Kite: A Family of Heterogeneous Interposer Topologies Enabled via Accurate Interconnect Modeling

Srikant Bharadwaj
Advanced Micro Devices, Inc.
Georgia Institute of Technology
srikant.bharadwaj@amd.com

Jieming Yin
Advanced Micro Devices, Inc.
jieming.yin@amd.com

Bradford Beckmann
Advanced Micro Devices, Inc.
brad.beckmann@amd.com

Tushar Krishna
Georgia Institute of Technology
tushar@ece.gatech.edu

***Abstract*—Recent advances in die-stacking and 2.5D chip integration technologies introduce in-package network heterogeneities that can complicate the interconnect design. Integrating chiplets over a silicon interposer offers new opportunities of optimizing interposer topologies. However, limited by the capability of existing network-on-chip (NoC) simulators, the full potential of the interposer-based NoCs has not been exploited. In this paper, we address the shortfalls of prior NoC designs and present a new family of chiplet topologies called Kite. Kite topologies better utilize the diverse networking and frequency domains existing in new interposer systems and outperform the prior chiplet topology proposals. Kite decreased synthetic traffic latency by 7% and improved the maximum throughput by 17% on average versus Double Butterfly and Butter Donut, two previous proposals developed using less accurate modeling.**

I. Introduction

The performance and power benefits of technology scaling that propelled the semiconductor industry for decades have been diminishing since Dennard's scaling ended and Moore's Law slowed down. However, the demand for computing continues to grow at an increasing rate. Thus, major chip vendors such as AMD, Intel [1], NVIDIA, and others have embraced 2.5D interposer-based integration to scale the performance of individual computation node. One of the key technologies enabled by the 2.5D systems is the integration of multiple discrete chips (often called *chiplets)* within a package either over a silicon interposer or via other packaging technologies [2, 3]. Furthermore, chiplets could also be integrated using Multi-chip Module SoCs (MCM-SoCs). Figure 1(a) shows an example of a 64-core monolithic chip while its chipletized [4] version is shown in Figure 1(b). Chiplet-based architectures not only help address design costs and yield issues, but also enable integration of chips built using heterogeneous technology nodes [2], which is not feasible in a monolithic design. These architectures are becoming common in commercial products [1, 2, 5] and also find heavy support from government programs [6].

In an interposer-based system, chiplets as well as the interposer can have their own interconnect networks, operating at different clock domains and/or link widths. This leads to a heterogeneous Network-on-Chip (NoC) + Network-on-Interposer (NoI) interconnection fabric across the system. Designing the NoI topology can be critical for the overall system performance because the interposer is a shared resource for inter-chiplet communication, off-chip data transfer, and I/O. Previous work have proposed new topologies for hybrid NoC + NoI architectures[1] [4, 7, 8]. However, they assume the NoCs and NoI are operating at the same frequency, which might not be economical or beneficial in commercial products. For an industrial design, it is rather feasible that the chiplets and the interposer have their own clock domains to exploit the maximum potential of chiplet-interposer integration. Therefore, we revisit the NoI topology design under more realistic assumptions and constraints.

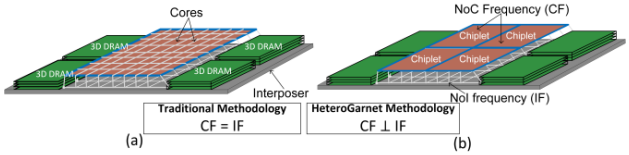

Figure 1: (a) A 64-core monolithic chip connected to 3D DRAM (b) 4x16-core chiplet system showing the heterogeneous fragments of NoC + NoI.

The computer architecture design and research community heavily relies on simulators to facilitate exploration and evaluation of promising designs. An important design aspect that traditional NoC simulators lack is decoupling the NoI from the core and memory chiplets. As a result, most existing tools assume NoI frequency and link width equal to NoC frequency and link width. As we show in this paper, such a methodology restricted designers and researchers from evaluating the trade-off of topology designs [4, 7]. Moreover, insufficient modeling of the NoC + NoI interactions loses out on a wide plethora of design options. To fully explore the NoC + NoI design space, accurate modeling of the heterogeneous interconnect is necessary.

In this paper, we accurately re-evaluate and improve upon prior academic approaches by designing a new set of topologies, called *Kite*, which outperform the current published state-of-the-art topologies in terms of both core-to-core and core-to-memory traffic. Specifically, by decoupling the NoI from the NoC frequency and power constraints, we accurately evaluate the trade-off in using longer links within the NoI. We adopt implementation-aware design-time metrics, called *effective hop count* and *effective bandwidth*, which account for the frequency constraints of different link lengths and lead to improved topology designs. As a result, our Kite topologies use longer links more efficiently and sustain higher throughputs than what has been previously concluded. Further, we introduce HeteroGarnet, a simulator for modeling heterogenous networks and utilize it to measure performance of NoI topologies

II. Background and Motivation

In this section, we first provide background and discuss previous work in the field of interposer-based systems. We then motivate the need for a tool to accurately model such systems.

