iSWAP - Interposer SWAP

Nesara Eranna Bethur
ECE department
Georgia Institute Of Technology
Atlanta, Georgia, USA

Atrey Hosmane
ECE department
Georgia Institute Of Technology
Atlanta, Georgia, USA

ABSTRACT

The exploding cost and complexity of monolithic SoCs have motivated the industry to move towards interposer based chiplet solutions which inherently give better yield and bandwidth. Deadlocks, which are cyclic resource dependencies, tend to occur when heterogeneous systems like chiplets are integrated. Even though the chiplets and the interposer are inherently deadlock- free, the resulting irregular topology might have deadlocks. Prior approaches have tried to solve this problem either using system-level knowledge of the SoC compromising on modularity or by incurring higher costs by using several Virtual Channels (VCs) or imposing some turn restrictions. In this paper, a general solution has been proposed to avoid deadlocks induced by the chiplet integration at the interposer level, independent of chiplets' NOC architecture. This has been achieved by taking advantage of the control over the interposer routing and flow control by implementing a sub-active deadlock resolution mechanism called SWAP (Synchronous Weaving of Adjacent Packets). Since SWAP is topology agnostic and does not require extra VCs, it suits well for the interposer design. Hence, we propose Interposer-SWAP (iSWAP), where-in SWAP is used only in the interposer routers, ensuring deadlock resolution with performance enhancement, as no turn-restrictions are imposed. We successfully emulated a chiplet based topology and achieved deadlock freedom, with slight improvement in performance, with Synthetic traffic pattern.

1 INTRODUCTION

Increase in System On Chip complexity has motivated architects to break the system into smaller designs called "Chiplets" and integrate them using a base structure called "Interposer". In our paper, we emulate a chiplet based topology and try to solve the problem of deadlocks using an interesting sub-active technique called (Interposer SWAP) iSWAP.

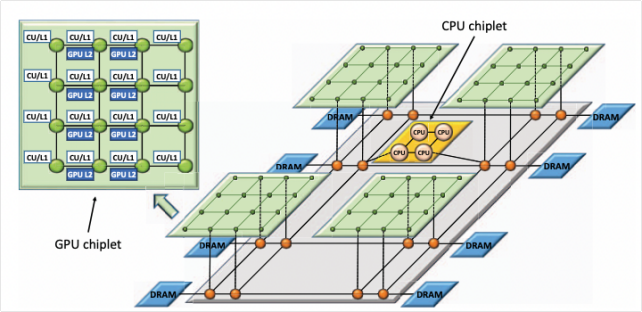

Figure 1: Chiplet based SoC

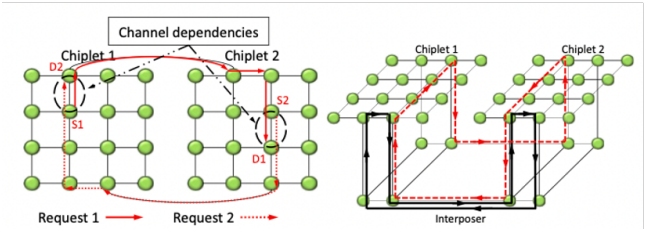

Figure 2: Deadlocks due to integration in Chiplet based Systems

Prior approaches have tried to solve this problem either using system-level knowledge of the SoC or by incurring higher costs by using several VCs. A few approaches also use local turn restrictions which curb performance. As we saw here, the metrics which make an approach suitable for a chiplet based system should be modular, involve lower costs and should not unnecessarily curb performance. Modularity is defined as the ability to design individual chiplets without having any knowledge of the complete SoC. There are multiple aspects of modularity which will be explained in the background section. However, a recent approach has also tried to solve this problem by interestingly placing boundary routers and turn restrictions on the inter chiplet traffic only. Development on top of that has tried a reactive approach to solve this problem with deadlock detection and direction packet forwarding. Since we felt that the two approaches have reduced performance and induced some modularity problems, we made a thorough case study of sub-active and reactive techniques to resolve deadlocks. In the background section, one can see a thorough evaluation of such techniques. Upon thorough brainstorming, we have decided to apply a sub-active deflection-based routing technique called SWAP in the interposer only reducing the complexity of chiplet NoC architecture to ensure functionality and increase performance without violating modularity.

