TNT: A Modular Approach to Traversing Physically Heterogeneous NOCs at Bare-wire Latency

GOKUL SUBRAMANIAN RAVI, University of Chicago, USA
TUSHAR KRISHNA, Georgia Institute of Technology, USA
MIKKO LIPASTI, University of Wisconsin-Madison, USA

The ideal latency for on-chip network traversal would be the delay incurred from wire traversal alone. Unfortunately, in a realistic modular network, the latency for a packet to traverse the network is significantly higher than this wire delay. The main limiter to achieving lower latency is the modular quantization of network traversal into hops. Beyond this, the physical heterogeneity in real-world systems further complicate the ability to reach ideal wire-only delay.

In this work, we propose **TNT** or **Transparent Network Traversal**. TNT targets ideal network latency by attempting source to destination network traversal as a single multi-cycle 'long-hop', bypassing the quantization effects of intermediate routers via transparent data/information flow. TNT is built in a modular tile-scalable manner via a novel control path performing neighbor-to-neighbor interactions but enabling end-to-end transparent flit traversal. Further, TNT's fine grained on-the-fly delay tracking allows it to cope with physical NOC heterogeneity across the chip.

Analysis on Ligra graph workloads shows that TNT can reduce NOC latency by as much as 43% compared to the state of the art and allows efficiency gains up to 38%. Further, it can achieve more than 3x the benefits of the best/closest alternative research proposal, SMART [43].

CCS Concepts: • **Computer systems organization** → **Architectures**; **Interconnection architectures**; • **Hardware** → **Interconnect;**

Additional Key Words and Phrases: On-chip networks, interconnection networks, clock cycle slack, microarchitecture

ACM Reference format:
Gokul Subramanian Ravi, Tushar Krishna, and Mikko Lipasti. 2023. TNT: A Modular Approach to Traversing Physically Heterogeneous NOCs at Bare-wire Latency. *ACM Trans. Arch. Code Optim.* 20, 3, Article 35 (July 2023), 25 pages.
https://doi.org/10.1145/3597611

1 INTRODUCTION

[1]**Communication in the exascale era:** Multi-core processors with 10s to 100s of cores are commonplace in today's servers, supercomputers, and high-end workstations. Intel's line of Xeon Phi

[1]New Paper, Not an Extension of a Conference Paper.

Authors' addresses: G. S. Ravi, Department of Computer Science, University of Chicago, 5730 S Ellis Ave, Chicago, IL 60637, USA; email: gravi@uchicago.edu; T. Krishna, School of Electrical and Computer Engineering, Georgia Institute of Technology, 266 Ferst Drive, Atlanta, GA 30332, USA; email: tushar@ece.gatech.edu; M. Lipasti, Department of Electrical and Computer Engineering, University of Wisconsin-Madison. 1415 Engineering Drive, Madison, WI 53706, USA; email: mikko@engr.wisc.edu.

1544-3566/2023/07-ART35
https://doi.org/10.1145/3597611

processors have 50–100 cores per chip targeting high performance computing and Machine Learning. Moreover, we are ushering in the exascale era in which there is an exponential increase in data processing requirements from cloud computing, ML, and scientific applications - forcing futuristic **CMPs (chip multiprocessors)** to target 100s to 1000s of cores and other compute nodes on each chip [5, 14, 26, 37, 49].

The capabilities of such systems are often limited by the inefficiency of on-chip communication. Prior research has shown that server workloads executing on a CMP can lose as much as half of their potential performance due to long latency LLC accesses [24, 28, 29]. A key contributor to the long LLC access latency is the need for data/coherence traffic to often travel across many cores when slices of the LLC are distributed among them. Thus, the design of highly efficient low-latency on-chip communication is fundamental to further the exascale era. In this work, we tackle the challenges stemming from the need for modularity and the prevalence of heterogeneity in the **Network-On-Chip (NOC)**, which are critical to reducing the latency of on-chip communication.

Modularity: As the number of on-chip cores increases, the ability to build NOCs in a modular tile-scalable manner becomes critically important to streamline design and verification [11, 21, 52]. Such a modular design is composed of self-contained NOC tiles, replicated across the entire network. Each of these tiles are made up of the controlling router and the communicating links. Importantly, communication occurs in the form of hops between adjacent routers. The canonical topology for modular tile-scalable networks is the 2-dimensional mesh [30, 31, 71]. While the design-time benefits of modularity are abundantly clear, modularity can severely impair a rather important feature of the network - transmission latency. This is explained below.

We begin by noting that the optimal latency (T_{base}) of a packet in the network is described [20] by Equation (1), where H is the number of hops, t_r is the router pipeline delay, t_w is the wire (between two routers) delay, T_c is the contention delay at routers, and L/b is the serialization delay for the body and tail flits, i.e., time for a packet of length L to cross a channel with bandwidth b. With traditional synchronous NOCs, all of the above delays are quantized to clock cycles. In a 2-dimensional mesh, the average hop counts increase linearly with expansion in each dimension of the mesh, therefore linearly increasing average packet latency.

$$T_{base} = H.t_r + H.t_w + T_c + \frac{L}{b} \tag{1}$$

$$T_{wire} = (\eta_{chip}.H).t_w + \frac{L}{b} \tag{2}$$

$$T_{wire}(Physical) = \left(\Sigma_1^H \eta_{link}\right).t_{w\_max} + \frac{L}{b} \tag{3}$$

$$T_{TNT} = 1.t_r + \left(\Sigma_1^H \eta_{link}\right).t_w + T_c + \frac{L}{b} \tag{4}$$

Latency: Next, we examine the network's true traversal capability. The ideal latency for a packet to traverse through the network would be the delay incurred from *bare-wire* traversal alone (i.e., without restrictions imposed by routers and congestion). The wire delay along a packet route of H hops, is proportional to H times η, with $0 < \eta <= 1$. Here η is the *cycle utilization per hop*. It implies the fraction of a clock cycle that it actually takes to traverse one hop of wire, and is a function of frequency, technology node, and so on. Thus, the ideal *bare-wire* time delay for an uncontested packet to traverse H hops is given by Equation (2). Note: this uses η as a conservative chip wide (η_{chip}) estimate.

Thus, it is clear that in a canonical modular network topology like the mesh, the latency of a packet to traverse the network (T_{base} in Equation (1)), is significantly higher than the *bare-wire* delay through the network (T_{wire} in Equation (2)). The main limiter to achieving lower latency is the impact of modular design: the quantization of network traversal into hops, introduced by the packet's interaction with routers at each node in the network. Hop quantization primarily increases packet latency in two ways: (a) it introduces router delays at each hop, and (b) it underutilizes the wiring's per-cycle traversal capability, i.e., traversing only a single hop in a cycle even if the wire is capable of multiple hops worth of traversal per cycle ($1/\eta$). This is a cause for concern because for exascale systems, the high hop counts would lead to horrendous on-chip network traversal latency and energy, creating a stumbling block to core count scaling [43].

Heterogeneity: Previously when we discussed wire traversal capability (η_{chip}), we remained agnostic to the impact of physical chip heterogeneity. We assumed that all links are of same length and the wire delay is constant across the chip. In reality, even given a logically homogeneous system, the physical design is often heterogeneous. Heterogeneity stems from physical die characteristics: ① Unequal sizes and aspect ratios of cores, **memory controllers (MCs)**, ② Their sparse distribution across the system, and ③ Within-die process variations. Thus, in a real layout, η across a chip is not constant and is dependent on the route of interest, with each link having individual η_{link}. Thus, Equation (2) is modified to Equation (3). Note that t_{w_max} here is the wire delay for the longest link—since our baseline assumes at most a single cycle per link, for the longest link with maximum physical variation bound, η_{link} would be 1. For all others, η_{link} would be less than one.

TNT - coping with modularity and heterogeneity for ideal latency: To reach *bare-wire* packet latencies in traditional NOCs, the network design should attempt to achieve the above T_{wire} packet delay, but without straying away from the modular design philosophy. This is clearly a challenge considering the conflicting elements in both goals. Further, a solution that is exploiting inherent design characteristics (wire delay), should be able to cope with the physical heterogeneity that is part and parcel of the design. We propose ***TNT*** *or* ***Transparent Network Traversal*** to overcome these challenges - ideally achieving end-to-end network traversal in a single pass at the best physically capable wire delays, while performing only neighbor-to-neighbor control interactions.

TNT pushes closer to the ideal *bare-wire* latency by attempting source to destination traversal as a single multi-cycle *long-hop*, bypassing the quantization effects of intermediate routers via *transparent* data/information flow, reducing the packet latency to Equation (4). In TNT, the *long-hop* from source to destination, if without conflicts, results in ⓐ only a one-time router delay, and ⓑ allows the physical wiring's traversal capability to be utilized via multiple, possibly non-integral, hops worth of traversal per cycle. Further, ⓒ by tuning η individually for each potentially heterogeneous link/tile, TNT exploits any route's traversal capability to the best extent.

TNT's transparent traversal is achieved by designing *A Modular and Flexible Lookahead Network* - a novel lookahead control path. The control path performs simple delay-aware neighbor-to-neighbor interactions via lookahead requests to enable end-to-end transparent flit traversal. Note that these neighbor-to-neighbor interactions create potential for conflicting requests under high traffic. Thus, to allow for TNT's capability of traversing multiple hops per cycle, multiple conflict resolutions would need to be performed in sequence within a cycle, without being hindered by the synchronous quantization introduced by conventional sequential logic. To achieve this, the control path is designed as follows. Control is built atop a light-weight mesh. It ⓐ carries lookahead requests to set up transparent paths, ⓑ tracks wire delay (using time stamps) to monitor accumulation of path delay from one link to the next, ⓒ performs simple combinational lookahead-request conflict resolutions at routers, and ⓓ respects timing (violation) constraints - a must-have for multi-cycle designs. Figure 1 illustrates the above discussion.

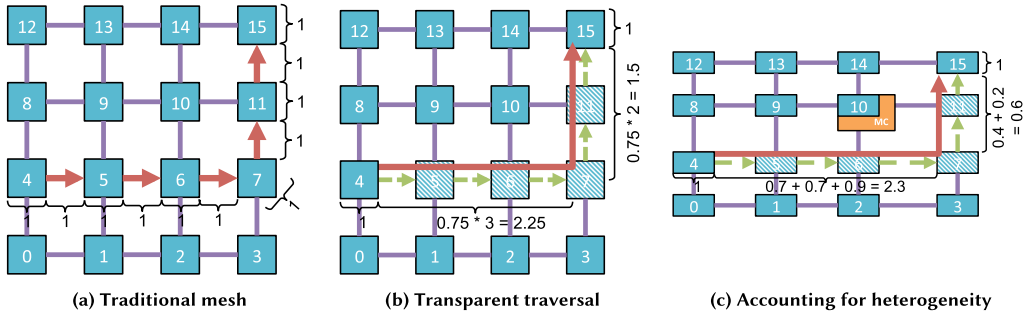


Fig. 1. 5 hop request in a 4*4 mesh. (a) Traditional mesh takes 11 cycles - 1 per router/link. (b) In TNT, the request flows through in a single pass bypassing intermediate routers. The lookahead requests enabling this are shown in green. In this homogeneous design, the link+switch wire traversal capability is $\eta_{chip} = 0.75$, thus the request completes in 6 cycles. (c) In this non-homogeneous design, nodes have a greater width than length, and node 10 is attached to a large memory controller. TNT is still able to account for per-link delay, allowing request completion in 5 cycles.

Summary of contributions:

① TNT is the first NOC proposal to incorporate on-the-fly delay tracking to enable multi-hop many-cycle traversals, avoid collisions and metastability, and approach ideal *wire-only* network latency.

② TNT uses a novel transparent traversal approach, enabling flits to travel the entire route from start-end node in a single "long-hop" without requiring require all-to-all request links in the NOC to achieve this.

③ It is the first design to exploit wire heterogeneity arising from physical constraints like node placements and variation.

④ TNT is built in a modular tile-scalable manner, a key factor for realistic implementation. It utilizes a Modular Flexible Lookahead Network control path that only requires neighbor-to-neighbor control interactions, introduces a novel time-stamp based wire delay tracking approach, performs combinational resolution of request conflicts, and is designed with awareness to timing constraints.

