DeFT: A Deadlock-Free and Fault-Tolerant Routing Algorithm for 2.5D Chiplet Networks

Ebadollah Taheri, Sudeep Pasricha, and Mahdi Nikdast
Department of Electrical and Computer Engineering, Colorado State University, USA

***Abstract*—By interconnecting smaller chiplets through an interposer, 2.5D integration offers a cost-effective and high-yield solution to implement large-scale modular systems. Nevertheless, the underlying network is prone to deadlock, despite deadlock-free chiplets, and to different faults on the vertical links used for connecting the chiplets to the interposer. Unfortunately, existing fault-tolerant routing techniques proposed for 2D and 3D on-chip networks cannot be applied to chiplet networks. To address these problems, this paper presents the first deadlock-free and fault-tolerant routing algorithm, called *DeFT*, for 2.5D integrated chiplet systems. *DeFT* improves the redundancy in vertical-link selection to tolerate faults in vertical links while considering network congestion. Moreover, *DeFT* can tolerate different vertical-link-fault scenarios while accounting for vertical-link utilization. Compared to the state-of-the-art routing algorithms in 2.5D chiplet systems, our simulation results show that *DeFT* improves network reachability by up to 75% with a fault rate of up to 25% and reduces the network latency by up to 40% for multi-application execution scenarios with less than 2% area overhead.**

I. Introduction

The continuous demand for higher computation power necessitates further scalability improvements in systems-on-chip (SoCs). Unfortunately, conventional 2D SoCs suffer from low scalability because of their low yield, and hence a high manufacturing cost when scaling up to support higher complexities [1]. To this end, 3D integration partitions an SoC into multiple smaller dies that can be vertically stacked using through-silicon vias (TSVs). However, 3D integration can result in a high fabrication cost and power density, and low cooling conductivity, creating thermal hotpots and degrading the reliability [2]. To alleviate these issues, the 2.5D integrated approach presents a modular solution by placing several chiplets on an interposer [3], on which inter-chiplet communication is supported. In addition, and similar to 3D SoCs, in such a modular integration, each chiplet can be designed heterogeneously [4] and independently in a short design-time, as off-the-shelf chiplets can be integrated on an interposer [3].

However, 2.5D integration introduces two main challenges, including deadlock avoidance and low reliability due to vertical-link (VL) faults. First, the underlying network in a 2.5D chiplet system (i.e., the intra- and inter-chiplet interconnect) suffers from deadlock, when a cyclic dependency of requests occurs in the network. Conventionally, to avoid deadlock, some turns can be restricted to break the cyclic dependency in a network [5]. However, even when deadlock-freedom is guaranteed in intra-chiplet networks using conventional approaches [5], the inter-chiplet packets can still create some dependencies, and hence deadlock. Second, enabling link and router fault-tolerance is essential in 2.5D chiplet systems [6], but it has not been addressed. Existing fault-tolerant solutions in 2D and 3D networks cannot be applied to 2.5D networks, where deadlock-freedom is more challenging due to higher irregularity in the network. In particular, enabling fault-tolerance in 2.5D chiplet systems requires higher path redundancy to be supported in the routing algorithm, which makes the deadlock-freedom and load-distribution even more complex. Current routing algorithms [7], [8] to address the deadlock in 2.5D chiplet systems limit the VL selection, and hence deteriorate network reliability and performance.

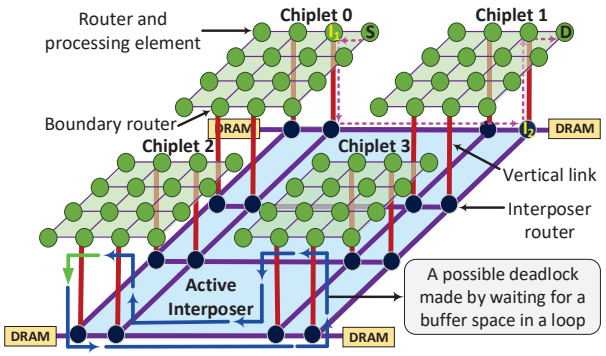


Figure 1. An abstract overview of the baseline 2.5D network with four chiplets on an active interposer.

The novel contribution of this paper is on developing the first Deadlock-free and Fault-Tolerant routing algorithm, called *DeFT*, for 2.5D chiplet systems. In particular, deadlock-freedom is guaranteed by employing a novel virtual-network (VN) assignment strategy to ensure that the network virtual-channel (VC) utilization is highly balanced. In addition, *DeFT* proposes a novel dynamic VL-selection strategy that improves both the fault-tolerance and the load-distribution on the VLs to reduce the network latency. Our results show that *DeFT* can achieve 100% reachability under different VL-fault scenarios (e.g., 25% faults) in chiplet systems, while related work can only tolerate less than 3.1% and 2.1% faults on VLs for systems with 4 and 6 chiplets, respectively. Moreover, *DeFT* improves the latency under real-application traffic by 3% and 13.5%, on average, for relatively low and high traffic scenarios, respectively, with less than 2% area overhead.