2 BACKGROUND STUDY

Design modularity is one of the important aspects of design consideration when choosing a suitable deadlock mechanism for Chiplet based systems. Modularity guarantees that each chiplet can work independently and optimized in its own design space. Having modularity in these design attributes is the focus of this project – Topology, VC configurations, full path diversity, and network flexibility. Based on that We explored existing Reactive/ Sub active methods of avoiding deadlocks suitable for Modular Chiplet based systems.

Deadlock Freedom Mechanisms	Topology	VC	Flow Control	Router Micro-architecture	Full path Diversity	w/o Injection Control	Topology Independence
CDG/Dally's Theory	✗	✓	✓	✓	✗	✓	✗
Duato's Theory	✗	✗	✓	✓	✗	✓	✗
BFC-based	✓	✓	✗	✗	✓	✓	✓
Deflection based	✓	✓	✗	✗	✓	✓	✓
SPIN-based	✓	✓	✗	✗	✓	✓	✓
Composable Routing	✓	✓	✓	✓	✗	✓	✗
Remote Control	✓	✓	✓	✓	✗	✗	✗
UPP(Upward packet propagation)	✓	✓	✓	✗	✓	✓	✓
iSWAP(this work)	✓	✓	✓	✓	✓	✓	✓

Table 1: Qualitative Comparison of Existing Deadlock Freedom Mechanisms

2.1 Composable

In this paper, the authors propose an interesting modular way of handling deadlocks without much overhead. They intelligently control the traffic by imposing turn restrictions on the inter-chiplet traffic and by placing the boundary routers in appropriate locations. Even though this technique follows modularity at all levels, the imposition of turn restrictions significantly reduces the performance.[1]

2.2 UPP (Upward Packet Propagation)

UPP is a reactive deadlock-free mechanism, where it resolves deadlock by forcing the upward packet to eject at the destination router after detection. This idea is based on an observation – integration-induced deadlock involves at least one upward packet. This mechanism involves microarchitectural changes in both Chiplet's routers and Interposer routers to support backpressure handling while ejecting upward packets through the Network Interface controller. Hence it violates the router modularity across chiplet systems.

2.3 SPIN

SPIN views deadlock as a lack of coordination. It resolves deadlock by orchestrating the movement of packets along the deadlock chain. Synchronized packet movement in SPIN requires Virtual cut-through flow control in all chiplet systems– violates Flow control mechanism modularity.[2]

2.4 DRAIN

DRAIN removes deadlock by draining resources periodically and obliviously. It constructs a unidirectional ring escape path in the time domain. All packets are forced to move along it periodically for deadlock freedom. The construction requires global topology information and violates modularity.[3]

2.5 BINDU and SWAP

BINDU moves at least one free VC throughout network paths in periodic time to ensure deadlock removal in cyclic dependent paths.[4] SWAP involves interchanging two packets from two adjacent routers to ensure deadlock removal.[5] These approaches achieve full path diversity regardless of the network topology, but they incur packet truncation if the wormhole flow control is used.

Overall Qualitative comparison of Existing Deadlock Freedom Mechanism is tabulated in 1 including iSWAP(this work).