⑤ Analysis on Ligra graph workloads shows that TNT is able to to reduce LLC latency by up to 43%, improves performance by up to 38%, reduces dynamic energy by as much as 35%, compared to the baseline 1-cycle router NOC. Further, it can achieve more than 3x the benefits of best/closest alternative research proposal, SMART [43].

2 MOTIVATION: TRAVERSING THE NOC

Heterogeneity in physical dimensions: The logical topology of a multi-node system and its NOC can often be different from the physical design. The floor plan of the entire chip needs to carefully consider the contents of tiles (cores, caches, memory controllers, etc.), the distribution of the tiles, and the placement of links relative to these tiles. Wire congestion, delay and energy overheads, as well as the use of global/semi-global metal layers, all factor into floor planning considerations. Such factors can impose significant physical heterogeneity on an otherwise logically homogeneous system. In this work, we are specifically concerned with the impact of this heterogeneity on link lengths/delay in the NOC, which in turn impact TNT's NOC traversal latency. The specific factors we discuss here are: the types of nodes in the system, their aspect ratios, and their distribution/connectivity.

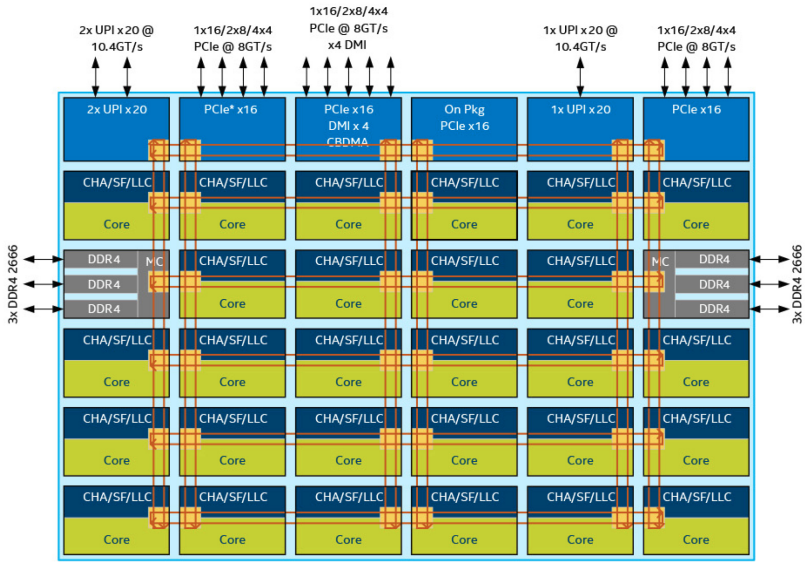


Fig. 2. Intel Skylake "typical" floorplan [33].

The nodes in the CMP are usually made up of cores plus cache slices, or **memory controllers (MC)**. As the number of processor cores grow, it is not practical to assume each tile will have an MC directly attached - they are limited by packaging constraints, primarily the number of available pins. For instance, Intel Skylake servers [33] can have 10–28 cores with 2–4 MCs, connected by a mesh topology, with MCs at the top and bottom of the fabric. Further, the NOC nodes potentially have non 1:1 aspect ratios. Die images of the Intel Icelake server shows that its Sunny Cove cores are twice as wide as they are long, with dimensions of roughly 4mm by 2mm [34]. Also, the MC nodes can be multiple times the size of the core nodes - die images of the AMD Zeppelin system shows MCs twice the area of its cores, with 1 MC for every 4 cores [4].

Figure 2 shows the floorplan for Intel Skylake with 36 nodes - 28 cores, 2 memory controllers, and the rest for UPI, PCIe/PCH connectivity, Omni-Path fabric, and so on [33]. The non-uniform node dimensions and distributions are evident from the figure. Further, it is visible that since the core tiles are mirrored with the **NIC (network interface controller)** in the corner, the east-west links from column 1 to 2, 3 to 4, and 5 to 6 are very short, while 2-3, 4-5 are very long, and the north-south links are uniformly medium length. By rough estimation, the resulting mesh link lengths are in the 1(25%): 2(50%): 8(25%) ratio. Thus, it is evident that even systems thought to be logically homogeneous, are impacted by heterogeneity imposed by physical floorplan constraints, resulting in link lengths (and traversal times) being different across different routes on the chip.

Wire transmission distance: The time delay for bits to traverse across a wire is managed via repeaters. Repeaters are sized appropriately to allow bits to traverse wires of a particular length in a required amount of time. The limiting constraint to longer wires and higher frequencies is timing closure failure and unacceptable energy consumption. In Figure 3, we use DSENT [66] to plot the achievable transmission distance over wires per clock cycle, against the consumed energy. We analyze the data for different technology nodes and for different frequencies. At a frequency of 1 GHz, link drivers are able to theoretically drive up to as much as 20mm across 45/32/22nm technology nodes. At higher frequencies the traversal capability drops super-linearly but can still be considerable. The reason for the slowdown is the slew rate of the signal [42]. At higher frequencies (i.e., shorter clock periods), the repeater is unable to produce the extra current

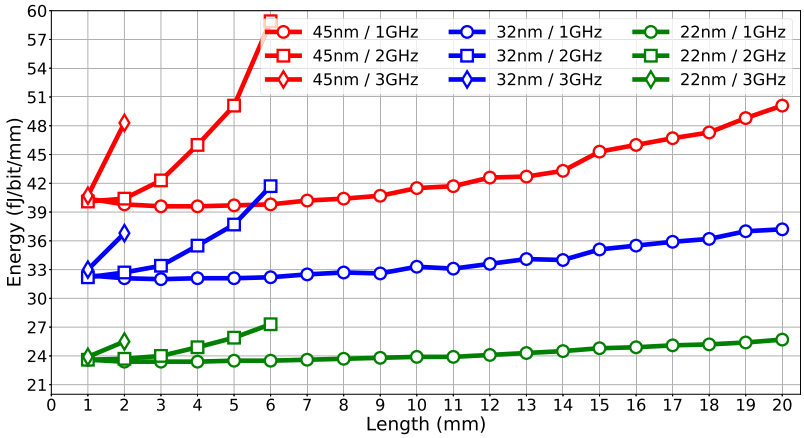


Fig. 3. Transmission distance per clock cycle.

required to reduce the signal's rise/fall time proportionally [42]. At lower technology nodes, while transmission energy shrinks, there isn't a significant impact on the wire delay due to compensatory effects from scaling resistances and capacitances [42].

Impact of variation: Process variations are caused due to wafer characteristics, doping fluctuations [15], imperfections in manufacturing process, and so on, leading to potentially large variations in device attributes [27]. Since on-chip networks connect distant parts of the chip, they are especially vulnerable to within-die parameter variations. The systematic portion of process variation can be characterized based on three parameters - μ (mean wire delay), σ (standard deviation), and ϕ (spatial correlation) [62]. NOCs are usually designed to work under the most unfavorable parameter values in the chip [6] - thus the latency of traversing a single hop is governed by worst case delay characteristics. This potentially results in a considerable fraction of the switch/link traversal clock cycle being wasted as slack. Prior works have noticed more than 30% difference in the voltage (or correspondingly, delay) required for error-free functioning of routers across a 64-node NOC [6]. It is intuitive that capturing the delay characteristics across each of the different tiles in a chip at design time, and tuning the local η accordingly can improve the traversal capability along variation-robust routes.

3 TRANSPARENT NETWORK TRAVERSAL

TNT uses transparent flow based flit traversal to enable flits to travel the entire route from start node to end node in a single "long-hop," covering multiple hops over the minimum circuit-constrained (i.e., *bare-wire*) number of cycles. Only a single pass for the entire flit route instead of a sequence of hops, avoids the quantization effects of intermediate routers. The flow of a single-flit packet in TNT is described below.

Figure 4(a) shows five routers with additional per-router control enablers to allow transparent traversal. There are three primary control enablers: (a) Buffer Write ($buffer$) at the input flip-flop to determine if an incoming input signal should be latched or not, (b) Bypass Mux (mux) at the input of the crossbar that chooses between a local buffered flit and the incoming bypassing flit on the link, and (c) Crossbar select ($xbar$) connecting inports to outports. The packet traverses 5 hops from router R0 to R5 and the chip-wide wire delay analysis (at design time) estimates that $\eta_{chip} = 0.45$ i.e., 2.25 hops per cycle. For the sake of simplicity, this example assumes no heterogeneity. The life time of the 5-hop packet is as follows:

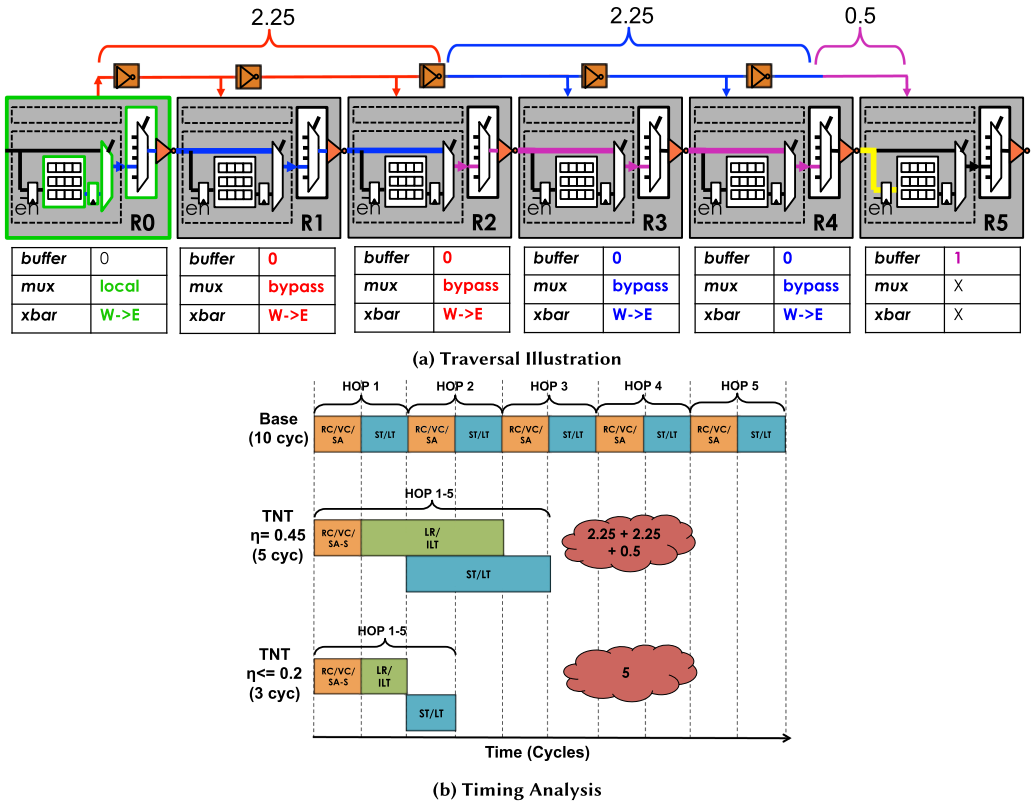


Fig. 4. TNT: Transparent Network Traversal in a single long-hop.

① Cycle 1: A *long-hop* starts from start router R0 where the flit is initially buffered (green outlined in Figure 4(a)). Apart from traditional RC/VS, this stage performs **Switch Allocation at Source (SA-S)**, identical to SA in a conventional pipeline: every source router chooses a winner for each outport from among its buffered flits. Assuming the flit wins **SA-S**, TNT will attempt to send the flit all the way to the destination R5 in a single transparent multi-cycle *long-hop*.

