ReD: A Reliable and Deadlock-Free Routing for 2.5D Chiplet-based Interposer Networks

Ebadollah Taheri, *Student Member, IEEE,* Sudeep Pasricha, *Fellow, IEEE,* and Mahdi Nikdast, *Senior Member, IEEE*

***Abstract*—2.5D integration offers a cost-effective and reliable solution for implementing large-scale modular systems. A 2.5D chiplet system can be designed by connecting smaller chiplets through an interposer, where the chiplets may have heterogeneous architectures. In addition to the intra-chiplet network (e.g., a network-on-chip on the chiplet), a network is used on the interposer to enable efficient and scalable communication among different chiplets. However, this global network, which consists of intra-chiplet and inter-chiplet networks, is susceptible to deadlock, despite using deadlock-free networks on the chiplets and interposer. Moreover, 2.5D networks are not only vulnerable to horizontal link (HL) faults but also to those in the vertical links (VLs) connecting the chiplets to the interposer. In addition, such faults cannot be effectively addressed by existing fault-tolerant routing techniques designed for 2D and 3D networks-on-chip. To overcome these challenges, this paper introduces a novel Reliable and Deadlock-free routing algorithm, called *ReD*, for fault-tolerant communication in 2.5D chiplet systems. *ReD* leverages a virtual-network-based approach to guarantee deadlock freedom while tolerating VL and HL faults. Besides VL faults, due to the difference in VL technology (i.e., microbump technology), the number of VLs connecting a chiplet to the interposer is limited, making VLs a source of congestion. *ReD* enhances VL selection in such scenarios to tolerate VL faults and improve network congestion by balancing VL utilization. Compared to the state-of-the-art routing algorithms, simulation results obtained by simulating chiplet systems under HL and VL faults demonstrate that *ReD* significantly improves the network reachability by up to 75% and reduces the network latency by up to 40%, while incurring less than 2% area overhead.**

***Index Terms*—Chiplet Systems, NoCs, Deadlock Freedom, Fault-Tolerance.**

I. Introduction

Large-scale systems-on-chip (SoCs) with increasing heterogeneity and integration density have recently received much attention due to the growing demand for high computation capabilities [1], especially for emerging machine learning applications. However, conventional 2D SoCs lack scalability because manufacturing yield significantly decreases as the chip size increases [2]. Therefore, high manufacturing costs are imposed when attempting to manufacture a system with high computation capabilities on a single large-scale chip [2]. To this end, 3D and 2.5D integration partitions an SoC into multiple smaller dies that are connected using different technologies. In the 3D integration, the dies are vertically stacked and interconnected using through-silicon vias (TSVs) to enable higher integration density and improved performance. However, 3D integration can result in high fabrication costs, power density, and low cooling conductivity, creating thermal hotspots and degrading the system reliability [3]. On the other hand, the 2.5D integrated approach presents a modular solution by placing several chiplets on an interposer on which inter-chiplet communication is supported [4], [5]. In addition, and similar to 3D SoCs, in such a modular integration, each chiplet can be designed heterogeneously and independently [6]–[8]. Moreover, the design time of chiplet systems is significantly shorter than the traditional monolithic designs since off-the-shelf chiplets can be integrated on an interposer to make chiplet systems with different computation capabilities for various purposes [4], [9]. Therefore, chiplet-based systems can leverage the strengths of different manufacturing processes and technologies for each chiplet. However, 2.5D integration introduces different challenges, including deadlock avoidance [10] and fault tolerance [11] due to both vertical link (VL; connecting chiplets to the interposer) and horizontal link (HL; on the chiplets and the interposer) faults.

The intra- and inter-chiplet networks in a 2.5D chiplet system can suffer from deadlock when a cyclic dependency of requests for buffers occurs in the routing process. Conventionally, to avoid deadlock, some turns (i.e, some requests for buffers in specific directions) can be restricted to break the cyclic dependency in a network [12]. However, even when deadlock-freedom is guaranteed in intra-chiplet networks using conventional approaches [12], the inter-chiplet packets can still create some dependencies, and hence deadlock. In [10], a modular-turn restriction (MTR) routing algorithm was proposed where the routing on each chiplet is designed separately by avoiding some turns from the chiplet to the interposer and vice versa. However, the interposer network needs to be aware of the restricted turns within chiplets. This violates a key attribute in 2.5D systems to enable the design of each chiplet and the interposer network independently without the knowledge of the whole design [13]. Moreover, the turn restrictions limit routing adaptation and impose performance overhead by creating imbalances in load among routers, especially on the boundary routers that connect the chiplets to the interposer. To this end, in the Remote Control (RC) routing algorithm [13], deadlock-freedom of chiplet systems is guaranteed by reserving an extra buffer on the boundary routers to store and forward the inter-chiplet packets. However, RC not only imposes high area and power overhead due to the extra resources needed but also limits VL selection and path diversity in the routing process.

In addition to deadlock-freedom, enabling link and router fault-tolerance is essential in 2.5D chiplet systems [14], but it has not yet 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 Networks-on-Chip (NoCs) requires higher path redundancy to be supported in the routing algorithm [15], which when

The authors are with the Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, USA.
E-mail: {ebad.taheri,sudeep,mahdi.nikdast}@colostate.edu

extended to 2.5D chiplet systems makes the deadlock-freedom and load-distribution even more complex. Current routing algorithms [10], [13] to address deadlock in 2.5D chiplet systems limit the VL selection and hence deteriorate network reliability and performance. Furthermore, in 2.5D systems, it is imperative to acknowledge the inherent limitations in bandwidth at the boundaries of chiplets, particularly on the VLs connecting chiplets and the interposer. To mitigate potential congestion at these boundaries, the load distribution strategy becomes paramount. This becomes especially critical when considering fault scenarios, where a fault occurrence can lead to a reduction in the chiplet-to-interposer bandwidth. In such cases, distributing traffic efficiently across the available bandwidth is essential to alleviate and prevent congestion.

To address the aforementioned challenges and constraints, this paper develops a Reliable and Deadlock-free routing algorithm, called *ReD*, for 2.5D chiplet-based interposer networks. In particular, the main contributions of this paper are:

- We propose a novel virtual-network (VN) based approach for 2.5D networks to guarantee deadlock-freedom in both the intra- and inter-chiplet networks while offering high-performance communication. The VN-based approach uses a novel VN assignment strategy to maximize network virtual-channel (VC) utilization balance.
- We present a novel pseudo-dynamic VL-selection strategy that not only enables *ReD* to tolerate VL faults but also improves the load distribution on the VLs to reduce the network latency under fault scenarios.
- We propose a Hamiltonian-based routing technique to share the faulty/healthy status of VLs among all routers in the local chiplet/interposer. The presented mechanism is designed to tolerate HL faults without the need for any extra control network or VCs.

ReD only requires two VNs without imposing any limitations on VL selection. VN assignment in *ReD* is performed in such a way that intra-chiplet packets and inter-chiplet packets on the interposer are free to be routed using each of the two VNs. The freedom in VN assignment enables *ReD* to tolerate HL faults without adding any extra VCs. Our experimental results show the promise of *ReD*; it can achieve 100% reachability under different VL-fault scenarios (e.g., 25% faults) in chiplet systems of different sizes, ranging from 4 to 12 chiplets. In addition, with less than 2% area overhead, *ReD* improves the latency under real-application traffic on average by 3% and 13.5% for low and high traffic scenarios, respectively.

II. Background and Related Work

A. Active interposers and their challenges

There are two main types of interposers: active and passive. Passive interposers act as simple substrates with wiring layers that enable connectivity between chiplets. They are generally less complex and have lower power consumption compared to active interposers. However, passive interposers may face limitations in realizing flexible and efficient long-distance communication, especially when integrating a substantial number of chiplets. In contrast, active interposers incorporate additional circuitry, such as routers and power management units, enabling them to perform more complex functions. They

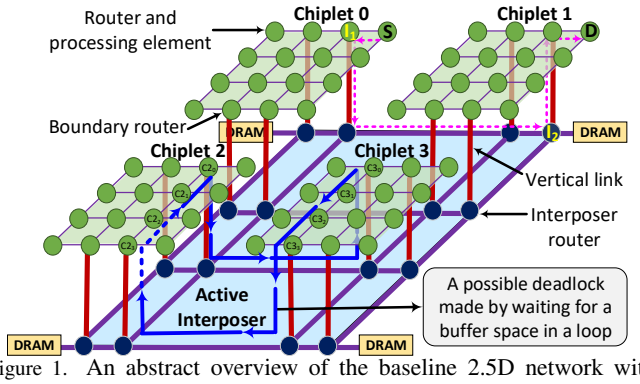

Figure 1. An abstract overview of the baseline 2.5D network with four chiplets on an active interposer.

offer advantages such as facilitating communication among chiplets with different protocols, supporting scalable cache coherency, and providing new opportunities for performance enhancement [16]. Despite their benefits, active interposers pose challenges, particularly in yield, power, and thermal issues. However, the lower complexity of the circuits on the interposer, including routers and circuits for power management and testing, contributes to a higher yield compared to the chiplets. Specifically, in [16], the interposer's complexity is 99.5% less than the chiplet complexity in terms of the number of transistors. This results in low power consumption in the interposer and mitigates thermal concerns due to the lower heat generation and heat confinement of the thinner silicon layer. Furthermore, employing older mature technology for the interposer enhances yield, as highlighted in [17].

B. Deadlock-Free Routing in Active Interposers

An example of a 2.5D chiplet system is shown in Fig. 1. Note that we also use this architecture as our case study in our evaluation presented in Section IV. Various integration approaches are available for chiplet systems that combine CPU, GPU, and DRAM chiplets [2]. Although we focus on the configuration shown in Fig. 1, our methodology can be applied to any chiplet system. The system in this example consists of an active interposer and four CPU chiplets. Each chiplet is connected to the interposer through four bidirectional VLs. The routers located on the chiplets that are connected to the VLs are referred to as *boundary routers*.