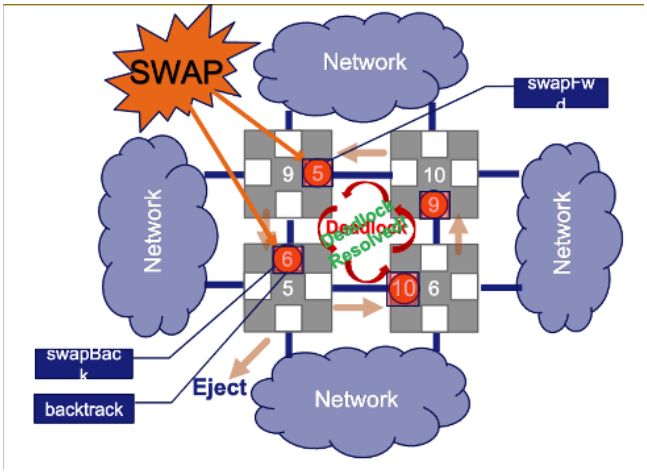

Figure 3: SWAP action in regular Mesh Topology

3 PROPOSED SOLUTION - ISWAP

In Multi chiplet systems, since interposer will have at least one path between two chiplet communicating, we can implement SWAP only in Interposer and let chiplets have the freedom of their local NOC design. Implementing one deadlock-free mechanism in Interposer, irrespective of other Chiplets NOC architecture we can achieve a deadlock-free mechanism, only modifying Interposer NOC microarchitecture. Thus, our work is not only achieving modularity in all aspects but also fulfilling other qualities such as Full path diversity, Topology and Routing agnosticism etc.

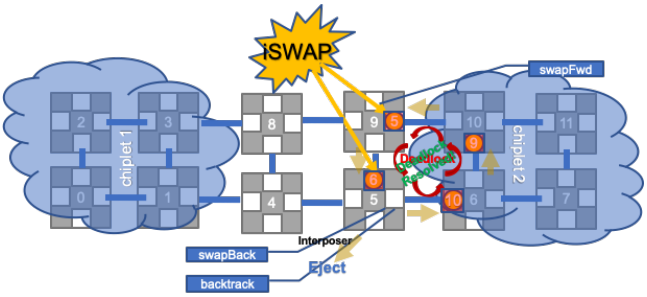

Figure 4: iSWAP in Action in Multi Chiplet based system

Router ID	Chiplet No.
0 to 15	1
16 to 31	2
32 to 47	3
48 to 63	Interposer

Table 2: Router division for Multi Chiplet System in Irregular Mesh

4 IMPLEMENTATION

For our implementation, we have used the on-chip network simulator called "Garnet2.0"[7]. Since we have a restriction of a number of routers as powers of 2 in Garnet2.0, we have used an inherent mesh topology from garnet and have modified the links to model the heterogeneity. Fig4. shows the topology used. Here, by choosing the edge routers as boundary routers, we are avoiding the usage of higher radix routers to support the heterogeneity. Currently, we have completed topology creation. We are demonstrating the routing by creating a function in the routing unit file of garnet2.0. All chiplets have been modeled to have XY routing, whereas the interposer can have adaptive routing as it will be enabled with ISWAP. By doing so, we are hoping to see a performance boost. Currently, we are debugging some issues with the routing. As far as traffic generation is considered, we are defining uniform random traffic within chiplet systems (Intra Chiplet Traffic) just to make sure within chiplet systems we don't have deadlocks and while generating Inter chiplet traffic we are generating the traffic in all routers excluding interposer routers. Currently debugging some issues with traffic generation.

We divided the implementation task into 3 sub tasks. The first task is to create heterogeneous topology and routing mechanism to resemble Chiplet Based Systems. Second task - understanding the basic principle of SWAP (Synchronized Weaving of Adjacent Packets) and go through implementation details in garnet 2.0 using Mayank's old data base, merge those with latest garnet 2.0 and test deadlock resolution using SWAP enable switch. Third and final task is to integrate both and implement SWAP only for interposer routers. Induce deadlock through intermesh traffic and resolve it using SWAP implemented in interposer. Details of these are stated in the following sections.

4.1 Topology and Routing

4.1.1 ***Topology.*** To demonstrate our proposed theory of deadlock resolution induced by integration of multi chiplets, we have created our own heterogeneous mesh topology as shown in **??**.

Steps followed while implementing topology –

(1) Create a new topology file – heterogeneous mesh - where four 4x4 meshes are connected in a heterogeneous manner to resemble 2.5D structure.
(2) Router IDs for each chiplet and interposer are assigned as shown in table - 3
(3) To establish connection within Meshes(chiplets), boundary routers are assigned, and those boundary routers are connected through interposer mesh.

