Enabling Interposer-based Disintegration of Multi-core Processors

Ajaykumar Kannan‡
kannan.ajay@gmail.com

Natalie Enright Jerger‡†
enright@ece.utoronto.ca

Gabriel H. Loh†
gabriel.loh@amd.com

‡Edward S. Rogers Dept. of Electrical and Computer Engineering
University of Toronto

†AMD Research
Advanced Micro Devices, Inc

ABSTRACT

Silicon interposers enable the integration of multiple stacks of in-package memory to provide higher bandwidth or lower energy for memory accesses. Once the interposer has been paid for, there are new opportunities to exploit the interposer. Recent work considers using the routing resources of the interposer to improve the network-on-chip's (NoC) capabilities. In this work, we exploit the interposer to "disintegrate" a multi-core CPU chip into smaller chips that individually and collectively cost less to manufacture than a single large chip. However, this fragments the overall NoC, which decreases performance as core-to-core messages between chips must now route through the interposer. We study the performance-cost trade-offs of interposer-based, multi-chip, multi-core systems and propose new interposer NoC organizations to mitigate the performance challenges while preserving the cost benefits.

Categories and Subject Descriptors

B.3 [**Memory Structures**]: Dynamic Memory; B.4.3 [**Interconnections**]: Topology

Keywords

Silicon interposer, die stacking, network-on-chip

1. INTRODUCTION

This paper is about the convergence of two well-known trends. The first is the often-discussed end or slow-down of Moore's law. For decades, the industry has enjoyed exponential growth in the functionality per unit-area of silicon. However, in recent years, fundamental physical limitations have slowed down the rate of transition from one technology node to the next, and the costs of new fabs are sky-rocketing.

MICRO 2015 Waikiki, Hawaii USA

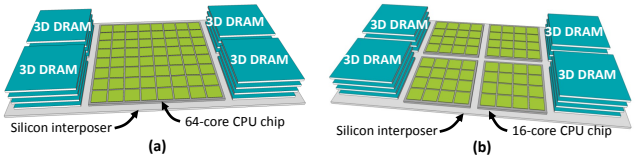

Figure 1: (a) An example interposer-based system integrating a 64-core processor chip with four 3D stacks of DRAM, (b) and a 64-core system composed of four 16-core processor chips.

Moore's Law has conventionally been used to increase integration, from floating point units to memory controllers, from GPUs to IO and the South Bridge. However going forward, almost everything that can be easily integrated has already been integrated! What remains is largely implemented in disparate process technologies (e.g., memory, analog) [7]. This is where the second trend comes into play, which is the maturation of die-stacking technologies. Die stacking enables the continued integration of system components in traditionally incompatible processes. A key initial application of die-stacking is silicon interposer-based integration of multiple 3D stacks of DRAM, shown in Figure 1(a) [1, 7, 13, 26], potentially providing several gigabytes of in-package memory[1] with bandwidths already starting at 128GB/s (per stack) [16].

The use of an interposer presents new opportunities; if one has already paid for the interposer to integrate memory, any additional benefits from exploiting the interposer can come at a relatively small incremental cost. One recent work considers using the additional routing resources (wires) and possibly devices (if an "active" interposer is used) to extend the traditional multi-core NoC [13]. We study how the interposer can be used to (at least in part) address the increasing costs of manufacturing chips in a leading-edge process technology. We propose to use the interposer to "disintegrate" a large multi-core chip into several small chips, such as in Figure 1(b). The chips are cheaper to manufacture due to a combination of higher yield and better pack-

[1]Several gigabytes of memory is unlikely to be sufficient for high-performance systems and will likely still require tens or hundreds of gigabytes of conventional (e.g., DDR) memory outside of the processor package. Management of a multi-level memory hierarchy is not the focus of this work.

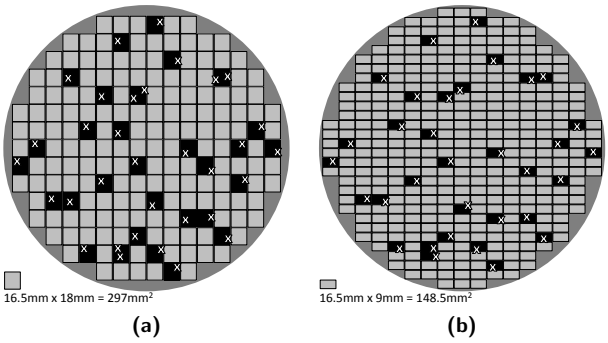

Figure 2: Example 300mm wafers with two different chip sizes of (a) 297mm^2 and (b) 148.5mm^2 (half), showing the overall number of chips and the impact on yield of an example defect distribution.

ing of rectangular die on a round wafer. Unfortunately, this approach fragments the NoC such that each chip contains only a piece of the overall network, and communications between chips must take additional hops through the interposer. This paper explores how to disintegrate a multi-core processor on an interposer while addressing the problem of a fragmented NoC.

2. BACKGROUND AND MOTIVATION

2.1 Disintegrating to Manage Silicon Costs

Manufacturing costs of integrated circuits are increasing at a dramatic rate. To reduce the cost of manufacturing a system on a chip, we can reduce the size of the chip. A larger chip's cost comes from two main sources. The first is geometry: fewer larger chips fit in a wafer. Figure 2 shows two 300mm wafers, one filled with 297mm^2 chips, and the other with 148.5mm^2 chips (half the size). We can pack 192 larger chips in a single wafer. The smaller chips can be packed more tightly (i.e., utilizing more of the area around the periphery of the wafer) resulting in 395 chips (more than 2×192).

The second cost of a larger chip is due to manufacturing defects. A defect that renders a large die inoperable wastes more silicon than one that kills a smaller die. Figure 2 illustrates an example defect distribution (identical for both wafers) that renders some fraction of the chips inoperable. This reduces the 192 original large die to 162 good die per wafer (GDPW), resulting in a ∼16% yield loss. For the half-sized die, we go from 395 die to 362 GDPW for a ∼8% yield loss. In general, a smaller chip gets you more chips, and more of them work.

While smaller chips result in lower costs, the downside is that they also provide less functionality (e.g., half the area yields half the cores). If we could manufacture several smaller chips and combine them together into a single system, then we would be able to have the functionality of a larger chip while maintaining the economic advantages of the smaller chips. Ignoring for the time being exactly how multiple chips could be combined back together, Table 1 summarizes the impact of implementing a 64-core system ranging from a conventional 64-core monolithic chip all the way down to

Cores Per Chip	Chips Per Wafer	Chips per Package	Area per Chip (mm2)	Chip Yield	Good Die Per Wafer	Good SoCs per Wafer
64	192	1	297.0	84.5%	162	162
32	395	2	148.5	91.7%	362	181
16	818	4	74.3	95.7%	782	195
8	1,664	8	37.1	97.8%	1,627	203
4	3,391	16	18.6	98.9%	3,353	209

Table 1: Example yield analysis for different-sized multi-core chips. A "SoC" here is a 64-core system, which may require combining multiple chips for the rows in the table corresponding to chips with less than 64 cores each.

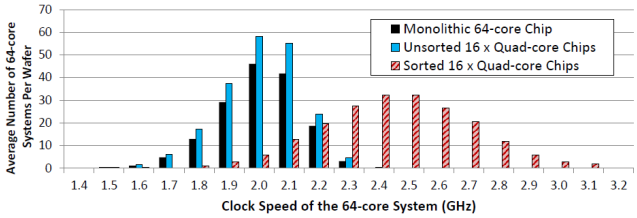

Figure 3: Average number of 64-core SoCs per wafer per 100MHz bin from Monte Carlo simulations of 100 wafers.

building it from sixteen separate quad-core chips.The last column in Table 1 shows that using a collection of quad-core chips to assemble a 64-core SoC yields 29% more working parts than the monolithic-die approach.

The results in Table 1 assume the usage of known-good-die (KGD) testing techniques so that individual chips can be tested before being assembled together to build a larger system. If die testing is used, then the chips potentially can also be speed-binned prior to assembly. We used Monte Carlo simulations to consider three scenarios: (1) a 300mm wafer is used to implement 162 monolithic good die per wafer (as per Table 1), (2) the wafer is used to implement 3,353 quad-core chips, which are then assembled without speed binning into 209 64-core systems, and (3) the individual die from the same wafer are sorted so that the fastest sixteen chips are assembled together, the next fastest sixteen are combined, and so on. Figure 3 shows the number of 64-core systems per wafer in 100MHz speed bins, averaged across one hundred wafer samples per scenario. The Monolithic 64-core chip and 16 quad-core approaches have similar speed distributions. However with speed binning, we avoid the situation where overall system speed is slowed down by the presence of a single slow chip, resulting in significantly faster average system speeds (the mean shifts by ∼400 MHz) and more systems in the highest speed bins (which usually carry the highest profit margins).

