2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA) | 978-1-6654-2027-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/HPCA53966.2022.00076

Upward Packet Popup for Deadlock Freedom in Modular Chiplet-Based Systems

Yibo Wu*, Liang Wang†, Xiaohang Wang‡, Jie Han§, Jianfeng Zhu*, Honglan Jiang¶, Shouyi Yin*, Shaojun Wei*, Leibo Liu*
* *School of Integrated Circuits, Tsinghua University, Beijing, China*
Corresponding author: Leibo Liu, liulb@tsinghua.edu.cn
† *School of Computer Science and Engineering, Beihang University, Beijing, China*
‡ *School of Software Engineering, South China University of Technology, Guangzhou, China*
§ *Department of Electrical and Computer Engineering, University of Alberta, Canada*
¶ *Department of Micro-Nano Electronic, Shanghai Jiao Tong University, Shanghai, China*

***Abstract*—Monolithic SoCs can be decomposed into disparate chiplets that support integration with advanced packaging technologies. This concept is promising in reducing the manufacturing cost of large scale SoCs due to the higher yield rate and reusability of chiplets. The chiplets should be designed in a modular manner without holistic system knowledge so that they can be reused in different SoCs. However, the design modularity is a major challenge to the networks-on-chip (NoCs) of chiplets.**

New deadlocks may occur across both the chiplets and the interposer due to the integration, even if the NoC of each individually designed chiplet is deadlock free. However, conventional deadlock freedom approaches are unsuitable to handle such deadlocks because they require holistic knowledge and violate the modularity. Although there are several modular approaches that specifically target at integration-induced deadlocks, their routing is overly restricted and the injection control incurs additional latency. They also lack flexibility in dynamically changing topologies due to their complex software algorithm and the hard-wired components.

In this paper, a key insight on the chiplet integration-induced deadlocks is gained, inspired by which a deadlock recovery framework (named UPP) is proposed. Specifically, it is verified that an integration-induced deadlock always involves a stalled upward packet moving from the interposer to the connected chiplet via the vertical link. Thus, UPP detects a deadlock by discovering the upward packet and recovers the system from deadlock by transmitting the upward packet to its destination. Hybrid flow control mechanisms are proposed to enable the upward packet to bypass the buffers and be transmitted via the normal router datapath. To guarantee the ejection of the upward packet after transmission, a lightweight protocol is proposed to reserve ejection queue entries of the network interface. Experimental results show that while adhering to design modularity, UPP provides an average runtime speedup of 3.1%∼10.3% with an area overhead of less than 4%.

***Keywords*-Deadlock Recovery, Modular, Chiplet, Network-on-Chip;**

I. Introduction

In chiplet-based systems, a SoC is decomposed into chiplets which are stacked on an interposer via advanced packaging technologies [5], [22]. Due to the high yield rate and the reusability of small chiplets, the manufacturing cost for large scale SoCs is significantly reduced [27]. A critical issue for chiplet-based systems is the modularity of chiplet design [51]. If the chiplet needs to be reused across variously configured systems [27], the design and verification of each chiplet should be modular and independent of the rest of the system.

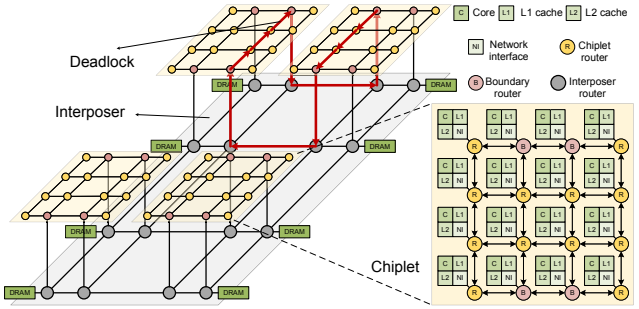


Figure 1. An overview of the baseline system and a deadlock buffer dependency chain that crosses two chiplets. The topologies of both the interposer and chiplets are 4×4 mesh networks. Each chiplet is connected to four interposer routers via four boundary routers and four vertical links.

There is an emerging research focus on constructing chiplet-based systems on active interposers [43]. While adhering to design modularity, such systems require additional effort in resolving routing deadlocks that may occur in the NoC. Even if each chiplet has a local NoC that is modularly designed and locally free of deadlocks, new deadlocks spanning across different chiplets and the interposer may be introduced when these chiplets are integrated together [51]. Fig. 1 shows a baseline system used in this paper, where a possible deadlock that crosses two chiplets occurs due to chiplet integration on the active interposer.

Conventional deadlock freedom approaches that are proposed for flat networks are unsuitable to handle integration-induced deadlocks. These approaches require a global view of the whole system and prevent each chiplet from independently configuring and optimizing its design space, thereby violating the modularity and reusability of chiplets. For example, spanning tree-based approaches execute multiple system-scale breadth-first searches for routing table construction and turn restriction placement [2], [32]. However,

978-1-6654-2027-3/22/$31.00 ©2022 IEEE
DOI 10.1109/HPCA53966.2022.00076

the knowledge of the whole system topology is unavailable at the time of designing modular chiplets.

Recently, there are approaches specifically proposed for modular chiplet-based systems [24], [51]. The composable routing [51] places unidirectional turn restrictions on the chiplet boundary routers. Remote control [24] uses a permission subnetwork for injection control to separate intra-chiplet and inter-chiplet traffic flows. However, these approaches are not satisfactory in terms of performance and flexibility. The excessive turn restrictions of the composable routing reduces path diversity and incurs load imbalance. The injection control of remote control results in a longer latency. Besides, if the network is dynamically reconfigured due to faulty components [2], [34] or power gating [38], [39], the complex software algorithm of the composable routing and the hard-wired permission subnetwork of remote control render themselves inflexible.

To achieve both high performance and flexibility while ensuring chiplet design modularity, UPP, which obtains routing deadlock freedom in modular chiplet-based systems through **U**pward **P**acket **P**opup is proposed. UPP leverages a key insight: an integration-induced deadlock crossing multiple chiplets always involves an upward packet. This upward packet is permanently stalled in the interposer router due to deadlock while attempting to reach the connected chiplet via the upward vertical link. This insight indicates that a deadlock buffer dependency chain can be broken if the upward packet could be successfully transmitted to its destination chiplet router and ejected. The popup here denotes the operations of both transmission and ejection of a packet from the interposer to the destination chiplet. As a deadlock recovery framework, UPP permits the formation of deadlocks, but always efficiently recovers the network from deadlocks after detecting them.

This paper makes the following contributions:

- We identify the close relationship between upward packets and integration-induced deadlocks. This observation is generic in any chiplet-based systems and provides a new thinking in resolving integration-induced deadlocks.
- We design UPP as a deadlock recovery framework. Deadlocks may occur in the fully adaptive network, while UPP always detects the deadlocks and recovers the system from deadlocks through upward packet popup. The strategy of detection and recovery achieves full path diversity and avoids performance penalty when the network is free of deadlocks.
- We present a detailed implementation of UPP. UPP adopts a simple timeout mechanism for deadlock detection and minimizes the negative impact of false positives. The recovery procedure of UPP consists of a lightweight protocol for ejection and hybrid flow control mechanisms for transmission. Compared with state-of-the-art approaches in modular chiplet-based systems, UPP improves the saturation throughput by 18%∼72% and reduces the latency by 4.5%∼8.2%, at the expense of less than 4% area overhead.

II. Background

A. Chiplet-Based System

To confront with the slowdown of Moore's law and Dennard' scaling, the concept of dividing a large SoC into small chiplets and integrating them with advanced packaging technology is becoming increasingly popular. Although most researches and industry products [4], [27], [41], [46] integrate chiplets on passive interposers that only serve the purpose of routing, recent studies have shown that using active interposers can be practical [22], [43]; thus, many components in chiplets are offloaded to the interposer and chiplets can be further simplified. The active interposer is a large chip with an NoC connected with chiplet NoCs via multiple vertical links for inter-chiplet communication. Chiplet-based systems should be composable and modular for reusing chiplets in different systems. Even if each chiplet may come from different vendors with variable hardware configurations, the system is still functionally correct when these independently designed and verified chiplets are integrated together [51]. However, chiplet design modularity poses a severe challenge in ensuring correctness in the system NoC. Integrating such modularly designed chiplets can introduce new deadlocks [1] that span across multiple chiplets and the interposer.

B. NoC Routing Deadlock Freedom

Routing deadlock is a cyclic buffer dependency chain. To resolve it, existing approaches can be classified into five categories according to their fundamental theories [29], [37]:

(1) Dally's theory-based approaches: These approaches construct an acyclic channel dependency graph (CDG) [11] for deadlock freedom [2], [11], [14], [23], [25], [32], [33].

