SiPterposer: A Fault-Tolerant Substrate for Flexible System-in-Package Design

Pete Ehrett Todd Austin Valeria Bertacco
University of Michigan, Ann Arbor
{wpehrett, austin, valeria}@umich.edu

***Abstract*—As Moore's Law scaling slows down, specialized heterogeneous designs are needed to sustain computing performance improvements. Unfortunately, the non-recurring engineering (NRE) costs of chip design—designing interconnects, creating masks, *etc.*—are often prohibitive. Chiplet-based disintegrated design solutions could address these economic issues, but current technologies lack the flexibility to express a rich variety of designs without redesigning the communication substrate. Moreover, as the number of chiplets increases, yield suffers due to 2.5D assembly defects. This work addresses these problems by presenting a flexible communication fabric that supports construction of arbitrary network topologies and provides robust fault-tolerance, demonstrating near-100% chip assembly yield at typical bonding defect rates. We achieve these goals with less than 3% additional power and zero exposed latency overhead for various real-world applications running on an example SiP.**

I. Introduction

Application-specific hardware design is widely regarded as one of the most promising methods for continuing improvements in chip performance and power efficiency, particularly when Moore's Law can no longer be relied upon as a primary driver of advancement [1]. Unfortunately, creating custom hardware requires a huge investment in development resources. First, per-design NRE costs are very high – a set of 16nm masks for a new design, for example, may cost nearly $6 million [2]. For high-volume chips, these costs can be amortized effectively, but they quickly become prohibitive for smaller-volume designs. Second, creating a custom chip often requires integrating common logic alongside custom, application-specific logic in a single monolithic die. Even though some soft IP components may be reused across many custom chips, which saves design time, there is no associated economy of scale in the fabrication process because each chip is still manufactured monolithically – thus, costs remain high.

Recently, research on reusable hardware design – building system components that can be reused across a wide variety of application-specific designs – has begun to address these problems. Some works target NRE costs, proposing architectures that compose application-specific hardware from arrays of generic functional units on a mass-produced chip [3]. Others focus on manufacturing costs, proposing systems in which chips are fabricated in small blocks, called 'chiplets', which are bonded in 2.5D to an interposer that delivers power, clock, *etc.* and provides inter-chiplet data wiring [4]. This System-in-Package (SiP) concept could enable solutions where resources used in many different designs are fabricated at high volume (hence, at lower per-unit cost) and integrated with smaller pieces of chip-specific custom logic on an interposer. Rather than incurring the high costs of designing and building a large monolithic die for each new chip, it would instead be possible to design and build only the smaller pieces of custom logic, saving time and money. Moreover, the chiplets in a given SiP need not even be manufactured at the same technology node, which could enable novel cost/performance optimizations.

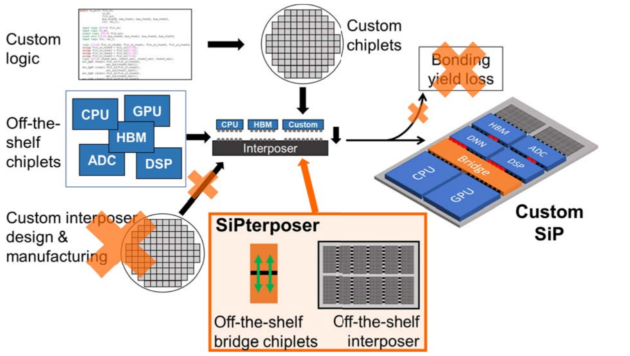


Fig. 1: **SiPterposer** proposes an off-the-shelf solution to reduce the cost of custom SiPs by eliminating the need for custom interposer designs and slashing chiplet bonding yield loss.

However, these methods require a robust means of integrating the components used to compose a new application-specific design – and integration alone can represent a quarter of the total NRE cost for a custom chip [5]. Thus, for reusable hardware design to succeed at reducing custom chip costs, it is crucial to consider not just the functional units or chiplets, but also the cost of the underlying integration/communication layer. In particular, cost-effective chiplet-centric design requires an interposer with three key properties: **generality**, **flexibility**, and **resilience**. Generality allows the interposer to be mass-produced at low cost. Flexibility is important because the substrate must support a wide range of applications – including arbitrary chiplets and interconnect topologies – without costly redesign. Resilience matters because the chiplet-to-interposer bonding process is quite defect-prone [6], lowering yields and raising costs if not handled carefully.

To that end, we propose a novel integration fabric, called SiPterposer, based on a generic passive interposer structure and the use of off-the-shelf bridge chiplets to create desired interconnect topologies, as outlined in Fig. 1. This work discusses SiPterposer's economic viability, demonstrates its capacity to achieve the design goals of generality, flexibility, and resilience, and evaluates its interconnect performance and system overheads. Our key contributions are:

- A **generic**, fully-passive interposer structure that may be configured at chip assembly-time to generate any custom interconnect topology, eliminating the need for custom interposer design and fabrication when creating a custom SiP.
- A set of design-independent, mass-producible 'bridge' chiplets that can be used to connect distinct regions of a SiP. These, along with the interposer structure, enable **flexible** assembly-time interconnect formation.
- Designs and analyses of three low-overhead ECC methods to improve **resilience** of chiplet-to-interposer bonds.