2.2 Options for Integrating Multiple Chips

We consider four technologies to reassemble multiple smaller chips into a larger system: multi-socket boards, multi-chip modules (MCM), silicon interposer-based integration (2.5D), and vertical 3D chip stacking.

Multi-socket: Symmetric multi-processing (SMP) systems on multiple sockets have been around for decades. The primary downsides are that the bandwidth and latency between sockets is worse than some of the packaging technologies discussed below (resulting

in a higher degree of non-uniform memory accesses or NUMA). The limitation comes from a combination of the limit on the per-package pin count and the intrinsic electrical impedance between chips (e.g., C4 bumps, package substrate metal, pins/solder bumps, printed circuit board routing).

Multi-chip Modules: MCMs take multiple chips and place them on the same substrate within the package. This avoids the pin count limitations of going package-to-package, but the bandwidths and latencies are still constrained by the C4 bumps and substrate routing that connect the silicon die.

Silicon Interposers: A silicon interposer is effectively a large chip upon which other smaller die can be stacked. The micro-bumps (μbumps) used to connect the individual die to the interposer have greater density than C4 bumps (e.g., $\sim$9$\times$ better assuming 150μm and 50μm pitches for C4 and μbumps), and the impedance across the interposer is identical to conventional on-chip interconnects (both are made with the same process). The main disadvantage is the cost of the additional interposer.

3D Stacking: Vertical stacking combines multiple chips, where each chip is thinned and implanted with through-silicon vias (TSV) for vertical interconnects. 3D has the highest potential bandwidth, but has the greatest cost and overall process complexity as nearly every die must be thinned and processed for TSVs. ■

The SMP and MCM approaches are less desirable as they do not provide adequate bandwidth for arbitrary core-to-core cache coherence without exposing significant NUMA effects, and they also have higher energy-per-bit costs compared to the die-stacked options. As such, we do not consider them further. 3D stacking by itself is not (at least at this time) as an attractive of a solution because it is more expensive and complicated, introduces potentially severe thermal issues, and may be an overkill in terms of how much bandwidth it can provide. This leaves us with silicon interposers.

2.3 Are (Active) Interposers Practical for NoCs?

The interposer is effectively a very large chip. Current approaches use *passive* interposers [7, 12] where the interposer has no devices, only routing. This greatly reduces the *critical area* (A_{crit}) of the interposer (i.e., unless a defect impacts a metal route, there are no transistors that can be affected), resulting in high yields. Based on our previous chip cost analysis, it would seem prohibitively expensive to consider an active interposer, but having the flexibility of placing routers on the interposer enables many more interesting NoC organizations. While the interposer could be implemented in an older process technology[2] [25] that is less expensive and more mature (i.e., lower defect rates), such a large chip, perhaps near the reticle limit, would still not be expected to be cheap if the yield rates remain low.

D_0	500	1000	1500	2000	2500
Passive	98.5%	97.0%	95.5%	94.1%	92.7%
Active 1%	98.4%	96.9%	95.4%	93.9%	92.5%
Active 10%	98.0%	96.1%	94.2%	92.4%	90.7%
Fully-active	87.2%	76.9%	68.5%	61.5%	55.6%

Table 2: Yield rates for 24mm×36mm interposers varying the active devices/transistors from none (passive) to 100% filled (fully-active) across different defect rates (D_0 in defects per m^2).

For regular chips, it is typically desirable to maximize functionality by cramming in as many transistors into one's chip-area budget. However, making use of every last mm^2 of the interposer would lead to a very high fraction of the area being critical $Frac_{crit}$ multiplied over a very large area (A_{crit} = chip-area $\times Frac_{crit}$), thereby leading to low yields and high costs. However, there is no need to use the entire interposer; its size is determined by the geometry of the chips and memory stacked upon it, therefore using more or fewer devices has no impact on its final size. As such, we advocate using a *Minimally-active Interposer*: implement the devices required for the functionality of the system (in our case it would be routers and repeaters), but no more. This results in a sparsely-populated interposer with a lower $Frac_{crit}$ and therefore a lower cost.

We used the same yield model from our earlier cost analysis to estimate the yields of different interposer options: a passive interposer, a minimally-active interposer, and a fully-active interposer. The interposer size assumed throughout this work is 24mm×36mm (864mm^2), and we assume six metal layers in the interposer. For a passive interposer, $Frac_{crit}$ for the logic is zero (it remains non-zero for the metal layers). For a fully-active interposer (i.e., if one fills the entire interposer with transistors), we use the same A_{crit} from Table 4. For a minimally-active interposer, we estimate the total interposer area needed to implement our routers (logic) and links (metal) to be only 1% of the total interposer area. To be conservative, we also consider a minimally-active interposer where we pessimistically assume the router logic consumes 10$\times$ more area although the metal utilization is unchanged. Minimizing utilization of the interposer for active devices also minimizes the potential for undesirable thermal interactions resulting from stacking highly active CPU chips on top of the interposer.

Table 2 shows estimated yield rates for the different interposer options. We show the same defect rate (2000) from Table 1, plus four other rates.[3] The two lowest rates reflect that the interposer is likely to be manufactured in an older, more mature process node with

[2]While an older process technology does not provide the same metal density at the lowest layers (e.g., M1), the long-haul NoC links would be routed in thicker upper-level metal layers that can be supported in older technology generations. The NoC's clock speed is likely in the 1-2 GHz range, which has been a practical speed for many generations of devices. Using an older process would require more area than in a leading edge process (but there is plenty of spare room on the interposer).

[3]These defect rates are selected simply to show a range of potential yield rates and for qualitatively demonstrating the tradeoffs among the different options. These rates are *not* in any way chosen to be representative of any specific processes, technology generations, device types, foundries, etc.

lower defect rates. The passive interposer has a non-perfect yield rate (<100%) as it still uses metal layers that can be rendered faulty by manufacturing defects. At the other extreme is a fully-active interposer, where the higher defect rates (1,500-2,500 defects per m^2) result in very low yields. This is not surprising given that a defect almost anywhere on the interposer could render it a loss. This is the primary reason why one would likely be skeptical of active interposers. However, Table 2 shows that when using only the minimum amount of active area necessary on the interposer, the yield rates are not very different from the passive interposer. The vast majority of the interposer is *not* being used for devices; defects that occur in these "white space" regions do not impact the interposer's functionality. So even with the conservative assumption that the NoC routers consume 10% of the interposer area at the highest defect rates considered, our model predicts yields of over 90%. As a result, we believe that augmenting an otherwise passive interposer with just enough logic to do what one needs has the potential to be economically viable,[4] and it should be sufficient for NoC-on-interposer applications.

It is important to note that the above yield models cannot replace a complete cost analysis. However, the lack of publicly available data makes it incredibly difficult to provide meaningful dollar-for-dollar cost comparisons. Factors such as additional costs for the extra masks (mask set costs are effectively amortized over the all units shipped) for an active interposer and additional processing steps (incurred per unit) must be combined with the yield analysis to arrive at a final decision as to whether a given SoC should use an active interposer.

2.4 The Research Problem

Taking the cost argument *alone* to its logical limit would lead one to falsely conclude that a large chip should be disintegrated into an infinite number of infinitesimally small die. The countervailing force is performance: While breaking a large system into smaller pieces may improve overall yield, going to a larger number of smaller chips increases the amount of chip-to-chip communication that must be routed through the interposer. In an interposer-based multi-core system with a NoC distributed across chips and the interposer, smaller chips create a more fragmented NoC resulting in more core-to-core traffic routing across the interposer, which eventually becomes a performance bottleneck. Figure 4 shows the cost reduction for three different defect rates, all showing the relative cost benefit of disintegration. The figure also shows the relative impact on performance.[5] So while more aggressive levels of disintegration provide better cost savings, it is directly offset by a reduction in performance.