② Cycle 2: The next step is to initiate the **Lookahead Request (LR)** (shown in red in Figure 4(a)). The LR flows ahead of the actual flit in the same route from source to destination but on a **Lookahead Network (LN)**. The role of LR is to set up router transparence (i.e., router bypass) *just in time for arrival of the data flit*. Note that LR can be a multi-cycle request (since it could take multiple cycles to travel from source to destination). In this example, based on the 2.25 hop per cycle wire delay, the LR has the capability to reach routers R1 and R2 within Cycle 2 and be 0.75 hops from R3. When the LR first reaches intermediate router R1, **Idle Link Takeover (ILT)** is attempted at the router. The role of ILT is to enable transparent use of the router's outport and link, in the required direction, if they are idle (i.e., not set up for use by already buffered flits) in this cycle. This determines if the data flit will be able to flow transparently through the router when it arrives there in the subsequent cycle. If the LR succeeds at ILT at R1, it flows through to R2 where the same process is repeated. If the LR wins ILT at R1, R2, the respective bypass muxes are set to *bypass-mode*. Buffering is disabled and the crossbar is set for West to East traversal. Note: The LR are also routed just like the data flits via lightweight routers - discussed later in Section 4. LR routing details are left out of Figure 4(a) for simplicity.

③ Cycle 3: There is both data flow and control flow, both shown in blue. The data flit traversal is initiated out from R0 to perform the multi-cycle 'long-hop' to destination R5. Based on the control signals set by the LR in the earlier cycle, the data flit is able to bypass buffering and flow through transparently at router R1 and R2. Further, the data flit flows beyond R2 speculatively so as to use up wire traversal capability completely. At the end of cycle 3, the data flit is in flow between routers R2 and R3 (it would have covered 2.25 hops and be 0.75 hops away from R3). Meanwhile, the LR-flow continues ahead of the data and it covers another 2.25 hops reaching routers R3, R4, attempting ILT at the routers, and is halfway between R4 and R5 (assuming successful ILT). At R3, R4 the control signals are set appropriately, so as to allow transparent data flit traversal in the following cycle.

④ Cycle 4: The data flit covers another 2.25 hops, transparently flows via R3, R4, and is half a hop away from R5. The LR covers its remaining 0.5 hop and reaches destination R5, where it enables buffering. All are shown in pink.

⑤ Cycle 5: The data flit travels the remaining 0.5 hop (shown in yellow) and gets latched at R5, completing the 5 hop traversal in a single *long-hop* pass, consuming five cycles.

Note that the links/routers over the path from R0-R5 are *not* being all held by this request throughout the duration of the long-hop - the usage is pipelined. For example, in cycle 3, routers R1 and R2 are free to receive new LR requests and in cycle 4 they are free to get new flits.

The benefits of TNT for the single flit-packet (discussed above) are shown with a timing illustration. Figure 4(b) depicts the timing diagram for the 5-hop traversal with no conflicts. First, it shows the baseline design which takes two cycles per hop. Thus the packet's 5 hop traversal consumes a total of 10 cycles. Next, the timing diagram for the previously described scenario with $\eta = 0.45$ is depicted, consuming a total of five cycles. Finally, the timing diagram for the lowest possible latency for this traversal is shown, achievable for $\eta <= 0.2$. The entire 5-hop flit/control traversal can complete in a single cycle each, resulting in a total request time of only three cycles. TNT benefits significantly over a wide η range, allowing its beneficial usage over different NOC designs, running at a range of frequencies, with a variety of tile sizes and so on.

4 MODULAR LOOKAHEAD NETWORK

An integral part of long-hop traversal model that TNT proposes is the **Lookahead Network (LN)** for control requests, which establishes the transparent multi-hop path *just in time* for the flow of the data flits. In accordance with the philosophy of building realistic NOCs in a modular and tile-scalable manner (Section 7.1), LN's features are self-contained within each tile and identical across every tile, even though it has an end-to-end influence on the NOC. LN's features are listed below and discussed in following sections:

① LN simply adds light-weight links between adjacent routers of the original data flit network (a 2D mesh).

② **Lookahead Requests (LR)** flow through the LN. LRs setup sections of the transparent path just prior to arrival of the data flits, up to $\frac{1}{\eta}$ routers in a cycle. LRs are routed through the LN, through the same set of routers as their respective data flits, but arrive at routers one or more cycles earlier.

③ LRs carry routing information along with a 4-bit **time-stamp** which is used to avoid collisions and prevent timing violations, enabling heterogeneity-cognizant transparent traversal.

④ An LR entering an intermediate router's inport performs **Idle Link Takeover (ILT)**, during which it attempts to take over the router's outport and link for a particular cycle. Only if it wins ILT can it flow through to the next routers. Further, winning ILT guarantees transparent flow at the router for its data flit.

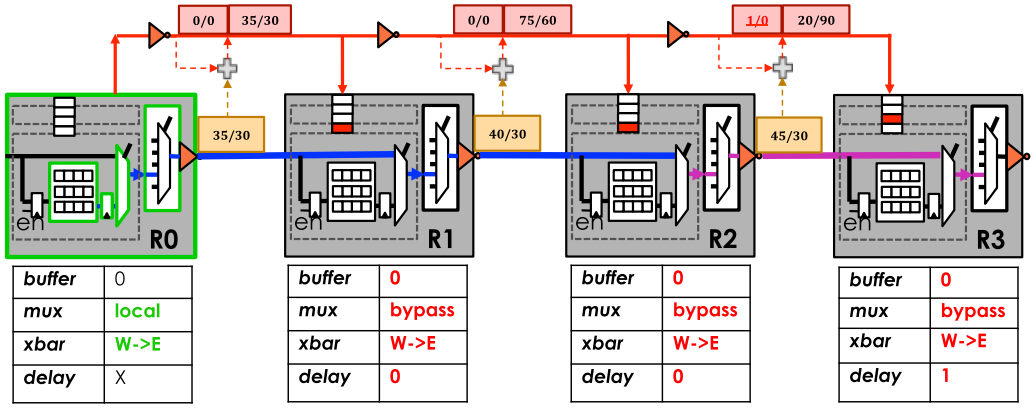


Fig. 5. Time-stamp based tracking design.

⑤ LR-flow through the router, along with ILT, is completely combinational (since multiple router hops are to be covered in a cycle) and the ILT scheme is implemented to suit combinational conflict resolution.

⑥ The entire design is implemented with a globally synchronous clock and timing violations are avoided via safeguards when requests cross clock edges/boundaries.

⑦ The design has low wiring/area overheads, low energy spent on signaling,p; no false negatives at routers and numbers of conflicting LRs resembling a traditional mesh.

4.1 Time-stamp based Delay Tracking

The design illustrated in the previous section assumed that link delays are constant across the route, and were the same for data (flits) and control (LR) paths. This is challenging to enforce in terms of circuit and layout. Thus, we design our delay-tracking scheme to support heterogeneity. We build a flexible design wherein control flow and data flow can be timed separately, with unique per-link delays. These per-link delays are stored as a look-up value at each link/router-outport. Measurement of per-link delay is discussed in Section 5.

The only requirement is that the control path is faster than the data path - easily ensured in LN implementation (Section 5). With this constraint, the flexible design is achieved via time-stamp based delay tracking. For each long-hop pass, the LR carries information (as time stamps) about the cumulative delay of the LR itself and the data flit respectively, over the route of the long-hop. This is achieved by accumulating wire delays over each flit/LR link, by means of time-stamps. At the start of a TNT pass, the time-stamp to be carried through by the LR, for itself and the data flit, are initialized to 0 and these time stamps are updated (accumulated) at every link along the route. Transparent traversal decisions are made based on these time-stamps, which are explained with an example and illustrated design in Figure 5.

The example illustrates 3 hops of traversal from router R0 to R3. Here, $\eta^{LR}_{0/1/2} = 0.30$ i.e., delay per LN control link is 0.30 cycles. Further, $\eta^{Data}_{0/1/2} = 0.35/0.40/0.45$ i.e., the delays for the sequence of 3 data hops in the route are 0.35/0.40/0.45 cycles, respectively (including within-router delays). These example η values of 0.3–0.5 are within the typical range. Further, the control path is faster than the data path as discussed above.

These delays are stored at 4-bit precision, i.e., 1/16 of mesh frequency, at the routers (yellow). We find 4 bits of precision to be near optimal but still very inexpensive. Details of timing measurement, verification and programming the routers is discussed in Section 5 under 'Timing Measurement'

and 'Timing Verification'. At the top of the figure, we see the accumulation of data/LR delays as well as overflow bits (red boxes), which are explained below:

① When the LR is sent out from R0 in Cycle 0, it separately accumulates the data link delay (D^{Data}) and the LR link delay (D^{LR}) and carries the information to the downstream routers (assuming it wins allocation throughout).

② Flowing from R0 to R1, it carries $D^{LR} = 30$ and $D^{Data} = 35$ i.e., LR reaches R1 at 0.30 cycles and the data flit will reaches at 1.35 cycles. Thus, R1 is required to be transparent on the immediate cycle after LR arrival (Cycle 1).

③ Flowing from R1 to R2, the cumulative delays are updated to $D^{LR} = 30 + 30 = 60$ and $D^{Data} = 35 + 40 = 75$, i.e., LR reaches R2 at 0.60 and data reaches at 1.75 cycles. Thus, R2 should also be transparent in Cycle 1.

④ Flowing from R2 to R3, we have $D^{LR} = 60 + 30 = 90$ and $D^{Data} = 75 + 45 = 120 > 100$ i.e., LR reaches R3 in cycle 0 (at 0.90) while data only reaches in Cycle 2 (at 2.2). This results in changes to overflow bits: Ovf^{LR} is low, while Ovf^{Data} is set to high - meaning that the lag between the LR and flit exceeds 1 cycle. Thus, this LR is buffered for a cycle at R3 and then attempts to win R3 transparence for Cycle 2.

The overflow bits are reset appropriately at clock boundaries. Note, it is possible for greater than 1 cycle delay between LR and data - the LR is buffered accordingly. We do not observe lag beyond 1-2 cycles, so small buffers suffice.

Key takeaway: TNT tracks the timings of heterogeneous links throughout the NOC, and routers are made transparent only in clock cycles that data flits will actually arrive.

4.2 Idle Link Takeover

TNT performs a sequence of up to $\frac{1}{\eta}$ ILTs combinationally every cycle. There is transparent flow contention possible at every router, from requests arriving at different timing instants within the same clock cycle. Thus, it is necessary to design an effective combinational contention resolution scheme which decides the appropriate requests to flow through.

A conventional priority scheme [40] for choosing requests at each router which, say, prioritizes buffered requests over straight over turn, will not suffice for TNT. With this conventional scheme, if at the start of a clock cycle only a turn request is present at a router, it will be allowed to flow through. But if a straight request arrives at a later instant in the same cycle, it will *still be allowed to flow through* since straight requests have higher priority (assuming no buffered requests). This leads to false requests and incorrect data/information flow which are detrimental to throughput. Thus, such a scheme is suited only to a traditional synchronous design wherein, all requests vying for contention are available prior to the priority scheme deciding on the winner (i.e., at clock edge).

On the other hand, in the case of TNT, requests can arrive at any instant within a clock cycle. The simple solution to this contention resolution challenge is to *prioritize the first arriving request* at a router in any cycle and block any future requests. In other words, an idle link/outport should be declared busy as soon as it is granted to an LR - hence the name "Idle Link Takeover". The logic to implement this is illustrated in Figure 6, which shows LR requests from North, West, and South inports of a router competing for the East outport. At the router, the LRs arrive at a latch whose opacity is controlled by the "OR" of all the LRs (i.e., their valid bits). Thus, the latch becomes opaque as soon as the first LR arrives during a clock cycle. This first arriving LR is declared winner and can potentially pass through transparently, followed by its data flit in the next clock cycle. Later arriving LRs all buffer at this router. Via this first-come first-serve priority logic, false requests and incorrect information flow to downstream routers is avoided (except as described in Section 4.3). Also, if a local/buffered request is present, priority is for that local request and incoming requests are buffered. Note: If any LR input is set during a cycle, it stays set throughout the cycle i.e., back

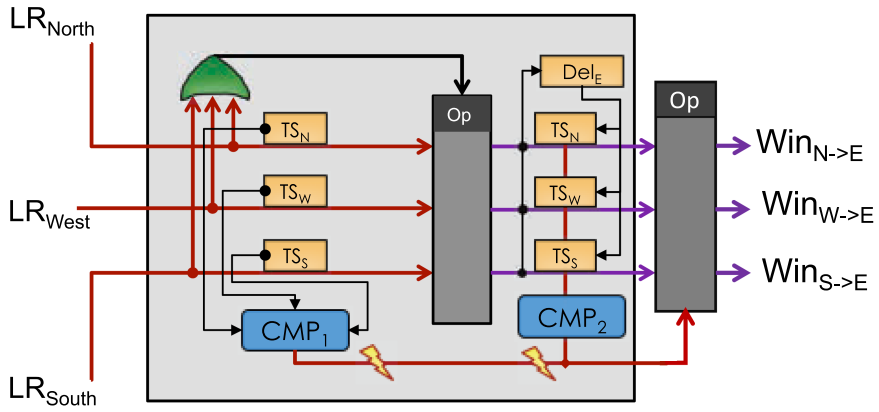


Fig. 6. ILT + Safeguard (Buffering and Reset mechanisms, as well as the "buffered is highest priority" logic are left out of the figure for simplicity.)

to back LRs are not sent on the same cycle from one direction. So hold requirements for the latch input (wrt the first LR) are easily met - the first LR is already present sufficiently before and after the latch is enabled. Handling multiple LRs from different directions is discussed in Section 4.3.