II. Background and Related Work

A. Deadlock-Free Routing in 2.5D Chiplet Systems

Fig. 1 shows the baseline 2.5D chiplet system considered in this paper with 4 CPU chiplets on an active interposer. Many integration approaches exist in chiplet systems [1], including different combinations of CPU, GPU, and DRAM chiplets. We consider Fig. 1 as our case study, but our approach can be employed in any chiplet system. In Fig. 1, each chiplet is connected by 4 bidirectional VLs to the interposer, where the

978-3-9819263-6-1/DATE22/©2022 EDAA

routers on chiplets attached to the VLs are called *boundary routers*. Routing in chiplet-based systems uses two intermediate destinations: one on the source chiplet and the other one on the interposer. For example, in Fig. 1, the pink routing path from S on Chiplet 0 to D on Chiplet 1 uses I_1 and I_2 as intermediate destinations. On the source chiplet, a VL is selected as the first intermediate destination. Then, the packet is routed to the selected VL, and from there it is vertically routed to the interposer. Similarly, the second intermediate destination is selected on the interposer for the packet to get to its destination chiplet. Finally, the packet can be routed to its eventual destination on the destination chiplet (Fig. 1).

Deadlock occurs when packets are in a cyclic dependency, waiting on each other to release reserved buffers. For instance, dimension-order routing algorithms(e.g. XY routing in 2D mesh) are deadlock-free [9], because the allowed turns in the network cannot make a cycle. While intra-chiplet routing can be deadlock-free (e.g., using XY routing or conventional turn models [5]), cyclic dependencies can occur when integrating several of these chiplets on an interposer, leading to deadlock. Fig. 1 shows an example for a deadlock scenario occurring from the combination of some turns (shown with blue and green arrows) between deadlock-free Chiplets 2 and 3.

Several routing algorithms have been recently proposed to overcome deadlock in mesh-based 2.5D chiplet systems, namely the modular-turn restriction (MTR) [7] and the remote control (RC) [8] algorithms. MTR employs turn restrictions to avoid some inter-chiplet turns, on the boundary routers, and break cyclic dependencies. As an example in Fig. 1, avoiding the left-to-down turn in Chiplet 2 (shown with the green arrow) can break the cyclic dependency. However, MTR makes the interposer router and chiplet designs dependent, violating the modular design requirement in chiplet-based systems, because each interposer router needs to know whether a packet can reach its destination through a VL while considering the restricted turns. To this end, RC routing [8] breaks the inter-chiplet cyclic dependency by employing an extra buffer (RC-buffer) on the boundary routers to store the whole packet. The RC-buffer is shared among the chiplet routers that utilize the boundary router for inter-chiplet communication. However, RC requires extra hardware for the RC-buffer and a permission network as the RC-buffer is shared among several routers. In addition, both MTR and RC limit VL selection, which restricts them to re-select VLs and makes them unable to tolerate VL faults. Other works on chiplet systems, such as [1], [10], only briefly discuss routing algorithms and consider using virtual-channel-based deadlock avoidance with unbalanced utilization of virtual channels while ignoring VL faults.

B. Fault-Tolerant Routing in 2.5D Chiplet Systems

To the best of our knowledge, there is no prior work on addressing VL faults in 2.5D chiplet systems. In addition, existing routing algorithms proposed for 3D networks—for both fully [11], [12] and partially [2], [13] connected networks—cannot be applied to 2.5D chiplet systems because: 1) in 3D networks, a packet goes from one direction to another vertically (up→down or down→up), while in a 2.5D network, packets go down and then up (up→down→up); i.e., an inter-chiplet packet requires two vertical routings and three intra-layer routings: on the source chiplet, interposer, and destination chiplet; 2) some routers are indirectly connected to other chiplets, and the chiplet and interposer sizes may also be different, which makes the topology more irregular than 3D networks.

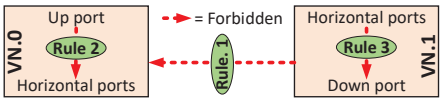


Figure 2. *DeFT*'s rules for VN utilization and deadlock-freedom.

Unlike conventional 2.5D routing algorithms [7], [8] *DeFT*, does not put any limitation on VL selection to realize deadlock-freedom. Therefore, any VL-fault patterns can be tolerated unless the network connectivity is not guaranteed (i.e., chiplets are disconnected). Moreover, *DeFT* includes a novel congestion-aware VL-selection strategy to improve traffic distribution especially in the presence of VL faults.