Routing in chiplet-based systems involves two intermediate destinations: one on the source chiplet and another on the interposer. For instance, in Fig. 1, the magenta routing path from node *S* on Chiplet 0 to node *D* on Chiplet 1 uses I_1 and I_2 as intermediate destinations. Initially, a VL is selected on the source chiplet as the first intermediate destination. The packet is then routed to the selected VL and vertically transmitted to the interposer. Subsequently, the second intermediate destination is chosen on the interposer to direct the packet toward its destination chiplet. Finally, the packet reaches its intended destination on the destination chiplet.

In the process of routing, when an incoming packet reaches a router's buffer, it can move forward by selecting an output port, based on the routing algorithm and the availability of the next router's buffer. However, waiting for buffer availability can create a cyclic dependency or deadlock, where packets are stuck in a loop, waiting for each other to release reserved

buffers. To prevent deadlock, certain routing algorithms like dimension-order routing (e.g., XY routing in a 2D mesh network topology) have been proven to be deadlock-free according to Glass et al. [18]. These algorithms restrict some turns in the network to prevent cycles. Although intra-chiplet and/or inter-chiplet network routing can be deadlock-free by using XY routing or conventional turn models as described in [12], integrating multiple chiplets on an interposer can introduce cyclic dependencies in the global network, which includes both intra-chiplet and inter-chiplet networks, resulting in deadlock. An illustration of a deadlock scenario caused by specific turns (indicated by blue arrows) between the deadlock-free Chiplet 2, interposer, and Chiplet 3 is depicted in Fig. 1. This cycle may occur with the following packet routing: $C3_2$ to $C2_2$, $C2_1$ to $C3_1$, $C3_0$ to $C3_3$, and $C2_3$ to $C2_0$.

Several routing algorithms have been proposed recently to address the issue of deadlock in 2.5D chiplet systems based on the mesh architecture. Two notable deadlock-free routing algorithms are the Modular-Turn Restriction (MTR) algorithm [10] and the Remote Control (RC) algorithm [13]. The MTR routing algorithm prevents certain inter-chiplet turns at the boundary routers, effectively breaking cyclic dependencies. For instance, in Fig. 1, avoiding the left-to-down turn in Chiplet 2 (indicated by the dashed blue arrow) can resolve the cyclic dependency. However, the MTR algorithm introduces a dependency between the interposer router and chiplet designs, which goes against the modular design requirement in chiplet-based systems. This is because each interposer router needs to know whether a packet can reach its destination through a VL while considering the turn restrictions. The RC routing algorithm [13] addresses the inter-chiplet cyclic dependency by incorporating an additional buffer called the RC-buffer in the boundary routers. The RC-buffer stores and forwards the entire packet and is shared among the chiplet routers that utilize the boundary router for inter-chiplet communication. The store and forward technique can break the dependency at the boundary router. However, implementing RC requires additional hardware for the RC-buffer and a permission network since the RC-buffer is shared among multiple routers. Moreover, both MTR and RC algorithms restrict VL selection, which limits their ability to dynamically re-select VLs and renders them unable to tolerate VL faults. In [19], Wu et al. proposed a deadlock recovery approach for 2.5D chiplet networks. However, while deadlock recovery approaches can potentially result in high performance for low-load traffic, they suffer from latency and energy overhead in high-load traffic due to the frequent need to recover from deadlocks in the network. Other research on chiplet systems, such as Kannan et al. [2] and Bharadwaj et al. [20], briefly touch upon routing algorithms and propose VC-based deadlock avoidance with unbalanced utilization of VCs. However, they also do not address VL and HL faults in their considerations.

C. Fault-Tolerant Routing in Conventional 2D Networks

Several fault detection techniques have been proposed for NoCs, such as those in [21] and [22]. To tolerate faults in 2D NoCs, increasing path redundancy in routing proves to be efficient. Ebrahimi et al. [23] introduced a deadlock-free routing approach that enhances path redundancy. Meanwhile, VCs are utilized in [24] and [15] to tolerate router and link faults. NARCO [25] generates redundant packets for fault tolerance, employing odd–even (OE) and inverted odd–even (IOE) turn models. OE+IOE [26] and NS-FTR [27] enhance reliability by detouring around faults and proactive packet replication. However, these approaches neglect load balancing over VCs, impacting traffic congestion.

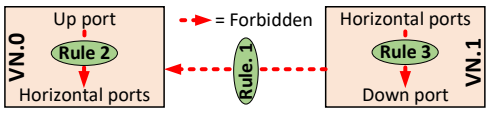

Figure 2. *ReD*'s rules for VN utilization and deadlock-freedom.

D. Fault-Tolerant Routing in 2.5D Chiplet Systems

To the best of our knowledge, no previous work has specifically addressed the issue of VL and HL faults in 2.5D chiplet systems. Furthermore, the existing routing algorithms proposed for 3D NoCs, including both fully connected networks [23] and partially connected networks [3], [28], cannot be directly applied to 2.5D chiplet systems due to two main reasons: 1) In 3D NoCs, packet routing can occur vertically from one layer to another (either up→down or down→up). In contrast, in a 2.5D network, packets follow a different routing pattern where they go down and then up (up→down→up). Consequently, an inter-chiplet packet in a 2.5D chiplet system requires two vertical routings and three intra-layer routings (on the source chiplet, interposer, and destination chiplet); 2) The connectivity between routers in 2.5D chiplet systems may be indirect, and the size of chiplets and interposer can differ, resulting in a more irregular topology compared to 3D NoCs.

In our prior work [11], we introduced DeFT, a fault-tolerant and deadlock-free routing algorithm designed for 2.5D integrated chiplet systems. DeFT improves redundancy in VL selection to tolerate VL faults while maintaining balanced vertical-link utilization in fault scenarios. However, DeFT relied on the assumption that each router had access to fault information from all routers, guiding its selection optimization. We propose *ReD* which is a significant extension of our proposed DeFT approach in [11]. *ReD* utilizes two VNs without imposing restrictions on VL selection where *ReD* can route both intra-chiplet packets and inter-chiplet packets on the interposer using either of the two VNs. *ReD*'s flexibility empowers it to tolerate HL faults without requiring additional virtual channels, effectively handling the majority of HL faults with minimal overhead. Moreover, we introduce a Hamiltonian-based routing mechanism that efficiently shares information about faulty or healthy VLs among all routers within the local chiplet or the interposer. This mechanism is designed to handle HL faults without extra control networks or VCs, further enhancing *ReD*'s fault tolerance capabilities.

III. *ReD*: Proposed Routing Algorithm

In this section, we discuss our VN-separation approach to enable deadlock-freedom. This VN-separation enables adaptation in VL selection to realize a fault-tolerant and congestion-aware VL selection, as discussed in Section III-B. The effective VN separation in *ReD* also enables flexibility to

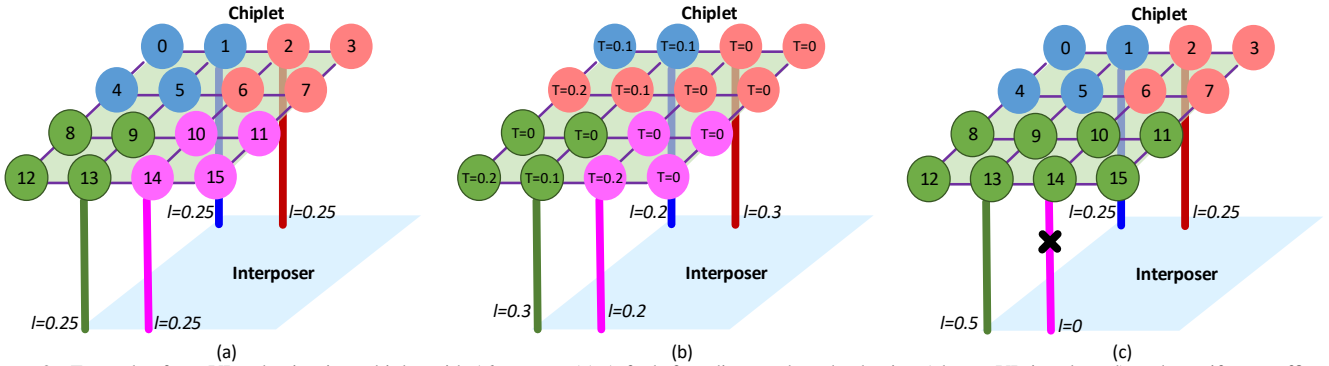

Figure 3. Examples for a VL selection in a chiplet with 16 routers: (a) A fault-free distance-based selection (closest VL is selected) under uniform traffic, (b) A good selection under non-uniform traffic, and (c) A distance-based selection with a VL fault and under uniform traffic, suffering from imbalanced traffic load on VLs. Here, T and l denote the inter-chiplet traffic rate of routers and VLs, respectively. Also, a router's color represents its selected VL.

effectively utilize VCs for the majority of packets in intra-chiplet/interposer routing. Therefore, in Section III-C we discuss *ReD*'s approach in tolerating HL faults without using any extra VCs. Subsequently, Section III-D outlines our methodology for sharing fault updates across routers. Lastly, in Section III-E we explore how *ReD* effectively guarantees deadlock- and livelock-freedom.

A. Proposed VN Separation and Deadlock-Freedom

The idea of employing VN separation for deadlock-freedom, in the context of 2D networks, is relatively old. However, employing VNs while considering hardware overheads and network latency, especially in 2.5D networks, is quite challenging. Consequently, existing routing techniques [10], [13] in chiplet systems have underestimated the efficiency of VN separation for deadlock-freedom. As we will show, *ReD* employs two VNs with negligible hardware overhead and fair utilization of VNs, ensuring its effectiveness even with a limited number of VCs, simplifying hardware requirements, and enhancing traffic distribution, all while tolerating both HL and VL faults.

