Designing a 2048-Chiplet, 14336-Core Waferscale Processor

Saptadeep Pal*, Jingyang Liu†, Irina Alam*, Nicholas Cebry†, Haris Suhail*, Shi Bu*, Subramanian S. Iyer*,
Sudhakar Pamarti*, Rakesh Kumar†, and Puneet Gupta*
*Department of Electrical and Computer Engineering, University of California, Los Angeles
†Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign
{saptadeep,puneetg}@ucla.edu

***Abstract*—Waferscale processor systems can provide the large number of cores, and memory bandwidth required by today's highly parallel workloads. One approach to building waferscale systems is to use a chiplet-based architecture where pre-tested chiplets are integrated on a passive silicon-interconnect wafer. This technology allows heterogeneous integration and can provide significant performance and cost benefits. However, designing such a system has several challenges such as power delivery, clock distribution, waferscale-network design, design for testability and fault-tolerance. In this work, we discuss these challenges and the solutions we employed to design a 2048-chiplet, 14,336-core waferscale processor system.**

***Keywords*—Waferscale Processors, Silicon Interconnect Fabric, Chiplet Assembly**

I. Introduction

The proliferation of highly parallel workloads such as graph processing, data analytics, and machine learning is driving the demand for massively parallel high-performance systems with a large number of processing cores, extensive memory capacity, and high memory bandwidth [1, 2]. Often these workloads are run on systems composed of many discrete packaged processors connected using conventional off-package communication links. These off-package links have inferior bandwidth and energy efficiency compared to their on-chip counterparts and have been scaling poorly compared to silicon scaling [3]. As a result, the overhead of inter-package communication has been growing at an alarming pace.

Waferscale integration can alleviate this communication bottleneck by tightly interconnecting a large number of processor cores on a large wafer. Multiple recent works have shown that waferscale processing can provide very large performance and energy efficiency benefits [4, 5] compared to conventional systems. Recently, Cerebras has successfully commercialized a waferscale compute engine. Similarly, in the BrainScaleS/FACETS [6] project, a waferscale brain emulation engine was built. These approaches rely on building one large monolithic waferscale chip with custom cross-reticle interconnections. Monolithic waferscale chips are, however homogeneous, and so cannot integrate components from heterogeneous technologies such as DRAM or other dense memory technologies. Moreover, in order to obtain good yields, redundant cores and network links need to be reserved on the waferscale chip.

A competing approach to building waferscale systems is to integrate pre-tested known-good chiplets (in this work, we call un-packaged bare-dies/dielets as chiplets) on a waferscale interconnect substrate [5]. Silicon interconnect Fabric (Si-IF) is a candidate technology which allows us to tightly integrate many chiplets on a high-density interconnect wafer [7]. Si-IF technology provides fine-pitch copper pillar based ($10\mu m$ pitch) I/Os which are atleast 16x denser than conventional μ-bumps used in an interposer based system [8], as well as $\sim 100\mu m$ inter-chiplet spacing. Therefore, it provides global on-chip wiring-like characteristics for inter-chiplet interconnects. Moreover, in a chiplet-based waferscale system, the chiplets can be manufactured in heterogeneous technologies and can potentially provide better cost-performance trade-offs. E.g., TBs of memory capacity at 100s of TBps alongside PFLOPs of compute throughput can be obtained which is suitable for big-data workloads in HPC and ML/AI.

978-1-6654-3274-0/21/$31.00 ©2021 IEEE

Large scale chiplet assembly based system design, however, has its unique set of challenges which encompass a wide range of topics from the underlying integration technology to circuit design and hardware architecture, and their impact on software. This work, for the first time, attempts to build a fine-grained chiplet-based waferscale processor prototype. The system comprises an array of 1024 tiles, where each tile is composed of two chiplets, for a total of 2048 chiplets and about 15,000 mm^2 of total area.

The scale of this prototype system forced us to rethink several aspects of the design flow. Because this is the first attempt at building such a system, there were several unknowns around the manufacturing and assembly process. As a result, fault tolerance and resiliency, was one of the primary drivers behind the design decisions we took. We also ensured that the design decisions were not too complex, such that they could be reliably implemented by a small team of 3-4 graduate students within a reasonable amount of time. The several challenges we faced while architecting and designing this system are as follows:

(1) How should we deliver power to all the flip-chip bonded chiplets across the wafer?

(2) How can we reliably distribute clock across such a large area?

(3) How can we design area-efficient I/Os when a large number of fine-pitch copper pillar-based I/Os need to be supported per chiplet, and how do we achieve very high overall chiplet assembly and bonding yield?

(4) What is the inter-chip network architecture and how do we achieve resiliency if a few chiplets fail?

(5) What is the testing strategy when I/O pads have small dimensions and how do we ensure scalability of the testing schemes?

(6) How can we design the chiplets and the substrate with the uncertainty and constraints of the manufacturing process ?

In this paper, we explain the challenges and possible solutions (including design decisions for our prototype system) for building scale-out chiplet-assembly based systems. To the best of our knowledge, **this is the largest chiplet assembly based system ever attempted**. In terms of active area, our prototype system is about 10x larger than a single chiplet-based system from NVIDIA/AMD etc. [9, 10], and about 100x larger than the 64-chiplet Simba (research) system from NVIDIA [11].