III. *DeFT*: Proposed Routing Algorithm

A. Proposed VN Separation and Deadlock-Freedom

The idea of employing VN separation for deadlock-freedom is relatively old. However, employing VNs while considering hardware overhead and network latency is quite challenging. Consequently, existing routing techniques [7], [8] in chiplet systems have underestimated the efficiency of VN separation for deadlock-freedom. As we will show, *DeFT* employs two VCs with negligible hardware overhead and fair utilization of VCs, to realize deadlock-freedom and improve hardware efficiency and traffic distribution.

DeFT utilizes two VNs for deadlock-freedom, where at least one VC is required for each VN. Here, we assume one VC per VN, but the number of VCs can be increased without loss of generality. We define "Down" port as the one going from a chiplet to the interposer, "Up" port goes from the interposer to a chiplet, and "Horizontal" ports (East, West, South, and North) are intra-chiplet and intra-interposer ports. The following rules, also shown in Fig. 2, are used for deadlock-free VN utilization:

Rule 1. *Routing from VN.1 to VN.0 is forbidden, while packets can go from VN.0 to VN.1.*

Rule 2. *For packets in VN.0, routing from an Up port to Horizontal ports is forbidden.*

Rule 3. *For packets in VN.1, routing from Horizontal ports to a Down port is forbidden.*

To satisfy these three rules, for *inter-chiplet packets injected from the non-boundary routers*, VN.0 is assigned to the packets in the source router. Such VN.0 assignment should be changed to VN.1 (VN0→VN1) between the first (leaving the source chiplet) and the second (entering the destination chiplet) vertical routing. While these rules guarantee deadlock freedom, VN utilization is efficient due to the following theorems:

Theorem III.1. *Both VNs can be assigned to intra-chiplet packets.*

Proof. Intra-chiplet packets do not use vertical ports (Up or Down ports), and hence they do not face any forbidden routings in VNs stated in Rules 2 and 3. □

Theorem III.2. *Both VNs can be assigned to inter-chiplet packets for routing on the interposer.*

Proof. Packets entering the interposer are in VN.0. They can remain in VN.0 based on Rule 2 (Horizontal to Down is not forbidden in VN.0) or, based on Rule 1, switch to VN.1. □

Theorem III.3. *Inter-chiplet packets on the source chiplet are free to be routed via any VLs towards the interposer.*

Proof. Packets from a source chiplet to the interposer need to go from the Horizontal ports to the Down port of the selected VL, and then from the Down port to the Horizontal ports of the interposer. Inter-chiplet packets on the source chiplet are in VN.0. When using any VL to go to the interposer, 1) if the packet stays in VN.0, based on Rule 2, routing from Horizontal to Down ports, and Down to Horizontal ports are not forbidden; and 2) if the packet is switched to VN.1, based on Rule 1, it can to go to a Down port and, based on Rule 3, it can go from Down to Horizontal ports. □

Theorem III.4. *Inter-chiplet packets on the interposer are free to be routed via any VLs towards the destination chiplet.*

Proof. Routing from the interposer to the destination chiplet is done by going from Horizontal ports to the Up port of the selected VL, and then from the Up port to Horizontal ports of the destination chiplet. Based on Rules 2 and 3, regardless of their VN, packets can go from Horizontal to the Up port. When using any VL to go to the destination chiplet, 1) if a packet is in VN.0, based on Rule 1, it can be switched to VN.1 to go from Up to Horizontal ports; and 2) if the packet is in VN.1, based on Rule 3, it can go from Up to Horizontal ports. □

Based on Theorems III.1 and III.2, the proposed VN separation offers a high balance of VC utilization to improve traffic distribution on VCs while also utilizing a small number of VCs (i.e., two VCs in total). For the cases where both VNs can be assigned, we use round-robin assignment to balance the VN load. Moreover, according to Theorems III.3 and III.4, the choices for VL selection are maximized to tolerate faults on VLs. In addition, as we will discuss in Section III-B, such flexibility in VL selection helps balance load-traffic on VLs and improves the network latency. Algorithm 1 summarizes our VN-assignment strategy. Note that, as shown in Algorithm 1 and Fig. 1, an interposer router can also be a source router (e.g., for the packets injected by DRAMs).