ReD 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. In cases where multiple VCs are utilized per VN, packets are distributed among the VCs in a round-robin manner to ensure a fair distribution of the load. 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 defined 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. This is because to route from the chiplet to the interposer, routing from Horizontal ports to down ports is required, while the routing is avoided in the VN.1 (Rule 3) and switching to the other VN is also avoided (Rule 1). Such VN.0 assignment is changed to VN.1 (VN.0→VN.1) 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 also 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 fault-free VLs toward 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, if the packet stays in VN.0, based on Rule 2, routing from Horizontal to Down ports, and Down to Horizontal ports is not forbidden. If the packet is switched to VN.1, based on Rule 1, it can 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 fault-free VLs toward 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 the 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, 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. 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 careful balance of VC utilization to improve

traffic distribution on VCs or tolerate HL faults (see Section III-C), while also utilizing a small number of VCs (i.e., two VCs in total). For the cases where both VNs can be assigned, round-robin assignment can be used 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 operates locally on each router with $O(1)$ time complexity, involving simple if-else conditions. Space complexity is negligible and included in our hardware analysis (see section IV-D). Moreover, since the algorithm is local, scaling up the system does not affect its input size, time, or space complexity.

Algorithm 1 VN Assignment in *ReD*

```
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
```

B. Proposed Fault-Tolerant Congestion-Aware VL Selection

1) Motivation for Fault-Tolerant and Traffic-Aware ReD: So far, we have presented our VN-based technique that provides high path redundancy while guaranteeing deadlock-freedom at a low cost. In this section, we focus on the fault tolerance and traffic challenges, which *ReD* can effectively address thanks to its flexibility, as discussed next.

First and foremost, it is essential to consider VL faults in 2.5D chiplet systems, which may arise due to inevitable microbump misalignment [29], electromigration in microbumps [30], and thermomigration in microbumps [29]. For the first time, *ReD* takes on this challenge that has been ignored in existing routing algorithms. To tolerate VL faults affecting inter-chiplet routing, *ReD* supports an adaptive VL selection where VLs are adaptively selected as intermediate destinations (see Section II-B). In addition to VL faults, an efficient VL selection should take the traffic profile into account to properly distribute the load on VLs [31].

In Fig. 3, we show several examples in a chiplet with 16 routers to illustrate how VL selection can significantly affect traffic distribution and fault tolerance in 2.5D networks. In each example, VL selection of a chiplet with four VLs is shown, where the color of each router indicates its selected VL. Here, T and l denote the inter-chiplet traffic rate of routers and VLs, respectively. The traffic rate of a VL (i.e., l) is the sum of the traffic rates of the routers (i.e., T of routers) selecting the VL. As is shown in Fig. 3(a), in the distance-based selection, where the closest VL to the source router is selected, traffic is well distributed on VLs when the traffic generated by the routers is uniform and VLs are fault-free. In this example, four routers are sharing a VL, and as the generated traffic by the groups of routers is assumed to be uniform, the traffic on the VLs is also uniform. However, when applying distance-based selection under non-uniform traffic, it may lead to congestion on specific links. For example, assuming a distance-based selection like the selection in Fig. 3(a), the non-uniform traffic in Fig. 3(b) imposes half of the total load on the blue VL, leaving no load on the red VL. On the other hand, with an effective VL selection, as illustrated in Fig. 3(b), the loads on each VL will be l_{blue}= 0.2, l_{red}=0.3, l_{green}= 0.3, and l_{magenta}= 0.2, demonstrating a more favorable load distribution. Therefore, disregarding the traffic generated by routers during the VL selection process can result in a significantly imbalanced traffic distribution on the VLs.

In the example shown in Fig. 3(c), in the presence of a VL fault and under uniform traffic, a distance-based selection suffers from an imbalanced traffic load on VLs. In this selection scenario in which one VL is faulty (the magenta VL), 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. In this case, all the traffic of the faulty magenta VL is misrouted to the green VL, imposing unbalanced traffic on VLs as half of the whole load is now on the green VL. Such an unbalanced utilization can significantly increase the chance of congestion and degrade 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. Therefore, when dealing with a VL fault, it is important to assign the load to healthy VLs while considering the distance to the boundary router and ensuring well-distributed traffic across all the VLs. *ReD* considers both the VL faults and traffic profile to enable a fault-tolerant and congestion-aware VL selection, as discussed in the next section.

In addition to VL faults, HL faults in 2.5D networks can arise from various factors including physical defects [25], aging [32], and electromigration [25]. These HL faults can introduce reliability concerns or even lead to a total loss of connectivity unless a fault tolerance mechanism is defined in the network. In the context of *ReD*, we specifically address HL faults and develop a fault-tolerant routing algorithm to handle both VL and HL faults. It is important to note that since router faults can be represented as certain link faults, our focus in *ReD* is solely on addressing link faults. To provide further clarification, a faulty router is regarded as one in which all of its links are deemed faulty.

2) ReD's Vertical-Link Selection: As discussed earlier, for inter-chiplet packets, two temporary destinations are selected by the VL-selection process. Based on a selection algorithm, a VL is selected and the packet is forwarded to the VL to route the packet towards its destination. To tolerate VL faults and improve congestion on boundaries of chiplets, during inter-chiplet routing, *ReD* supports an adaptive VL-selection approach where VLs are adaptively selected as intermediate

destinations, as discussed below.

The VL-selection strategy in *ReD* includes a design-time (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 (i.e., 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 [30]. 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 objectives, 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 *ReD* with an adaptive selection under different VL-fault scenarios. Here, the look-up table size and hardware cost are 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 influenced by packets from the same source-chiplet routers, whereas in the second VL selection, packets destined for the destination-chiplet routers play a role in the selection process. As both VL selections are performed in a similar manner, here we only discuss VL selection on the source chiplet for brevity (i.e,. the first VL selection).

Given a fault scenario and based on the VL selection on the source chiplet, i.e., the first VL 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 VLs (v): $VL_C = \{v_1, v_2, .., v_V\}$ and V is the number of fault-free VLs. 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. In the following, we discuss how *ReD* finds the optimized VL selections (i.e., s^*) during design-time.

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

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

where T_r^{inter} is the inter-chiplet traffic rate of router r. Also, $U_v^r = 1$ ($U_v^r = 0$) 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 VL v as:

Table I
ABBREVIATIONS USED IN THIS PAPER

Abbreviation	Description
T_r^{inter}	Inter-chiplet traffic rate of router r
U_v^r	Whether router r uses (does not use) VL v
l_{avg}	Average load over all VLs
l_v	Load of VL v
L_v	Load cost of VL v
D_v^r	Manhattan distance of router r and VL v
D_v	Distance cost of VL v to the routers that select v
C_s	Overall cost of a selection set s
ρ	Importance of load balancing versus distance objectives
S	Set of all possible selection sets
s^*	Optimal selection set with minimum cost C_s^*

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

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

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

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 *ReD*. 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 approach is summarized in Algorithm 2.

Algorithm 2 VL Selection in *ReD*

```
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)
```

The algorithm iterates through selection sets and vertical links, finding the overall costs. Therefore, time complexity is $O(N_s.V)$, where N_s is the number of selection sets and V V is the total number of VLs in the chiplet system. The space complexity is negligible since each time one set is evaluated. Executing the algorithm offline ensures time complexity does not affect online performance. The algorithm is executed for different VL-fault scenarios at design-time and the selection

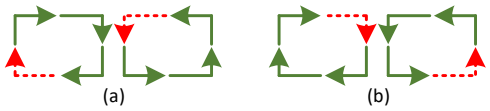

Figure 4. Deadlock-free turn models used for intra-chiplet and -interposer routing in (a) the first VN and (b) the second VN.

sets (s^*) are saved in routers for run-time use. For the baseline system (shown in Fig. 1), where each chiplet has four VLs, there are 14 combinations of faults—$\binom{4}{1} + \binom{4}{2} + \binom{4}{3}$—and hence 14 VL addresses are saved in each router.In chiplet system scaling, while chiplet and VL numbers may increase, VLs per chiplet typically remain low to minimize fabrication costs. If a large number of VLs per chiplet are still desired, two options are suggested to improve the look-up table size: 1) Use the same solutions for each set of fault scenarios (approximation) at the cost of performance, and 2) employ online search at the cost of increased complexity.

When there is a change related to the VL fault status (i.e., a new VL fault occurs or a faulty VL becomes healthy), the *ReD* selection mechanism dynamically adapts the VL-selection process to ensure continued fault-tolerance and optimized traffic routing. During run-time, upon detecting a fault, *ReD* leverages the pre-analyzed VL selections stored in the look-up tables within the routers, which were obtained from the design-time (offline) analysis considering various VL-fault scenarios, VL-utilization balancing, and distance minimization. This adaptive approach allows *ReD* to efficiently reroute affected packets by selecting the appropriate VLs from the pre-analyzed options, effectively avoiding VL faults and preventing potential congestion. The adaptive VL-selection process enhances the overall system reliability, reduces latency, and contributes to reduced stress-migration-based faults, thereby improving the 2.5D network's robustness and ensuring efficient packet delivery across chiplets.

C. Fault-Tolerant Routing to Handle Horizontal Link Faults

As discussed in Section III-B1, it is crucial to consider HL faults in interposer networks. These faults not only affect the sharing of VL updates, which will be discussed in the next section, but also impact the intra-chiplet and -interposer routing of network packets. One popular method for tolerating HL faults is to employ adaptive routing and divert traffic away from the faulty links. This adaptation in routing often requires the use of extra VCs (i.e., defining new VNs). Fortunately, *ReD* utilizes VNs already to ensure deadlock freedom (see Section III-A), eliminating the need to add any extra VCs for fault tolerance. Our proposed VN-separation technique, as supported by Theorems III.1 and III.2, provides the flexibility to effectively utilize VNs for the majority of packets in intra-chiplet and -interposer routing. We also show this flexibility in Algorithm 1 (Lines 3 and 8), where packets can be routed in both VNs. We use this opportunity here to equip *ReD* with intra-chiplet and -interposer adaptive routing that enables the ability to tolerate HL faults.