II. Overview of the Waferscale Processor System

To understand the design and implementation challenges as well as the opportunities when using a chiplet based waferscale processor system, we architected and designed a 2048-chiplet based 14,336 core processor system.[1]Here, we first provide a brief overview of the architecture of the overall system, the chiplets and the intra-chiplet and inter-chiplet network.

a) Overall System Architecture: We designed a scalable tile-based architecture for our system. This architecture can scale to a 32×32 tile array (see Figure 1), for a total of 1024 tiles. Each tile is comprised of two chiplets: a $Compute$ chiplet and a $Memory$ chiplet. It contains a total of 14 independently programmable processor cores and 512KB of globally shared memory. We architected this system as a unified memory system where any core on any tile can directly access the globally shared memory across the entire waferscale system using the waferscale interconnect network. Scaling a network across 14,336 cores

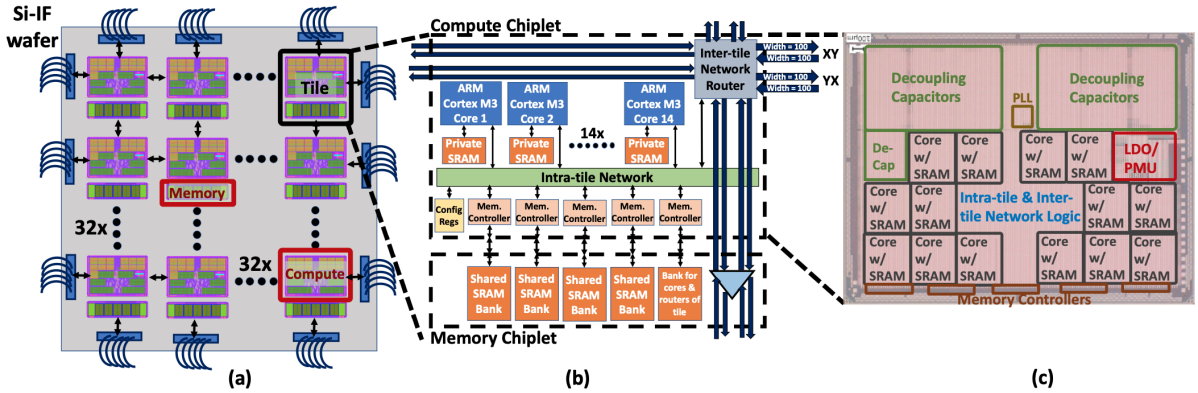

Fig. 1: (a) Waferscale Processor System Overview showing 32x32 tile array where each tile comprises of a compute chiplet and a memory chiplet. (b) Detailed overview of the compute and memory chiplets. (c) Micrograph of the compute chiplet.

TABLE I: Salient Features of the Waferscale Processor System

# Compute Chiplets	1024	**# Memory Chiplets**	1024	**# Cores per Tile**	14
Compute Chiplet Size	3.15mm x 2.4mm	**Memory Chiplet Size**	3.15mm x 1.1mm	**Network B/W**	9.83 TBps
Private Memory per Core	64KB	**Total Shared Memory**	512 MB	**Total # Cores**	14336
Compute Throughput	4.3 TOPS	**Shared Memory B/W**	6.144 TB/s	**# I/Os per Chiplet**	2020(C)/1250(M)
Total Area (w/ edge I/Os)	15100 mm^2	**Nominal Freq./Voltage**	300 MHz/1.1V	**Total Peak Power**	725W

however is challenging. Therefore, we designed a hierarchical network scheme with an intra-tile crossbar network and a waferscale inter-tile mesh network. The salient features of the system are listed in Table I.

The chiplets are designed and fabricated in the TSMC 40nm-LP process and terminated at the top copper metal layer where the fine-pitch I/O pads were built. The waferscale substrate is a passive substrate containing the interconnect wiring between the chiplets and copper pillars to connect to the chiplet I/Os. The chiplets are flip-chip bonded on to the waferscale substrate as shown in Figure 2 and we would connect the entire waferscale system to the power supply and external controllers using edge connectors.

b) Compute Chiplet: As shown in Figure 1, the compute chiplet contains 14 ARM CORTEX-M3 cores and their private SRAMs (64KB each), memory controllers (to access the banks in the memory chiplet), network routing infrastructure for inter-tile network and a chiplet-level intra-tile crossbar interconnect (implemented using ARM BusMatrix IP) to connect all these components. The power delivery related components are also contained within the compute chiplet. The network routing infrastructure was built around the open-source BSG IPs [12], but includes other custom units needed to support two independent networks, adapters to communicate with the intra-tile network and support various memory-mapped functionality. The micro-architectural details are out of scope for this paper.

c) Memory Chiplet: The memory chiplet comprises five 128KB SRAM memory banks. Four of these banks are addressable using the global shared memory address space while one bank can be accessed only by the cores and network routers on the same tile. All these banks can be accessed in parallel and are connected to the intra-tile network through the memory controllers on the compute chiplet. The memory chiplet also provides buffered feedthroughs for the north-south interconnect links and two banks of decoupling capacitors. Note that though the two chiplets in a tile are architecturally heterogeneous, we implemented them in the same technology node for ease of design. However, this chiplet can be easily implemented in a newer or denser memory technologies for higher memory capacity and/or area savings.

d) Features of the waferscale substrate: The waferscale substrate is built using the Si-IF technology. The I/O (i.e., copper pillar) pitch we use in our prototype is 10μm (minimum that the technology offers). The interconnect wiring pitch is 5μm (minimum offered is currently 4μm). With two layers of signaling, the edge interconnect density we achieve is 400 wires/mm.