Finally, it should be noted that, every cycle, the inputs to the block in Figure 6 and the internal latch are synchronously reset, plus the outputs of the block are latched. There is no carryover of signals within the block from one cycle to the next. This is unnecessary because, in any given cycle, any requests which do not flow through transparently are buffered (and are prioritized in subsequent cycles). So all the information flowing into the router in one cycle is captured in that same cycle. Thus, there is no interaction between the "new" LRs on a new cycle and "old" LRs from a previous cycle. Note that this does not require the old LRs signals to fall—the reset is sufficient. In all, the circuit in Figure 6 augments the baseline design which already synchronously handles buffering, routing, arbitration, and so on.

Key takeaway: Conventional conflict resolution schemes are insufficient for multi-hop multi-cycle traversal. We design a first-come first-serve combinational scheme to address this.

4.3 Timing Safeguard

In the above description of the ILT logic, it is possible that multiple LRs arrive at the same time or within very short intervals of each other - which could cause the latch in Figure 6 to glitch as it becomes opaque. This is avoided by the **ILT Safeguard** mechanism - a simple time-stamp compare logic CMP1, which compares the time-stamps of the incoming LRs ($TS_{N/W/S}$ in figure, left of the latch). If incoming LRs have time-stamps within a "violation window" of each other, the CMP1 logic triggers a fault, no LRs are allowed to pass through and all are buffered at the router. Thus, the ILT Safeguard prevents the scenario when *multiple* LRs arriving at a similar time and compete for priority.

There is one other potential for timing violation, caused even by a *single* LR request - due to the prevalence of multi-cycle multi-hop paths. If a request reaches a destination router very close to the clock boundary, a hold-time violation could occur leading to meta-stability if buffering is required. This is easily averted via the **LR Safeguard** - simple logic shown as CMP2 in Figure 6. Once an LR has been latched at a current router, its arrival time at the next router is estimated. This is calculated by summing the current time-stamp of the winning LR (one of $TS_{N/W/S}$, seen right of latch) and the wire delay of the control out-link (shown as Del_E). If this new summed up time stamp of the LR falls within a "violation window" of the rising edge of a clock, then there is

potential for timing violation at the next router. If so, the LR is buffered at the current router and not allowed transparent flow.

Thus, a request is sent out only if both CMPs are set. In the figure this is indicated by the latch at the output of the router which is only made transparent if both the CMPs are set. In both safeguards, the logic can be made conservative or aggressive by lengthening or shortening the violation window. In our work both CMP1 and CMP2 assume a +/− 5% timing violation window. Section 6.1 discusses its trade-offs. Also, Section 5 discusses timing measurement and verification.

Key takeaway: Multi-cycle schemes have timing constraints challenges that we solve by building safeguard mechanisms which utilize the request time-stamps.

4.4 Control Flow and Additional Optimizations

① *Control Flow:* We use credit-based flow-control. LR can flow ahead, only if buffering downstream is guaranteed. This is checked in parallel to ILT. Note that this "checking" is not full VC allocation. Rather, it is just a boolean yes/no check on the immediate downstream buffer. Credit signals travel only 1 hop worth distance.

② *Route computation for several hops:* We only allow flits and LR flow to follow deterministic routes (oblivious dimension-ordered routing).

③ *Avoiding flit reordering:* This is ensured by the deterministic routing, in conjunction with the priority mechanism.

④ *Avoiding premature buffering of Body/Tail flits:* The Head reserves a VC at all its intermediate routers, even though it does not stop there. These are freed by the Tail.

⑤ *Low-load SA-S bypassing [36]:* An incoming flit that is denied transparent flow can speculatively send the next LR without waiting for SA-S (Switch Allocation at Source), if no flits ahead of it in the input buffer queue.

⑥ *Preventing short paths/corruption:* An opaque boundary is maintained between physically adjacent flits on back-to-back cycles. We do not see any impact on gains at high traffic, but path diversity can be employed if required. If necessary, they can be alleviated by employing path diversity.

5 TNT IMPLEMENTATION

We design TNT for a frequency of 1 GHz at the 22nm technology node. TNT specific components were implemented in RTL and synthesized using Synopsys Design Compiler. The rest of the components of the NOC are challenging to model and synthesize in RTL. Therefore, energy and area numbers for router and link components are estimated with the help of DSENT [66]. Performance results for both Synthetic Traffic and Full System are obtained using GEM5 [12] + Garnet2.0 [2] infrastructure, which provides a cycle-accurate timing model. The evaluated system configuration is shown in Table 1. Our baseline incurs 2-cycles-per-hop (state-of-the-art 1-cycle router), similar to VC-based designs commonly adopted in industry/academic prototypes, including relatively larger networks [22, 32, 58] (more in Section 7.1). TNT assumes a chip-wide synchronous clock domain for the NOC routers, as is pursued in industry designs [32]. In contrast, the nodes attached to each router may (and often do) have private clock domains.

The TNT data-path is modeled as a series of 128-bit 2:1 mux (for bypass) followed by a 4:1 mux (crossbar), followed by 128-bit links. LR signals are 16-bit, carrying destination router id (6-bit), LR delay (4-bit) and flit delay (4-bit). The control-path consists of LR traversal via the LN mesh: a series of 16-bit links + logic delay (from ILT) at each router. In the LN router (Figure 6) - there are 2–4 LR links per router, thus up to 4 LRs competing for ILT. Delay operations are performed at 4-bit precision. Link lengths vary from 1-8mm.

Table 1. Target System

Processors	64 3-wide OOO	**Freq/Tech**	1 GHz/22 nm
L1 Cache	Pv. 32kB, 2 cyc	**Link/Flit**	1-8 mm/128bit
L2 Cache	Sh. 1MB/core, 12 cyc	**Topology**	8-ary 2-Mesh
Coherence	MESI	**Routing**	XY
Vir. Net	3 (req, fwd, resp)	**Vir. Chan**	4+4+4

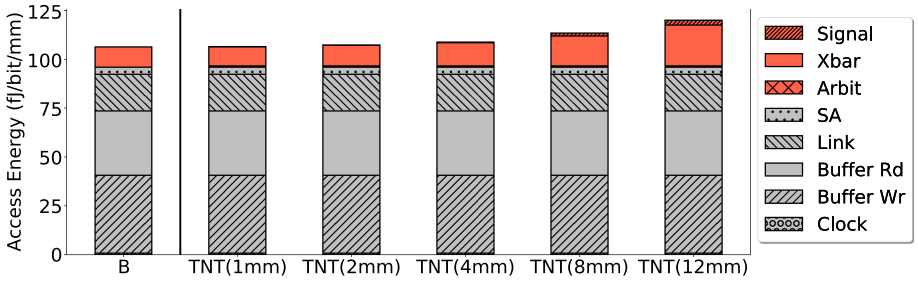


Fig. 7. Energy per access.

Design Complexity: TNT employs a mesh network for lookahead, resulting in only 2-4 links per router, only one request sent out for lookahead which is routed through the mesh and further, a fixed lookahead routing complexity which does not increase with larger networks. The mesh routing also ensures that there are no false lookaheads. None of TNT's signaling components increase in complexity with an increasing number of nodes in the system.

Design Timing: The total conflict-resolution delay for an N-hop request in the LN mesh design is optimally a sequence of N ILT actions on its constant complexity ILT logic—the critical path for the Lookahead Network consists of multiple hops worth of ILT logic (CMP1, CMP2, the latches, the delay lookups, and the timestamp increments), along with the LN traversal. The compute/decision logic in the above is only working on 4-bit timestamps, so it is very fast. The ILT TNT logic is only a small fraction of the control path, which, in totality, is only 16-bits wide, and is itself much faster than the 128-but data path, since it requires a much smaller crossbar and less wiring. Thus, though the number of ILT logic computations is higher in a mesh compared to point-to-point signaling, energy/delay analysis shows that overall cost is minimal (more details below), making it the all-round favorable option. Analysis on DSENT shows that for 1GHz, TNT's control/data path are able to achieve as much as 12mm per cycle traversal with low overheads, competitive with any traditional design. In the control path, ILT accounts for 20% of total delay while the rest is from LR traversal. Note that the design allows for the control path to be equal to or faster than the data path because the control path is only 16-bits wide while the data path is 128-bits wide. All the routing structures are also thus smaller (and faster) for the control path.

Access Energy: Figure 7 plots the energy/bit/hop for each NOC request component for TNT, when designed to achieve a traversal capability of 1–12mm in a clock cycle, compared to a traditional mesh. Energy components from clock, buffers and local switch allocation (for the data/base portions of all the NOCs) are similar across all designs. The data Xbar energy increases with traversal capability and comprises of the energy consumed by the data path repeaters for driving the bypass and crossbar muxes. The total energy consumed by the LN is broken into the *Signal* (LR/LN-links) and *Arbit* (ILT/LN-router) components. The *Arbit* also includes costs associated with safeguard mechanisms and delay tracking which are negligible relative to the payload, given the small number of bits required for them. *Signal* energy grows only linearly and *Arbit* energy per

hop remains constant (and low). Access energy increase for TNT over baseline is <1% when designed for low traversal capability and <10% even at high capability. Note, the baseline energy is only for a single hop.

Area: Area overheads of TNT range from 0.15% to 5% depending on the wire traversal capability of the design (ranging from 1–12mm per cycle) These are worst-case estimates since it is possible for wiring to be overlapped by NOC nodes/tiles, thus having less impact on the overall area [3, 36, 56].

Timing Measurement: Link delay varies with NOC heterogeneity, as discussed in Section 2. We model floorplan considerations based on the typical floorplan in Figure 2 and also sweep through a wide design space in Figure 11. Process variation is modeled similar to prior work [62]. In a real design, we expect these characteristics to be measured post-fabrication, interpreted as cycle slack, and written into the per-router lookups (discussed in Section 4.1). As shown previously [17, 59], we expect state-of-the-art CAD tools to be capable of (or extendable to) such analysis. The timing measurements/guardbands also account for clock skew as is standard for timing analysis, larger guardbands for designs with worse skew.

While more accurate timing measurements are always beneficial, the design is robust to less accurate measurements as well as fine grained dynamic variation (e.g., due to V/T fluctuation), thanks to the timing safeguard mechanism discussed in Section 4.3 and evaluated in Tables 2(a), 2(b). A first-cut design can employ a larger TS window, at the cost of losing a fraction of TNT's benefits. High-frequency industry standard methods of critical path monitoring [23, 50, 51] can also be employed if required. Further, coarse grained effects like DVFS and power state changes can be addressed by dynamically updating the tables when these occur. The corresponding timing can be characterized ahead of time for all DVFS states.