We employ two different turn model routing algorithms, as depicted in Fig. 4, for routing in the first and second VNs of the intra-chiplet and -interposer networks. Solid black lines represent allowed turns, while dashed red lines represent avoided turns. The deadlock-freedom of these turn models will

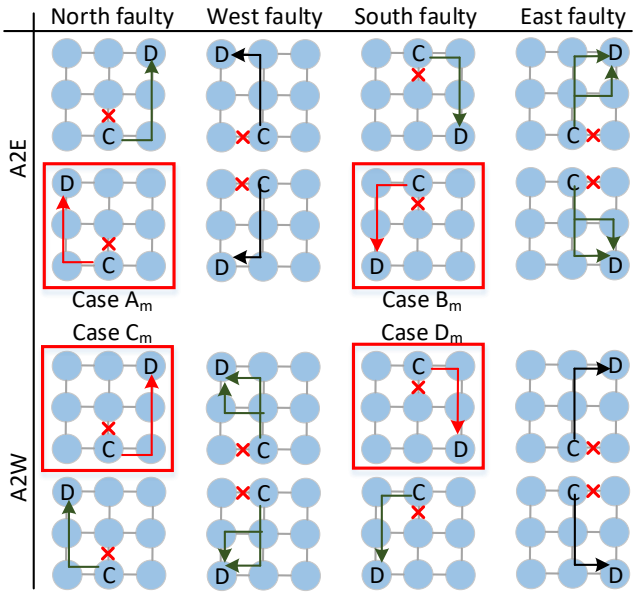

Figure 5. Minimal routing to bypass HL faults (A2E: adaptive-to-east VN, A2W: adaptive-to-west VN).

be discussed in Section III-E. The turn model of the first VN enables adaptive routing for packets from West to East, while the turn model of the second VN allows adaptive routing for packets from East to West. We select these turn models to provide adaptive routing in the two East and West directions so that we can combine them and make a fully minimal adaptive routing. A fully minimal adaptive routing is defined as the routing in which all possible paths, each having a length equal to the Manhattan distance, can be taken. Moreover, we choose these turn models to support YX routing in both VNs, regardless of routing to the East or West. Therefore, for the small portion of packets that cannot be routed in both VNs, YX routing can be used as the default routing. To elaborate further, when such a packet is assigned to the first VN, it employs YX routing for routing to the West and minimal adaptive routing for routing to the East, whereas for the second VN, YX routing is utilized for routing to the East, and minimal adaptive routing is used for routing to the West.

When facing HL faults, there are two primary scenarios to consider: 1) a minimal routing approach to bypass the fault, and 2) a non-minimal routing strategy to get around the fault. We illustrate how to address these two scenarios in Figs. 5 and 6, respectively. These primary scenarios are discussed in more detail below.

1) Minimal routing to bypass HL faults: When a packet faces an HL fault and, according to the implemented routing, there is an alternative route available to bypass the fault without the need for taking a non-minimal path, a minimal fault-tolerant routing can be achieved. All instances of this scenario are presented in Fig. 5. In the first VN, which is adaptive-to-East (A2E), not only minimal adaptive routing is achieved toward the East direction but also YX routing is allowed toward the West direction. Therefore, when a packet is routed in the first VN, the well-designed turn model enables *ReD* to tolerate faults on both the East and West links of the router, as demonstrated in columns 2 and 4 of Fig. 5. However, packets in the first VN are unable to bypass a fault

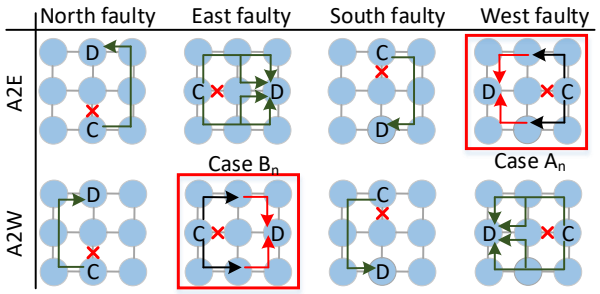

Figure 6. Non-minimal routing to get around HL faults (A2E: adaptive-to-east VN, A2W: adaptive-to-west VN).

on a North or South link if the destination is positioned on the West side of the source, denoted as cases A_m and B_m. Similarly, packets in the second VN cannot bypass a faulty North or South link if the destination is situated on the East side of the source, labeled as cases C_m and D_m. Therefore, to tolerate HL faults, instead of Round-Robin VN allocation shown in Algorithm 1 (Lines 3 and 8), here VNs are allocated to packets in a manner that minimizes cases A_m to D_m and, consequently, maximizes fault tolerance. To achieve this, the first VN is assigned to inter-chiplet packets if the destination is situated on the East side of the source, while the second VN is assigned when the destination is on the West side of the source. Additionally, when packets enter the interposer, if the destination boundary router is located on the West side of the source boundary router, they will be switched to the second VN. Conversely, if the destination boundary router is on the East side of the source boundary router, the packets will remain in the first VN.

2) Non-minimal routing to get around HL faults: To get around an HL fault, a non-minimal routing is required, when the current and destination routers are located on the same row or the same column. All possible cases are shown in Fig. 6. If a fault occurs on the North or South ports, it can be tolerated unless it is located on the East-most column in the first VN or the West-most column in the second VN. To maximize fault tolerance, packets in the first VN that are located on the East-most column are switched to the second VN when they encounter such a fault. Conversely, we prevent the situation where a packet in the second VN faces a fault on the West-most column. To avoid this, packets with destinations located on the West-most column are routed using YX routing if the packet is routed in the second VN. There are two cases in Fig. 6, cases A_n and B_n, in which the fault cannot be tolerated. Similar to cases A_m and D_m, cases A_n and B_n are minimized as VNs are allocated based on packet destinations, with inter-chiplet packets assigned to the first or second VN, and channel switching in the interposer is determined by the relative positions of source and destination boundary routers.

D. Sharing Fault Updates

To achieve fault-tolerant and traffic-aware VL selection, *ReD* relies on identifying faulty links. In our approach, updates related to HL faults are not shared; only a router is aware of the faults on its local HLs. This strategy minimizes the overhead of sharing HL fault updates, enabling packets to be detoured efficiently to avoid faulty HLs based on the rules in Figs. 5 and 6. On the other hand, *ReD* shares VL fault updates to proactively route packets to healthy VLs and avoid faulty VLs. We prioritize sharing VL fault updates because the number of VLs is much smaller than HLs, and not all routers have a VL locally. Additionally, as demonstrated in Section III-B2, VL faults significantly impact network traffic, making it crucial to distribute the load proactively among healthy VLs.

To facilitate sharing VL fault updates, *ReD* introduces a Hamiltonian-based routing technique, enabling the sharing of faulty/healthy status of VLs with all local routers within a chiplet or interposer. This approach enhances fault awareness and enables efficient traffic management to maintain fault tolerance and improve network performance. Fig. 7 shows the proposed Hamiltonian-based approach for sharing the updates. In Fig. 7(a), we demonstrate the Hamiltonian-based routing scheme. Each router is assigned an ID from West to East, with increasing IDs from North to South in even columns and from South to North in odd columns. We employed the Hamiltonian-based routing scheme to distribute the updates to higher and lower ID numbers in the second and first VNs, respectively. A Hamiltonian path is defined as a path that visits all routers exactly once. Therefore, this routing strategy ensures an efficient flow of the updates within the chiplet/interposer while leveraging the advantages of the Hamiltonian path. As is shown in Fig. 7(a), the proposed Hamiltonian-based routing guarantees that all the routers in the chiplet/interposer receive the updates of VLs, unless there is a faulty HL on the Hamiltonian path of the chiplet/interposer. To send updates, the local router generates two 1-flit packets and sends them through the increasing path (H_I) and the decreasing path (H_D). This approach allows us to update all routers in the Hamiltonian path while utilizing the existing VNs without the need for additional ones.

We designed our Hamiltonian-based routing to be compatible with our main routing without using any extra VCs. It is evident that the allowed turns shown in Fig. 4 support our Hamiltonian-based routing presented in Fig. 7 when routing H_I in the first VN and H_D in the second VN. Thus, our Hamiltonian-based routing effectively sends the update packets to all routers and updates the healthy/faulty status of VLs without creating any deadlock, unless there is a faulty HL on the Hamiltonian path.

As discussed in Section III-B1, it is important to consider the potential faults in HLs, in addition to VLs. To ensure robustness against faults in sharing the updates, we employ two distinct Hamiltonian paths with different directions for propagating the updates. While alternative Hamiltonian paths could be considered, we opt for these specific paths to align with our turn models (see Fig. 4), ensuring seamless compatibility and obviating the need for additional VCs to accommodate this Hamiltonian routing. To illustrate this approach, we present two examples in Figs. 7(b) and 7(c). In these figures, we introduce another Hamiltonian path and assign different IDs to the routers (shown in green). In the even columns, the router IDs increase from North to South, while in the odd columns, the router IDs increase from South to North. Moreover, in each Hamiltonian path, the packet takes a detour by routing in the direction of a non-faulty link within the Hamiltonian-compatible route. For instance, from the yellow router (router 11 on the black path), when routing through the

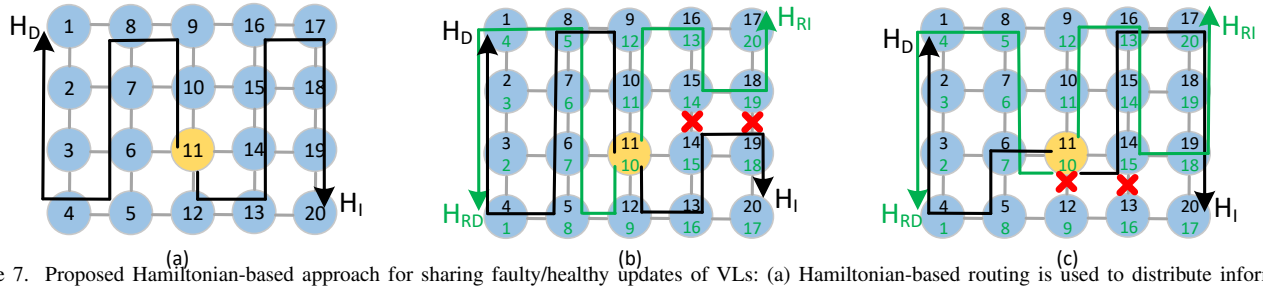

Figure 7. Proposed Hamiltonian-based approach for sharing faulty/healthy updates of VLs: (a) Hamiltonian-based routing is used to distribute information through increasing and decreasing IDs, routed in the first and second VNs respectively, (b) and (c) two Hamiltonian-based routings (green and black) are used for fault tolerance when sharing information, specifically designed to handle HL faults.