We validated the system design and architecture discussed in this paper by emulating a reduced-size multi-tile system on an FPGA platform (full waferscale system emulation was not possible due to scale). We were successfully able to run various workloads including graph applications such as breadth-first search (BFS), single-source shortest path (SSSP), etc. on this system. Next, we will discuss each of our design decisions in detail.

III. Waferscale Power Delivery and Regulation

Here, we ask the question: How to deliver power reliably to all the chiplets which are flip-chip bonded on to a ~15000mm^2 large waferscale interconnect substrate? Unlike a monolithic waferscale system where the power can be directly supplied to the top-most metal layer (face side), the chiplets are flip-chip bonded on to the thick waferscale substrate. As a result, either power can be delivered through the backside of the wafer using through-wafer-vias (TWVs) [13], which are 700μm deep vias across a full-thickness wafer, or can be delivered at the edge of the wafer. Since the integration of TWV technology in a Si-IF wafer is still under development and not ready for prime-time yet, we chose to use edge power delivery for our prototype system.

The peak power per tile is about 350mW when operating at a voltage of 1.21V (fast-fast corner). Therefore, about 290A of current needs to be delivered to the chiplets across the wafer. The number of metal layers in the substrate is restricted to four in order to maximize yield. Since two metal layers are dedicated to inter-chip signaling, two layers are available for power distribution. Maximum thickness of metal layers in the Si-IF technology is 2μm and thus, the resistance of the power distribution network would result in large voltage droop if the current that needs to be delivered is very large. As such, we considered two different strategies for power delivery: (1) High voltage (say 12V) power delivery at the edge and using down conversion (buck or switched capacitor based converters) near the chiplets [5], which would lower the current delivered through the power planes by ~12x, (2) Higher voltage power delivery (say 2.5V) at the edge and using low-dropout (LDO) based regulation in the chiplets, which would mean that a larger amount of current needs to traverse the PDN planes and, therefore, would sustain larger losses in the power delivery planes.

The first option of using down-conversion near the chiplets has high area overheads because bulky off-chip components such as inductors and capacitors need to be placed on the wafer. We estimate that about 25-30% of the area would be occupied by these components. Moreover,

[1]As of submission of this paper, the fabricated chiplets are back, and the waferscale assembly design and fabrication is in progress.

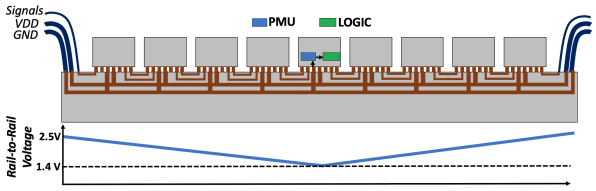

Fig. 2: Power is delivered from the edge. The chiplets at the edge of the wafer receive power at 2.5V. There is voltage droop as we move towards the center of the wafer and the chiplets at the center receive power at 1.4V.

integrating these components on the wafer would result in disruption of the regular structure of the chiplet array and increase the inter-chiplet distance, which would diminish the benefits of fine-pitch interconnects. This scheme would result in increased design complexity.

Since, this prototype is a sub-kW system, we chose to avoid this complexity in lieu of some power efficiency loss coming from the resistive power loss and poorer LDO efficiency. In our scheme, the chiplets near the edge would receive power at much higher voltage (2.5V) and the chiplets away from the edge would receive power at lower voltage due to resistive power loss related voltage droop. Our estimates show that the chiplets at the center would receive power at roughly 1.4V during peak power draw from all the chiplets. This however, makes the LDO design challenging as it has to produce a stable voltage of 1.1V (nominal) for the logic devices while the DC supply voltage can vary between 1.4V and 2.5V depending on where the chiplet is placed on the wafer. We built a custom LDO which can track this wide input voltage range.

The other challenge is that the LDO regulator has to support up to 350mW of peak power while sustaining up to 200mA current demand fluctuation (worst case) within a few cycles. In order to achieve good regulation under these operating conditions, the LDO regulator needs sufficient decoupling capacitance at the output. Such high capacitance requirements are usually fulfilled using off-chip discrete decoupling capacitors. However, in our waferscale system, off-chip capacitors can only be placed around the edge of the array. As a result, the chiplets at the center of the array can be as far as 70 mm away from the nearest capacitor. Hence, we designed a custom on-chip decoupling capacitor and dedicated $\sim$35% of the total tile area to decoupling capacitance giving about 20 nF per tile.[2] The eventual design ensures that the regulated voltage is always between 1.0V and 1.2V across process/voltage/temperature corners. We omit the circuit level details for brevity.

IV. Waferscale Clock Generation and Distribution

Next, we ask the question: How do we provide clock to all the chiplets across the $>$15,000mm^2 waferscale substrate?