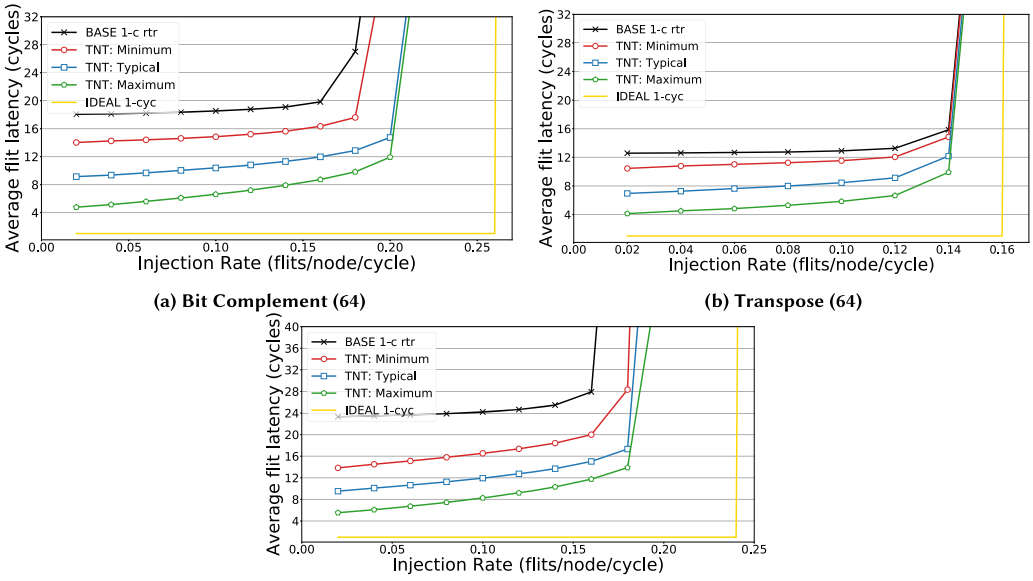
Timing Verification: Our design verification requirements do not violate our goal of design modularity. Specifically, TNT *does not require timing closure over multi-cycle multi-hop routes*. Timing measurements and their verification are only required on a per-link basis i.e., the timing information which is stored in the per-router lookups. TNT then accumulates these delays over the packet routes and makes timing-aware routing decisions, avoiding potent timing concerns.In a way, our timing accumulation and safeguard mechanisms perform top-level **STA (static timing analysis)** on the fly and avoid actions that are potentially timing concerns.

In CAD/EDA domain, multiple advancements have allowed for efficient timing closure, including enabling aggressive timing guardbands [18, 38, 46, 68, 70], support for multi-cycle paths [19, 61, 69, 72], and path based timing margins [68].

In the specific context of TNT, IR-ATA [68] significantly reduces timing guardband pessimism by generating path-based timing margins across an ASIC. Each timing path is margined specific to its unique topology and noise characteristics, removing the requirement for global worst-case margins. This allows large portions of previously unusable timing slack in the majority of timing paths to be utilized towards a variety of goals. This method of timing analysis is well suited to TNT's use of heterogeneous per-link timing delays. Also, CAD tools already use separate PR passes for chip wide interconnects like clock and reset. It is plausible that similar passes can be implemented for the TNT NOC paths if required.

6 EVALUATION

Evaluations discussed are at 1GHz and assume an 8mm per cycle wire traversal capability - a conservative assumption at 1GHz (Figure 3). This NOC frequency range is in line with designs built by industry and academia [10, 22, 32, 58, 65]. Benefits at higher frequencies would vary with wire traversal capability, as discussed in Section 2. Gains would be higher with more conservative



(a) Bit Complement (64)

(b) Transpose (64)

(c) Uniform Random (256)

Fig. 8. Flit latency analysis for TNT.

variation guardbands and for smaller tile sizes. We use a traditional mesh as baseline. TNT results are shown for three scenarios:

① *Min* shows the "minimum" benefits of TNT over the baseline. This is achieved on a floorplan wherein all tiles are homogeneous and 8mm, with no exploitable variation guardband. Thus $\eta = 1$.

② *Typical* shows the "typical" floorplan illustrated in Figure 2 wherein link lengths range from 1mm to 8mm. Typical variation characteristics leads to a roughly a 20% variation in wire delays across the chip [6]. This leads to $\eta = 0.2 - 1$ range across the chip.

③ *Max* shows "maximum" benefits of TNT, achieved on a homogeneous floorplan wherein all links are 1mm. Further, a maximum exploitable variation guardband (up to 35% variation in delay) is assumed. $\eta = 0.085$ across the chip.

6.1 Synthetic Traffic

First, we analyze TNT features via synthetic traffic. We show results only for Uniform Random, Bit Complement, and Transpose in interest of space. We primarily focus on 1-flit packets to understand benefits without secondary effects.

Latency vs Injection Rate: Figures 8(a) and 8(b) show the average flit latency for increasing injection rates (until saturation) for two different kinds of synthetic traffic, for a 64 node system. TNT performs consistently better than the baseline, achieving latency reductions of up to 25% for "Min", up to 57% for "Typical" and up to 74% for "Max". TNT is also able to achieve throughput equal to or greater than the baseline mesh, even in scenarios like Transpose which is often adversarial for many optimizations under DOR [25]. The yellow line shows ideal one cycle across the network latency until saturation. As the injection rate increases, TNT's benefits decrease and the average flit latency drifts further away from the ideal case. This is because if a router has a buffered flit then the incoming flit will not be able to flow through transparently. But TNT's throughput is always better than the baseline, even just prior to saturation because TNT still enables more hops in a single cycle, whenever possible. This means that buffers free up faster and are relatively less

Table 2. Analysis of Blocked Requests (%)

η/W	5%	10%	20%
0.4-1	0	0	2.8
0.2-0.4	0	4	10.6
0.1-0.2	1.7	4.9	11.4

(a) LR: Window vs η

IR/W	5%	10%	20%
0.01	0.3	0.59	1.18
0.1	2.7	5.5	10.9
0.3	6.3	12.5	25.2

(b) ILT: Window vs Injection

bottle-necked. This can also be interpreted from Little's law: faster service time (i.e., lower latency) leads to improved throughput.

Scaling to a larger mesh: Next, we analyze TNT on a larger 256-node mesh for uniform random traffic in Figure 8(c). TNT is able to achieve latency reductions of up to 76% across the different scenarios, resulting in across-the-chip traversal in as low as 6 cycles. Clearly, TNT is able to scale well with increasing cores - it provides even closer to ideal latencies as the hops for flit traversal increases in larger meshes.

Timing Safeguards: If timing measurements are less precise and/or if the chip experiences higher fine-grained heterogeneity, larger ILT/LR Safeguard windows should be employed. First, in Table 2(a) we analyze different window sizes (W: 5% - 20%) for LR Safeguard and their NOC impact in terms of the % of LRs that are blocked at the routers (over different η). The table shows that a shorter window has no/minimal impact on the number of buffered LRs, across η. With very conservatively large windows (e.g., 20%), buffered LRs can increase by around 10% at low η. Second, Table 2(b) shows how different window sizes for ILT Safeguard impact the % of blocked LRs (over different injection rates). At lower IR, there is negligible increase in LR buffering, irrespective of the window size. At higher IR but smaller window sizes, there is minimal increase in LR buffering. These numbers increase to 25% at very conservative timing windows and near-saturation IR. Thus, reasonably large window sizes can usually be employed if required, without having a significant impact on TNT benefits. Note: this analysis is very pessimistic because it assumes that in the absence of safeguard-based conflict, LRs will necessarily flow through transparently. This is not always the case, especially at high IR and/or low η.

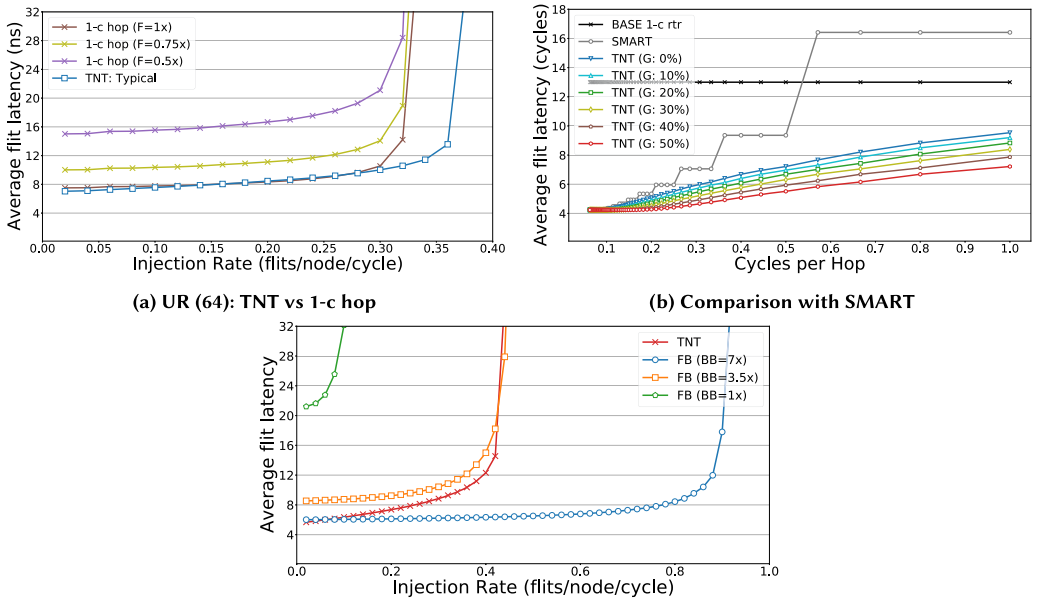
Comparing against an optimistic single cycle design: In Figure 9(a) we compare TNT:Typical against an optimistic design which allows for 1 hop (i.e., router + ST + LT) in a single clock cycle - shown as *"1-c hop"* in the figure. This is meant to resemble a VC-free design which might be adopted when scaling to larger NOCs, as it can reduce router complexity [54, 60, 67, 71]. Our *"1-c hop"* implementation is optimistic because we do not model router overheads.

We model three scenarios of running this design at 1x/0.75x/0.5x of the baseline/TNT frequency, respectively. For fair comparison, reducing the frequency of *"1-c hop"* is intuitive - combining router + ST + LT in a single cycle can significantly impact the clock frequency by reducing it to as low as 0.5x the original. From the figure, it is clear that TNT:Typical outperforms all three *"1-c hop"* scenarios. Despite TNT's longer routing time(Figure 4(b)), TNT's gains from multi-link per cycle traversal outweigh the *"1-c hop"* design's gains from fast routing. Further, as frequency of *"1-c hop"* is realistically reduced, benefits of TNT become far more significant. Note: The TNT approach can be adopted atop 1-cycle per hop designs, but is beyond current scope.

6.2 Comparisons with Prior Work

SMART: SMART [43] is the closest prior work to TNT—it also achieves multi-hop per cycle traversal using lookahead requests. Refer Section 7.2 for description and qualitative comparisons.

Modularity/Complexity: SMART uses a dedicated point-to-point network within a *smart-hop* neighborhood. SMART's router logic complexity, number of links, and number of signalling



(a) UR (64): TNT vs 1-c hop

(b) Comparison with SMART

(c) Comparison with Flattened Butterfly

Fig. 9. Analysis of Uniform Random Traffic on TNT.

requests grow enormously as the wire traversal capability increases (i.e., lower η). SMART's links and arbitration logic complexity grow as $O(\frac{1}{\eta^2})$ due to the dedicated point-point links. Prior work [16] discusses potential SMART wire overheads up to 1000% in large networks. On the other hand, these are constant in TNT. Related to the point-to-point signaling, false negatives occur in SMART when the lookahead signal independently reaches a router but the flit is forced to prematurely stop earlier due to conflicts. Further, note that total conflict-resolution delay for an N-hop request in the LN mesh design is optimally a sequence of N ILT arbitrations on its constant complexity ILT logic. On the other hand, SMART's dedicated point-point signaling network would require $\lceil N * \eta \rceil$ arbitrations on a $O(\frac{1}{\eta^2})$ arbitrator (i.e., arbitration happens only at the end of smart-hops, but there are more requests to arbitrate). While the number of ILT logic computations is higher in the LN mesh, energy/delay analysis shows that the overall TNT cost is far lower, making it the all-round favorable option. Thus, TNT is more scalable and has lower overheads. TNT energy overheads are lower than SMART by almost 50% at lower η.