978-3-9819263-2-3/DATE19/©2019 EDAA

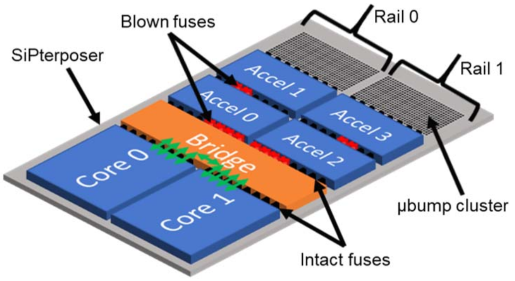


Fig. 2: **Assembly-time customization with SiPterposer.** The bridge chiplet (Fig. 4b) connects two cores on different rails (denoted by green arrows). The blown fuses enable the cores to communicate in isolation from accelerator pairs 0/1 and 2/3, and vice versa.

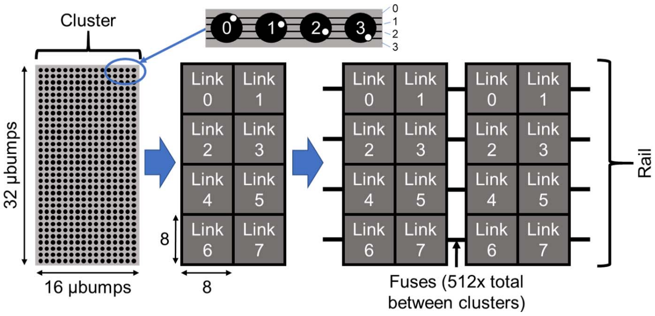


Fig. 3: **µbump clusters** comprise a grid of 512 µbumps, each attached to a distinct interposer wire. The top shows a simplified view of the internal structure; the white dots indicate which µbump connects to each wire. Each wire may be fused between each cluster on a rail.

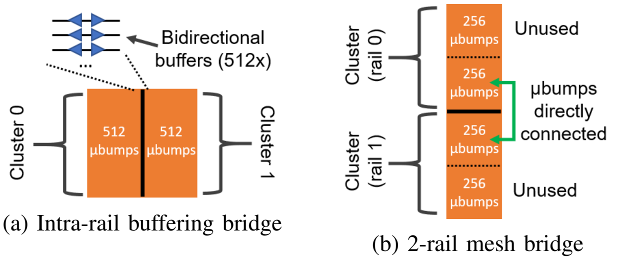
Our evaluation finds that SiPterposer has significant economic advantages over traditional 2.5D methods while providing increased reliability and assembly-time flexibility, which could substantially lower the cost of custom silicon.

II. Related Work

Recent work in the SiP space has detailed the economic and technological advantages of building large chips in a 'disintegrated' fashion – dividing them into multiple independently-fabricated chiplets and then integrating them on an interposer [7]. Most such work assumes the use of a custom interposer, and those that attempt to provide greater flexibility either continue to impose some design restrictions or else require active logic within the interposer [4].

The materials and mechanical reliability of microbumps (µbumps) in chiplet/interposer bonds have been studied extensively [8]. [9] presented an empirical study of defect rates across an image sensor bonded to a substrate using a large array of fine-pitch µbumps. There has also been extensive prior work on correcting bonding defects between layers of a 3D chip, most of which focuses on replacing defective through-silicon vias (TSVs) with redundant ones [10]. [11] proposed adding ECC to correct TSV link defects. However, all these are designed either to minimize the number of TSVs in a 3D system or to provide error correction far stronger than SiPs demand. Few, if any, address µbump defects in general or 2.5D integration in particular. By contrast, this paper takes a targeted approach by accounting for 2.5D-specific design considerations and realistic defect models.

Finally, in the NoC space, interconnect reliability has been explored broadly, with various works discussing routing around failed components [12] or applying ECC to correct transient faults and crosstalk [13]. To our knowledge, however, these methods have not been leveraged in the SiP space, and no prior work has applied ECC to tolerate SiP assembly defects.



(a) Intra-rail buffering bridge

(b) 2-rail mesh bridge

Fig. 4: **Bridge chiplet examples.** (a) is an active bridge that connects two adjacent clusters on the same rail via bidirectional buffers; if attached across a set of blown fuses, it can act as a repeater for long interconnect paths. (b) is a passive bridge suitable for constructing a small mesh by directly connecting parts of two adjacent rails; this pattern can extend across additional rails to enable larger designs.