We included a phase-locked loop (PLL) in the compute chiplet which can take an input clock with frequency between 10 and 133MHz and generate an output clock with frequency up to 400 MHz. Therefore, one option is to distribute a slow clock across the wafer using a passive clock distribution network (CDN) built on the Si-IF. However, there are two challenges in such a scheme. First, the parasitics of a passive CDN which spans an area of about 15,100 mm^2 and has 1024 sinks are very large ($>$450pF and $>$120nH). So, the clock distribution can only be done at sub-MHz frequency. Also, getting a good crystal oscillator which can drive large capacitive load while ensuring absolute jitter performance of sub-100 pico-seconds is hard. Second, the PLL IP we used requires a stable reference voltage for reliable operation. However, the voltage regulation in the chiplets away from the edge is not perfect and the regulated voltage could fluctuate between 1.0V and 1.2V. As a result, stable clock can only be generated near the edge of the wafer where the chiplets can access near-by off-chip decoupling capacitors. Therefore, in this system, a fast clock (up to 350 MHz) will be generated in one of the edge tiles and then forwarded throughout the tile array using forwarding circuitry built inside every tile. Next, we briefly describe the clock selection and forwarding circuitry.

Clock Selection and Forwarding

The clock selection and forwarding circuitry is a part of the compute chiplet. As shown in Figure 3, the compute chiplet has multiple clock inputs: master (slow) clock, software-controlled test/JTAG clock and four forwarded clocks (one from the neighboring chiplet on each side); and four outputs to forward a clock to the neighboring chiplets on all sides. During testing and program/data loading phases, the JTAG clock is selected as the functional clock for the tile. During the program execution phase however, either one of the four forwarded input clocks or the master clock can be selected as the functional clock for the tile logic. If the frequency of the selected tile clock needs to be multiplied, it can be optionally passed on to the PLL. Moreover, one of these five clocks is selected to be forwarded to all the neighboring tiles.

During boot-up, the clock selector circuitry defaults to the software-controlled JTAG clock. Using JTAG, we then initiate the *clock setup* phase. In this phase, first we select one or multiple edge tiles and configure them to generate a faster clock from the slower system clock that is provided from an off-the-wafer crystal oscillator source. The generated faster clocks from the edge chiplets are forwarded to their neighboring chiplets. The non-edge chiplets are then configured for the auto-clock selection phase. In this phase, the clock selection circuitry selects the forwarded clock which starts toggling and is the first to reach a pre-defined toggle count (default is 16). Once a forwarded clock is selected, the clock setup phase for that tile terminates and the selected clock is forwarded to its neighboring tiles. This ensures that no live-lock scenarios occur in the clock forwarding process.

However, one issue with such a clock forwarding scheme is that the fast clock can accrue duty cycle distortion because of pull-up/pull-down imbalance in the buffers, inverters, forwarding unit components and inter-chiplet I/O drivers [15]. As the clock traverses across multiple tiles in the array, this duty cycle distortion can potentially kill the clock, e.g., a 5% distortion per tile could kill the clock with in just 10 tiles. In order to avoid this issue, we forward an inverted version of the clock. This ensures that the distortion is alternated between the clock cycle halves. [3] Moreover, we also implemented a duty cycle distortion correction (DCC) unit [16], which can correct any residual distortion.

Resiliency in Clock Forwarding Network

Faulty chiplets can potentially disrupt the clock forwarding mechanism. Our clock generation and forwarding scheme however, has resilience built in. Because any chiplet at the edge can generate a faster clock, there isn't a single point of failure in clock generation. Moreover, because every non-edge tile receives a toggling clock from all four directions, this ensures that if at least one of the neighboring chiplets out of the four is not faulty, then the clock can reach that chiplet and be further forwarded. By induction, it can be proven that the generated fast clock can reach all non-faulty tiles on the wafer, unless all the neighboring tiles of a specific tile are faulty.

Figure 4 shows one possible clock forwarding configuration for an 8x8 tile array with faulty tiles. The edge tile ❶ generates the faster clock that gets forwarded across the entire wafer. Even with 6 faulty tiles in a 64 tile mesh, all tiles, except tile ❷, receive the forwarded clock. Tile ❷ has faulty tiles on all four sides and hence, is unable to receive the generated clock. Even otherwise, this tile would have been rendered unusable since there is no available path for other tiles to communicate with this tile using the waferscale inter-tile network.

[2]In the future, incorporation of deep trench decoupling capacitors [14] (currently under development) in to the waferscale substrate has the potential to significantly improve PDN performance and will also reduce the area overhead of on-chip decoupling capacitors.

[3]The half-cycle phase delay and any jitter introduced is not a concern since our inter-chiplet communication uses asynchronous FIFOs [12]

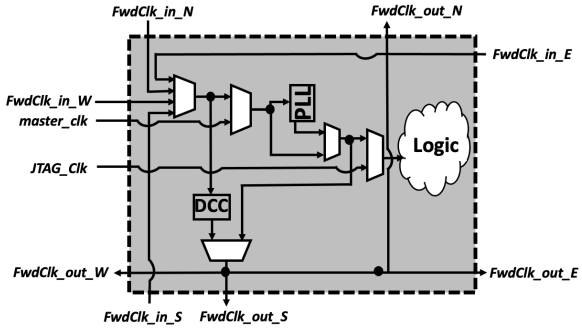

Fig. 3: Schematic of clock selection and forwarding circuitry

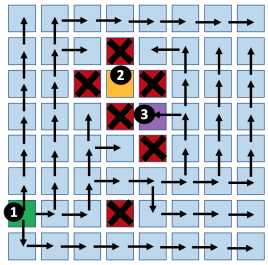

Fig. 4: A clock forwarding configuration is shown for a system with faulty tiles. All tiles except the yellow one can receive the forwarded clock

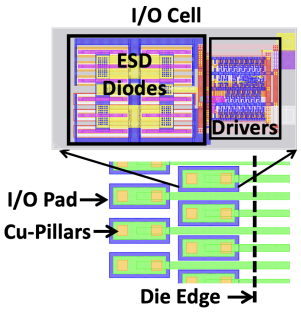

Fig. 5: Fine-pitch I/O layout with ESD protection circuitry and two Cu-pillars per I/O pad

On the other hand, tile 3 can still receive the forwarded clock even when surrounded by three faulty tiles. This is because it has one non-faulty neighbor from which it receives the generated clock.