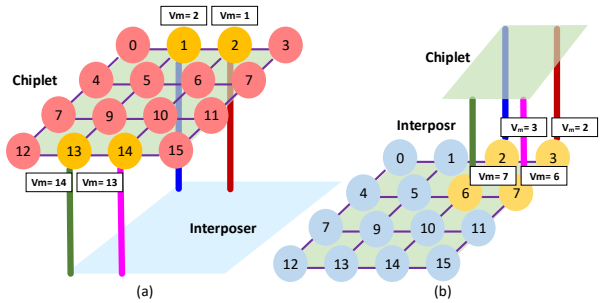

Figure 8. Spare VLs used for packets which failed to receive VL updates and are routed to a faulty VL: (a) Spare VL on chiplets and (b) Spare VL on the interposer.

H_I path, if the South port is faulty, the packet is routed to the East port to get router 14 with a larger ID number instead of 10. This means that when routing through the increasing path (H_I), the route going to a larger ID is taken, while when routing through the decreasing path (H_D), the route to a smaller ID is taken. Therefore, with two Hamiltonian-based paths and the simple fault-tolerant detouring, most of the HL faults can be tolerated, when sharing VL updates. Note that in the last subsection (Section III-C), we elaborated on our methodology for handling HL faults when routing network packets, extending beyond updates alone.

While the proposed Hamiltonian-based routing effectively tolerates the majority of HL faults, as demonstrated in section IV, rare instances may arise where certain routers fail to receive updates. This is more prevalent in scenarios of exceptionally high fault rates or when the faults are region-specific. For instance, in Fig. 7(c), two routers do not receive updates. To address this, we implement a mechanism wherein routers equipped with VL routers (i.e, the routers connected to VLs) store the address of an additional spare VL. Consequently, in cases where update reception fails and a faulty VL is encountered, packets can be rerouted to the alternate VL. Importantly, the choice of the spare VL must align with our turn models, as illustrated in Fig. 4, to ensure compatibility with the misrouting scenario. In our case study, we establish the spare VL for chiplets and the interposer, as depicted in Fig. 8. It is worth noting that for interposer-bound packets, we consider switching or maintaining the second VN after any misrouting to a spare VL. Therefore, the misrouting on chiplets must conform to the turn model of the first VN in Fig. 4, while those on the interposer adhere to the turn model of the second VN in Fig. 4.

We summarized our Hamiltonian-based technique in Algorithm 3. For brevity, we show only the primary Hamiltonian paths in the algorithm. Each router only needs to be aware of

Algorithm 3 Hamiltonian-based routing in *ReD*

P_{H_I}: Port to next router on increasing Hamiltonian
P_{H_D}: Port to next router on decreasing Hamiltonian
P_{SH_I}: Port to next router on spare increasing Hamiltonian
P_{SH_D}: Port to next router on spare decreasing Hamiltonian
Input: Fault sharing flit (F-flit)
Output: Out port
if Incoming flit is F-flit **then**
  Update fault table of current router based on F-flit
  if F-flit is increasing Hamiltonian **then**
    if P_{H_I} is not faulty **then**
      Out port $\leftarrow P_{H_I}$
    else
      Out port $\leftarrow P_{SH_I}$
  else if F-flit is decreasing Hamiltonian **then**
    if P_{H_D} is not faulty **then**
      Out port $\leftarrow P_{H_D}$
    else
      Out port $\leftarrow P_{SH_D}$

the ports required to reach the next routers on the Hamiltonian paths. Please note that in the algorithm, the term ”Port,” refers to the port’s address. Our Hamiltonian-based routing does not require any additional ports for sharing updates. Moreover, the terms ”spare increasing Hamiltonian” and ”spare decreasing Hamiltonian” denote the additional port addresses strategically employed along the increasing and decreasing Hamiltonian paths, respectively. These spare port addresses contribute to enhancing fault-tolerance capabilities.The algorithm, with $O(1)$ time complexity and minimal space usage, remains unaffected by system size, except for the VL selection table storage, which we previously discussed in section III-B2.

E. Deadlock- and Livelock-Freedom

1) Deadlock-freedom: ReD 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. Considering these reasons, *ReD* is deadlock-free, if no cycle occurs in chiplets and the interposer locally.

In *ReD* routing, both chiplets and the interposer are locally deadlock-free. This is achieved by utilizing two turn models for intra-chiplet and -interposer routing, both of which are known to be deadlock-free. The first turn model (see Fig. 4(a)) is similar to a North-last turn model, rotated counterclockwise

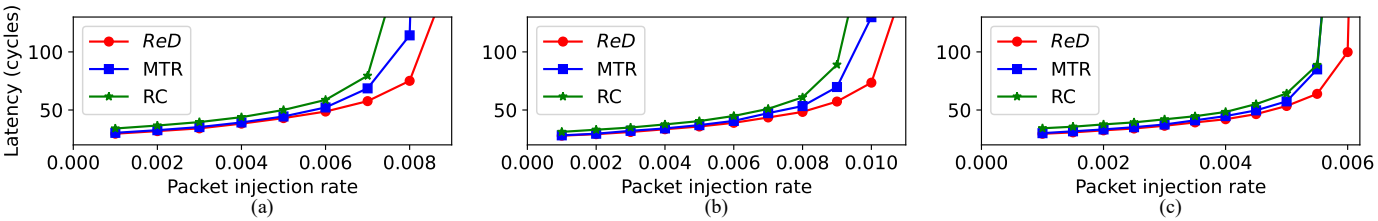

Figure 9. Average latency comparison among *ReD*, MTR, and RC routing algorithms when applied to 2.5D networks with four-chiplet system. The comparison is conducted under different synthetic traffic patterns: (a) Uniform, (b) Localized, and (c) Hotspot.

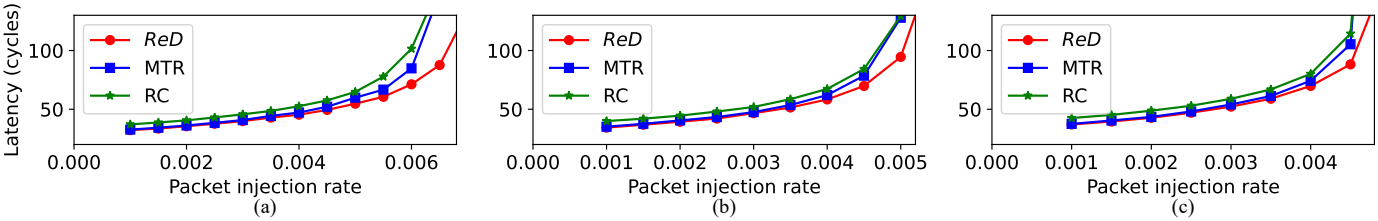

Figure 10. Average latency comparison under different system sizes and Uniform traffic. (a) 6 chiplets, (b) 8 chiplets, and (c) 12 chiplets.

Table II
SIMULATION SETUP.

Number of chiplets	4,6,8, and 12	VL per chiplet	4
Packet size	8 flits	Flit width	32 bits
Buffer size	4 flits	Number of VCs	2
Router frequency	2 Ghz	VL/HL latency	1 cycle
Simulation time	1 million cycles	Warmup time	10K cycles

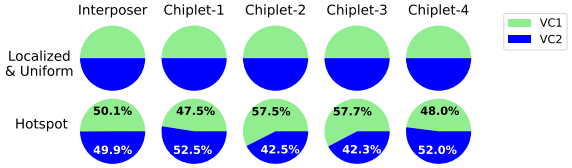

Figure 11. VC utilization of *ReD* under Uniform, Localized, and Hotspot traffic patterns.

by 90 degrees. Similarly, the second turn model (see Fig. 4(b)) is also like a North-last turn model but rotated clockwise by 90 degrees. The deadlock-freedom of the North-last turn model has been proven in the context of a 2D mesh [18]. In the North-last turn model, North-East and North-West turns are avoided. As the North-last turn model is deadlock-free, the rotated turn models employed in the first and second VCs of *ReD* routing are also deadlock-free.

2) Livelock-freedom: *ReD* is livelock-free in any 2.5D chiplet system, given that both chiplets and the interposer exhibit local livelock-freedom. The inter-chiplet packets in *ReD* follow a specific routing path: source chiplet → interposer → destination chiplet. This routing is facilitated by two intermediate destinations (see Section II-A). Since packets do not route using any intermediate chiplet, packets in *ReD* are routed within a finite number of hops, ensuring that livelocks do not occur, provided that chiplets and the interposer maintain local livelock-freedom. *ReD* is also livelock-free in intra-chiplet and -interposer routing. It is important to note that the minimal routing approach is followed for most cases in *ReD*, ensuring livelock freedom in such cases. However, in instances where packets encounter HL faults and the destination is in the same row or column, non-minimal routing may be employed, as illustrated in Fig. 6. As illustrated in this figure, packets do not use backward routing, and after misrouting, the rest of routing is done minimally. Therefore, the packets are routed within a finite number of hops and livelock cannot occur.