Latency: In Figure 9(b) we compare the average flit latency of UR traffic for TNT and SMART, on a homogeneous design with an $\eta = 0.1 - 1$ range. The injection rate is fixed at 25% of maximum. In order to mimic heterogeneity (which only TNT can exploit) we plot TNT with different exploitable delay guardbands ranging from 0–50%. Higher guardband implies higher exploitable heterogeneity. TNT and SMART perform similarly at very low η. At high η, SMART performs similar to the baseline or worse - on the other hand, TNT is able to benefit from considerable latency reduction. Overall, TNT can provide 1.6x - 2.3x flit latency reduction over SMART.

Heterogeneity: SMART does not exploit heterogeneity and focuses on NOCs with uniformly high hops-per-cycle.

Flattened Butterfly: We compare TNT against an FB topology [41] for a 256 node system. We use a concentration of 4 for both FB as well as TNT (for fair comparison).

Modularity/Complexity: Each FB router has 14 input ports, 14 output ports + 4 ports for the local nodes [25]. Further, it has dedicated 1-cycle links to every node in both dimensions. We

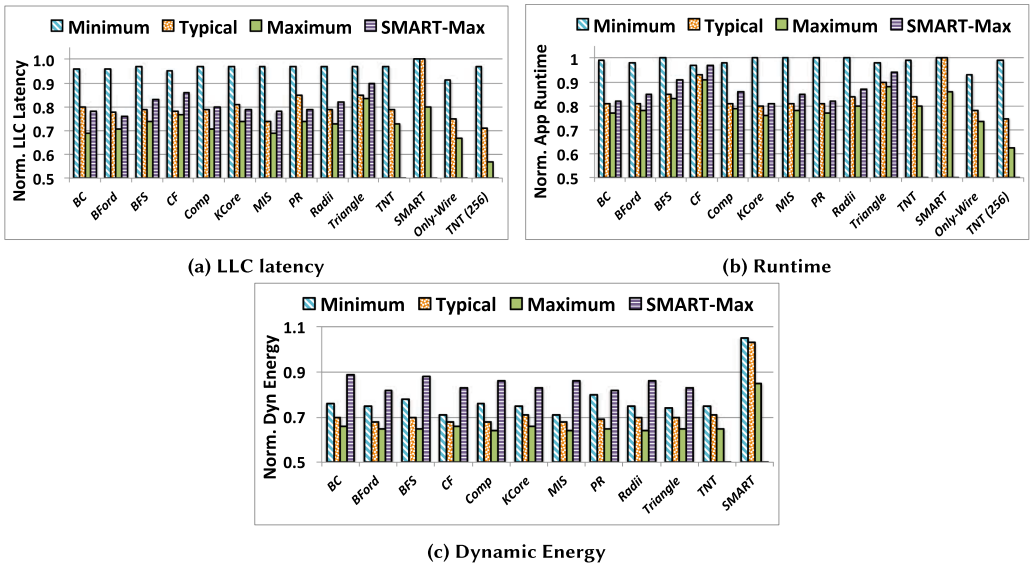


Fig. 10. Full-system analysis of TNT.

assume that the FB router delay is 1-cycle - this is an aggressive assumption, especially since the SA stage needs to perform 18:1 arbitrations (usually assumed to be 4-cycle [36]). TNT, on the other hand, has only mesh-style single I/O ports per direction (i.e., one each for data and LN). Note that complexity of the crossbar scales quadratically with the number of output ports [25] (and concentration) meaning that the FB router's crossbar has roughly 7x the complexity of TNT, at fixed bandwidth/channel. Further, the number of wires in FB scales with the size of the network potentially leading to area and other overheads. While TNT does have a lookahead network which is absent in FB, the LN is considerably lower in overhead, since it is in the style of a mesh and the link widths are only 16 bit (discussed in Section 5). TNT is especially beneficial as we scale to larger networks as concentration is often kept low, due to growing router power and complexity.

Latency: We use 8 VCs per port with virtual cut-through to allow more buffer resources for FB. We analyze on three configurations of FB wherein the Bisection Bandwidth BB (i.e., the total wires) is 1x, 3.5x, and 7x that of TNT. This means that at BB equal to TNT, FB would require 7 flits for each 128-bit packets. Results shown for UR in Figure 9(c). At BB = 1x, FB loses to TNT, both in terms of latency and throughput, due to heavy serialization delay. Here, TNT average flit latency can be as low as 0.25x of FB. At BB = 3.5x, FB is able to match the throughput of TNT but TNT achieves 35–40% lower latency. At BB = 7x, FB matches or performs at lower latency than TNT and has higher potential throughput. To achieve this, FB incurs heavy overheads in terms of area and power. The FB router at BB = 3.5x incurs an area/power overhead of 9x/5x over TNT.

Heterogeneity: FB does not account for physical heterogeneity in NOCs and has to be designed conservatively targeting the worst case link lengths and variation characteristics.

6.3 Full-system Analysis

We run graph applications from Ligra [63], a lightweight graph processing framework that is particularly well suited for graph traversal problems, where only a subset of the vertices are processed at once. Ligra is designed for shared memory systems—the largest publicly available real-world graphs all fit in shared memory. Processing them using Ligra can give performance improvements of up to orders of magnitude compared to distributed-memory graph processing [63]. We model

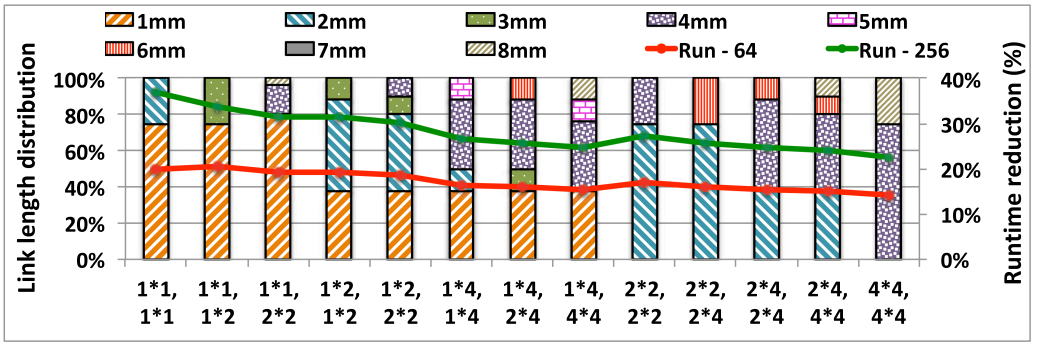


Fig. 11. Design space exploration.

Ligra's shared memory communication as multi-flit packets. We evaluate on accompanying input graphs, specified in PBBS [64] format. Applications are compiled for RISC-V ISA and run with 64/256 threads.

Latency Impact: First we look at the average LLC access latency in Figure 10(a) and analyze the latency reductions from TNT. Results are normalized to the mesh baseline. We also show mean comparisons to SMART and an ideal only-wire delay network, as well as TNT benefits on a 256-node system. TNT's benefits for "Min" scenario are negligible (3%) - this is not unexpected, due to no exploitable wire capability. We see considerable latency reduction with TNT of 21% in "Typical" and 27% in "Max". TNT is within 4% of the ideal Only-Wire case for 64 nodes. SMART is unable to achieve latency reduction in "Min" and "Typical". It is designed for homogeneity in wire traversal capability. Thus, it is forced to be conservative in the "Typical" case (which has some links with no exploitable wire delay). SMART achieves 20% reduction in the homogeneous "Max" scenario. For 256 nodes, TNT achieves a 29% reduction in "Typical" and 43% in "Max".

Performance: Figure 10(b) shows the reduction to application runtime from TNT compared to the baseline, SMART and the ideal only-wire delay network (normalized to the baseline). Results are intuitive based on the LLC access latency reduction discussed above. TNT achieves mean runtime reductions of 1%, 16%, and 20% for the "Min", "Typical", and "Max" scenarios. TNT is within 5-6% of ideal. On the other hand, SMART achieves no reductions in "Min" and "Typical" while 14% in "Max" scenario. Scaling TNT to 256-nodes increases benefits to 2%/25%/38%, respectively.

Network Dynamic Energy: Figure 10(c) measures the total dynamic energy of the network consumed with TNT vs baseline and SMART (normalized to baseline). TNT achieves a mean energy reduction of 24%, 29%, and 35% in the three scenarios. Interesting to note is that TNT achieves considerable energy savings even in the "Min" case due to reduced buffering at intermediate routers, even though there is no significant latency reduction. In comparison, the energy increases in SMART for the first two scenarios, due to no latency reduction but continued buffering, while it reduces by 15% in "Max".

Design Space Exploration: Finally, we sweep through a broad design space in terms of core and memory controller dimensions. For each design, we build (by hand) floorplans and accordingly estimate the distribution of link lengths. The floorplans target a diamond shaped MC distribution as proposed by Abts et al. [1], who show that such a distribution is highly optimal for DOR. It is intuitive that a distribution with a large % of short links would enjoy greater TNT benefits. Figure 11 shows 13 different designs where "X*Y, W*V" design point refers to a core size of $X * Ymm^2$ and an MC of $W * Vmm^2$. The figure also plots the runtime reductions from TNT (vs. baseline) when each of these floorplans are applied to a 64-node system and a 256-node system. This is shown on the secondary axis. Runtime reductions in the range of 12%–20% are

observed for the 64-node system while a broader range of 22%–38% is observed for the 256-node system.

TNT at higher frequencies: In the analysis so far, we have designed for a frequency of 1GHz with a conservative assumption of 8mm per cycle wire traversal capability (our design is able to support 12mm per cycle at low overheads, as shown in Figure 7). At higher frequencies, per-cycle traversal capability will decrease, as discussed in Figure 3. TNT can achieve runtime reduction of 11% at a 2GHz frequency for the 64-node system described above and 17% for the 256-node system (for the "Typical" scenario). The energy reduction for the 64-node system is 21%. It is intuitive that TNT benefits are greater at lower frequencies. However, even at higher operational frequency, the benefits will be greater as core dimensions decrease.

7 RELATED WORK

7.1 State-of-the-art NOCs

Microarchitecture: NOC routers primarily perform the following actions [20]: **Buffer Write (BW)**, **Route Compute (RC)**, **Switch Allocation (SA)**, and **VC Selection (VS)**. Innovations within routers have allowed it to move from serial execution to parallel execution, via lookahead routing [20], simplified VC selection [47], speculative switch arbitration [53, 55], non-speculative switch arbitration via lookaheads [44, 48] to bypass buffering, and so on. Router delay has dropped to 1-cycle in academic prototypes [45, 57] - we use this 1-cycle router as our baseline. Winners of SA proceed to **Switch Traversal (ST)**, and **Link Traversal (LT)**. ST and LT can be performed within a cycle [31, 57] allowing $t_w = 1$. Thus our baseline incurs 2-cycles-per-hop, similar to VC-based designs commonly adopted in industry/academic prototypes, including relatively larger networks [22, 32, 58].

Modular design: It is evident from Equation (1) that packet latency decreases linearly with reductions in hop count (H). Reducing hop count via high-radix routers [20] and pt-to-pt designs is challenging because adding more direct links to distant routers results in an increase in (i) serialization delay due to use of thinner channels (i.e., smaller b), and (ii) router delay t_r and power, due to a higher number of router ports. Such designs (more details in Section 7) can increase complexity and complicate layout, and so on, since multiple point-to-point global wires need to span across the chip, resulting in loss of modularity. Also, loss of design modularity often goes hand in hand with increasing overheads of the design, again impacting scalability. Therefore, industry-standard on-chip networks are often designed with regular topologies (e.g., mesh), short interconnects of fixed length, and self-contained tiles which can be optimized and built modularly using regular repetitive structures, easing the burden of verification, especially as number of on-chip cores increases [7, 11, 21, 52].