V. I/O Architecture

In this section, we ask the question: Since, each chiplet needs to support a large number of I/Os (transceiver circuitry and Cu pads) for fine-pitch copper pillar interconnects, how do we design area-efficient I/Os and achieve high bonding yield?

The Si-IF technology allows inter-chiplet links to be as short as 200-300 μm. As a result, the links can be easily driven by small, energy efficient I/O circuitry that can operate at 1GHz. Besides, the Si-IF technology also offers fine pitch copper pillars ($10\mu m$ pitch) for bonding the chiplet on to the substrate and fine pitch interconnect wiring (4 μm pitch) for inter-chiplet communication [7]. In order to support a large number of I/Os without large area overhead, the size of the I/O cells need to be small. Moreover, if the I/O cells are large, they have to be placed at a distance from the I/O pads, thereby, significantly reducing the energy benefits of short inter-chiplet Si-IF links. Therefore, if the I/O circuitry can be completely encompassed under the pad, it would enable us to obtain optimal energy efficiency for the I/O transceivers. The transmitter was designed using simple appropriately-sized cascaded inverters which can drive signals at 1GHz for link length of up to $500\mu m$. The receiver was designed using two minimum sized inverters. Managing electrostatic discharge (ESD) in small I/O cell area is a challenge but fortunately, unlike packaged parts which usually have to deal with large ESD events corresponding to 2kV human body model, bare-die chiplet to wafer bonding only needs to address the less stringent 100V human-body model (HBM) or machine-model (MM) specifications [17] (similar to silicon interposers).

The final area of the I/O cells, along with the stripped down ESD circuitry, was about $150\mu m^2$. This is larger than the area that could be accommodated under one copper pillar. Therefore, we designed the I/Os such that two copper pillars can land on each pad. This also enhances the bonding-related yield. For a single pillar, the expected bonding yield is $>$99.99% [7]. With two pillars per pad, per-I/O bonding yield can be improved significantly. With over 2000 I/Os per chiplet, bonding yield for a chiplet would therefore improve from 81.46% to 99.998%. This is critical for system yield since our wafer-scale system comprises of 2048 chiplets (i.e., at the wafer-level this would reduce expected number of faulty chiplets from 380 down to 1). As shown in Figure 5, in order to achieve the maximum I/O density per mm of chiplet edge, the I/O pads were placed such that the two pillars landing on each I/O pad would be orthogonal to the chiplet edge. Overall, this I/O design is area-efficient (total I/O area is only $0.4mm^2$), energy-efficient (0.063pJ/bit) and improves system yield dramatically.

VI. Waferscale Network Architecture and Resiliency

The next question we ask is: How do we architect the waferscale network and ensure good connectivity among working chiplets when a small number of chiplets may fail?

We use a mesh network to connect the chiplets across the wafer. The network routers reside on the compute chiplet. In order to avoid deadlocks, we use dimension-ordered routing (DoR). However, as mentioned in Section V, even with an excellent bonding yield of 99.998% per chiplet, the overall system of 2048 chiplets might have one or few faulty chiplets. Using Monte-Carlo simulation we estimate the percentage of source destination pairs that will get disconnected if there is a single path between any pair of chiplets. As shown in Figure 6, with just five faulty chiplets (out of 2048) in the wafer, $>$12% of paths get disconnected.

In order to overcome this issue, we designed two independent networks across the wafer; one with X-Y dimension-ordered routing and the other with Y-X dimension ordered routing as shown in Figure 7. With this, most chiplet pairs (all pairs where the two chiplets are not in the same row/column) on the wafer have two distinct paths between them. This dramatically reduces the number of paths that get disconnected when a certain number of chiplets on the wafer are faulty. For example, with five faulty chiplets on the wafer, the percentage of disconnected paths reduces from $>$12% to $<$2% as we go from a single DoR network to two independent DoR networks on the wafer. The paths that still get disconnected with two DoR networks mostly connect those pairs of chiplets that are in the same row/column. Moreover, in our system, network request-response communication happens using the complimentary networks (this is baked in to the router hardware). As shown in Figure 7, if a request from chiplet A to chiplet B is sent along the X-Y direction, the response from B-to-A is sent in the Y-X direction in order to ensure that the same path is taken by the request-response pair. This makes sure that two-way communication between chiplets is possible whenever one non-faulty path exists. This also avoids deadlocks between request-response pairs.

Given the length of our chiplet edge, we can support 400-bit wide parallel inter-chiplet network link escaping each side of the tile. The width of an entire packet in our case is 100 bits. Thus, we divide the inter-chiplet links into four separate parallel wide buses. Two of them are dedicated to the X-Y network and the other two are dedicated to the Y-X network. The two buses corresponding to each DoR network are ingress and egress links.