Algorithm 1 Virtual-Network (VN) Assignment in *DeFT*

```
if Current router is Source then
    if Source ∈ {interposer ∪ dest. chip ∪ boundaries} then
        Do round-robin assignment between VN.0 and VN.1
    else if Destination is on a different chiplet then
        Assign VN.0
else if Current router ∈ boundary routers then
    if going to the interposer then
        Do round-robin reassignment between VN.0 and VN.1
    else if coming from the interposer then
        Go to (remain in) VN.1
else
    Stay in the previously assigned VN
```

Deadlock-freedom: *DeFT* is deadlock-free in any 2.5D chiplet system where each chiplet is locally deadlock-free, because of two main reasons: 1) there is no inter-chiplet cyclic dependency in VN.0 and VN.1. In VN.0, based on Rule 2, routing from Up port to Horizontal ports is avoided. In VN.1, based on Rule 3, routing from Horizontal ports to Down port is avoided; and 2) there is no cyclic dependency between VNs because Rule 1 avoids routing from VN.1 to VN.0.

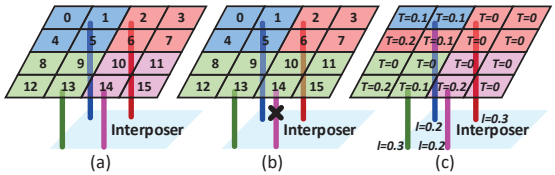


Figure 3. Examples for a VL selection: (a) a fault-free distance-based selection (closest VL), (b) a distance-based selection in the presence of a VL fault, and (c) a good selection under non-uniform traffic. T and l denote the inter-chiplet traffic rate of routers and VLs, respectively. Each tile is a router and router's color represents its selected VL.

Livelock-freedom: *DeFT* is livelock-free in any 2.5D chiplet system where each chiplet is locally livelock-free. The only possible routing for inter-chiplet packets is source-chiplet → interposer → destination-chiplet. This is done by the two intermediate destinations (see Section II-A) where the packets are minimally routed between the intermediate and the main destinations. Intra-chiplet packets are also routed minimally. Therefore, packets are routed in a finite number of hops, and hence *DeFT* is livelock-free.

B. Proposed Fault-Tolerant Congestion-Aware VL Selection

Considering VL faults—due to inevitable mismatch [14], electromigration [15], and thermomigration [14] in microbumps—is essential in 2.5D chiplet systems. For the first time, *DeFT* takes on this challenge that has been ignored in existing routing algorithms. To tolerate VL faults, affecting inter-chiplet routing, *DeFT* supports an adaptive VL selection where VLs are adaptively selected as intermediate destinations (see Section II-A). Fig. 3 shows several examples of VL selection in a chiplet with four VLs where the color of each router (shown as tiles) indicates its selected VL. Here, T and l denote the inter-chiplet traffic rate of routers and VLs, respectively. Traffic rate of a VL (l) is the sum of the traffic rates of the routers (T of routers) selecting the VL.

Fig. 3(a) shows an example of a fault-free distance-based selection (i.e., selecting the closest VL for each router) where the distribution of VLs and traffic pattern are both uniform. In Fig. 3(b), in which one VL is faulty, a distance-based selection approach decides that, for example, eight routers should utilize the green VL while the two other VLs are utilized by four routers each. But such an unbalanced utilization can significantly increase the chance of congestion, degrading the overall performance. A more efficient selection in this example would be for Routers 8 and 11 to utilize the blue and the red VL, respectively. That way, for example, although the blue VL is farther to Router 8, packets injected by Router 8 can benefit from a lower traffic load on the blue VL and latency.

In addition to VL faults, an efficient VL selection should consider the traffic profile into account to properly distribute the load on VLs [16]. Fig. 3(c) shows an example where an efficient

selection is done based on the traffic profile. Applying distance-based selection (similar to Fig. 3(a)) to Fig. 3(c), the loads on each VL will be l_{blue}=0.5, l_{red}=0, l_{green}=0.3, and l_{purple}=0.2. This puts half of the total load on the blue VL while there is no load on the red VL. *DeFT* considers both the VL faults and traffic profile to enable a fault-tolerant and congestion-aware VL selection, as discussed next.

The VL-selection strategy in *DeFT* includes an offline step to analyze optimal VLs under different VL-fault scenarios and an online selection among such pre-analyzed VLs when a fault occurs. During design-time and considering VL-utilization balancing and distance (source to VL hop-count) minimization, we explore different VL selections for all the possible VL-fault scenarios. VL-utilization balancing is important because it relieves over-utilized VLs from extra load. It not only improves the network latency but also increases the system reliability, as over-utilization of VLs can increase stress-migration-based faults [15]. Moreover, distance minimization can also improve energy consumption, as reducing the path length of a packet can reduce dynamic power dissipation in routers due to that packet, as well as its end-to-end latency. Considering these two objective functions, the best resulting VL selections are analyzed and saved in look-up tables in routers to be utilized during run-time. This improves hardware complexity when calculating an optimized selection, providing *DeFT* with an adaptive selection under different VL-fault scenarios. Here, the look-up table size and hardware cost is negligible compared to the overall size of a router (see Section IV-D for cost analysis).