A. Silicon Interposer Systems

Breaking large monolithic chips into smaller *chiplets* improves the collective yield and reduces system cost [4]. In such disintegrated chiplet-based systems, chiplets are integrated via the interposer. Therefore, the NoI carries both memory traffic and coherency traffic

[1] These works did not use the term NoI, instead referring to the chip and interposer networks collectively as "hybrid NoC". In this work, we use the terms NoC and NoI to distinguish between the networks on individual chiplets and between chiplets on the interposer, respectively.

978-1-7281-1085-1/20/$31.00 ©2020 IEEE

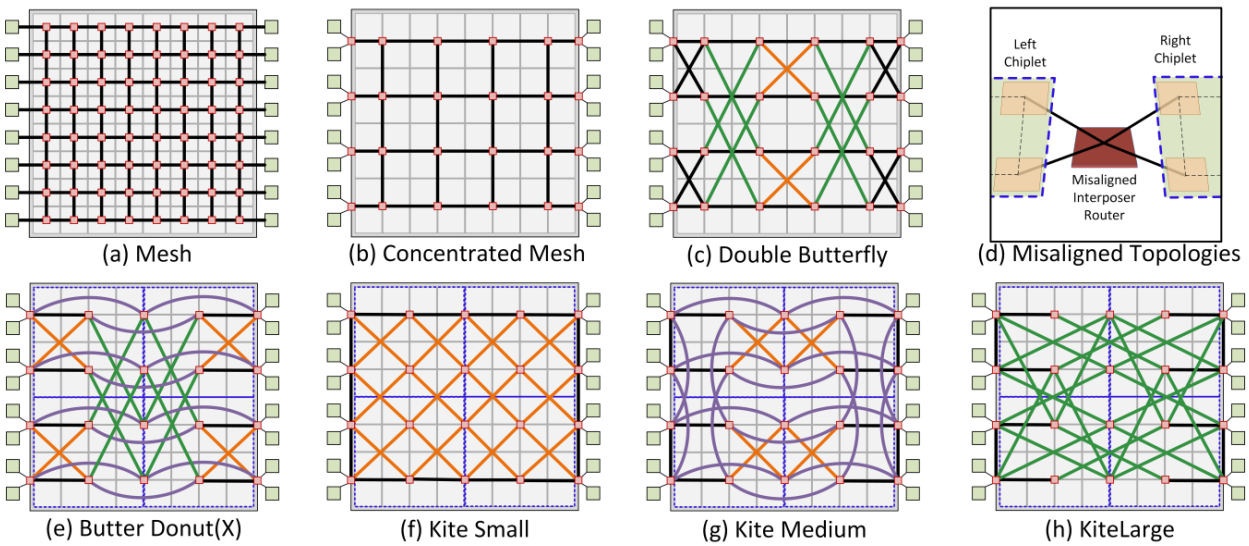

Figure 2: The target architecture includes 64 cores (small grey boxes) arranged within 4x4 chiplets (blue dotted lines). The red boxes denote interposer routers which connect to the green memory controllers on the side. Network-on-Interposer topologies considered for exploration: (a) Mesh, (b) Concentrated Mesh, (c) Double Butterfly [7], and (e) Butter Donut [4] are shown here. (d) Shows a misaligned [4] interposer router connected to routers from chiplet. (f) Kite Small, (g) Kite Medium, and (h) Kite Large each use a different link-length as the longest link as shown by the colored lines.

between chiplets. Chiplet-based systems introduce many research questions around the right chiplet boundaries and topologies connecting the chiplets. In addition, the uses of silicon interposers also lead to new opportunities in designing modern systems. An earlier study showcased using the interposer to route memory traffic more efficiently by creating a *hybrid NoC + NoI*. Taking advantage of the interposer resource, prior work further proposed to create a network within the interposer. This NoI is then connected to cores using µbumps to facilitate communication. The interposer network has been proposed to be used for both core-to-memory traffic as well as core-to-core coherency traffic as described below [4, 8].

B. Baseline NoI topologies

Several approaches to build the NoI topology have been discussed and evaluated in previous works [4, 7]. A simple topology is to connect a core-level router to an interposer router forming a simple mesh, as shown in Figure 2(a). However, this design consumes large area and most of the links are left under-utilized. A natural alternative is to use a concentrated-mesh network where multiple cores are connected to a router in the interposer, as shown in Figure 2(b). The concentrated mesh (CMesh) takes advantage of high-radix routers (degree 8) to connect to memory controllers in lesser number of network hops. Compared to mesh, CMesh delivers lower throughput because of lower bisection bandwidth. Double Butterfly [Figure 2(c)] takes advantage of *longer links* and reduces average hop count; in addition, diagonal links increase the bisection bandwidth [7]. A later study optimized the NoI topology by utilizing a "misaligned" network. A misaligned interposer network offsets the location of its routers, such that cores on the edge of two adjacent chiplets share the same router [4], as shown in Figure 2(d). They further proposed the ButterDonut topology to reduce the average hop count as well as inter-chiplet coherency traffic latency, as shown in Figure 2(e). ButterDonut increases the bisection bandwidth while keeping the router complexity similar to CMesh. We follow the nomenclature from the original paper [4] for misaligned topologies where ButterDonut(X) refers to misalignment in the X-dimension.