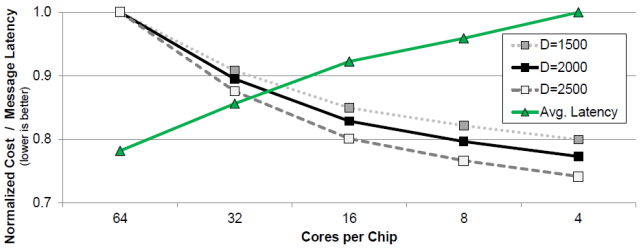

Figure 4: Normalized cost and execution time (lower is better for both) for different multi-chip configurations. 64 cores per chip corresponds to a single monolithic 64-core die, and 4 cores per chip corresponds to 16 chips, each with four cores. Cost is shown for different defect densities (in defect/m^2), and the average message latency is normalized to the 16 quad-core configuration.

The problem explored in the remainder of this paper is how one can get the cost benefits of a disintegrated chip organization while providing a NoC architecture that, while physically fragmented across multiple chips, still behaves (performance-wise) at least as well as one implemented on a single monolithic chip.

3. ARCHITECTING THE NOC FOR MULTI-CHIP INTERPOSERS

The analysis in the preceding section shows that there are economic incentives for disintegrating a large multi-core chip into smaller die, but that doing so induces performance challenges with respect to the interconnect. We now discuss how to address these issues.

3.1 Baseline Multi-chip NoC Topologies

In an interposer-based system employing a monolithic multi-core chip, the NoC traffic can be cleanly separated into cache coherence traffic routed on the CPU layer, and main memory traffic on the interposer layer [13]. This minimizes interference and contention between the traffic types and more easily allows for per-layer customized topologies that best match the respective traffic patterns. When the multi-core chip has been broken down into smaller pieces, coherence traffic between cores on different chips must now venture off onto the interposer. As a result, this mixes some amount of coherence traffic in with the main memory traffic, which in turn disturbs the traffic patterns observed on the interposer.

Figure 5 shows per-link traffic for a horizontal set of routers across the interposer for several topologies. The first is the baseline case for a monolithic 64-core chip stacked on an interposer that also implements a 2D mesh. All coherence stays on the CPU die, and memory traffic is routed across the interposer, which results in relatively low and even utilization across all links. Figure 5(b) shows four 16-core chips on top of the same interposer mesh. Traffic between chips must now route through the interposer, which is reflected by an increase particularly in the middle link right between the two chips. Figure 5(c) shows the same four chips, but now stacked on an interposer with a concentrated-mesh

[4]Today, active interposers are economically challenging. However, this research is targeted at systems which may not be manufactured for another 5-8 years (or more), at which point the economics of die stacking, the impact of the slowing of Moore's Law/Dennard scaling, etc. will have likely shifted what is practical and cost effective.

[5]We show the average message latency for all traffic (coherence and main memory) in a synthetic uniform-distribution workload, where CPU chips and the interposer respectively use 2D meshes vertically connected through μbumps. See Section 4 for full details.

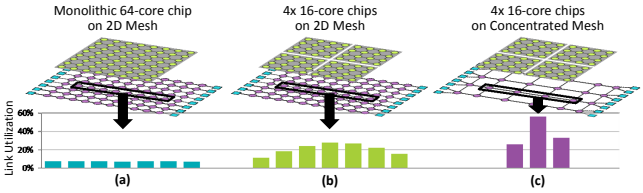

Figure 5: Link utilization from a single "horizontal" row of routers on the interposer for (a) a monolithic 64-core chip stacked on a interposer with a 2D mesh, (b) four 16-core chips stacked on a 2D mesh, and (c) four 16-core chips stacked on a concentrated mesh.

network. Any traffic from the left side chips to the right must cross the middle links, causing further contention with memory traffic. The utilization of the middle link clearly shows how the bisection-crossing links can easily become bottlenecks in multi-chip interposer systems.

In addition to regular and concentrated mesh topologies, shown in Figure 6(a) and (b), we consider two additional baseline topologies for the interposer portion of the NoC to address the traffic patterns induced by chip disintegration. The first is the "Double Butterfly" [13] that optimizes the routing of traffic from the cores to the edges of the interposer where the memory stacks reside. The Double Butterfly in Figure 6(c) has the same number of nodes as the CMesh, but provides the same bisection bandwidth as the conventional mesh. Next, we consider the Folded Torus,[6] shown in Figure 6(d). Similar to the Double Butterfly, the Folded Torus provides twice the bisection bandwidth compared to the CMesh. The Folded Torus actually can provide "faster" east-west transit as each link spans a distance of two routers, but main-memory traffic may not be routed as efficiently as a Double Butterfly due to the lack of the "diagonal" links. Both of these topologies assume the same 4-to-1 concentration as the CMesh.

3.2 Misaligned Topologies

When using either Double Butterfly or Folded Torus topologies on the interposer layer, overall network performance improves substantially over either the conventional or concentrated meshes (see Section 5). However, the links that cross the bisection between the two halves of the interposer still carry a higher amount of traffic and continue to be a bottleneck for the system. We now introduce the concept of a "misaligned" interposer topology. For our concentrated topologies thus far, every four CPU cores in a 2×2 grid share an interposer router that was placed in between them, as shown in both perspective and side/cross-sectional views in Figure 7(a). So for a 4×4 16-core chip, there would be four "concentrating" router nodes *aligned* directly below each quadrant of the CPU chip.

A misaligned interposer network offsets the location of the interposer routers. Cores on the edge of one chip now share a router with cores on the edge of the adjacent chip as shown in Figure 7(b). The change is subtle but important: with an "aligned" interposer NoC, the key resources shared between chip-to-chip coherence and memory traffic are the links crossing the bisection line, as shown in the bottom of Figure 7(a). If both a memory-bound message (M) and a core-to-core coherence message (C) wish to traverse the link, then one must wait as it serializes behind the other. With misaligned topologies, the shared resource is now the *router*. As shown in the bottom of Figure 7(b), this simple shift allows chip-to-chip and memory traffic to flow through a router simultaneously, thereby reducing queuing delays for messages to traverse the network's bisection cut.

[6]This is technically a 2D Folded Torus, but omit the "2D" for brevity.

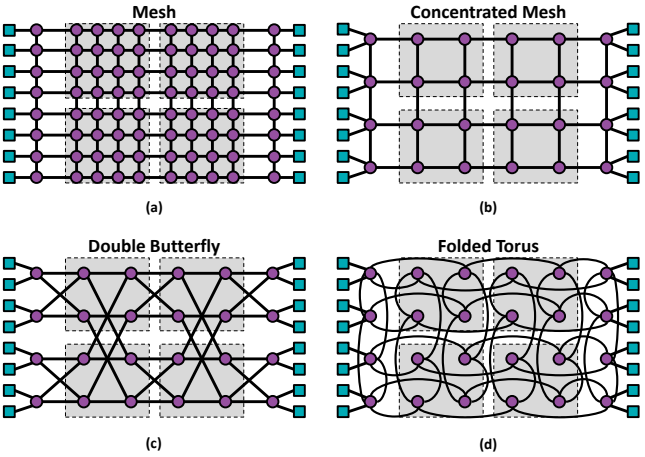

Figure 6: Baseline topologies for the interposer portion of the NoC, including (a) 2D Mesh, (b) 2D Concentrated Mesh, (c) Double Butterfly, and (d) 2D Folded Torus. The squares on the edges are memory channels; the four large shaded boxes illustrate the placement of four 16-core chips above the interposer.

Depending on the topology, interconnect misalignment can be applied in one or both dimensions. Figure 8(a) shows a Folded Torus misaligned in the X-dimension only, whereas Figure 8(b) shows a Folded Torus misaligned in both X- and Y-dimensions. Note that misalignment changes the number of nodes in the topology (one fewer column for both examples, and one extra row for the X+Y case). For the Double Butterfly, we can only apply misalignment in the X-dimension as shown in Figure 8(c) because misaligning in the Y-dimension would change the number of rows to five, which is not amenable to this butterfly organization.