For any inter-chiplet packet, two VL selections are required: one on the source chiplet (towards the interposer) and one on the interposer (towards the destination chiplet). The first selection is done based on the packets generated from the source-chiplet routers. For VL selection on the interposer, packets headed for the destination-chiplet routers affect the selection. As both VL selections are similar, here we only discuss VL selection on the source chiplet for brevity. Given a fault scenario and based on the VL selection on the source chiplet, i.e., the first selection, we consider the source-chiplet routers (r) which are included in the VL-selection process: $R_C = \{r_1, r_2, .., r_R\}$. Selection is done per chiplet, where C is the chiplet and R_C is the chiplet's routers. The chiplet is connected to the interposer using a set of fault-free vertical links (v): $VL_C = \{v_1, v_2, .., v_V\}$. The objective is to find a set of optimized VL selections for all the source-chiplet routers while balancing the VL utilization: $s^* = \{V_{r_1}, V_{r_1}, ..., V_{r_R}\}$, where V_{r_i} is the VL selected for router i.

To balance VL utilization, the load on each VL should be close to the average load on all the VLs. The load on VL_i depends on the inter-chiplet traffic (packet injection) rate among the routers that select VL_i. The load on VL v is:

$$l_v = \sum_{r \in R_C} T_r^{inter} \times U_r^v, \tag{1}$$

where T_r^{inter} is the inter-chiplet traffic rate of router r. Also, $U_r^v = 1$ ($U_r^v = 0$) whenever router r utilizes (does not utilize) VL v for vertical routing. Based on (1), the average load over all the VLs is:

$$l_{avg} = \frac{1}{V} \sum_{v \in VL_C} l_v. \tag{2}$$

Considering (1) and (2), we define the load cost of a VL v as:

$$L_v = \left| \frac{l_v - l_{avg}}{l_{avg}} \right|. \tag{3}$$

The second objective in VL selection is distance minimization. Considering a mesh network, the distance (i.e., hop count) between a router r to a VL v (on the same chiplet) is:

$$D_v^r = |x_r - x_v| + |y_r - y_v|, \tag{4}$$

where x_r and y_r are the coordinates of the router r, and x_v and y_v are the coordinates of the VL v. The distance cost of a VL v to the routers that select v is:

$$D_v = \sum_{r \in R_C} D_v^r \times U_r^v. \tag{5}$$

Based on (3) and (5), the overall cost of a selection set s is:

$$C_s = \sum_{v \in VL_C} (\rho \times D_v) + L_v, \tag{6}$$

where ρ can decide the importance of the load-balancing versus distance objectives. In our analyses (see Section IV), we experimentally found $\rho = 0.01$ to be efficient in *DeFT*. Based on the cost function C_s in (6), an optimization search O can be used to find the optimal selection set s^* with the minimum cost C_s^* (Algorithm 2):

$$s^* \leftarrow O(s \in S, \text{Objective} : \min(C_s)). \tag{7}$$

Here, S denotes all the possible selection sets. We used an exhaustive search to address the optimization in (7) because the search space is small. In large networks with a large design space, efficient search algorithms should be used to reduce the optimization complexity. Our proposed VL-selection algorithm is shown in Algorithm 2. The algorithm is done for different VL-fault scenarios at design-time and the selection sets (s^*) are saved in routers for run-time use. For the baseline system (Fig. 1), where each chiplet has 4 VLs, there are 14 combinations of faults ($\binom{4}{1} + \binom{4}{2} + \binom{4}{3}$). Therefore, 14 VL addresses are saved in each router.

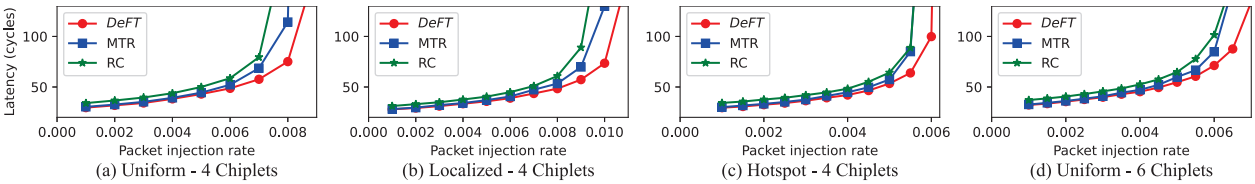
Algorithm 2 Vertical-Link (VL) Selection in *DeFT*

```
for each s in all the possible selection sets (s ∈ S) do
  for v ∈ VL_C do
    L_v ← find load cost of v (using (3))
    D_v ← find distance cost of v (using (5))
  C_s ← find the overall cost of selection set s (using (6))
  if C_s < C_s^* then
    C_s^* ← C_s
    Update the saved selection sets (s^* ← s)
```