III. System Overview

SiPterposer structure. To achieve our goals of generality, flexibility, and resilience, we propose constructing SiPs using a generic, fully-passive interposer based on a simple internal wiring pattern consisting of long, straight data *rails* (see Fig. 2). Each rail consists of a large number of parallel wires that span the entire width of the interposer but are not directly connected either to each other or to wires in any other rail. µbumps are connected to each wire at regular intervals, and a group of µbumps, one per wire in the rail, comprises a *cluster*. A chiplet may span and connect to one or more of the clusters, either on the same rail or on different rails.

For our analysis throughout the remainder of this paper, we define each cluster to support a 512-bit data connection. Because µbumps are much larger than interwire distances, they are offset slightly from the centers of the wires to which they attach, forming a two-dimensional grid (Fig. 3). In our design, the µbumps are a typical 20µm wide with 40µm pitch [8] and are configured in a 16x32 grid, resulting in cluster dimensions of ~0.7mm by ~1.4mm. Further, we partition each cluster into eight 64-bit logical *links* (akin to one node in a full-mesh network with 64-bit full-duplex links). Different sets of chiplets may communicate simultaneously by using disjoint subsets of the available links. Fig. 3 illustrates this layout/partitioning scheme.

Portions of interposer wiring between each cluster of µbumps may be separated during the assembly process, causing them to act as small electronic *fuses* (Figs. 2 and 3). Although we refer to these wire segments as 'fuses', there is no need to add specialized fuse components or other discontinuities to the fabric; existing technology can fuse link wires at the pitch we propose [14]. At assembly-time, blowing all fuses at a specific point in a rail can completely disconnect a set of chiplets from others. Alternatively, blowing fuses to disconnect only a subset of the logical links within a rail would allow, for example, system-wide broadcasts over one link, while other links carry out communication between adjacent chiplets. Blowing fuses can also enable a chiplet to act as a repeater (Fig. 4a) or as a node in a mesh with different parts of a cluster connecting different network edges (Fig. 5).

Other electrical- and protocol-level considerations either can be handled with standard techniques or are design-dependent.

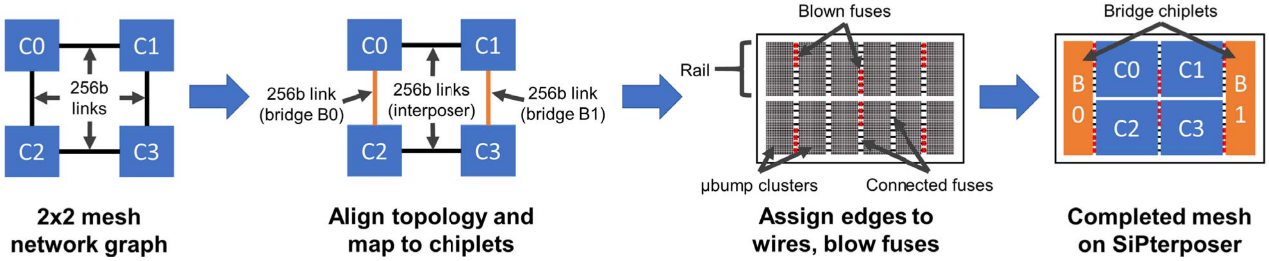


Fig. 5: **Construction of a 2x2 mesh with SiPterposer.** Each line between µbump clusters represents 64 fuses on the interposer. East/West edges (C0-C1 and C2-C3) use half the available interposer wires. North/South edges (C0-C2 and C1-C3) use passive bridge chiplets B0 and B1 (see Fig. 4b) in conjunction with the other half of the interposer wires. The blown fuses electrically isolate the edges from one another.

Power distribution, for instance, may use typical VLSI [15] and SiP [16] methods, while interfaces between clock domains are handled within the chiplets using existing NoC methodologies [17]. The proposed physical structure may also be tessellated to produce a different interposer size without costly redesign. Since SiPterposer is based on a passive interposer, communication protocols are defined and handled chiplet-side. These can range from AMBA buses to packet-switched networks to fully-custom designs. For the rest of this work, all chiplets are assumed to use a packet-based network protocol.

Bridge chiplets. In addition, we propose the use of dedicated, generic *bridge chiplets* which, in conjunction with the interposer structure, enable assembly-time customization of the system's connectivity. These bridge chiplets can be designed in a handful of different patterns and then mass-produced. Simple units (*e.g.*, Fig. 4b) include only passive wiring; these 'bridge' electrical gaps by directly connecting wires in one data rail to another. Other bridges may be active devices, deployed horizontally across portions of a rail separated by blown fuses to act as a buffer in the middle of that rail (*e.g.*, Fig. 4a), create clock boundaries, or be full-blown routing devices. To illustrate the reusability of a small set of bridges across many different designs, we limit our selection for the remainder of this work to Fig. 4b's passive unit (and its derivatives).