7.2 Comparisons to Closest Prior Work: SMART

SMART [43] is the closest prior work to TNT - it also achieves multi-hop per cycle traversal using lookahead requests. SMART provides the illusion of dedicated physical express channels. It embeds asynchronous repeaters within each router's crossbar, and sizes them drive signals up to multiple integral hops (called a *smart-hop*) within a single-cycle. While SMART is a significant step towards achieving ideal T_{wire} delay, it has key differences and limitations compared to TNT in terms of its latency, modularity, and heterogeneity attributes. Qualitatively, these are highlighted in Table 3 (Note: In Table 3, N implies N-hop packet traversal) and discussed below. In Sections 5, 6.2 we provide quantitative comparisons.

Latency: SMART transforms the quantization granularity of network traversal from a traditional 1-router hop to a multi-router hop. The underlying limitation is that quantization still exists, resulting in hurdles to lower latency, especially at higher η. In comparison, TNT does away with all

Table 3. Comparing Attributes: TNT vs SMART

Attribute	SMART	TNT
Speedup Potential	$\frac{2N}{3*\lceil N*\eta_{worst} \rceil}$	$\frac{2N}{2+\lceil N*\eta_{path} \rceil}$
Modularity (Lookahead)	No (pt-pt)	Yes (mesh)
Heterogeneity Support	No	Yes (link delay)

quantization and requests are stalled only due to conflicts, allowing for greater speedup potential as described by the equation in Table 3.

Modularity: SMART uses a dedicated point-to-point network within a *smart-hop* neighborhood for lookahead requests. It sends independent requests across these dedicated links to each path router for every *smart-hop*. This results in a loss of tile-level modularity, as well as increased logic complexity, wire overheads, and arbitration conflicts. In contrast, TNT's lookahead signaling is implemented as a mesh resulting in better scalability and lower overheads.

Heterogeneity: SMART does not exploit heterogeneity in the NOC and is focused on NOCs which uniformly have a low η, i.e., NOCs which are able to allow for multiple hops-per-cycle traversal capability. While TNT performs equivalent to SMART at low η and can significantly outperform at higher η, it is especially effective in scenarios with heterogeneity, as it is able to track wire delay at a per-link granularity.

7.3 Other Works

Asynchronous NOCs: Statically programmed purely asynchronous NOCs have been proposed [8, 13], targeting deterministic traffic. Such a network is programmed statically to preset contention-free routes for QoS. On the other hand, TNT 'reconfigures' paths on every cycle handling general-purpose CMPs with non-deterministic traffic and variable contention. Also, Asynchronous Bypass Channels [35] target chips with multiple clock domains across a die. In contrast, TNT currently targets a single clock domain across the entire die.

Lower hop count topologies: TNT is not restricted to a mesh. Other topologies might reduce hop count in comparison to the mesh, but the hop count still grows as the number of nodes on the chip scale up. For example, the average number of hops on a torus is 0.75x that of the mesh [36] - lower, but still growing with chip size. Moreover, the torus can have unequal link lengths (due to the wrap around links) or have links double the size of the mesh if all links are equally sized [36] - both of which can significantly increase TNT's benefits.Similar to the Flattened Butterfly [41], other high-radix router designs such as Concentrated Mesh [9], Fat Tree [20], Clos [39], and MECS [25] are topology solutions to reduce average hop counts, and advocate adding physical express links between distant routers. These proposals usually suffer from increased router area/power/delay, potentially reduced channel bandwidth, complexity in routing, as well as loss of network modularity.

Reducing router complexity: This work utilizes a baseline 2-cycle per hop with a 1-cycle router, which is state-of-the-art with VC-based router designs. VC-based designs have been commonly adopted in industry and academic prototypes, including relatively larger networks [22, 32, 58]. Other proposals have adopted VC-free designs when scaling to larger sizes, which reduces router complexity and can allow for 1-cycle per hop [54, 60, 67, 71]. We compared TNT against an optimistic VC-based 1-cycle per hop design in Figure 9(a) (Section 6), highlighting TNT's benefits. Note that TNT's transparent flow based approach can be adopted atop 1-cycle per hop designs as well, but is beyond the scope of this work.

TNT Trade-offs: It is evident that TNT has considerable potential to reduce on-chip network latency and energy consumption. Further, the overheads in terms of access energy and area are

minimal. The main challenge stems from leveraging wire delay information and the timing based tracking and safeguard mechanisms, which imply an increase in design verification complexity - our design still limits these complexities by building a modular NOC. While such complexities should not be ignored, and we hope to address them in future work, we believe that proposals like TNT are imperative for next-generation innovation towards both classical and emerging computing domains.

8 CONCLUSION

We are far away from near-ideal latencies in current-day CMPs due to limitations imposed by the inherent traversal quantization of traditional modular NOC designs as well as the physical heterogeneity of real systems. TNT overcomes these challenges - achieving end-to-end network traversal in a single pass, at the best physically capable *bare-wire* delays, while performing only neighbor-to-neighbor control interactions. TNT aggressively exploits wire capability towards near-ideal communication in the exascale era.

REFERENCES

[1] Dennis Abts, Natalie D. Enright Jerger, John Kim, Dan Gibson, and Mikko H. Lipasti. 2009. Achieving predictable performance through better memory controller placement in many-core CMPs. In *Proceedings of the 36th Annual International Symposium on Computer Architecture* (Austin, TX, USA) *(ISCA'09)*. Association for Computing Machinery, New York, NY, USA, 451–461. https://doi.org/10.1145/1555754.1555810

[2] N. Agarwal, T. Krishna, L. Peh, and N. K. Jha. 2009. GARNET: A detailed on-chip network model inside a full-system simulator. In *2009 IEEE International Symposium on Performance Analysis of Systems and Software*. 33–42. https://doi.org/10.1109/ISPASS.2009.4919636

[3] F. Alazemi, A. AziziMazreah, B. Bose, and L. Chen. 2018. Routerless network-on-chip. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 492–503. https://doi.org/10.1109/HPCA.2018.00049

[4] AMD. 2022. Zen - Microarchitectures - AMD. https://en.wikichip.org/wiki/amd/microarchitectures/zen.

[5] Angstrom. 2022. Angstrom. http://projects.csail.mit.edu/angstrom.

[6] A. Ansari, A. Mishra, J. Xu, and J. Torrellas. 2014. Tangle: Route-oriented dynamic voltage minimization for variation-afflicted, energy-efficient on-chip networks. In *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*. 440–451. https://doi.org/10.1109/HPCA.2014.6835953

[7] ARM. 2022. ARM core link interconnect. https://www.arm.com/products/system-ip/corelink-interconnect.

[8] J. Bainbridge and S. Furber. 2002. Chain: A delay-insensitive chip area interconnect. *IEEE Micro* 22, 5 (Sep. 2002), 16–23. https://doi.org/10.1109/MM.2002.1044296

[9] James Balfour and William J. Dally. 2014. Design tradeoffs for tiled CMP on-chip networks. In *ACM International Conference on Supercomputing 25th Anniversary Volume* (Munich, Germany). ACM, New York, NY, USA, 390–401. https://doi.org/10.1145/2591635.2667187

[10] Jonathan Balkind, Michael McKeown, Yaosheng Fu, Tri Nguyen, Yanqi Zhou, Alexey Lavrov, Mohammad Shahrad, Adi Fuchs, Samuel Payne, Xiaohua Liang, Matthew Matl, and David Wentzlaff. 2016. OpenPiton: An open source manycore research framework. *SIGPLAN Not.* 51, 4 (March 2016), 217–232. https://doi.org/10.1145/2954679.2872414

[11] L. Benini and G. De Micheli. 2002. Networks on chips: A new SoC paradigm. *Computer* 35, 1 (Jan. 2002), 70–78. https://doi.org/10.1109/2.976921

[12] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. 2011. The Gem5 simulator. *SIGARCH Comput. Archit. News* 39, 2 (Aug. 2011), 1–7. https://doi.org/10.1145/2024716.2024718

[13] T. Bjerregaard and J. Sparso. 2005. A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip. In *Design, Automation and Test in Europe*, Vol. 2. 1226–1231. https://doi.org/10.1109/DATE.2005.36

[14] B. Bohnenstiehl, A. Stillmaker, J. J. Pimentel, T. Andreas, B. Liu, A. T. Tran, E. Adeagbo, and B. M. Baas. 2017. KiloCore: A 32-nm 1000-processor computational array. *IEEE Journal of Solid-State Circuits* 52, 4 (April 2017), 891–902. https://doi.org/10.1109/JSSC.2016.2638459

[15] K. A. Bowman, S. G. Duvall, and J. D. Meindl. 2002. Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration. *IEEE Journal of Solid-State Circuits* 37, 2 (Feb. 2002), 183–190. https://doi.org/10.1109/4.982424

[16] X. Chen and N. K. Jha. 2016. Reducing wire and energy overheads of the SMART NoC using a setup request network. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 24, 10 (Oct. 2016), 3013–3026. https://doi.org/10.1109/TVLSI.2016.2538284

[17] H. Cherupalli, R. Kumar, and J. Sartori. 2016. Exploiting dynamic timing slack for energy efficiency in ultra-low-power embedded systems. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. 671–681.

[18] H. Cherupalli and J. Sartori. 2015. Graph-based dynamic analysis: Efficient characterization of dynamic timing and activity distributions. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 729–735.

[19] Jason Cong, Yiping Fan, Xun Yang, and Zhiru Zhang. 2003. Architecture and synthesis for multi-cycle communication. In *Proceedings of the 2003 International Symposium on Physical Design* (Monterey, CA, USA) *(ISPD'03)*. Association for Computing Machinery, New York, NY, USA, 190–196. https://doi.org/10.1145/640000.640040

[20] William Dally and Brian Towles. 2003. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[21] W. J. Dally and B. Towles. 2001. Route packets, not wires: On-chip interconnection networks. In *Proceedings of the 38th Design Automation Conference (IEEE Cat. No.01CH37232)*. 684–689. https://doi.org/10.1109/DAC.2001.156225

[22] B. K. Daya, C. O. Chen, S. Subramanian, W. Kwon, S. Park, T. Krishna, J. Holt, A. P. Chandrakasan, and L. Peh. 2014. SCORPIO: A 36-core research chip demonstrating snoopy coherence on a scalable mesh NoC with in-network ordering. In *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*. 25–36.

[23] A. J. Drake, M. S. Floyd, R. L. Willaman, D. J. Hathaway, J. Hernandez, C. Soja, M. D. Tiner, G. D. Carpenter, and R. M. Senger. 2013. Single-cycle, pulse-shaped critical path monitor in the POWER7+ microprocessor. In *International Symposium on Low Power Electronics and Design (ISLPED)*. 193–198.

[24] Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, and Babak Falsafi. 2012. Clearing the clouds: A study of emerging scale-out workloads on modern hardware. In *Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems* (London, England, UK) *(ASPLOS XVII)*. ACM, New York, NY, USA, 37–48. https://doi.org/10.1145/2150976.2150982

[25] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu. 2009. Express cube topologies for on-chip interconnects. In *2009 IEEE 15th International Symposium on High Performance Computer Architecture*. 163–174. https://doi.org/10.1109/HPCA.2009.4798251

[26] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu. 2011. Kilo-NOC: A heterogeneous network-on-chip architecture for scalability and service guarantees. In *2011 38th Annual International Symposium on Computer Architecture (ISCA)*. 401–412.

[27] Meeta Gupta, Jude A. Rivers, Pradip Bose, Gu-Yeon Wei, and David Brooks. 2009. Tribeca: Design for PVT variations with local recovery and fine-grained adaptation. In *MICRO*.

[28] Nikos Hardavellas, Michael Ferdman, Babak Falsafi, and Anastasia Ailamaki. 2009. Reactive NUCA: Near-optimal block placement and replication in distributed caches. In *Proceedings of the 36th Annual International Symposium on Computer Architecture* (Austin, TX, USA) *(ISCA'09)*. ACM, New York, NY, USA, 184–195. https://doi.org/10.1145/1555754.1555779

[29] Nikos Hardavellas, Ippokratis Pandis, Ryan Johnson, Naju Mancheril, Anastassia Ailamaki, and Babak Falsafi. 2007. Database servers on chip multiprocessors: Limitations and opportunities. In *Proceedings of the Biennial Conference on Innovative Data Systems Research*.