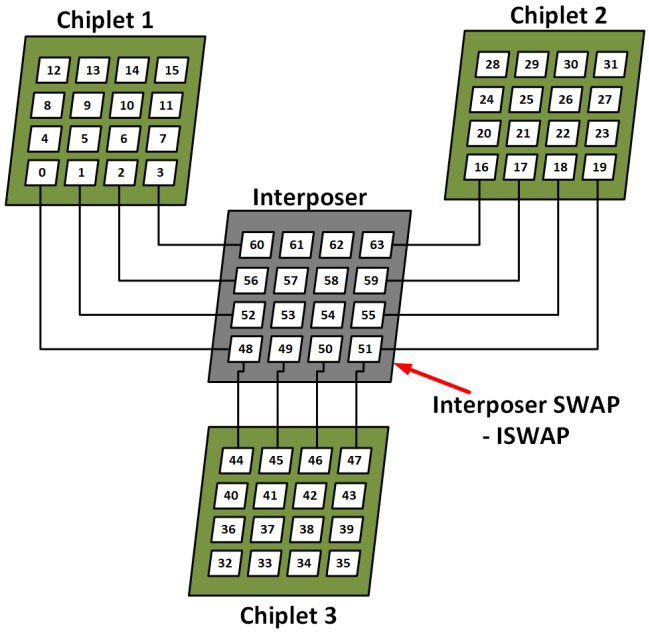

Figure 5: Customized Heterogeneous Mesh Topology to resemble the Multi Chiplet system

(4) Boundary routers are shown in the newly proposed topology diagram.

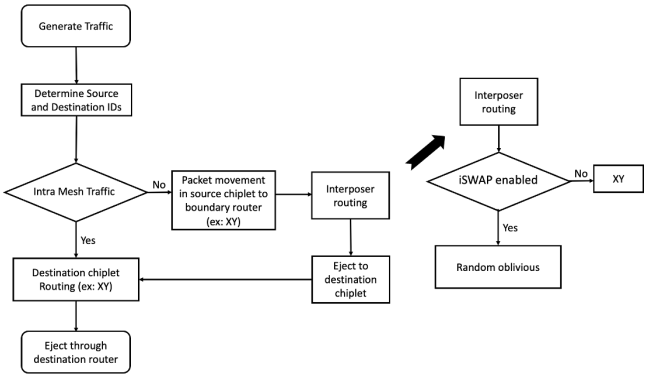

Figure 6: Overview of Customized Heterogeneous Routing Algorithm

4.1.2 ***Routing algorithm.*** In order to simulate multi-chiplet systems with an active interposer, a custom routing algorithm is developed. As mentioned in the topology section, each individual mesh represents a chiplet and each chiplet has its own routing algorithm which handles deadlock within the chiplet. Usually, chiplet systems have two types of traffic – Intra-Chiplet Traffic and Inter-Chiplet Traffic. To implement the same behavior, the following protocol is followed – If the source and destination nodes are within a sub-mesh, XY routing is implemented within that mesh. If the source and destination routers are across the chiplets, then these 3 steps

were followed: source chiplet routing, interposer routing, and destination routing. In the source chiplet routing, the packets are routed to the boundary router of the sub mesh containing the source node. Then, the packets are injected into interposer boundary routers. Here in the interposer, the boundary router is determined within the interposer using global destination router ID. Again, in the destination chiplet routing, it reaches the destination node using X-Y routing. There is a scope to have different routing algorithms and the number of VCs for each chiplet in this scheme. With this routing algorithm, 2 routing schemes are supported: Composable routing and iSWAP routing.