IV. EVALUATION AND SIMULATION RESULTS

A. Simulation Environment and Configuration

We employed the Noxim simulator [33] for our network simulations, GEM5 [34] as our system simulator, and Cadence Genus for estimating the area and power consumption of the 2.5D networks considered. We enhanced the Noxim simulator, which is a cycle-accurate simulator for NoCs [33], to support 2.5D chiplet systems, and the system depicted in Fig. 1. In this configuration, each chiplet is connected to the interposer utilizing four bidirectional VLs. The decision to connect each chiplet with four VLs instead of a fully connected setup was motivated by the goal of reducing fabrication costs [10]. To further reduce VLs and enhance fabrication-cost optimization, serialization techniques can be employed [35]. According to [10], for a 4×4 chiplet system, placing the four VLs on the top and the bottom rows of the chiplet is an optimal choice, as shown in Fig. 1, when considering hardware complexity and network latency. It is important to note that the efficiency of *ReD* is not impacted by the the placement and density of VLs, as it does not rely on turn models to achieve inter-chiplet deadlock-freedom. In addition to the baseline four-chiplet system, we conducted simulations on six-chiplet, eight-chiplet, and twelve-chiplet systems to examine how the efficiency of *ReD* scales with system size.

We compared the performance of *ReD* against state-of-the-art routing algorithms for chiplet-based systems, MTR [10] and RC [13]. While MTR and RC do not impose any specific requirements on the number of VCs, we utilized two VCs in all algorithms to ensure a fair comparison with *ReD* (considering a single VC would result in inferior performance for MTR and RC). More details of our simulation setup is included in table II. For offline VL-selection optimization, we focused on Uniform traffic as the most pessimistic assumption, although our simulations encompassed different traffic scenarios. Taking traffic information into account during the offline optimization process would yield further performance improvements.

B. Latency Analysis

The analysis of latency for synthetic traffic is depicted in Fig. 9. We employed three synthetic traffic patterns: Uniform, Localized, and Hotspot. In the Uniform traffic pattern, packets

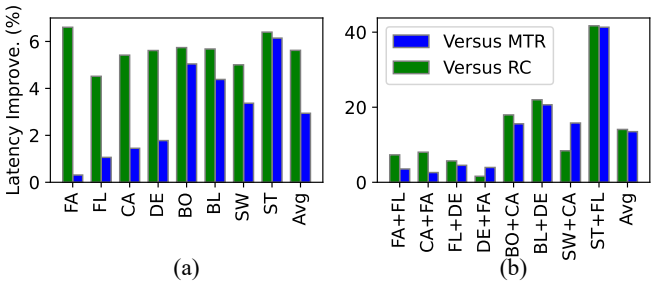

Figure 12. Latency improvement under real-application traffic using (a) a single application, and (b) two applications simultaneously.

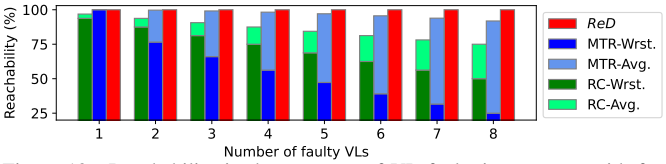

Figure 13. Reachability in the presence of VL faults in a system with four chiplets. Note that *ReD*-Wrst. and *ReD*-Avg. are the same (both shown by *ReD*).

are randomly generated between all feasible source-destination pairs. This modeling simulates a scenario where data transmission is evenly distributed across the network, representing a balanced and unbiased traffic flow. The Hotspot traffic pattern involves a subset of nodes that receive a significantly higher volume of traffic than others. This setup facilitates the assessment of congestion handling. For the Localized traffic pattern, we assume uniform communication within chiplets, with nodes on the same chiplet exhibiting higher communication compared to nodes on different chiplets. This configuration emphasizes intra-chiplet communication over inter-chiplet communication. As shown in Fig. 9, *ReD* exhibits the lowest average latency in all traffic patterns compared to MTR and RC due to its balanced VL selection and VC utilization strategies. In the Localized traffic scenario, 40% of the packets are categorized as intra-chiplet packets, where both the source and destination are located on the same chiplet. As shown in Fig. 9(b), *ReD* demonstrates low latency in this scenario by effectively distributing the VC utilization for intra-chiplet packets, as supported by Theorem III.1. For Hotspot traffic, presented in Fig. 9(c), *ReD* shows a slightly lower improvement due to the restriction of incoming packets to use only the second VC, leading to back-pressure. However, it is worth noting that our simulations employed a relatively high rate of hotspots (3 hotspot points with a 10% rate each).

In Fig. 10, we analyze the performance of six-chiplet, eight-chiplet, and twelve-chiplet systems, and it is noteworthy that *ReD* continues to exhibit high performance compared to the other approaches. The consistent performance gains observed across different system sizes highlight *ReD*'s scalability and suitability for large-scale chiplet-based architectures. *ReD* demonstrates performance improvement with larger systems, even with a 12-chiplet system (192 cores). However, it exhibits slightly less improvement compared to MTR and RC as system size increases. To understand the reason for this, we conducted an analysis of the load distribution; however, due to space constraints, we have chosen not to include it for brevity. Nevertheless, it is important to note that in the comparison of load distributions across different system sizes, particularly in the 12-chiplet system, congestion becomes more pronounced on the interposer rather than the VLs. This shift can be attributed to the larger size of the interposer in the 12-chiplet system, leading to congestion at its center when managing communication among the twelve chiplets. This results in a slightly lesser improvement in average latency in larger systems for *ReD* compared to MTR and RC.

The distribution of VC utilization for the synthetic traffic patterns is presented in Fig. 11. In Uniform and Localized traffic, the VC utilization is balanced with a deviation of less than 0.4%, which is illustrated in the same chart. In Hotspot traffic, despite the relatively high rate of hotspots, the deviation in VC utilization remains below 8%. The balanced VC utilization achieved by *ReD* contributes to its low average latency while effectively avoiding deadlock situations.

To incorporate real-application traffic, we generated traffic using GEM5 [36] from PARSEC benchmarks [34] and simulated it using our chiplet-based Noxim simulator. The simulations were conducted 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). Fig. 12 illustrates the latency improvement achieved in eight PARSEC applications (blackscholes, bodytrack, canneal, dedup, facesim, fluidanimate, streamcluster, and swaptions), where the first two letters of each application are represented on the x-axis. In order to assess network performance under higher congestion, we also simulated the scenario of two applications running simultaneously, with each application utilizing 32 x86 cores and two shared L2 cache banks (as shown in Fig. 12(b)).

On average, *ReD* exhibits greater improvement when multiple applications are considered, mainly due to the increased likelihood of network congestion in such cases, which *ReD* is able to address more efficiently, compared to MTR and RC. In Fig. 12(b), the two-application combinations are arranged based on the traffic load, ranging from low (FA+FL) to high (ST+FL). Notably, for high traffic loads, *ReD* demonstrates a significant improvement of up to 40% compared to both MTR and RC.

C. Fault-Tolerance Analysis

1) VL faults: In order to evaluate the fault tolerance of *ReD*, we analyze network reachability in the presence of faults, following a similar approach as in the study by [28]. In fault scenarios, reachability is defined as the proportion of successfully routed packets to the total number of injected packets. Reachability in the presence of VL faults is shown in Fig. 13(a). We conducted fault injections for all possible combinations of fault patterns, excluding those that result in complete disconnection of chiplets (i.e., when all VLs of a chiplet are faulty). The figure presents both average and worst-case reachability for injected fault rates ranging from 3.125% to 25%, corresponding to 1 to 8 faulty VLs. In the average case, *ReD* enhances network reachability by 3.1–25% compared to RC and by 0–8.1% compared to MTR. In the worst case, *ReD* improves network reachability by 6.25–50% compared to RC and by 0–75% compared to MTR. Fig. 14 shows network reachability as the system scales up, comparing

4-chiplet and 12-chiplet systems. In both system sizes, *ReD* achieves 100% reachability for the given fault injection rates, while MTR and RC fail to achieve 100% reachability even at the lowest fault injection rate. The restricted turns in MTR and the permission network of RC limit their VL selection and, consequently, their fault tolerance against faulty VLs.

Fig. 15(a) presents the effect of VL faults on latency. The analysis excludes MTR and RC due to their inability to ensure complete reachability in fault scenarios. Two fault injection rates are considered where four faults correspond to 12.5% and eight faults correspond to 25% of VL links being faulty. Furthermore, in addition to utilizing *ReD* with our proposed VL selection (*ReD* in Fig. 15(a)), we incorporate the average latency of *ReD* with distance-based VL selection strategies (*ReD*-Dis.) as commonly employed in 3D networks [28]. *ReD*-Dis. refers to routing using *ReD* while selecting VLs based on distance instead of our proposed method. Notably, *ReD* demonstrates significantly lower latency compared to distance-based VL selections at fault injection rates of 12.5% and 25%, respectively. As can be seen, in the higher fault injection rate, *ReD* shows more improvement. This confirms that *ReD* is able to handle the higher congestion due the smaller number of healthy VLs.