(2) Duato's theory-based approaches: Duato's theory [13] proves that in a cyclic CDG, the network is still deadlock free by maintaining an acyclic escape path [3], [34], [39], [45].

(3) Bubble flow control (BFC)-based approaches: BFC [8], [9], [35], [38], [47] uses bubble buffers to sustain packets' movement for deadlock freedom.

(4) Deflection-based approaches: These approaches [16], [17], [26], [29]–[31], [49], [50] deflect a flit to another output port when contention exists. There might be misrouting, but the packet movement is guaranteed for deadlock freedom.

(5) SPIN-based approaches: SPIN [37] views the deadlock as a lack of coordination. It resolves deadlocks by orchestrating packets along the deadlock chain to move one hop forward.

[1] UPP targets at routing deadlocks. Protocol deadlocks are assumed to be handled by using multiple virtual networks.

Table I
QUALITATIVE COMPARISON OF EXISTING DEADLOCK FREEDOM APPROACHES

		Design Modularity			System Performance		Network Flexibility
		topology	VC	flow control	full path diversity	w/o injection control	topology independence
Conventional approaches	Dally's theory-based	✗	✓	✓	✗	✓	✗
	Duato's theory-based	✗	✗	✓	✗	✓	✗
	BFC-based	✓	✓	✗	✓	✓	✓
	Deflection-based	✓	✓	✗	✓	✓	✓
	SPIN-based	✓	✓	✗	✓	✓	✓
Modular approaches	Composable routing	✓	✓	✓	✗	✓	✗
	Remote control	✓	✓	✓	✓	✗	✗
	UPP (this work)	✓	✓	✓	✓	✓	✓

III. MOTIVATION

Among the numerous NoC deadlock freedom approaches, the topology-agnostic ones can address deadlocks in any irregular topologies. However, they are unsuitable to handle integration-induced deadlocks because they can violate design modularity (Sec. III-A). On the other hand, although there are other approaches proposed specifically for modular chiplet-based systems, they suffer from poor performance (Sec. III-B) and low flexibility (Sec. III-C). As summarized in Tab. I, existing approaches have weaknesses in terms of either design modularity, performance or flexibility, which motivates the necessity of developing UPP.

A. Design Modularity of Chiplets

Design modularity is imperative to guarantee that each chiplet can be independently configured and optimized in its design space. However, conventional NoC deadlock freedom approaches typically require holistic system knowledge and violate the design modularity. Although there can be multiple definitions of design modularity, this paper focuses on three modularity attributes for the system NoC:

Topology modularity: Each chiplet has its local topology that is optimized for its configurations and applications. Topology modularity is considered to be satisfied if an individual chiplet design is not affected by the topology of the rest of the system. Dally's theory-based approaches that place turn or VC usage restrictions violate the topology modularity. Duato's theory-based approaches that place turn restrictions in the escape path also violate the topology modularity.

VC modularity: Each chiplet independently configures its number of VCs according to its hardware budget, power, traffic load, etc. VC modularity is considered to be satisfied if an approach allows minimally 1 VC per virtual network (VNet). Duato's theory-based approaches that use additional VCs to construct the escape path violate the VC modularity.

Flow control mechanism modularity: Most NoC designs adopt wormhole or virtual cut-through flow control mechanisms [21]. Flow control mechanism modularity is considered to be satisfied if an approach can support both of these flow control mechanisms. BFC-based approaches require virtual cut-through and violate this modularity. Deflection-based approaches need to support packet truncation and reassembling if they are applied in networks using wormhole. The procedures of truncation and reassembling are hardware-expensive and usually not supported in most NoCs. The synchronized packet movement in SPIN also requires virtual cut-through flow control and violates this modularity.

Consequently, none of the conventional deadlock freedom approaches possesses all the three attributes for design modularity, as shown in Tab. I.

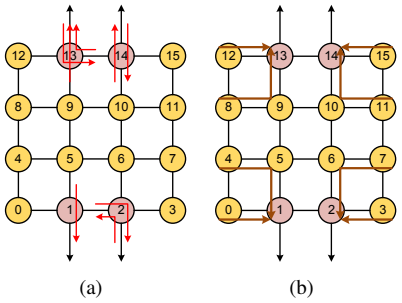


Figure 2. (a) The composable routing imposes 8 unidirectional turn restrictions (red arrows) on the 4 red boundary routers in a 4×4 mesh chiplet. (b) Remote control uses tree-like permission subnetworks (brown arrows) for boundary buffer reservation.

B. System Performance

The composable routing [51] and remote control [24] are specifically designed for modular chiplet-based systems. The composable routing observes that from the perspective of an individual chiplet, the rest of the system can be abstracted into a virtual node connecting with the chiplet via boundary routers. Therefore, it is possible to place unidirectional turn restrictions solely on the boundary routers to avoid deadlocks.

Remote control avoids integration-induced deadlocks through injection control. It observes that in the deadlock dependency chain, there is always an intra-chiplet packet being blocked by an inter-chiplet packet. Hence, deadlocks can be avoided by isolating inter-chiplet packets and intra-chiplet packets. Remote control adds additional buffers at boundary routers that can store all inter-chiplet packets for the isolation. Before an inter-chiplet packet is injected, it needs to send a signal on a hard-wired permission subnetwork to the boundary router and reserve a buffer slot.

Although both of these two approaches satisfy the modularity attributes, their throughput and latency are penalized due to the wasted path diversity and the injection control.

Path diversity: The composable routing places excessive turn restrictions on boundary routers. As the chiplets are connected with the interposer via boundary routers, the network path diversity is severely affected. Fig. 2(a) shows the composable routing in a 4×4 mesh chiplet; it places 8 unidirectional turn restrictions on the 4 boundary routers. The throughput is reduced due to the wasted bandwidth. Worse still, the turn restrictions force all inter-chiplet packets from router 2∼11 to reach the interposer via boundary router 2. Many packets are forced to take a longer route to reach the interposer, resulting in an increase in latency.

Injection control: In remote control, the inter-chiplet packet injection control requires at least 2 additional cycles for the handshaking round-trip time. Besides, due to the limited boundary buffer resources, this latency grows due to the contention in buffer reservation. When inter-chiplet packets are crossing the boundary routers, remote control requires these packetst to attend switch allocation, switch traversal and VC allocation in three separate stages. This further induces one-cycle latency when remote control is applied in state-of-the-art router pipelines [21] where the VC allocation and switch allocation stages are performed in parallel.[2]

C. Network Flexibility

In large scale SoCs where the chiplet design concept is more likely adopted, the hardware faults and power gating can lead to a dynamic reconfiguration of the network topology [2], [18], [38]. To achieve a higher flexibility in dynamic topologies, an approach should be topology-independent and be able to reconfigure within an acceptable time [23], [38].

The composable routing lacks network flexibility. In order to place the turn restrictions, it uses a complex software algorithm that sweeps all possible combinations of turn restrictions, checks their network connectivity if the turn restrictions were applied and finally selects the one that likely has the best performance. Such procedures with exponential complexity are practical only during the design time of a chiplet. Remote control constructs a tree-like permission subnetwork for the inter-chiplet packet injection control. In order to reduce the reservation round-trip delay, the subnetwork is hard-wired as shown in Fig. 2(b). Such an architecture lacks flexibility and cannot dynamically reconfigure when the topology changes.

[2]Although we acknowledge that the injection control of remote control could be more efficient by using specialized circuits or credit-based control, possible optimization mechanisms are outside the scope of this paper.

IV. Deadlock freedom theory and the basic design

A. Deadlock Freedom Theory

UPP is proposed based on a key **observation: in a chiplet-based system, an integration-induced deadlock always involves at least an upward packet, that is stalled while attempting to move upward from an interposer router to the connected chiplet boundary router.** Therefore, by removing the upward packet from the interposer router and sending it to the destination router through the chiplet, the buffer dependency chain is broken and the corresponding deadlock is resolved. The procedures of transmitting and ejecting a packet is denoted as popup. Fig. 3 exemplifies an upward packet and how UPP resolves the deadlock.

1) Benefit of the upward packet popup: An integration-induced deadlock can theoretically involve any packets in the system. Among all candidate packets, why is the upward packet in particular worth being handled for deadlock freedom? To answer it, upward packet popup is compared with other possible options in resolving deadlocks.

The first option is to deal with all packets that have the chance to be in a deadlock. This option is the one adopted by conventional NoC deadlock approaches. As any packets can be involved in a deadlock, this option requires a holistic view of the whole system and violates the design modularity.