C. Challenges of hybrid NoC exploration[2]

The hybrid NoC + NoI architectures pose new design challenges including routing, composability, yield, thermal, and more. In this work, we focus on practical NoI topology design constraints.

The decoupled nature of NoC + NoI provides us opportunities to further optimize the NoI topology. On the other hand, it introduces new challenges: evaluation of design topologies should include modeling the operating parameters accurately, such as latency and operating frequency.

Moreover, given the comparatively low cost of adding more wires [7], a natural direction is to explore the benefit of increasing the NoI link bandwidth without affecting the NoC design parameters. An ideal infrastructure should have the flexibility of configuring the clock domain and link/flit width for each individual network (NoC or NoI) element (router, link, etc.).

III. Kite Topologies

We propose a novel set of topologies, called *Kite*, which efficiently utilizes the links in the network. We first describe the general topology design metrics, and then describe our proposed methodology while designing Kite topologies.

A. Design Metric: Effective Hops

The network latency (T) of a packet traversing between any source-destination pair is defined as [9]:

$$T = H \times t_w + H \times t_r + T_c + T_s$$

H is the number of hops between the source-destination pair, t_w is per-hop wire delay, t_r is router delay, T_c and T_s are contention and serialization delay, respectively. All these delays are in *cycles*. t_w, t_r, and T_s are design-time metrics, while H and T_c depend on traffic characteristics at runtime. Since the network topology is fixed at design-time, it needs to accommodate myriad traffic patterns. A design-time proxy for latency, independent of runtime traffic contention, is *average hop count* (H_{avg}). H_{avg} is determined by averaging H over all possible source-destination pairs and is usually combined with other metrics such as diameter (proxy for worst-case

[2] While this paper focuses on a hybrid NoC + NoIs using a silicon interposer, most of these arguments are true for modern many-core systems as well.

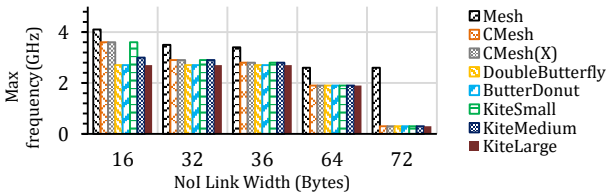

Figure 3: Maximum Operating Frequency, for NoI topologies with different link widths at 22nm technology.

latency) and bisection bandwidth (proxy for peak throughput) to measure the effectiveness of a topology [4, 7]. Reducing H_{avg} is the gold standard for topology design–and has led to multiple popular express-link based on-chip (e.g., Flattened Butterfly [10]), on-interposer (e.g., ButterDonut [4]), and HPC (e.g., SlimFly [11]) topologies.

The catch in H_{avg}, however, is that the operating frequency f (time duration of each hop) is often not taken into account. For example, a topology could use long express links to provide single hop transmission, but could be restricted by the maximum possible f. Hence the t_w and t_r values would go up, making it worse than a topology with higher H_{avg} but higher f. This is often not an issue in NoCs since the (low) operating frequency of the NoC is limited by CPU/GPU cores and caches, which have plateued since Dennard's scaling stopped. However, NoIs need not be coupled to core frequencies, and could thus operate at a different and (multi-GHz) operating frequency. This necessitates co-optimizing H_{avg} and the operating frequency f. To this end, we define a metric called effective hop count (H_{eff}) as the proxy for designing our proposed NoI topologies.

$$Eff.Hop\ Count\ (H_{eff}) = \frac{Average\ Hop\ Count\ (H_{avg})}{Operating\ Frequency\ (f)}$$

H_{eff} measures the effective time taken to traverse H_{avg} hops within the network. Extending this implementation specific metric idea to bisection bandwidth leads us to an effective bisection bandwidth metric defined as:

$$Eff.Bisection\ BW = Bisection\ BW \times Operating\ Frequency(f)$$

Two topologies that have the same H_{avg} and Bisection BW may lead to vastly different operatable frequencies (and thus H_{eff} and *Eff. BW*) as we show in this work.