[30] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar. 2007. A 5-GHz mesh interconnect for a teraflops processor. *IEEE Micro* 27, 5 (Sep. 2007), 51–61. https://doi.org/10.1109/MM.2007.4378783

[31] J. Howard, S. Dighe, Y. Hoskote, S. Vangal, D. Finan, G. Ruhl, D. Jenkins, H. Wilson, N. Borkar, G. Schrom, F. Pailet, S. Jain, T. Jacob, S. Yada, S. Marella, P. Salihundam, V. Erraguntla, M. Konow, M. Riepen, G. Droege, J. Lindemann, M. Gries, T. Apel, K. Henriss, T. Lund-Larsen, S. Steibl, S. Borkar, V. De, R. V. D. Wijngaart, and T. Mattson. 2010. A 48-Core IA-32 message-passing processor with DVFS in 45nm CMOS. In *2010 IEEE International Solid-State Circuits Conference - (ISSCC)*. 108–109. https://doi.org/10.1109/ISSCC.2010.5434077

[32] J. Howard, S. Dighe, S. R. Vangal, G. Ruhl, N. Borkar, S. Jain, V. Erraguntla, M. Konow, M. Riepen, M. Gries, G. Droege, T. Lund-Larsen, S. Steibl, S. Borkar, V. K. De, and R. Van Der Wijngaart. 2011. A 48-core IA-32 processor in 45 nm CMOS using on-die message-passing and DVFS for performance and power scaling. *IEEE Journal of Solid-State Circuits* 46, 1 (Jan. 2011), 173–183. https://doi.org/10.1109/JSSC.2010.2079450

[33] Intel. 2022. Skylake - Microarchitectures - Intel. https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(server).

[34] Intel. 2022. Sunny Cove - Microarchitectures - Intel. https://en.wikichip.org/wiki/intel/microarchitectures/sunny_cove.

[35] T. N. K. Jain, P. V. Gratz, A. Sprintson, and G. Choi. 2010. Asynchronous bypass channels: Improving performance for multi-synchronous NoCs. In *2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip*. 51–58. https://doi.org/10.1109/NOCS.2010.15

[36] Natalie Enright Jerger, Tushar Krishna, and Li-Shiuan Peh. 2017. On-chip networks. *Synthesis Lectures on Computer Architecture* 12, 3 (2017), 1–210.

[37] D. Johnson, M. Johnson, J. Kelm, W. Tuohy, S. Lumetta, and S. Patel. 2011. Rigel: A 1,024-core single-chip accelerator architecture. *IEEE Micro* 31, 4 (July 2011), 30–41. https://doi.org/10.1109/MM.2011.40

[38] J. Jung and T. Kim. 2012. Variation-aware false path analysis based on statistical dynamic timing analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31, 11 (2012), 1684–1697.

[39] Y. Kao, M. Yang, N. S. Artan, and H. J. Chao. 2011. CNoC: High-radix Clos network-on-chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 30, 12 (Dec. 2011), 1897–1910. https://doi.org/10.1109/TCAD.2011.2164538

[40] John Kim. 2009. Low-cost router microarchitecture for on-chip networks. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture* (New York, New York) *(MICRO 42)*. Association for Computing Machinery, New York, NY, USA, 255–266. https://doi.org/10.1145/1669112.1669145

[41] J. Kim, J. Balfour, and W. Dally. 2007. Flattened butterfly topology for on-chip networks. In *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007)*. 172–182. https://doi.org/10.1109/MICRO.2007.29

[42] Tushar Krishna. 2014. *Enabling Dedicated Single-cycle Connections over a Shared Network-on-chip*. Ph. D. Dissertation. MIT.

[43] T. Krishna, C. O. Chen, W. C. Kwon, and L. Peh. 2013. Breaking the on-chip latency barrier using SMART. In *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*. 378–389. https://doi.org/10.1109/HPCA.2013.6522334

[44] T. Krishna, A. Kumar, L. Peh, J. Postman, P. Chiang, and M. Erez. 2009. Express virtual channels with capacitively driven global links. *IEEE Micro* 29, 4 (July 2009), 48–61. https://doi.org/10.1109/MM.2009.64

[45] T. Krishna, J. Postman, C. Edmonds, L. Peh, and P. Chiang. 2010. SWIFT: A swing-reduced interconnect for a token-based network-on-chip in 90nm CMOS. In *2010 IEEE International Conference on Computer Design*. 439–446. https://doi.org/10.1109/ICCD.2010.5647666

[46] A. Krstic, Yi-Min Jiang, and Kwang-Ting Cheng. 2001. Pattern generation for delay testing and dynamic timing analysis considering power-supply noise effects. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 20, 3 (2001), 416–425.

[47] A. Kumar, P. Kundu, A. P. Singh, L. Pehy, and N. K. Jha. 2007. A 4.6Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS. In *2007 25th International Conference on Computer Design*. 63–70. https://doi.org/10.1109/ICCD.2007.4601881

[48] A. Kumar, L. Peh, and N. K. Jha. 2008. Token flow control. In *2008 41st IEEE/ACM International Symposium on Microarchitecture*. 342–353. https://doi.org/10.1109/MICRO.2008.4771803

[49] G. Kurian, J. E. Miller, J. Psota, J. Eastep, J. Liu, J. Michel, L. C. Kimerling, and A. Agarwal. 2010. ATAC: A 1000-core cache-coherent processor with on-chip optical network. In *2010 19th International Conference on Parallel Architectures and Compilation Techniques (PACT)*. 477–488.

[50] C. R. Lefurgy, A. J. Drake, M. S. Floyd, M. S. Allen-Ware, B. Brock, J. A. Tierno, and J. B. Carter. 2011. Active management of timing guardband to save energy in POWER7. In *2011 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 1–11.

[51] C. R. Lefurgy, A. J. Drake, M. S. Floyd, M. S. Allen-Ware, B. Brock, J. A. Tierno, J. B. Carter, and R. W. Berry. 2013. Active guardband management in power7+ to save energy and maintain reliability. *IEEE Micro* 33, 4 (2013), 35–45.

[52] Li-Shiuan Peh and W. J. Dally. 2000. Flit-reservation flow control. In *Proceedings Sixth International Symposium on High-Performance Computer Architecture. HPCA-6 (Cat. No.PR00550)*. 73–84. https://doi.org/10.1109/HPCA.2000.824340

[53] H. Matsutani, M. Koibuchi, H. Amano, and T. Yoshinaga. 2009. Prediction router: Yet another low latency on-chip router architecture. In *2009 IEEE 15th International Symposium on High Performance Computer Architecture*. 367–378. https://doi.org/10.1109/HPCA.2009.4798274

[54] M. McKeown, A. Lavrov, M. Shahrad, P. J. Jackson, Y. Fu, J. Balkind, T. M. Nguyen, K. Lim, Y. Zhou, and D. Wentzlaff. 2018. Power and energy characterization of an open source 25-core manycore processor. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 762–775.

[55] Robert Mullins, Andrew West, and Simon Moore. 2004. Low-latency virtual-channel routers for on-chip networks. In *Proceedings of the 31st Annual International Symposium on Computer Architecture* (Munchen, Germany) *(ISCA'04)*. IEEE Computer Society, Washington, DC, USA, 188–. http://dl.acm.org/citation.cfm?id=998680.1006717.

[56] D. Pamunuwa, J. Öberg, L. R. Zheng, M. Millberg, A. Jantsch, and H. Tenhunen. 2003. Layout, performance and power trade-offs in mesh-based network-on-chip architectures. In *Proc. of the 12th IFIP International Conference on Very Large Scale Integration (VLSI-SoC 2003)*. Citeseer.

[57] S. Park, T. Krishna, C. Chen, B. Daya, A. Chandrakasan, and L. Peh. 2012. Approaching the theoretical limits of a mesh NoC with a 16-node chip prototype in 45nm SOI. In *DAC Design Automation Conference 2012*. 398–405.

[58] Sunghyun Park, Tushar Krishna, Chia-Hsin Chen, Bhavya Daya, Anantha Chandrakasan, and Li-Shiuan Peh. 2012. Approaching the theoretical limits of a mesh NoC with a 16-node chip prototype in 45nm SOI. In *Proceedings of the 49th Annual Design Automation Conference* (San Francisco, California) *(DAC'12)*. Association for Computing Machinery, New York, NY, USA, 398–405. https://doi.org/10.1145/2228360.2228431

[59] Gokul Subramanian Ravi and Mikko H. Lipasti. 2019. Recycling data slack in out-of-order cores. *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)* (2019), 545–557.

[60] A. Rovinski, C. Zhao, K. Al-Hawaj, P. Gao, S. Xie, C. Torng, S. Davidson, A. Amarnath, L. Vega, B. Veluri, A. Rao, T. Ajayi, J. Puscar, S. Dai, R. Zhao, D. Richmond, Z. Zhang, I. Galton, C. Batten, M. B. Taylor, and R. G. Dreslinski. 2019. A 1.4 GHz 695 giga risc-V Inst/s 496-core manycore processor with mesh on-chip network and an all-digital synthesized PLL in 16nm CMOS. In *2019 Symposium on VLSI Circuits*. C30–C31.

[61] S. Ryu, J. Koo, and J. Kim. 2017. Low design overhead timing error correction scheme for elastic clock methodology. In *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. 1–6.

[62] S. R. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas. 2008. VARIUS: A model of process variation and resulting timing errors for microarchitects. *IEEE Transactions on Semiconductor Manufacturing* (2008). https://doi.org/10.1109/TSM.2007.913186

[63] Julian Shun and Guy E. Blelloch. 2013. Ligra: A lightweight graph processing framework for shared memory. In *Proceedings of the 18th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (Shenzhen, China) *(PPoPP'13)*. ACM, New York, NY, USA, 135–146. https://doi.org/10.1145/2442516.2442530

[64] Julian Shun, Guy E. Blelloch, Jeremy T. Fineman, Phillip B. Gibbons, Aapo Kyrola, Harsha Vardhan Simhadri, and Kanat Tangwongsan. 2012. Brief announcement: The problem based benchmark suite. In *Proceedings of the Twenty-fourth Annual ACM Symposium on Parallelism in Algorithms and Architectures*. 68–70.

[65] A. Sodani. 2015. Knights landing (KNL): 2nd generation Intel® Xeon Phi processor. In *2015 IEEE Hot Chips 27 Symposium (HCS)*. 1–24.

[66] C. Sun, C. O. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L. Peh, and V. Stojanovic. 2012. DSENT - A tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling. In *2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*. 201–210. https://doi.org/10.1109/NOCS.2012.31

[67] M. D. Taylor, W. Lee, S. P. Amarasinghe, and A. Agarwal. 2005. Scalar operand networks. *IEEE Transactions on Parallel and Distributed Systems* 16, 2 (2005), 145–162.

[68] Ashkan Vakil, Houman Homayoun, and Avesta Sasan. 2019. IR-ATA: IR annotated timing analysis, a flow for closing the loop between PDN design, IR analysis and timing closure. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference* (Tokyo, Japan) *(ASPDAC'19)*. Association for Computing Machinery, New York, NY, USA, 152–159. https://doi.org/10.1145/3287624.3287683

[69] V. Vorisek, B. Swanson, Kun-Han Tsai, and D. Goswami. 2006. Improved handling of false and multicycle paths in ATPG. In *24th IEEE VLSI Test Symposium*. 6 pp.–165.

[70] M. Wagner and H. Wunderlich. 2013. Efficient variation-aware statistical dynamic timing analysis for delay test applications. In *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*. 276–281.

[71] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C. Miao, J. F. Brown III, and A. Agarwal. 2007. On-chip interconnection architecture of the tile processor. *IEEE Micro* 27, 5 (Sep. 2007), 15–31. https://doi.org/10.1109/MM.2007.4378780

[72] H. Zheng, S. T. Gurumani, L. Yang, D. Chen, and K. Rupnow. 2013. High-level synthesis with behavioral level multicycle path analysis. In *2013 23rd International Conference on Field Programmable Logic and Applications*. 1–8.

Received 19 February 2022; revised 18 February 2023; accepted 24 April 2023