IV. Evaluation and Simulation Results

A. Simulation Environment and Configuration

We enhanced the Noxim simulator [17], which is a cycle accurate simulator for networks-on-chip, to support 2.5D chiplet systems as shown in Fig. 1, where each chiplet is connected using four bidirectional VLs to the interposer. Each chiplet is connected to interposer with 4 VLs rather than fully connected, because reducing number of VLs alleviates fabrication costs. To further reducing VLs and improving fabrication cost serialization can be employed [18]. According to [7], for a 4×4



(a) Uniform - 4 Chiplets (b) Localized - 4 Chiplets (c) Hotspot - 4 Chiplets (d) Uniform - 6 Chiplets

Figure 4. Average latency comparison among *DeFT*, MTR, and RC routing algorithms when applied to the network shown in Fig. 1 and under (a) Uniform, (b) Localized, and (c) Hotspot synthetic traffic patterns. (d) shows the same but for 6 chiplets and Uniform traffic.

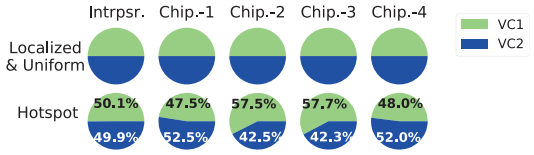
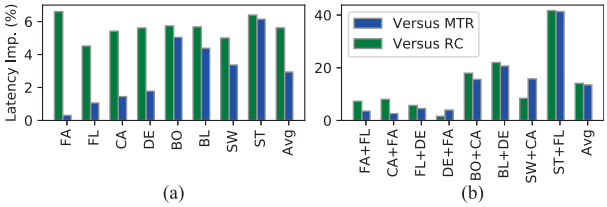


Figure 5. VC utilization in *DeFT* under synthetic traffic.



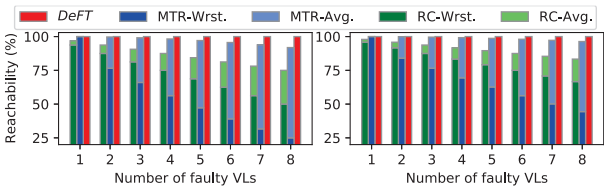
(a) (b)

Figure 6. Latency improvement under real-application traffic using (a) a single application, and (b) two applications simultaneously.

chiplet, placing the four VLs on the borders of the chiplet is an optimal choice when considering hardware complexity and network latency. Note that *DeFT* efficiency is independent of the placement and density of VLs as it does not use turn models for achieving inter-chiplet deadlock-freedom. Besides the baseline 4-chiplet system, we simulated a system with 6 chiplets to show how *DeFT*'s efficiency scales with system size. We compare *DeFT* against state-of-the-art routing algorithms for chiplet-based systems, MTR [7] and RC [8]. While MTR and RC have no requirement for the number of VCs, two VCs are used in all the algorithms to have a fair comparison with *DeFT* (i.e., considering a single VC will make MTR and RC perform worse). A packet size of eight flits and a buffer size of four flits are considered, where a flit width is 32 bits. We performed offline VL-selection optimization considering Uniform traffic, the most pessimistic assumption, while our simulations include different traffic scenarios. Including traffic information in the offline optimization results in further improvements.

B. Latency Analysis

The latency analysis for synthetic traffic is shown in Fig. 4. Compared to MTR and RC, *DeFT* has the lowest average latency because of its balanced VL selection and VC utilization. In Localized traffic, shown in Fig. 4(b), for 40% of the packets, the source and destination are on the same chiplet, i.e., intra-chiplet packets. *DeFT* shows a low latency because the VC utilization for intra-chiplet packets is highly distributed (see Theorem.III.1). In Hotspot traffic, shown in Fig. 4(c), *DeFT* shows slightly lower improvement because incoming packets to chiplets are allowed to only use the second VC, which can result in a brief back-pressure. However, we used a relatively high rate of hotspot in our simulation (3 hotspot points with



(a) 4 Chiplets (b) 6 Chiplets

Figure 7. Reachability in the presence of VL faults in a system with (a) 4 chiplets (total VLs=32), and (b) 6 chiplets (total VLs=48). Note that *DeFT*-Wrst. and *DeFT*-Avg. are the same (both shown by *DeFT*).