The second option is to deal with all intra-chiplet and inter-chiplet packets on the boundary routers. This option is adopted by the composable routing and remote control. Due to the typically low frequency of deadlocks [18], [28], [37], most packets are not involved in real deadlocks. This option thus unnecessarily leads to performance penalty on these packets.

The third option is to deal with a downward packet (symmetric to the upward packet) that is involved in a deadlock. Due to the routing algorithm of chiplet-based systems (detailed in Sec. V-D), an upward packet can directly reach the destination chiplet via popup, whereas a downward packet needs to enter the interposer first, then cross the interposer and finally reach the destination chiplet. Thus, the upward packet popup incurs a shorter latency and is obviously more efficient.

B. Basic Design of UPP

UPP is a deadlock recovery framework [12], [38], i.e., UPP allows integration-induced deadlocks to form and

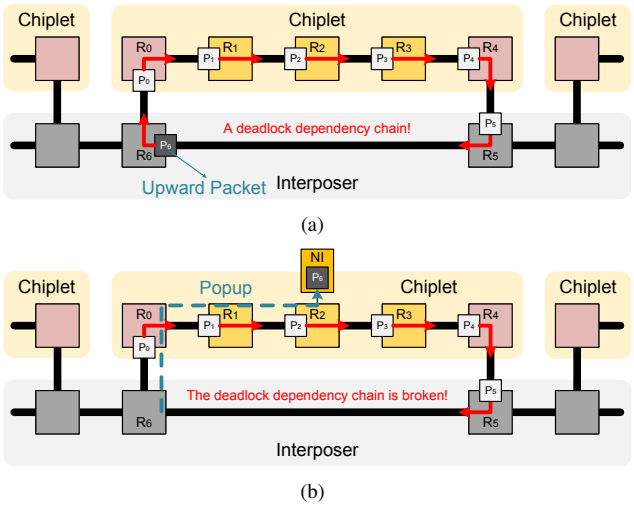


Figure 3. Deadlock freedom with UPP. The grey squares represent interposer routers. The pink squares represent chiplet boundary routers. The yellow squares represent the other normal chiplet routers. (a) Packet P0∼P6 constitute an integration-induced deadlock which crosses both the chiplet and the interposer. Packet P6 is the upward packet. (b) After upward packet popup, packet P6 is sent through the chiplet and reaches the NI of its destination router R2. Now the buffer dependency chain is broken and the deadlock is resolved.

temporarily exist, but then detects and recovers from all deadlocks with upward packet popup.

1) Deadlock detection: For the deadlock detection (detailed in Sec. V-A), UPP first determines the existence of a deadlock by using a simple timeout mechanism. When a deadlock is there, UPP then selects a possible upward packet from all stalling packets in a round robin manner for popup. Since any packets attempting to move upward from the interposer either successfully proceed upward to their destination chiplets, or keep being stalled until they are deemed as upward packets, this two-step mechanism is able to detect all deadlocks. However, as the timeout mechanism does not distinguish a real deadlock from a temporary congestion [38], [50], a concern for using this mechanism is the possible false positive detections. The implementation and evaluation of UPP demonstrate that: first, dealing with false positives is actually resolving congestion and promoting packets' forward progress; second, the popup procedures merely spend a small number of resources on handling false positives.

2) Deadlock recovery: For the deadlock recovery, UPP needs to address two problems. The first one relates to the upward packet ejection (detailed in Sec. V-B). When the upward packet reaches its destination NI, it might cannot be ejected due to a full ejection queue. UPP addresses this problem by using a lightweight protocol that reserves an empty ejection queue entry for the upward packet to eject. The protocol requires additional three types of signals, UPP_req, UPP_ack and UPP_stop. Two 32-bit buffers are added in each chiplet router so that these signals can be transmitted in the same router datapath as the normal head flits.

The second problem relates to the upward packet transmission through the chiplet to its destination router (detailed in Sec. V-C). The routing path might be full of congested packets and there are no free buffers in intermediate chiplet routers to temporarily hold the upward packet. UPP addresses it by using hybrid flow control mechanisms of both wormhole (or virtual cut-through) and circuit switching in the chiplet [10], [20]. The aforementioned protocol helps set up a circuit for the upward packet to bypass buffers and be transmitted without any blocking.

During the deadlock recovery, an assumption is made that packets proceeding to the same destination chiplet router should enter the destination chiplet through the same boundary router. The reasonability of the assumption is disscussed in Sec. V-D.

V. UPP IMPLEMENTATION

A. Deadlock Detection

UPP uses a two-step mechanism for deadlock detection. The first step confirms whether an interposer router is likely to be involved in a deadlock. The second step discovers the upward packet of a deadlock for subsequent popup procedures.

In the first step, UPP equips each interposer router with one timeout counter per VNet. The timeout counter records the duration that there exist packets of this VNet that are stalled while moving upward from the interposer router, but none of them has been sent out from the up output port to the connected chiplet. If the counter is over a pre-configured threshold value, UPP deems that there is a deadlock in this VNet. This design leverages the key observation that an integration-induced deadlock always involves an upward packet, i.e., always involves the up output port. Thus, when there is a deadlock, the up output port should be permanently blocked for the corresponding VNet.

After confirming a deadlock, UPP enters the second step. In the deadlocked VNet, among all the VCs that has a packet stalling while moving upward, a round robin arbiter selects a packet from one VC as the upward packet. As it is expensive to tell whether these packets are real upward packets or they encounter a serious congestion, the arbiter ensures that sooner or later all packets stalled while moving upward have the chance to be selected as the upward packet for popup.

There is a restriction on deadlock detection that each interposer router can select at most one upward packet for each VNet, regardless of the number of input ports and the number of VCs. In this way, if multiple deadlocks of the same VNet involve the same interposer router, they are resolved serially via popup so as to relieve the pressure of recovery and reduce hardware overhead. On the contrary, deadlocks of different VNets are resolved concurrently to

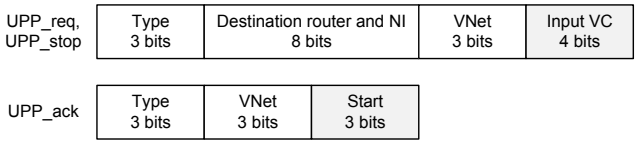


Figure 4. The message encoding of UPP_req, UPP_ack and UPP_stop. The grey fields are only used in wormhole flow control.

guarantee the correctness of popup, as further detailed in Sec. V-B4.

Although the timeout mechanism might introduce false positives, handling false positives together with deadlocks is necessary for better network performance. In the first detection step, when UPP discovers a potential deadlock, the up output port of the interposer router has been idle for a long time, which is indeed a tremendous waste of network bandwidth. Therefore, dealing with false positives promotes the forward movement of congested packets, makes a better usage of bandwidth and improves throughput.

B. Upward Packet Ejection

To cope with the situation that an upward packet cannot get ejected due to a full ejection queue when it reaches the destination NI, UPP uses a lightweight protocol to reserve an ejection queue entry before the upward packet is popped up. The protocol adds three types of signals that are transmitted through the normal router datapath by adding two 32-bit buffers in each chiplet router.

1) Protocol design: The protocol adds three types of signals, UPP_req, UPP_ack and UPP_stop that are used obeying the following three rules.

• After an interposer router selects an upward packet, the router sends an UPP_req to the destination NI to reserve an ejection queue entry before popup.

• After the destination NI receives an UPP_req and successfully reserves an ejection queue entry, it sends an UPP_ack back to the interposer router. The interposer router can start the upward packet popup process for deadlock recovery only if the UPP_ack is received.

• Before receiving the UPP_ack, if the upward packet is not involved in a real deadlock and proceeds to the connected chiplet, the interposer router sends an UPP_stop to the destination NI so that the reserved ejection queue entry can be recycled. Later when the interposer router receives the UPP_ack, the UPP_ack is discarded.

2) Protocol signals transmission: The three new types of signals are transmitted between an interposer router and a chiplet NI in the same manner as a normal head flit, i.e., they go through the same router pipeline and use the same crossbars and links. One additional buffer is added in every chiplet router to transmit both UPP_req and UPP_stop and another is added to transmit UPP_ack. As no data payload is carried, these signals use a compact encoding format in order to narrow down the required buffer width, as shown in Fig. 4. UPP_req and UPP_stop have the same encoding format, where 3 bits specify the signal type, 8 bits specify the destination router and NI for route computation, and another 3 bits specify the VNet of the upward packet with one-hot encoding (the MESI cache coherence protocol used for evaluation in this paper requires 3 VNets). If the interposer uses the wormhole flow control, UPP_req has an additional 4-bit field to specify the input VC where the upward packet locates at (detailed in Sec. V-B3). As for UPP_ack, the 8-bit field for destination router and NI are removed because UPP_ack does not attend normal route computation as explained in the next paragraph. An additional 3-bit field with one-hot encoding specifies whether the popup process of each VNet has started. If the interposer uses the wormhole flow control, UPP_ack has another 3-bit field with one-hot encoding to indicate whether the popup process of this VNet has started when the UPP_ack is received. Thus, UPP_req and UPP_stop require an additional 18-bit buffer, while UPP_ack requires a 9-bit buffer. In the implementation and evaluation, these two buffers are 32-bit wide for a conservative estimation.