(1) **Composable Routing**

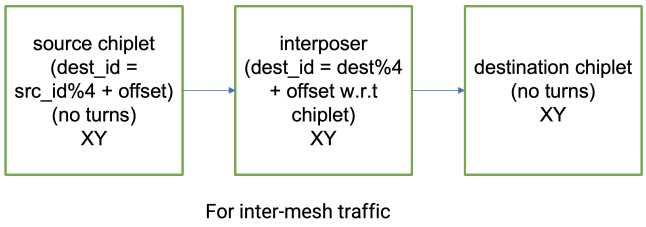

Figure 7: Pseudo code representation for Composable Routing Algorithm - here in order to determine the boundary routers source ID is used

In composable routing, turns are prohibited in source and destination chiplets during inter-chiplet traffic. By doing so we are avoiding the cyclic dependence which would have resulted in a deadlock. The interposer will also have XY routing, like the rest of the chiplets, to avoid inter-mesh deadlocks. In the source chiplet, the local destination is chosen by taking the source id modulo 4 with some additional offset. This makes sure there are no turns in the source chiplets. Now, in the destination chiplets, the interposer makes sure that the packet reaches the boundary router in the same column ensuring no turns. Whereas the interposer will obviously have turns, which are catered by XY routing.
For example, packet movement from node 9 to node 27 is assumed. Here, the local destination in the source chiplet is chosen to be node 1. So, the packet moves to node 1 from where it will reach the interposer at 52. The interposer through the XY routing logic routes the packet to node 51. After reaching the destination chiplet at boundary node 19 through interposer boundary node 51, the packet is further routed to 27. As seen through the entire packet movement, one can see that there are no turns involved in the source and destination chiplets during inter chiplet traffic making the routing deadlock free.

(2) **iSWAP Routing**
In iSWAP routing, there are no restrictions on the turns taken in the chiplets. In our case, in the source chiplet, the packet moves based on the destination id modulo 4 with an additional offset. This may incur a turn within chiplets. From here, the packet gets routed in the interposer using

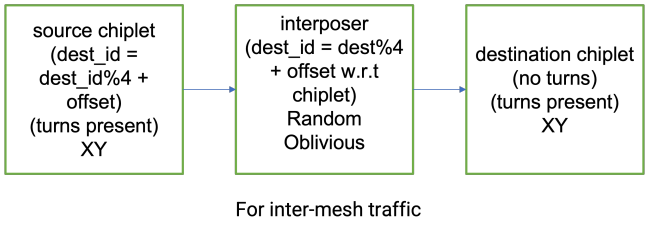

Figure 8: Pseudo code representation for iSWAP Routing Algorithm - here in order to determine the boundary routers destination ID is used. This will enable some turns within boundary routers

random adaptive routing. The packet after entering the destination chiplet further may have turns due to the adaptive nature of the interposer routing. The main reason for including random adaptive routing in the interposer is to boost performance. Here one can see the packets taking turns which would result in cyclic dependence leading to a deadlock. Since iSWAP is used over this, the deadlock would be resolved.
For example, packet movement from 9 to 27 is assumed. Here, the local destination in source chiplet is chosen to be node 3. So, the packet moves to node 3 from where it will reach the interposer at 60. In the process as seen, the packet takes a turn. The interposer through the XY routing logic routes the packet to a random boundary. After reaching the destination chiplet at the boundary node through the interposer boundary node, the packet is further routed to 27. Here, there may be turns involved, which would be further resolved by iSWAP.

4.2 SWAP

As mentioned earlier in the proposal section, SWAP[5] fits well to handle deadlock induced by Chiplet integration to the interposer. The basic principle of SWAP is to enable a blocked packet to perform an in-place swap with a buffered packet at the next hop as shown in 3. This method is agnostic to the underlying topology, which makes it even more suitable for interposer design where deadlock should be handled for intra- and inter-chiplet traffic.

4.2.1 *Definitions.*

Deadlock: Deadlock happens when packets remain inside the network indefinitely without reaching their destination. This occurs due to cyclic dependency between the buffers which results in a lack of resources leading packets to wait forever.

Forward Progress: Packet is moving towards its destination is termed as forward progress.

Backtrack: It's a mechanism where a packet yields its position and moves back to an upstream router during swap.