B. Classification of Links: k-straight *and* k-m-diagonal

Taking cue from express-link based topologies, we identify two classes of links that we call k-straight and k-m-diagonal. *k*-straight links directly connect two routers along the same dimension (horizontal or vertical), physically skipping or bypassing k-1 intermediate routers. For example, 1-straight links are nearest neighbor links, 2-straight links connect routers skipping one in the middle, as shown in Figure 2 using purple lines. *k-m*-diagonal links follow the same rule, with *k* hops in one direction and *m* in another. The orange and green links in Figure 2 show 1-1-diagonal and 2-1-diagonal respectively. We assume that the longest class of links in the topology limit and determine the overall operating frequency of the NoI.

C. Kite Topology Construction

We now create new topologies to maximize H_{eff}. This is done by maximizing the usage of long links within the topology. It is important to note that we combine H_{eff} with other known metrics such as bisection bandwidth and diameter to quickly compare each topology and pick the optimal one.

For the purposes of illustration, we assume a system with 16 memory controllers, 4 CPU/GPU chiplets (with 16 cores each), a concentration factor of 4 at each router, and a maximum router degree of 8. The Kite topologies adopt the misaligned NoI topology [4] to reduce the hop count required for inter-chiplet traffic.

Topology	Topology Specific						Technology Specific		Impl. Specific	
	Routers	Links	Diameter	Avg Hop Count H_{avg}	Bisection BW	Router Deg.	Longest Link (mm)	Max Freq (GHz)	Eff. Hop Count H_{eff} (ns)	Eff. Bisection BW
Mesh	64	112	16	6.12	8	5	2.2	**4.0**	1.49	32.0
CMesh	24	32	8	3.75	4	8	4.4	**3.6**	1.04	14.4
CMesh(X)	**20**	40	7	3.25	4	8	4.4	**3.6**	0.90	14.4
Double Butterfly	24	40	4	2.85	8	8	9.8	**2.7**	1.05	21.6
Butter Donut	**20**	36	4	2.21	**12**	8	9.8	**2.7**	0.81	**32.4**
Kite Small	**20**	38	4	2.39	8	8	6.2	**3.6**	**0.66**	28.8
Kite Medium	**20**	40	4	**2.17**	**12**	8	8.8	**3.0**	**0.72**	**36.0**
Kite Large	**20**	36	**3**	**2.03**	**12**	8	9.8	**2.7**	**0.75**	**32.4**

Table 1: Comparison of NoI topologies: The average effective hop, calculated using the max operating frequency, determines the minimum latency offered by the topology.

All Kite topologies are constructed by picking one of the *k*-straight or *k-m*-diagonal links as the *longest link* to provide connectivity between as many routers as possible. After that, any remaining routers and ports, are connected using shorter links, till the maximum router degree is met. For the purposes of illustration, we demonstrate this policy with three different link lengths – 1-1-diagonal, 2-straight, and 2-1-diagonal, constructing three topologies - *Kite-Small*, *Kite-Medium*, and *Kite-Large* use these as the longest links respectively. The resulting topologies are shown in Figure 2.

Other variants can be constructed by using longer and/or heterogeneous links. Like prior work on express topologies [10], the Kite topologies can be scaled to larger core counts by either increasing the concentration factor at each router, or following the same design rule across larger number of routers, or connecting together multiple copies of the same topology. Without loss of generality, we evaluate Kite-Small, Medium, and Large in this paper.

D. Technology-Specific Evaluation

We use HeteroGarnet (Section IV) and DSENT [12] to support router and link power modeling as detailed later in Section V.B. The maximum operating frequency is determined by the radix of the largest router and length of the longest wire, as shown in Figure 3. Mesh topology has the shortest links and low radix routers and hence it can operate at 4.0 GHz. CMesh can withstand up to 3.6GHz. The topologies with longer links (i.e., Double Butterfly and ButterDonut), however, can only be clocked at a maximum of 2.7 GHz. We consider a 24mm × 36mm [4] active interposer system for our evaluations, which has a lower µbump area overhead [7]. Figure 3 also shows that increasing the link width reduces the maximum operating frequency (i.e., inability to meet faster timing).

E. Kite Topology Analysis

Table 1 summarizes the Kite topologies in comparison to other NoI topologies over several metrics. Topology-specific metrics are combined with technology-specific metrics to determine the final implementation (rightmost column). Compared to state-of-the-art NoI topologies such as DoubleButterfly and ButterDonut(X), we can see that Kite topologies provide the lowest H_{eff} owing to their efficient usage of longest links (which dictate the maximum operational frequency). Even Kite Small, which has a higher average hop count, has a lower effective hop rate of 0.68 compared to the H_{eff} of ButterDonut(X). Table 1 also shows the traditional metric, average

Features	Topaz [24]	BookSim2 [21]	Garnet [23]	SuperSim [25]	Hetero-Garnet
Synthetic Traffic	✓	✓	✓	✓	✓
Full System	✓	✓	✓✓		✓✓
Variable Delay Router		✓		✓	✓
Message-class exclusive links					✓
Heterogeneous Frequency				✓	✓
DVFS Support					✓
Heterogeneous Link Width					✓
On-package Interconnects					✓

Table 3: Summary of features supported by HeteroGarnet compared to other openly available NoC simulation tools.

number of hops required by flits to reach the memory controllers. The state-of-the-art topology ButterDonut(X) takes advantage of the long links to deliver an average memory hops of 2.21. The Kite Medium and Kite Large topologies make full use of the longest links resulting in a low average hop of 2.17 and 2.03, respectively.