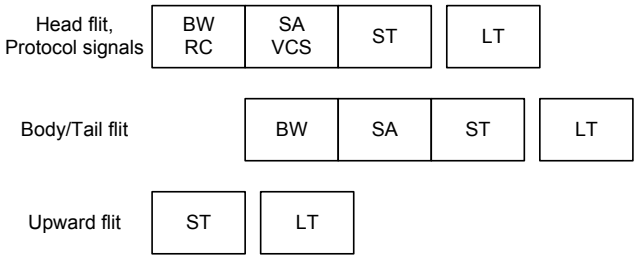


Figure 5. The router pipeline of different types of flits/protocol signals. 'BW' and 'RC' are short for buffer write and route computation, respectively. 'SA' and 'VCS' stand for switch allocation and VC selection, respectively. 'ST' denotes switch traversal. 'LT' means link traversal.

Fig. 5 shows the router pipeline used in this paper. When a flit enters a router, it attends buffer write as well as route computation if it is a head flit. In the next cycle, it attends switch allocation. If the flit succeeds in switch allocation, it is randomly allocated a free VC by the VC selection [21]. The flit enters switch traversal and link traversal stages in the following two cycles. Body and tail flits bypass the route computation and VC selection stages. The three new types of signals attend the same pipeline stages as head flits, except that they have a higher priority than the other normal flits during the switch allocation to assure they reach the destination without blocking. When an UPP_ack is sent from a chiplet router back to an interposer router, it does not attend the normal route computation but instead follows the reverse routing path of its corresponding UPP_req.

3) Supporting wormhole flow control: If the interposer uses wormhole flow control and the discovered upward packet has partly been transmitted to the chiplet, starting upward packet popup from the interposer router can result in packet truncation. Therefore, several slight modifications

are added to support wormhole flow control.

First, if a discovered upward packet is partly-transmitted, the UPP_req follows the route of the upward packet (with the 4-bit input VC field) until the head flit is found. Then, the corresponding chiplet router tags the VC of the head flit so that upward packet popup would start from this position.

Second, when the UPP_ack follows the reverse route of the UPP_req and reaches the chiplet router where the head flit locates at, if the head flit has been sent out, the UPP_ack is directly discarded. Otherwise, the upward packet popup starts instantly. Then, the corresponding VNet in the 3-bit start field of the UPP_ack is set to 1 to indicate the start of popup and the UPP_ack continues to be sent to the interposer router. When the UPP_ack reaches the interposer router, the interposer router is informed that the upward packet popup has already started in the chiplet by scanning the 3-bit start field. The interposer router behaves as normal until the last flit of the upward packet is sent out to the up output port. Then, the upward packet popup process completes for this VNet.

4) Proof of correctness: To prove the correctness of the above protocol, i.e., an UPP_req can always succeed in reserving an ejection queue entry, an exemplary scenario of a simple request-response cache coherence protocol is used.

In the NoC model, a router is connected with a processing element (PE, e.g., core, cache, directory) via NI. The NI has injection queues to receive messages from the PE and segment messages into packets for injection. Also, the NI has ejection queues to receive packets from the network and reassemble them into messages before they are consumed by the PE. Both NI injection and ejection queues are separated for each message class to avoid protocol deadlocks [18], [42]. Based on this model, there are two cases to discuss.

In the first case, an UPP_req is reserving a response ejection queue entry. When response messages reach the PE through the ejection queue, they can always get consumed timely by the PE because response message is the terminating message type of the request-response cache coherence protocol [42]. Therefore, the response message ejection queue would never be permanently full and the UPP_req can always succeed in reserving an entry. In addition, as UPP ensures concurrent upward packet popup for different VNets, response packets can always reach their destinations and get consumed by the PE, regardless of the routing condition of other message types.

In the second case, an UPP_req is reserving a request ejection queue entry. When the PE is processing a request message from the ejection queue, it checks whether there is an available injection queue entry to buffer the response message that the message processing will generate. If there is, the request message is consumed. An ejection queue entry is freed up and the UPP_req is able to reserve an entry. Otherwise, the PE delays the processing of the request message until the response injection queue has available entries. As proved in the first case that any response packets can always get consumed, the VNet for response packets become idle at last so that all response packets in the injection queue can be injected. Therefore, there will be available response injection queue entries and the request message can get consumed. Consequently, the UPP_req is able to reserve an entry when a request message is consumed.

5) Avoiding protocol signal contention: As each chiplet router incorporates only two additional 32-bit buffers, UPP needs to address the potential contentions among the protocol signals that can result in buffer overflow. This problem is discussed by considering the following four cases.

First, UPP_req and UPP_stop from the same interposer router: Protocol signals from the same interposer router are sent out in a serial manner. By ensuring that the minimum time gap between the two consecutive signals is $Size\_of\_Data\_Packet + 1$ cycles, no contention occurs between them.

Second, UPP_req and UPP_stop from different interposer routers: One way to avoid the possible contention is by coordinating among the interposer routers connected with the same chiplet, so that for each VNet, there is only one popup underway in the chiplet. The other is by modifying the routing algorithm. It trades off routing adaptiveness for better popup parallelism. As detailed later in Sec. V-D, the routing algorithm used in UPP has a characteristic that in order to reach one chiplet router, flits (including the signals) from the interposer must have entered the chiplet from the same boundary router, i.e., the same connected interposer router.

Third, multiple UPP_acks from the same chiplet router: As UPP_acks follows the reverse routing paths of their UPP_reqs, these signals should have the same destination interposer router. The contention can be avoided by sending these signals in a serial manner.

Fourth, multiple UPP_acks from different chiplet routers: In this case, contentions are possible if these signals have the same destination interposer router. However, as the UPP deadlock detection ensures at most $number\_of\_VNets$ concurrent upward packet popups of different VNets from one interposer router, these signals can thus be merged by ORing their 3-bit one-hot VNet and start fields of Fig. 4.

C. Upward Packet Transmission

While transmitting the upward packet to its destination NI, there might be no available buffers in the routing path to temporarily store the upward packet. Although additional flit-sized buffers can be added, this hardware overhead is avoided by adopting hybrid flow control mechanisms in the chiplet. The chiplet uses wormhole flow control (or virtual cut through) for both normal flits and the protocol signals, whereas uses circuit switching for transmitting the upward flits in a buffer-bypassing manner.

The insight is that, before transmitting an upward flit, an UPP_req has already been sent from the interposer router to the same destination router. Therefore, the UPP_req can help set up a circuit so that the upward flit can be transmitted in a buffer-bypassing manner. The UPP_req goes through the same router pipeline stages as a normal head flit. After the UPP_req attends route computation, the connection between the input and the output ports can be recorded in the chiplet router. As a chiplet router can receive at most $Number\_of\_VNets$ UPP_reqs, the chiplet router needs to record at most $Number\_of\_VNets$ connections. The upward flit follows the same routing path of the UPP_req by looking up the connection records using its VNet id instead of attending route computation. It directly attends switch traversal with a higher priority than any other flits (including the protocol signals) without attending switch allocation. In this way, the upward flit can bypass buffers and only go through one router pipeline stage of switch traversal, as shown in Fig. 5.

The overhead of popup is the bandwidth used for sending the additional protocol signals because upward flits always make forward progress. As shown in the evaluation of Sec. VI, the bandwidth waste due to signal transmission has a negligible influence on performance and false positive in deadlock detection should not be a concern for UPP.

1) Avoiding conflicts between upward flits and protocol signals: Although it is mentioned in Sec. V-B5 that protocol signals from the same interposer router are sent out serially and contention-free among them, there could however be contention between upward flits and protocol signals due to their different pipeline stages (one stage for upward flits and three stages for protocol signals). A later upward flit can catch up with a previous protocol signal, which results in conflicts of crossbar switch usage. To avoid such conflict, the upward flit is given a higher priority. Consequently, the protocol signal is delayed for one cycle.

After delaying the protocol signal, a new concern arises. When a previously sent protocol signal is delayed by upward flits, could it be caught up by a later signal without being delayed? There are two cases to discuss.