Arbitrary topology construction. By blowing fuses in the interposer wiring and connecting bridge chiplets across disconnected regions, we can create any arbitrary interconnect topology that a chip may require, as follows:

1) Align the network graph to a Manhattan layout.
2) Rotate the graph so as many edges as possible run along the axis of SiPterposer's internal wiring, lowering overhead by reducing the number of bridge chiplets required.
3) Map the nodes in the graph to chiplets and arrange them as blocks atop SiPterposer's µbump clusters according to their logical layout in the network graph.
4) For each edge in the graph:
    a) If possible, map the edge to an unused subset of interposer/bridge wires already in the design.
    b) If too few bridge wires: (i) add or extend a bridge, or (ii) time-multiplex access via chiplet-side logic, akin to virtual channels (VCs) in a NoC.
    c) If too few interposer wires: (i) move the nodes connected by the edge to another rail (adjusting previously mapped edges accordingly), or (ii) time-multiplex access.
    d) Blow fuses at the endpoints of the new link, to reduce wire loading and permit other parts of the newly separated interposer wires to be used freely for other edges.

As an example, we illustrate building a simple 2x2 mesh using this process, in Fig. 5.

Importantly, because SiPterposer's electrical structure does not require an active interposer, it may be implemented on any desired substrate material – Si, organic, glass, *etc.* (our evaluation assumes Si). Furthermore, it may easily be layered with other chiplet placement or system configuration techniques (*e.g.*, [18]) as part of a holistic design methodology.

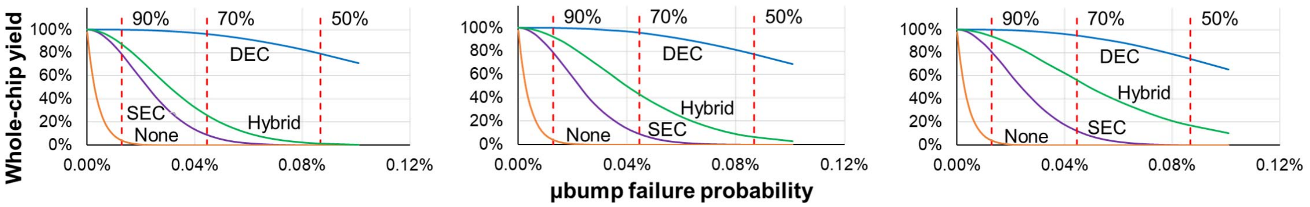
Defect-tolerance. The overall SiP concept is quite promising, but it also introduces a new point of failure into chip fabrication via the chiplet-to-interposer assembly process. Prior work estimates this process yield at 99%-99.5% per 1024-µbump chiplet – a loss that, even in a system with relatively few chiplets, can be responsible for as much as 26% of the total manufacturing cost [6]. Rather than trying to reduce the incidence of these defects, we instead propose *tolerating* them by adding a module to each chiplet to provide lightweight ECC on each interposer bond. Although this requires both additional wires to carry parity information and chiplet-side encode/decode logic, it requires no active logic on the interposer, preserving its simplicity and its low manufacturing cost.

In our design, each µbump cluster provides a 512-bit data connection to an interposer rail, partitioned into eight 64-bit logical links. We further divide each 64-bit link into four 16-bit sublinks and then apply a form of ECC to each sublink, either Hamming single-error-correction (SEC) or Bose-Chaudhuri-Hocquenghem double-error-correction (DEC). SEC requires 5 parity bits per sublink (672 total µbumps per cluster), while DEC requires 10 parity bits per sublink (832 µbumps per cluster). The sublinks within each logical link are interleaved to better resist physically-adjacent defects [13].

The nature of chip warpage during die bonding [8] inspired us to also design a third, defect-pattern-aware coding method. Since warpage-induced mechanical stress causes bonding defects to occur most often at a chip's edges, we suggest a hybrid concentric coding structure, with four logical links in the center of the chiplets (using SEC) and four links along their more-defect-prone edges (using DEC). As our evaluation shows, this hybrid approach provides a better yield-overhead balance than either SEC-only or DEC-only.

IV. Evaluation

We evaluated SiPterposer's defect-tolerance and overall performance by simulating assembly of a hypothetical 48-chiplet system. First, we determined the whole-chip assembly yield for varying defect rates, ECC, and bonding defect patterns. Second, we synthesized our ECC hardware and used the results with whole-system models to simulate SiPterposer's impact on network performance and overall chip power/area overhead.



(a) Uniform

(b) Edge-weighted

(c) Empirical

Fig. 6: **SEC, DEC, and Hybrid coding performance on uniform, edge-weighted, and empirical defect patterns.** The vertical lines on each graph denote specific per-chiplet bond yields (a 90%, 70%, *etc.*, chance that there are zero faulty µbump bonds between a single chiplet and the interposer), corresponding to particular per-µbump failure probabilities on the x-axis.