Swap: An act of interchanging two packets from two adjacent routers.

swapFwd packet: During swap, the upstream router chooses and routes the swapFwd packet towards a downstream router.

swapBack packet: The packet which switches its position with swapFwd packet and backtracked to upstream router during swapCycle.

Swap Period: Number of cycles required for the same router in the network to try and initiate swap for one of its buffered packets.

4.2.2 *Theory*.

The basic theory behind the SWAP mechanism is that it removes one edge from the runtime CDG and adds one new edge to the same. The edge which is removed is the original direction the swapBack packet was intended to go towards, while the edge that is added is the new direction swapFwd packet is intended to go towards. One of the intermediate routers will either be its destination, or an exit point out of the ring. Hence, the deadlock will be avoided without the use of extra VC's. As long as the system is allowed to perform SWAP, the network is deadlock free. To avoid livelocks, SWAP implementation allows the backtracked packet to move forward by two hops before being backtracked again.

4.2.3 *SWAP Implementation details*.

Deadlock recovery can be triggered after the detection; a controlled SWAP action will periodically resolve any network deadlock without much hardware overhead easily. This method of deadlock recovery is called Sub-Active mechanism. To initiate the SWAP at each router, we need a swappointer in each router. It cycles through each VC to decide which VC will undergo SWAP operation. To ensure that the system allows a packet to move two hops in an non-congested scenario without being backtracked, we need to account for the worst-case delay for a packet in two adjacent routers without any stall due to insufficient credits. This will be a packet in a VC competing with all other VCs at that router for a specific output port. This would be a packet in a VC contending with all other VCs at that router for a specific output port, followed by traversing the router and the link, and repeating the same at the next router. Thus,

$$swapPeriod \geq 2 \text{ X } (\#ports \text{ X } \#vcs/port + (router\_pipeline\_delay + link\_delay)) + serialization\_delay$$

In this implementation, each router performs a swap during its swapCycle which is defined as –
$(cycle/m)\%(K \times N) == router\_id$
Where K is a configurable swapDutyCycle, N is number of routers in the network, m is the maximum number of flits of any packet in the system and K X N is the swapPeriod. During the SwapCycle, the upstream router selects a swapFwd packet from one of its internally buffered packets. It sends a swap request signal via 1-bit wire to the downstream router at the output port of this packet. The downstream router selects a swapBack packet and sends ACK. After receiving ACK, swap is performed over m cycles with both the routers sending their packets out at the same time using the bidirectional link between them. These links are reserved for swap by ACK and are not allocated to any other packets by the switch allocators at the two routers. Since deadlocks are rare, swap system allows only one packet exchange in the system at any time. This reduces the number of backtrackings, race conditions and complexity of the design. Using a valid swapPointer, the upstream router picks a swapFwd packet. If the swapFwd packet exists, the router initiates a swap by sending a swap_req to the downstream router. This carries the VC ID and if it is ACK'd, the swap operation is set up in the next cycle. Similarly, swapBack packet will be selected by the downstream router using VC ID and it sends back the ack. A swapBack packet effectively makes a U-Turn and requests to go to the same router it was swapped from unless the router finds an alternative minimal path. After the swap, it will move again either via switch allocation or a swap.

(1) **Conditions for Updating Swappointer Valid -**
   (a) When the packet pointed by swapPointer leaves the router naturally by winning switch arbitration, the swapPointer moves in round-robin fashion to the next non-empty VC.
   (b) When a swapFwd packet arrives at this router from an upstream router via a swap. This packet now becomes the swapFwd packet to give it the highest priority at the next swapCycle in case it does not leave naturally

(2) **Conditions for Failed Swaps -**
   (a) At least one of the VCs within the virtual network of the swap_req is empty. In this case, the packet could arrive by normal means, and a swap is not required
   (b) In virtual cut-through routers, if the candidate swapFwd and swapBack packet is distributed across two routers, a swap is not performed. In wormhole, this condition leads to packet truncation.