In the first case, the later signal is sent out from the interposer router before upward flits are sent out. If the previous signal is caught up by these upward flits, then the later signal must have also been caught up by these flits, resulting in no conflict between the two signals.

In the second case, the later signal is sent out after upward flits are sent out. Assuming that N upward flits are sent out after the previous signal and before the later signal. There must be a time gap of at least $N + P$ cycles between the previous and the later signals. P represents the number of pipeline stages. In the worst scenario, the previous signal is delayed by upward flits by N cycles. Then, there is still a time gap of P cycles. Hence, there is no conflict between the two signals.

D. Routing Algorithm

In chiplet-based systems, the routing algorithm decides how the three types of packets are transmitted:

First, packets moving within the chiplet or the interposer: Each chiplet and the interposer has a local routing algorithm (as in normal flat networks) to transmit these packets.

Second, packets moving from a chiplet to the interposer: When a packet is injected into the chiplet, the routing algorithm selects a chiplet boundary router as an intermediate destination router for the packet. After reaching the boundary router by the local chiplet routing algorithm, this packet can enter the interposer via the downward vertical link.

Third, packets that move from the interposer to a chiplet: When a packet enters the interposer, the routing algorithm selects an interposer router as an intermediate destination router for the packet. After reaching the interposer router by the local interposer routing algorithm, this packet can reach the chiplet via the upward vertical link.

For the first type of packets, UPP allows each chiplet and the interposer to use a locally optimized routing algorithm, e.g., XY routing in regular mesh networks and ARIADNE [2] in irregular networks. For the second and third type of packets, UPP statically binds each chiplet router with the closest boundary router of the same chiplet together. If there are multiple boundary routers with the same minimal distance, the chiplet router is randomly bound with one of them. To transmit the second type of packets, the routing algorithm selects the boundary router bound with the source chiplet router. To transmit the third type of packets, the routing algorithm selects the interposer router whose connected boundary router is bound with the destination chiplet router. This mechanism adheres to chiplet design modularity since the static binding does not require system knowledge outside an individual chiplet.

In dealing with the third type of packets, the static binding between chiplet routers and boundary routers avoids the contention among signals as mentioned in Sec. V-B. If different packets have the same destination chiplet router, they will enter the destination chiplet from the same interposer router. A concern is whether the static binding mechanism degrades the network performance due to the lack of adaptiveness. However, it is observed that a dynamic binding incurs non-minimal routing because an inter-chiplet packet might select a more distant boundary router, thereby increasing the latency and reducing the throughput. Therefore, the modified routing algorithm is preferred over interposer router coordination.

E. UPP Microarchitecture

Fig. 6 shows the microarchitectural modifications to an interposer router, a chiplet router and a chiplet NI.

Modifications to an interposer router: The grey components around the up output port are added in every interposer

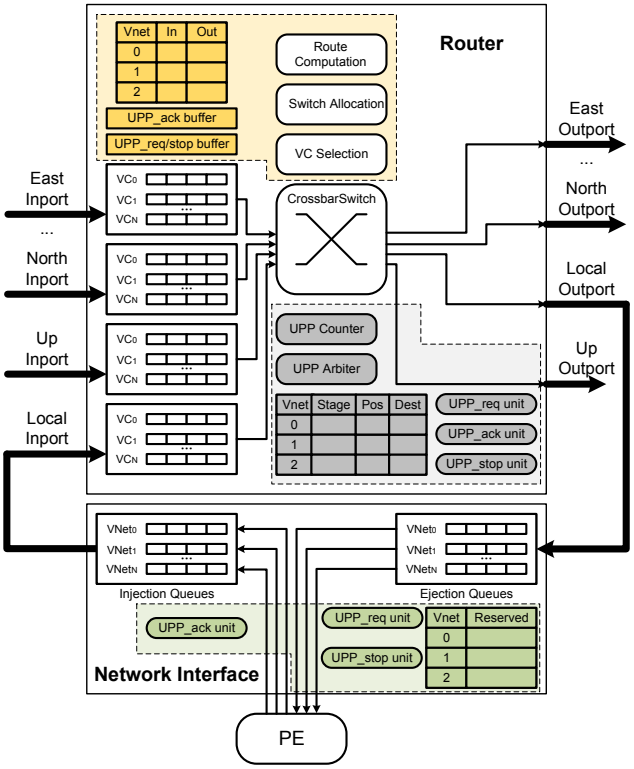


Figure 6. UPP router and NI microarchitecture. The light grey box in the middle encloses the modifications to an interposer router. The light yellow box on the top encloses the modifications to a chiplet router. The light green box at the bottom encloses the modifications to the chiplet NI.

router. The UPP counter of each VNet is used to record the idle time of the up output port. The UPP arbiter of each VNet is a round robin arbiter; it arbitrates over all VCs of its VNet to select a stalled packet as the upward packet. The table with an entry for each VNet records the stage of the popup, the position and the destination of the upward packet. The stage of the popup includes the following information: whether a popup occurs in this VNet, whether the UPP_req has been sent, whether the UPP_ack has been received, whether an UPP_stop should be sent and whether the upward packet has been sent out. The UPP_req, UPP_ack and UPP_stop units are used to send/receive the corresponding signals in a serial manner.

Modifications to a chiplet router: The yellow components on the top of Fig. 6 are added in every chiplet router. A 32-bit UPP_ack buffer is used to transmit all UPP_ack signals without buffer backpressure [21]. To avoid buffer overflow, the buffer is able to merge UPP_ack signals belonging to different VNets. A 32-bit UPP_req/UPP_stop buffer is used to transmit both UPP_req and UPP_stop signals without buffer backpressure. There is no buffer overflow as proved in Sec. V-B. These two buffers are shared by all input ports by adding several multiplexers. The table with an entry for each VNet records the input and the output

Table II
SIMULATION CONFIGURATIONS

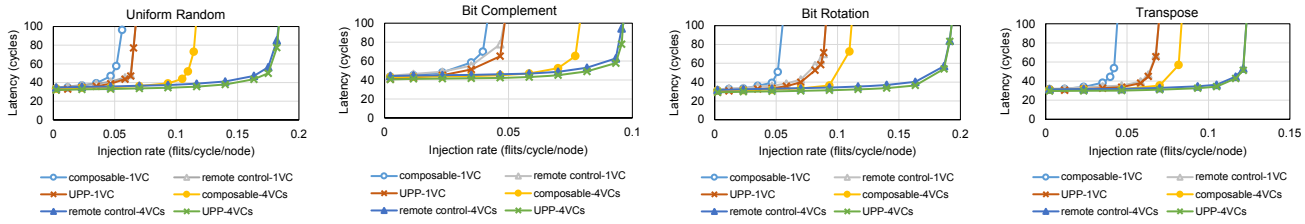
Full system simulation configurations	
Cores	x86 out-of order cores, 1 GHz, 192-entry reorder buffer, issue width: 8
L1 I/D cache	32KB per core, private, 4-way set associative
L2 cache	1MB per core, shared, 8-way set associative
Directories	8 directories on the interposer
Cache coherence	MESI directory coherence protocol
Network configurations	
Topology	baseline system: 1 4x4 mesh interposer, 4 4x4 mesh chiplets
VCs	3 VNets, with 1 or 4 VCs per VNet, each VC has 4 flit-sized buffers
Router pipeline	3 stages
Link	latency: 1 cycle, width: 128 bits
Flow control	wormhole
Packet size	data packet: 5 flits, control packet: 1 flit
Synthetic simulation	uniform random, bit complement, bit rotation, transpose, with a mix of control and data packets
Full system simulation	PARSEC [6], SPLASH-2 [48]
UPP detection threshold	20 cycles

ports of the crossbar switch connection. UPP_ack signals look up this table to follow the reverse routing path of the corresponding UPP_req signals. Upward packets look up this table for circuit switching flow control. When an upward flit arrives, it directly enters switch traversal without allocation. The other flits and protocol signals are forced to stall even if they have been granted the same input port or output port. The switch allocator gives a higher priority to protocol signals than normal flits, by always giving grants to signals during allocation. Normal flit requests for the same ports as protocol signals are rejected by ANDing with 0. There is neglible complexity except for a few gates.

Modifications to a chiplet NI: The green components at the bottom of Fig. 6 are added in every chiplet NI. At the ejection queue side, the UPP_req and UPP_stop units process the received UPP_req and UPP_stop signals and interact with the table. The table with an entry for each VNet records the reservation conditions of the ejection queue entry. After successfully reserving an entry, the UPP_ack unit at the injection queue side generates an UPP_ack and injects it into the network.