A. *Chiplet Bond Resilience with ECC*

We began our evaluation of defect-tolerance by creating a worst-case scenario for the error correction schemes we propose, configuring our 48 chiplets into a fully-connected system. Each chiplet uses exactly one µbump cluster, and every µbump on a given chiplet is directly connected to the corresponding µbump on every other chiplet (assuming no defects). We defined a failed chip as one in which there exists an uncorrectable fault in any link between any pair of chiplets. We assumed known-good-dies in our simulations in order to isolate the effects of coding on µbump bonding defects.

To model assembly defects, we assigned each µbump bond an independent failure probability based on its physical position within a cluster, relative to one of three potential defect patterns. The first pattern, **uniform**, assumes that each µbump bond has an equal chance of failure. The second pattern, **edge-weighted**, incorporates the effect of die warpage via a linear increase in bond failure probability with a µbump's distance from the center of a chiplet, from a baseline at the center to 10x that value at the outermost corner. The third pattern, **empirical**, simulates real-world failures using data derived from [9].

For each coding method and defect pattern, we conducted Monte Carlo simulations (100K trials) of chip assembly to calculate whole-chip assembly yields while sweeping the base per-µbump failure probability. For the edge-weighted and empirical defect patterns, we normalized the failure probability of an overall chiplet bond to that of a chiplet bond having a uniform defect pattern with the same base per-µbump failure probability. Fig. 6 compares each coding method vs. a system with no error correction. In general, there is little effect on chip assembly yield with increasing defect pattern complexity, from uniform to edge-weighted to empirical. Hybrid coding is an exception: its defect-tolerance increases significantly on the edge-weighted defect pattern. It performs even better with an empirical defect distribution, since this pattern's defects are even more heavily biased towards the edges of each chiplet.

B. *Interconnect Performance*

To evaluate SiPterposer's network performance and electrical characteristics, we modeled two systems-in-package – one synthetic, one inspired by real-world SoCs – and evaluated their overheads vs. SoC and traditional-SiP equivalents.

Mesh network, synthetic traffic. Our first, synthetic, system is an 8x6 full-mesh network containing 48 identical 4mm^2 chiplets with 1W nominal power consumption, in which each chiplet has a 64-bit full-duplex data connection to each of its neighbors. This is representative of a homogeneous multicore chip, and constitutes a worst-case scenario for SiPterposer, since the large number of links required in the topology increases the number of bridge chiplets needed, which, in turn, accentuates our proposed system's overheads.

TABLE I: Router synthesis results (with ECC)

	Baseline	SEC	DEC	Hybrid
Power (mW)	3.09	4.92	13.25	9.09
Area (µm^2)	2108	4973	15752	10363
Area overhead - 8x6 mesh	–	0.07%	0.34%	0.21%
Area overhead - SoC	–	0.04%	0.22%	0.13%

TABLE II: Network params

Network clk	2GHz
Routing fn	dor (mesh), min (SoC)
VCs/buffer size	3/8
Router pipeline	4 cycles
Link traversal	1 cycle
Pkt size (flits)	16 (mesh), 1 (SoC)

TABLE III: Chipset params

Chiplet area	72mm^2
Chiplet pwr	4W total (56mW/mm^2)
CPU clk	900MHz
Chiplet clk	500MHz
Memory	2GB/500MHz
Vid/img size	1080p

We established two baseline systems: a monolithic SoC, and a traditional SiP using a fixed-topology passive interposer. The latter is necessary because the novelty of this work lies in the methods we propose for constructing SiPs; thus, it is important to evaluate SiPterposer against both traditional SoCs and non-reusable interposer designs. All active logic in each system is assumed to use a 45nm process node; interposers use 65nm global wire widths. Since the interposers and bridges are passive, the three systems differ only in encode/decode logic overhead and link wire dimensions.

First, we constructed HDL models of our SEC and DEC ECC modules. We then integrated these into an open-source NoC router model [19] and synthesized the modified router with Synopsys Design Compiler using an IBM 45nm library to determine the ECC modules' area and power overheads; these are summarized in Table I. As the ECC modules were inserted directly before/after the input/output buffers, off the critical path of the router, they exposed no additional timing overhead to the design. Hybrid coding would entail area and power consumption exactly between the SEC and DEC routers.

Next, we evaluated network performance and power overhead using a combination of BookSim [20], ORION 2.0 [21], and LTSPICE. Since adding ECC to the routers introduced no additional latency, and because we assume equal link widths across the three example systems, performance overhead could only come from added delay from longer inter-chiplet links on SiPterposer. Using LTSPICE with ORION's wire models and [22]'s µbump models, we computed the delay of the longest link as 62.2ps, small enough to not impact network timing.