While a low network latency is beneficial, topologies should support high bandwidth as well. Table 1 shows the bisection bandwidth of the topologies. While Kite Small fails to offer a good bisection bandwidth. Both Kite Medium and Kite Large offer the same bandwidth as ButterDonut(X).

These results show that even though topologies like CMesh may have a high average hop count, they could perform better when their maximum operating frequencies are considered. While the state-of-the-art topologies have a lower average hop count, they face a disadvantage because their longer links must operate at a lower clock frequency.

While abstract metrics such as hop count and bisection bandwidth provide a proxy for the measuring performance of topology, it has been necessary to evaluate topologies over synthetic and real application traffic to understand the true performance of topology designs. However, evaluating complex heterogeneous topologies such as Kite requires the support of tools.

IV. HeteroGarnet – NoC Simulator[3]

Traditional NoC topologies were largely explored along three dimensions of cost: number of links, number of routers, and number of router ports. These dimensions are insufficient to qualitatively and quantitatively explore the complete space of design possibilities for hybrid NoC + NoIs because of two major differences. First, their physical interconnects can be diverse, where interconnect materials (e.g., on-chip wires, TSVs, µbumps) and widths can be different across the whole system. Second, their system structures can be diverse, where multiple small systems operating at different voltages and/or frequencies are connected to form a larger system on a chip or package. Exploring the design space in such directions needs the support of tools that can model such heterogeneity. This motivated us to develop HeteroGarnet, building upon the widely used Garnet NoC model within gem5. Table 3 shows the new features of HeteroGarnet for supporting accurate modeling of heterogenous networks.

V. Methodology

In this section, we discuss the overall methodology for NoI topology exploration.

A. System Configuration

We use HeteroGarnet for evaluation. We also integrated HeteroGarnet into the widely used gem5 simulator. For a fair comparison, we evaluate a system similar to previous studies [4, 7]. This involves 64 cores integrated to stacks of 3D DRAM using silicon interposer. Apart from mesh, all interposer networks use a 4-core-to-1-interposer-router concentration with synchronization buffers at all the clock-domain crossings. The 3D DRAM has 16 channels with each channel connected to exclusive or shared interposer router depending on the topology. The rest of the details are summarized in Table 2.

Parameter	Value
Virtual Networks	2 Mem/2 Coherence (4VCs each)
NoC topology (Chiplet)	Mesh, 16-Byte links
Router Latency	4 Cycles
Synthetic Core + NoC Frequency	4 GHz
GPU Core + NoC Frequency	1.7 GHz
Memory Controller Frequency	1.8 GHz
Memory link width	16-Byte
Clock-Domain-Crossing latency	1 cycle in slower clock domain
SerDes latency	2 cycles

Table 2: Simulation parameters for evaluations.

Workloads. We use both synthetic traffic and real applications for topology evaluation. The uniform random synthetic traffic includes 50% coherence traffic (between any pair of cores) and 50% memory traffic (between cores and memory channels). Within each coherence/memory traffic, half of the traffic is *Read* and half is *Write*. Read request and write response messages are 8-Byte in length, while read response and write request messages are 72-Byte.

Routing and Deadlock-Freedom. To keep the comparison fair, we use shortest path routing combined with escape virtual channels (VC) [13] for all topologies. In Mesh, CMesh, and CMesh(X), the escape VCs are restricted to deadlock free dimension order routing. For other topologies, we do not allow double-back turns in escape VCs. Any routing optimization would be orthogonal to our results.

B. Technology Model

HeteroGarnet is coupled with DSENT to extract the power and area estimations for the interconnect. For simplicity, we only report the area and power of the NoI, as other parts of the NoC are same in all the evaluations. Results are collected using a 22nm bulk/SOI low-V_t process node, with core-to-core minimal distance of 2.0 mm and µbump overhead of 0.2mm. All the topologies were swept over a frequency range to find their maximum operating points and power and area at those frequencies are reported.

VI. Design Evaluations

In this section, we evaluate different interposer topologies using the methodology and infrastructure discussed in previous sections.

A. NoC and NoI Split Clock Domains

HeteroGarnet enables us to evaluate a chiplet-interposer system where NoCs and NoI operate at different clock frequencies. In this experiment, we first reproduce results using the traditional methodology adopted by previous works, that is, NoCs and NoI operate at the same frequency. Then, we apply our new methodology to evaluate the performance impact when NoI is operating at its maximum frequency. Such a methodology will allow us to see the trade-off of designing an NoI with longer links.