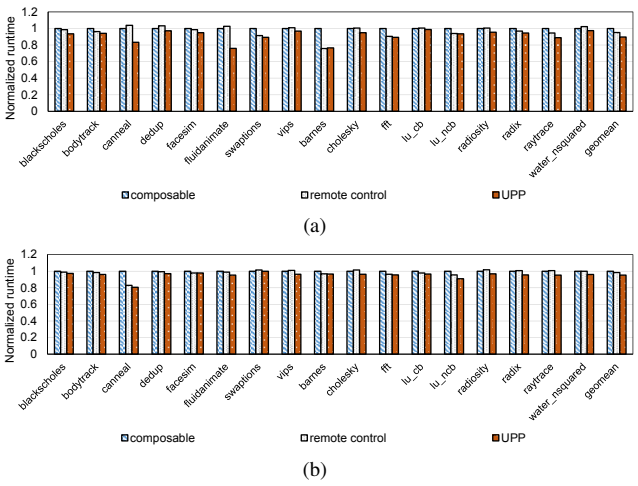
VI. EVALUATION

In this section, UPP is evaluated in gem5 simulator [7] with the Garnet NoC model [1]. The adopted topology is the baseline system shown in Fig. 1. To illustrate the generality of UPP, experiments upon larger and faulty systems are also conducted. UPP is compared with two other approaches that



(a) (b) (c) (d)

Figure 7. Latency comparison under four synthetic traffic patterns as the injection rate increases. Each VNet has 1 or 4 VCs.



(a) (b)

Figure 8. Normalized runtime comparison of full system simulations. (a) System with 1 VC per VNet. (b) System with 4 VCs per VNet.

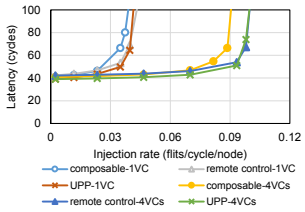


Figure 9. Latency comparison in a 128-node chiplet-based system.

are also proposed for modular chiplet-based systems, i.e., the composable routing [51] and remote control [24]. All three approaches use XY routing in both chiplets and the interposer for local deadlock freedom. The composable routing uses a different boundary router selection mechanism [51] that leads to non-minimal routing due to its unidirectional turn restrictions. Remote control uses the same boundary router selection mechanism as UPP. Each boundary router in remote control is equipped with four data packet-sized buffers. Other simulation configurations are given in Tab. II.

A. *Performance in the baseline system*

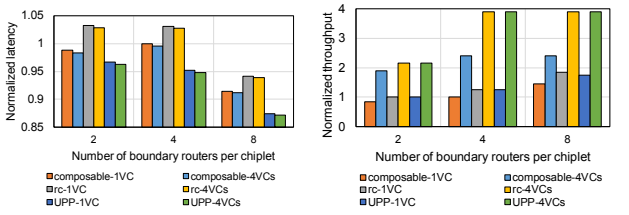
Fig. 7 and show the latency comparisons as the injection rate increases under four synthetic traffic patterns. The warmup time is 10K cycles and the simulation time is 100K cycles. Regardless of the imposed traffic pattern and the number of VCs, UPP always has a higher saturation throughput and a lower latency. Compared with the composable routing, UPP improves the saturation throughput by 18%∼72% as well as reduces the latency by 4.5%∼6.6%. The performance benefits can be attributed to three reasons. First, UPP does not impose turn restrictions on boundary routers and thus maximizes the inter-chiplet bandwidth utilization. Second, UPP does not force any packets to take a non-minimal route to reach the destination and thus has a lower latency. Third, the boundary router selection mechanism of UPP achieves a better load balance than the composable routing. In fact, the composable routing also claims to achieve load balance in boundary router selection [51]. Nonetheless, it brings about an unbalanced network due to the excessive turn restrictions as described in Sec. III-B. The saturation throughput of remote control is almost the same as that of UPP. This is because both approaches achieve full path diversity and load balance in the baseline system. However, remote control has a 5.7%∼8.2% longer latency than UPP due to its injection control. Although the permission network for injection control is hard-wired and optimized for a low round trip latency, there is still a latency penalty of minimally 2 cycles for each inter-chiplet packet before injection.

Fig. 8 shows the runtime results of gem5 full system simulations with PARSEC [6] and SPLASH-2 [48] benchmarks. The results are normalized to those of the composable routing. On average, UPP reduces the runtime by 5.7%∼10.3% in 1-VC system and 3.1%∼4.6% in 4-VCs system. Although remote control has the same high saturation throughput as UPP, it occasionally degrades the performance (e.g., canneal in 1-VC system) due to its longer latency.

B. *Generality Analysis*

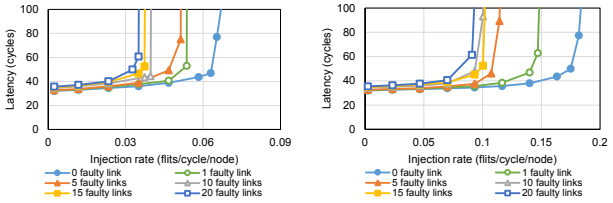
Sec. VI-A shows the experimental results in one specific chiplet-based system topology. In this section, more experiments are conducted to demonstrate the generality of UPP in other system topologies.

Fig. 9 shows the latency comparison in a larger system with a 4×8 interposer and 8 4×4 chiplets. Only the results under uniform random traffic is presented because the general trend is very similar among different traffic patterns. UPP still achieves a saturation throughput improvement of



(a) (b)

Figure 10. Sensitivity study of the number of boundary routers per chiplet. Results are normalized to that of the composable routing when there is 1 VC per VNet. (a) Normalized latency comparison. (b) Normalized throughput comparison.



(a) (b)

Figure 11. Latency comparison in irregular systems with variable numbers of faulty links. (a) Systems with 1 VC per VNet. (b) Systems with 4 VCs per VNet.

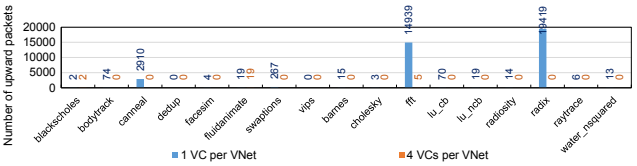
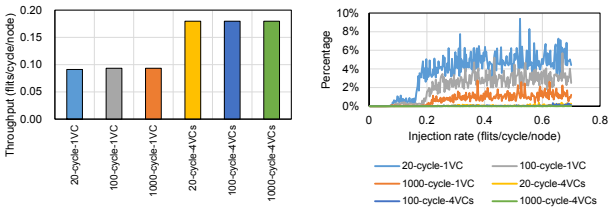


Figure 12. The number of detected upward packets during gem5 full system simulations.



(a) (b)

Figure 13. Sensitivity study of UPP detection threshold value. (a) The impact on saturation throughput. (b) The percentage of upward packets among all packets.

11%∼13% and a latency reduction of 4.4%∼7.4%. However, compared with the results in the baseline system of Fig. 7, the throughput gap is much smaller. The reason is that the network is inherently less load-balanced as the network scales up. Therefore, the throughput loss due to load imbalance in the composable routing is relatively less significant.

Fig. 10 is the sensitivity study of the number of boundary routers per chiplet. All three approaches have a higher throughput and a lower latency. But UPP always deliver a better performance than the others.

Fig. 11 shows the latency comparisons in faulty systems. The systems are devised by designating a certain number of links in the baseline system to be faulty [2], [18], [28], [38]. Several different faulty topologies are randomly generated and the average results are presented. The composable routing is not compared against, as its complex algorithm traverses all possible combinations of turn restrictions and takes an unacceptably long time in a faulty chiplet-based system. Remote control is also not compared against because its hard-wired permission subnetwork cannot adapt to a dynamically changing topology [3]. As the number of faulty links increases, the saturation throughput of UPP shows a graceful degradation and the latency slightly increases.

Although UPP is proposed for chiplet-based systems that use active interposers, UPP can also be applied to systems using passive substrates. For example, consider a star-like system where chiplets are integrated on a passive substrate [27], [51], a central chiplet serves the purpose of I/O and is connected with the other chiplets. From the perspective of the network, the system is same as the baseline system of Fig. 1 and the central chiplet can be treated equivalently as an active interposer [51]. Therefore, UPP still applies in this scenario.

C. Performance Impact of False Positives

As mentioned earlier, the performance impact of false positives lies in the bandwidth utilized for transmitting the protocol signals, which is proportional to the number of detected upward packets. Thus, the numbers of detected upward packets during full system simulations are recorded to demonstrate the performance impact due to false positives, as shown in Fig. 12. Across all the benchmarks, the number of upward packets never exceeds 20000, which is less than 0.01% of the total number of packets (that is in the order of 10^8). When the number of VCs per VNet increases from 1 to 4, the number of upward packets is significantly reduced. Therefore, the bandwidth waste of protocol signals and the false positives have little impact on system performance.