Finally, we constructed a model of each interconnect (SoC, SiP, and SiPterposer) in BookSim and analyzed the link wire power with ORION for uniform random traffic at varying

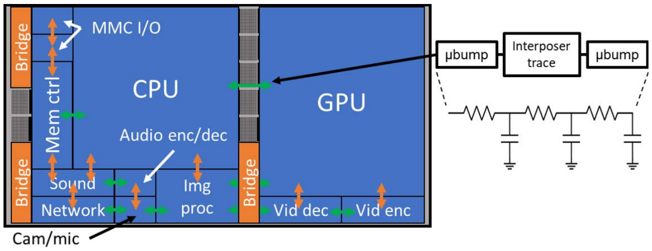


Fig. 7: **Realistic-chipset layout on SiPterposer.** All inter-chiplet connections are dedicated, 64-bit, full-duplex links. Green arrows denote links via the interposer alone. Orange arrows show links that require bridges (Fig. 4b). μbumps, interposer traces, and bridges use RC models derived from [21] and [22].

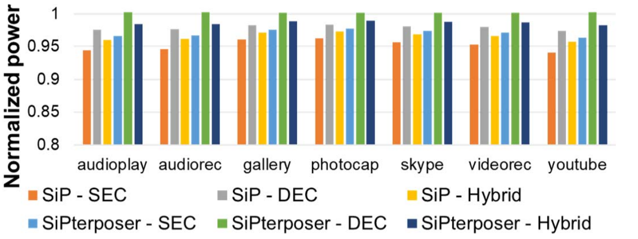


Fig. 8: **Whole-system power overhead on real-world applications,** normalized to SoC. The impact of increased wire length on SiPterposer vs. a traditional SiP is minimal. SiPterposer actually *saves* power in most cases vs. a monolithic SoC since its larger interconnect wires are more energy-efficient.

injection rates. Network parameters are given in Table II. We then combined these results with the synthesized router designs and our prior chiplet power assumption to determine the total power overhead (vs. SoC) of each system. This ranged from 0.98x for the SiP (using SEC) at packet injection rate r=0.001 to 1.10x for SiPterposer (using DEC) at r=0.0225.

Realistic chipset, real-world traffic. Our second evaluation framework approximates a real-world mobile chipset with a mesh-like network topology. Using die shot analyses [23] and other SoC power/area data [24] as guides, we defined a 72mm^2 system with 4W average power consumption, comprising 12 core chiplets (see Fig. 7 and Table III). As before, we use 45nm technology for chiplets/SoC, and 65nm for interposers.

In this system, each chiplet has one radix-5 router with 64-bit full-duplex links, similar to that of the previous section (see Table II), which may communicate through μbump clusters at the corners of the chiplet (this permits shorter inter-chiplet links in both the baseline SiP and SiPterposer, but could have worse yield due to die warpage). Adapting this system for SiPterposer requires three bridges (see Fig. 7). Our analysis is based on closed-loop simulation with GemDroid [25] and BookSim, plus ORION's link power models, with GemDroid's IP block power models normalized to a 4W 45nm SoC. We evaluated real-world performance on various application traces from the Android Emulator [26]. Again, since integrating ECC into our router requires no additional latency and the delay contribution of increased wire length is small (222.3ps at maximum, well within our 0.5ns link traversal budget), the only overhead we need to consider is wire power. These results, in whole-system context, are detailed in Fig. 8. Even using DEC, SiPterposer incurs no more than 0.2% power overhead vs. the baseline SoC and 2.9% vs. a traditional SiP.

V. Discussion

Resilience and performance. At currently achievable 2.5D chiplet bonding defect rates (99%+ per-bond yield), any of our proposed ECC methods can achieve assembled-chip yields near 100%. Differences in resilience become more apparent at higher defect rates. SEC drops off quickly below a 90%-95% per-chiplet bond yield, but DEC remains strong even with bond yield as low as 50%-70%, at the cost of a traffic-dependent increase in power consumption. Overheads for any of our methods are quite low, particularly with real-world application traffic, considering the substantial yield gains they provide.

Hybrid coding, especially with our empirical defect pattern, is an intriguing case. It has chip yield substantially better than SEC at low to moderate defect rates, but at high defect rates, yield drops far more quickly than with DEC. In addition, the hybrid structure has power and area overhead exactly in between SEC-only and DEC-only. Applying the principle of routing around failed network components [12] further improves the case for hybrid coding. As a proof-of-concept, we examined the average number of non-defective links in a SiP with each ECC method and used the result as a proxy for the total bisection bandwidth available to the system, as shown in Fig. 9. Here, hybrid coding performs closely to DEC-only, but with much lower power and area overhead.

Note that our hybrid coding technique is applied to a single μbump cluster; for chiplets spanning multiple clusters, this technique will become less effective as its structure less closely mirrors the warpage pattern created on the chiplet at any individual cluster. However, a similar principle could be applied at the inter-cluster level – *i.e.*, using DEC for whole clusters towards the edges of a chiplet and SEC for clusters near its center. We leave further exploration of this and other aspects of SiP defect-pattern-aware coding for future work.