2) HL faults: In Figs. 15(b) and 15(c), we show the effect of HL faults on network latency, when the faults are on the chiplet and interposer, respectively. We also demonstrate the impact of combined faults in Figs. 15(d). In this scenario, we examined uniform combinations of VL faults, chiplet HL faults, and interposer HL faults. HL faults have a more pronounced impact on the interposer, as shown in Figs. 15(c), primarily due to the heightened traffic congestion it experiences. This congestion is further exacerbated by the interposer's role in managing global traffic flow among multiple chiplets. In the fault-free case, we show two versions of *ReD*: ReD_{XY} and $ReD_{AdaptiveXY}$. In ReD_{XY}, XY routing is used for the inter-chiplet/interposer routing, while $ReD_{AdaptiveXY}$ employs our adaptive routing, which is discussed in Section III-D, to tolerate HL faults. Our $ReD_{AdaptiveXY}$ imposes small overhead on latency under Uniform traffic because it separates traffic into VCs based on source and destination location. This separation can make the VC utilization slightly imbalanced. However, the VC separation enables adaptive routing which not only provides us with the opportunity to handle the HL faults but also gives the routing the adaptation to avoid traffic congestion. Therefore, in the fault-free case, $ReD_{AdaptiveXY}$ offers high performance when considering non-uniform traffic (e.g., Hotspot). As we discussed in Section III-D, HL faults not only can negatively affect the routing process of the NoC packets, but also they prevent the sharing of faulty/healthy status of VLs. Therefore, as we discussed, our proposed redundant Hamiltonian-based routing, which shares the updates in a different direction, significantly improves fault tolerance against the HL faults when sharing the updates.

Fig. 16 illustrates the average percentage of updated routers when sharing the status of faulty/healthy VLs for chiplets and the interposer. The analysis focuses on the effectiveness of updating the routers in the network when informing them about the status of VLs in terms of their faultiness or healthiness. Please note that an update is triggered, starting from the router connected to the VL, in an event-driven scheme: when a healthy VL becomes faulty or when a faulty VL becomes healthy. As detailed in Section III-D, it is evident that HL faults can impede the sharing mechanism. Therefore, this analysis focuses on fault tolerance against HL faults when sharing VL fault updates. The percentages reflect the proportion of routers that receive updated information regarding the VLs. In the figure, "Hamiltonian" represents the simple Hamiltonian-based routing with one path in the increasing Hamiltonian path and one path in the decreasing Hamiltonian path. "Hamiltonian$_F$" represents Hamiltonian routing that incorporates the selection of a spare path in the case of HL faults. Moreover, "Hamiltonian$_{2F}$" represents our proposed Hamiltonian routing with two Hamiltonian routes sharing the updates in two different direction, as shown in Fig. 7. As can be seen, "Hamiltonian$_{2F}$" is able to offer significantly high fault tolerance under HL-fault scenarios. Even with eight faulty HLs, which is a high fault rate, more than 80% of the routers are updated. Therefore, the proposed Hamiltonian-based routing ("Hamiltonian$_{2F}$") is efficient in sharing the faulty/healthy status of VLs, although it does not need extra VCs or any control network to share the updates. It is worth noting that we used "Hamiltonian$_{2F}$" across all the other experiments detailed in this paper. Regular packets are delayed to the next cycle when F-flits are being shared, but since F-flits occur less frequently, the impact on regular packet routing is negligible.

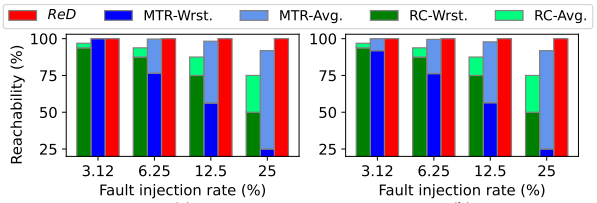

Figure 14. Reachability in the presence of VL faults in a system with (a) four chiplets (total VLs=32), and (b) twelve chiplets (total VLs=96). Note that *ReD*-Wrst. and *ReD*-Avg. are the same (both shown by *ReD*).

Table III
AREA AND POWER ANALYSIS OF *ReD*, MTR, AND RC

	MTR	$RC_{non\text{-}bndry}$	RC_{bndry}	*ReD*
Router area (μm^2)	45878	46663	51984	46730
Norm. router area	1	1.017	1.133	1.019
Router power (mW)	11.644	11.76	12.841	11.802
Norm. router power	1	1.009	1.102	1.014

D. Hardware and Power Analysis

We utilized Cadence Genus and ORION 3.0 [37] for estimating the area and power of the router using the 45-nm technology. Table III presents a comparison of the area and power estimates for a six-port *ReD* router compared to six-port MTR and RC routers. It should be noted that the hardware implementations of non-boundary and boundary routers differ in RC, so we provide separate values in the table. The results in Table III demonstrate that *ReD* incurs a hardware overhead of less than 2% and a power overhead of less than 1%, relatively, when compared to related existing work. This minimal overhead includes all the techniques and

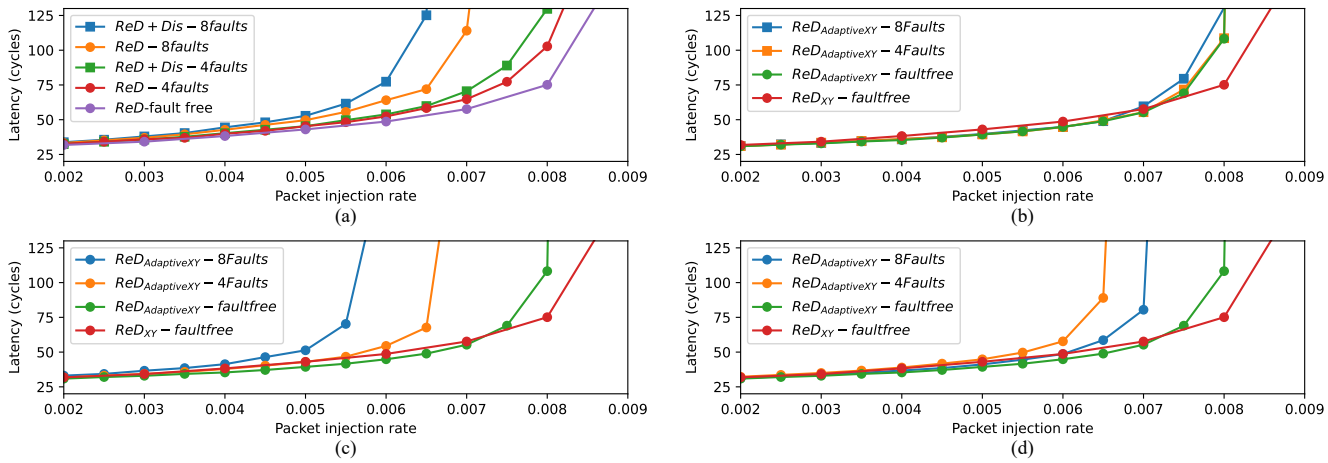

Figure 15. Average latency in *ReD* with different VL-selection strategies, fault-tolerant approaches, and fault-injection rates for a four-chiplet system. (a) VL faults, (b) HL faults on chiplets, (c) HL faults on the interposer, and (d) uniformly combined faults on VLs, HL faults on chiplets, and HL faults on the interposer.

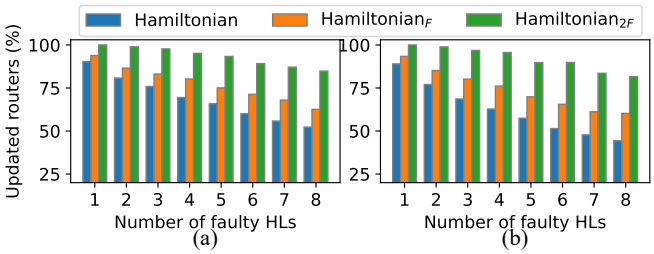

Figure 16. Average percentage of updated routers when sharing the status of faulty/healthy VLs for (a) chiplets and (b) the interposer.

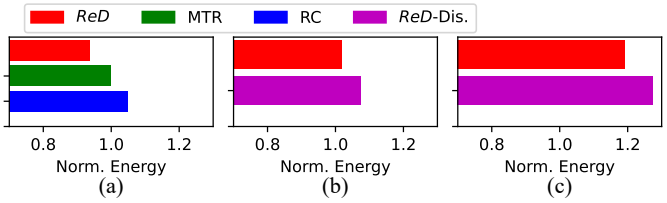

Figure 17. Normalized energy comparison for a four-chiplet network with (a) no faults, (b) 12.5% faulty VLs, and (c) 25% faulty VLs.

functionalities used in *ReD*: the main fault-tolerant routing, the logic for the VN-assignment algorithm, the logic for sharing VL updates, and look-up tables used for fault-tolerant VL selection.

E. Energy Analysis

Fig. 17(a) shows energy analysis normalized to MTR for 100K packets injected under Uniform traffic and 0.01 packets/node/cycle injection rate. *ReD* achieves lower energy by 6.1% and 10.7% compared to, respectively, MTR and RC as it can route the injected packets faster. In addition, RC uses extra buffers for deadlock-freedom, which increases energy costs. MTR sends packets to non-minimal VLs to guarantee deadlock-freedom, while *ReD* is free to select any VLs. Also, under 12.5% and 25% VL-fault scenarios, shown in Figs. 17(b) and 17(c), compared to distance-based (*ReD*-Dis.) VL selection, *ReD* with its congestion-aware VL selection is able to achieve lower energy.

V. Conclusion

This paper introduced a fault-tolerant routing algorithm, named *ReD*, specifically designed for 2.5D chiplet networks. The main objective of *ReD* is to provide a deadlock-free and fault-tolerant routing solution, while minimizing traffic congestion and area overhead. *ReD* achieves VC-based deadlock-freedom by allowing packets to freely choose any VL and efficiently balances VC utilization. This flexibility in VL-selection enables *ReD* to tolerate various patterns of VL faults. To address network congestion in the presence of VL faults, *ReD* incorporates a dynamic traffic-aware VL selection strategy to enhance runtime routing efficiency. Simulation results comparing *ReD* to state-of-the-art routing algorithms demonstrate significant improvements. *ReD* enhances network reachability by up to 75% when operating with a fault rate of up to 25%. Additionally, it reduces network latency by up to 40% in multi-application execution scenarios, all while incurring less than 2% area overhead. These findings underscore the potential of *ReD* in advancing the performance of emerging 2.5D chiplet systems.