3.3 The ButterDonut Topology

One of the key reasons why both Double Butterfly (DB) and Folded Torus (FT) topologies perform better than the CMesh is that they both provide twice the bisection bandwidth. In the end, providing more bandwidth tends to help both overall network throughput and latency (by reducing congestion-related queuing delays). One straightforward way to provide more bisection bandwidth is to add more links, but if not done carefully, this can cause the routers to need more ports (higher degree), which increases area and power, and

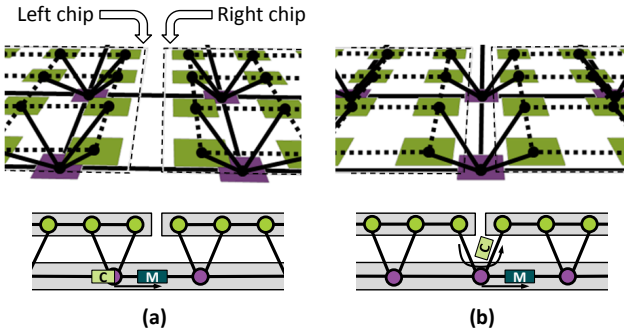

Figure 7: Perspective and side/cross-sectional views of (a) 4-to-1 concentration from cores to interposer routers aligned beneath the CPU chips, and (b) 4-to-1 concentration misaligned such that some interposer routers are placed "in between" neighboring CPU chips. The cross-sectional view also illustrates the flow of example coherence (C) and memory (M) messages.

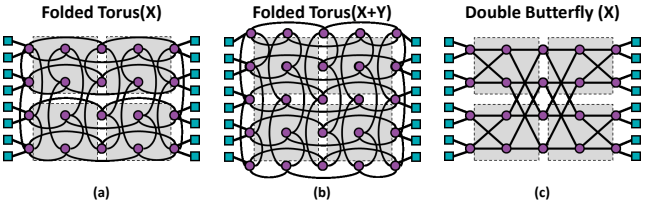

Figure 8: Example implementations of misaligned interposer NoC topologies: Folded Torus misaligned in the (a) X-dimension only, (b) both X and Y, and (c) a Double Butterfly misaligned in the X-dimension.

can decrease the maximum clock speed of the router. Note that the topologies considered thus far (CMesh, DB, FT) all have a maximum router degree of eight for the interposer-layer routers (i.e., four links concentrating from the cores on the CPU chip(s), and then four links to other routers on the interposer).

By combining different topological aspects of both DB and FT topologies, we can further increase the interposer NoC bisection bandwidth without impacting the router complexity. Figure 9 shows our *ButterDonut* topology, which is a hybrid of both the Double Butterfly and the Folded Torus. [7] All routers have at most four interposer links (in addition to the four links to cores on the CPU layer); this is the same as CMesh, DB, and FT. However, as shown in Figure 9(a), the ButterDonut has twelve links crossing the vertical bisection (as opposed to eight each for DB and FT, and four for CMesh).

Similar to the DB and FT topologies, the ButterDonut can also be "misaligned" to provide even higher throughput across the bisection. An example is shown in Figure 9(b). Like the DB, the misalignment technique can only be applied in the X-dimension as the ButterDonut still makes use of the Butterfly-like diagonal links.

Topologies can be compared among several different metrics. Table 3 shows all of the concentrated topologies considered in this paper along with several key network/graph properties. The metrics listed correspond only to the interposer's portion of the NoC (e.g., nodes on the CPU chips are not included), and the link counts exclude both connections to the CPU cores as well to the memory channels (this is constant across all configurations, with 64 links for the CPUs and 16 for the memory channels). Misaligned topologies are annotated with their misalignment dimension in parenthesis, for example, the Folded Torus misaligned in the X-dimension is shown as "FoldedTorus(X)". As shown in the earlier figures, misalignment can change the number of nodes (routers) in the network. From the perspective of building minimally-active interposers, we want to favor topologies that minimize the number of nodes and links to keep the interposer's A_{crit} as low as possible. At the same time, we would like to keep network diameter and average hop count low (to minimize expected latencies of requests) while maintaining high bisection bandwidth (for network throughput). Overall, the X-misaligned ButterDonut topology has the best properties out of all of the topologies except for the link count, for which it is a close second behind DoubleButterfly(X). ButterDonut(X) combines the best of all of the other non-ButterDonut topologies, while providing 50% more bisection bandwidth.

[7] "Butter" comes from the Butterfly network, and "Donut" is chosen because they are torus-shaped and delicious.

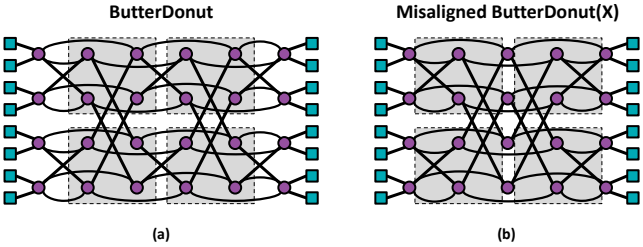

Figure 9: The (a) aligned and (b) misaligned ButterDonut topologies combine topological elements from both the Double Butterfly and Folded Torus.

	Topology	Nodes	Links	Diameter	Avg Hop	Bisection Links
	CMesh	24 (6x4)	38	8	3.33	4
	DoubleButterfly	24 (6x4)	40	5	2.70	8
	FoldedTorus	24 (6x4)	48	5	2.61	8
	ButterDonut	24 (6x4)	44	**4**	2.51	**12**
Misaligned	FoldedTorus(X)	**20** (5x4)	40	**4**	**2.32**	8
Misaligned	DoubleButterfly(X)	**20** (5x4)	**32**	**4**	2.59	8
Misaligned	FoldedTorus(XY)	25 (5x5)	50	**4**	2.50	10
Misaligned	**ButterDonut(X)**	**20** (5x4)	**36**	**4**	**2.32**	**12**

Table 3: Comparison of the different interposer NoC topologies studied in this paper. In the node column, n×m in parenthesis indicates the organization of router nodes. Bisection Links are the number of links crossing the vertical bisection cut.

3.4 Deadlock Freedom

The Folded Torus and ButterDonut topologies are susceptible to network-level deadlock due to the presence of rings within a dimension of the topology. Two approaches have been widely employed to avoid deadlock in torus networks: virtual channels [9] and bubble flow control [28]. We leverage recently proposed flit-level

bubble flow control [8, 24] to avoid deadlock in these rings. As the ButterDonut topology only has rings in the X-dimension, bubble flow control is applied in that dimension only and typical wormhole is applied for packets transiting through the Y-dimension.[8] For the Folded Torus, bubble flow control must be applied in both dimensions. As strict dimension order routing cannot be used in the ButterDonut topology (packets can change from X to Y and from Y to X dimensions), an additional virtual channel is required. We modify the original routing algorithm for the DoubleButterfly [13]; routes that double-back (head E-W and then W-E on other links) are not possible due to disintegration. Table-based routing based on extended destination tag routing coupled with extra VCs maintain deadlock freedom for these topologies.

4. METHODOLOGY

In this section, we discuss the methodology for the cost and yield analysis in Section 2. We also describe the simulation framework and workloads used in the evaluation of disintegrated interposer-based systems.

4.1 Yield and Cost Models

For the yield and relative cost figures used in Section 2, we make use of analytical yield models, a fixed cost-per-wafer assumption, automated tools for computing die-per-wafer [14], and consider a range of defect densities. All analyses assume a 300mm wafer. Our baseline monolithic 64-core die size is 16.5mm × 18mm (same assumption as used in the recent interposer-NoC paper [13]). Smaller-sized chips are derived by halving the longer of the two dimensions (e.g., 32-core chip is 16.5mm × 9mm). The yield rate for individual chips is estimated using a simple classic model [29]:

$$Yield = \left(1 + \frac{\frac{D_0}{n} A_{crit}}{\alpha}\right)^{-\alpha}$$

where D_0 is the defect density (defects per m^2), n is equal to the number of vulnerable layers (13 in our analyses, corresponding to one layer of devices and 12 metal layers), A_{crit} is the total vulnerable area (i.e., a defect that occurs where there are no devices does not cause yield loss), which is the product of the chip area and $Frac_{crit}$, and α is a clustering factor to model the fact that defects are typically not perfectly uniformly distributed. We ran our experiments for several other values of α, but the overall results were not qualitatively different. For A_{crit}, we assume different fractions of the total chip area are critical depending on whether it is a device or metal layer. Our model values are listed in Table 4 for reproducibility, but it should be noted that the exact parameters here are not crucial: the main result (which is not new) is that smaller chips are cheaper.