Figure 9 shows the microarchitecture of the SWAP router.We show a mesh router for simplicity, though the same idea works for a router with any number of ports. Datapath: We assume bidirectional links. A swap operation requires both a forward path and a backward path to be setup between the upstream and downstream routers to swap the swapFwd and swapBack packets between their respective VCs. The additions to the conventional router are quite minimal: one 2:1 mux and one 2:1 demux in front of every input port, u-turn support in the crossbar, and a bus connecting all input ports.

4.2.4 *iSWAP implementation*.

SWAP garnet flow has been shown in 10. It is borrowed from SWAP database[8] and merged with the latest Garnet 2.0. After the initial setup, the deadlock has been induced using regular oblivious random routing. With SWAP Enable switch, with same topology and routing algorithm, deadlock has been resolved. We integrated heterogeneous mesh and custom routing algorithms with the SWAP database. The SWAP mechanism has been applied only to interposer routers by controlling the swap pointer valid variable.

5 EVALUATION

5.1 Methodology

We use gem5[6] to model the networks with different configurations, then we use Garnet. The table provides detailed configuration parameters. Our baseline includes a Composable Routing deadlock

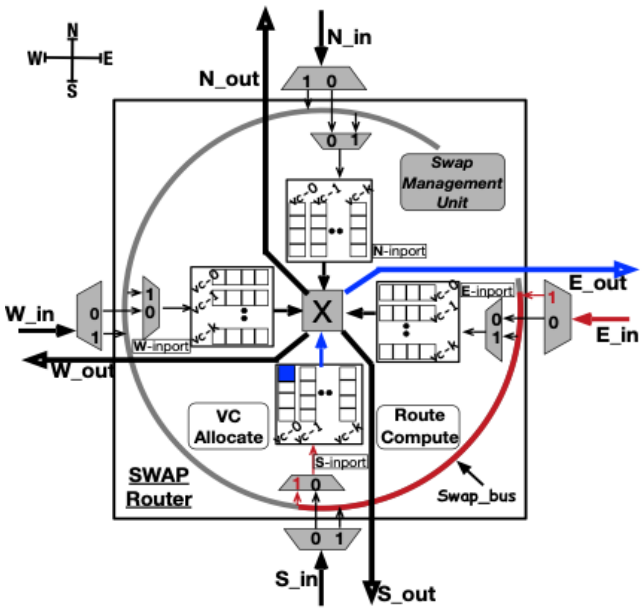

Figure 9: SWAP Router Micro architecture. Features added by SWAP are shaded in grey. Datapath: bus connecting all input ports to allow a swapBack packet from the downstream router to get buffered at any input VC, and u-turn support in the crossbar. Control path: Swap Management Unit controlling when and what to swap. The blue and red paths show a swapFwd packet going from South in port to East out port, and a corresponding swapBack packet entering from East out port and getting buffered in the South in port[5]

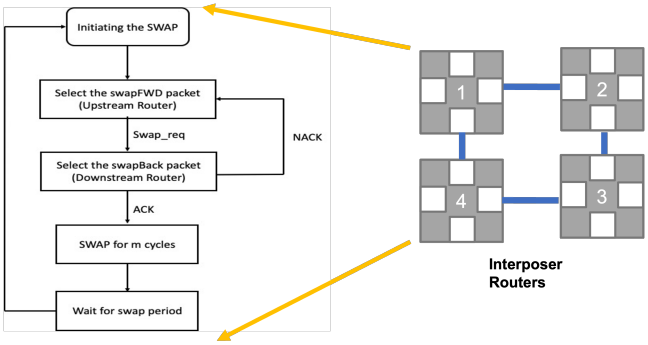

Figure 10: Detailed SWAP implementation Flow in Interposer Routers - By controlling swapPointer valid SWAP has been applied only to Interposer routers

avoidance mechanism for modular systems. Our irregular mesh topology resembles Modular chiplet based systems.

Network Configuration	
Topology	Heterogeneous Mesh
Routing	Custom
Num VCs	4
Link	128 bit
Flow Control	Virtual Cut Through
Packet Size	1 flit per packet
Deadlock Avoidance	iSWAP, Composable Routing
Synthetic Traffic	Uniform Random.