10% rate on each). As shown in Fig. 4(d), where the 6-chiplet system is considered, *DeFT* performance is not limited by system size. The distribution of VC utilization for synthetic traffic patterns are shown in Fig. 5. VC utilization is balanced (50% with less than 0.4% tolerance) in Uniform and Localized traffic, hence results for both are presented in the same chart. In Hotspot traffic, although hotpot rate is relatively high, deviation in VC utilization is less than 8%. Balanced VC utilization in *DeFT* offers a low average latency, while avoiding deadlock.

To consider real-application traffic, we generated traffic from PARSEC benchmarks [19] using GEM5 [20], and simulated the generated traffic with our chiplet-based Noxim simulator. We simulated the benchmarks in full-system mode with 64 x86 cores, four coherence directories, and four shared L2 cache banks (each core also has a private L1 cache). The latency improvement in eight PARSEC applications (blackscholes, bodytrack, canneal, dedup, facesim, fluidanimate, streamcluster, and swaptions) is shown in Fig. 6, where the first two letters of each application are used on the x-axis. Because of low congestion in the network when running a single application, we also simulated two applications running simultaneously, where each application executed on 32 cores (see Fig. 6(b)). On average, *DeFT* shows more improvement when multiple applications are considered due to the higher chance of network congestion. On the x-axis in Fig. 6(b), the two-application combinations are sorted based on traffic load, from low (FA+FL) to high (ST+FL). For high traffic loads, *DeFT* shows a notable improvement of up to 40% compared to both MTR and RC.

C. Fault-Tolerance Analysis

To assess *DeFT*'s ability to tolerate faults, we analyze network reachability in the presence of faults, similar to [13]. In VL-fault scenarios, reachability is defined as the ratio of packets that can be successfully routed, to the total number of injected packets. As shown in Fig. 7(a), *DeFT* achieves complete (i.e., 100%) reachability for the considered fault-injection rates. We injected all combinations of fault patterns excluding those that disconnected chiplets completely (i.e., when all VLs of a chiplet

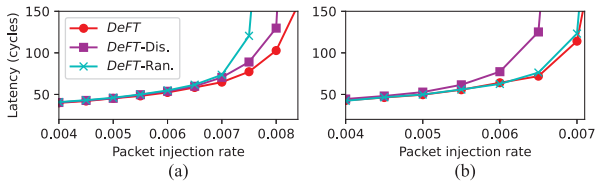


Figure 8. Average latency in *DeFT* with different VL-selection strategies and fault-injection rates for a 4-chiplet system. (a) VL fault-rate of 12.5% (4 faulty VLs), and (b) VL fault-rate of 25% (8 faulty VLs).

Table I
AREA AND POWER ANALYSIS OF *DeFT*, MTR, AND RC

	MTR	$RC_{non\text{-}bndry}$	RC_{bndry}	*DeFT*
Router area (μm^2)	45878	46663	51984	46651
Norm. router area	1	1.017	1.133	1.016
Router power (mW)	11.644	11.76	12.841	11.693
Norm. router power	1	1.009	1.102	1.004

are faulty). The figure shows both the average and the worst-case reachability under an injected fault rate of 3.125% to 25% (1–8 faulty VLs). Considering the average case, *DeFT* improves the network reachability by 3.1–25% compared to RC, and by 0–8.1% compared to MTR. In the worst case, *DeFT* improves the network reachability by 6.25–50% compared to RC, and by 0–75% compared to MTR. Fig. 7(b) shows network reachability with the 6-chiplet system. Here, *DeFT* can achieve 100% reachability for the fault injection rates shown, while MTR provide 100% reachability for only the low fault-rate scenario with 2.1% faults (one faulty VL). Also, RC cannot tolerate any faults. The restricted turns in MTR and the permission network of RC limit their VL selection and, therefore, fault-tolerance.

Fig. 8 shows the impact of VL faults on latency. Here, MTR and RC are not considered because they are unable to offer a complete reachability under fault scenarios. Moreover, in addition to using *DeFT* with the proposed VL selection (*DeFT* in Fig. 8), we include the average latency in *DeFT* with distance-based (*DeFT*-Dis.)—a common approach in 3D networks [13]—and random (*DeFT*-Ran.)—a VL is selected randomly among eligible VLs—VL-selection strategies. Here, *DeFT* with distance-based selection (*DeFT*-Dis. in Fig. 8) means that the routing is *DeFT* while VL selection is based on distance instead of our proposed selection. *DeFT* shows significantly lower latency than those with random- and distance-based VL selections under 12.5% and 25% fault injection rates, respectively. Random selection offers a good load distribution when the number of faulty VLs is large (e.g., under 25% fault rates). But random selection imposes latency overhead under 12.5% fault rates (see Fig. 8(a)). As shown in Fig. 8(b), latency in *DeFT* is still lower under 25% fault rates.