The task of choosing the correct network to use is left up to the kernel software. Once a system is fully assembled, we identify the faulty tiles and store them in a fault-map. The kernel software then uses the information of the fault map to decide the network to use for a source-destination tile pair. If both the paths are available between two tiles, the kernel software is used to distribute the source-destination pairs to the network in a way such that both the networks are equally utilized. While doing so, we ensure packet consistency (i.e., packets arrive in order) by allocating all communication between a source-destination pair to a single network only.

Additionally, we can also use kernel software to circumvent the issue

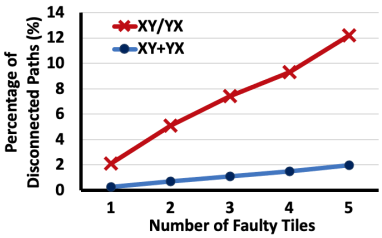

Fig. 6: Average percentage of disconnected source-destination pairs for the conventional scheme with one DoR network versus when two networks are used. This result is obtained using a set of randomly generated fault maps

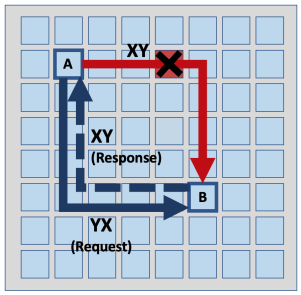

Fig. 7: Fault-tolerant waferscale mesh network architecture

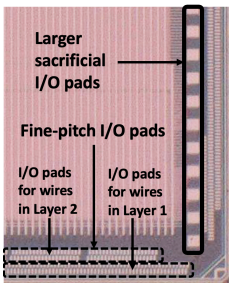

Fig. 8: Larger pads are for probing and fine-pitch pads are for inter-chiplet communication.

of disconnected paths. Every time a packet needs to be sent through a path with a faulty tile, it can divert the packets to an intermediate tile and then route it from the intermediate tile to the final destination. The response packet will also follow the same path. However, this will require cores to allocate cycles towards network routing instead of executing the actual application process and hence, can adversely affect the overall performance. Since our solution of using two DoR networks significantly reduces the number of disconnected paths, as compared to a single network, this performance impact is expected to be minimal.[4]

VII. Testing Infrastructure

Chiplet-based waferscale technology promises to provide better system yield than a monolithic approach. However, it depends on the identification of known-good dies (KGD) and reliable chiplet assembly on the Si-IF. Therefore, we ask the question: How to do design our test scheme for pre-bond testing as well as testing the system post assembly?

a) Test infrastructure inside a tile: As shown in Figure 9, the ARM CORTEX-M3 core provides debug access through a Debug Access Port (DAP, based on IEEE 1149.1 JTAG protocol minus boundary scan). External communication with the DAP port is done using a JTAG interface. Each tile in our system has fourteen cores and, therefore, fourteen DAP interfaces. One option was to bring all the fourteen DAP interfaces out to the edge of the compute chiplet; and eventually to the edge of the wafer for testing (using ARM-based MBED microcontrollers, in our case). However, such a scheme would require a lot of I/Os at the edge of the wafer, e.g., for just the 32 chiplets at the edge, a 1792-bit interface is needed. Though handling this many I/Os at the edge is possible using advanced connectors or by using serialization/de-serialization at the edge of the wafer, this was beyond the scope of this work. Therefore, we daisy-chained all the DAP interfaces inside the compute chiplet (as shown in Figure 9) and so, only one JTAG interface is required to connect to the multiple DAPs in a chain. We also provision the daisy-chain such that it can be extended to include DAPs across multiple chiplets.

During testing, all the cores would usually run the same set of test instructions. Also, upon analysis of many irregular workloads, we found that majority of the cores would actually run the same program (albeit independently). Therefore, in order to minimize the program loading time, we provision for the same program to be broadcasted to multiple cores in a tile. The optimization is to broadcast the input at TDI_{tile} (Test Data In) to the TDI pin of all the DAP ports and the TDO (Test Data Out) of the first core is forwarded to TDO_{tile}. Thus, in this mode, the external controller sees only one DAP per tile and, therefore, the JTAG bit shifting latency reduces by 14x.

A. Pre-bond Testing

Pre-bond testing is essential for identifying KGD parts. However, there are two issues which make pre-bond testing of un-packaged chiplets designed with fine-pitch IO pads for Si-IF assembly difficult: (1) the fine-pitch IO pads with 10 μm pitch and 7 μm width are not amenable to probe-card-based testing. The probe pitch usually is larger than 50 μm. (2) Once the probes land on a pad, it damages the planarity of the pad surface which is critical for reliable subsequent direct metal-to-metal bonding.

Therefore, in order to get around these issues, we designed larger duplicate pads for the JTAG and some auxiliary test signals. These larger pads are designed such that probe-card testing can be done, while their fine-pitch counterparts are used for bonding to the Si-IF. Using this approach, we can thoroughly test the chiplets and eliminate faulty chiplets before bonding. Once the chiplets are tested, we ensure that, for die-to-wafer bonding, we don't use the larger pads which are probed. We only have copper pillars for bonding the fine-pitch pads which are not probed. The same JTAG interface can be used to load test routines and programs in the chiplet after bonding, but now using the fine-pitch pillars.