Fig. 13(a) demonstrates that the UPP detection threshold value has little impact on the saturation throughput. Fig. 13(b) gives the reasons. When there is 1 VC per VNet, most of the network bandwidth is left unused due to head-of-line blocking [21] or deadlocks. The percentage of upward packets is higher and false positives can be frequent. But UPP utlizes the wasted bandwidth for recovery and thus does not impact the throughput. When there are 4 VCs per VNet, the percentage of upward packets never exceeds 0.4%. Therefore, the false positives are too rare to affect the throughput.

[3]Even if remote control uses an idealized permission subnetwork that can adapt to any faulty systems, after conducting experiments, remote control still has the same saturation throughput as UPP due to the full path diversity but a longer latency due to the injection control.

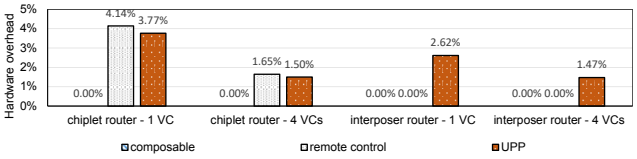


Figure 14. Hardware overhead comparison in chiplet and interposer routers. The system has either 1 or 4 VCs per VNet.

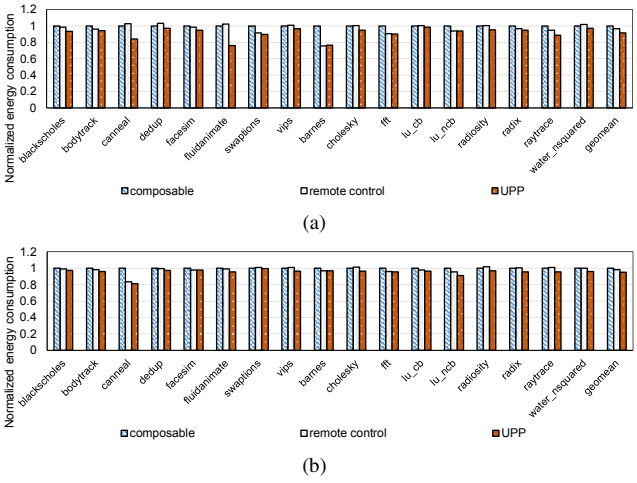


Figure 15. Normalized energy consumption comparison of full system simulations. (a) System with 1 VC per VNet. (b) System with 4 VCs per VNet.

D. Hardware Overhead and Energy Consumption

Fig. 14 shows the comparison of the hardware overhead in chiplet and interposer routers, where the network frequency is 1 GHz. Synthesizing using the Synopsys Design Compiler under 45nm TSMC library, the baseline router has an area of 135,083 μm^2 with 1 VC per VNet and 339,371 μm^2 with 4 VCs per VNet. The hardware overhead on chiplet routers and interposer routers are considered respectively; the area of NI is included in chiplet routers. As shown in Fig. 14, the composable routing consumes almost zero additional hardware; this is because it only uses turn restrictions for deadlock freedom. Although UPP incurs a slightly larger area overhead, it is worthwhile due to its benefits in performance and flexibility.

Fig. 15 shows the energy consumption comparisons of gem5 full system simulations, where the results are normalized to those of the composable routing. The energy is calculated by gathering the runtime statistics and using the DSENT tool [44] for estimation under 22 nm technology. As the average traffic load of real benchmarks is quite low, the network power consumption is dominated by static power. Therefore, the energy consumption results are quite similar to the runtime results of Fig. 8. It can be seen that UPP consumes the least energy on average due to its shorter runtime.

VII. Related Works

A. Routing Deadlock Freedom

Dally's theory-based approaches: Segment routing [23], [25] divides a network into subnetworks and further divides subnetworks into segments. Then it places one bidirectional turn restriction in each segment for deadlock freedom. ARIADNE [2], [33] traverses the whole network with breadth-first searches and places up*/down* [40] turn restrictions. uDIREC [32] extends ARIADNE to be applicable in faulty networks with unidirectional faulty links. These approaches traverse the network topology for turn restriction placement. Besides, their turn restrictions reduce path diversity and lead to non-minimal routing on irregular topologies [38]. VC partitioning [11], [14] separates different traffic flows with multiple VCs. EbDa [15] introduces three theorems that directly lead to fully-adaptive deadlock-free routing algorithms. These approaches require the topology and number of VCs for VC usage restriction decisions. The additional VCs increase the hardware overhead.

Duato's theory-based approaches: Router Parking [39] places up*/down* turn restrictions in an escape VC. Immunet [34] and Immucube [36] applies BFC in a ring escape subnetwork. These approaches require to add more VCs to construct the escape VC. Differently, DRAIN [28] constructs a unidirectional ring escape path in time domain. All packets are forced to move along it periodically for deadlock freedom. The construction requires the global topology information and violates the modularity. DISHA [3], [45] constructs the escape path with additional buffers at every router and Pitstop [18] utilizes the idle NI buffers. They guarantee deadlock freedom by allowing only one packet in the escape path at any time. Because their escape paths can cover the whole network, when DISHA or Pitstop are applied in chiplet-based systems, they demand that all chiplets have to support them. In contrast, when UPP are applied on a chiplet and its connected interposer routers, the other chiplets are free to use the composable routing or remote control to avoid deadlocks. Therefore, DISHA and Pitstop lack the compatibility with other approaches. They prevent different chiplets making individual optimizations and violate the modularity.

Deflection-based approaches: These approaches [16], [17], [26], [49], [50] are originally proposed to completely remove NoC buffers and reduce the associated hardware and power. BBR [31], BINDU [30] and SWAP [29] use controlled deflection [29] to resolve deadlocks in irregular topologies. These approaches achieve full path diversity regardless of the network topology, but they incur packet truncation if the wormhole flow control is used. Therefore, when they are applied in chiptlet-based systems, they are not compatible with chiplets that do not support packet truncation or reassembly, and violate the flow control modularity.

B. Protocol Deadlock Freedom

Most NoC designs use multiple VNets [19] for protocol deadlock freedom. mDISHA [42] extends DISHA [3] to resolve protocol deadlocks after detection. DRAIN [28] is the first one to achieve both routing and protocol deadlock freedom without adding buffers. However, the necessary condition of DRAIN that the packets of one message class cannot use up all network buffers is proved to be impractical [18]. Pitstop [18] transmits packets of different message classes concurrently along the escape path to achieve both routing and protocol deadlock freedom. By utilizing the idle NI buffers, no additional buffers is required. The proposed UPP is a routing deadlock recovery framework that allows individual chiplet and the interposer to use locally optimized routing algorithms. By applying Pitstop in both chiplets and the interposer, UPP can also ensure both routing and protocol deadlock freedom with 1 VNet.

VIII. Conclusion

The integration-induced deadlock is intractable in chiplet-based systems. To overcome the challenge of chiplet design modularity, performance and flexibility, UPP, a deadlock recovery framework for modular chiplet-based systems is proposed. Compared with state-of-the-art approaches, UPP improves the performance by up to 24% with a hardware overhead of less than 4%.

Acknowledgment

This work is supported by the Beijing Superstring Academy of Memory Technology.

References

[1] N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jha, "Garnet: A detailed on-chip network model inside a full-system simulator," in *2009 IEEE International Symposium on Performance Analysis of Systems and Software*, 2009, pp. 33–42.

[2] K. Aisopos, A. DeOrio, L.-S. Peh, and V. Bertacco, "Ariadne: Agnostic reconfiguration in a disconnected network environment," in *2011 International Conference on Parallel Architectures and Compilation Techniques*, 2011, pp. 298–309.

[3] K. Anjan and T. Pinkston, "An efficient, fully adaptive deadlock recovery scheme: Disha," in *Proceedings 22nd Annual International Symposium on Computer Architecture*, 1995, pp. 201–210.

[4] A. Arunkumar, E. Bolotin, B. Cho, U. Milic, E. Ebrahimi, O. Villa, A. Jaleel, C.-J. Wu, and D. Nellans, "Mcm-gpu: Multi-chip-module gpus for continued performance scalability," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 2017, p. 320–332.

[5] 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)*, 2020, pp. 1–6.

[6] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: Characterization and architectural implications," in *2008 International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2008, pp. 72–81.

[7] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *Association for Computing Machinery*, vol. 39, no. 2, p. 1–7, 2011.

[8] L. Chen and T. M. Pinkston, "Worm-bubble flow control," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, 2013, pp. 366–377.