For the speed binning results in Section 2, we simulate the yield of a wafer by starting with the good-die-per-wafer based on the geometry of the desired chip (Table 1). For each quad-core chip, we randomly select its speed using a normal distribution (mean 2400MHz, standard deviation of 250MHz).[9] Our simplified model treats a 64-core chip as the composition of sixteen adjacent (4×4) quad-core clusters, with the speed of each cluster chosen from the same distribution as the individual quad-core chips. Therefore the clock speed of the 64-core chip is the minimum from among its constituent sixteen clusters. For each configuration, we simulate 100 different wafers worth of parts, and take the average over the 100 wafers. Similar to the yield results, the exact distribution of per-chip clock speeds is not so critical: so long as there exists a spread in chip speeds, binning and reintegration via an interposer can potentially be beneficial for the final product speed distribution.[10]

Parameter	Value
n	13
$Frac_{crit}$ (wire)	0.2625
$Frac_{crit}$ (logic)	0.7500
α	1.5

Table 4: Parameters for the chip yield calculations.

4.2 Performance Model and Workloads

To evaluate the performance of various interposer NoC topologies for our disintegrated systems, we use a cycle-level network simulator [17] with the configuration parameters listed in Table 5. We use DSENT [30] as described below to estimate frequency, area, and power for the different topologies. We assume all NoC topologies are clocked at 1GHz. Long links needed by some topologies are properly repeated in order to meet timing. All configurations assume an appropriately sized mesh on the core dies (e.g., a 16-core chip uses a local 4×4 mesh). All interposer networks use 4-to-1 concentration except for the mesh. There are two DRAM stacks on each of the two sides of the interposer. Each DRAM stack provides four independent memory channels, for a system-wide total of 16 channels. Each DRAM channel consists of a 128-bit data bus operating at 1.6GHz. DRAM-specific timing (e.g., bank conflicts, refresh) are modeled. The interposer network dimensions include the end nodes that interface with the DRAM memory channels.

To evaluate the baseline and proposed NoC designs, we use both synthetic traffic patterns and SynFull traffic models [3]; these two evaluation approaches cover a wide range of network utilization scenarios and exercise both cache coherence traffic and memory traffic. For the SynFull workloads, we run multi-programmed combinations composed of four 16-way multi-threaded applications from PARSEC [5]. The threads of the applications are distributed across the 64 cores and they

[8] This discussion treats diagonal links as Y-dimension links.

[9] These values are not meant to be representative of any specific processor, SoC, technology generation, etc., but are chosen as a working example.

[10] Expenses associated with the binning process are not included our cost metric, as such numbers are not readily available, but it should be noted that the performance benefits of binning could incur some overheads. Similarly, disintegration into a larger number of smaller chips requires a corresponding increase in assembly steps, for which we also do not have relevant cost information available.

Common Parameters (all routers)	
VCs	8, 8 flit buffers each
Pipeline	4 stages
Multi-core Die NoC Parameters	
all configs	Standard 2D mesh, DOR routing
Interposer NoC Parameters	
Mesh, Cmesh, All Folded Tori	DOR routing
All DoubleButterfly variants	Extended destination tag routing
ButterDonut variants	Table-based routing

Table 5: NoC simulation parameters.

share all 16 memory channels. We construct workload combinations based on their memory traffic intensity. For synthetic traffic patterns, we use uniform random traffic where we varied the ratio of coherence and memory requests injected into the network, but the majority of our results are reported with a 50-50 ratio between coherence and memory traffic. Other ratios did not result in significantly different behaviors. We validate our network-level simulations by running a subset of PARSEC [5] in full-system simulation using gem5 [6] and booksim [17]. Each workload is run on 64 out-of-order 2GHz cores with 32KB of private L1 and 512KB private L2 cache per core. Each program is executed for over one billion instructions.

With the exception of the cost-performance analysis in Section 5.4, our performance results do *not* factor in benefits from speed binning individual chips in a disintegrated system. Our performance comparisons across different granularities of disintegration show only the cycle-level tradeoffs of the different configurations. If the binning benefits were included, then the overall performance benefits of our proposal would be even greater.

Power and area are modeled using DSENT [30]. Results are collected using a 45nm bulk/SOI low-V_t process node with 3.4mm^2 cores, with a worst-case CPU-layer core-to-core link length of 2.0mm (1.6mm CPU width, plus 0.4mm for μbump area). Frequency was swept to determine the maximum operating point for each topology. Long link lengths are faithfully modeled across each topology. μbump power was computed using the activity factors measured from simulation, a system voltage of 0.9V, and a μbump capacitance of 0.7fF [15].

5. EXPERIMENTAL EVALUATION

In this section, we explore network performance under different chip-size assumptions and compare latency and saturation throughput across a range of aligned and misaligned topologies. Our evaluation features synthetic and SynFull traffic patterns and full-system simulations. We also evaluate the power and area of the proposed topologies and present a unified cost-performance analysis.

5.1 Performance

Figure 10(a) shows the average packet latency for the Mesh, CMesh, DB, FT, and ButterDonut topologies assuming uniform random traffic with 50% coherence requests and 50% memory requests at a 0.05 injection rate. At a low injection rate, latency is primarily determined by hop count; as expected, the mesh performs the worst. As the number of cores per die decreases, the mesh performance continues to get worse as more and more packets must pay the extra hops on to the interposer to reach their destinations. For the other topologies, performance actually *improves* when going from a monolithic die to disintegrated organizations. This is because the static routing algorithms keep coherence messages on the CPU die; in a monolithic chip, the coherence messages miss out on the concentrated, low hop-count interposer networks. DB, FT, and ButterDonut all have similar performance due to similar hop counts and bisection bandwidths. At low loads, results are similar for other coherence/memory ratios as the bisection links are not yet a bottleneck. The results show that any of these networks are probably good enough to support a disintegrated chip at low traffic loads. Disintegrating a 64-core CPU into four chips provides the best performance, although further reduction in the sizes of individual chips (e.g., to 8 or even 4 cores) does not cause a significant increase in average packet latency.

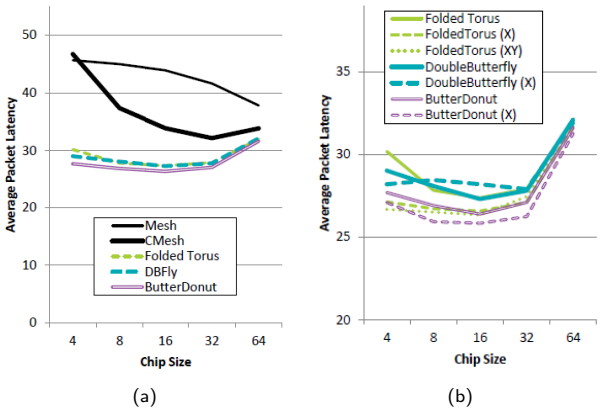

Figure 10: Average packet latency for different interposer NoC topologies. The x-axis specifies the individual chip size (16 = four 16-core chips, 64 = a single monolithic 64-core chip). Results are grouped by (a) aligned and (b) misaligned topologies. Traffic is split 50-50 between coherence and memory, with a 0.05 injection rate.

Figure 10(b) shows the average packet latency for FT, DB, and ButterDonut along with their misaligned variants. Note that the y-axis is zoomed in to make it easier to see the differences among the curves. The misaligned topologies generally reduce network diameter/hop count, and this is reflected by the lower latencies. The FT enjoys the greatest relative reductions in latency from misalignment.

The results in Figure 10 use synthetic uniform random traffic. Figure 11 shows average packet latency results when the system executes multi-programmed SynFull workloads. The results are for a system consisting of four 16-core CPU chips. The SynFull results show a greater difference in the performance between the different topologies as the workloads exercise a greater diversity and less uniform set of routes. Across the workloads, the misaligned ButterDonut(X)

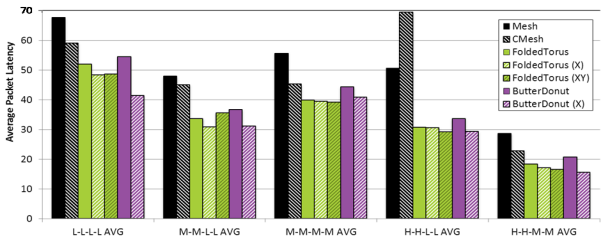

Figure 11: Average packet latency results for different multi-programmed SynFull workloads with varying network load intensities.