To understand the impact of SiPterposer's flexibility on chip performance, we compared its power consumption to that of a traditional SiP with a custom interposer. With synthetic traffic on our 8x6 mesh, we observed a roughly linear increase in power consumption with increasing network load: 1%-7% with no ECC, 1%-9% with SEC, and 1%-11% with DEC (reaching a limit as the network saturates). As previously noted, though, this mesh topology with uniform random traffic represents a worst-case-scenario for SiPterposer. More-realistic loads on our example chipset showed overheads on the low end of these ranges, averaging 1.5% with no ECC, 2.0% with SEC, and 2.3% with DEC. These values are quite reasonable considering the degree of flexibility SiPterposer offers—and the impact of that flexibility on chip cost, as discussed in the next section.

Finally, using LTSPICE, we performed rudimentary analysis of the longest interposer-bridge-μbump links within each system to understand how they might perform in mixed-signal environments (note that discontinuities at μbump junctions were not modeled). An AC frequency sweep showed worst-case -3dB points of 2.48GHz for the mesh and 681.1MHz for the SoC, which could be further improved by, for instance, adjusting wire dimensions, modifying chiplet placement, or including dedicated analog transmission lines as part of the interposer fabric. We leave more-comprehensive analysis and optimization of such possibilities for future work.

Economic analysis. Since the novelty of this work lies in the methods we propose for constructing SiPs, we compare the economics of SiPterposer against custom, non-reusable interposer designs. In this analysis, we use the 48-chiplet mesh and the realistic-chipset from Section IV-B as examples.

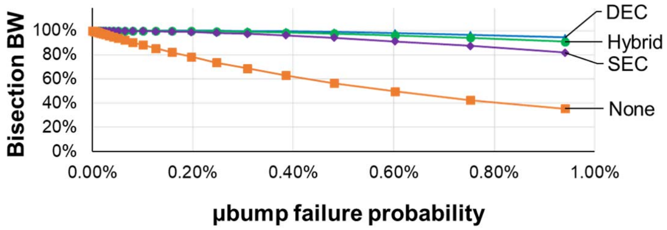


Fig. 9: **Avg. bisection bandwidth vs. defect rate** (empirical pattern). Hybrid coding outperforms SEC-only and nearly matches DEC-only.

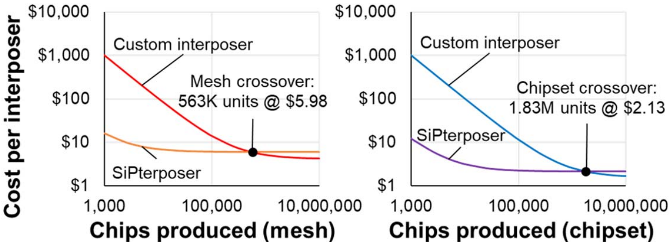


Fig. 10: **Cost of SiPterposer vs. custom-interposer solutions.**

We can characterize the cost of producing an interposer as $c = (f/(d * n)) + v$, where f represents the fixed/NRE costs of design and verification, masks, *etc.*, d is the number of distinct interposer designs built using those fixed costs, n is the quantity produced of each design, and v is the variable cost of making each die. For a traditional custom-interposer SiP, d=1; for SiPterposer, d may be much higher – *i.e.*, SiPterposer's NRE costs are amortized across a far greater total volume.

First, we assume that a custom interposer has the same variable cost per unit area as SiPterposer. As Figs. 5 and 7 suggest, SiPterposer requires more interposer area because of the additional space needed to attach bridge chiplets, plus the area of the bridge chiplets themselves. To create either the mesh or the chipset on SiPterposer, only passive bridge chiplets are needed; thus, we lump the added interposer and bridge chiplet area together to determine SiPterposer's total area overhead vs. a custom interposer (mesh=1.42x, chipset=1.35x). From these values, using a manufacturing cost of ~$1,500 per 300mm-diameter interposer wafer (~2.19-cents per mm^2) [27], we determine v for each scenario: $4.20 vs. $5.96 for the mesh and $1.58 vs. $2.12 for the chipset on a custom interposer vs. SiPterposer, respectively.

Next, we assume that the NRE cost f for each distinct interposer design is $1 million – an intentionally conservative figure. Given that 65nm masks alone cost about $700,000 [2], this assumption creates a worst-case scenario for SiPterposer since, the higher the NRE cost, the stronger the argument for using a single interposer design. Letting d=100 (*i.e.*, SiPterposer is used for 100 different designs for which a custom interposer would otherwise be needed), we can compute the break-even quantities for the mesh and the chipset on SiPterposer vs. a custom interposer (see Fig. 10).

Thus, even when using unfavorable assumptions, SiPterposer's flexibility has economic benefits for designs with volumes up to the hundreds of thousands or low millions of units. The benefits are even greater at lower volumes; for the example chipset with n=10,000, each custom interposer would cost $101.58; SiPterposer, just $3.12. These estimates do not include the effect of yield gains from ECC on chiplet/interposer bonds, which would further reduce fabrication costs.