Memory Traffic Latency. Figure 4(a) provides the load-latency curves of the evaluated NoI topologies when the entire NoC + NoI is operating at 1.8GHz (*Traditional Methodology*). We see that Mesh and CMesh suffer from high average latency even at low injection rate. Topologies such as Butter Donut, Kite Medium, and Kite Large take advantage of long links and thus are able to provide low latency. These topologies also deliver low latency at higher injection rates because of higher bisection bandwidth. Overall, Kite Large reduces latency by 1% while improving the saturating load by 8% over ButterDonut. The above observation motivates for long links within the NoI.

However, if the NoC + NoI is modeled faithfully (*HeteroGarnet Methodology*), we will draw a different conclusion. Figure 4(b) shows

[3] Detailed information about the features and use-cases of HeteroGarnet are available at http://synergy.ece.gatech.edu/tools/heterogarnet/

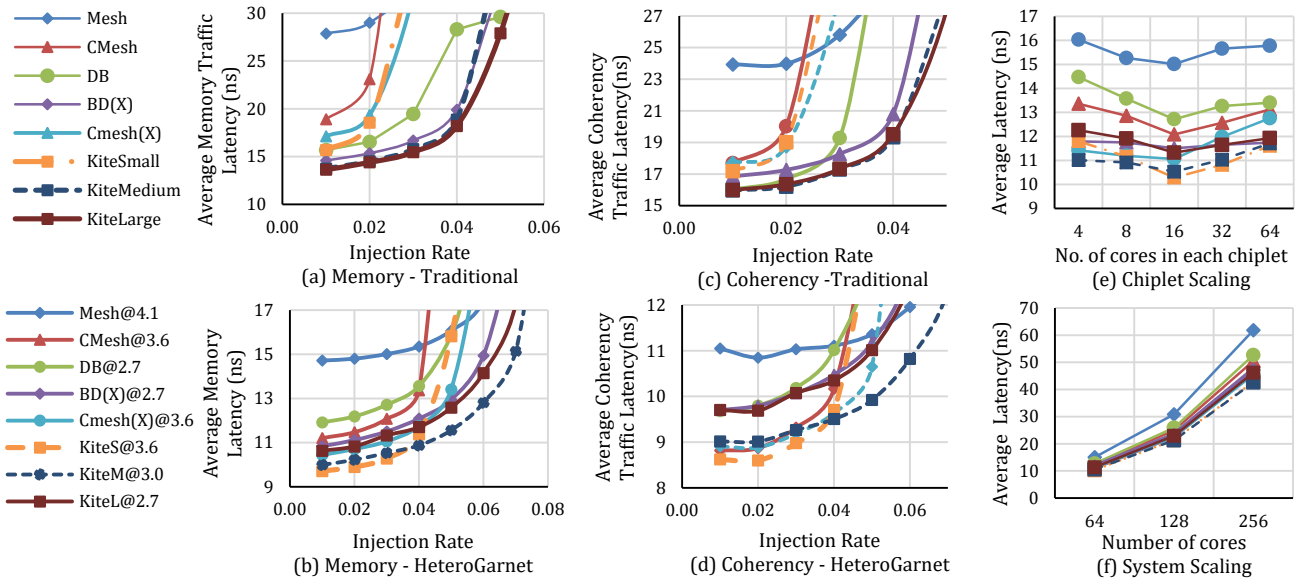

Figure 4: Memory traffic latency for NoI topologies (a) at equal freq. (1.8GHz) and (b) at max. freq. of each topology as shown in GHz in legend. Coherency traffic latency (c) at equal freq. (1.8GHz) and (d) at max. freq. of each topology. Average packet latency (0.03 injection) for different interposer topologies scaled to (e) different chiplet sizes and (f) different core counts.

the load-latency curves for the same set of topologies, with each operating at its maximum operational frequency. We see that topologies that have shorter links take advantage of the higher operating frequency and thus deliver better latency and improved throughput. In fact, the misaligned topology CMesh(X) outperforms ButterDonut in terms of latency at low injection rates by 4%. The DoubleButterfly topology, which was proposed as an improvement over CMesh topology, is only able to beat a regular CMesh topology at higher injection rates. This shows that while longer links can reduce hop count, the trade-off of operating at a lower frequency can take away some of the latency advantage.

The Kite based topologies, however, take full advantage of the longest links and offer low latency at lower injection rates and improved throughput at higher injection rate compared to their non-efficient counter parts. All Kite topologies offer lower latency than ButterDonut(X) while Kite Medium and Kite Large have improved saturation limits. Kite Small offers the lowest latency at low injection rates, improving over ButterDonut(X) by 10%. These latencies echo our estimations using the H_{eff} metric in Section III. Kite Medium, thanks to its low H_{eff} and high bisection bandwidth, has a low latency as well as better throughput. It improves over the latency offered by 7.5% while improving the saturating load limit by 17%. It is important to note that Kite Medium uses shorter links compared to Butter Donut and Kite Large and still outperforms. The Kite topologies' efficient usage of their longest links pays off both in terms of low latency and higher saturation limits for memory traffic.