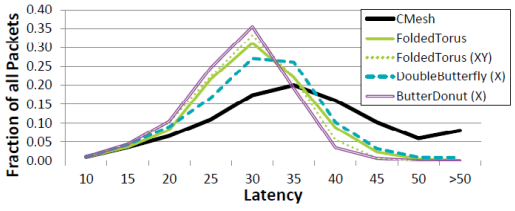

Figure 12: Distribution of message latencies (0.05 injection rate).

and FT(XY) consistently perform the best.

Figure 12 shows histograms for several interposer topologies' packet latencies for a system with four 16-core chips running uniform random traffic at a 0.05 injection rate. The CMesh suffers from both the highest average and highest variance in packet latency. The other higher-bandwidth solutions all have similarly low and tight latency distributions, with the ButterDonut performing the best. The low average hop counts of these NoCs keep average latency down, and the higher bandwidth reduces pathological traffic jams that would otherwise result in longer tails in the distributions.

Figure 13 shows the full-system results for several PARSEC benchmarks normalized to a mesh network. These results are consistent with the network latency results in Figure 10. For workloads with limited network or memory pressure, many topologies exhibit similar network latencies which results in little full-system performance variation across topologies. Blackscholes puts limited pressure on the memory system and therefore sees little difference across topologies. Ferret is more memory intensive and benefits from topologies with higher bandwidth to memory and lower diameter.

5.2 Load vs. Latency Analysis

The SynFull simulations hint at the impact of network load on the performance of the NoC. To generate latency-load plots, we revert back to synthetic uniform random traffic (50-50 cache and memory). We evaluate a system with four 16-core chips. Figure 14(a) and (b) show the latency-load curves for the same topologies shown earlier in Figure 10. For the aligned topologies, the CMesh quickly saturates as its four links crossing the bisection of the chip quickly become a bottleneck. The remaining topologies maintain consistently low latencies up until they hit saturation. The misaligned topologies generally follow the same trends, although

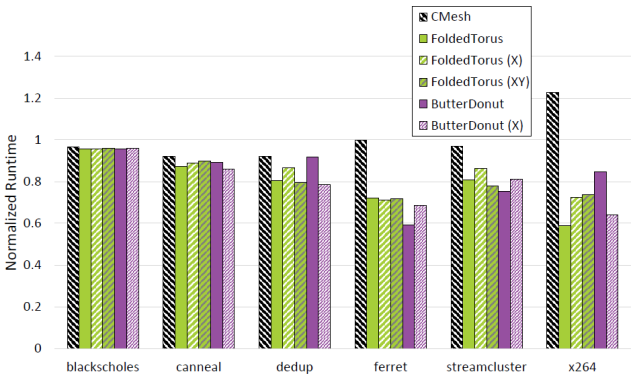

Figure 13: Normalized run-time for full-system simulation with a four 16-core chips.

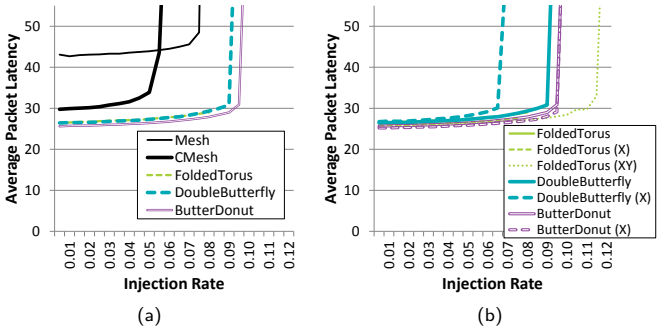

Figure 14: Latency and saturation throughput for (a) conventional NoC topologies and (b) misaligned topologies.

FT(XY) shows a substantially higher saturation bandwidth. The reason is that the Butterfly-derived topologies are optimized for east-west traffic to route memory traffic, and as a result are slightly imbalanced. Looking back at Figure 9, ButterDonut has 12 links across the east-west bisection, but only eight links going across the north-south cut. FT(XY) on the other hand has ten links across both bisections, and scales out better. The cost (see Table 3) is that FT(XY) has 25% more router nodes and 39% more links than ButterDonut.

Figure 15 further breaks down the load-latency curves into (a) memory traffic only, and (b) cache coherence traffic only. The results are fairly similar across the two traffic types, with the memory traffic having a slightly larger spread in average latencies due to the higher diversity in path lengths (memory requests always have to travel farther to the edges). The plots come from the same set of experiments, and therefore the saturation onset occurs at the same point in both graphs.

In general, from standard topologies to their misaligned variants, latency decreases and saturation throughput improves. Saturation throughput improvements come from taking pressure off of the east-west bisection links in the aligned topologies. Latency reductions come from shorter hop counts in these networks.

5.3 Power and Area Results

In Figure 16, we show power and area results normalized to a mesh for each topology. Longer links for some

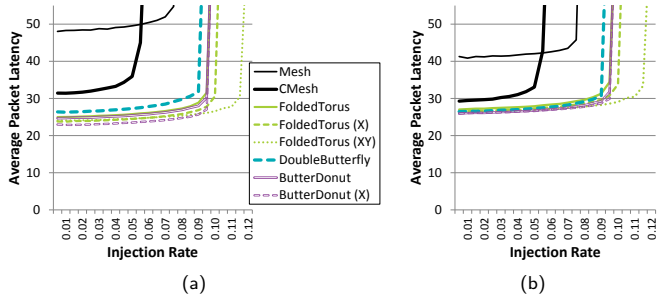

Figure 15: Latency and saturation throughput separated into (a) only memory messages, and (b) only core-to-core coherence messages.

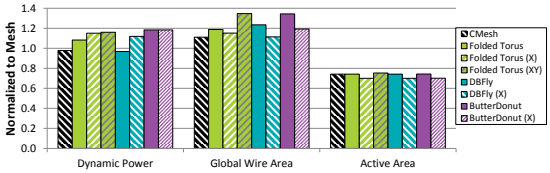

Figure 16: Power and area results in 45nm normalized to mesh.

topologies (e.g., FT(XY), DB and ButterDonut) lead to increased global wiring area; however increases are minor and do not exhaust the routing budget of the interposer. Likewise, longer links will consume more dynamic power (due to the presence of more repeaters); however the absolute power consumed by these topologies is small and will have a negligible impact on the overall thermals of the system. Although not shown, DSENT reports reduced clock frequency for all topologies with longer links; ButterDonut has the lowest operating frequency due to long links. However, it can still be clocked at 2.5GHz which is more than sufficient to saturate the DRAM bandwidth. Power results assume 1GHz frequency for all topologies.

5.4 Combining Cost and Performance

Higher levels of disintegration (smaller chips) can result in lower overall cost based on our analysis from Section 2 due to better yields and more chips per wafer. The smaller chips can potentially decrease interconnect performance due to fragmentation of the NoC, but the finer-grained binning also increases average CPU clock speeds. To put it all together, we consider two figures of merit (FOM) based on cost and performance, or "delay" for brevity.[11] The first, FOM$^{\times}$, is the product of delay and cost, which gives both factors similar influence. The second, FOM$^{\wedge}$, is delay$^{\text{cost}}$, which provides a greater emphasis on performance. The rationale for the performance-heavy FOM is that for high-end servers,

[11]Our delay metric is the average network latency adjusted by the average clock-speed improvement due to binning (Monte Carlo simulations were repeated for all chips sizes considered) assuming that only 30% of performance is impacted by the CPU frequency while the NoC frequency remains unchanged. 20% of total performance is impacted by network latency. The cost metric is the cost per 64-core system (considering yield and chips per wafer). As we are only comparing interposer-based systems, the cost of the interposer is not factored into the cost metric. Both delay and cost are normalized against the monolithic 64-core.

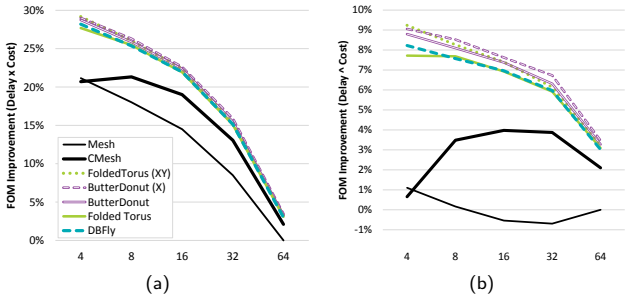

Figure 17: Figures of Merit based on (a) delay $\times$ cost, and (b) delay$^{\text{cost}}$. The x-axis specifies cores per chip (64=monolithic).

even relatively smaller performance differentiations at the high-end can translate into substantially higher selling prices and margins.