D. Hardware and Power Analysis

We used Cadence Genus and ORION 3.0 [21] to estimate router area and power at the 45 nm technology and considering a 1 GHz clock frequency. Table I compares the area and power estimation of a six-port *DeFT* router with routers used in MTR and RC. Note that the hardware implementations of the non-boundary and boundary routers are different in RC, hence we report both separately in the table. From Table. I, *DeFT* imposes less than 2% and 1% hardware and power overhead, relatively, at most compared to related work. The small overhead includes the logic for VN-assignment algorithm and look-up tables to store data for fault-tolerant VL selection.

V. CONCLUSION

This paper presents the first deadlock-free and fault-tolerant routing algorithm, called *DeFT*, for 2.5D chiplet networks. *DeFT* offers VC-based deadlock-freedom where not only are packets free to select any VL, but also VC utilization is efficiently balanced. Freedom in VL-selection allows *DeFT* to tolerate any pattern of VL faults. To improve network congestion in VL-fault scenarios, *DeFT* employs a dynamic traffic-aware VL selection to enhance run-time routing efficiency. Compared to the state-of-the-art routing algorithms, our simulation results show that *DeFT* improves network reachability by up to 75% with a fault rate of up to 25% and reduces the network latency by up to 40% for multi-application execution scenarios with less than 2% area overhead. These results highlight the promise of *DeFT* for improving emerging chiplet systems.

ACKNOWLEDGMENT

This work was supported by the National Science Foundation (NSF) under grant number CNS-2046226.

REFERENCES

[1] A. Kannan *et al.*, "Enabling interposer-based disintegration of multi-core processors," in *MICRO*, 2015.
[2] E. Taheri *et al.*, "Addressing a new class of reliability threats in 3-D network-on-chips," *IEEE TCAD*, vol. 39, no. 7, pp. 1358–1371, 2019.
[3] J. Kim *et al.*, "Architecture, chip, and package co-design flow for 2.5D IC design enabling heterogeneous IP reuse," in *DAC*, 2019.
[4] Y. Ma *et al.*, "TAP-2.5D: A thermally-aware chiplet placement methodology for 2.5D systems," in *DATE*, 2021.
[5] M. Ebrahimi *et al.*, "EbDa: A new theory on design and verification of deadlock-free interconnection networks," in *ISCA*, 2017.
[6] R. Wang *et al.*, "Pre-bond testing of the silicon interposer in 2.5D ICs," in *DATE*, 2016.
[7] J. Yin *et al.*, "Modular routing design for chiplet-based systems," in *ISCA*, 2018.
[8] P. Majumder *et al.*, "Remote control: A simple deadlock avoidance scheme for modular systems-on-chip," *IEEE TC*, 2020.
[9] C. J. Glass *et al.*, "The turn model for adaptive routing," *ACM SIGARCH Computer Architecture News*, vol. 20, no. 2, pp. 278–287, 1992.
[10] S. Bharadwaj *et al.*, "Kite: A family of heterogeneous interposer topologies enabled via accurate interconnect modeling," in *DAC*, 2020.
[11] S. Pasricha *et al.*, "A low overhead fault tolerant routing scheme for 3D networks-on-chip," in *ISQED*, 2011.
[12] M. Ebrahimi *et al.*, "Fault-tolerant routing algorithm for 3D NoC using hamiltonian path strategy," in *DATE*, 2013.
[13] R. Salamat *et al.*, "A resilient routing algorithm with formal reliability analysis for partially connected 3D-NoCs," *IEEE TC*, vol. 65, no. 11, pp. 3265–3279, 2016.
[14] Y. Li *et al.*, *3D microelectronic packaging: From fundamentals to applications*. Springer, 2017, vol. 57.
[15] H. Hsiao *et al.*, "Electromigration reliability and morphologies of Cu pillar with microbump under high current density stressing," in *EPTC*, 2015.
[16] E. Taheri *et al.*, "Adele: An adaptive congestion-and-energy-aware elevator selection for partially connected 3D NoCs," in *DAC*, 2021.
[17] V. Catania *et al.*, "Cycle-accurate network on chip simulation with noxim," *ACM TOMACS*, vol. 27, no. 1, pp. 1–25, 2016.
[18] S. Pasricha, "Exploring serial vertical interconnects for 3D ICs," in *DAC*, 2009.
[19] C. Bienia and K. Li, *Benchmarking modern multiprocessors*. Princeton University Princeton, NJ, 2011.
[20] N. Binkert *et al.*, "The Gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.
[21] A. B. Kahng *et al.*, "ORION3.0: A comprehensive NoC router estimation tool," *IEEE Embedded Systems Letters*, vol. 7, no. 2, pp. 41–45, 2015.