Coherency Traffic Latency. Similar trend is observed with coherency traffic as well. Figure 4(c)-(d) presents the load-latency curves for coherency traffic with four chiplets comprising of 16 cores each [Figure 1(b)] with about 75% of the total coherency traffic going through the interposer. Figure 4(d) shows the latency experienced by coherency traffic when the NoI are operated at the maximum frequency possible with respective topologies (*HeteroGarnet Methodology)*. Unlike observations with *Traditional Methodology*, both CMesh and CMesh(X) deliver better latency than Double Butterfly, Butter Donut, and Kite Large. Kite Small delivers the lowest zero-load latency, improving over Butter Donut by 11%, but fails to provide comparable

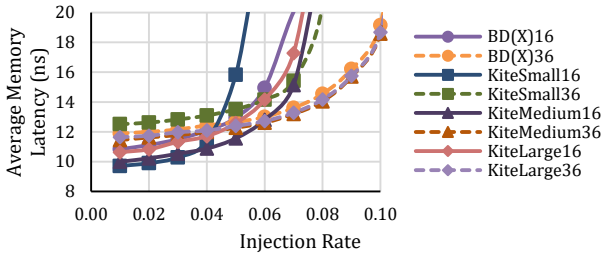

Figure 5: Load versus Memory traffic latency at varying injection rates comparing Kite Small at different link widths.

throughput. Kite Medium offers an economical performance by improving latency and maximum throughput as well.

These results show that accurate modeling of hybrid NoC avoids the pitfalls of inaccurate conclusions. Unlike previous works, our results show that while longer links build low-hop count topologies, they come at a cost of operating frequency. An efficient usage of long links will lead us to better designs such as showcased by the Kite topologies. *Although Kite Medium performed best in our evaluations, any of the Kite variants could be optimal depending on the design and technology constraints.*

B. NoI Link Width

Considering that a lot of topologies saturate at relative low injection rates, we increase the link width of NoI while keeping the core level NoC link width constant at 16 Bytes. We introduce Serializer-Deserializer (SerDes) units at the NoI boundaries with a latency of two cycles. Figure 5 compares a 36-Byte topology to its 16-Byte variant. All the topologies gain advantage in the form of improved throughput at higher injection rates. However, the latency at minimum injection rates is lower for a 16-Byte topology because of the lack of SerDes latency. Kite Small with 36-Byte, specifically, loses out considerably on its low latency. Butter Donut, Kite Medium, and Kite Large scale similarly when the link width is increased to 36-Bytes. The 36-Byte Kite Medium delivers the best performance by improving the maximum channel load by 37% compared to a 16-Byte Butter Donut.

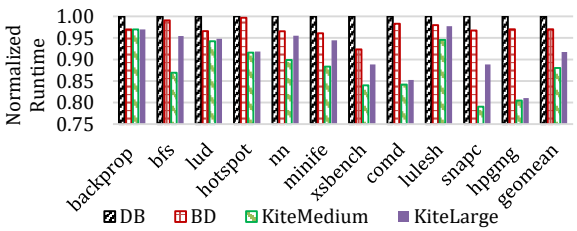

Figure 6: Normalized application runtime with key NoI topologies operated at their max frequencies.

C. Area and Power

The overhead of operating any logic at higher frequencies and link widths is the effect on area and power. Area on interposer is critical to maintain the minimal active logic [4] as described in Section II.C. Furthermore, high power consumption would not justify the performance improvements at higher frequencies. The Mesh topology suffers from high power and area cost because of the large number of routers present in the system. The Kite topologies have similar area and power overhead compared to ButterDonut. Our best performing topology, Kite Medium-32Byte has an area overhead of only 3.83 mm^2. This is only 1.3% of the total geometric area of the interposer since the shape of the interposer is defined by the cores and memory controller.

D. Scalability of NoI topologies

Figure 4(e) shows the average packet latency experienced by packets at different chiplet sizes. Scaling the chiplet size down to four cores increases the amount of coherency traffic within the interposer topology, resulting in higher latencies because of contention. On the other hand, having a single 64-core monolithic chiplet leads to all the coherency traffic being routed within the chiplet, and therefore restricting the coherency packets from using the optimized interposer topology. This leads to a degradation in average access latency at larger chiplet size. All topologies, including the Kite family, experience this trade-off with the 16-core chiplet being an optimal size. Kite Medium and Kite Small outperform other topologies even at other chiplet sizes and show that efficient interposer topologies scale well with chiplet size.

Figure 4(f) shows the average packet latency at injection rate of 0.03 with increasing number of cores. In general, the latency experienced by packets increases as we increase the number of cores because of increase in hops. The Kite family outperforms other topologies at all core counts, with the Kite Medium performing better even at a core count of 256.

