4 Gb/s ≈3 GB/s and the pair occupies a total pitch of 45 μm. At this high bandwidth, a single pair of TL can send a flit (128 bits) in a single cycle. Assuming each tile of size 5 mm $\times$ 5 mm the TLR area overhead for the transceiver circuit is around 0.003%. A pair of TL link consumes an area of just 45 μm and our six TLs occupy 0.011% of the total area. Thus for a 64-core system, combined area for both transceiver circuit as well as TL (total six) is around 0.08% of total chip area. This makes TL as a feasible interconnect for on-chip communication. The area is estimated based on the number of LUTs, registers and TLQ. In SBTR, the FSM restricts the maximum size of TLQ as six. There are total of 12 TLQs in the network. Therefore, total storage overhead to implement the TLQ is around 1 kB ($12 \times 6 \times 128$ bits). Six bits are required to maintain the *t_dest* information and an extra bit is used to indicate if the flit is *TL-candidate*. In our proposed techniques only TLRs and TL consume additional area.

We also considered the reflection as well as crosstalk between TLs. Reflection from nodes can be avoided by adding impedance matching loads and crosstalk can be avoided by using coplanar strips. Coplanar strips use differential signaling where a pair of TL runs parallel to each other. The transceiver circuit in our case is made of basic differential transmitter and receiver. The power consumption for the transceiver circuit is 9.5 mW [10] (Transmitter: 3.1 mW and Receiver: 6.4 mW).

7.2. Timing parameters for TL

Out of our six TL pairs, four (A, B, E and F) are having either a vertical or a horizontal orientation that spans over five hops. The

latency (propagation delay) for a TL of length 5 mm (1 hop) is 22 ps [10]. Since each tile is of size 5 mm × 5 mm, a TL of length 25 mm (5 hops) has a latency of 110 ps ≈ 0.111 ns. For an NoC operating at 1 GHz (1 cycle = 1 ns) a signal through a 25 mm TL requires less than 1 cycle to reach the other end. The remaining slack (1 ns–0.11 ns) is used to de/serialize the flit. Two of the diagonal TLs (C and D) have to span across ten hops (length >25 mm) thereby incurring a TL latency of two cycles. Our Verilog synthesis at 90 nm CMOS technology validates that the latency of transceiver circuit is 0.76 ns which is within the available slack. Note that TL latency is dependent on the length of the TL and the TL length varies as the tile size in TCMP varies.

8. Conclusion

CPU performance is greatly dependent upon miss penalties in TCMPs. In this work, we focus on reducing the number of intermediate hops that a long distance packet requires to reach the destination. Reducing intermediate hops also reduces the packet latency of cache miss packets thereby increasing the system performance. SBTR continuously monitors the system by changing the status of TLQ. Adding EVCs to SBTR benefits all types of packets in the network. Experimental analysis shows that in e-SBTR the number of packets flowing through TL is double than SBTR. The dynamic behavior to handle the load in e-SBTR is experimentally observed at high network traffic rates. Hence e-SBTR can improve network performance and also reduce miss penalties of cache misses in TCMPs. Considering the scaling of TCMPs, e-SBTR could be a better design choice for achieving reduced end to end communication latency.

References

[1] M.O. Agyeman, Q.T. Vien, A. Ahmadinia, A. Yakovlev, K.F. Tong, T. Mak, A resilient 2-D waveguide communication fabric for hybrid wired-wireless noc design, IEEE Trans. Parallel Distrib. Syst. 28 (2) (2017) 359–373.

[2] A. Allam, I. O'Connor, A. Scandurra, Optical network-on-chip reconfigurable model for multi-level analysis, in: Proceedings of the IEEE International Symposium on Circuits and Systems, 2010, pp. 3609–3612.

[3] Y. Asgarieh, B. Lin, Sharing a global on-chip transmission line medium without centralized scheduling, in: Proceedings of the 10th IEEE/ACM International Symposium on Networks-on-Chip (NOCS), 2016, pp. 1–8.

[4] R. Balasubramonian, N.P. Jouppi, N. Muralimanohar, Multi-Core Cache Hierarchies, Morgan and Claypool Publishers, 2011.

[5] C. Batten, A. Joshi, J. Orcutt, A. Khilo, B. Moss, C. Holzwarth, M. Popovic, H. Li, H. Smith, J. Hoyt, F. Kartner, R. Ram, V. Stojanovic, K. Asanovic, Building manycore processor-to-DRAM networks with monolithic silicon photonics, in: Proceedings of the 16th IEEE Symposium on High Performance Interconnects, 2008, pp. 21–30.

[6] B.M. Beckmann, D.A. Wood, TLC: Transmission Line Caches, in: Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2003, pp. 43–54.

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

[8] C. Bienia, Benchmarking Modern Multiprocessors (Ph.D. thesis), Princeton University, 2011.

[9] N. Binkert, B. Beckmann, G. Black, S.K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D.R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M.D. Hill, D.A. Wood, The Gem5 simulator, SIGARCH Comput. Archit. News 39 (2) (2011) 1–7.

[10] A. Carpenter, J. Hu, J. Xu, M. Huang, H. Wu, A case for globally shared-medium on-chip interconnect, in: Proceedings of the 38th Annual International Symposium on Computer Architecture (ISCA), 2011, pp. 271–281.

[11] A. Carpenter, J. Hu, J. Xu, M. Huang, H. Wu, P. Liu, Using transmission lines for global on-chip communication, IEEE J. Emerg. Sel. Top. Circuits Syst. 2 (2) (2012) 183–193.