Table 3: Network Configuration

5.2 Correctness

We start by demonstrating why a deadlock-freedom solution is imperative in any network. By disabling SWAP through command-line arguments and allowing all turns, we induced a deadlock in the system. Garnet has an inbuilt deadlock check, where it uses some threshold value to abort the command if packets are not moving within the routers. The occurrence of deadlock causes the percentage of delivered packets to drop sharply; and the onset of deadlock depends upon the traffic pattern, the injection rate, and the number of VCs. By keeping the same configuration and enabling SWAP, the deadlock has been resolved. We tested the same using a limited number of packets, just to ensure we received all packets injected by the simulator. Thus, we successfully induced the deadlock, and using SWAP at the interposer, we were able to resolve the same.

5.3 Performance

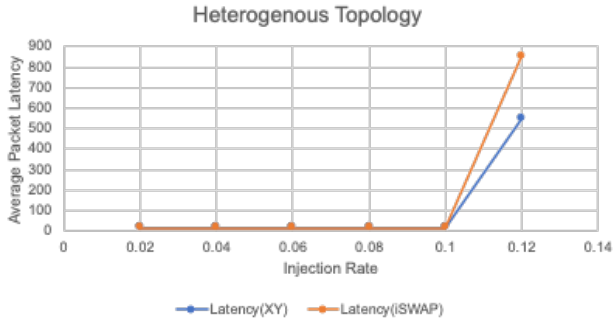

Figure 11: Performance of iSWAP vs Composable routing in Customized Heterogeneous Mesh - using Uniform Random as synthetic traffic

Although our focus was to achieve functional correctness, to evaluate performance, we swept through different injection rates and plotted corresponding average packet latency as shown in the graph. Compared to Composable routing with turn restrictions – iSWAP is able to achieve almost equal performance with random oblivious routing in the interposer. This implies that with fully adaptive routing in the interposer we can achieve better performance with iSWAP. Fully adaptive routing algorithm implementation is ongoing, which will be a part of our future work for the project.

6 CONCLUSION AND FUTURE WORK:

We present iSWAP, an extension of a novel technique called SWAP that enables in-packet swaps across neighboring routers applied to interposer routers. iSWAP guarantees deadlock freedom not only for intra-traffic within interposer but also for inter chiplet traffic. With a single method, we can achieve deadlock freedom within and resolved integration-induced deadlocks at interposer. iSWAP guarantees deadlock freedom by design as it can dynamically break any buffer dependence cycles, within and across interposers. It also enables the network to take full advantage of the available path diversity without any turn restrictions, injection restrictions, or escape VCs to avoid deadlocks. Since it is a sub active method, there is no requirement for deadlock detection and recovery. iSWAP is a powerful idea that goes beyond just deadlock resolution as it supports modularity. iSWAP doesn't need any additional change in Chiplets NoC architecture to resolve the integration-induced deadlocks. As of now, we successfully emulated a chiplet-based system using heterogeneous mesh topology and showed that deadlock can be induced and resolved using SWAP. In the near future, we will focus on improving performance either by implementing fully adaptive routing in Interposer with SWAP support or by improvising SWAP flow. Hetero garnet has some inbuilt features which more accurately resemble chiplet-based systems. So, we are planning to implement our design in Garnet 3.0 to emulate the systems with more accuracy. In order to perform power estimation, we are planning to write an RTL design of a basic router with SWAP to get a more realistic idea of power usage and compare it with the composable routing technique.

7 ACKNOWLEDGEMENT

We thank Mayank Parasar, a former PhD student under Professor Tushar Krishna for guiding us throughout the course of the project. Also, special mention to Professor Tushar Krishna who gave us the opportunity to work on this research problem as a part of the Interconnect Network Course.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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," SIGARCH Comput. Archit. News, Aug. 2011.

[7] N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jha, "Garnet: A detailed on-chip network model inside a full-system simula- tor," in 2009 IEEE International Symposium on Performance Analysis of Systems and Software, 2009.

[8] https://github.com/noc-deadlock/swap