Figure 17(a) and (b) show FOM$^{\times}$ and FOM$^{\wedge}$, respectively. For the cost-delay product (FOM$^{\times}$), even though higher levels of disintegration cause average NoC performance to degrade, these are more than offset by the cost benefits combined with CPU frequency improvements. The exponential FOM$^{\wedge}$ tells a more interesting story. For a basic Mesh on the interposer, the performance loss due to disintegration actually hurts more than the cost reductions help until one gets down to using 8-core chips or smaller. The CMesh provides an initial benefit with two 32-core chips, but the lower bisection bandwidth of the CMesh is too much of a drag on performance, and further disintegration is unhelpful. The FOM$^{\wedge}$ results show that the remaining topologies are effectively able to ward off the performance degradations of the fragmented NoC sufficiently well that the combination of binning-based improvements and continued cost reductions allow more aggressive levels of disintegration.

With different FOMs, the exact tradeoff points will shift, but FOM$^{\wedge}$ in particular illustrates that simple disintegration (using Mesh, CMesh) alone may not be sufficient to provide a compelling solution for both cost and performance. However, interposer-based disintegration appears promising when coupled with an appropriate redesign of the interposer NoC topology.

6. OTHER ISSUES AND OPPORTUNITIES

6.1 Clocking Across Chips

The NoCs span multiple die and the interposer. Thus far we have modeled a fully-synchronous NoC where all nodes on all chips are clocked together. Considering die-to-interposer and die-to-die parametric variations and the possibility of implementing the interposer and individual chips in different technology generations, building a high-speed, low-skew, low-jitter global clock tree across multiple chips and the interposer is likely to be *very* challenging. While it makes sense to run each chip's portion of the NoC at the same clock speed (otherwise the slowest portion will likely become a bottleneck), from a physical-design perspective it is easier to clock chips independently. Thus, each chip's NoC portion operates in its own local timing domain, thereby

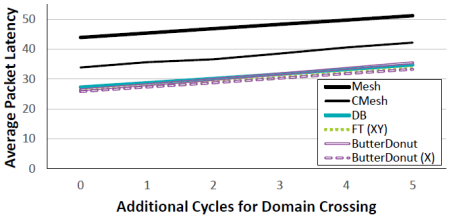

Figure 18: Impact of clock crossing latency on average packet latency with 16-core die.

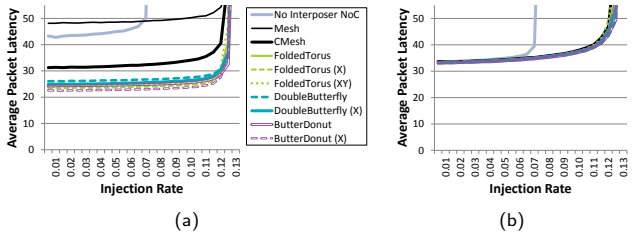

Figure 19: Load-latency curves for a monolithic 64-core chip stacked on the interposer, with traffic separated into (a) memory-only and (b) coherence-only traffic.

requiring domain-crossing synchronizing FIFOs when going from CPU chip to interposer, and vice versa. If all domains operate at roughly the same frequency, the domain-crossing FIFOs may be relatively low latency (consisting of phase-detection logic and some registers). If each portion runs at arbitrary clock speeds, then higher-latency synchronizers may be needed.

We repeated some of our experiments where each crossing from a CPU chip to the interposer, and back, must incur an additional 1 to 5 cycles for the FIFOs to synchronize between clock domains. Figure 18 shows the average latency for uniform random traffic on a four 16-core configuration. The additional latency affects all of the topologies in a uniform manner, effectively offsetting the performance in proportion to the penalty of crossing timing domains. This shifts the trade-off point in terms of how aggressively one can/should disintegrate a large chip. The higher degree of disintegration (smaller chips) increases the amount of traffic that goes chip-to-chip. Referring back to Figure 10, going from a monolithic chip to four 16-core chips decreases average packet latency by about 5 cycles. From Figure 18, this would suggest that a domain-crossing FIFO latency of 3-4 cycles would be tolerable to maintain the same performance levels as the original monolithic chip.

6.2 New Chip-design Concerns

This work assumes the disintegration of a multi-core chip into smaller die, with the implicit assumption that all die are identical. This creates a new physical design and layout challenge as each die must implement a symmetric interface; for example, a given die could be placed on the left side of the interposer or the right, and that same exact die must correctly interface with the interposer regardless of its mounting position. Conventional SoCs have no such requirements for their layouts. This extends beyond functional interfaces, to issues such as power delivery (the power supply must be reliable independent of chip placement) and thermal management (temperature may vary depending on chip location). Many of these challenges are more in the domain of physical design and EDA/CAD tool optimization, but we mention it here to paint a fair picture of some of the other challenges for interposer-based disintegration.

6.3 Software-based Mitigation of Multi-Chip Interposer Effects

Disintegration introduces additional challenges and opportunities for optimization at the software level. With a monolithic 64-core NoC, there is already non-uniformity in latency between various source-destination pairs as hop count increases, which can be exacerbated in a disintegrated system. Careful scheduling to place communicating threads on the same chips could reduce chip-to-chip traffic and alleviate congestion on the interposer. Similarly, careful data-allocation to place frequently used data in the 3D stacks close to the threads that use the data can minimize memory latency and cut down on cross-interposer traffic. Coordinating scheduling decisions to simultaneously optimize for both coherence and memory traffic could provide further benefits but would require support from the operating system. While we believe there are many interesting opportunities to involve the software layers to help mitigate the impacts of a multi-chip NoC organization, we leave it for future research.

6.4 Non-disintegrated Interposer NoCs

Prior work from Enright Jerger et al. did not contemplate using the interposer for chip disintegration [13]. However, they evaluate several different interposer NoC topologies for improving system interconnect performance. Figure 19 replicates that type of study by evaluating a monolithic 64-core chip stacked on an interposer, but we consider several of the topologies introduced in this paper. For the memory traffic in Figure 19(a), the results show that our new approach of misaligned topologies improves upon the prior topologies even for a non-disintegrated system. Our best performing topology, ButterDonut(X), provides a ∼9% improvement in average packet latency over the best prior topology (the aligned DB). Optimizing NoC designs for an interposer-based *monolithic* SoC was not the goal of our work, but the additional improvements are a welcome result.

7. RELATED WORK

In this section, we discuss related work in the areas of multi-die systems, hierarchical networks, 3D NoC designs, and multi-NoC designs. Brick and Mortar [21] integrates heterogeneous components on individual die connected through an interposer using a highly-reconfigurable network substrate that can accommodate different router architectures and topologies [20]. This flexibility leads to a high A_{crit} which negatively impacts interposer yield. Our system, with less heterogeneity, performs well with a regular topology which reduces areas to improve yield and cost.

Through interposer-based disintegration, we have essentially constructed a set of hierarchical topologies (locally with each die, and globally on the interposer). Hierarchical NoCs have been explored for *single-chip* systems. To improve latency for local references, bus-based local networks have been employed [10, 31] with different strategies for global connections. Buses and rings [2] are effective for a small number of nodes; although we use a mesh, alternative intra-die topologies could further reduce complexity and improve latency.

Our 2.5D NoCs bear some similarity to 3D die stacked NoCs [19, 22, 27, 35]. These NoCs are typically identical across all layers and assume arbitrary or full connectivity within a layer. Our system does not allow arbitrary connectivity within the core layer as it consists of multiple die. Considering the system constraints for the hybrid memory cube, Kim *et al.* route all traffic through a memory-centric network leading to longer core-to-core latencies [18]; similarly, we route some core-to-core traffic through the interposer but develop optimized topologies to offset negative performance impacts.

When using multiple networks to improve performance [4, 11, 13, 23, 32, 34, 33], traffic can be partitioned across these networks uniformly or functionally. We take a functional partitioning approach: memory traffic uses the interposer, while core traffic stays on the local core mesh; however, due to the presence of multiple separate die for our cores, our partitioning is not strict (some core traffic *must* use the interposer network to reach remote cores).