[9] L. Chen, R. Wang, and T. M. Pinkston, "Critical bubble scheme: An efficient implementation of globally aware network flow control," in *2011 IEEE International Parallel & Distributed Processing Symposium*, 2011, pp. 592–603.

[10] J. Cong, M. Gill, Y. Hao, G. Reinman, and B. Yuan, "On-chip interconnection network for accelerator-rich architectures," in *DAC '15: Proceedings of the 52nd Annual Design Automation Conference*, 2015, pp. 1–6.

[11] Dally and Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computers*, vol. C-36, no. 5, pp. 547–553, 1987.

[12] W. Dally and B. Towles, *Principles and Practices of Interconnection Network*, 2004.

[13] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 12, pp. 1320–1331, 1993.

[14] J. Duato and T. M. Pinkston, "A general theory for deadlock-free adaptive routing using a mixed set of resources," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 12, p. 1219–1235, 2001.

[15] 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, p. 703–715.

[16] C. Fallin, C. Craik, and O. Mutlu, "Chipper: A low-complexity bufferless deflection router," in *2011 IEEE 17th International Symposium on High Performance Computer Architecture*, 2011, pp. 144–155.

[17] C. Fallin, G. Nazario, X. Yu, K. Chang, R. Ausavarungnirun, and O. Mutlu, "Minbd: Minimally-buffered deflection routing for energy-efficient interconnect," in *2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*, 2012, pp. 1–10.

[18] H. Farrokhbakht, H. Kao, K. Hasan, P. V. Gratz, T. Krishna, J. S. Miguel, and N. E. Jerger, "Pitstop: Enabling a virtual network free network-on-chip," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021, pp. 682–695.

[19] A. Hansson, K. Goossens, and A. Rădulescu, "Avoiding message-dependent deadlock in network-based systems on chip," *VLSI design*, 2007.

[20] N. D. E. Jerger, L.-S. Peh, and M. H. Lipasti, "Circuit-switched coherence," in *Second ACM/IEEE International Symposium on Networks-on-Chip (nocs 2008)*, 2008, pp. 193–202.

[21] N. E. Jerger, T. Krishna, L.-S. Peh, and M. Martonosi, *On-Chip Networks: Second Edition*. Morgan & Claypool, 2017.

[22] A. Kannan, N. E. Jerger, and G. H. Loh, "Enabling interposer-based disintegration of multi-core processors," in *2015 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2015, pp. 546–558.

[23] D. Lee, R. Parikh, and V. Bertacco, "Brisk and limited-impact noc routing reconfiguration," in *Proceedings of the Conference on Design, Automation & Test in Europe*, 2014.

[24] 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, 2021.

[25] A. Mejia, J. Flich, J. Duato, S.-A. Reinemo, and T. Skeie, "Segment-based routing: an efficient fault-tolerant routing algorithm for meshes and tori," in *Proceedings 20th IEEE International Parallel Distributed Processing Symposium*, 2006, pp. 10 pp.–.

[26] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networkp," in *Proceedings of the 36th Annual International Symposium on Computer Architecture*, 2009, p. 196–207.

[27] S. Naffziger, N. Beck, T. Burd, K. Lepak, G. H. Loh, M. Subramony, and S. White, "Pioneering chiplet technology and design for the amd epyc™ and ryzen™ processor families," in *Proceedings of the 48th Annual International Symposium on Computer Architecture*, 2021.

[28] M. Parasar, H. Farrokhbakht, N. Enright Jerger, P. V. Gratz, T. Krishna, and J. San Miguel, "Drain: Deadlock removal for arbitrary irregular networks," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2020, pp. 447–460.

[29] M. Parasar, N. E. Jerger, P. V. Gratz, J. S. Miguel, and T. Krishna, "Swap: Synchronized weaving of adjacent packets for network deadlock resolution," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019, p. 873–885.

[30] M. Parasar and T. Krishna, "Bindu: Deadlock-freedom with one bubble in the network," in *Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip*, 2019.

[31] M. Parasar, A. Sinha, and T. Krishna, "Brownian bubble router: Enabling deadlock freedom via guaranteed forward progress," in *2018 Twelfth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, 2018, pp. 1–8.

[32] R. Parikh and V. Bertacco, "Udirec: Unified diagnosis and reconfiguration for frugal bypass of noc faultsp," in *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, 2013, p. 148–159.

[33] R. Parikh, R. Das, and V. Bertacco, "Power-aware nocs through routing and topology reconfiguration," in *Proceedings of the 51st Annual Design Automation Conference*, 2014, p. 1–6.

[34] V. Puente, J. A. Gregorio, F. Vallejo, and R. Beivide, "Immunet: A cheap and robust fault-tolerant packet routing mechanism," in *Proceedings of the 31st Annual International Symposium on Computer Architecture*, 2004, p. 198.

[35] V. Puente, C. Izu, R. Beivide, J. Gregorio, F. Vallejo, and J. Prellezo, "The adaptive bubble router," *Journal of Parallel and Distributed Computin*, vol. 61, no. 9, pp. 1180–120, 2001.

[36] V. Puente and J. A. Gregorio, "Immucube: Scalable fault-tolerant routing for k-ary n-cube networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 6, pp. 776–788, 2007.

[37] A. Ramrakhyani, P. V. Gratz, and T. Krishna, "Synchronized progress in interconnection networks (spin): A new theory for deadlock freedom," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, 2018, pp. 699–711.

[38] A. Ramrakhyani and T. Krishna, "Static bubble: A framework for deadlock-free irregular on-chip topologies," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2017, pp. 253–264.

[39] A. Samih, R. Wang, A. Krishna, C. Maciocco, C. Tai, and Y. Solihin, "Energy-efficient interconnect via router parkingp," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, 2013, pp. 508–519.

[40] M. Schroeder, A. Birrell, M. Burrows, H. Murray, R. Needham, T. Rodeheffer, E. Satterthwaite, and C. Thacker, "Autonet: a high-speed, self-configuring local area network using point-to-point links," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 8, pp. 1318–1335, 1991.

[41] Y. S. Shao, J. Clemons, R. Venkatesan, B. Zimmer, M. Fojtik, N. Jiang, B. Keller, A. Klinefelter, N. Pinckney, P. Raina, S. G. Tell, Y. Zhang, W. J. Dally, J. Emer, C. T. Gray, B. Khailany, and S. W. Keckler, "Simba: Scaling deep-learning inference with multi-chip-module-based architecture," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019, p. 14–27.

[42] Y. H. Song and T. Pinkston, "A progressive approach to handling message-dependent deadlock in parallel computer systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, no. 3, pp. 259–275, 2003.

[43] D. Stow, Y. Xie, T. Siddiqua, and G. H. Loh, "Cost-effective design of scalable high-performance systems using active and passive interposers," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2017, pp. 728–735.

[44] C. Sun, C.-H. O. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L.-S. Peh, and V. Stojanovic, "Dsent - a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in *2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*, 2012, pp. 201–210.

[45] A. K. Venkatramani, T. M. Pinkston, and J. Duato, "Generalized theory for deadlock-free adaptive wormhole routing and its application to disha concurrent," in *Proceedings of the 10th International Parallel Processing Symposium*, 1996, p. 815–821.

[46] T. Vijayaraghavan, Y. Eckert, G. H. Loh, M. J. Schulte, M. Ignatowski, B. M. Beckmann, W. C. Brantley, J. L. Greathouse, W. Huang, A. Karunanithi, O. Kayiran, M. Meswani, I. Paul, M. Poremba, S. Raasch, S. K. Reinhardt, G. Sadowski, and V. Sridharan, "Design and analysis of an apu for exascale computing," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2017, pp. 85–96.

[47] R. Wang, L. Chen, and T. M. Pinkston, "Bubble coloring: Avoiding routing- and protocol-induced deadlocks with minimal virtual channel requirement," in *Proceedings of the 27th International ACM Conference on International Conference on Supercomputing*, 2013, p. 193–202.

[48] S. Woo, M. Ohara, E. Torrie, J. Singh, and A. Gupta, "The splash-2 programs: characterization and methodological consideration," in *Proceedings 22nd Annual International Symposium on Computer Architecture*, 1995, pp. 24–36.

[49] Y. Wu, L. Liu, L. Wang, X. Wang, J. Han, C. Deng, and S. Wei, "Aggressive fine-grained power gating of noc buffers," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3177–3189, 2020.

[50] Y. Wu, L. Wang, X. Wang, J. Han, S. Yin, S. Wei, and L. Liu, "A deflection-based deadlock recovery framework to achieve high throughput for faulty nocs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2020.

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