E. Real Application

We integrated HeteroGarnet into gem5 [15] and evaluated a subset of Rodinia [16] and HPC proxy applications [17] using a 64 CU AMD GPU model [18]. Figure 6 shows the overall runtime of these applications normalized to the Double Butterfly based interposer NoC. We see that the performance improvement with Kite Medium is more visible in the case of memory intensive applications like *xsbench*, *snapc,* and *hpgmg*. Overall, compared to Double Butterfly and ButterDonut topologies, Kite Medium results in a performance improvement of about 12% and 9.2%, respectively.

VII. Conclusions

With the increasing complexity of NoCs in diverse modern system architectures, the need for an accurate heterogenous interconnect modeling has become vital to reach improved designs. In this paper, we proposed Kite topology family for chiplet-interposer systems. Compared to state-of-the-art NoI topologies, Kite topologies reduce network latency and improve throughput. We further presented HeteroGarnet NoC simulator that enabled accurate cycle-level modeling of the heterogeneities present in modern day hybrid NoCs. Using a technology-specific design metric and the new simulator, we identified the pitfalls in traditional methodology and integrated a methodology that caters to the heterogeneities. While topology exploration is one use case of HeteroGarnet, we hope that this work draws attention in NoC community and helps designers and researchers better understand the importance of accurate interconnect modeling.

Acknowledgement

We would like to thank Gabriel Loh and the anonymous reviewers of DAC-2020 for their feedback and suggestions.

References

[1] Intel, "Intel Foveros Interconnect.," 2019. [Online].

[2] K. Lepak et al., "The Next Generation AMD Enterprise Server," in *HOTCHIPS*, 2017.

[3] N. Beck et al., "'Zeppelin': An SoC for multichip architectures," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, 2018.

[4] A. Kannan et al., "Enabling interposer-based disintegration of multi-core processors," *IEEE Micro,* pp. 546-558, 2015.

[5] A. Arunkumar et al., "MCM-GPU: Multi-Chip-Module GPUs for Continued Performance Scalability," in *ISCA*, 2017.

[6] D. Green, "Common Heterogeneous Integration and IP Reuse Strategies (CHIPS)".

[7] J. Yin et al., "Modular Routing Design for Chiplet-Based Systems," in *2018 ACM/IEEE (ISCA)*, 2018.

[8] N. E. Jerger, T. Krishna and L.-S. Peh, "On-Chip Networks, Second Edition," *Synthesis Lectures on Computer Architecture,* vol. 12, pp. 1-210, 2017.

[9] N. D. E. Jerger et al., "NoC Architectures for Silicon Interposer Systems," *IEEE Micro,* pp. 458-470, 2014.

[10] J. Kim et al., "Flattened Butterfly Topology for On-Chip Networks," *IEEE Computer Architecture Letters,* 2007.

[11] M. Besta and T. Hoefler, "Slim fly: a cost effective low-diameter network topology," 2014.

[12] C. Sun et al., "DSENT - A Tool Connecting Emerging Photonics with Electronics for Opto-Electronic Networks-on-Chip Modeling," in *NOCS*, 2012.

[13] J. Duato, "A theory of deadlock-free adaptive multicast routing in wormhole networks," *IEEE Transactions on Parallel and Distributed Systems,* vol. 6, no. 9, pp. 976-987, 1995.

[14] N. Binkert et al., "The gem5 simulator," in *SIGARCH Comput. Archit. News*, 2011.

[15] S. Che et al., "Rodinia: A benchmark suite for heterogeneous computing," in *IISWC*, 2009.

[16] J. R. Tramm et al., "XSBench-the development and verification of a performance abstraction for Monte Carlo reactor analysis," *PHYSOR,* 2014.

[17] A. Gutierrez et al., "Lost in Abstraction: Pitfalls of Analyzing GPUs at the Intermediate Language Level," in *HPCA*, 2018.

[18] A. Shilov, "AMD previews EPYC 'Rome'processor: Up to 64 Zen 2 cores," 2018.

[19] N. Jiang et al., "A detailed and flexible cycle-accurate Network-on-Chip simulator," in *2013 IEEE (ISPASS)*, 2013.

[20] O. Villa et al., "Scaling the power wall: a path to exascale," in *SC*, 2014.

[21] N. Agarwal et al., "GARNET: A detailed on-chip network model inside a full-system simulator," in *ISPASS*, 2009.

[22] P. Abad et al., "TOPAZ: An Open-Source Interconnection Network Simulator for Chip Multiprocessors and Supercomputers," in *NOCS*, 2012.

[23] N. McDonald et al., "SuperSim: Extensible Flit-Level Simulation of Large-Scale Interconnection Networks," 2018. [Online].

[24] C. Nicopoulos et al., "ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers," *IEEE Micro,* 2006.

[25] I. Karlin, J. Keasler and J. R. Neely, "Lulesh 2.0 updates and changes," 2013.