B. Post Assembly Testing

After pre-bond testing, the non-faulty chiplets are passed on to the die-to-wafer bonding process. Past work on fine-pitch die-to-wafer bonding [7] has shown to achieve excellent bonding yield (>99.99%). In our waferscale system, the total number of inter-chip I/Os is 3.7M+. Even with the I/O redundancy scheme described in section V, a few bonding-related failures may still occur. Therefore, it is important to pin-point the location of the faulty chiplets. Moreover, since the number of chiplets to test in a waferscale system is very large, this testing process needs to be done at high throughput and demands scalability.

a) Progressive multi-chiplet JTAG chain unrolling: As shown in Figures 9 and 10, we designed the JTAG chaining mechanism in a way where the TDO_{tile} signal can either be forwarded to the next tile in the chain or can loop-back towards TDO_{loop}.[5] Therefore, each chiplet in the chain can be tested progressively and independently. On power-up, the default mode is the chain loop-back mode. The first chiplet in the chain is tested first. Once the chiplet passes the test, its test mode is changed so that the TDO_{tile} from the first chiplet is forwarded to the second chiplet. The second chiplet is still in the loop-back mode and, therefore, the TDO_{tile} signal from the second chiplet is eventually brought out through the TDI_{bypass} and TDO_{loop} signals of the first chiplet to the external controller. The chain is progressively unrolled and the test procedure is repeated for all the chiplets in a chain. This helps to identify the faulty chiplet as the chain unrolls. This mechanism can also be used for during-assembly testing to intermittently check for failures in a partially bonded system. This scheme would help to identify and discard partially populated faulty systems and minimize wastage of KGD chiplets.

b) Multi-chain JTAG: To achieve high-throughput and scalability, we adopted a multi-chain debug methodology. Instead of creating one JTAG daisy-chain with 1024 tiles, we chose to split the array in to 32 chains with each running across the rows. This has two

[4]In the future, we will incorporate sophisticated routing schemes [18, 19] for improved waferscale fault tolerance as well as performance.

[5]Similar to the under-development IEEE P1838 standard [20] for 3D devices

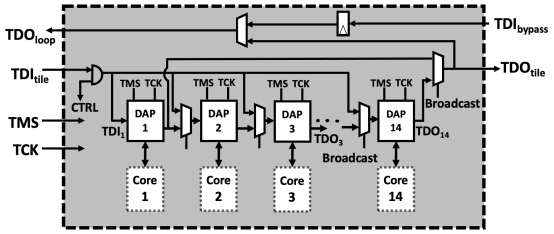

Fig. 9: Schematic of the test circuitry inside a tile

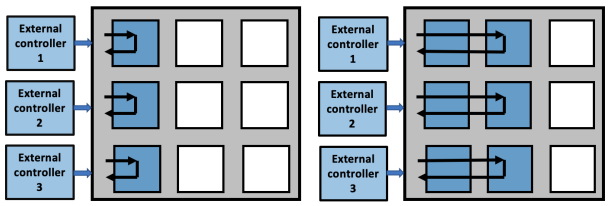

Fig. 10: Progressive unrolling scheme is shown. The JTAG chain is unrolled progressively to identify the faulty chiplet in the chain.

primary benefits: (1) Testing and program/data loading to the tiles in the rows can be done in parallel. As a result, this can speedup these processes by up to 32x, speeding up loading all the memory on the system from 2.5 hours (with a single chain) to roughly under 5 minutes. (2) The TMS and TCLK signals are broadcast to the tiles in a chain. Splitting the chains allows us to have independent TMS and TCLK signals for each row and helps reduce the load on these signals; this would enable us to run these signals at up to 10 MHz.

VIII. Waferscale Substrate and its Impact on Design

For such a large integration substrate, we were unsure of the wafer substrate yield (a problem exacerbated by fabrication in a research facility). As a result, we designed the chiplet I/Os in such a way that, even with one routing layer, we would have a working processor system, albeit with reduced shared memory capacity. We have two sets of I/O columns on each side of chiplet (as shown in Figure 8), one set per layer of signal routing. The first set comprises of the two I/O columns closest to the die edge and consists of all the absolutely essential network link I/Os. It also comprises of I/Os corresponding to two out of the five memory banks in our memory chiplet. The other set of I/O columns that would require the second routing layer, consists of the non-essential I/Os and the ones corresponding to the remaining three memory banks. Thus, if we have just a single layer of routing, we will be able to build the entire processor system by connecting just the I/Os in the first set. The only downside would be the reduction of shared memory capacity by 60%.

Since the size of the wafer is much larger than the maximum size of a reticle, the Si-IF substrate had to be designed such that it is step-and-repeatable.[6] The entire wafer is divided into smaller identical reticles and is fabricated by stitching these reticles. Each reticle consists of 72 tiles (12x6). The inter-chiplet links, within each reticle have width of 2 μm and spacing of 3 μm. However, at the edge of each reticle, the links escaping are made fatter (width increases to 3 μm and spacing reduces to 2 μm), while keeping the pitch constant, in order to reduce the impact of reticle stitching error [21]. Besides, a number of I/Os from each of the tiles at the edge of the mesh needs to fan-out to the edge of the wafer and connect to the external connectors. We designed the fan-out wiring and the edge I/O pads into each reticle. The chiplet slots on the Si-IF substrate from the edge reticles would remain un-populated and the external connectors would connect to the pads in these reticles. To ensure that these I/O pads don't cause an issue where chiplets are bonded, we use a custom block etch process to remove the pads wherever they are not needed. If a foundry supports multiple reticles per wafer, the edge of the wafer can also be printed using a separate mask.