VI. Conclusions

In this paper, we presented SiPterposer, a mass-producible, flexible, and defect-resistant communication fabric for SiPs. We showed how it can be used to build arbitrary network topologies, evaluated the potential cost savings of our assembly-time-configurable structure, and demonstrated how tolerating defects by applying ECC to chiplet/interposer µbump bonds allows us to realize near-100% chip assembly yields at typical defect rates. An example SiPterposer system achieves these benefits on real-world applications with less than 3% additional power and zero exposed latency overhead compared with traditional SiPs or SoCs.

Acknowledgment. This work was supported by the Center for Applications Driving Architectures (ADA), one of six centers of JUMP, a Semiconductor Research Corporation (SRC) program co-sponsored by DARPA.

References

[1] N. Lu, "A new silicon age 4.0: Generating semiconductor-intelligence paradigm with a virtual moore's law, economics and heterogeneous technologies," in *ISLPED '17*.

[2] M. Khazraee *et al.*, "Moonwalk: Nre optimization in asic clouds," *SIGPLAN Not., Apr 2017*.

[3] C. Tan *et al.*, "Stitch: Fusible heterogeneous accelerators enmeshed with many-core architecture for wearables," in *ISCA '18*.

[4] D. Stow *et al.*, "Cost analysis and cost-driven ip reuse methodology for soc design based on 2.5d/3d integration," in *ICCAD '16*.

[5] M. Deneroff, "Building an soc: How to do it? what will it cost?" in *Workshop on System-on-Chip Design for HPC*, Jul 2015.

[6] C. Palesko and A. Palesko, "Cost breakdown of 2.5d and 3d packaging," *Additional Confs. (Dev. Pkg., HiTEC, HiTEN, & CICMT)*, vol. 2016.

[7] A. Kannan, N. E. Jerger, and G. H. Loh, "Enabling interposer-based disintegration of multi-core processors," in *MICRO '15*.

[8] T. Hisada *et al.*, "Study of warpage and mechanical stress of 2.5d package interposers during chip and interposer mount process," in *Int'l Symposium on Microelectronics*, vol. 2012, 01 2012, pp. 967–974.

[9] Y. Takemoto *et al.*, "Characterization of 4m micro-bump interconnections at 7.6μm pitch for 3d stacked 16m pixel image sensor," *IEEE Trans. on Semicond. Mfg.*, 2017.

[10] L. Jiang, Q. Xu, and B. Eklow, "On effective tsv repair for 3d-stacked ics," in *DATE '12*.

[11] V. Pasca, L. Anghel, and M. Benabdenbi, "Fault tolerant communication in 3d integrated systems," in *2010 Int'l Conf. on Dependable Systems and Networks Workshops (DSN-W)*, June 2010, pp. 131–135.

[12] R. Parikh and V. Bertacco, "udirec: Unified diagnosis and reconfiguration for frugal bypass of noc faults," in *MICRO '13*.

[13] S. Shamshiri, A. Ghofrani, and K. T. Cheng, "End-to-end error correction and online diagnosis for on-chip networks," in *2011 IEEE Int'l Test Conf.*, Sept 2011, pp. 1–10.

[14] J. Lee and J. J. Griffiths, "Polarization effect in laser processing of fine pitch link structures for advanced memory designs," *IEEE Trans. on Semicond. Mfg.*, vol. 22, no. 4, pp. 572–578, Nov 2009.

[15] M. Zhao *et al.*, "Hierarchical analysis of power distribution networks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 2002.

[16] G. Kim *et al.*, "Chip-package co-design of power distribution network for system-in-package applications," in *EPTC 2004*.

[17] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Comput. Surv.*, vol. 38, no. 1, Jun. 2006.

[18] A. Coskun *et al.*, "A cross-layer methodology for design and optimization of networks in 2.5d systems," in *ICCAD '18*.

[19] "Lisnoc." [Online]. Available: http://www.lisnoc.org

[20] N. Jiang *et al.*, "A detailed and flexible cycle-accurate network-on-chip simulator," in *ISPASS '13*.

[21] A. B. Kahng *et al.*, "Orion 2.0: A power-area simulator for interconnection networks," *IEEE Trans. Very Large Scale Integr. Syst.*, 2012.

[22] P. Ehrett *et al.*, "Analysis of microbump overheads for 2.5d disintegrated design," University of Michigan, Ann Arbor, MI, Tech. Rep., 2017.

[23] C. Ward, "Chipworks tears down apple's a5 chip," in *Engadget*.

[24] H. Esmaeilzadeh *et al.*, "Looking back and looking forward: Power, performance, and upheaval," *Commun. ACM*, vol. 55, no. 7, Jul. 2012.

[25] C. Nachiappan *et al.*, "Gemdroid: A framework to evaluate mobile platforms," in *SIGMETRICS '14*.

[26] "The android emulator." [Online]. Available: https://developer.android.com/studio/run/emulator.html

[27] H. Y. Li *et al.*, "The cost study of 300mm through silicon interposer (tsi) with beol interconnect," in *EPTC 2013*.