[12] M.C.F. Chang, J. Cong, A. Kaplan, C. Liu, M. Naik, J. Premkumar, G. Reinman, E. Socher, S.W. Tam, Power reduction of CMP communication networks via RF-interconnects, in: Proceedings of the 41st IEEE/ACM International Symposium on Microarchitecture, 2008, pp. 376–387.

[13] S. Das, H.K. Kapoor, A framework for block placement, migration, and fast searching in tiled-DNUCA architecture, ACM Trans. Des. Autom. Electron. Syst. (TODAES) 22 (1) (2016) 4:1–4:26.

[14] S. Deb, K. Chang, X. Yu, S.P. Sah, M. Cosic, A. Ganguly, P.P. Pande, B. Belzer, D. Heo, Design of an energy-efficient cmos-compatible noc architecture with millimeter-wave wireless interconnects, IEEE Trans. Comput. 62 (12) (2013) 2382–2396.

[15] A. Ganguly, P. Pande, B. Belzer, A. Nojeh, A unified error control coding scheme to enhance the reliability of a hybrid wireless network-on-chip, in: Proceedings of the IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, 2011, pp. 277–285.

[16] J.L. Henning, SPEC CPU2006 benchmark descriptions, ACM SIGARCH Comput. Archit. News 34 (4) (2006) 1–17.

[17] Q. Hu, K. Wu, P. Liu, Exploiting multi-band transmission line interconnects to improve the efficiency of cache coherence in multiprocessor system-on-chip, in: Proceedings of the 28th IEEE International System-on-Chip Conference (SOCC), 2015, pp. 390–395.

[18] H. Ito, M. Kimura, K. Miyashita, T. Ishii, K. Okada, K. Masu, A bidirectional- and multi-drop-transmission-line interconnect for multipoint-to-multipoint on-chip communications, IEEE J. Solid-State Circuits 43 (4) (2008) 1020–1029.

[19] N. Jiang, D.U. Becker, G. Michelogiannakis, J.D. Balfour, B. Towles, D.E. Shaw, J. Kim, W.J. Dally, A detailed and flexible cycle-accurate network-on-chip simulator, in: The Proceedings of Performance Analysis of Systems and Software (ISPASS).

[20] X. Jiang, Y. Xie, Tranmission line bridge interconnects, uS Patent App. 14/860, 565 (Mar. 23 2017).

[21] C. Kim, D. Burger, S.W. Keckler, An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches, SIGARCH Comput. Archit. News 30 (5) (2002) 211–222.

[22] A. Kumar, L.-S. Peh, P. Kundu, N.K. Jha, Express virtual channels: Towards the ideal interconnection fabric, in: Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA), 2007, pp. 150–161.

[23] M. Meyer, Y. Okuyama, A.B. Abdallah, Microring fault-resilient photonic network-on-chip for reliable high-performance many-core systems, J. Supercomput. 73 (4) (2017) 1567–1599.

[24] H.K. Mondal, S. Deb, An energy efficient wireless Network-on-Chip using power-gated transceivers, in: Proceedings of the 27th IEEE International System-on-Chip Conference (SOCC), 2014, pp. 243–248.

[25] H. Mondal, S. Gade, M. Shamim, S. Deb, A. Ganguly, Interference-aware wireless network-on-chip architecture using directional antennas, IEEE Trans. Multi-Scale Comput. Syst. pp (99) (2016) 1–1.

[26] U.Y. Ogras, R. Marculescu, "It's a small world after all": NoC performance optimization via long-range link insertion, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 14 (2006) 693–706.

[27] M. Petracca, B.G. Lee, K. Bergman, L.P. Carloni, Photonic nocs: System-level design exploration, IEEE Micro 29 (4) (2009) 74–85.

[28] G. Sun, S.-H. Weng, C.-K. Cheng, B. Lin, L. Zeng, An on-chip global broadcast network design with equalized transmission lines in the 1024-core era, in: Proceedings of the International Workshop on System Level Interconnect Prediction, SLIP '12, 2012.

[29] C. Williams, B. Banan, G. Cowan, O. Liboiron-Ladouceur, A source-synchronous architecture using mode-division multiplexing for on-chip silicon photonic interconnects, IEEE J. Sel. Top. Quantum Electron. (2016) 473–481.

Dipika Deb is a research scholar at the Department of CSE, Indian Institute of Technology Guwahati, India. She did her B.Tech (CSE) from Tezpur University, Assam, India and M.Tech (CSE) from Indian Institute of Technology Guwahati, India. Her research interests are multicore computer architecture, network-on-chip and machine learning.

John Jose is an Assistant Professor at the Department of CSE, Indian Institute of Technology Guwahati, India. He did his Ph.D. (CSE) from Indian Institute of Technology Madras, India in 2014. Previously he did his M.Tech from Vellore Institute of Technology, Tamil Nadu, India. His research interest is on computer architecture with a special focus to performance optimization related to on-chip memory and communication aspects of multi/many core processors.

Shirshendu Das received the Ph.D. degree (CSE) from Indian Institute of Technology Guwahati, India in 2016. Previously he did M.Tech (CSE) from Indian Institute of Technology Guwahati, India. Presently he is an Assistant Professor in the Department of CSE, Indian Institute of Technology Ropar, India. His area of research includes computer architecture, network-on-chip and formal verification.

Hemangee K. Kapoor received Ph.D. degree (Computer Science) from London South Bank University, London, UK, in 2004. Previously she did M.Tech (CSE) from Indian Institute of Technology Bombay, India. Presently she is a Professor in the Department of CSE, Indian Institute of Technology Guwahati, India. Her current research interests include computer architecture, network-on-chip design, system-on-chip interconnects and process calculi.