The Si-IF substrate for this processor system consists of four metal layers. The bottom two layers are built as dense slotted planes and are dedicated to power supply. The top two layers are sparse and are dedicated to inter-chiplet signal routing. The major challenge in using today's conventional tools for designing the interconnect substrate is the sheer scale of the system. The memory footprint when designing a four layer >15000 mm^2 wafer using current commercial tools explodes, which, in turn, leads to very large design time. Hence, we developed our own lightweight custom router for designing the four layer waferscale substrate. The current version of the router supports only jog-free routing for inter-chiplet connections, which is sufficient for this prototype. Developing a general, scalable router for large (but low wiring density) chiplet assemblies is part of our future work.

[6]This won't be a problem if a direct-write lithography system is employed but most commercial and research foundry patterning processes employ steppers.

IX. Summary and Conclusion

Chiplet assembly is a very promising approach to build scale-out heterogeneous computing systems. We describe our experience designing largest ever such chiplet assembly based waferscale processor (at least 10X larger than largest known commercial chiplet-based systems) and develop a design methodology for the same. We highlight challenges and potential solutions in power delivery, clock distribution, network design, design for testability and fault-tolerance for waferscale systems. To best of our knowledge, this is the first work discussing design methodology challenges for large chiplet-assembly based systems. Our ongoing work aims at characterizing the waferscale prototype and developing design methods for higher-power waferscale systems.

Acknowledgement

The authors would like to thank Matthew Tomei, Ananya Ravikumar, Alexander Graening, Sorin Dobre (Qualcomm), UCLA-CHIPS, CDEN, Qualcomm Innovation Fellowship and TSMC for their support.

References

[1] *Workload Analysis of Blue Waters*. https://arxiv.org/ftp/arxiv/papers/1703/1703.00924.pdf. (accessed Nov 23, 2020).
[2] K. Shirahata et al. "A Scalable Implementation of a MapReduce-based Graph Processing Algorithm for Large-Scale Heterogeneous Supercomputers". *13th International Symposium on Cluster, Cloud, and Grid Computing*. 2013.
[3] S. Pal et al. "A Case for Packageless Processors". *IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 2018.
[4] Kamil Rocki et al. *Fast Stencil-Code Computation on a Wafer-Scale Processor*. 2020. arXiv: 2010.03660.
[5] S. Pal et al. "Architecting Waferscale Processors - A GPU Case Study". *IEEE International Symposium on High Performance Computer Architecture*. 2019.
[6] J. Schemmel et al. "A wafer-scale neuromorphic hardware system for large-scale neural modeling". *International Symposium on Circuits and Systems*. 2010.
[7] A. A. Bajwa et al. "Demonstration of a Heterogeneously Integrated System-on-Wafer (SoW) Assembly". *68th ECTC*. 2018.
[8] Suresh Ramalingam. *3D-ICs: Advances in the industry*. Accessed Nov 23, 2020.
[9] NVIDIA. *A100 TENSOR CORE GPU*. Accessed Nov 22, 2020.
[10] AMD. *Ryzen™ Threadripper™ Processors*. Accessed Nov 23, 2020.
[11] Yakun Sophia Shao et al. "Simba: Scaling Deep-Learning Inference with Multi-Chip-Module-Based Architecture". *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. USA, 2019.
[12] Shaolin Xie and Michael Bedford Taylor. *The BaseJump Manycore Accelerator Network*. 2018. arXiv: 1808.00650 [cs.AR].
[13] M. Liu et al. "Process Development of Power Delivery Through Wafer Vias for Silicon Interconnect Fabric". *69th ECTC*. 2019.
[14] K. T. Kannan and S. S. Iyer. "Deep Trench Capacitors in Silicon Interconnect Fabric". *IEEE 70th Electronic Components and Technology Conference*. 2020.
[15] Inc. Kaijian Shi Synopsys. *Clock Distribution and Balancing Methodology For Large and Complex ASIC Designs*. Accessed Nov 23, 2020.
[16] Yi-Ming Wang and Jinn-Shyan Wang. "An all-digital 50% duty-cycle corrector". *IEEE International Symposium on Circuits and Systems*. 2004.
[17] Global Semiconductor Alliance. "Electrostatic Discharge (ESD) in 3D-IC Packages" (Accessed Nov 21, 2020).
[18] Jie Wu. "A fault-tolerant and deadlock-free routing protocol in 2D meshes based on odd-even turn model". *IEEE Transactions on Computers* (2003).
[19] Alessandro Zorat. "Construction of a fault-tolerant grid of processors for wafer-scale integration". *Circuits, Systems and Signal Processing* (1987).
[20] E. J. Marinissen, T. McLaurin, and Hailong Jiao. "IEEE Std P1838: DfT standard-under-development for 2.5D-, 3D-, and 5.5D-SICs". *21st IEEE ETS*. 2016.
[21] W. C. Chen et al. "Wafer level integration of an advanced logic-memory system through 2nd generation CoWoS technology". *VLSI Technology Symposium*. 2017.