Acknowledgment

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

References

[1] Y. S. Shao, J. Clemons, R. Venkatesan, B. Zimmer, M. Fojtik, N. Jiang, B. Keller, A. Klinefelter, N. Pinckney, P. Raina *et al.*, "Simba: Scaling deep-learning inference with multi-chip-module-based architecture," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019, pp. 14–27.

[2] A. Kannan, N. E. Jerger, and G. H. Loh, "Enabling interposer-based disintegration of multi-core processors," in *Proceedings of the 48th international symposium on Microarchitecture*, 2015, pp. 546–558.

[3] E. Taheri, M. Isakov, A. Patooghy, and M. A. Kinsy, "Addressing a new class of reliability threats in 3-D network-on-chips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 7, pp. 1358–1371, 2019.

[4] J. Kim, G. Murali, H. Park, E. Qin, H. Kwon, V. Chaitanya, K. Chekuri, N. Dasari, A. Singh, M. Lee *et al.*, "Architecture, chip, and package co-design flow for 2.5D IC design enabling heterogeneous IP reuse," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.

[5] G. H. Loh, S. Naffziger, and K. Lepak, "Understanding chiplets today to anticipate future integration opportunities and limits," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021, pp. 142–145.

[6] Y. Ma, L. Delshadtehrani, C. Demirkiran, J. L. Abellan, and A. Joshi, "TAP-2.5D: A thermally-aware chiplet placement methodology for 2.5D systems," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021, pp. 1246–1251.

[7] F. Sunny, E. Taheri, M. Nikdast, and S. Pasricha, "Machine learning accelerators in 2.5 d chiplet platforms with silicon photonics," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2023, pp. 1–6.

[8] E. Taheri, M. A. Mahdian, S. Pasricha, and M. Nikdast, "Trine: A tree-based silicon photonic interposer network for energy-efficient 2.5 d machine learning acceleration," in *Proceedings of the 16th International Workshop on Network on Chip Architectures*, 2023, pp. 15–20.

[9] E. Taheri, S. Pasricha, and M. Nikdast, "Resipi: A reconfigurable silicon-photonic 2.5 d chiplet network with pcms for energy-efficient interposer communication," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, 2022, pp. 1–9.

[10] J. Yin, Z. Lin, O. Kayiran, M. Poremba, M. S. B. Altaf, N. E. Jerger, and G. H. Loh, "Modular routing design for chiplet-based systems," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018, pp. 726–738.

[11] E. Taheri, S. Pasricha, and M. Nikdast, "Deft: A deadlock-free and fault-tolerant routing algorithm for 2.5 d chiplet networks," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022, pp. 1047–1052.

[12] M. Ebrahimi and M. Daneshtalab, "EbDa: A new theory on design and verification of deadlock-free interconnection networks," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 2017, pp. 703–715.

[13] P. Majumder, S. Kim, J. Huang, K. H. Yum, and E. J. Kim, "Remote control: A simple deadlock avoidance scheme for modular systems-on-chip," *IEEE Transactions on Computers*, vol. 70, no. 11, pp. 1928–1941, 2020.

[14] R. Wang, Z. Li, S. Kannan, and K. Chakrabarty, "Pre-bond testing of the silicon interposer in 2.5D ICs," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 978–983.

[15] H. Zhu, P. P. Pande, and C. Grecu, "Performance evaluation of adaptive routing algorithms for achieving fault tolerance in noc fabrics," in *2007 IEEE International Conf. on Application-specific Systems, Architectures and Processors (ASAP)*. IEEE, 2007, pp. 42–47.

[16] P. Vivet, E. Guthmuller, Y. Thonnart, G. Pillonnet, C. Fuguet, I. Miro-Panades, G. Moritz, J. Durupt, C. Bernard, D. Varreau *et al.*, "Intact: A 96-core processor with six chiplets 3d-stacked on an active interposer with distributed interconnects and integrated power management," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 1, pp. 79–97, 2020.

[17] P. Coudrain, J. Charbonnier, A. Garnier, P. Vivet, R. Vélard, A. Vinci, F. Ponthenier, A. Farcy, R. Segaud, P. Chausse *et al.*, "Active interposer technology for chiplet-based advanced 3d system architectures," in *2019 IEEE 69th Electronic Components and Technology Conference (ECTC)*. IEEE, 2019, pp. 569–578.

[18] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," *ACM SIGARCH Computer Architecture News*, vol. 20, no. 2, pp. 278–287, 1992.

[19] Y. Wu, L. Wang, X. Wang, J. Han, J. Zhu, H. Jiang, S. Yin, S. Wei, and L. Liu, "Upward packet popup for deadlock freedom in modular chiplet-based systems," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2022, pp. 986–1000.

[20] S. Bharadwaj, J. Yin, B. Beckmann, and T. Krishna, "Kite: A family of heterogeneous interposer topologies enabled via accurate interconnect modeling," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[21] M. Hasanzadeh, E. Taheri, M. Shafaei, and A. Patooghy, "Fastest: A concurrent strategy to test components of 3d network-on-chips," in *2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2019, pp. 65–68.

[22] J. Wang, M. Ebrahimi, L. Huang, X. Xie, Q. Li, G. Li, and A. Jantsch, "Efficient design-for-test approach for networks-on-chip," *IEEE Transactions on Computers*, vol. 68, no. 2, pp. 198–213, 2018.

[23] M. Ebrahimi, M. Daneshtalab, and J. Plosila, "Fault-tolerant routing algorithm for 3D NoC using hamiltonian path strategy," in *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2013, pp. 1601–1604.

[24] ——, "High performance fault-tolerant routing algorithm for noc-based many-core systems," in *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. IEEE, 2013, pp. 462–469.

[25] Y. Zou and S. Pasricha, "NARCO: Neighbor aware turn model-based fault tolerant routing for NoCs," *IEEE Embedded Systems Letters*, vol. 2, no. 3, pp. 85–89, 2010.

[26] S. Pasricha, Y. Zou, D. Connors, and H. J. Siegel, "Oe+ ioe: A novel turn model based fault tolerant routing scheme for networks-on-chip," in *Proceedings of the eighth IEEE/ACM/IFIP international conference on hardware/software codesign and system synthesis*, 2010, pp. 85–94.

[27] S. Pasricha and Y. Zou, "Ns-ftr: A fault tolerant routing scheme for networks on chip with permanent and runtime intermittent faults," in *16th Asia and South Pacific Design Automation Conference (ASP-DAC 2011)*. IEEE, 2011, pp. 443–448.

[28] R. Salamat, M. Khayambashi, M. Ebrahimi, and N. Bagherzadeh, "A resilient routing algorithm with formal reliability analysis for partially connected 3D-NoCs," *IEEE Transactions on Computers*, vol. 65, no. 11, pp. 3265–3279, 2016.

[29] Y. Li and D. Goyal, *3D microelectronic packaging: From fundamentals to applications*. Springer, 2017, vol. 57.

[30] H.-Y. Hsiao and J.-K. Lin, "Electromigration reliability and morphologies of Cu pillar with microbump under high current density stressing," in *2015 IEEE 17th Electronics Packaging and Technology Conference (EPTC)*. IEEE, 2015, pp. 1–4.

[31] E. Taheri, R. G. Kim, and M. Nikdast, "Adele: An adaptive congestion-and-energy-aware elevator selection for partially connected 3D NoCs," in *Design Automation Conference*, 2021.

[32] K. Bhardwaj, K. Chakraborty, and S. Roy, "An milp-based aging-aware routing algorithm for nocs," in *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2012, pp. 326–331.

[33] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, "Cycle-accurate network on chip simulation with noxim," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 27, no. 1, pp. 1–25, 2016.

[34] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti *et al.*, "The Gem5 simulator," *ACM SIGARCH computer architecture news*, vol. 39, no. 2, pp. 1–7, 2011.

[35] S. Pasricha, "Exploring serial vertical interconnects for 3D ICs," in *Proceedings of the 46th Annual Design Automation Conference*, 2009, pp. 581–586.

[36] C. Bienia, *Benchmarking modern multiprocessors*. Princeton University, 2011.

[37] A. B. Kahng, B. Lin, and S. Nath, "ORION3.0: A comprehensive NoC router estimation tool," *IEEE Embedded Systems Letters*, vol. 7, no. 2, pp. 41–45, 2015.

Ebadollah Taheri (Student Member, IEEE) earned his M.Sc. degree in Electronic Engineering from Iran University of Science and Technology in 2016. Currently, he is a Ph.D. candidate in Computer Engineering with the Department of Electrical and Computer Engineering at Colorado State University, USA. His research primarily revolves around Reliable and High-Performance Computer Architecture, with a specific focus on On-Chip Interconnection Networks.

Sudeep Pasricha (Fellow, IEEE), received his Ph.D. in Computer Science from the University of California, Irvine in 2008. He is currently a Professor in the Department of Electrical and Computer Engineering, at Colorado State University. His research focuses on the design of innovative software algorithms, hardware architectures, and hardware-software codesign techniques for energy-efficient, fault-tolerant, real-time, and secure computing. He is a Fellow of the IEEE, Fellow of the AAIA, and Distinguished Member of the ACM.

Mahdi Nikdast (Senior Member, IEEE) is an Associate Professor and an Endowed Rockwell-Anderson Professor in the Department of Electrical and Computer Engineering at the Colorado State University (CSU), Fort Collins, CO. He received his Ph.D. in Electronic and Computer Engineering from The Hong Kong University of Science and Technology (HKUST), Hong Kong. His research focuses on the design and development of photonic systems-on-chip and next-generation data-communication, computing, and sensing systems employing integrated photonics while emphasizing energy efficiency and robustness. He is a senior member of the IEEE.