8. CONCLUSIONS

We explored the potential for leveraging interposers that are already present for memory integration as a substrate for reintegration of a disintegrated multi-core chip. Our analysis suggests that disintegrating complex chips may be a promising approach for improving yield and reducing costs. However, the economic benefits may only be realized if the system design can overcome the performance effects of the disintegrated architecture. We introduce *minimally-active interposers* as a potentially practical (cost-effective) way to build NoC logic on the interposer. We examine the traffic impact of chip-to-chip coherence traffic across the interposer, which lead to the development of the new, interposer-specific, concept of *misaligned* NoCs. Handling both coherence and memory traffic across the interposer lead to the creation of the *ButterDonut* topology that combines the strengths of the previously proposed Double Butterfly with a classic Folded Torus approach. The combination of these contributions show that the performance impact of multi-chip interposer systems can be addressed, which leads us to conclude that interposer-based disintegration is a promising approach for building future SoCs.

Acknowledgments

The authors would like to thank the anonymous reviewers for their suggestions on how to improve this work. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada and the Canadian Foundation for Innovation. AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

9. REFERENCES

[1] Advanced Micro Devices, Inc., "AMD Ushers in a New Era of PC Gaming with Radeon^{TM} R9 and R7 300 Series Graphics Line-Up including World's First Graphics Family with Revolutionary HBM Technology," June 16, 2015, press Release from http://www.amd.com.

[2] R. Ausavarungnirun, C. Fallin, X. Yu, K. Change, G. Nazario, R. Das, G. Loh, and O. Mutlu, "Design and evaluation of hierarchical rings with deflection routing," in *Proceedings of the International Symposium on Computer Architecture and High Performance Computing*, 2014.

[3] M. Badr and N. Enright Jerger, "SynFull: Synthetic traffic models capturing a full range of cache coherence behaviour," in *Intl. Symp. on Computer Architecture*, June 2014.

[4] J. Balfour and W. J. Dally, "Design tradeoffs for tiled CMP on-chip networks," in *Intl. Conf. on Supercomputing*, 2006.

[5] C. Bienia, "Benchmarking Modern Processors," Ph.D. dissertation, Princeton University, 2011.

[6] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "gem5: A Multiple-ISA Full System Simulator with Detailed Memory Model," *Computer Architecture News*, vol. 39, June 2011.

[7] B. Black, "Die Stacking is Happening," in *Intl. Symp. on Microarchitecture*, Davis, CA, December 2013.

[8] L. Chen and T. M. Pinkston, "Worm-bubble flow control," in *19th Intl. Symp. on High Performance Computer Architecture*, February 2013.

[9] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2003.

[10] R. Das, S. Eachempati, A. K. Mishra, V. Narayanan, and C. R. Das, "Design and evaluation of a hierarchical on-chip interconnect for next-generation cmps," in *Intl. Symp. on High Performance Computer Architecture*, 2009.

[11] R. Das, S. Narayanasamy, S. K. Satpathy, and R. Dreslinski, "Catnap: Energy proportional multiple network-on-chip," in *Intl. Symp. on Computer Architecture*, 2013.

[12] Y. Deng and W. Maly, "Interconnect Characteristics of 2.5-D System Integration Scheme," in *Intl. Symp. on Physical Design*, Sonoma County, CA, April 2001, pp. 171–175.

[13] N. Enright Jerger, A. Kannan, Z. Li, and G. H. Loh, "NoC architectures for silicon interposer systems," in *47th Intl. Symp. on Microarchitecture*, Cambridge, UK, December 2014, pp. 458–470.

[14] M. Hackerott, "Die Per Wafer Calculator," Informatic Solutions, LLC, 2011.

[15] Institute of Microelectronics, "Process Design Kit (PDF) for 2.5D Through Silicon Interposer (TSI) Design Enablement & 2.5D TSI Cost Modeling," August 2012.

[16] JEDEC, "High Bandwidth Memory (HBM) DRAM," http://www.jedec.org/standards-documents/docs/jesd235.

[17] N. Jiang, D. U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, J. Kim, and W. J. Dally, "A detailed and flexible cycle-accurate network-on-chip simulator," in *Intl. Symp. on Performance Analysis of Systems and Software*, 2013.

[18] G. Kim, J. Kim, J.-H. Ahn, and J. Kim, "Memory-centric system interconnect design with hybrid memory cubes," in *Intl. Conf. on Parallel Architectures and Compilation Techniques*, 2013.

[19] J. Kim, C. Nicopoulos, D. Park, R. Das, Y. Xie, N. Vijaykrishnan, M. S. Yousif, and C. R. Das, "A Novel Dimensionally-Decomposed Router for On-Chip Communication in 3D Architectures," in *34th Intl. Symp. on Computer Architecture*, San Diego, CA, June 2007.

[20] M. M. Kim, J. D. Davis, M. Oskin, and T. Austin, "Polymorphic on-chip networks," in *Intl. Symp. on Computer Architecture*, 2008.

[21] M. M. Kim, M. Mehrara, M. Oskin, and T. Austin, "Architectural implications of brick and mortar silicon manufacturing," in *Intl. Symp. on Computer Architecture*, 2007.

[22] F. Li, C. Nicopoulos, T. Richardson, Y. Xie, V. Narayanan, and M. Kandemir, "Design and Management of 3D Chip Multiprocessors Using Network-in-Memory," in *33rd Intl. Symp. on Computer Architecture*, Boston, MA, June 2006, pp. 130–141.

[23] P. Lotfi-Kamran, B. Grot, and B. Falsafi, "NOC-Out: Microarchitecting a Scale-Out Processor," in *Intl. Symp. on Microarchitecture*, Vancouver, BC, December 2012, pp. 177–187.

[24] S. Ma, Z. Wang, Z. Liu, and N. Enright Jerger, "Leaving one slot empty: Flit bubble flow control for torus cache-coherent NoCs," *IEEE Transactions on Computers*, vol. 64, pp. 763–777, March 2015.

[25] N. Madan and R. Balasubramonian, "Leveraging 3D Technology for Improved Reliability," in *40th Intl. Symp. on Microarchitecture*, Chicago, IL, December 2007, pp. 223–235.

[26] M. O'Connor, "Highlights of the High-Bandwidth Memory (HBM) Standard," in *Memory Forum Workshop*, June 2014.

[27] D. Park, S. Eachempati, R. Das, A. K. Mishra, Y. Xie, N. Vijaykrishnan, and C. R. Das, "MIRA: A Multi-layered On-chip Interconnect Router Architecture," in *Intl. Symp. on Computer Architecture*, Beijing, China, June 2008, pp. 251–261.

[28] V. Puente, C. Izu, R. Beivide, J. A. Gregorio, F. Vallejo, and J. M. Prellezo, "The adaptive bubble router," *Journal of Parallel and Distributed Computing*, vol. 64, no. 9, pp. 1180–1208, 2001.

[29] C. H. Stapper, "The Effects of Wafer to Wafer Defect Density Variations on Integrated Circuit Defect and Fault Distributions," *IBM Journal of Research and Development*, vol. 29, pp. 87–97, January 1985.

[30] C. Sun, C.-H. O. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L.-S. Peh, and V. Stojanovic, "DSENT - a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in *NOCS*, May 2012.

[31] A. N. Udipi, N. Muralimanohar, and R. Balasubramonian, "Towards scalable, energy-efficient, bus-based on-chip networks," in *Intl. Symp. on High Performance Computer Architecture*, 2010.

[32] S. Volos, C. Seiculescu, B. Grot, N. K. Pour, B. Falsafi, and G. D. Micheli, "CCNoC: Specializing On-Chip Interconnects for Energy Efficiency in Cache Coherent Servers," in *6th NOCS*, Lyngby, Denmark, May 2012, pp. 67–74.

[33] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. Brown, and A. Agarwal, "On-chip interconnection architecture of the Tile Processor," *Micro, IEEE*, vol. 27, no. 5, pp. 15 –31, Sept.-Oct. 2007.

[34] Y. J. Yoon, N. Concer, M. Petracca, and L. Carloni, "Virtual channels vs. multiple physical networks: a comparative analysis," in *Design Automation Conference*, 2010.

[35] A. Zia, S. Kannan, G. Rose, and H. J. Chao, "Highly-scalable 3D Clos NoC for Many-core CMPs," in *NEWCAS Conference*, Montreal, Canada, June 2010, pp. 229–232.